전자 주식 매매 시스템에서의 보안 트랜잭션 관리를 위한 단일 스냅샷 알고리즘

(One-Snapshot Algorithm for Secure Transaction Management in Electronic Stock Trading Systems)

김 남 규 † 문 송 천 †† 손 용 락 †††
(Namgyu Kim) (Songchun Moon) (Yonglak Sohn)

요 약 최근 전자 상거래 인프라의 발전으로 인해, 온라인 상에서 주식의 매매가 이루어지는 전자 주식 매매 시스템(Electronic Stock Trading Systems: 약칭 *ESTS*)의 사용이 확산되고 있다. ESTS 상에서는, 다양한 기밀 등급을 가진 정보가 서로 다른 신뢰 등급을 갖는 사용자에 의해 공유된다. 특정 정보가 허가된 사용자에 의해서만 접근되도록 보장하기 위해서는, 트랜잭션의 동시성 제어 과정에서의 다등급 보안 데이타베이스 시스템의 사용이 반드시 필요하다. 한편 ESTS 상에서는 분석적인 성향의 트랜잭션과, 매매 체결을 목적으로 하는 실시간 트랜잭션이 동시에 수행되므로, 기존에 고안된 여러 보안 동시성 제어 기법들이 적용되는 데 어려움이 있다. 본 논문에서는 ESTS 환경에서의 보안 동시성 제어를 위한 프로토콜인 보안 단일 스냅샷(Secure One Snapshot: 약칭 *SOS*) 프로토콜을 제안한다. SOS는 운용 데이타베이스 외에 하나의 스냅샷을 추가로 유지하여 비밀 경로의 생성 가능성을 차단함과 동시에 실시간 동시성 제어 알고리즘이 용이하게 적용될 수 있는 유연성을 제공한다. 또한 SOS는 완화된 정확성 기준을 사용함으로써 데이타의 신선도를 유지하기 위해 관리되는 큐의 길이를 감소시킬 수 있는 방법도 제시한다. 본 논문에서는, SOS의 동작 과정을 예를 통해 소개하고, 프로토콜의 정확성에 대한 분석을 제공한다.

키워드 : 주식 매매 시스템, 비밀 경로, 판독 전용 트랜잭션, 다등급 보안 데이타베이스 시스템, 데이타베이스 보안, 뷰 일관성

Abstract Recent development of electronic commerce enables the use of *Electronic Stock Trading Systems*(ESTS) to be expanded. In ESTS, information with various sensitivity levels is shared by multiple users with mutually different clearance levels. Therefore, it is necessary to use *Multilevel Secure Database Management Systems*(MLS/DBMSs) in controlling concurrent execution among multiple transactions. In ESTS, not only analytical OLAP transactions, but also mission critical OLTP transactions are executed concurrently, which causes it difficult to adapt traditional secure transaction management schemes to ESTS environments. In this paper, we propose *Secure One Snapshot*(SOS) protocol that is devised for *Secure Transaction Management* in ESTS. By maintaining additional one snapshot as well as working database, SOS blocks covert-channel efficiently, enables various real-time transaction management schemes to be adapted with ease, and reduces the length of waiting queue being managed to maintain freshness of data by utilizing the characteristics of less strict correctness criteria. In this paper, we introduce the process of SOS protocol with some examples, and then analyze correctness of devised protocol.

Key words : Stock Trading Systems, Covert Channel, Read-Only Transaction, Multilevel Secure Database Systems, Database Security, View Consistency

† 학생회원 : 한국과학기술원 경영공학과
ngkim@mail.kaist.ac.kr
†† 종신회원 : 한국과학기술원 경영공학과 교수
scmoon@kgsm.kaist.ac.kr
††† 종신회원 : 서경대학교 컴퓨터공학과 교수
syl@seokyeong.ac.kr
논문접수 : 2002년 8월 14일
심사완료 : 2002년 12월 28일

1. 서 론

1.1 배 경

전자상거래(Electronic Commerce: 약칭 *EC*)는 컴퓨터 망을 통한 재화 및 서비스의 매매를 의미하는 새로운 개념이다[1]. EC의 대표적인 응용 분야로서, 인터넷을 통해 주식 매매가 이루어지는 시스템인 전자 주식

매매 시스템(Electronic Stock Trading Systems: 약칭 *ESTS*)을 들 수 있다. 다른 EC 응용과 마찬가지로, ESTS는 매우 많은 수의 사용자로 하여금 데이타베이스에 저장된 막대한 양의 데이타를 공유할 수 있도록 지원해야 한다. 한편 데이타베이스 내에 저장된 각 데이타들은 응용 시스템의 보안 정책에 따라 어떤 사용자에게는 접근이 허용되고, 다른 사용자들에게는 접근이 불허되어야 한다. 이러한 환경을 지원하기 위해서는 사용자에 따라 신뢰 등급을 다르게 부여하고, 데이타에 따라 기밀 등급을 다르게 부여하여 관리하는 다등급 보안 데이타베이스 관리 시스템의 사용이 필수적이다. 대부분의 다등급 보안 트랜잭션 관리 기법에는 Bell-LaPadula(약칭 *BL*) 모형[2]의 제약 조건에 기반한 접근 제어 기법이 적용된다. BL 모형에 의하면, 임의의 트랜잭션 T의 보안 등급을 $L(T)$, 데이타 항목 x의 보안 등급을 $L(x)$라고 할 때, $L(T) = L(x)$인 경우에만 T가 x에 대해 기록 연산을 수행할 수 있고, $L(T) \geq L(x)$인 경우에만 T가 x에 대해 판독 연산을 수행할 수 있다. 이러한 제약 조건은 상위 보안 등급을 갖는 데이타가 하위 보안 등급을 갖는 트랜잭션에게 직접 노출되는 것을 효과적으로 방지한다. 하지만 이 모델의 제약 조건만으로는 비밀 경로(Covert Channel)[3]로 알려진 간접적인 정보의 유출을 완벽하게 차단할 수 없다. 비밀 경로는 서로 다른 보안 등급을 갖는 둘 이상의 트랜잭션이 공모하여, 상위 보안 등급을 갖는 트랜잭션이 하위 보안 등급을 갖는 트랜잭션에게 정보를 유출할 수 있는 불법적인 경로를 의미한다. 트랜잭션의 동시성 제어 과정에서 충돌이 발생할 수 있는데, 이 충돌로 인하여 하위 보안 등급을 갖는 트랜잭션이 상위 보안 등급을 갖는 트랜잭션의 간섭을 받는 경우 상위 등급에서 하위 등급으로의 간접적인 정보의 흐름이 생성되는 것이다.

비밀 경로는 ESTS의 안전한 수행에 치명적인 요소로 작용할 수 있다. 예를 들어 타인의 주식 거래 내역에 대한 정보는 일반 개인 투자자들의 접근으로부터 보호되어야 하는 반면, 관리적 목적을 위해 사용되는 몇몇 프로그램들은 특정 기간에 이루어진 모든 거래 내역에 대해 분석 및 요약 작업을 수행할 수 있어야 한다. 이러한 상황에서, 관리적 프로그램을 수행하는 트랜잭션과 일반 투자자가 제기한 트랜잭션간의 공모를 통해, 고급 정보가 일반 투자자들에게 노출될 위험이 있다. 예를 들어, F_1, F_2, … F_{101}에 해당하는 101명의 투자자가 동일한 주식 종목을 소유하고 있다고 가정하자. 어떤 특정 기간에 F_1이 10000주의 매도 주문을 내고, F_2, F_3, …, F_{101}의 100명의 투자자가 각각 100주씩 매수주문을 냈다고 하자. 이 때 전체 매수 주문량과 매도 주문량은 모두 10000주씩으로 동일하다. 한편 이와 별도로 전체 투자자의 투자 현황을 파악하는 분석적인 트랜잭션 T_a가 수행되고 있다고 하자. 이 때, T_a가 F_{101}이 제기한 트랜잭션 T_{101}과 모하여, 매도 주문 10000주는 한 사람이 요청하고, 매수 주문 10000주는 100 사람에 의해서 요청되었다는 정보를 F_{101}에게 유출할 수 있다. 이 경우 F_{101}은 전체 매수 주문량과 매도 주문량만을 파악하고 있는 다른 투자자 F_1, F_2, …, F_{100}에 비해서 보다 유리한 투자 결정을 할 수 있다. 이러한 예는 불법적인 경로를 통해 정보가 유출되는 상황을 묘사한 것으로, 유사한 시나리오가 현실화 되는 것을 막기 위해 ESTS는 트랜잭션 처리 과정에서 모든 비밀 경로의 생성 가능성을 반드시 차단하여야 한다.

비밀 경로를 통한 불법적인 정보의 유출을 막기 위해 다양한 연구가 수행되어 왔다. 그 중 대표적인 접근 방법으로 하위 보안 등급을 갖는 트랜잭션에게 상대적으로 높은 우선순위를 부여하여, 하위 등급의 트랜잭션이 상위 등급 트랜잭션의 간섭을 받지 않도록 하는 불간섭(Non-Interference)[4] 방식이 있다. 이 접근 방식에 따른 연구는 각 알고리즘이 트랜잭션 관리를 위해 유지하는 데이타 버전의 수에 따라 크게 3 가지 접근 방식으로 분류될 수 있다. 우선 별도의 복사본 없이 하나의 버전만으로 트랜잭션을 관리하는 단일 버전 기반 방식[5, 6]이 있고, 여러 개의 복사본을 두되 그 수에 제한을 두는 방식[7, 8]이 있으며, 복사본의 수에 제한을 두지 않는 멀티 버전 기반 방식[9, 10, 11]이 있다. 이들 세 가지 접근 방식은 관리 비용과 가용성 측면에서 상충관계를 보인다. 단일 버전 기법이 사용되는 경우에는 추가적인 복사본을 관리할 필요가 없기 때문에 관리 비용이 상대적으로 낮아진다는 장점이 있다. 하지만 별도의 복사본을 관리하지 않으므로, 서로 다른 보안 등급을 갖는 트랜잭션이 서로 충돌하는 연산을 제기하는 경우, 비밀 경로의 차단을 위해 상위 보안 등급을 갖는 트랜잭션이 철회 혹은 지연되어야 하는 불이익을 감수해야 한다. 이러한 정책은 보안 등급간의 형평성을 보장하지 못한다는 단점을 갖는다. 반대로, 관리하는 버전의 수가 많을수록 가용성 및 형평성은 향상되며, 버전의 관리에 따른 추가 비용 또한 증가함을 알 수 있다. 둘 이상의 버전을 관리하는 전통적인 시스템에서는 데이타의 일관성을 검증하기 위해 단일 복사본 직렬화 가능성(One-Copy Serializability) [10, 11, 12, 13]을 정확성의 기준으로 삼는다. 버전 관리 비용과 가용성간의 상충 관계로 인해, 기법간의 성능 차이는 그 기법이 대상으로 하는 응용 영역의 특성에 따라 다르게 나타난다. 예를 들어, ESTS 환경에서는 매우 방대한 양의 데이타가 사용되므

로, 서로 다른 등급간에 충돌하는 연산이 제기되는 회수가 다른 응용 환경에 비해 상대적으로 적게 나타난다. 따라서 매우 드물게 발생하는 서로 다른 등급 트랜잭션간의 충돌을 막기 위해 많은 수의 복사본을 유지한다는 것은 매우 비효율적이다. 오히려 적은 수의 복사본을 유지하면서, 드물게 나타나는 상이 등급 트랜잭션간의 충돌을 안전하게 관리하는 방식이 ESTS 환경에 더 적합하다.

