디지털산업정보학회 논문지 제5권 제4호-2009년 12월

보안환경에서 이질형 시스템의 전역 복구 관리 프로토콜

정 현 철*

Global Recovery Management Protocol for Heterogeneous System in Security Environments

Jeong, Hyun Cheol

〈Abstract〉

Many failures are due to incorrectly programmed transactions and data entry errors. System failure causes the loss or corruption of the contents of volatile storage. Although global processing protects data values to detect direct or indirect information effluence, security environments are very important in the recovery management of heterogeneous systems. Although transaction can't control system fault, the restart for the system can cause information effluence by low bandwith. From various faults, it is not easy to maintain the consistency and security of data. This paper proposes recovery management protocols to assure global multilevel secure one-copy quasi-serializability in security environments of heterogeneous systems with replicated data and proves its correctness. The proposed secure protocols guarantee the reliability and security of system when the system fault is happened.

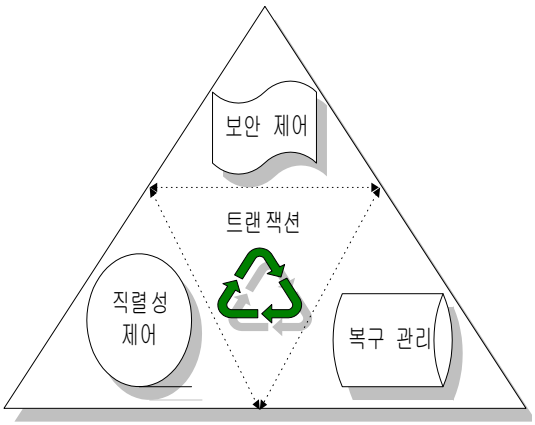
Key Words : Failure, Recovery, Management, Security, Heterogeneous, System

Ⅰ. 서론

많은 결함은 올바르지 않게 프로그램된 트랜잭션과 데이터 입력 에러 때문에 발생한다. 특히 시스템 결함은 휘발성 기억 장치의 내용에 대한 손실이나 파괴를 일으킨다. 이질형 시스템에서 트랜잭션의 직렬성 제어와 복구[1-2]는 투명성 있게 시스템의 사용자들에게 제공되어야 한다. 그런데, 시스템이 복구를 위해 재시작 될 때 트랜잭션들이 시스템 결함을 제어할 수 없다 할지라도 낮은 대역폭으로 인하여 직접 혹은 간접적인 정보 유출이 발생할 수 있다. 따라서, 직렬성 제어와 복구 그리고 보안성은 트랜잭션 처리에서 아주 중요한 문제로 대두되어 왔다. 그런데, 기존 연구에서는 주로 비밀 경로를 통한 불법적인 정보유출을 방지하는 알고리즘[3-5] 개발이 대부분 이다. <그림 1>에서 처럼 보안 환경에서 트랜잭션 관리는 직렬성 제어, 복구 그리고 보안성이라는 세가지 요소가 삼위일체를 이뤄야만 한다. 정보 시스템은 높은 성능이 보장되어야 하므로 빠른 응답성과 신뢰성을 고려해야 하고 데이터 보호를 위하여 데이터를 활용할 수 있

* 광주보건대학 병원전산관리과 부교수

는 권한을 가진 관계자만이 접근할 수 있도록 보안성이 고려되어야 한다. 전역적 복구의 목적은 결함이 발생하더라도 전역트랜잭션이 다단계 보안 1-사본 준직렬성[6]을 유지하는 것이다. 재시작 트랜잭션과 지역트랜잭션이 지역 데이터에 대해서 아무런 제약 없이 함께 수행될 경우 전역적 복구나 지역 수행에서 전역적 준직렬성이 위배될 수 있다.



<그림 1> 트랜잭션 관리의 세 가지 요소

따라서, 본 논문에서는 데이터의 일관성과 보안성을 위하여 이질형 시스템인 멀티데이터베이스의 보안 환경 즉, 전역적 보안 관리자와 각 지역별 보안 관리자를 갖으면서 보안 자치성을 유지하고 전역트랜잭션의 1-사본 준직렬성을 보장하는 이질형 시스템에 대해서, 발생되는 시스템의 결함을 허용할 수 있도록 효율적인 보안 복구 관리 기법을 제안하고 그 프로토콜의 정확성을 검증한다. 본 논문은 2장의 관련 연구에서 결함의 형태와 기존 연구를 분석하며 3장에서는 보안 환경의 복구 처리 시스템과 기존 연구에 대한 문제점을 정의하고 복구 관리를 위한 보안 프로토콜(Secure Recovery Management Protocol, 이하 S-ReMP)를 제안한다. 또 이 알고리즘에 대한 정확성을 증명한다. 4장에서는 결론과 향후 연구 방향을 기술한다.

