술어를 이용한 내용 의존적 권한부여 기법
(Content-Dependent Authorization Mechanism using Predicates)

홍 성 림 † 박 창 원 †† 정 진 완 †††
(Sung-Lim Hong) (Chang-Won Park) (Chin-Wan Chung)

요 약 본 논문은 객체지향 데이타베이스 시스템에서의 내용 의존적 권한부여 기법을 제시한다. 현재까지 객체지향 데이타베이스를 위한 많은 권한부여 모델들이 제안되었으나 대부분 데이타베이스의 내용에 기반한 권한부여를 지원하지 못하였다.

본 논문은 객체지향 데이타베이스를 위한 기존의 내용 독립적 권한부여 모델을 클래스의 애트리뷰트의 값에 대한 술어를 이용하여 확장한 내용 의존적 권한부여 모델을 제시하였다. 제시된 모델은 객체의 값에 대한 명시된 조건을 만족하는 객체들을 집단화하여 그러한 객체들에 대해 하나의 권한을 부여할 수 있게 한다. 또한 부정 권한을 지원하며 긍정 권한과 부정 권한 사이의 충돌을 해결하기 위하여 강성 권한과 약성 권한의 개념을 지원한다. 또한 권한에 술어를 결합시킴으로써 생기는 여러 가지 문제점들을 지적하고 이를 해결한다. 특히 내용 독립적 권한부여 모델에서의 권한 연산들이 본 논문에서 제시하는 모델에서 그대로 적용될 수 없음을 보이고 연산들의 의미를 재정의 하였다.

국문키워드 : 객체지향데이타베이스, 권한부여, 접근제어

Abstract In this paper, we present a content-dependent authorization mechanism for object-oriented database systems. So far, several models of authorization for object-oriented databases have been proposed, but most of these models do not support the authorization based on the database content.

This paper shows how the traditional content-independent authorization model can be extended to provide the content-dependent authorization using predicates on the values of attributes of a class. The proposed model makes it possible to group objects that satisfy the specified conditions on the values of the objects and to grant a single authorization on those objects. This model supports the negative authorization and provides the concept of the strong and weak authorization to resolve conflicts between positive and negative authorizations. In addition, we address and resolve some of the problems that arise when the predicates are associated with the authorization. In particular, since the authorization operations of the traditional content-independent model become inadequate for our mode, we redefine the semantics of the authorization operations.

key word : object oriented database, authorization, access control

1. 서 론

데이타베이스 시스템에서 권한이란 데이타베이스 내의 객체에 대해 어떤 사용자가 어떤 접근 모드로의 접근이 허용되는 지를 명시하는 것이다. 정당한 권한을 가진 사용자만이 데이타를 접근할 수 있도록 제어해 주는 권한부여는 다중 사용자 DBMS의 매우 기본적인 기능 중 하나이다[1, 2].

권한부여 기법은 객체의 이름이나 객체 식별자에 기반하여 권한을 부여하는 내용 독립적 권한부여(content-independent authorization)와 객체 식별자나 이름에 따른 권한부여 뿐 아니라 객체의 내용 즉 값에 따라 객체에 대한 접근의 허용 여부를 판단할 수 있는 내용 의존적 권한부여 방식으로 분류할 수 있다[3]. 내용 의존

† 비 회 원 : (주)데이콤 멀티미디어 인터넷
syrup@hanmir.com
†† 비 회 원 : 한국과학기술원
chanwon park@origo.net
††† 종신회원 : 한국과학기술원 전산학과 교수
chungcw@islab.kaist.ac.kr
논문접수 : 2000년 7월 27일
심사완료 : 2002년 10월 31일

적 권한부여 기법은 어떤 사용자에게 특정 객체에 대한 권한을 부여할 때 그 객체의 애트리뷰트의 값에 대한 조건을 명시하여 명시된 조건을 만족하는 경우에만 그 사용자가 해당 객체를 접근할 수 있도록 한다. 내용 의존적 권한부여는 어떤 클래스의 특정 애트리뷰트들의 값에 대해 명시된 요구 조건을 만족하는 인스턴스들의 집단화하여 하나의 권한을 부여할 수 있게 함으로써 각각의 인스턴스에 권한을 부여해야하는 수고를 덜어주고 저장 공간을 절약할 수 있게 하며 다양한 보안 요구 조건을 표현할 수 있게 한다.

