다단계 보안 응용에서의 일반화 계층구조의 스키마 변환

박운재○ 송의철 송호영 김정종
경 남 대 학 교 컴 퓨 터 공 학 과
woojpark@weboption.net, {ecsong, hysong, jjkim}@eros.kyungnam.ac.kr

Schema Transformation of Generalization Hierarchy with Multilevel Secure Application

Woon-Jae Park○ Eui-Chul Song Ho-Young Song Jung-Jong Kim
Dept. of Computer Science, Kyungnam University

요 약

개념-논리 스키마 설계의 목적은 개념 스키마를 사용 가능한 데이터의 구조화와 제약사항 모델화를 위한 기능들을 가능한 한 효과적으로 표현하는 것이다. 스키마 변환은 이러한 표현을 구현 언어나 데이터베이스와는 독립적인 차원에서 고려할 수 있도록 해준다. 개념-논리 스키마로의 변환에 프리미티브 변환을 적용함으로써 얻을 수 있는 이점은 개념 스키마에 나타난 정보를 손실 없이 구현 모델에 반영할 수 있게 한다는 것이다. 본 논문에서는 다단계 보안 응용을 위한 일반화 계층구조의 스키마 변환에서 발생할 수 있는 프리미티브 변환을 논의한다. 이러한 변환은 다단계 보안 응용의 개념 논리화 설계 단계에 대한 질적 향상을 기할 수 있는 기본적인 도구의 토대가 될 수 있으며 또한 이를 적용함으로서 설계 과정을 변환의 작은 단위로 생각할 수 있게 해 줌으로서 설계 과정을 단순화 할 수 있다.

1. 서 론

데이터의 중요성이 증가하면서 데이터베이스의 도입이 늘고 있으며 이에 따라 데이터베이스 시스템은 응용 소프트웨어의 중심이 되고 있다[1, 2]. 또한 소프트웨어 개발과정에서도 데이터베이스 응용(database application) 설계의 중요성이 부각되고 있다.

현재 소프트웨어 설계를 위한 지배적인 방법론은 객체지향 방법론이다[3, 4]. 그리고 데이터베이스 응용의 중요성이 부각됨에 따라 객체지향 방법론의 데이터베이스 응용에의 적용을 위한 연구 또한 활발해지고 있다.

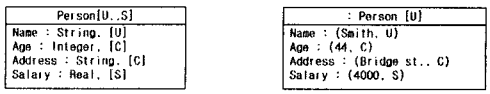
보안 데이터베이스 응용 설계에서는 우선 개체 및 개체와 연관된 보안등급 등을 적절한 모델을 사용하여 모호성 없이 표현하여야 한다.

본 논문에서는 데이터베이스 응용 설계에서 유용하게 사용되는 프리미티브 변환이 보안 응용에도 적용될 수 있는지를 일반화 계층구조의 변환에 적용될 수 있는 모든 변환에 대해 알아본다.

2. 관련 연구

2.1 다단계 보안

다단계 보안은 서로 다른 보안등급을 가진 사용자마다 접근하는 데이터 집합이 다른 다단계로 되어 있음을 의미한다[30]. 이를 위해서는 데이터베이스에 저장된 객체가 서로 다른 보안등급을 갖고 저장되어야 한다. 그림 2.1은 다단계 보안 모델인 Multi-View 모델에서의 인스턴스 생성 예를 보이고 있다. Person 클래스는 U, C, S등급의 객체를 가질 수 있으며 U등급의 인스턴스 생성을 보이고 있다.



(a) (b)

그림 2.1 Multi-View 모델의 클래스와 인스턴스

그림 2.2는 Multi-View 모델의 저장 구조를 보이고 있다. 각각의 데이터베이스는 해당 등급에서 생성된 데이터를 저장하고 있다. 그림 2.1은 U등급의 데이터베이스에 저장되며 U등급 보다 높은 보안등급의 모든 속성 값은 보안등급의 값이 할당되어 저장된다.

