보안 객체-관련성 모델에서 응용 트랜잭션 모델링

○
*심 갑 식, **조 일 래, *노 봉 남
* 전남대학교 전산과
** 순천공업전문대 전자계산과

Application Transaction Modeling in Secure Object-Relationship Model

*G.S. Sim, **I.R. Cho, *B.N. Noh
* Dept. of Computer Science
Chonnam National University
** Dept. of Computer Science
Sooncheon Junior Colleage

< 요 약 >

지금까지는 데이타의 무결성 보장을 위해 데이타베이스의 개념적 설계단계에서 트랜잭션 모델링은 데이타의 무결성 성질만을 언급하였고 보안 성질은 표현하지 못하였다. 또한, 데이타 모델에서 데이타의 동적 성질을 모델링하는 트랜잭션은 데이타의 무결성을 완전히 보장하기 어려웠다. 응용영역의 효과적인 분석과 설계를 위해서는 데이타 모델링에서 객체, 속성, 관련성 등과 같은 정적 성질 뿐만 아니라 데이타의 무결성과 보안성을 보장하는 동적 성질의 모델링이 필요하다.

본 논문은 데이타의 무결성과 보안성 정보를 표현하는 보안 객체 관련성 모델(Secure Object RElationship Model : SOREM)에서 보안성이 첨가되어 자동으로 생성되는 기본 연산을 바탕으로 응용 트랜잭션을 모델링하는 방법을 제시한다. 또한, 보안 트랜잭션 모델링 과정을 정확하고 용이하게 수행하기 위한 보안 트랜잭션 정의 언어(Secure Transaction Definition Language : STDL)를 사용하여 보안 트랜잭션 모델링 자동화 도구를 X 윈도우 환경에서 설계 및 구현하였다.

1. 서 론

지금까지 데이타베이스의 개념적 스키마 설계도구는 이해하기 쉽고 단순한 Chen의 개체-관련성 모델(Entity-Relationship Model)이 1976년에 처음으로 제안되어 현재는 논리적 데이타 모델링을 위하여 가장 많이 사용되고 있다[3]. 이런 유용성에도 불구하고 개체-관련성 모델은 정적 스키마의 표현만 가능하고 데이타에 대한 동적 연산을 모델링할 수 있는 방법이 어려웠다[1]. 개체-관련성 모델의 이해성, 확장성, 성능을 개선한 객체 모델링 기법(Object Modeling Technique : OMT)이 나오게 되었는데 객체 모델링 기법에서는 객체에 대한 속성과 연산을 모두 하나의 객체로 모델링하여 객체 중심의 개념을 강조한다[1,6]. 그러나 객체 모델링 기법에서도 데이타의 모델링된 동적 성질을 구현하는 방법을 제시하지 않고 의미적 무결성을 보장하는 트랜잭션에 대한 언급을 하지 않았다.

데이타베이스 시스템은 데이타가 가지고 있는 의미(semantics)에 관한 지식을 가져야 한다. 데이타의 의미는 데이타 객체의 구조나 정적 내용 뿐만 아니라 동적(dynamic) 특성들과 그러한 객체들 사이의 관련성을 포함하고 있다. 데이타베이스 설계 과정에서 응용 영역(application domain)의 데이타 의미를 정확하게 표현하는 것은 매우 중요하다.

기존의 데이타 모델들은 주로 데이타의 무결성(integrity)만을 고려하여 정확한 개념적(conceptual) 스키마를 표현하는데 중점을 두었다. 그러나 개념적 스키마는 데이타의 무결성 뿐만 아니라 보안성(security)을 함께 표현하여야 사용자 요구를 충족시키고 훨씬 안전한 데이타베이스 시스템을 구축할 수 있다. 데이타 무결성은 데이타에 대한 변화가 생겼을 때 고의적이든 부주의든 데이타가 틀리지 않게 하는 것을 보증하는 것이며, 보안성은 무권한자로부터 중요한 정보의 유출을 방지하는 것이다.

최근들어 컴퓨터 보안에 관한 보호 메카니즘의 중요성이 증가함에 따라 데이타 모델링에 보안성을 포함시키려는 시도가 관심거리로 대두되고 있다. 특히, 고도의 안전성과 보안성을 유지하기 위해서 다단계(multilevel) 보안을 취급할 필요성이 증가되어 여러 형태의 다단계 데이타베이스들이 소개되고 있다[4,7,8].