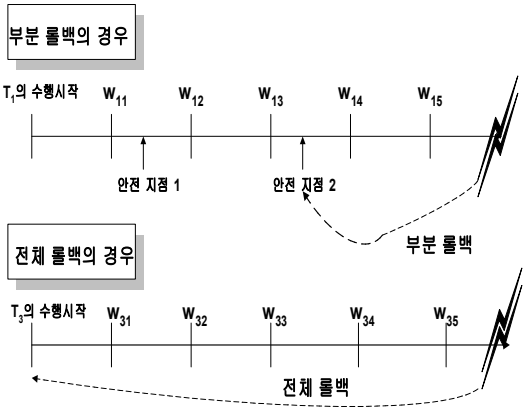
Ⅱ. 관련연구

이 장에서는 발생 가능한 여러 형태의 결함과 기존 연구에서의 복구 관리 메커니즘들을 살펴본다.

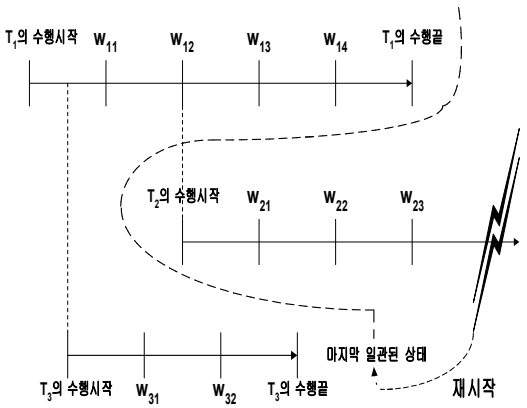
2.1 결함의 형태

결함[7]에는 트랜잭션 결함, 시스템 결함, 통신 결함 그리고 미디어 결함이 있다. 트랜잭션 결함에서 트랜잭션 부분 결함은 트랜잭션의 다중 복구 절차가 요구되며 트랜잭션 부분의 복구 절차가 결함일 수 있고 현재의 트랜잭션은 여전히 수행 중에 있다. 트랜잭션의 전체 결함은 제출된 개별적 트랜잭션 결함이며 시스템은 동작 중이고 다른 트랜잭션도 수행 중임을 의미한다. 시스템 결함에서 모든 트랜잭션은 수행을 멈추며 휘발성 기억 장치에 있는 정보는 손실된다. 그러나 비휘발성 장치에 있는 정보는 안전하다. 미디어 결함에서는 비휘발성 기억장치의 정보는 손실되지만 아카이브 기억 장치에 있는 정보는 안전하다. 이러한 결함들은 모두 탐지가 가능하고 아카이브 기억장치에는 결함이 발생하지 않는다고 가정한다. 결함에 대한 복구 동작을 살펴보면 트랜잭션 부분 결함인 경우는 부분 롤백의 복구 동작이 이뤄지고 트랜잭션의 전체 결함에 대해서는 전체 롤백이 이뤄진다. <그림 2>에서는 부분 롤백과 전체 롤백을 나타낸다. 부분 롤백의 경우는 미리서 정의된 안전 지점까지 롤백한다. 전체 롤백은 트랜잭션의 시작시점으로 롤백한다. 미디어 결함에 대해서는 재저장/롤 전진 동작이 수행되고 시스템의 결함에서는 재시작이 요구된다.

본 논문에서는 시스템 결함에 대한 S-ReMP를 제안한다. 재시작은 가장 복잡한 복구 동작으로 맨 마지막 일관된 상태까지 되돌아가는 것이다. 여기에는 아직 성공적으로 완료되지 않은 트랜잭션을 갱신하는 비수행(UnDo) 후진과 성공적으로 완료된 트랜잭션을 갱신하는 재수행(ReDo) 전진이 있다. 다음 <그림 3>은 재시작을 나타낸다.



<그림 2> 트랜잭션 결함의 복구 동작



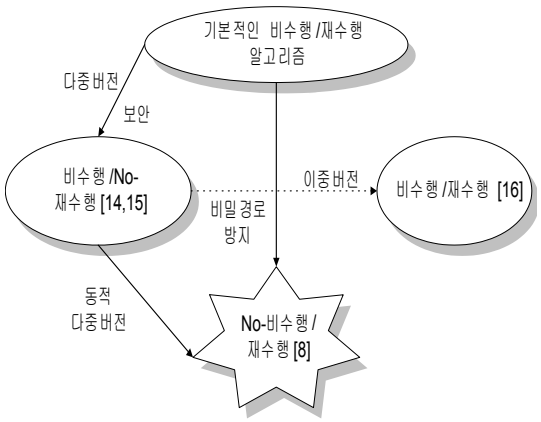
<그림 3> 시스템 결함의 재시작

T_1과 T_3는 성공적으로 완료된 트랜잭션이므로 재수행 동작을 하고 T_2는 성공적으로 완료하기 전에 시스템 결함이 발생하였으므로 비수행 동작을 한다.