C등급의 인스턴스는 최소 중복으로 갱신되며 U등급의 데이터베이스에서 "Confidential" 값인 C등급의 속성이 이제 실제 값을 갖는다. S등급 인스턴스도 C등급 인스턴스와 유사하게 저장된다.

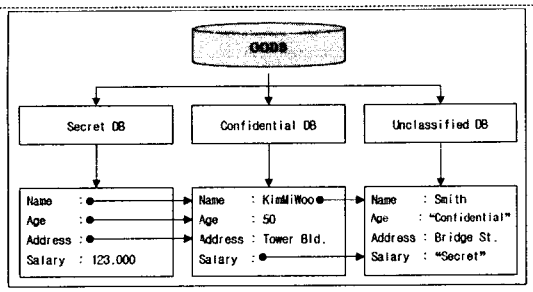


Fig. 2.2 Multi-View 모델의 구조

C등급의 인스턴스는 최소 중복으로 갱신되며 U등급의 데이터베이스에서 "Confidential" 값인 C등급의 속성이 이제 실제 값을 갖는다. S등급 인스턴스도 C등급 인스턴스와 유사하게 저장된다.

저장된 데이터에 대한 갱신은 해당 등급의 데이터베이스에 접근하여 현재의 속성 값을 갱신한다. 그림 2.2는 U, C, S등급의 인스턴스가 순서대로 입력되어 있는 상태를 보이고 있다.

위와 같은 구조에 의해 서로 다른 분류등급의 사용자는 서로 다른 등급의 데이터베이스를 검색함으로 서로 다른 데이터 집합에 접근하게 된다.

2.2 스키마 변환

논리적 설계의 목적은 개념 스키마를 사용자의 목적에 따라 특정 DBMS에 맞는 논리 스키마로 변환하는 것이다. 개념적 설계의 기본 목적이 스키마의 완전성과 표현성인 데 비해 논리적 설계는 논리적 모델에서 사용가능한 데이터의 구조화와 제약사항 모델화를 위한 기능들을 가능한 효과적으로 표현하는데 있다. 논리적 설계는 모델 독립적인 상위 레벨 논리적 설계와 모델 의존적 논리적 설계로 나누어진다.

상위 레벨 논리적 설계는 개념 스키마를 단순화하고 최적화한 개념-논리 스키마로 변환한다. 이러한 변환은 클래스의 분할과 조합, 유도된 데이터에 대한 결정, 일반화 계층의 제거와 같이 개념적이라기보다는 논리적인 세부사항을 표현하는 것이기 때문에 개념-논리 스키마(conceptual-to-logical schema)라고 한다.

개념 스키마의 개념-논리 스키마로의 변환에 적용될 수 있는 프리미티브 변환에 대해 Batini 등은 데이터베이스 설계 과정에서 적용 가능한 프리미티브 변환의 종류를 하향식 프리미티브와 상향식 프리미티브로 분류하여 개념적 설계 과정에서 이들을 적용한 설계 방법론을 제시하고 있다.

한편, Blaha 등은 프리미티브 변환의 종류를 적용 대상에 따라 단일 구성체에 대한 변환, 다중 구성체에 대한 변환, 상속의 변환 등으로 분류하고 이를 상세 설계 단계 즉, 개념-논리 스키마 설계 단계에 적용하여 개념적 모델을 단순화하고 최적화할 것을 제안하고 있다.

3. 일반화 계층구조의 변환

개념 모델에서 정의된 일반화 계층구조에 대한 초기의 정의가 여전히 유효한가를 검증할 목적으로 일반화 계층구조를 검토할 필요가 있다. 즉, 개념 모델의 클래스들 사이에 유사한 속성이 있는가를 찾아내어 이들이 가지고 있는 공통된 속성과 연산을 상위클래스나 하위클래스에 배치하거나 새로운 상위클래스를 생성한다. 이는 각각의 클래스에 대한 이해를 분명히 하여 클래스로 변경되어야 할 속성과 속성으로 변경되어야 할 클래스들을 발견하는 것이다. 일반화 계층에 대한 변환의 단계와 보안성에 대한 논의는 다음과 같다.