기존의 보안 관련 연구들은 실시간 트랜잭션[8, 9, 14, 15, 16], 판독 전용 장기 트랜잭션[17] 등, 각 응용 영역의 특정한 성격에 따라 알고리즘의 초점을 맞추어 수행되어 왔다. 하지만 실시간 트랜잭션과 판독 전용 트랜잭션을 동시에 효과적으로 관리하면서 비밀경로의 생성 가능성을 차단하기 위한 시도는 매우 부족하다. ESTS는 다양한 속성을 갖는 트랜잭션이 동시에 수행되는 대표적인 환경이다. 예를 들어 ESTS에서 주식의 매매 주문이 발생하는 동시에, 주식 시장 전체의 흐름에 대한 통계적 분석을 수행하는 관리 트랜잭션이 실행되고 있는 상황을 살펴보자. 이 경우 주문을 위한 각각의 트랜잭션은 OLTP 트랜잭션의 성향을 갖는 반면, 분석을 위한 관리 트랜잭션은 OLAP 트랜잭션의 성향을 갖는다. OLAP 트랜잭션은 주로 최고 경영자 및 경영 분석가와 같이 상대적으로 높은 보안 등급을 갖는 사용자들에 의해서 제기된다[18]. OLAP과 OLTP 트랜잭션의 대표적인 차이점을 표 1에 기록하였다.

일반적으로 주식 매매 시스템에서의 OLAP 트랜잭션은 다양한 인덱스와 요약 데이타를 추출하기 위한 판독 전용 트랜잭션의 형태로 수행된다[19]. OLAP과 OLTP 트랜잭션이 동시에 수행되는 ESTS 환경에서의 트랜잭션 관리를 위해 스케줄러는 다음과 같은 사항을 고려하여야 한다. 우선 높은 보안 등급을 갖는 OLAP 트랜잭션의 수행 결과가 낮은 보안 등급을 갖는 OLTP 트랜잭션에게 노출되는 가능성을 차단하여야 한다[18]. 또한 OLAP 트랜잭션이 긴 시간에 걸쳐 수행되더라도, OLTP 트랜잭션의 간섭으로 인해 그 수행이 반복 철회되지 않도록 보장해주어야 한다.

1.2 동기 및 목적

본 논문에서는 OLAP과 OLTP 트랜잭션이 동시에 수행되는 환경인 ESTS에서의 보안 트랜잭션 관리를 위해, 보안 단일 스냅샷(Secure One Snapshot: 약칭 *SOS*) 프로토콜을 제안한다. SOS는 다음의 요구 조건을 만족시키는 것을 목적으로 고안되었다. 우선 동시성 제어 과정에서 데이타의 일관성을 보장하여야 하며, 동시에 비밀 경로의 생성 가능성을 차단하여 보안성을 유지하여야 한다. 또한 많은 수의 OLTP 트랜잭션이 동시에 수행되는 ESTS의 특성상, 실시간 트랜잭션 관리를 위해 고안된 다양한 동시성 제어 알고리즘이 쉽게 적용될 수 있어야 한다. 그리고 최소한의 복사본만을 유지하여 복사본 관리의 부담을 줄임과 동시에, 복사본 갱신을 위해 주기억장치 상에서 관리되는 큐의 길이를 짧게 유지하면서도 판독 연산을 수행하는 트랜잭션에 대하여 가능한 한 최근의 값을 제공하는 것을 그 목적으로 한다.

SOS의 동작 과정은 다음과 같이 요약될 수 있다. SOS는 복사본 관리의 부담을 최소화하기 위해 하나의 운용 데이타베이스(Working Database: 약칭 *WDB*)와 단 하나의 스냅샷(Snapshot: 약칭 *SS*)만을 유지한다. 임의의 트랜잭션 T_i와 데이타 항목 x에 대해서 $L(T_i) = L(x)$인 경우, x에 대한 T_i의 기록 및 판독 연산은 WDB에서 수행된다. 한편 $L(T_i) > L(x)$인 경우 x에 대한 T_i의 하향 판독 연산은 SS에 저장된 복사본의 값을 판독한다. 하향 판독 연산이 지나치게 오래된 값을 읽어가는 현상을 방지하기 위해, SS의 데이타 값은 주기적 혹은 비주기적 방법을 통해 WDB에 저장된 값으로 갱신되어야 한다. 만약 이러한 갱신이 지나치게 늦게 이루어진다면, 하향 판독 연산에 대해서 점차적으로 오래된 값을 제공하게 되어 의미 있는 결과를 도출할 수가 없다. 반대의 경우로 WDB에 값이 기록되는 순간 SS에 그 값이 반영된다면 데이타의 일관성 유지가 어렵게 된다. SOS는 BL 모형의 다음과 같은 속성을 활용하여, 앞서 언급한 요구 조건을 만족시키며 WDB와 SS을 관

표 1 OLAP 트랜잭션과 OLTP 트랜잭션의 속성 비교

	OLAP	OLTP
Feature	Analytical	Operational
Clearance Level	Usually High	Usually Low
Occurrence	Unusual	Very Often
Execution Time	Usually Long Time is Required	Usually Short and Deadline Critical
Main Operation	Mostly Read	Write as well as Read

리하기 위한 알고리즘을 제시한다. 상이한 보안 등급을 갖는 트랜잭션간의 연산 충돌은 임의의 데이타에 대하여 하향 판독 연산과 동급 기록 연산이 서로 충돌하는 경우 이외에는 발생하지 않는다. 또한 모든 트랜잭션은 자신보다 낮은 보안 등급을 갖는 데이타에 대해서는 판독 연산만을 수행할 수 있기 때문에, 이들 데이타의 관점에서는 상위 보안 등급을 갖는 트랜잭션은 판독 전용 트랜잭션으로 간주될 수 있다는 속성을 활용한다.

본 논문의 이후 구성은 다음과 같다. 다음 절인 2절에서는 본 논문에서 제안하는 알고리즘과 관련된 이전의 연구 결과에 대해 간략히 살펴보며, 3절에서는 ESTS 환경에 적합한 보안 동시성 제어 기법을 새롭게 제안한다. 제안하는 알고리즘에 대한 정확성 및 보안성 분석은 4절에서 다루며, 기존에 고안된 기법과의 성능 비교 평가 및 결과 분석은 5절에서 수행한다. 마지막으로 본 연구의 한계점 및 결론은 6절에서 제시한다.

2. 관련 연구

본 절에서는 SOS와 관련된 이전의 연구 결과에 대한 간략한 설명을 제시한다. 우선 실시간 트랜잭션의 효과적 수행을 위해 판독 전용 트랜잭션과 갱신 트랜잭션의 동시성 제어 알고리즘을 분리한 [20]의 연구에 대해 살펴본다. [20]은 보안 환경을 가정하진 않았으나, 하향 판독 연산을 수행하는 상위 보안 등급 트랜잭션은 판독 전용 트랜잭션의 속성을 가지므로, [20]의 연구 결과는 본 논문에서 제안하는 프로토콜과 밀접한 관계가 있다. 다음으로 보안 환경에서 실시간 트랜잭션을 효과적으로 수행하기 위한 시도인 [21]의 연구 결과를 살펴본다. [21]은 이한 보안 등급을 갖는 트랜잭션간의 충돌을 등급간 충돌(Inter-Conflict)로, 동일한 보안 등급을 갖는 트랜잭션간의 충돌을 등급내 충돌(Intra-Conflict)로 구분함으로써 실시간 보안 동시성 제어를 위한 알고리즘을 제시하였다. 다음으로 살펴볼 [8]의 연구에서는 하나의 운용 데이타베이스와 하나의 복사본, 그리고 큐를 사용하여 실시간 보안 동시성 제어를 시도한 알고리즘이 제시되었다. [8, 20, 21]의 세 가지 기존 연구에 대해 간략하게 살펴본 후, 다음 절에서 보다 향상된 실시간 보안 동시성 제어 프로토콜인 SOS의 동작 과정을 기술하기로 한다.

2.1 뷰 일관성(View-Consistency)[20]

[20]은 트랜잭션의 동시성 제어에 있어서, 갱신 트랜잭션과 판독 전용 트랜잭션을 분리하여 관리하는 기법을 제시하였다. 즉 모든 판독 전용 트랜잭션은 시작 전에 자신이 기록 연산을 수행하지 않을 것임을 명시하며, 이들 트랜잭션에 대해서는 충돌 직렬화 가능성(Conflict-Serializability)이 아닌 뷰 일관성을 유지함으로써 데이타 일관성을 보장할 수 있음을 보였다. 정확성 기준으로 충돌 직렬화 가능성을 채택하는 경우와 뷰 일관성을 채택하는 경우에 대한 성능의 차이를 그림 1[20]에서 파악할 수 있다.

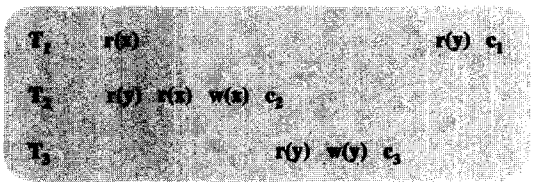


그림 1 뷰 일관성 관점에서 적법하게 수행되는 수행 기록 예

그림 1에서 T_1은 판독 전용 트랜잭션을, T_2 및 T_3는 갱신 트랜잭션을 나타낸다. 위 그림에 묘사된 수행 기록의 충돌 직렬화 가능성 그래프는 $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_1$으로 나타난다. 하지만 [20]의 연구에서는 그림 1의 $T_2 \rightarrow T_3$의 관계를 충돌로 파악하지 않는다. 왜냐하면 T_3의 연산은 T_2가 기록한 어떤 값도 참조하지 않았기 때문이다. [20]에서는 뷰 일관성을 보장하기 위해 판독 종속성 그래프(Read-From Graph: 약칭 *RFG*)를 사용한다. RFG에서는 기록 - 판독 충돌, 그리고 판독 전용 트랜잭션의 판독 - 갱신 트랜잭션의 기록 충돌 시에만 간선을 추가한다. 즉 갱신 트랜잭션간의 판독 - 기록 충돌, 그리고 기록 - 기록 충돌의 경우는 간선을 추가하지 않는다. 따라서 그림 1에 해당하는 RFG는 $T_3 \rightarrow T_1 \rightarrow T_2$로 나타난다. 즉 그림 1의 수행 기록은 충돌 직렬화 가능성 관점에서 보면 적법하지 않은 것으로 파악되지만, 뷰 일관성의 측면에서 보면 적법한 것으로 파악될 수 있다. [20]은 갱신 트랜잭션간에는 충돌 직렬화 가능성을, 판독 전용 트랜잭션간에는 뷰 일관성을 유지함으로써 데이타의 일관성을 보장하면서 성능의 향상을 가져올 것을 제안하였다. SOS는, 서로 다른 보안 등급을 갖는 트랜잭션간의 연산이 충돌하는 경우, 상위 보안 등급을 갖는 트랜잭션은 항상 하향 판독 연산의 형태로 충돌에 포함되므로 판독 전용 트랜잭션과 같이 동작한다는 특성에 착안하여 [20]의 기법을 적용하였다. 즉 SOS는 복사본의 갱신 과정에서 충돌 직렬화 가능성뿐 아니라 뷰 일관성을 정확성 기준으로 채택할 수 있으며, 두 가지 기준을 각각 적용했을 경우의 성능 이득 분석을 5절에서 수행하였다.