관계형 데이타베이스에서는 뷰를 이용하여 내용 의존적 권한부여를 지원한다[4, 5]. 이에 반해 객체지향 데이타베이스 시스템을 위한 대부분의 권한부여 모델들은 내용 의존적 권한부여를 지원하지 못한다. 가장 큰 이유는 객체지향 데이타베이스 시스템에서는 객체 식별, 클래스 계승 계층, 메소드 등 그 의미적 복잡성 때문에 뷰가 정의되기 어렵기 때문이다[6]. 또한 뷰를 이용한 권한부여 기법은 뷰에 대한 권한이 그 뷰가 직접 또는 간접적으로 참조하는 테이블(뷰 또는 릴레이션, 객체지향 데이타베이스에서는 클래스에 해당)들에 대한 권한들과 일관성을 가져야 하고 자신과 같은 테이블을 참조하여 정의된 다른 뷰들에 대한 권한들과도 일관성을 유지해야하는 문제점이 있다. 이것은 어떤 뷰의 인스턴스들이 그 뷰가 참조하는 테이블의 인스턴스들과의 사이와, 자신과 같은 테이블을 참조하는 다른 뷰들의 인스턴스들과의 사이에 교집합이 가질 수 있기 때문에 하나의 튜플(객체지향 데이타베이스 시스템에서는 객체 또는 인스턴스에 해당)에 대해 서로 다른 여러 권한들을 허가할 수 있기 때문이다. 예를 들어 A라는 클래스가 있고 이 클래스의 인스턴스들의 부분집합으로 정의된 뷰 B와 C가 있고 클래스 A의 어떤 인스턴스 a는 뷰 B와 C에 동시에 속한다고 하자. 사용자 U에게 뷰 B에 대해서는 읽기를 금지시키고(부정 읽기 권한을 허가) 뷰 A에 대해서는 읽기를 허용(긍정 읽기 권한을 허가)한다면 B와 C에 동시에 속하는 인스턴스 a의 경우 어떤 권한을 적용해야 할 지 모호해진다. 따라서 어떤 한 객체에 대한 어떤 접근 요구가 있을 때 그것이 타당한 지를 판단하기 위해서는 권한 검사 연산 시에 적절한 우선 순위를 적용하여 하나의 권한을 선택하던가 권한 허가 연산 시에 관련되는 모든 테이블과 뷰들의 권한을 조사하여 충돌을 해결하여야 하는 문제점이 있다. 이러한 복잡성을 이유로 뷰를 이용한 대부분의 권한 부여 기법에서 뷰에 대해서는 부정 권한을 지원하지 않는다.

본 논문에서는 객체지향 데이타베이스 시스템을 위한 기존의 내용 독립적 권한부여 모델을 확장하여 내용 의존적 권한부여 모델을 제시한다. 제안된 권한 부여 모델은 클래스에 대한 권한에 그 권한이 유효한 조건을 나타내는 술어를 추가하여 그 술어를 참으로 만드는 클래스의 인스턴스들에 대하여 접근을 허용할 수 있는 방법을 제공하며 내용 의존적 권한 부여에 관한 기존 연구들과는 달리 부정 권한을 지원한다.

본 논문에서 제안한 모델은 클래스에 대한 권한이 권한 객체인 클래스 전체에 대한 권한을 의미하지 않고 전체 인스턴스 중 명시된 조건을 만족하는 임의의 부분집합에 대한 권한이 될 수 있으므로 실제로 권한이 적용되는 대상과 명시된 권한 객체가 일치하지 않을 수 있는 문제가 발생한다. 또한 어떤 한 객체의 값이 변화함에 따라 권한의 명시적 취소와 허가가 없이도 객체에 대한 사용자의 권한이 변할 수 있다. 따라서 기존의 내용 독립적 모델에서의 권한 연산의 의미와 알고리즘을 그대로 적용할 수가 없게 된다. 이러한 문제들을 해결하기 위하여 본 논문에서는 권한 객체 중 실제로 권한이 적용되는 부분을 실 권한 객체라 정의하고 실 권한 객체들 사이의 관계를 토대로 권한 주체와 권한 객체가 같은 임의의 두 권한 사이의 관계를 분류하였다. 또한 권한 연산 시 분류한 관계를 바탕으로 하여 권한 연산의 의미를 재정의하였다.

제안한 모델에서는 기초로 한 내용 독립적 모델에서의 5-tuple의 권한이 8-tuple로 확장되어 권한 자체의 크기가 늘어남에 따라 하나의 권한을 저장하기 위한 저장 공간상의 부담이 커지게 되나 이는 어떤 조건을 만족하는 객체들을 집단화하여 하나의 명시적 권한만을 부여함으로써 명시적 권한의 수를 줄이게 되어 얻는 저장 공간상의 이점으로 보상될 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 객체지향 데이타베이스에서의 기존의 권한부여 기법에 관한 기존 연구 결과들에 관해 기술하였다. 3절에서는 객체지향 데이타베이스를 위한 ORION의 내용 독립적 권한부여 모델을 확장하여 술어를 이용한 내용 의존적 권한부여 모델을 제안하였다. 4절은 3절에서 제안한 모델에서의 권한 연산들에 대해 기술하였다. 마지막으로 5절에서는 결론에 대해 논의하였다.

2. 관련 연구

본 절에서는 먼저 객체지향 데이타베이스 시스템에서 연구된 권한부여 기법들에 대하여 설명하고 내용 의존적 권한부여를 지원하기 위해 제안된 기존의 방법들과 그들의 문제점들에 관하여 논의하였다.

2.1 객체지향 데이타베이스를 위한 권한부여 기법

권한부여의 단위가 릴레이션 또는 뷰인 관계형 모델과는 달리 객체지향 모델에서는 객체가 접근 단위이자 권한부여의 단위가 된다[1]. 객체 수준의 권한부여를 지원하기 위해서 명시적으로 저장해야 하는 권한의 수가 매우 증가함에 따라 각 객체마다 권한을 허가하고 저장하는 오버헤드를 줄일 수 있는 묵시적 권한의 필요성이 증가하게 되었다. 묵시적 권한이란 명시적 권한에서 암시될 수 있는 권한으로 권한 객체, 권한 주체, 권한 타입 세 차원(dimension)을 따라 모두 유도될 수 있다[1,2]