본 논문은 데이타베이스의 개념적 설계 도구인 보안 객체 모델에서 객체 클라스들 사이에 존재하는 데이타 보안성과 무결성 제약조건을 동적으로 모델링할 수 있는 보안 트랜잭션 모델링 방법론을 제시하고, 트랜잭션 모델링 과정이 일관성있고 효율적이기 위한 보안 트랜잭션 정의 언어를 제안하고 이를 바탕으로 자동화 도구를 설계하고 구현하였다.

본 논문의 구성은 2장에서 관련연구와 배경을 설명하고, 3장에서 보안 객체-관련성 모델의 기본개념들과 표기법을 다룬다. 4장에서는 보안 트랜잭션 모델링의 개념과 방법론을 서술한다. 5장에서 보안 트랜잭션 정의 도구의 구성과 구현환경 그리고 사용 예를 설명하며, 마지막 6장에서는 결론과 추후 연구방향을 언급하였다.

2. 관련 연구

데이타베이스의 정적 스키마의 모델링에 관한 연구들은 대부분 Chen의 개체-관련성 모델에 기초를 두고있다[3]. 그러나 개체-관련성 모델은 단지 데이타베이스 구조의 설계에 대해서만 설명하고 프로그래밍의 기능에 대해서는 표현할 수 없다. 즉, 정적 스키마의 표현만 가능하고 이들 데이타에 대한 동적 성질을 모델링할 수 있는 방법은 고려하지 못하였다.

객체 모델링 기법은 개체-관련성 모델의 변형으로 객체에 대한 속성과 연산을 모두 하나의 객체로 모델링하여 객체 중심의 개념을 강조한다[1,6,7]. 그러나 객체 모델링 기법에서는 모델링된 동적 성질을 구현하는 방법을 제시하지 않고, 데이타베이스의 의미적 무결성을 보장하는 트랜잭션 모델링에 대한 언급을 하지 않았다.

그러나 본 자동화 도구는 편리한 사용자 인터페이스를 통하여 보안 객체 그림표를 입력하면 데이타의 의미적 무결성과 보안성을 만족하는 기본 트랜잭션을 자동으로 생성한다. 사용자는 기본 연산과 트랜잭션 정의 언어를 사용하여 시스템에서 제공하는 프레임(frame)으로 응용 트랜잭션을 정의할 수 있다.

2.1 ACM/PCM

능동/수동 요소 모델링(Active/Passive Component Modeling : ACM/PCM)은 데이타의 추상화, 제어(control) 추상화, 그리고 절차(procedural) 추상화 등을 이용하여 데이타베이스의 구조와 트랜잭션을 통합한 데이타베이스 설계 방법론이다[5]. 그러나 데이타베이스의 개념적 설계 단계에서 데이타의 동적 성질을 표현할 수 없고 다단계 데이타베이스 시스템의 보안성 또한 보장할 수 없으므로 데이타베이스의 개념적 설계방법으로 부족하다.

2.2 Gambit

Gambit는 확장된 개체-관련성 데이타 모델(extended entity-relationship data model)에 기초하여 데이타 구조, 무결성 제약조건 그리고 트랜잭션을 정의할 수 있

는 자동화 도구이다[2]. Gambit에서 제약조건은 제 1차 술어해석(first order predicate calculus)에 기초한 데이타베이스 프로그래밍 언어인 Modula/R을 사용하여 표현한다. Gambit의 가장 큰 장점은 객체 블럭 다이어그램에서 전파경로를 나타내는 트랜잭션을 자동적으로 생성해 내는 것이다.

Gambit에서 정적 스키마와 트랜잭션을 분류하기 위한 데이타 추상화로는 일반화와 세분화의 개념을 사용한다. 또한 트랜잭션의 설계에 있어서 사용자에게 대화식의 환경을 제공한다. Gambit는 정적 스키마 구조를 모델링하기 위해 확장된 개체-관련성의 정의와 의미적 무결성 제약조건을 표현하기 위한 데이타베이스 프로그래밍 언어를 지원한다. 그러나 다단계 데이타베이스 시스템의 보안성을 지원하고 있지 않다.

2.3 SDMS

데이타 객체가 의미를 가질려면 구조의 정적 그리고 동적 원리, 데이타 객체의 내용, 데이타 객체들간의 관련성이 있어야 한다[7]. SDMS(Semantic Data Model for Security)는 데이타 의미를 모델링하는데 풍부한 구조를 제공하고 데이타의 구조적 원리를 지원할 뿐만 아니라 관계된 무결성 제약조건 등을 표현한다. 또한, 데이타베이스 요구는 전통적인 접근 제어의 범위를 넘어 보호되어야 할 데이타 사이의 관련성을 포함한다. 이런 과정은 데이타 모델링 개념을 확장하여 데이타의 보안 원리를 지원하고 있다.