2.2 이중 접근 방법(Dual Approach)[21]

[21]의 연구에서는 보안과 실시간 트랜잭션의 요구 조건을 동시에 고려한 알고리즘이 제시되었다. 즉 서로 다른 보안 등급에 속하는 트랜잭션간의 등급간 동시성 제어에는 보안 동시성 제어 기법을, 동일한 보안 등급을 갖는 트랜잭션간의 등급 내 동시성 제어에는 실시간 동시성 제어 기법을 적용하는 이중 접근 방법(Dual Approach)을 제안하였다. [21]은 보안 요구 사항을 정확성 기준에 속하는 것으로 인식하여 반드시 지켜져야 한다고 주장한 반면, 실시간 요구 사항은 성능 기준에 속하는 것으로 파악하여 가능한 범위 내에서 고려되어야 한다고 주장하였다. 하지만 이러한 접근 방식은 상위 보안 등급을 갖는 트랜잭션이 항상 낮은 우선 순위를 갖도록 강요함으로써 평균 응답시간 측면에서 보안 등급간의 형평성을 보장하지 못한다. SOS는 하나의 운용 데이타베이스와 하나의 복사본을 관리하여 서로 다른 보안 등급을 갖는 트랜잭션간에는 연산 충돌이 일어나지 않게 함으로써, 보안과 실시간의 요구 조건을 분리하였다. 즉 연산 충돌은 항상 같은 보안 등급을 갖는 트랜잭션간에만 발생하므로, 같은 보안 등급을 갖는 트랜잭션 내에서 실시간 기준으로 우선 순위를 부여함으로써 보안 및 실시간 요구 조건을 충족시킬 수 있다.

2.3 실시간 보안 이단계 로킹(Secure Real-Time Two-Phase Locking: 약칭 *SRT-2PL*)[8]

SRT-2PL은 비밀 경로의 생성을 막기 위해 기본 복사본(Primary Copy)과 이차 복사본(Secondary Copy)을 유지한다. 이차 복사본은 상위 보안 등급을 갖는 트랜잭션의 하향 판독 연산에만 사용되며, 나머지 연산들은 기본 복사본에서 수행된다. 기본 복사본에 새로 기록된 값은 일시적으로 별도의 큐에 저장된다. 큐에 저장된 값은 일정 시간이 지나 특정 조건이 만족되면 이차 복사본으로 반영된다. 이러한 자료 구조를 사용하여 서로 다른 보안 등급을 갖는 트랜잭션간에는 충돌이 생기지 않도록 하고, 같은 등급에 속하는 트랜잭션간에는 실시간 제약에 따라 우선 순위를 부여함으로써 실시간 요구 조건도 효과적으로 관리할 수 있다. 하지만 SRT-2PL은 모든 트랜잭션으로 하여금 시작 시점에 자신이 접근할 모든 데이타에 대한 로크(Lock)를 소유하도록 강요하기 때문에, 트랜잭션 수행의 동시성이 낮아지게 된다. 또한 SRT-2PL은 데이타의 신선도를 향상시키기 위해, 기록 연산의 수행 즉시 변경된 데이타의 값을 주기억장치 상의 큐에 추가시킴으로써 이후 제기되는 상위 등급 트랜잭션의 하향 판독 연산에게 큐에 임시 저장된 값을 제공하는 정책을 사용한다. 이러한 정책은 데이타의 신선도를 향상시킬 수 있지만, 변경된 값이 큐에 머무는 시간이 길어짐으로 인해 주기억장치 상의 큐의 길이가 점차 길어진다는 부작용을 가져온다. 이러한 부작용은 그림 2를 통해 설명할 수 있다(x는 초기값으로 x_0를 갖는 것으로 가정함).

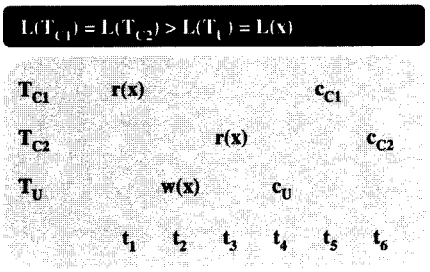


그림 2 기록 연산의 수행으로 인한 큐 길이의 증가

그림 2에서 T_{C1}은 초기 상태의 x 값인 x_0를 판독하는 반면, T_{C2}는 T_U에 의해 기록된 x_U 값을 판독한다. 하지만 이를 위해 SRT-2PL은 T_U의 기록 연산이 수행된 직후인 t_2부터 x_U 값을 큐에 추가할 것을 강요한다. 큐에 저장된 x_U 값은 그 값을 판독하고 있는 T_{C2}의 완료 이후에 큐에서 제거될 수 있으므로, x_U 값은 t_2 시점부터 t_6 시점까지 큐에 머물게 된다. 반면 SOS는 변경된 데이타 값을 기록 연산을 수행한 트랜잭션이 완료 연산을 수행한 이후 큐에 추가하므로, SOS의 경우 그림 2에서 x_U 값이 큐에 머무는 기간은 t_4 시점부터 t_6 시점까지로 SRT-2PL에 비해 짧게 나타난다. 하지만 SOS의 경우 T_{C1}의 판독 연산과 T_{C2}의 판독 연산 모두에게 x_0 값을 제공하게 된다. 즉 SOS와 SRT-2PL은 데이타의 신선도와 큐 관리 부담 측면에서의 이율배반적인 특성을 갖게되며, 이에 대한 분석이 5절에 제시되어 있다. 한편 SOS는 트랜잭션이 시작 시점에 데이타에 대한 모든 로크를 소유할 필요가 없으므로 보다 유연한 동시성 제어가 가능할 뿐 아니라, 상위 보안 등급 트랜잭션이 갖는 판독 전용 트랜잭션의 특성을 이용하여 하향 판독 연산에 제공되는 데이타의 신선도 및 큐 관리 부담 측면에서의 향상을 가져올 수 있다.

3. SOS 프로토콜

3.1 데이타 구조

우선순위 역전 현상 없이 비밀경로를 차단하기 위해서, SOS는 각 데이타에 대해 WDB 외에 SS를 별도로 관리한다. 즉, $L(T_i) = L(x)$인 경우에는 WDB에서 동급 기록 혹은 동급 판독 연산이 수행되며, $L(T_i) > L(x)$인

하향 판독 연산의 경우, 데이타 x의 값을 SS로부터 제공받는다. BL 모형의 제약 조건에 의하여, 높은 등급을 갖는 트랜잭션의 하향 판독 연산과 낮은 등급을 갖는 트랜잭션의 동급 기록 연산이 충돌하는 경우 이외에는 상이한 등급 트랜잭션간의 충돌은 발생할 수 없다. 그런데 SOS는 데이타 x에 대해 $L(T_i) > L(x)$인 T_i의 판독 연산에는 SS의 값을 제공하고, $L(T_i) = L(x)$인 T_i의 기록 연산은 WDB에 반영되므로, 비밀 경로의 생성 위험이 있는 상이 등급 트랜잭션간의 충돌을 사전에 방지할 수 있다. 한편 WDB의 데이타에 대해 판독 혹은 기록 연산을 수행하는 트랜잭션은 모두 해당 데이타와 동일한 보안 등급을 가지므로, WDB 내에서는 비밀 경로의 생성 위험 없이 기존의 동시성 제어 기법을 사용하여 데이타의 일관성을 유지할 수 있다.

SS에서는 기록 연산이 수행되지 않으므로, SS에 저장된 데이타 값들은 트랜잭션의 기록 연산 수행을 통한 직접적인 갱신이 이루어지지 않는다. 따라서 SS의 데이타 값이 지나치게 오래된 값을 갖지 않도록 하기 위해, WDB에 저장된 최근의 데이타 값을 SS에 반영하기 위한 알고리즘이 필요하다. 본 논문에서는 이처럼 WDB의 값을 SS에 반영하는 행위를 공표(Publish: 약칭 *Pub*)라고 명명하였다. 또한 임의의 트랜잭션의 공표를 나타내는 연산을 *Pub*( )로 표기하였다. 예를 들어 $Pub(T_i)$는 T_i의 기록 연산에 의해 WDB에 저장된 데이타의 값을 SS에 공표하는 연산을 의미한다. 공표 과정에서 데이타의 일관성과 보안성을 유지하기 위해서는 몇 가지 제약이 필요하다. 이를 위해 SOS는 공표 순서 그래프(Publication Order Graph: 약칭 *POG*)를 관리한다.

3.2 공표 순서 그래프

POG는 트랜잭션들의 공표 순서를 명시하며, 판독 종속성[20]에 근거하여 작성된다. 즉, 트랜잭션의 완료 시점에서 다른 트랜잭션과의 판독 종속성이 발견되는 경우 POG에 해당 간선이 추가된다. 한편 모든 공표 연산은 POG에 명시된 순서에 의해서 수행되어야 한다. 예를 들어 POG에 $T_d \rightarrow T_i$의 관계가 존재하는 경우, $Pub(T_i)$는 $Pub(T_d)$의 이후에만 수행될 수 있다. 이처럼 POG에 간선을 추가하는 경우를 묘사하기 위해 규칙 1, 2, 그리고 3을 정의한다. 또한 POG의 순서에 근거하여 공표 연산을 수행하는 과정을 위해 규칙 4와 5를 정의한다. 모든 규칙은 트랜잭션의 완료 시점에 적용되며, 적용 순서는 규칙 1, 2, 3, 4, 그리고 5의 순서를 따른다. 만약 이러한 규칙의 도입 없이 트랜잭션의 완료 직후 공표 연산이 수행된다면 데이타의 일관성이 보장될 수 없다(그림 3).

우선 그림 3의 수행 기록 중 T_S와 T_C만을 고려하여 보자. T_S는, T_C에 의해 기록되고 공표된 데이타 k를 판독하는 반면 아직 T_C의 영향을 받지 않은 데이타 t를 판독한다. 이 경우 T_S는 T_C의 부분적인 영향만을 반영하였기 때문에 비 일관적 데이타를 판독한 것으로 파악될 수 있다. 데이타 일관성의 보장은, 해당 RFG에서 순환 고리가 형성되지 않음을 보임으로써 증명될 수 있다. 그림 3에 해당하는 RFG에서는 데이타 t에 대한 종속성을 나타내는 $T_S \rightarrow T_C$의 간선과 데이타 k에 대한 종속성을 나타내는 $T_C \rightarrow T_S$의 간선으로 인해 순환 고리가 형성됨을 알 수 있다. 이러한 유형의 고리가 형성되는 것을 방지하기 위해 SOS는 다음의 규칙 1을 규정한다.

규칙 1: 완료된 임의의 트랜잭션 T_i와 $L(T_d) > L(T_i)$이며 현재 수행중인 트랜잭션 T_d, 그리고 $L(T_i) = L(x)$인 임의의 데이타 항목 x에 대해 $r_d(x) <_H w_i(x)$ 혹은 $w_i(x) <_H r_d(x)$이 수행 기록 내에 존재하는 경우 $T_d \rightarrow T_i$를 POG에 추가한다.