객체지향 데이타베이스에서의 임의적 접근제어를 위한 대표적인 모델로 Rabitti[2]가 제안한 ORION의 권한부여 모델을 들 수 있다. 이 모델에서 묵시적 권한은 객체 계층을 따라 계승되며, 묵시적 권한의 예외를 나타내기 위하여 부정 권한의 개념을 도입하였다. 부정 권한의 도입은 긍정 권한과의 충돌의 가능성을 내포하는데 이것을 해결하기 위한 방법으로 most specific rule이나 denials-take precedence rule과 강성/약성 권한등이 있다. ORION에서는 강성/약성 권한으로 권한들 사이의 우선 순위를 나타내어 권한들 사이의 충돌을 해결하였다. 미리 정해진 규칙에 따라 권한들 사이의 우선 순위가 정해지는 most specific rule이나 denials-take precedence rule과 달리 강성/약성 권한은 권한 허가자가 권한에 따라 적절히 강성 또는 약성으로 명시함으로써 해당 권한이 다른 권한들에 의해 오버라이드될 수 있는지의 여부를 표현할 수 있도록 한다[7].

데이타베이스에서의 권한은 특정 사용자의 특정 객체에 대한 특정 접근모드로의 접근의 허용 또는 금지를 나타내며, Bertino[7]에 따르면 권한 a는 5-tuple (s, o, m, as, at)로 정의된다. s는 권한 주체(Authorization Subject), o는 권한 객체(Authorization Object), m은 R(Read) 또는 W(Write) 등 접근 모드(Access Mode), as는 권한 부호(Authorization Sign)로 '+' 또는 '-'이며 각각 긍정 권한과 부정 권한을 의미하고, at는 권한 타입(Authorization Type)으로 'st' 또는 'wk'의 값을 가지며 각각 강성 권한과 약성 권한을 나타낸다. 권한 a는 권한 주체 s가 권한 객체 o에 대해 접근 모드 m, 권한 부호 as, 권한 타입 at의 권한을 가짐을 의미한다.

최근 들어 데이타웨어하우스, 시공간 데이타베이스(spatiotemporal database), 전자상거래, 전자도서관과 같은 응용분야에서 시간적(temporal) 데이타의 사용이 활발해 졌고 이에 따라 데이타의 시간적 속성에 기초한 권한 부여 모델들도 많이 제안되었다[8,9].

2.2 내용 의존적 권한부여에 관한 기존 연구

관계형 데이타베이스 시스템에서는 뷰를 정의하여 뷰에 대해 권한을 허가하고 취소함으로써 내용 의존적 권한부여를 지원한다. 뷰에 대한 권한을 허가 받은 사용자는 뷰에 명시된 조건을 만족하는 데이타베이스의 임의의 부분 집합에 대한 접근이 가능하다. 뷰를 이용한 권한부여 방법은 여러 테이블(릴레이션 또는 뷰)들에 동시에 포함되는 튜플의 경우 각각의 테이블을 통해 접근 권한을 허가 받을 수 있으며 또한 자신이 속한 여러 테이블들을 통해 접근이 가능하다. 동일한 튜플에 대해 권한을 명시할 수 있는 방법과 접근할 수 있는 방법이 여러 가지가 될 수 있으며 이에 따라 특정 튜플을 어떤 테이블을 통해 접근하는지에 따라 그 접근이 허용될 수도 금지될 수도 있는 문제가 발생한다. 이러한 테이블간의 권한의 일관성 문제 때문에 뷰에 대해서는 대부분 부정 권한을 지원하지 않는다[4].

Alturi[8]는 정보포탈(Information potal)에서 기초 데이타로부터 매핑을 통해 가공되어 유도된 데이타들에 대한 간접적인 허가받지 않은 접근을 금지하기 위한 일관성있고 안정한 방식의 구조를 제안하였다. 이 연구에서 Alturi는 매핑의 특성과 접근주체의 기초 데이타와 매핑에 대한 권한에 기초하여 유도된 정보들에 대해 권한의 유도를 가능하게 하는 "안전한 권한부여"(safe authorization)이라는 개념을 정의했다. [8]에서 Alturi는 기초 데이타로부터 매핑을 통해 생성된 유도된 데이타에 대해 매핑과 유도된 데이타를 가지고 기초 데이타를 재생 해 낼 수 있는지의 여부에 따라 매핑을 재생가능 매핑(reconstructive mapping)과 재생불가능 매핑(non-reconstructive mapping)으로 분류하였다. 이 방식은 재생가능 매핑을 통해 유도된 데이타들에 대해서는 그것의 기초 데이타에 존재하지 않는 권한을 가질 수 없도록 보장함으로써 유도된 데이타들에 대한 권한을 이용해 기초 데이타를 부적절하게 접근하는 것을 막는다. 재생불가능 매핑의 경우 기초 데이타에 대한 접근 권한이 없더라도 유도된 데이타들에 대해 권한을 부여하는 것은 가능하다. 예를 들어 '봉급' 속성에서 유도된 '평균봉급' 속성의 경우 평균봉급을 가지고 각각의 봉급을 유추해 낼 수 없으므로 이 매핑은 재생 불가능 매핑이고 봉급에 대한 읽기 권한이 없는 권한주체에게도 평균봉급에 대한 권한은 부여할 수 있다.