3.1 일반화 계층구조에서의 속성 이동

개념적 스키마의 어떤 속성을 효율적인 상속을 위해서나 널 값의 방지를 위해 일반화 계층구조의 상위 클래스나 하위 클래스로 이동할 필요가 있는지 결정한다. 그림 3.1은 이러한 이동에 대한 프리미티브 변환을 나타내고 있다.

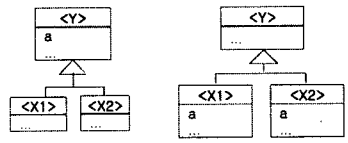


그림 3.1 속성의 이동

속성의 이동에 대한 보안성질은 속성의 보안성질인 자신을 정의하는 클래스의 보안등급 이상이 되도록 하여야 한다. 그러므로 이동하고자 하는 속성의 보안등급이 이동할 클래스의 보안등급 보다 낮다면 이동할 수 없다. 즉, 상위 클래스의 속성을 하위 클래스로 이동하는 것은 보안 위반을 일으키지 않는다. 반면에 하위 클래스에서 상위 클래스로의 이동은 속성의 보안등급이 클래스의 보안등급 보다 높은 경우만 가능하다.

3.2 일반화 계층구조에서의 연관성 이동

속성과 마찬가지로 연관성도 이동될 수 있다. 일반화 계층구조에서 연관성의 이동은 상위 클래스와의 연관성을 하위클래스로 이동하는 경우와 하위클래스의 연관성을 상위 클래스로 이동하는 경우를 생각할 수 있다. 그림 3.2는 상위 클래스의 연관성을 하위 클래스 연관성으로 변환하는 것을 보이고 있다.

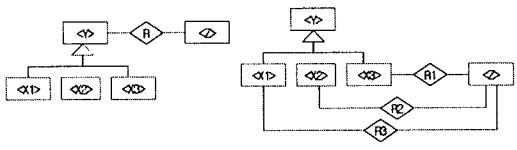


그림 3.2 연관성의 이동

상위 클래스로의 연관성 이동은 연관성의 보안 성질인 연관성에 참여하는 클래스의 보안등급 이상이 되도록

보장되어야 한다. 그러나 하위 클래스로의 연관성 이동은 연관성의 분할을 일으키며 보안성에는 영향을 미치지 않는다. 그림 3.2에서 연관성 R은 R1, R2, R3로 분할되었다.

3.3 일반화 계층구조의 생성

일반화 계층구조를 만들기 위한 어떤 기준이 있어야 한다. 임의의 하위 클래스는 다음 목적 중의 하나를 제공할 때 하위 클래스로의 의미가 있다.

① 상위 클래스의 정적 상태에 대해서, 부가적인 정보를 저장할 수 있도록 적절히 확장될 정적 상태(static state)를 가져야 한다.
② 다른 클래스와의 관계에서 다른 유형의 관계를 가져야 한다.
③ 직접적으로 하위 클래스의 객체에 적용되지만 상위 클래스가 지원하지 못하는 어떤 연산이나 제약조건이 있어야 한다.

보안성을 고려할 경우 상위 클래스로 정의될 속성들은 하위 클래스에서 동일한 보안등급을 갖고 있는 속성들이어야 하며 하위 클래스로 정의될 속성들 보다 낮거나 같은 등급이어야 한다. 만약 상위 클래스로 정의하고자 하는 속성들이 서로 다른 보안등급을 갖고 있다면 이는 상위 클래스로 정의될 수 없다. 이러한 이유는 일반화 계층구조에 대한 보안 성질 때문이다.