규칙 1에 의해서, T_C의 완료 직후 $T_S \rightarrow T_C$의 간선이 POG에 추가되어야 한다. 따라서 T_C의 공표는 T_S의 공표 이후로 그 수행이 지연되어야 한다. 이러한 지연으로 인해 T_S는 데이타 t와 k 모두에 대해서 T_C의 영향을 받지 않은 값을 판독하므로 데이타의 일관성이 보장될 수 있다. 다음으로, T_S, T_C, 그리고 T_{U1}을 함께 고려하여 보자. T_{U1}의 완료 시점을 보면, 그 시점에서 현재

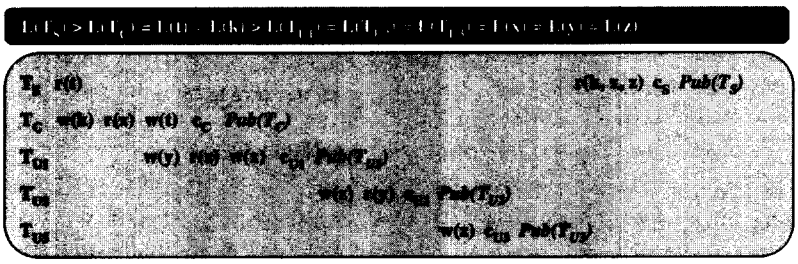


그림 3 공표 연산의 즉시 수행으로 인해 데이타의 일관성이 보장되지 않는 예

수행 중인 트랜잭션 중 x 혹은 y에 대해 하향 판독 연산을 수행한 트랜잭션은 존재하지 않음을 알 수 있다. 즉, T_{U1}은 규칙 1에 의해 어떤 간선도 POG에 추가하지 않으므로 완료 즉시 공표 연산을 수행할 수 있다. 하지만, 이러한 즉각적인 공표 연산의 수행으로 인해 T_S가 T_{U1}에 의해 기록된 x 값을 판독함으로써 RFG에 순환 고리가 형성될 수 있다. 따라서 $Pub(T_{U1})$의 수행을 지연시키기 위해 규칙 2를 도입한다.

규칙 2: 완료된 임의의 트랜잭션 T_i와 $L(T_d) > L(T_i)$이며 완료된 트랜잭션 T_d, 그리고 $L(T_i) = L(x)$인 임의의 데이타 항목 x에 대해 $r_d(x) <_H w_i(x)$ 혹은 $w_i(x) <_H r_d(x)$이 수행 기록 내에 존재하고 T_d에 해당하는 노드가 이미 POG에 존재하는 경우 $T_d \rightarrow T_i$를 POG에 추가한다.

규칙 2는 T_i의 공표를 위해서는 T_i와, 하향 판독 연산 수행 후 완료된 상태에서 공표되기를 기다리고 있는 트랜잭션 T_d와의 판독 종속성 조사가 선행되어야 함을 규정한다. 이 시점에서 T_S, T_C, 그리고 T_{U1} 과 함께 T_{U2}를 고려하여 보자. T_{U2}의 완료 시점에서는 z에 대해 하향 판독 연산을 수행한 후 완료 혹은 진행 상태에 있는 트랜잭션이 존재하지 않는다. 따라서 T_{U2}는 규칙 1 또는 2에 의해 어떤 간선도 POG에 추가하지 않으므로 $Pub(T_{U2})$ 연산은 T_{U2}의 완료 즉시 수행될 수 있다. 하지만 이 경우 역시 T_S의 z에 대한 하향 판독 연산이 T_{U2}에 의해 기록된 값을 제공받게 됨으로써 RFG에 순환 고리가 형성되게 된다. 이 경우는 $T_S \rightarrow T_C$의 판독 종속성이 $T_C \rightarrow T_{U1}$과 $T_{U1} \rightarrow T_{U2}$의 관계를 통해 T_{U2}에게 전달된 것이다. 따라서 $Pub(T_{U2})$ 연산은 T_{U1}의 공표 이후에 수행되어야 한다. 이처럼 판독 종속성의 전달 현상을 방지하기 위해 SOS는 규칙 3을 정의한다.

규칙 3: 완료된 임의의 트랜잭션 T_i와 $L(T_d) = L(T_i)$이며 완료된 트랜잭션 T_d, 그리고 $L(T_i) = L(x)$인 임의의 데이타 항목 x에 대해 $w_d(x) <_H r_i(x)$가 수행 기록 내에 존재하고 T_d에 해당하는 노드가 이미 POG에 존재하는 경우 $T_d \rightarrow T_i$를 POG에 추가한다.

규칙 3에서 $L(T_d) = L(T_i)$인 경우 POG의 $T_d \rightarrow T_i$의 간선은 $w_d(x) <_H r_i(x)$가 수행 기록 내에 존재하는 경우에만 추가됨에 유의하여야 한다. 즉, $w_d(x) <_H w_i(x)$ 또는 $r_d(x) <_H w_i(x)$의 경우에는 별도의 노드 혹은 간선을 POG에 추가하지 않는다. 이는 규칙 3이 뷰 일관성[20]에 근거하여 작성되었음을 의미한다. [20]은 판독 전용 트랜잭션에 대해서는 충돌 직렬화 가능성이 아닌 뷰 일관성에 관한 제약만으로도 데이타의 일관성을 유지할 수 있음을 보이고, 뷰 일관성의 검증을 위해 RFG를 사용할 것을 제안하였다. 임의의 데이타 x에 대하여 $L(T_i) > L(x)$인 트랜잭션 T_i는 x에 대하여 판독 연산만을 수행할 수 있으므로, x의 관점에서는 T_i를 판독 전용 트랜잭션으로 취급할 수 있다. 즉 SS에서 값을 판독하는 모든 트랜잭션은 실제 기록 연산의 수행 여부와 관계 없이, SS의 관점에서는 판독 전용 트랜잭션으로 취급하여 [20]의 기법을 적용할 수 있다. [20]에서는 판독 전용 트랜잭션이 다른 트랜잭션의 수행 결과의 부분적으로 반영하는 것을 막기 위해 RFG를 사용하였으며, 갱신 트랜잭션간의 충돌은 $w_d(x) <_H r_i(x)$인 경우에만 $T_d \rightarrow T_i$를 추가함으로써 데이타의 일관성 유지가 가능함을 보였다. 따라서 T_{U3}의 경우 다른 어떤 트랜잭션에 대해서도 판독 종속성을 가지고 있지 않으므로, 완료 즉시 공표 연산을 수행할 수 있다. 그림 3은 규칙 1, 2, 그리고 3의 적용 후에 그림 4로 수정되어야 한다. 수정된 수행 기록의 RFG에서 순환 고리가 존재하지 않음을 보임으로써 뷰 일관성의 보장 여부를 검증할 수 있다. 그림 5는 그림 4에 대한 RFG를 나타낸다. 보다 자세한 증명은 4절에서 다루기로 한다.

다음으로 POG의 선후 관계에 의하여 적법하게 공표 연산을 수행하기 위한 규칙을 정의한다. 임의의 트랜잭션 T_i의 완료 순간에 규칙 1, 2, 그리고 3이 순차적으로 적용된다. 만약 위의 세 규칙이 적용된 후에도 POG 상에 T_i의 노드가 존재하지 않는 경우에는 $Pub(T_i)$ 연산

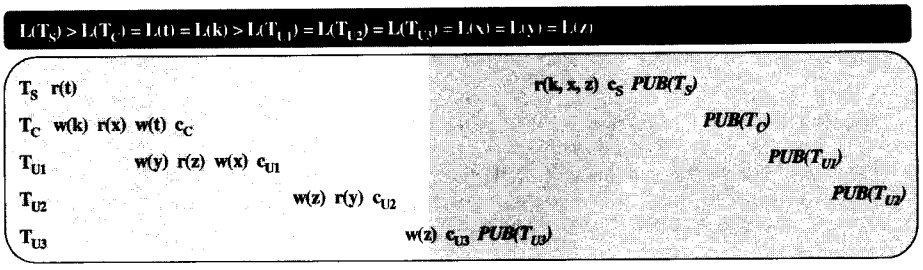


그림 4 규칙 1, 2, 그리고 3에 의한 공표 연산의 지연 예

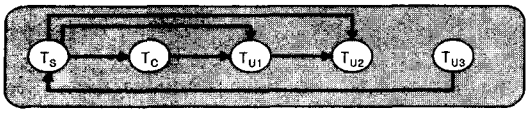


그림 5 그림 4에 대한 RFG

이 곧바로 수행될 수 있다. 이 경우를 위해 규칙 4를 정의한다. 또한 POG에 T_i의 노드가 존재하는 경우라도, T_i로 향하는 내향 간선이 존재하지 않는 경우에는 $Pub(T_i)$ 연산이 수행될 수 있다. 이 경우를 위해 규칙 5를 정의한다. 규칙 5는 다른 트랜잭션의 공표를 기다리고 있는 트랜잭션들에 대하여 반복적인 공표 연산의 수행이 가능하도록 하기 위해 순환적으로 정의된다.

규칙 4: POG 상에 T_i 노드가 존재하지 않는 경우 $Pub(T_i)$를 즉시 수행한다.

규칙 5: POG 상에 T_i 노드가 존재하지만 T_i로 향하는 내향 간선이 없는 경우 $Pub(T_i)$를 수행한다. 또한 T_i로부터 출발하는 모든 외향 간선을 제거한 후 노드 T_i를 제거한다. T_i의 후행 노드 중 변경된 POG 상에서 내향 간선을 가지고 있지 않는 노드에 대하여 규칙 5를 순환적으로 적용한다

그림 4의 수행 기록에 대해서 다섯 개의 규칙이 적용되는 과정을 살펴보면 다음과 같다. 간선 $T_S \rightarrow T_C$, $T_C \rightarrow T_{U1}$, 그리고 $T_{U1} \rightarrow T_{U2}$ 이 각각 규칙 1, 2, 그리고 3에 의하여 POG에 추가되어야 한다. 그 결과로 트랜잭션 T_C, T_{U1}, 그리고 T_{U2}는 공표 시점을 T_S의 공표 이후로 연기하여야 한다. 한편 T_{U3}는 규칙 1, 2, 그리고 3에 의해서 어떠한 간선 및 노드도 POG 상에 추가시키지 않으므로, 규칙 4에 의해 완료 즉시 공표 연산을 수행할 수 있다. T_S의 완료 시점을 보면, T_S에 대한 노드가 POG 상에 존재하지만 T_S로 향하는 내향 간선은 존재하지 않으므로 이 경우 규칙 5가 적용된다. 규칙 5의 순환적 적용에 의하여, T_S의 후행 노드인 T_C 역시 공표 연산을 수행하게 된다. 동일한 규칙에 의해, $Pub(T_C)$의 수행 이후 $Pub(T_{U1})$, $Pub(T_{U2})$의 연산이 차례로 수행된다. 결과로 형성되는 전체 공표 순서는 $T_{U3} \rightarrow T_S \rightarrow T_C \rightarrow T_{U1} \rightarrow T_{U2}$ 로 나타나게 된다.

3.3 SOS의 의사 알고리즘

본 절에서 제시한 POG에 대한 다섯 가지 규칙들을 조합함으로써 SS의 갱신 과정 전체에 대한 의사 알고리즘을 생성할 수 있다. 생성된 의사 알고리즘은 다음과 같다.