이 모델은 지금까지 데이타의 무결성만 언급했던 기존의 모델들과는 달리 보안 특징을 함께 표현하는 확장된 데이타 모델이다. 이해하기 쉬운 표현 기법을 제공함으로써 데이타 보안 요구 분석과 데이타베이스 설계에 도움을 줄 뿐 아니라 응용분야에 대한 모든 범위의 보안의미를 표현할 수 있다. 그러나 이 모델에서는 트랜잭션 스키마를 지원하지 못한다.

3. 보안 객체-관련성 모델

제안된 객체-관련성 모델은 객체 지향 데이타 모델, 개체-관련성 모델 그리고 관계형 모델 등의 혼합형이다. 보안 객체-관련성 모델을 구성하는 기본적인 구성요소는 객체, 클라스, 속성, 연산, 트랜잭션, 관련성 등이 있다.

본 논문에서 연산(operation)은 객체에 가해지는 함수 또는 변환(transformation)으로 단위연산(unit operation), 기본연산(primitive operation) 그리고 응용연산(application) 등이 있다. 단위연산은 include, exclude, get, replace 그리고 take와 같은 속성 수준에서 이루어지는 연산이며, 기본연산은 객체 클라스에 속하는 의미적 무결성을 만족하도록 객체 인스턴스 수준에서 이루어진다. 특히, 기본 연산들 중에서 insert, delete, update와 같은 갱신 연산과 select, project와 같은 검색 연산은 데이타베이스의 무결성과 보안성 보장을 위해 트랜잭션으로 자동 생성된다. 응용 연산은 기본 연산과 순차(sequence), 선택(selection) 그리고 반복(repeatition) 등의 결합규칙을 이용하여 사용자가 정의하는 연산이다. 본 논문에서 트랜잭션은 다음과 같이 정의한다.

[정 의] 트랜잭션

> 객체 클라스에 정의된 연산의 수행결과가 보안 객체 데이타 모델에 표현된 데이타의 의미적 무결성 제약 조건과 보안성 성질을 항상 만족할 때 이 연산을 트랜잭션이라 정의한다.

보안 객체-관련성 모델에서 연관성은 대응수(cardinality)와 존재성(existence)의 관점에서 정의된다. 대응수는 연관성에서 객체의 사상관계를 나타내는 것으로 어떤 객체 클라스가 그와 관련된 객체 클라스의 한 객체에 대해 얼마만큼 대응하는가를 보여주며, 각 객체 클라스의 대응수는 일(one) 또는 다(many)의 값을 갖는다. 존재성은 한 객체의 존재가 다른 객체의 존재에 종속되는 관계를 나타낸다. 즉, 객체

의 존재 종속성을 표현하는 방법이다. 존재성은 필수(mandatory)와 선택(optional)으로 구분한다. 존재성과 대응수의 표현은 그림1의 표기법을 따른다.

대응수 \ 존재성	필 수	선 택
다(many)	C1 ●— C2	C1 ⊙— C2
일(one)	C1 — C2	C1 ∅— 2

(그림1) 연관성에서 대응수와 존재성

보안 객체-관련성 모델에서 보안등급은 U, C, S, TS가 있다. 보안 객체 데이타 모델에서 객체의 보안수준은 [L, H]로 표현한다. 여기서 L은 객체가 가질 수 있는 하계값이고 H는 상계값을 나타낸다. 또한 보안수준이 [H]와 같이 단일 값만을 갖을 때는 상계값과 하계값이 같은 경우이다.

주체 S가 객체 O를 판독(read)하기 위한 조건은 첫째 주체 S의 보안수준은 객체 O의 보안수준보다 높거나 같아야 하고, 둘째 S의 범주가 객체 O를 포함하는 것이다. 주체 S가 객체 O를 기록(write)하기 위한 조건은 첫째 주체 S의 보안수준은 객체 O의 보안 수준과 같아야 하고, 둘째 주체 S의 범주는 객체 O를 포함해야 한다.

보안성 의미의 표현은 그림2와 같이 속성, 클라스, 연관성 등에 굵은 실선으로 나타낸다. 이들의 보안등급 표시는 클라스와 속성일 때는 표현된 클라스나 속성의 인접한 곳에 표기하고, 연관성의 경우에는 그 보안등급을 굵은 실선의 중앙에 표시한다.