내용 의존적 권한부여를 적용하기에 알맞은 응용 분야로 전자도서관을 들 수 있다. 전자 도서관에서의 접근 정책은 사용자의 식별성보다 R등급의 비디오는 18세 이상의 사용자들만 접근 가능하다든지 하는 사용자의 자격과 특성에 기초하는 경우가 많고 또한 전자도서관 객

체들에 대해 내용 의존적인 권한부여를 지원하는 것이 필수적이다(예를 들어 총 사용법에 과한 내용을 포함하는 모든 문서들은 18세 이상의 사용자들에게만 허용가능 하도록 제한할 수 있어야 한다.) [10]. Adam[10]는 전자도서관 환경에서 사용자의 자격과 특성에 기초한 유연한 권한부여와 전자도서관 객체에 내한 내용의존적/내용독립적 접근제어를 지원하는 모델을 제안하였다.

Ahad[11]는 내용 기반 권한부여를 제공하기 위해 객체지향 데이타베이스 시스템에서의 메소드를 이용하는 방법을 제시하였다. 이러한 방법은 객체의 내용에 관한 조건을 메소드를 이용하여 표현하였는데 이는 권한의 변화가 메소드 구현의 변화를 초래하는 단점이 있다. 또 다른 방법은 클래스에 대한 권한에 클래스의 애트리뷰트 값에 대한 술어를 이용하여 그 술어를 참으로 만족시키는 인스턴스들에 대해서 권한을 부여하는 것이다. Gude[12]가 제안한 이 방법은 술어를 이용한 클래스의 수평적 분할과 애트리뷰트를 명시한 클래스의 수직적 분할에 권한을 부여할 수는 있으나 그 둘이 어떤 연관관계를 가지지 못한다. 또한 부정 권한을 지원하지 못하며, 권한 연산시 술어의 영향에 관한 분석과 그 해결책을 제시하지 못하는 단점이 있다. 객체지향 데이타베이스를 위한 내용 의존적 권한부여 기법에 관한 기존의 연구들은 위에서 언급한 문제점들을 지니고 있으므로 이를 해결하기 위한 연구가 필요하다.

3. 내용 의존적 권한부여 모델

본 절에서는 2절에서 언급한 ORION의 권한부여 모델을 기초로 하여 술어를 사용한 내용 의존적 권한부여 모델을 제안하기로 한다. 권한의 정의를 기존의 5-tuple에서 (s, o, m, as, at, p, aset, mset)의 8-tuple로 확장하였다. 확장된 권한의 형식적인 정의는 다음과 같다.

정의 3.1 권한(authorization) a를 (s, o, m, as, at, p, aset, mset)의 8-tuple로 정의한다.

s ∈ S, S는 시스템내의 권한 주체(Authorization Subject)들의 집합;

o ∈ O, O는 시스템내의 권한 객체(Authorization Object)들이 집합;

m ∈ M, M은 접근 모드(Access Mode)의 집합, {R, W}

as ∈ AS, AS는 권한 부호(Authorization Sign)의 집합, {+, -};

at ∈ AT, AT는 권한 타입(Authorization Type)의 집합으로 {wk, st};

p, p는 'True', 'False' 또는 임의의 CNF(Conjunctive Normal Form) 논리식의 술어;

aset ⊂ (Aset ∪ {'All'}), Aset은 클래스 o의 애리뷰트들의 전체 집합;

mset ⊂ (Mset ∪ {'All'}), Mset은 클래스 o의 메소드들의 전체 집합이다.

s, o, m, as, at는 2절에서와 마찬가지로 각각 권한 주체, 권한 객체, 접근 모드, 권한 부호, 권한 타입을 나타내며 p는 'True', 'False' 또는 임의의 CNF(Conjunctive Normal Form)논리식인 술어이고, aset은 클래스 o의 애트리뷰트의 부분 집합이거나 모든 애트리뷰트를 나타내기 위한 'All이며 mset은 클래스 o의 메소드의 부분 집합으로 o의 모든 메소드를 나타내기 위해 'All'의 값을 가질 수 있다. 그 의미는 권한 객체 o가 클래스라면 권한 주체 s가 술어 p를 참으로 만드는 클래스 o의 인스턴스들에 대해 aset 에 속하는 애트리뷰트들과 mset에 속하는 메소드들에 대해 접근 모드 m, 권한 부호 as, 권한 타입 at의 권한을 가짐을 뜻한다. 만약 권한 객체 o가 클래스가 아니라면 술어 p는 항상 True, 애트리뷰트 집합 aset과 메소드 집합 mset은 항상 'All'의 값을 가지며 p, aset, mset 세 필드는 의미가 없다. S(a), O(a), M(a), AS(a), AT(a), P(a), Aset(a)와 Mset(a)는 각각 권한 a의 권한 주체, 권한 객체, 접근 모드, 권한 부호 권한 타입, 술어, 애트리뷰트 집합, 메소드 집합을 나타낸다. 객체의 값에 상관없이 유효한 내용 독립적인 권한은 객체의 값에 대한 조건을 나타내는 술어 p를 항상 참을 나타내도록 'True'로 함으로써 표현 가능하다. 본 논문에서 메소드에 대한 권한은 메소드 내부에서 접근하는 객체나 메소드 내부에서 실행시키는 다른 메소드들에 대한 권한까지 필요로 하지는 않는다. 본 논문은 객체지향의 중요 개념인 객체 은닉(encapsulation)에 대해 충분히 고려하지 않았으며 따라서 객체에 대한 모든 접근이 메소드를 통해서만 가능한 완벽히 은닉화된 시스템에는 적합하지 않다.