2.2 기존 복구 메커니즘

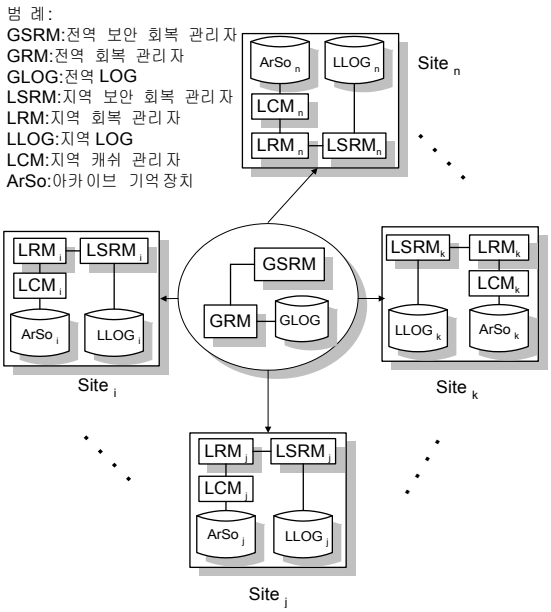
데이터 항목에 대해 단일 값을 갖는 경우 전역적 복구를 위한 재시작의 접근 방법[9]을 살펴보면 다음과 같다. 재시작 트랜잭션이 전역적 일관성을 위해서 재저장 되기 전에 지역트랜잭션이 수행된다면 전역적 일관성을 유지할 수 없다. 재시작 트랜잭션과 지역트랜잭션이 함께 수행되므로 전역적 일관성이 손실되는 것을 방지하기 위하여 [10, 11]은 지역 데이터베이스를 상호 배타적인 두개의 데이터 항목으로 분할하였다. 즉, 전역트랜잭션과 지역트랜잭션에 의해서 각각 갱신될 수 있는 데이터 항목으로 분할하였다. 상호 배타적이기 때문에 전역적으로 갱신 가능한 데이터 항목을 수정하는 트랜잭션은 지역적으로 갱신 가능한 트랜잭션을 판독하도록 허용하지 않았다. 이 방법은 지역과 전역트랜잭션을 아주 엄격하게 제약하고 지역 자치성을 위반한다. [12]은 덜 제약적인 복구 조건을 제안하였다. 지역과 전역트랜잭션이 동일한 데이터 항목을 판독하고 갱신하도록 허용하는 것이다. 지역 데이터베이스가 지역트랜잭션 처리의 중지 연산자와 속계 연산자를 지원한다는 가정을 하여 배타적으로 지역 데이터베이스를 접근하도록 전역적 복구를 고려하였다. 지역 데이터베이스로 배타적으로 접근하기 위하여 이질형 시스템은 합당한 지역에 지역트랜잭션 처리의 중지 연산자를 제출한다. 이것을 양도하기 위해서 이질형 시스템은 지역트랜잭션 처리의 속계 연산자를 발행한다. [13]는 부트랜잭션들 사이에 데이터 종속성이 없다는 가정을 함으로써 지역으로의 배타적 접근이 요구되는 복구 기법을 제안하였다. 또한, 하나의 데이터 항목이 여러 값을 갖는 다중버전 방법에서는 복구를 위하여 엄정함이 요구되지 않는 비수행/No-재수행 알고리즘을 사용한다. [14, 15]은 보안 환경에서 다중버전의 복구를 위한 프로토콜인 비수행/No-재수행 알고리즘을 제안하였다. 비수행/No-재수행 알고리즘은 빈번히 갱신되는 데이터에 대해서 입출력의 횟수가 많은 단점이 있다. 재수행이 불필요하므로 로그에 사후 값을 유지할 필요가 없지만 물리적인 기억 장치로 데이터를 언제 저장시킬 것인가의 시점이 상당히 제약적이다. 그래서, 사용이 빈번한 데이터에 대해서 입출력 부담이 아주 크다. 이에 대하여 [16]의 비수행/재수행 알고리즘을 사용한다면 데이터에 대해 버전 생성 간격과 사후 값의 최근성 유지에 단점을 갖는다. 이는 [16]이 수정된 보안 이중버전 2단계 잠금 기법을 사용하는 시스템의 환경에서 알고리즘을 제안했기 때