종 류	보안 미분류	보 안 분 류
클라스		[S]
속 성		[U,S]
관련성		[TS]

(그림2) 보안성 표현

4. 기본 트랜잭션 모델링

다중 인스턴시에이션 인스턴스나 속성들은 높은 보안등급에서 보일 수 있으므로 기본연산(검색, 삽입, 삭제, 갱신)에서 어느 인스턴시에이션들을 원하는가를 명시해야 한다. 이것에 대해서는 몇가지 방법이 있다. 첫째, 모든 인스턴시에이션들을 선택하는 방법이다. 둘째, 가장 높은 보안등급을 가진 인스턴시에이션을 선택하는 방법이다. 세째, 가장 최근의 인스턴시에이션을 선택하는 방법이다. 세째 방법에서는 릴레이션 스키마가 타임스탬프(timestamp)인 속성을 포함할 때에 가능하다. 본 논문에서는 일반적인 보안성을 지원하는 두번째 방법으로 트랜잭션을 모델링한다. SOREM에서는 그래픽 그림표를 입력하여 자동으로 기본 보안 트랜잭션을 생성한다[9].

4.1 검색 트랜잭션

주어진 주키값의 인스턴스를 사용자 S가 검색하고자 할 때, 먼저 사용자의 보안등급이 객체 클라스 보안등급보다 같거나 높아야 한다. 그렇지 않으면 사용자가 접근할 수 없으므로 검색은 실패한다. 마찬가지로, 객체 클라스의 한 속성의 모든 값을 검색할 때도 먼저 사용자 S의 보안등급이 클라스 보안등급보다 같거나 높아야 하고 또한 각 속성값의 보안등급을 지배해야 한다. 그렇지 않은 경우는 실패다.

4.2 삽입 트랜잭션

하나의 인스턴스를 객체 클라스에 삽입할 때, 사용자가 객체 클라스를 접근할 수 있어야 하므로 사용자의 보안등급이 객체 클라스의 보안등급의 하계값 이상이어야 한다. 그리고 인스턴스 다중 인스턴시에이션이나 속성 다중 인스턴시에이션이 발생하거나 삽입하고자 하는 인스턴스의 주키값이 삽입할 객체 인스턴스 집합에 존재하지 않을 때에만 삽입할 수 있다.

그러나 다단계 참조 무결성을 만족시켜야 하므로 삽입할 인스턴스에 외래키 속성이 있는 경우와 없는 경우로 나누어서 처리해야 한다. 먼저 외래키가 포함되는 경우에는 외래키값이 널일 때는 다른 인스턴스를 참조하지 않는 경우가 되므로 다단계 참조 무결성 제약조건에 제약받지 않는다. 그래서 삽입할 인스턴스를 그대로 삽입하면 된다. 그리고 외래키값이 널이 아니고 참조될 인스턴스가 다른 객체 클라스에 존재할 때도 외래키값이 널인 경우와 같이 하면 된다. 참조될 인스턴스가 다른 객체 클라스에 존재하지 않을 경우는 다단계 참조 무결성을 만족하도록 대응 객체 클라스에 먼저 삽입한 후에 삽입하고자 하는 객체 클라스에 포함시키면 된다.

외래키가 포함되지 않는 경우는 객체 클라스에 먼저 인스턴스를 포함시키고 그 객체 클라스가 참여하는 관련성의 대응수와 존재성에 따라 대응 객체 클라스들에게도 대응 인스턴스를 포함시켜야 한다.

4.3 삭제 트랜잭션

하나의 인스턴스를 객체 클라스에서 제거할 때 사용자가 객체 클라스를 접근할 수 있어야 하고 또 보안성질을 만족해야 하므로 사용자의 보안등급이 객체 클라스의 보안등급과 같아야 한다. 또 다단계 참조 무결성을 만족시키기 위하여 제거될 인스턴스에 외래키 속성이 있는 경우와 없는 경우를 생각할 수 있다.

먼저 제거될 객체 클라스의 인스턴스에 외래키가 포함되는 경우 참조되는 객체 클라스의 등급이 사용자의 등급과 같을 때 해당 인스턴스를 제거하고 대응수와 존재성에 따라 대응 인스턴스를 제거한다. 외래키가 포함되지 않는 경우는 사용자의 등급이 참조할 객체 클라스의 등급과 같고, 인스턴스에서 제거될 주키 값과 참조하는 객체 클라스의 주키 값이 같으면 참조되는 인스턴스를 제거하고 대응수와 존재성에 따라 대응 인스턴스를 제거한다.

4.4 갱신 트랜잭션

사용자의 등급이 갱신될 객체 클라스의 주키값의 등급과 같고, 또 갱신될 객체 클라스의 속성값의 등급과 같게 되어 속성 다중 인스턴시에이션이 발생하면 그 속성값을 갱신하고 그렇지 않으면 삽입 연산을 수행한다.