4.보안성 및 데이타 일관성 분석

본 절에서는 SOS에 의해 생성된 수행 기록이 데이타 일관성을 유지함과 동시에 비밀 경로를 효과적으로 차단할 수 있다는 사실에 대한 근거를 제시한다. 증명의 형식은 이해도를 높이기 위해 준 정형적인 형태를 취하기로 한다.

보조정리 1: 동일한 보안 등급을 갖는 모든 트랜잭션은 충돌 직렬화 가능하다.

증명: SOS는 동일 등급 트랜잭션간의 동시성 제어에는 별도의 제약을 가하지 않는다. 즉 기존에 제안된 이단계 로킹 기법(2 Phase Locking: 약칭 *2PL*)이나 타임스탬프 순서화 기법(Timestamp Ordering: 약칭 *TO*)을 사용하여 동일 등급의 트랜잭션간 충돌 직렬화 가능성을 보장할 수 있으며, 이에 대한 정형적인 증명은 [12]

```
when transaction Ti is committed{
        for every active transaction Td such that L(Td) > L(Ti)
            if rd(x) <H wi(x) or wi(x) <H rd(x) exists in schedule H, then add Td → Ti to POG      //규칙 1//
        for every committed transaction Td in POG such that L(Td) > L(Ti)
            if rd(x) <H wi(x) or wi(x) <H rd(x) exists in schedule H, then add Td → Ti to POG      //규칙 2//
        for every committed transaction Td in POG such that L(Td) = L(Ti)
            if wd(x) <H ri(x) exists in schedule H, then add Td → Ti to POG                        //규칙 3//
        if there is still no node for Ti in POG, then execute Pub(Ti)                              //규칙 4//
        else if there is no incoming edge to the node for Ti, then execute recursive_Pub(Ti)       //규칙 5//
}
recursive_Pub(Ti){
        for every transaction Tk such that Ti → Tk exists in POG{
            delete edge of Ti → Tk from POG
            if there is no more incoming edge to Tk, then execute recursive_Pub(Tk)
        }
        delete node for Ti from POG
        execute Pub(Ti)
}
```

에 소개되어 있다. □

SOS는 $L(T_C) > L(T_{U1}) = L(T_{U2}) = \cdots = L(T_{Un})$의 트랜잭션이 포함된 수행 기록에서 T_C가 T_{U1}, T_{U2}, …, T_{Un}에 대하여 뷰 일관성을 유지할 수 있도록 보장한다. 즉 상위 등급에 속하는 트랜잭션이 하위 등급을 갖는 임의의 트랜잭션의 수행 결과를 반영할 경우, 그 수행 결과를 모두 반영하거나, 혹은 전혀 반영하지 않도록 보장한다.

정리 1: 완료된 모든 트랜잭션은 보안 등급에 관계없이 뷰 일관성이 보장된다.

증명: $L(T_{C1}) = L(T_{C2}) = \cdots = L(T_{Cn}) > L(T_{U1}) = L(T_{U2}) = \cdots = L(T_{Um})$인 트랜잭션 T_{C1}, T_{C2}, …, T_{Cn}, T_{U1}, T_{U2}, …, T_{Um}으로 구성된 수행 기록을 가정하자. 보조정리 1에 의해 동일한 보안 등급을 갖는 모든 트랜잭션은 직렬화 가능하므로, $T_{C1} \rightarrow T_{C2} \rightarrow \cdots \rightarrow T_{Cn}$, $T_{U1} \rightarrow T_{U2} \rightarrow \cdots \rightarrow T_{Um}$의 직렬화 순서를 일반성을 해치지 않고 가정할 수 있다. 뷰 일관성의 보장은, RFG에서 순환 고리가 형성되지 않음을 보임으로써 증명된다. 여기에서는 RFG에서 순환 고리가 형성될 수 있는 경우를 제시하고, 그러한 경우가 SOS의 제약 조건과 모순됨을 보임으로써 정리 1의 타당성을 증명한다.

경우 1: $T_{Ci} \rightarrow T_{Ud} \rightarrow T_{Uk} \rightarrow T_{Ci}$의 순환 고리가 형성되는 경우(단 $1 \le i \le n$, $1 \le d, k \le m$).

$T_{Uk} \rightarrow T_{Ci}$는 $Pub(T_{Uk})$가 수행된 이후에 T_{Uk}에 의해 기록된 데이타 값을 T_{Ci}가 SS에서 판독한 경우를 묘사한다. 그런데 $T_{Ci} \rightarrow T_{Ud}$ 와 $T_{Ud} \rightarrow T_{Uk}$에 의해서 $Pub(T_{Uk})$는 $Pub(T_{Ci})$가 수행된 이후에만 수행 가능하다. 즉 $Pub(T_{Uk})$는 T_{Ci}가 완료된 후에만 수행 가능하므로 T_{Ci}가 $Pub(T_{Uk})$에 의해 기록된 값을 SS에서 읽어가는 경우는 발생할 수 없다. 즉, 경우 1은 SOS에 따르는 수행 기록에서 존재할 수 없다.

경우 2: $T_{Ui} \rightarrow T_{Cd} \rightarrow T_{Ck} \rightarrow T_{Ui}$의 순환 고리가 형성되는 경우(단 $1 \le d, k \le n$, $1 \le i \le m$).

$T_{Ui} \rightarrow T_{Cd}$는 $Pub(T_{Ui})$가 수행된 이후에 T_{Ui}에 의해 기록된 데이타 값을 T_{Cd}가 SS에서 판독한 경우를 묘사한다. 그런데 $T_{Cd} \rightarrow T_{Ck}$ 와 $T_{Ck} \rightarrow T_{Ui}$에 의해서 $Pub(T_{Ui})$는 $Pub(T_{Cd})$가 수행된 이후에만 수행 가능하다. 즉 $Pub(T_{Ui})$는 T_{Cd}가 완료된 후에만 수행 가능하므로 T_{Cd}가 $Pub(T_{Ui})$에 의해 기록된 값을 SS에서 읽어가는 경우는 발생할 수 없다. 즉, 경우 2는 SOS에 따르는 수행 기록에서 존재할 수 없다.

경우 1, 2는 SOS에 의해 생성된 수행 기록 내에 존재할 수 없다. 즉 SOS에 의해 생성된 모든 수행 기록 내에서는 뷰 일관성이 보장됨을 알 수 있다. □

또한 SOS는 다음의 정리 2에 의해 비밀 경로의 생성 가능성도 효과적으로 차단할 수 있음을 알 수 있다.

정리 2: 낮은 보안 등급을 갖는 트랜잭션은 높은 보안 등급을 갖는 트랜잭션과의 연산 충돌로 인해 지연되거나 철회되지 않는다.

증명: $L(T_C) > L(T_U)$인 트랜잭션 T_C와 T_U가 수행 기록에 포함되어 있다고 가정하자. BL 모형의 제약 조건에 의하여 임의의 트랜잭션이 자신과 다른 보안 등급을 갖는 데이타에 접근할 수 있는 연산은 오직 하향 판독 연산밖에 존재하지 않으므로, T_C와 T_U가 동일한 데이타에 접근하는 경우는 다음의 세 경우 외에는 존재하지 않는다.

경우 1: T_C와 T_U가 x에 대한 하향 판독 연산을 수행하는 경우

이 경우는 $L(T_C) > L(T_U) > L(x)$인 경우로, 두 연산이 모두 판독 연산이므로 서로 충돌을 일으키지 않는다.

경우 2: x에 대하여 T_C가 하향 판독 연산을, T_U가 동급 판독 연산을 수행하는 경우

이 경우는 $L(T_C) > L(T_U) = L(x)$인 경우로, 두 연산이 모두 판독 연산이므로 서로 충돌을 일으키지 않는다. 이 경우, T_C는 SS에서, T_U는 WDB에서 x의 값을 판독한다.

경우 3: x에 대하여 T_C가 하향 판독 연산을, T_U가 동급 기록 연산을 수행하는 경우

이 경우는 $L(T_C) > L(T_U) = L(x)$인 경우로, T_U는 WDB에 x 값을 기록하는 반면, T_C는 SS에서 x 값을 판독하므로 두 트랜잭션간의 연산 충돌은 발생하지 않는다.

경우 1, 2, 3에서 낮은 보안 등급을 갖는 트랜잭션이 높은 보안 등급을 갖는 트랜잭션의 간섭을 받지 않으므로 SOS는 모든 비밀 경로의 생성 가능성을 차단함을 알 수 있다. □

정리 1과 정리 2에 의해 SOS에 의해 생성된 수행 기록은 데이타의 뷰 일관성을 유지함과 동시에 비밀 경로를 효과적으로 차단함을 알 수 있다.

5. 성능 평가

SOS 기법은 변경된 데이타 값이 주기억장치 상의 큐에 머무는 시간을 줄임으로써, SRT-2PL 기법에서 나타나는 큐 관리 부담을 감소시킬 수 있다. 하지만 높은 보안 등급을 갖는 트랜잭션의 하향 판독 연산의 경우, 기록 연산의 수행 즉시 변경된 값을 큐에 저장하는 SRT-2PL 기법에 비해 SOS 기법은 데이타의 신선도가

다소 낮아지게 된다는 단점이 있다. 따라서 큐의 길이와 하향 판독 데이타의 신선도간의 이율배반적인 성격에 대하여 보다 다양한 환경에서의 분석이 필요하다. 한편 SOS 기법은 완료된 데이타의 공표 과정에 관한 제약 조건만을 규정하므로, 동시성 제어에 있어서 기존의 다양한 기법들이 적용될 수 있다는 유연성을 갖는다. 또한 완료된 데이타의 공표 과정에서 정확성 기준으로 뷰 일관성을 채택하는 경우, 충돌 직렬화 가능성을 기준으로 채택하는 경우에 비해 큐의 길이를 더욱 짧게 유지하면서도 하향 판독 연산에 제공되는 데이타의 신선도를 높일 수 있다. 본 장에서는 표 2에 제시된 기법들에 대하여 다양한 환경에서 모의 실험을 수행한 결과를 소개한다. 표 2에는 SRT-2PL과 SOS 시리즈(*SOS_CL_CS, SOS_CL_VS, SOS_TO_CS, SOS_TO_VS*)에 대한 동시성 제어 기법과 정확성 기준이 나타나 있다.

5.1 가정

본 성능 평가는 실제의 시스템이 아닌 가상 환경에서 수행되었으므로, 모의 실험의 수행을 위한 몇 가지 가정의 도입이 필수 불가결하다. 표 2에 나타난 기법들은 모두 동일한 가정 하에서 수행되므로, 아래 가정의 도입은 기법간 성능의 상대적 평가에 영향을 미치지 않는다.

가정 1(닫힌 대기 모형: *Closed Queuing Model*): 본 모의 실험에서는 닫힌 대기 모형의 가정을 도입하였다. 따라서 다중 프로그래밍의 정도에 따라 시스템의 작업량이 정해지며, 특정한 시점에 활성화 상태에 있는 트랜잭션의 수는 항상 일정하게 유지된다.

가정 2(실질 재시작: *Real Restart*): 철회된 모든 트랜잭션은 재시작을 요청하게 되며, 재시작되는 모든 트랜잭션은 최초 실행시 제기하였던 모든 연산을 동일하게 제기하는 것으로 가정한다.