위의 정의3.1에 따르면 클래스에 대한 권한이 실제로 적용되는 대상은 권한 객체인 클래스 전체가 아니라 전체 인스턴스 중 명시된 조건을 만족하는 부분 집합이 될 수 있는데 이것을 실 권한 객체(Actual Authorization Object)라 정의하고 이와 대칭적으로 정의 3.1에서의 권한 객체를 형식 권한 객체(Formal Authorization Object)라 부르기로 한다. 권한 a의 실권한 객체는 AO(a)로 나타내기로 한다.

정의 3.2 어떤 권한이 실제로 적용되는 객체를 그 권한의 실 권한 객체(*Actual Authorization Object*)라 정의한다.

형식 권한 객체가 시스템이나 데이타베이스일 경우 실 권한 객체는 형식 권한 객체와 동일하나 형식 권한 객체가 클래스일 경우는 실 권한 객체는 형식 권한 객체의 부분집합이 될 수 있다. 어떤 클래스에 대한 권한의 실 권한 객체는 클래스의 인스턴스들의 값이 변화함에 따라 변화할 수 있다. 만약 어떤 권한의 술어가 항상 'False'의 값을 가지거나 애트리뷰트 집합과 메소드 집합이 공집합이라면 그 권한의 실 권한 객체는 공집합이 되며 이러한 권한의 무의미하다고 한다. 이러한 무의미한 권한은 권한의 집합에 저장하지 않도록 하여 저장 공간의 낭비와 권한 연산시의 비교 횟수를 줄일 수 있게 한다.

권한 주체와 권한 객체가 동일한 임의의 두 권한 사이의 실 권한 객체 사이의 관계는 무관, 동일, 포함, 피포함 또는 교차하는 경우로 나눌 수 있으며 이러한 관계는 두 권한의 술어, 애트리뷰트 집합, 메소드 집합 사

(a) 무관(disjoint) : 두 권한 A와 B의 실 권한 객체 사이에 교집합이 있을 수 없는 경우

(P(A) AND P(B) = False) OR ((Aset(A) ∪ Mset(A)) ∩ (Aset(B) ∪ Mset(B)) = ∅)

=

(P(A) AND P(B) = False) OR ((Aset(A) ∩ Aset(B) = ∅) AND (Mset(A) ∩ Mset(B) = ∅))

(b) 동일(equal) : 두 권한 A와 B의 실 권한 객체가 항상 동일한 경우

((P(A) = P(B)) AND ((Aset(A) ∪ Mset(A)) = (Aset(B) ∪ Mset(B)))

=

((P(A) AND NOT P(B) = False) AND (NOT P(A) AND P(B) = False)) AND

((Aset(A) = Aset(B)) AND (Mset(A) = Mset(B)))

(c) 포함(include) : 두 권한 A와 B의 실 권한 객체가 동일하지 않고 A의 실 권한 객체가 B의 실 권한 객체를 항상 포함하는 경우

NOT (b) AND (P(A) ⊃ P(B)) AND ((Aset(A) ∪ Mset(A)) ⊃ (Aset(B) ∪ Mset(B)))

=

NOT (b) AND

(NOT P(A) AND P(B) = False) AND ((Aset(A) ⊃ Aset(B)) AND (Mset(A) ⊃ Mset(B)))

(d) 피포함(included) : 두 권한 A와 B의 실 권한 객체가 동일하지 않고 B의 실 권한 객체가 A의 실 권한 객체를 항상 포함하는 경우

NOT (b) AND (P(B) P(A)) AND ((Aset(B)) Mset(B)) (Aset(A) Mset(A)))

=

NOT (b) AND (P(A) AND NOT P(B) = False) AND

((Aset(B) Aset(A)) AND (Mset(B) Mset(A)))

(e) 교차(overlap) : 두 권한 A와 B의 실 권한 객체가 동일하거나 포함/피포함의 관계에 있지 않고 무관하지 않은 경우.

NOT (a) AND NOT (b) AND NOT (c) AND NOT(d) AND NOT(e)

그림 1 실 권한 객체 사이의 관계

이의 관계를 비교하여 그림 1과 같이 판별 가능하며 그 구체적인 알고리즘은 [8]에 제시되어 있다. 두 권한의 술어를 논리곱한 결과가 항상 거짓이거나 애트리뷰트 집합의 교집합이 공집합이거나 메소드 집합이 공집합인 경우 두 권한의 실 권한 객체 사이에는 교집합이 존재할 수 없으며 이 경우 두 권한이 무관하다고 정의한다. 애트리뷰트 집합과 메소드 집합이 동일하고 술어가 동일한 경우(권한 객체 클래스에 두 술어를 만족시키는 인스턴스들이 항상 동일한 경우)는 두 권한의 실 권한 객체가 항상 동일하며 이 때 두 권한을 동일하다고 정의한다. 두 권한이 동일하지 않고 한 권한의 실 권한 객체가 항상 다른 실권한 객체를 포함하는 경우를 각각 포함, 피포함의 관계로 정의하며 앞의 네 가지 경우에 모두 속하지 않는다면, 두 권한의 실 권한 객체는 교집합이 공집합이 아닐 가능성이 있으며 (비록 객체의 값이 변화하고 새로운 객체가 클래스에 삽입되고 삭제됨에 따라 어떤 순간에는 교집합이 없을 수도 있지만) 이 경우 두 권한이 교차한다고 한다.