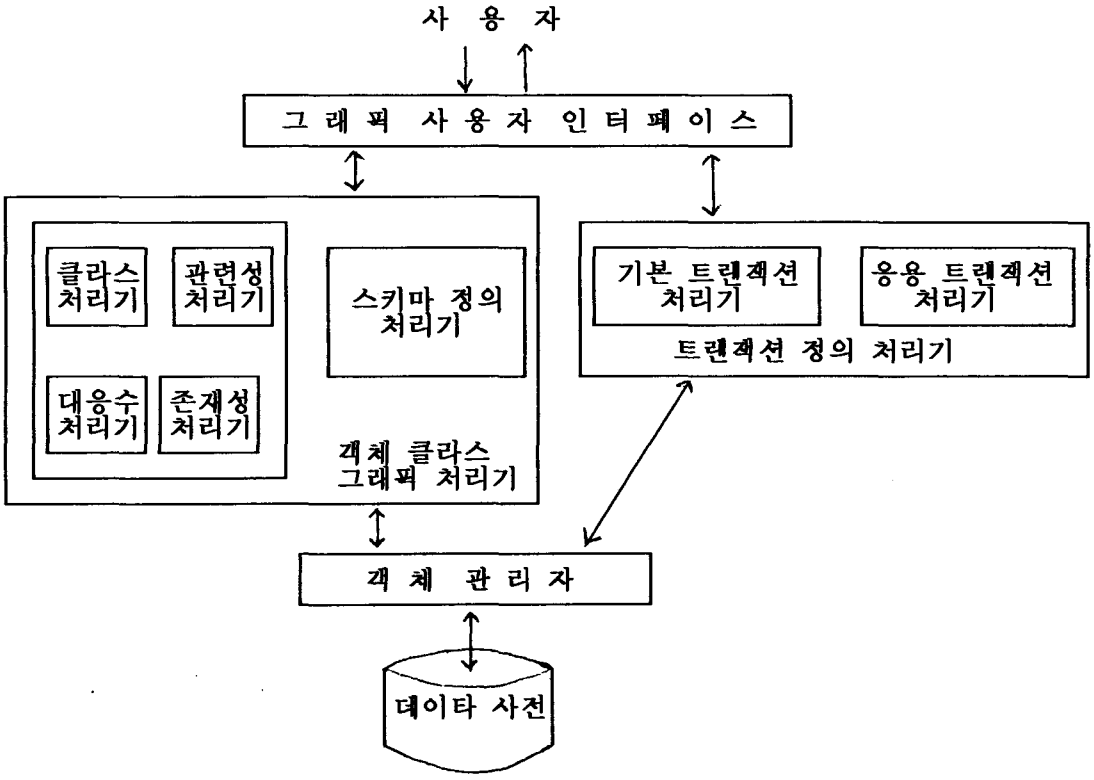
5. 보안 트랜잭션 모델링 도구

보안 트랜잭션 정의 언어(STDL : Secure Transaction Definition Language)는 데이타베이스 설계시 개념적 데이타의 정의 단계에서 데이타의 무결성과 보안성을 보장하기 위하여 사용한다. 사용자는 STDL을 사용하여 개념적 설계 단계에서 응용 트랜잭션을 정의하여 데이타베이스 무결성을 보장할 수 있다.

5.1 시스템 구성

시스템은 스키마 정의 모듈, 트랜잭션 정의, 객체 관리자 모듈 그리고 그래픽 사용자 인터페이스로 구성되어 있다. 그래픽 사용자 인터페이스는 정적 스키마와 보안 트랜잭션을 정의하기 위한 SODE(Secure Object Diagram Editor)라는 그래픽 편집기를 제공한다. 스키마 정의 모듈은 SODE에서 사용자가 입력한 보안 객체 다이어그램을 보안 관계 스키마로 자동 생성한다.

트랜잭션 정의 모듈은 SODE를 통하여 받아들인 정보와 스키마 정의 모듈에서 자동으로 생성된 보안 관계 스키마 정보를 이용하여 의미적 무결성과 보안성을 만족하는 기본 트랜잭션을 자동으로 생성한다. 사용자가 해당 클라스에서 기본 트랜잭션을 호출하여 STAL의 구조에 따라 응용 트랜잭션을 입력하면 시스템은 이를 파싱하여 데이타 사전에 저장한다. 전체 시스템 구성도는 그림3과 같다.



(그림3) 시스템의 구성도

5.2 트랜잭션 정의 모듈

보안 트랜잭션 정의는 데이타 모델링에서 정적 스키마 정의에 대응되는 것으로 객체에 대한 동적인 성질을 표현한다. 트랜잭션 정의는 아래와 같다.