문이다. [8]에서는 다중버전과 사후 값이 차지하는 디스크 공간을 줄이고 저장 시점이 [14, 15]보다 완화된 복구 관리로써 No-비수행/재수행 알고리즘을 제안하였다. 빈번한 갱신 횟수에 따른 대량의 버전 생성과 장기 판독 전용 트랜잭션에 의해서 더 이상 접근될 필요가 없는 버전 제거를 고려함으로써 버전 생성과 제거의 연산을 추가 정의하여 장기 판독 전용 트랜잭션에 대한 동적 다중 버전 유지의 복구 관리가 효율적으로 수행되도록 하였다.



<그림 4> 보안환경의 다중버전 복구관리

GT_n 즉, $SubT_{11}$, ... , $SubT_{nn}$ 혹은 전역 보안 관리자가 각 지역의 중복 데이터를 관리하기 위해서 제출했던 ST에 대한 수행 결과를 관리한다. GSRM은 전역 보안 관리자에게서 그 트랜잭션들에게 할당했던 보안등급의 정보를 복구하기 위해 관리한다.



<그림 5> 보안 복구관리 모델

Ⅲ. 복구 관리

이 장에서는 보안 복구 관리 프로토콜인 S-ReMP를 위한 시스템을 서술하고 복구 관리 기법과 그 알고리즘의 정확성을 증명한다.

3.1 복구 관리 시스템

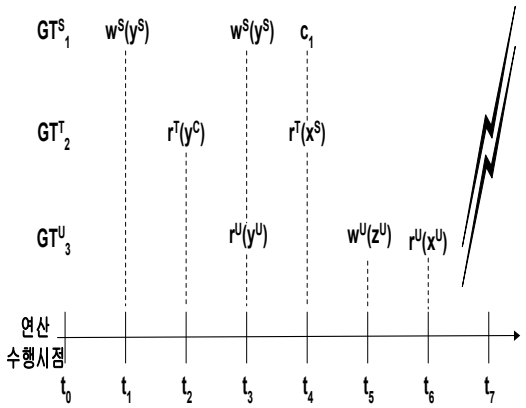
시스템이 결함으로부터 복구되기 위해서 복구 처리 모델에 필요한 부분이 <그림 5>에 나타나 있다. 복구 관리를 위한 모델의 각 기능을 살펴보면 다음과 같다. GRM은 각 지역으로 제출되는 전역트랜잭션인 GT_1.....

이때, GLOG에서는 GRM과 GSRM이 관리했던 정보들의 로그 기록을 유지한다. LRM은 전역 모듈에서 제출되어진 부트랜잭션과 보안 트랜잭션 그리고 그 지역으로 제출된 트랜잭션에 대한 수행 결과를 관리한다. LSRM은 지역 보안 관리자가 각 트랜잭션에 할당했던 보안등급 정보를 유지하며 전역 모듈과 지역 모듈 사이에서 발생하는 보안등급 충돌에 대한 연산을 제어한다. LLOG는 LRM과 LSRM이 수행한 일들의 로그 기록을 갖고 있다. LCM은 각 트랜잭션들의 합당한 보안등급에 따른 연산을 위해 필요한 데이터들을 비휘발성 기억 장치인 아카이브에서 호출하거나 성공적으로 수행이 완료된 결과를 저장시킨다. ArSo는 최종적인 물리적 기억 장치로써 실

제 데이터가 백업 저장되어 있고 이 기억 장치의 정보가 손실된 경우는 결함을 복구 할 수가 없기 때문에 시스템 결함이 발생하더라도 이 기억장치는 안전하게 수행 결과를 유지한다. 또한, 기존 시스템에서 처럼 전역적 복구를 위하여 지역 시스템에서는 지역 트랜잭션을 중지시키거나 속계시킬 수 있도록 하는 해당 연산자를 지원하고 전역 모듈에서 중지와 속계 연산자를 제출할 수 있다.