3.4 일반화 계층구조의 제거

위에서 언급한 일반화 계층구조를 만들기 위한 기준의 조건 중 어느 것도 갖추지 않으면, 하위 클래스는 연관된 의미가 없다. 클래스의 원소를 표시할 명칭을 도입하는 것이 여전히 편리하지만, 실제로 하위 클래스가 자율적으로 도입하는 부가적인 구조적 정보는 없다. 그러므로, 이 단계에서 하위 클래스를 제거할 수 있다. 그러나 위의 조건이 충족되지 않아도 보안 관점에서는 유효하다. 즉, 하위 클래스들의 보안등급이 서로 다를 때는 위의 사항에 따라 일반화를 제거할 수 없다. 그렇지만 하위 클래스들의 보안 등급이 동일할 때는 상위 클래스의 정적 상태에 제거될 하위 클래스를 모델링하는 하나 이상의 부가적인 속성을 첨가해야 한다. 물론 부가적으로 첨가된 속성의 보안등급은 제거된 하위 클래스의 보안등급을 그대로 따른다.

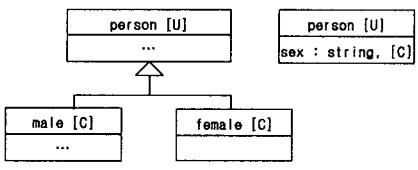


그림 3.3 일반화 계층의 제거

예를 들어, U 등급의 클래스 person에 C 등급의 sex 속성을 간단히 첨가함으로써 person 일반화에서 male과 female을 제거한다(그림 3.3).물론, sex 속성에 적당한 값을 줌으로써 특정 하위 클래스에 있는 멤버쉽(membership)을 모델링한다.

3.5 다중 상속의 구조 결정

다중 상속은 가능하면 억제하는 것이 좋다. 다중 상속을 동일한 의미의 단일 상속으로 변환한다. 그림 3.4의 (a)는 다중 상속의 분할 전을 (b)는 분할 후를 예시하고 있다. 다중 상속을 분할할 경우 분할된 클래스는 연관성에 의해 서로의 관계가 설정된다. 다중 상속의 분할에서 보안성은 분할 전의 보안등급을 따른다.

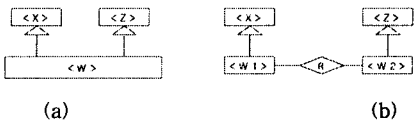


그림 3.4 다중 상속의 분할

4. 결 론

다단계 보안 응용 설계를 위한 정형적인 방법론은 아직 정립되지 않았다. 기존의 다단계 보안 응용 설계에 대한 접근들은 개념 스키마의 설계 단계에 초점을 두고 있는데 반해 본 논문은 개념 스키마의 개념 논리화 스키마로의 변환에 초점을 두고 있다. 특히 일반화 계층구조의 변환에서 발생할 수 있는 모든 프리미티브 변환에 대해 논의하였다. 제시된 변환 방법은 설계 방법론의 일부로 적용될 수 있으며 설계 품질의 향상에 기여할 수 있다. 앞으로의 연구 과제는 기능적 모델과 같은 다른 모델링 요소에서의 보안 응용 설계의 영향 및 규칙 기반 시스템에서의 보안 응용 설계 방법과 보안 응용 설계를 위한 도구의 개발이다.

참고문헌

[1] Blaha and Premerlani, "Object-Oriented Modeling and Design for Database Applications", Prentice-Hall, 1998.

[2] Stefano Ceri and Piero Fraternali, "Designing Database Applications with Objects and Rules", Addison-Wesley, 1997.

[3] Goers J., Lisson K. P. and Linde-Goers H. G. "Experiences in Object-Oriented Modeling of a Real Database Application", 1998.

[4] Rahayu W., Chang E. and Dillon T. S. "Implementation of Object-Oriented Association Relationships in Relational Databases", Proceedings of the 1998 International Database Engineering & Applications Symposium, 1998.