TRANSACTION : Secure Transaction name Class_name
FROM <Related_Object>
BEGIN
  IF <Precondition>
      THEN <Action part>
      ELSE <Exception_handling>
  IF <Postcondition>
      THEN COMMIT
      ELSE ABORT
END

<Precondition>은 NOT, AND, OR로 연결된 술어들로 구성되고 그 결과는 참 또는 거짓의 논리값을 만들어 낸다. 이 결과값에 따라 <Action_part> 또는 <Exception_handling> 중의 하나가 수행된다. <Action_part>는 객체 클라스 class_name에 정의된 기본 또는 응용 트랜잭션 중의 하나이다. <Exception_handling>은 오류 처리 기능을 수행하는 프로시듀어로서 이것이 수행된 후 트랜잭션은 종료된다.

<Postcondition>은 연산이 수행된 후에 무결성 조건을 조사하기 위한 것으로 <Precondition>과 같은 구조를 가지고 있다. 만일 <Postconditon>의 결과가 참이면 트랜잭션의 수행 결과가 데이타베이스에 반영되고 거짓이면 그 결과는 철회되어야 한다. <Related_Object>는 트랜잭션과 관련된 클라스 객체이며 사용자는 작은 윈도우에 나타난 클라스 정보를 이용하여 손쉽게 응용 트랜잭션을 정의할 수 있다.

5.3 그래픽 사용자 인터페이스

본 논문에서 사용하는 인터페이스는 보안객체를 모델링하기 위한 편집기로 이름을 SODE(Secure Object Diagram Editor)라 한다.

편집기의 구성은 다이어그램을 생성하고 관리하는 메뉴 영역과 클라스와 속성, 속성 링크, 관련성 정의 방법 도구를 제공하는 도구 버튼 영역이 있다. 모델링된 다이어그램은 캔버스 영역에 그려진다. 캔버스 영역은 크기가 큰 다이어그램을 편집할 수 있게 스크롤 기능을 가지고 있다. 메시지 윈도우는 객체 그림표를 수행하는 과정이나 오류 정보를 문자 정보로 나타내어 준다.

5.4 객체 관리자

객체 관리자는 각 정의 모듈에서 정의된 객체의 보안등급과 관련 정보를 관리하는데, 클라스 정의 모듈에서 정의된 클라스 보안등급과 범주에 관한 정보를 관리한다. 임의의 사용자가 객체 관리자에 의해 관리되는 정보 접근을 시도하였을 때, 사용자의 보안등급과 범주 정보를 비교하여 접근이 허용된 정보만을 제공한다.

5.5 데이타 사전

하나의 객체가 생성될 때마다 하나의 자료구조가 생성된다. 이러한 자료구조는 연결구조(linked list) 형태로 구성되었다. 시스템에서 사용되는 데이타 사전의 구성은 자료의 형태에 따라 클라스, 속성, 속성 링크, 관련성으로 구분된다. 클라스와 속성의 자료구조는 보안등급 정보와 이름을 갖고 그래픽 정보를 제공하기 위한 위치를 텍스트 정보로 갖게 된다. 속성 링크의 자료구조는 보안 등급을 가지고 속성 자료구조에 연결되어 있다. 관련성 자료구조의 구성은 관련성 형태에 따라 일항, 이항, 일반화, 집단화, 삼항으로 구분되고 각각의 관련성 정보는 대응수와 대응되는 클라스의 인덱스를 갖고 있다.