5.2 입력 매개변수

모의 실험에 사용되는 입력 매개변수는 시스템 매개변수, 데이타베이스 매개변수, 트랜잭션 매개변수로 분류된다. 시스템 매개변수는 실험이 수행되는 시스템의 환경을 기술하며, 데이타베이스 매개변수는 데이타베이스의 크기와 특성을 나타낸다. 그리고 트랜잭션 매개변수는 본 모의 실험에서 처리되는 트랜잭션들의 다양한 속성을 나타낸다. 이상의 입력 매개변수들에 대한 간략한 설명 및 그 설정 값이 표 3에 제시되어 있다. 표 3의 값은 기존 연구와의 일관성 유지를 위해 [22]의 결과를 근거로 설정되었다.

데이타베이스 및 트랜잭션 매개변수 값들의 여러 조합에 대하여 모의 실험을 수행함으로써, 제안된 기법에

표 2 모의 실험 대상 프로토콜의 동시성 제어 기법 및 정확성 기준

Protocol	Concurrency Control	Correctness Criteria for Data Publication
SRT-2PL	Conservative 2PL	N / A
SOS_CL_CS	Conservative 2PL	Conflict-Serializability
SOS_CL_VS	Conservative 2PL	View-Consistency
SOS_TO_CS	Strict Timestamp Ordering	Conflict-Serializability
SOS_TO_VS	Strict Timestamp Ordering	View-Consistency

표 3 입력 매개변수 목록 및 설정 값

시스템 매개변수		
num_cpus	2 CPUs	Number of cpus
num_disks	4 Disks	Number of disks
cpu_cc_delay	7.5 milliseconds	CPU time for controlling concurrent execution
obj_io_delay	35 milliseconds	I/O time for accessing an object
obj_cpu_delay	15 milliseconds	CPU time for accessing an object
데이타베이스 및 트랜잭션 매개변수		
db_size	600, 800, ***1000***, 1200, 1400 and 1600 pages	Number of objects in database
num_terms	30, 60, 90, 120, 150, ***200*** and 250 terminals	Number of terminals
tr_size_min	***3*** pages	Size of smallest transaction
tr_size_max	4, 6, 8, ***10***, 15, 20 pages	Size of largest transaction
update_pct	5, 10, 15, 20, ***25***, 30, 40 and 50%	Percentage of update operations
max_c_levels	2, 4, 6, ***8***, 10 and 12 levels	Number of security levels

대한 다양한 환경 하에서의 성능 평가가 가능하다. 각 매개변수들에 대하여 기준으로 사용된 값을 굵게 표시하였으며, 이 값들은 [22]에 근거하여 선정되었다. 예를 들어 데이타베이스의 크기 변화에 따른 성능 평가는 *num_terms* = 200, *tr_size_min* = 3, *tr_size_max* = 10, *update_pct* = 25, *max_c_level* = 8로 설정한 뒤 *db_size*를 600에서 1600까지 변화시켜가며 수행되었다.

본 모의 실험은 Windows XP 운영체제 상에서 Microsoft Visual C++ 6.0 컴파일러와 CSIM version 18[23]을 사용하여 수행되었다.

5.3 성능 지수

여러 입력 매개변수 조합에 대하여 표 2에 소개된 기법들의 성능을 비교하고 그 결과를 분석하기 위해 다음의 성능 지수를 도입한다.

- Average_Response_time(*resp_time*): 각 트랜잭션이 처음 제기되는 시점부터 완료되는 시점까지 소요되는 시간의 평균을 나타낸다(N은 완료된 전체 트랜잭션의 수).

$$\frac{\sum_{i=1}^{N} FinishTime\ (T_i) - StartTime\ (T_i)}{N}$$

- Average_Queue_Length_Ratio(*q_ratio*): 변경된 내용을 스냅샷(SRT-2PL의 경우 이차 복사본)에 반영하기 위해 임시적으로 주기억장치에 보관되는 데이타 큐의 길이가 전체 데이타베이스의 크기에 대해 차지하는 비율을 나타낸다. 큐의 길이는 각 트랜잭션의 완료 시점마다 측정하여 그 값에 대한 평균치를 성능 지수로 사용한다(x_1, x_2, …, x_{db_size}는 각 데이타 항목).

$$\frac{\sum_{i=1}^{db\_size} Average\_queue\_length\_of\_x_i}{db\_size}$$

- Stale_Read_Ratio(*stale_ratio*): 전체 판독 연산 중 최근의 값이 아닌 데이타 값을 판독하는 경우가 발생하는 비율을 나타낸다(N은 완료된 전체 트랜잭션의 수).

$$\frac{\sum_{i=1}^{N} Stale\_read\_count\_of\_T_i}{\sum_{i=1}^{N} Total\_read\_count\_of\_T_i}$$

본 절에서는 SRT-2PL과 SOS 시리즈에 대하여 위의 *resp_time*, *q_ratio*, *stale_ratio*을 성능 지수로 하여 모의 실험을 수행한 결과를 제시한다. 또한 서로 다른 보안 등급을 갖는 트랜잭션간의 형평성 분석과 ESTS 환경 하에서의 성능 비교 결과를 추가적으로 제시한다.

5.4 성능 이득 평가

입력 매개변수의 변화에 따른 각 기법별 성능 지수 변화의 추세가 표 4에 요약되어 있다. 표 4에서 (↑)는 특정 매개변수의 값이 증가할수록 해당 성능지수가 양의 상관관계를 보이며 증가하는 것을, (↓)는 음의 상관관계를 보이며 감소하는 것을 그리고 (↔)는 특별한 상관관계를 보이지 않음을 의미한다. 또한 기본 상태(*default state*) 열에는 각 매개변수를 *num_terms* = 200, *tr_size_min* = 3, *tr_size_max* = 10, *update_pct* = 25, *max_c_level* = 8 로 설정하였을 경우에 나타나는 SOS_TO_VS와 SRT-2PL에 대한 성능 지수들의 실제 관측 값을 표시하였다.

표 4에서 보는 바와 같이 SOS 시리즈는 응답시간과 큐 관리의 부담 측면에서는 SRT-2PL에 비해 성능상의 이득을 보이는 것으로 나타나지만, 데이타 값의 신선도 측면에 있어서는 SRT-2PL에 비해 다소 오래된 데이타 값을 제공한다는 한계점을 가짐을 알 수 있다. 본 절의 이후 부분에서는 모의 실험의 결과에 대한 성능 지수별 분석을 수행한다.

5.4.1 응답시간 분석

SRT-2PL과 마찬가지로 SOS 시리즈의 복사본 갱신 규약은 트랜잭션들의 동시성 제어에 어떠한 영향도 주지 않는다. 즉 SRT-2PL과 SOS 시리즈의 복사본 갱신 알고리즘의 차이는 데이타 값의 신선도와 큐의 길이에

표 4 실험 환경의 변화에 따른 각 기법별 성능 지수의 변화 추세

		default state	db_size	Num_ terms	tr_size_max	update_ pct	max_ c_levels
resp_time	SOS Series	21.12	↔	↑	↑	↑	↔
	SRT 2PL	26.46	↓	↑	↑	↑	↓
q_ratio	SOS Series	0.059	↓	↑	↑	↑	↔
	SRT 2PL	0.661	↓	↑	↑	↑	↓
stale_ratio	SOS Series	0.087	↓	↑	↑	↑	↑
	SRT 2PL	0.051	↓	↑	↑	↑	↑

만 영향을 줄 뿐, 트랜잭션의 응답시간에는 영향을 주지 않는다. 따라서 SRT-2PL, SOS_CL_CS, SOS_CL_VS 세 기법에 있어서 트랜잭션의 평균 응답시간은 항상 동일하게 나타난다. 하지만 SRT-2PL이 보수적 이단계 로킹(Conservative Two-Phase Locking) 규약에 기반하여 복사본의 갱신을 하는 반면, SOS는 동시성 제어 알고리즘의 선택에 유연성을 갖는다. 즉 SOS 기법은 보수적 이단계 로킹 규약을 채택한 SOS_CL_CS, SOS_CL_VS로 구현될 수 있을 뿐 아니라, 엄격한 타임스탬프 순서화 기법(Strict Timestamp Ordering)을 채택하여 SOS_TO_CS, SOS_TO_VS 형태로도 구현 가능하다. 물론 SOS_TO_CS 기법을 채택할 때 나타나는 평균 응답시간과 SOS_TO_VS 기법을 채택할 때 나타나는 평균 응답시간은 서로 동일하다. 동시성 제어 기법으로 엄격한 타임스탬프 기법을 채택함으로써 SRT-2PL에 비해 평균 응답시간 측면에서 성능상의 이득을 얻을 수 있다.

입력 매개변수의 값을 변화시켜가며 수행한 모의 실험의 모든 경우에서 SOS_TO_VS의 응답시간은 SRT-2PL의 응답시간보다 짧게 나타난다. 즉 SOS 기법은 채택 가능한 동시성 제어 기법의 유연성으로 인해 SRT-2PL에 비해 평균 응답시간 면에서 성능상의 이득을 가져올 수 있다. *num_terms*, *update_pct*, *tr_size_max*의 값이 증가할수록 SRT-2PL과 SOS_TO_VS 기법 모두 평균 응답시간이 길게 나타난다. 또한 이들 매개변수의 값이 증가할수록 두 기법의 평균 응답시간의 차이는 더욱 크게 나타난다. 이러한 결과는 트랜잭션간의 경쟁이 치열해질수록 트랜잭션의 평균 응답시간이 길어지고, 이러한 지연 현상은 트랜잭션의 제기 시점에서 필요한 모든 로크를 소유해야 하는 SRT-2PL의 경우에 더 심각하게 나타나기 때문으로 사료된다. 한편 *db_size*와 *max_c_levels* 매개변수에 대해서는 그 값이 증가할수록 SRT-2PL의 평균 응답시간이 짧게 나타나는데, 이러한 현상 역시 해당 매개변수의 값이 증가할수록 트랜잭션간 경쟁이 완화된다는 관점에서 파악될 수 있다. 한편 타임스탬프 순서화 기법을 사용하는 SOS_TO_VS의 경우는 *db_size*와 *max_c_levels* 값의 변화에 의해 평균 응답시간이 큰 영향을 받지 않는 것으로 나타났다. 그림 6은 트랜잭션이 가질 수 있는 최대 연산의 개수인 *tr_size_max*의 변화에 따른 SRT-2PL과 SOS_TO_VS의 평균 응답시간의 변화를 조사한 결과이다.

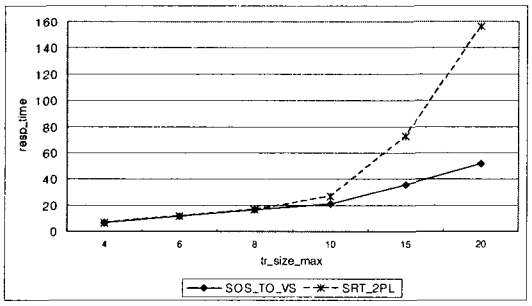


그림 6 트랜잭션의 최대 크기 변화에 따른 평균 응답시간의 변화

5.4.2 큐의 길이 분석

SRT-2PL과 SOS 시리즈는 복사본 갱신 시기 및 방법이 서로 다르므로, 동일한 동시성 제어 알고리즘을 채택하는 경우에도 큐의 길이와 판독 데이타의 신선도에서 서로 차이를 나타내게 된다. 앞서 언급한 바와 같이 SRT-2PL과 SOS시리즈는 하향 판독 연산에 제공되는 복사본의 갱신을 위해 변경된 데이타를 일시적으로 주기억장치에 보관한다. SRT-2PL은 데이타에 대한 기록 연산이 수행된 즉시 변경된 값을 주기억장치 상의 큐에 추가시킨다. 이러한 정책은 하향 판독 연산에 대하여 가능한 한 최근의 데이타 값을 제공할 수 있다. 하지만 기록 연산 수행 직후 그 값을 큐에 추가시키는 정책은, 데이타가 큐에 머무는 시간의 증가를 가져오므로, 큐의 길이가 결과적으로 길어져서 주기억장치의 공간을 필요 이상으로 차지하게 된다는 부작용을 나타낸다. 이와는 달리 SOS 시리즈는 완료된 트랜잭션이 수행한 기록 연산에 대해서만 그 갱신 값을 큐에 보관하기 때문에, SRT-2PL에 비해 상대적으로 큐의 길이를 짧게 유지할 수 있다. 비교 평가에 사용된 성능 지수인 *q_ratio*는 각 트랜잭션의 완료 시점에 모든 데이타에 대한 큐의 길이의 합을 측정한 후, 측정 값들의 평균치를 *db_size*로 나눈 값을 나타낸다.