3.2 문제점 정의

복구 관리를 위한 4 가지 형태에는 비수행/재수행, 비수행/No-재수행, No-비수행/재수행, No-비수행/No-재수행이 있다. 비수행 규칙[17]에서는 캐쉬에 있는 데이터 x값의 장소가 데이터 x의 마지막 완료 값을 가지고 있다면 완료되지 않은 트랜잭션이 갱신 후 값을 기록하기 전에 갱신 전 값이 비휘발성 기억장치에 존재해야 한다. 또, 재수행 규칙에서는 트랜잭션이 끝나기 전에 각 데이터에 대하여 트랜잭션이 성공적으로 완료한 값은 비휘발성 기억장치에 존재해야한다. [14, 15]에서는 비수행/No-재수행 알고리즘을 사용하였다. 시스템 결함은 트랜잭션이 제어할 수 없기 때문에 재시작에 의한 정보 유출은 발생하지 않는다고 가정하였다. 또한, 전역 모듈과 지역 모듈 사이에서 간접적인 정보 유출이 발생할 수 있는 비밀경로 즉, 2 단계 완료 프로토콜에 존재할 수 있는 경우는 대역폭이 낮기 때문에 경로 설정이 의미가 없다고 하였으나 이것은 복구 관리를 위한 보안 측면에서 볼때 근본적인 해결책이라고 할 수 없다. 비수행/No-재수행에서는 재수행을 하지 않으므로 트랜잭션이 완료될 때 캐쉬에 있는 갱신된 값들은 일단 안정성 있는 기억 장치에 먼저 저장되어야 한다. 갱신 트랜잭션들이 자주 접근하는 데이터가 안전성 있는 기억 장치에 많다면 입출력 부담은 더 커지게 된다. <그림 6>에서와 같이 연산 수행시점 t_7에서 시스템 결함이 발생했다면 결함 이전에 전역트랜잭션 GT^S_1이 성공적으로 완료한 값들이 존재하므로 바로 기억 장치에 저장된다.



<그림 6> 비수행/No-재수행

그러나, 전역트랜잭션 GT^T_2와 GT^U_3는 처음 상태로 비수행 되어 다시 수행되어야 한다. 따라서, 저장 및 수행되는 횟수가 증가하여 입출력에 대한 부담이 커진다. 재수행이 없으므로 버퍼 장치에 항상 완료된 값이 존재하면 바로 기록되어 안정성 있는 기억 장치에 유지되고 어떤 트랜잭션이 수행 도중에 결함이 발생하게 되면 그 트랜잭션은 다시 수행되어야 하므로 시스템의 성능은 비효율적인 측면이 존재한다. 또한, 트랜잭션의 직렬성만을 고려하고 보안등급 관계를 생각하지 않을 경우 전역적 보안등급과 각 지역의 보안 자치성에 의한 보안등급 충돌로 인하여 시스템을 재시작하더라도 정보 유출 방지와 트랜잭션의 전역적 일관성에 영향을 끼치게 되므로 시스템의 복구에 문제를 발생시킨다. 시스템의 복구 관리를 위해서 모든 결함은 탐지될 수 있고 비휘발성 기억장치의 백업인 문서 기억장치에서는 결함이 발생하지 않는다고 가정한다. 본 논문에서는 여러 가지 결함 중에서 시스템 결함을 고려한다. 시스템 결함은 모든 트랜잭션을 정지시킨다. 그래서, 수행 중이지만 완료되지 않은 트랜잭션들은 철회될 수 있다. 휘발성 기억장치의 데이터는 손실되나 비휘발성 기억장치에 있는 데이터는 손실되지 않는다. 본 논문에서는 가장 복잡하지만 융통성이 있고 효율적이면서 상업용 시스템에서 가장 광범위하게 사용되

는 비수행/재수행 방법을 이용하여 복구 관리 기법을 제시한다.

3.3 S-ReMP

보안 복구 관리 프로토콜인 S-ReMP는 시스템 결함이 발생하였을 때 결함 이전에 스케쥴링된 트랜잭션이 완료한 값과 동일하게 기억 장치의 값이 유지될 수 있도록 수행하여야 한다. 이를 위해서 S-ReMP에서는 비수행 규칙과 재수행 규칙을 만족시켜야 한다.

S-ReMP에서 사용하는 자료구조로는 성공적 완료 리스트인 Commit_List와 철회되는 트랜잭션을 갖고 있는 Abort_List가 있고 현재 수행 중인 상태의 트랜잭션을 나타내는 Operation_List가 있다. 또한, SecurityConflict_List는 하향 충돌 리스트인 LowConflict_List와 상향 충돌 리스트인 HighConflict_List를 갖으며 보안등급 충돌 상태 정보를 갖는다. SecureSlotLock_List는 지역 정보 사전에 있는 데이터의 각 보안 슬롯을 충돌 없이 접근할 수 있도록 하는 보안 잠금에 대한 리스트이다.