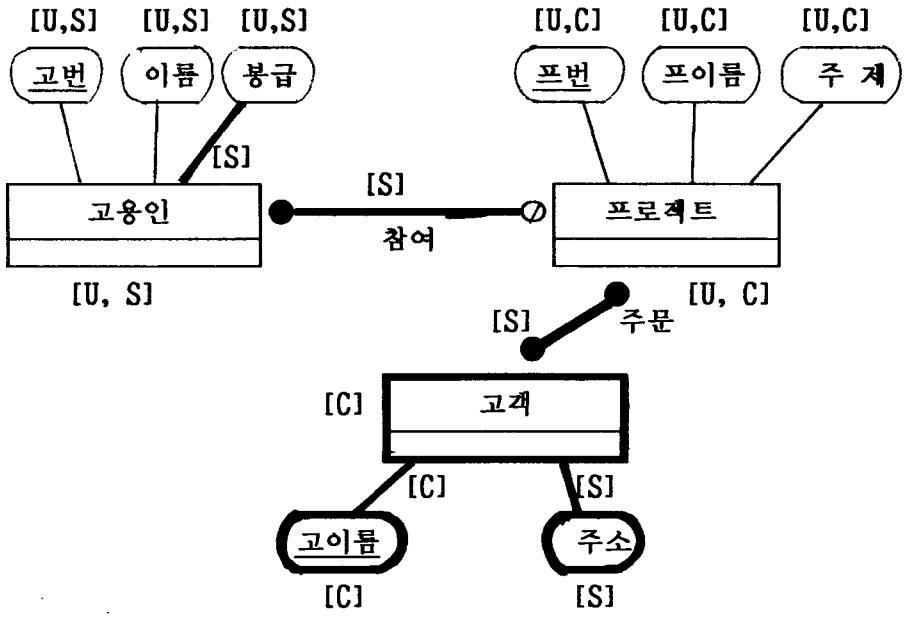
5.6 보안 트랜잭션 생성 예

그림4는 고용인, 프로젝트 그리고 고객 등의 객체 클라스들을 SOREM 표기법에 따라 모델링한 것이다. 이 예제는 일쪽이 선택인 일대다 이항 연관성과 양쪽이 필수인 다대다 이항 연관성으로 이루어져있다. 일쪽이 선택이면 일쪽에서 다쪽으로 참조 무결성이 존재하고, 다쪽에서 일쪽으로 객체 함수 종속성이 존재한다. 양쪽이 필수인 다대다 이항 연관성에서는 '주문'이라는 관련성 객체 클라스에서 트랜잭션을 모델링한다. '주문' 관련성 객체 클라스로부터 기존의 객체 클라스 방향으로 객체 함수 종속성과 참조 무결성이 존재한다.

그림4에 대한 검색, 삽입, 삭제, 갱신 등과 같은 기본 트랜잭션은 모델링 도구에서 자동적으로 생성된다. 예를 들어, 객체 클라스 고용인에 대한 삭제 기본 트랜잭션은 아래와 같이 생성된다.

```
TRANSACTION DELETE_고용인
   BEGIN
      GET(#고번)
      GET(이름)
      GET(봉급)
      IF (AND (Permitted_Security_Class) (EXIST(고용인, #고번)))
      THEN
```

```
        EXCLUDE(#고번)
        EXCLUDE(이름)
        EXCLUDE(봉급)
        IF (EXIST(프로젝트, #고번))
          THEN
          DELETE_프로젝트(#고번)
        ENDIF
      ELSE EXIT
      ENDIF
END
```



(그림4) 프로젝트 관리 모델링

또한, 응용 트랜잭션은 응용분야에 따라서 여러가지로 정의될 수 있다. 그림4에서 한 예로 "어떤 고용인이 참여한 프로젝트를 주문한 고객은 몇 명인가?"라는 응용 트랜잭션은 아래와 같다.

```
TRANSACTION  COUNT_주문
   BEGIN
     GET(#고번)
     kount := 0
     IF(EXIST(고용인, #프번))
      THEN
        REPEAT
          SELECT_주문(#고이름)
          kount := kount + 1
        UNTIL  NO_MORE_EXIST
     ENDIF
   END
```

다른 예제로 "특정 프로젝트에 참여한 고용인들의 봉급 총합은 얼마인가?"라는 응용 트랜잭션은 다음과 같다.

```
TRANSACTION  SUM_SALARY_고용인
   BEGIN
     GET(#프번)
     sum := 0
```

```
  REPEAT
    IF(EXIST(고용인, #프번))
    THEN
    SELECT_고용인(#고번)
    PROJECT_고용인(봉급)
    sum := sum + 봉급
    ENDIF
  UNTIL  NO_MORE_EXIT
END
```

6. 결 론

데이타베이스 개념적 설계는 데이타의 정적인 스키마 구조 정의와 이들 데이타 구조에 대한 동적 성질의 정의를 포함해야 응용영역의 정확한 분석 및 설계를 할 수 있다. 또한, 데이타베이스 설계 단계와 구현 단계 사이의 연산의 불일치 상태를 미리 발견함으로써 비용을 감소시킬 수 있는 장점이 있다. 지금까지 데이타베이스의 정확한 스키마 모델링을 위해 많은 모델들이 제시되었지만, 데이타의 동적 성질을 모델링하기 위한 방법론들은 미약하였고 데이타베이스의 동적 성질에 보안성을 첨가한 자동화 도구는 제공되지 않고 있다.