객체지향 모델은 기존의 클래스를 세분화(specialization)하여 새로운 하위 클래스를 생성하는 것을 지원함으로써 재사용(reuse)을 장려한다. 이것은 객체지향 데이타베이스 시스템에서도 예외가 아니며 따라서 권한부여 모델에서도 이러한 재사용의 이점을 저하시키지 않도록 주의해야 한다. 하위 클래스의 인스턴스에 대한 권한에 있어서는 두 가지 상반된 견해가 있다[2]. 그 하나는 어떤 클래스의 생성자에게 그 클래스로부터 유도된 모든 하위 클래스의 인스턴스들에 대해 묵시적 권한을 갖게 하는 것이다. 이 경우 클래스 C에 대해 접근 권한을 가진 사용자의 경우 클래스 C 뿐 아니라 그 클래스의 하위 클래스 계층을 포함하는 질의를 모두 수행 할 수 있는 권한을 갖는다. 이러한 입장의 단점은 기존의 클래스에서 하위 클래스을 유도하여 사용하려는 사용자들의 경우 자신이 생성한 하위 클래스의 인스턴스들에 대해 상위 클래스에 대한 권한을 가지는 사용자들로부터 프라이버시를 지킬 수 없는 단점이 발생한다. 두 번째 입장은 어떤 클래스의 생성자에게 그 클래스로부터 파생된 하위 클래스의 인스턴스들에 대해서 묵시적 권한을 갖지 않도록 하는 것이다. 즉 클래스 계층을 따라서 권한의 상속이 발생하지 않도록 함으로써 다른 클래스로부터 파생된 클래스의 인스턴스들을 상위 클래스에 대한 권한을 갖는 사용자들로부터 보호할 수 있다. 본 논문에서는 재사용성보다는 안전성(security)을 중요시하여 클래스 계층을 따라 묵시적으로 권한의 상속이 일어나지 않도록 하는 후자를 선택하였다. 객체지향 데이타베이스를 위한 모델은 아니지만 유도된 데이타들에 대한 권한에 대한 좀 더 자세한 내용은 Alturi[8]을 참고하기 바란다.

그림 2는 6개의 인스턴스와 dept, name, birthday, score1, score2 5개의 attribute와 compute_age(), compute_ave() 두 개의 메소드를 가지는 클래스 Student를 위와 같이 2차원 테이블로 표시한 그림이다. 술어 필드는 수평적 조건을 나타내고 애트리뷰트 집합과 메소드 집합 필드는 수직적 조건을 나타내므로 그림 2와 같이 2차원으로 표현할 수 있다.

그림 3은 클래스 Student에 대한 교차하는 두 권한 (user1, Student, R, +, st, dept = CS, {name, ssn, gpa, birthday}, {compute_age, compute_gpa})와 (user1, Student, W, -, st, sex = F, {name, ssn, score}, {compute_age})의 실 권한 객체들 사이의 관계를 추상화하여 나타낸 그림이다.

암시 규칙 역시 두 권한의 실 권한 객체 사이의 관계를 고려하여 수정되어야 한다. 사용자 u가 클래스 Student의 age 애트리뷰트의 값이 20이하인 인스턴스들에 대해 강성 긍정 쓰기 권한 A1 = (u, Student, W, +, st, age<=20, 'All', 'All')을 가지고 있고 사용자 u는 클래스 Student에 대해 다른 어떤 명시적 또는 묵시적 권한도 가지고 있지 않다고 가정하자. 이 때 사용자 u가 Student클래스의 age 애트리뷰트 값이 23이상인 인스턴스들에 대해 읽기 접근을 요청했다면 이 접근 요청은 (u, Student, R, age >= 23, 'All', 'All')로 표현할 수 있다. 권한 A1은 age가 20이하인 인스턴스들에 대한 긍정 읽기 권한을 암시할 수는 있지만 age가 20을 초과하는 인스턴스들에 대해서는 어떠한 권한도 암시하지 못하므로 접근 요청은 거절되어야 한다. 그러나 실 권한 객체 사이의 관계를 고려하지 않고 Orion에서의 암시 규칙을 그대로 적용한다면 (u, Student, W, +, st)이 (u, Student, R, +, st)을 암시하므로 접근이 허용되는 잘못된 결과를 초래한다. 따라서 본 모델에서 권한 A_1이 권한A_2를 암시한다 함은 ORION의 암시 규칙의 조건을 만족함과 동시에 A_1의 실 권한 객체가 A_2의 실 권한 객체를 포함하던가 동일한 관계임을 만족하여야 한다[8].

4. 권한 연산

본 절에서는 3절에서 제안한 모델에서의 권한 연산들에 대해 설명한다. 기본적인 권한 연산은 권한 허가(grant), 권한 취소(revoke), 권한 검사(check)의 세 가지가 있다[1]. 내용 독립적 모델에서의 권한 연산들은 3

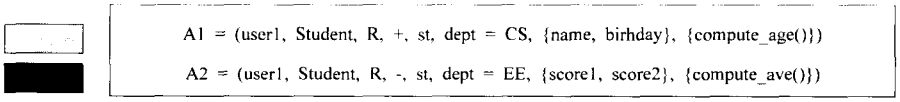
절에서 제안한 모델에 그대로 적용되기에 부적합하며 실 권한 객체사이의 관계를 고려하여 새로운 모델에 맞게 그 의미를 재정의 하였다.