재시작 알고리즘에서 GRM은 전역 로그 파일에서 완료, 철회, 동작에 대한 리스트를 검색하여 수행된 트랜잭션의 비수행 혹은 재수행을 실행한다. GSRM에서는 보안등급 충돌에 대하여 보안 충돌 리스트를 검색하고 하향 충돌과 상향 충돌에 대한 판독 혹은 기록 연산을 복구 처리한다. LRM은 지역 로그 파일을 참조하여 지역 수행을 처리하며 LSRM은 판독과 기록 연산에 대한 데이터의 각 보안 슬롯의 수행을 복구 관리한다.

다음은 GRM, GSRM, LRM 그리고 LSRM에서 전역트랜잭션과 지역트랜잭션 그리고 판독 연산과 기록 연산에 관하여 S-ReMP에 대한 재시작 알고리즘을 서술한다.

```
AAlgorithm S-ReMP_Restart()
1. in GRM, /* 전역 복구 관리 */
  { GRM-1.  Search Commit_List, Abort_List,
            Operation_List in GLOG
    GRM-2. If GT∈{ Operation_List }
/* 전역트랜잭션(GT)의 연산 리스트 */
/*어느 지역에서*/ Then{ If (∃site) = Commit_OK
                        Then ReDo
                        Else UnDo
                   Else Go To Abort_List }
    GRM-3. If GT∈{Abort_List} /* 철회 리스트 */
           Then (∀site) is UnDo /*모든 지역에서*/
           Else Go To Commit_List
    GRM-4. If GT∈{Commit_List} /* 완료 리스트 */
           Then (∀site) is ReDo /* 모든 지역에서 */
    GRM-5. Repeat GRM-1. to GRM-4. for other GT
 }

2. in GSRM, /* 전역 보안 복구 관리 */
   { GSRM-1. Search SecurityConflict_List
     GSRM-2. If GT ∉{{LowConflict_List}∩
/*상하향 충돌 없는 경우*/   {HighConflict_List}}
/*판독연산*/      Then {If Read_Op=Commit_OK
/*Primary_Copy:원본*/      Then ReDo Primary_Copy
/*Secondary_Copy:사본*/    Else UnDo Secondary_
/*AeV: 갱신후 값*/                  Copy(AeV)
/*기록연산*/          If Write_Op=Commit_OK
                      Then ReDo Primary_Copy
                      Else UnDo Primary_Copy
     GSRM-3. If GT∈{LowConflict_List}/*하향충돌*/
/*판독연산*/      Then { If Read_Op=Commit_OK
/*LSM(SL):지역할당등급*/      Then ReDo LSM(SL)
/*GSM(SL):전역할당등급*/      Else UnDo GSM(SL)
/*기록연산*/                If Write_OP=Commit_OK
                              Then ReDo GSM(SL)
                              Else UnDo LSM(SL) }
GSRM-4. If GT∈{HighConflict_List} /*상향충돌*/
/*판독연산*/      Then { If Read_Op=Commit_OK
                      Then ReDo GSM(SL)
```

```
                    Else UnDo LSM(SL)
/*기록연산*/          If Write_OP=Commit_OK
                    Then ReDo LSM(SL)
                    Else UnDo GSM(SL) }
GRM-5. Repeat GSRM-1. to GSRM-4.
       until SecurityConflict_List is Empty  }

3. in LRM, /* 지역 복구 관리 */
  { LRM-1.  Search Commit_List, Abort_List,
            Operation_List in LLOG
    LRM-2. If LT ∈{ Operation_List }
/* 지역트랜잭션(LT)의 연산 리스트 */
/*한 지역에서*/ Then {If (a local site)=Commit_OK
                    Then ReDo
                    Else UnDo
                 Else Go To Abort_List }
    LRM-3. If LT ∈ {Abort_List}/* 철회 리스트 */
            Then (LT's site) is UnDo
            Else Go To Commit_List
    LRM-4. If LT ∈ {Commit_List}/* 완료 리스트 */
            Then (LT's site) is ReDo
    LRM-5. Repeat LRM-1. to LRM-4.
           for other LT.  }

4. in LSRM, /* 지역 보안 복구 관리 */
   { LSRM-1. Search SecureSlotLock_List
     LSRM-2. If LT ∉{SecureSlotLock_List}
/* 보안잠금 리스트 */ Then { If (Read_Op|| Write_Op)
/* 에 없는 경우 판독 기록연산*/   =Commit_OK
                    Then ReDo
                    Else  UnDo  }
LSRM-3. If LT∈{{SecureSlotLock_List}∩
/*하향충돌*/          {LowConflict_List}}
         Then{ If Read_Op=Commit_OK
/*해당슬롯의 갱신전 값*/Then ReDo it's SecureSolt(BeV)
/*하위보안슬롯*/Else UnDo LowLevel SecureSolt(BeV)
/*의 갱신전 값*/  If Write_OP=Commit_OK
/*해당슬롯의 갱신후 값*/Then ReDo it's SecureSolt(AeV)
                    Else UnDo it's SecureSolt(BeV) }
LSRM-4. If LT∈{{SecureSlotLock_List}∩
/*상향충돌*/          {HighConflict_List}}
         Then{ If Read_Op=Commit_OK
               Then ReDo it's SecureSolt(AeV)
               Else UnDo it's SecureSolt(BeV)
/*상위슬롯의*/      If Write_OP=Commit_OK
/*갱신후 값*/   Then ReDo HighLevel SecureSolt(AeV)
/*해당슬롯의 갱신전 값*/Else UnDo it's SecureSolt(BeV) }
LSRM-5. Repeat LSRM-1. to LSRM-4.
        until SecureSlotLock_List is Empty  }
    }
```