본 논문에서는 데이타베이스의 개념적 모델링을 하기위한 보안 객체-관련성 모델에서 객체의 무결성과 보안성을 보장하는 보안 트랜잭션 모델링 방법론을 설명했다. 그리고 보안 트랜잭션 모델링 과정을 자동화하는 일관적이고 효율적인 보안 트랜잭션 자동화 도구를 설계 및 구현하였다. 모델링된 데이타베이스 스키마와 트랜잭션은 설계와 구현 단계에서 데이타에 대한 무결성을 보장한다. 그리고 보안 객체 데이타 모델 그림표를 곧바로 인식하여 자동으로 기본 트랜잭션을 생성하며 사용자는 STDL을 이용하여 응용 트랜잭션을 정의할 수 있다.

추후에 데이타베이스의 개념적 설계단계에서 모델링된 트랜잭션의 목표 데이타베이스 시스템에 탑재에 대한 연구가 필요하다.

< 참 고 문 헌 >

[1] Blah, M. R., Premerlanl, W. J., and Rumbaugh, J.E., "Relational Database Design Using an Object-Oriented Methodology", Comm. ACM Vol.31, No.4, 1988, pp414-427.

[2] Braegger, R.P., Budler, A.M., Rebasmen, J., and Zehender, C. A., "Gambit : An Interactive Database Design Tool for Data Structure, Integrity Constraints, and Transaction", IEEE Transactions on Software Engineering, Vol.SE-11, No.7, July 1985, pp574-582

[3] Chen, P. P., "The Entity-Relationship Model -- toward a unified view of data", ACM TODS, March 1976, pp9-36.

[4] Denning, D. E., Lunt, T. F., Shell. R. R., Shockely, W. R. and Heckamn, M., " The SeView Security Model", IEEE Transactions On Software Engineering. Vol.16, No.6, June 1990, pp593-607.

[5] Lipeck, U. W., "Stepwise Specification of Dynamic Database Behavior", ACM SIGMOD, May 1986, pp387-397.

[6] Loomis, M. E. S., Shan, A. V., and Rumbaugh, J. E., "An Object Modeling Technique for Conceptual Design", '87 Proc. of ECOOP, June 1987, pp192-202.

[7] Smith, G. "Modeling Securtiy-Relevant data Semantics", IEEE Transactions On Software Engineering, Vol.17 No.11, November 1991, pp1195-1203.

[8] Stachour, P. and Thuraisingham, B., "Design of LDV : A Multilevel Secure Relational Database Management System", IEEE Transactions on Knowledge and Data Engineering Vol.2, No.2, June 1990, pp190-209.

[9] 노봉남, 심갑식, "A Secure Data Modelling for Database Design", 데이타베이스학회 국제학술대회 논문집, 1992.

< 부 록 > STDL 문법

```
STDL_Definition : STDL_Header
                  BEGIN
                  STDL_Body
                  END
STDL_Header : TRANSACTION Transaction_Name
Transaction_Name : IDENTIFIER
STDL_Body : STDL_Body Transactions
          ¦ Transactions
Transactions : IF '(' Predicate_List ')'
               THEN
                STDL_Body
               ELSE
                STDL_Body
               ENDIF
             ¦ IF '(' Predicate_List ')'
               THEN
                STDL_Body
               ENDIF
             ¦ Transaction
Transaction : INCLUDE '(' Arg_List ')'
            ¦ EXCLUDE '(' Arg_List ')'
            ¦ GET '(' Arg_List ')'
            ¦ IDENTIFIER '(' Arg_List ')'
            ¦ WHILE '(' Predicate ')'
                STDL_Body
              ENDWHILE
            ¦ REPEAT
                STDL_BODY
              UNTIL NO_MORE_EXIST
            ¦ CASE Exp OF
                Tag_Lists
              ENDCASE
            ¦ EXIT
Predicate_List : Predicate_List Predicate
               ¦ Predicate
Predicate : EXIST '(' Arg_List ')'
          ¦ LAST_INSTANCE '(' Arg_List ')'
          ¦ IDENTIFIER '(' Arg_List ')'
          ¦ Predicate AND Predicate
          ¦ Predicate OR Predicate
          ¦ NOT '(' Predicate ')'
Tag_Lists : Tag_Lists Tag_List
          ¦ Tag_List
Tag_List : '''IDENTIFIER''' ':'
         ¦ STDL_Body
Arg_List : Arg_List ',' Exp
         ¦ Exp
Exp : Exp Operator Exp
    ¦ IDENTIFIER
    ¦ NUMBER
    ¦ '(' Exp ')'
Operator : '>' ¦ '<' ¦ '>=' ¦ '<=' ¦ '==' ¦ '!='
         ¦ '+' ¦ '-' ¦ '*' ¦ '/' ¦ '='
```