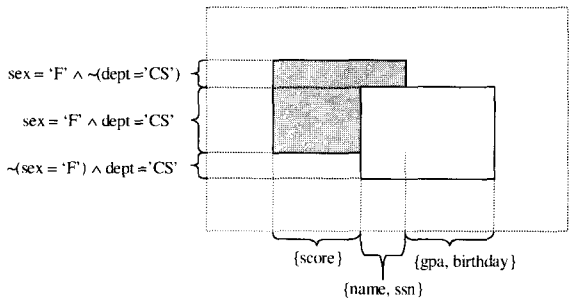
4.1 권한 허가 연산

권한의 명시적 금지를 나타내는 부정 권한을 지원하는 경우 동일한 객체에 대해 긍정 권한과 부정 권한이 동시에 존재하게 된다면 권한의 충돌이 발생할 수 있다. 이 경우 저장된 권한 규칙들의 집합에 충돌하는 권한을 동시에 존재하는 것을 허용하지 않거나 충돌하는 권한이 상존하는 것을 허용하고 권한 검사 연산 시에 이를 해결(resolve)하는 두 가지 입장이 있다. 본 논문에서는 권한 허가 연산 시에 권한들간의 충돌을 해소하여 권한 규칙의 집합에 일관성을 유지하면서 권한 검사 연산의 부담을 줄이도록 하였다. 대체로 권한 검사 연산이 권한 허가 연산에 비해 매우 빈번하므로 전체적인 성능을 고려하였을 때 이 방법이 더 좋은 결과를 얻을 수 있다.

	dept	name	birthday	score1	score2	compute_age()	compute_ave()
inst[1]	CS	Joe	...	...	...	...	...
inst[2]	CS	...	...	...	...	...	...
inst[3]	CS	...	...	...	...	...	...
inst[4]	EE	...	...			...	
inst[5]	EE	...	...			...	
inst[6]	EE	...	...			...	

그림 2 student 클래식의 2차원적 표현





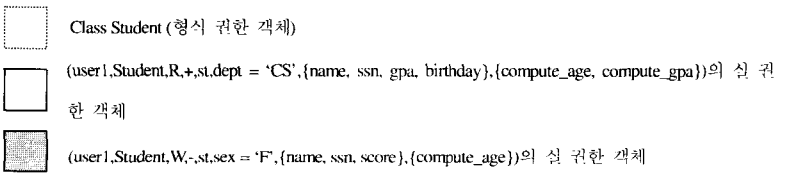


그림 3 실 권한 객체의 2차원적 표현

사용자 s가 클래스 Student의 dept 애트리뷰트의 값이 'CS'인 인스턴스들에 대해 강성 긍정 읽기 권한 (s, Student, R, +, st, dept = 'CS', 'All', 'All')을 가지고 있다고 가정하자. 내용 독립적 권한부여 모델의 경우 사용자 s에게 Student 클래스에 대한 어떤 권한을 허가하려는 시도도 강성 권한의 정의에 의해 충돌이 발생하여 추가 권한은 거절되어야 한다. 그러나 본 모델에서는 기존 권한과 추가 권한의 실 권한 객체들 사이의 관계에 따라 추가 권한이 허가될 수도 있다. 예를 들어 (s, Student, R, -, st, dept = 'EE', 'All', 'All')를 허가하려할 경우 기존 권한과 추가 권한의 실 권한 객체가 무관하므로 실제로 충돌이 일어나지 않으므로 추가 권한은 허가되어야 한다.

기존 권한과 추가 권한의 실 권한 객체가 교차하는 경우 두 권한의 실 권한 객체의 교집합 부분에서만 충돌이 일어날 수 있다. 이 경우 실제 충돌이 일어나지 않는 부분에 대해 권한 허가를 허용하는 연산의 부분 성공을 허용하는 입장과 허용하지 않는 두 가지 입장이 있을 수 있다. 두 방법은 나름대로의 장단점이 있는데 본 논문에서는 전자를 강성 권한 허가 연산, 후자를 약성 권한 연산이라 하고 이 둘을 모두 지원하여 권한 허가자가 임의로 선택할 수 있도록 한다. 제안한 두 가지 권한 허가 연산 모두에서 권한 객체와 주체가 동일한 권한 집합에 속하는 임의 의 두 권한의 실 권한 객체들은 항상 무관한 관계가 되도록 하므로 임의의 실 권한 객체에 대해 두 개 이상의 명시적 권한이 적용될 수 없다. 이에 따라 어떤 객체에 대해서도 모순되는 권한이 존재할 수 없게되어 기존 연구와는 달리 부정 권한을 포함하더라도 권한 규칙의 집합을 일관성 있게 유지할 수 있으며 권한 검사 연산 시에 충돌 해결을 할 필요가 없게 한다.

권한 허가 연산 시에 술어를 제외한 모든 필드가 동일한 두 권한은 두 권한의 술어를 논리합한 술어를 가지는 하나의 권한으로 대치하여 저장할 수 있다. 또한 애트리뷰트 집합과 메소드 집합을 제외한 모든 필드가 동일한 두 권한은 애트리뷰트 집합과 메소드 집합을 각각 합집합으로 연결한 하나의 권한으로 대치하여 저장하여 명시적 권한의 수를 줄이도록 한다. 이러한 권한의 결합을 통해 명시적 권한의 수를 가능한 한 줄이도록 하여 저장 공간을 절약하고 권한 연산 시에 비교해야하는 권한의 수를 줄여 권한 연산의 효율성을 높이도록 한다.