입력 매개변수의 값을 변화시켜가며 수행한 모의 실험의 모든 경우에서 SOS 시리즈의 *q_ratio*는 SRT-2PL의 *q_ratio*보다 작게 나타난다. 또한 SOS 시리즈에 있어서 동일한 동시성 제어 알고리즘이 사용되는 경우, 정확성 기준으로 뷰 일관성을 채택한 기법이 충돌 직렬화 가능성을 채택한 기법보다 낮은 *q_ratio*를 갖는 것으로 나타났다. *q_ratio*는 트랜잭션간의 경쟁이 치열해질수록 증가하므로, *num_terms*, *tr_size_max*, *update_pct*의 증가는 *q_ratio*의 증가를 가져온다. 또한 같은 이유로 *db_size*와 *max_c_levels*의 증가는 *q_ratio*의 감소를 가져오지만, SOS 시리즈의 경우는 *max_c_levels*의 변화의 영향을 크게 받지 않는 것으로 나타났다. 그림 7은 각 트랜잭션이 수행한 연산 중 기록 연산이 차지하는 비율인 *update_pct*

의 변화에 따른 SRT-2PL과 SOS 시리즈의 *q_ratio*의 변화를 조사한 결과이다.

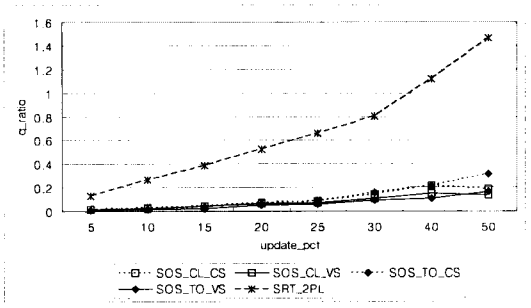


그림 7 기록 연산 비율의 변화에 따른 큐 관리 부담의 변화 분석

5.4.3 판독 데이타의 신선도 분석

SOS 시리즈는 큐의 길이를 짧게 유지하는 대신, 하향 판독 연산시 제공되는 데이타의 신선도가 SRT-2PL에 비해 다소 낮게 나타난다는 부작용이 있다. 즉 데이타의 신선도와 큐의 평균 길이간에는 이율배반적인 상충관계가 존재하게 된다. 데이타의 신선도 분석을 위해 *stale_ratio*라는 성능 지수를 도입하였는데, *stale_ratio*는 각 트랜잭션이 제기한 판독 연산 중 최근의 값이 아닌 오래된 값을 판독한 빈도수의 평균치로 정의된다. 표 4에서 보는 바와 같이, *stale_ratio* 역시 트랜잭션간의 경쟁 정도에 따라 그 값이 증감한다. 즉 *num_terms*, *tr_size_max*, *update_pct*, *max_c_levels*의 값이 증가할수록 *stale_ratio*의 값은 증가하고, *db_size*의 값이 증가할수록 *stale_ratio*의 값은 감소한다. 그림 8은 전체 데이타베이스의 크기인 *db_size* 값의 변화에 따른 SRT-2PL과 SOS 시리즈의 *stale_ratio*의 변화를 조사한 결과이다.

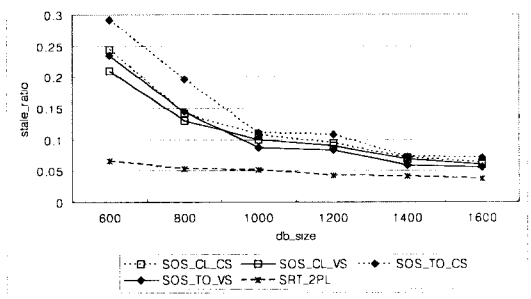


그림 8 데이타베이스의 크기 변화에 따른 데이타의 신선도 변화 분석

5.4.4 형평성 분석

다등급 보안 데이타베이스 시스템 상에서는 서로 다른 보안 등급을 갖는 트랜잭션들이 동시에 수행된다. 따라서 동시성 제어 및 복사본 갱신 알고리즘에 따라 보안 등급별 성능 지수가 서로 다르게 나타날 수 있다. 따라서 어떤 기법에 대해 각 보안 등급별 성능 지수가 큰 차이를 나타낸다면, 이 기법은 보안 등급별 형평성 관점에서 바람직하지 않은 것으로 파악된다. 본 모의 실험에서는 각 보안 등급에 대하여 앞서 언급한 세가지 성능 지수들이 어떠한 형태로 나타나는지에 대한 분석을 수행하였다. 수행 결과 데이타의 신선도 측면에 있어서는 SRT-2PL과 SOS 시리즈 모두 각 보안 등급에 걸쳐 유사한 *stale_ratio* 값을 갖는 것으로 나타났다. 하지만 응답 시간 면에서는 엄격한 타임스탬프 기법을 채택한 SOS_TO_VS는 모든 등급에 걸쳐 거의 유사한 *resp_time* 값을 나타낸 반면, SRT-2PL은 낮은 보안 등급을 갖는 트랜잭션들이 다소 높은 *resp_time* 값을 갖는 것으로 나타났다. 이러한 현상은 높은 보안 등급을 갖는 트랜잭션일수록 하향 판독 연산을 제기할 수 있는 가능성이 높아지고, 하향 판독 연산은 동시성 제어 과정에서 로크를 필요로 하지 않는다는 특성에 기인한 것이다. 즉 SRT-2PL에서 낮은 등급이 제기하는 판독 연산은 판독 로크를 필요로 하는 동급 판독 연산일 가능성이 크므로, 하위 두 등급에 속하는 트랜잭션의 평균 응답시간이 다른 트랜잭션들의 응답시간에 비해 다소 높게 나타난 것으로 사료된다. 따라서 SOS는 보안 등급간의 형평성 측면에서도 SRT-2PL에 비해 바람직한 특성을 가짐을 알 수 있다. 형평성 비교를 위한 각 보안 등급별 평균 응답시간의 분석 결과를 그림 9에 나타내었다(작은 값이 낮은 보안 등급을 의미함).

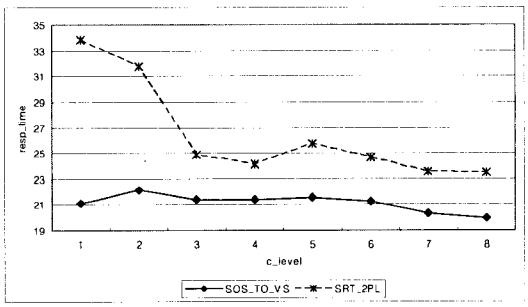


그림 9 각 보안 등급별 평균 응답시간

5.4.5 ESTS 환경 하에서의 성능 분석

지금까지의 모의 실험 결과를 통해 SOS 시리즈가

표 5 H_Tr(OLAP) 및 L_Tr(OLTP) 트랜잭션의 특성에 대한 가정

	c_level	tr_size	update_pct	db_size	num_terms
H_Tr	high	12 pages	10 %	1600 pages	200 terms
L_Tr	low	5 pages	90 %		

SRT-2PL에 비해 평균 응답시간과 큐 관리 부담 측면에서는 바람직한 특성을 갖는 반면, 데이타의 신선도 측면에서는 SRT-2PL이 SOS 시리즈에 비해 바람직한 특성을 나타내는 것을 알 수 있다. 하지만 이러한 비교는 입력 매개변수, 즉 가정하는 환경의 변화에 따라 그 결과가 다소 달라질 수 있다. ESTS 환경 하에서 바람직한 특성을 갖는 보안 동시성 제어 기법을 선택하기 위해서는 입력 매개변수들이 ESTS 환경을 묘사할 수 있도록 설정된 환경에서의 모의 실험이 필요하다. ESTS 환경은 기본적으로 방대한 크기의 데이타베이스를 가정한다. 또한 ESTS 상에서는 분석적인 작업을 수행하는 상위 보안 등급의 트랜잭션과 갱신 작업을 주로 수행하는 하위 보안 등급의 트랜잭션이 동시에 수행된다. 본 모의 실험에서는 보안 등급의 구분을 단순화하여 2 개의 보안 등급만을 사용하고, 상위 등급 트랜잭션의 크기가 하위 등급 트랜잭션의 크기보다 크며 하위 등급 트랜잭션의 경우 상위 등급 트랜잭션에 비해 높은 값의 갱신 비율을 가짐을 가정하였다. 이처럼 서로 다른 특성을 갖는 트랜잭션인 H_Tr과 L_Tr에 대한 가정이 표 5에 요약되어 있다.

평균 응답시간은 동시성 제어 기법의 영향만을 받으므로 *resp_time*에 대한 분석은 SRT-2PL과 SOS_TO_VS 두 기법에 대해서만 수행하였다. 두 기법 모두 H_Tr이 L_Tr에 비해 높은 *resp_time* 값을 갖는 것으로 나타났는데, 이러한 현상은 H_Tr이 L_Tr보다 많은 수의 연산을 제기한다는 가정에 기인하는 것으로 사료된다. 따라서 보다 정확한 평가를 위해 새로운 성능 지수로 각 연산별 평균 응답시간을 의미하는 *resp_time/ops*를 도입하였다. *resp_time/ops*는 각 트랜잭션의 응답시간을 각 트랜잭션이 제기한 연산의 수로 나눈 값의 평균치로 정의된다. 그림 10은 H_Tr과 L_Tr에 대한 *resp_time/ops* 분석 결과를 나타낸다.

그림 10에서 전체 트랜잭션의 평균 응답시간은 SOS_TO_VS가 SRT-2PL보다 낮게 나타나며, L_Tr의 평균 응답시간은 SRT-2PL에서 다소 낮게 나타난다. 하지만 H_Tr의 평균 응답시간은 SRT-2PL에서 매우 높게 나타난다. 즉 SOS_TO_VS는 ESTS 상에서 상위 보안 등급을 갖는 트랜잭션의 응답시간을 단축시킴으로써 SRT-2PL에 비해 전체적인 응답시간 면에서의 성능

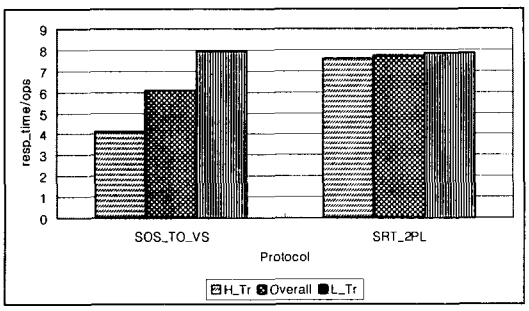


그림 10 ESTS 환경 하에서의 각 연산별 평균 수행시간 비교

상 이득을 가져온 것으로 파악될 수 있다.