3.4 알고리즘의 정확성

전역적 복구 관리 알고리즘의 정확성을 증명하기 위하여 보안 판독 관계를 다음과 같이 정의한다.

【정의 1】 보안 판독 관계

$GT_i \rightarrow GT_j$에 대해서 다음 조건 1)과 2)를 만족할 경우 트랜잭션 GT_j가 GT_i로부터 데이터 항목 x를 보안 판독한다고 한다.

1) 보안등급 A, B에 대해서 $SL(T_i) =_H SL(x)$이고 $SL(T_j) \geq_H SL(T_i)$일 경우 $w^B_i(x^B) \rightarrow r^A_j(w^B_i)$
2) T_i의 기록 연산이 철회되지 않고 T_j의 판독 연산 수행

또한, 전역적 복구가 되기 위해서는 선후관계가 있는 트랜잭션이 모두 성공적으로 완료되어야 한다.

【정의 2】 복구 가능

두 트랜잭션들 사이에 보안 판독 관계가 성립할 경우 $C_i \rightarrow C_j$이면 시스템은 복구 가능하다.

제안된 전역적 복구 관리 프로토콜은 다음에 오는 (정리 1)에 의해서 올바름이 증명된다.

【정리 1】 S-ReMP는 전역적 일관성을 위반하지 않고 재시작이 수행됨으로 인하여 결함으로부터 시스템을 복구할 수 있다.

증명) 시스템 결함으로 재시작이 수행될 때 지역 트랜잭션과 선후 관계로 수행되어 전역적 일관성이 위배된다고 가정하자. 그렇다면 지역 트랜잭션의 수행은 재시작에 영향을 미치게 되고 이는 전역적 일관성을 위반될 수 있게 만든다. 그래서 결함이 발생한 지역에서 재시작과 지역 트랜잭션의 선후 관계에서 문제가 발생해야만 한다. 그러나, 전역적 복구 관리를 위해서 전역 모듈에서는 지역 트랜잭션의 수행 중지와 수행 속계에 대한 연산자를 지역으로 제출할 수 있으며 지역에서는 이것을 지원할 수 있다. 따라서, 재시작이 지역 트랜잭션과의 선후 수행관계로 인한 전역적 일관성 위배는 발생시키지 않는다. 그러므로, 정의 1)과 정의 2)에 의해서 S-ReMP는 전역적 일관성을 유지하면서 시스템을 복구시킬 수 있다.

Ⅳ. 결론

신뢰성 있고 안전한 정보 시스템을 위하여 정보 보안의 중요성은 아주 필수적으로 강조되고 있다. 특히, 시스템 결함으로부터 복구 관리는 일반적으로 보안성을 간과하기 때문에 매우 중요하다. 결함이후에 시스템의 재시작은 정보에 대한 해킹의 대상이 될 수도 있다. 수정된 복구 관리 로그 파일은 인증되지 않은 사용자에 의해 불법적으로 접근되어 사용될 수 있다. 보안 환경에서 시스템의 복구 관리 기법은 시스템에서 발생할 수 있는 예측 불허의 결함에 대해 시스템의 상태를 일관성 있게 유지하고 결함에 견고한 시스템을 구축하기 위해서 반드시 필요하다. 따라서, 본 논문에서는 데이터가 중복되어 있는 이질형 시스템의 보안 환경에서 복구 관리를 위한 보안 프로토콜을 제시하고 그 정확성을 증명하였다. 보안 복구 관리 프로토콜인 S-ReMP는 전역트랜잭션들이 전역적으로 일관성을 갖도록 제어하면서 정보 유출 방지에 대한 보안 정책에 위배되지 않게 시스템의 결함에 대하여 안전하게 복구될 수 있다. 보안정책을 고려한 보안 판독관계에 따라서 수행된 트랜잭션들이 전역적 선행 관계를 유지하면서 성공적으로 완료되어 복구되었을 때 시스템의 신뢰성은 보장될 수 있다. 향후 방향으로는 복구 관리자와 캐쉬 관리자사이에 인터페이스를 위한 연산자의 간소화와 디스크 관리자에 대한 비밀 기억 장치 경로 설정의 방지책이 필요하며 시뮬레이션에 의한 시스템의 성능 비교와 분석 및 향상이 고려될 필요성이 있다.