그림 11은 H_Tr에 대한 *stale_ratio*의 분석 결과를 나타낸다. 이전의 실험에서 알 수 있듯이 SRT-2PL은 SOS 시리즈에 비해 낮은 *stale_ratio*를 갖는다. 즉 데이타의 신선도 측면에서는 SRT-2PL이 SOS 시리즈에 비해 보다 바람직한 특성을 갖는 것으로 파악될 수 있다. 하지만 SOS 시리즈 중 정확성 기준으로 뷰 일관성을 적용하고 동시성 제어 기법으로 엄격한 타임스탬프 순서화 기법을 채택한 SOS_TO_VS의 경우 SRT-2PL에 비해 크게 높지 않은 *stale_ratio* 값을 가짐을 알 수 있다.

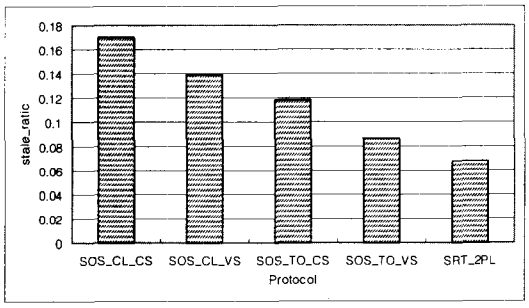


그림 11 ESTS 환경 하에서의 각 데이타 신선도 비교

마지막으로 그림 12는 두 SRT-2PL과 SOS 시리즈에 대한 *q_ratio*의 분석 결과를 나타낸다. *q_ratio*의 경우 SOS 시리즈는 대체적으로 낮은 값을 나타내는 반면, SRT-2PL은 매우 높은 값을 나타냄을 알 수 있다. 특히 SOS_TO_VS와 SRT-2PL의 *q_ratio* 값의 차이가

가장 큰 것으로 나타났다.

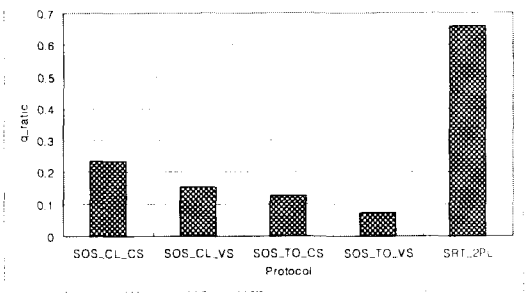


그림 12 ESTS 환경 하에서의 큐 관리 부담 비교

이상의 분석 결과에서 알 수 있듯이, SOS_TO_VS는 ESTS 환경 하에서 동작하기에 매우 바람직한 특성을 가지고 있다. SOS_TO_VS는 SRT-2PL에 비해 상위 보안 등급을 갖는 트랜잭션의 평균 응답시간을 상당 부분 단축시킬 뿐 아니라 큐의 길이도 매우 짧게 유지할 수 있다. 또한 데이타의 신선도 측면에서도 SRT-2PL에 비해 크게 뒤지지 않음으로써 ESTS 환경 하에서 매우 잘 동작할 것으로 사료된다.

6. 결론 및 연구의 한계점

본 논문에서는 전자 주식 매매 시스템에서의 보안 동시성 제어를 위한 프로토콜인 SOS를 제안하였다. SOS는 높은 보안 등급을 갖는 트랜잭션의 하향 판독 연산을 위해 별도의 복사본을 제공하여 상이 등급간 연산의 충돌을 방지함으로써, 비밀 경로의 생성 가능성을 차단함과 동시에 동일한 보안 등급을 갖는 트랜잭션간의 동시성 제어 알고리즘에 대한 유연성을 제공하였다. 이러한 동시성 제어 기법 선택의 유연성으로 인해 SOS는 평균 응답시간 측면에서의 성능 향상을 가져올 수 있을 뿐 아니라, 기존에 제안된 실시간 동시성 제어 알고리즘의 확장 적용이 매우 용이하다는 장점을 갖는다. 또한 SOS는 운용 데이타베이스 외에 단 하나의 복사본만을 유지함으로써 복사본 관리 부담의 최소화를 시도하였으며, 복사본 관리 과정에서 큐의 길이를 짧게 유지함으로써 주기억장치의 부담을 줄일 수 있는 기법을 제시하였다. SOS를 채택할 경우 주기억장치의 부담을 줄일 수 있는 반면, 기존의 기법에 비해 하향 판독 연산에 제공되는 데이타의 신선도는 다소 낮아지는 경향이 나타난다. 하지만 복사본의 갱신 과정에서 하향 판독 연산을 수행하는 트랜잭션은 판독 전용 트랜잭션의 특성을 갖는다는 점을 활용하여, 기존의 기법에 비해 데이타의 신선도가 크게 낮아지는 현상을 방지할 수 있는 방법을 동시에 제시하였다. 추가적으로 복사본의 불필요한 중복 갱신 빈도수를 줄임으로써 큐 관리 부담 및 데이타의 신선도 측면에서 성능의 향상을 가져올 수 있는 기법도 소개하였다. 특히 SOS는 OLAP 트랜잭션과 OLTP 트랜잭션이 동시에 수행되는 환경인 ESTS 환경에서 보다 바람직한 특성을 가짐을 모의실험을 통해 증명하였다.

하지만 보안 등급이 세분화되어 많은 수의 보안 등급이 존재하는 환경에서는, SOS에서 높은 보안 등급을 갖는 트랜잭션일수록 오래된 데이타 값을 판독할 가능성이 여전히 크게 나타한다. 따라서 SOS는 보안 등급이 여러 등급으로 세분화되어 있고 데이타의 신선도가 중요하게 작용하는 환경에서는 바람직한 특성을 갖는다고 보장할 수 없다. 또한 실시간 보안 동시성 제어를 위한 다양한 기법들은 가정하는 응용의 환경에 따라 그 성능 지수가 상이하게 나타날 수 있으므로, ESTS 환경을 보다 구체적으로 투영시킨 상황 하에서의 SOS 프로토콜 및 기존 기법들에 대한 성능 비교 평가가 향후 수행되어야 한다.

참 고 문 헌

[1] Turban, McLean and Wetherbe, *Information Technology for Management*, 2nd ed., John Wiley, 1999.

[2] Bell and LaPadula, "Secure Computer Systems: Mathematical Foundations and Model," *Technical Report*, MITRE Corporation, 1974.

[3] B. W. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, 1973.

[4] J. Goguen and J. Meseguer, "Security Policy and Security Models," *Proceedings of IEEE Symposium on Security and Privacy*, 1982.

[5] J. McDermott and S. Jajodia, "Orange Locking: Channel-Free Database Concurrency Control Via Locking," *Database Security, VI: State and Prospects*, Elsevier Science Publishers, 1993.

[6] P. Ammann and S. Jajodia, "A Timestamp Ordering Algorithm for Secure, Single-Version, Multi-Level Databases", *Proceedings IFIP WG11.3 Working Group on Database Security*, 1991.

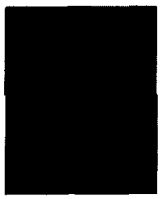
[7] P. Ammann, F. Jaeckle, and S. Jajodia, "A Two Snapshot Algorithm For Concurrency Control In Multi-Level Secure Databases," *Proceedings IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[8] R. Mukkamala and S. H. Son, "A Secure Concurrency Control Protocol for Real-Time Databases," *Proceedings of Annual IFIP WG 11.3*

Conference of Database Security, 1995.
[9] 박찬정, 한희준, 박석, "실시간 보안 데이타베이스 시스템에서 병행수행 제어를 위한 얼림 기법," *정보과학회 논문지*, 2002.
[10] Y. L. Sohn and S. C. Moon, "Confidential Concurrency Control for Secure Transaction Management in Database Systems: C3," *Ph. D. thesis, Korea Advanced Institute of Science and Technology*, 1999.
[11] T. F. keefe and W. T. Tsai, "Multiversion Concurrency Control for Multilevel Secure Database Systems," *Proceedings of IEEE Symposium on Research in Security and Privacy*, 1990.
[12] P. A. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.
[13] P. A. Bernstein and N. Goodman, "Multiversion Concurrency Control - Theory and Algorithms," *ACM Transactions on Database Systems*, 1983.
[14] Q. Ahmed and S. Vrbsky, "Maintaining Security in Firm Real-Time Database Systems," *Proceedings Computer Security Applications Conference*, 1998.
[15] B. Kao, K. Y. Lam, B. Adelberg, R. Cheng, and T. Lee, "Updates and View Maintenance in Soft Real-Time Database Systems," *Proceedings ACM International Conference on Information and Knowledge Management*, 1999.
[16] S. H. Son, "Database Security Issues for Real-Time Electronic Commerce Systems," *IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, 1998.
[17] S. Pal, "A Locking Protocol for MLS Databases Providing Support For Long Transactions," *Database Security IX: Status and Prospects, Chapman&Hall*, 1995.
[18] T. Priebe and G. Pernul, "Towards OLAP Security Design - Survey and Research Issues," *Proceedings of the ACM third international workshop on Data warehousing and OLAP*, 2000.
[19] K. Lam, T. Kuo and L. Shu, "On Using Similarity to Process Transactions in Stock Trading Systems," *Proceedings of the IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, 1998.
[20] K. W. Lam, S. H. Son, V. C. S. Lee and S. L. Hung, "Using Separate Algorithms to Process Read-Only Transactions in Real-Time Systems," *IEEE Real-Time Systems Symposium*, 1998.
[21] B. George and J. Haritsa, "Secure Concurrency Control in Firm Real-Time Database Systems," *International Journal on Distributed and Parallel Databases*, 2000.
[22] R. Agrawal, M. J. Carey and M. Livny, "Concurrency Control Performance Modeling: Alternatives and Implications," *ACM Transactions on Database Systems*, 1987.
[23] Herb Schwetman, "CSIM Users' Guide for use with CSIM Revision 16," *Microelectronics and Computer Technology Corporation*, 1992.

김 남 규
1998년 2월 서울대학교 컴퓨터공학과 졸업(학사). 2000년 2월 한국과학기술원 경영공학과 졸업(석사). 2000년 3월 ~ 현재 한국과학기술원 경영공학과 박사과정 재학중. 관심분야는 데이타베이스 보안, 데이타 웨어하우스, XML

문 송 천
1975년 1월 숭실대학교 저자계산학과 졸업(학사). 1977년 2월 한국과학원 수학물리학과 졸업(석사). 1985년 5월 Illinois 대학교(UIUC) Computer Science 학과 졸업(박사). 1985년~현재 한국과학기술원 교수. 관심분야는 데이타베이스 보안, 데이타베이스 설계, 분산 데이타베이스



손 용 락
1986년 2월 경북대학교 전자공학과 졸업(학사). 1988년 8월 고려대학교 전자및전산학과 졸업(석사). 1999년 8월 한국과학기술원 정보및통신공학과 졸업(박사). 1995년 3월~현재 서경대학교 조교수. 관심분야는 데이타베이스 보안, 데이타 웨어하우스, 분산 데이타베이스