참고문헌

[1] C. P. Pfleeger, *Security in Computing*, Prentice Hall, 1989, pp. 249-250.

[2] S. Castano, *DatabaseSecurity*, Addison-Wesley, 1994, pp. 82-96.

[3] O. Costich, "Transaction Processing Using an Untrusted Scheduler in a Multilevel Datab- ase with Replicated Architecture," Database Security V: Status and Prospects, C. E. Landw- ehr and S. Jajodia(Editors), Elsevier Science Publishers B. V. (North-Holland) IFIP, 1992, pp. 173-189.

[4] M. H. Kang, O. Costich, and J. N. Froscher, "A Practical Transaction Model and Untrusted Transaction Manager for a Multilevel-Secure Database System," Database Security VI: Status and Prospects (A-21), B. M. Thuraisingham and

C. E. Landwehr (Editors), Elsevier Science Publishers B. V. (North-Holland) IFIP, 1993, pp. 285-300.

[5] O. Costich, "Maintaining Multilevel Transaction Atomicity in MLS Database Systems with Replicated Architecture," Database Security, VII(A-47), T. F. Keefe and C. E. Landwehr(Editors), Elsevier Science B. V. (North-Holland) IFIP, 1994, pp. 329-355.

[6] H. C. Jeong, "Transaction Serializability with Secu- rity in Heterogeneous Medical Database Systems," Journal Korean Society of Medical Inform- atics, 1999, pp. 73-86.

[7] P. C. Kim, "OLTP: Introduction," ETRI Database Section, SIGDB Tutorial on Transaction-Oriented Recovery in Centralized Database systems, July 1994, pp. 5-9.

[8] H. C. Jeong, "Multilevel Secure Recovery Management of Medical Database in Hospital Information System," Journal of Korean Society of Medical Informatics, June 2000, pp. 33-52.

[9] B. H. Hwang, "Three-level Transaction Scheduling in Multidatabase Systems," Kaist Ph. D. Thesis, 1994, pp. 41-42.

[10] Y. Breitbart, A. Silberschatz, G. Thompson, "Reliable Transaction management in Multidat- abase System," ACM SIGMOD, 1990, pp. 215- 224.

[11] A. Wolski, J. Veijalainen, "2PC Agent Method: Achieving Serializability in Presence of Failures in a Heterogeneous Multidatabase," PARBASE-90, 1990.

[12] D. Geogkopoulos, "Multidatabase Recoverability and Recovery," 1991, pp. 348-355.

[13] K. Barker, "Transaction Management on Multidata- base Systems," Ph. D. Thesis, University of Alb- erta, 1990.

[14] I. E. Kang and T. F. Keefe, "Recovery Management for Multilevel Secure Database Systems," Proc. of the IFIP WG 11. 3 Workshop on Database Security, 1992, pp. 225-247.

[15] I. E. Kang and T. F. Keefe, "Recovery Management for Multilevel Secure Database Systems," Technical Report TR-92-103, Dept. of Electrical and Computer Engineering, Pennsylvanial State University, 1992.

[16] Y. L. Jang and S. Park, "Recovery Management Using a Secure Undo/Redo Algorithm Supporting a Multilevel Secure DBMS," Korea Information Science Society, Special Interest Group on Databases Winter Conference, February 1997.

[17] Berrnstein, *Concurrency Control & Recovery in Database Systems*, Addison-Wesley, 1987.

■ 저자소개 ■

정 현 철
Jeong, Hyun Cheol

1998년 3월~현재 광주보건대학 병원전산관리과 교수
1997년 8월 전남대학교 전산학과(이학박사)
1989년 2월 중앙대학교 전산학과(이학석사)
1987년 2월 조선대학교 전산통계학과(이학사)

관심분야 : 정보보안, 의료정보, 유비쿼터스
E-mail : hcjeong@ghc.ac.kr

논문접수일 : 2009년 11월 1일
수 정 일 : 2009년 11월 17일
게재확정일 : 2009년 11월 22일