규칙 4.1 (권한의 결합)

(1) 술어를 제외한 모든 필드가 동일한 두 권한 (s, o, m, as, at, p_1, aset, mset)과 (s, o, m, as, at, p_2, aset, mset)은 (s, o, m, as, at, p_1 OR p_2, aset, mset)으로 술어를 논리합으로 결합한 하나의 권한으로 표현할 수 있다.

(2) 애트리뷰트 집합과 메소드 집합을 제외한 모든 필드가 동일한 두 권한 (s, o, m, as, at, p, $aset_1$, $mset_1$)과 (s, o, m, as, at, p, $aset_2$, $mset_2$)는 (s, o, m, as, at, p, aset1 ∪ $aset_2$, $mset_1$ ∪ $mset_2$)로 애트리뷰트 집합과 메소드 집합을 각각 합집합으로 연결한 하나의 권한으로 표현할 수 있다.

4.1.1 강성 권한 허가 연산

권한을 허가하기 위해서는 일단 권한 규칙의 집합에 있는 기존 권한들과 충돌 여부를 탐지해야한다. 기존 권한과 실 권한 객체가 동일하다면 내용 독립적 모델에서와 같이 두 권한의 우선 순위에 따라 추가 권한을 허가할 것인가 거절할 것인가를 결정하고 무관하다면 일단 충돌이 없는 것으로 간주하고 다른 기존 권한들과의 충돌 여부를 계속적으로 탐지하도록 한다.

실 권한 객체가 교차, 포함 또는 피포함의 관계라면 다음의 원칙에 따라 최소의 수의 권한들이 생성되도록 추가 권한 또는 기존 권한을 분할한다. 강성 권한 허가 연산은 하나의 추가 권한을 어떤 기존 권한과 실 권한 객체가 무관하지 않은 부분과 무관한 부분으로 분할한 각각의 부분 권한들에 대한 강성 권한 허가 연산들로 재귀적으로 처리한다. 권한의 분할은 규칙 4.2를 따른다.

규칙 4.2 (권한의 분할)

권한 주체와 권한 객체가 동일한 임의의 두 권한 A_1=(s, o, m_1, as_1, at_1, p_1, $aset_1$, $mset_1$)과 A_2=(s, o, m_2, as_2,, at_2, p_2,, $aset_2$, $mset_2$,)에 대해서 A_1을 분할하여 그 분할된 결과들이 A_2와 무관, 동일 또는 포함의 관계가 되도록 다음과 같이 분할할 수 있다. toCNF는 임의의 WFF(Well Formed Formula)형태의 술어를 CNF로 바꾸어 주는 함수라고 가정한다.

(1) A_1과 A_2의 실권한 객체 사이의 관계가 동일, 무관하거나 A_2의 실권한 객체가 A_1의 실권한 객체를 포함하는 경우는 분할이 일어나지 않는다.

(2) A_1의 실권한 객체가 A_2의 실권한 객체를 포함하거나 교차하는 경우

① $P(A_1)$와 $P(A_2)$가 동일한 관계일 경우.

A_{11} = (s, o, m_1, as_1, at_1, p_1, $aset_1$ ∩ $aset_2$, $mset_1$ ∩ $mset_2$) ,

A_{12} = (s, o, m_1, as_1, at_1, p_1, $aset_1$ - $aset_2$, $mset_1$ - $mset_2$)의 두 개의 권한으로 분할된다.

procedure new_grant (grant_request, result)
input : grant_request (s, o, m, as, at, p, aset, msset)
output : result ∈ {True, False, PartialTrue}
begin
step 1. 추가하려는 권한 grant_request의 권한 객체 계층의 상위 노드나 하위 노드에 있는 기존 강성 권한들과 충돌이 있다면 result에 False를 리턴하고 끝낸다.
step 2. 권한을 추가하려는 노드에 있는 기존 권한들과 각각 충돌이 있는지 다음과 같은 방법으로 조사한다.
(a) 두 권한의 실권한 객체 사이의 관계가 무관할 경우 : 무시하고 다른 권한과 비교한다.
(b) 두 권한의 실권한 객체가 동일할 경우 : 충돌이 없다면 추가 권한을 허가하고 result에 True를 리턴하고 끝내며 충돌이 있다면 result에 False를 리턴하고 끝낸다.
(c) 기존 권한의 실권한 객체가 추가 권한의 실 권한 객체를 포함하는 경우 : 충돌이 있다면 추가 권한을 거절하고 result에 False를 리턴하고 끝내며 충돌이 없는 경우 추가 권한을 허가하고 기존 권한을 취소하며 기존 권한을 분할하여 추가 권한과 무관한 권한을 구하여 그 권한을 허가하며 result에 Partial True를 리턴하고 끝낸다.
(d) 추가 권한의 실 권한 객체가 기존 권한의 실 권한 객체를 포함하거나 교차하는 경우 : 추가 권한을 분할하여 분할된 추가 권한들에 대해 new_grant를 반복한다. 모든 추가 권한들의 허가 연산 결과가 True이면 result에 True를 리턴하고 끝내고 모든 추가 권한들의 허가 연산 결과가 False이면 result에 False를 리턴하고 끝내고 그 외의 경우는 result에 PartialTrue를 리턴하고 끝낸다.
step 3. 모든 기존 권한들에 대해 disjoint하다면 추가 권한을 허가하고 result에 True를 리턴하고 끝내며 이때 결합 가능한 기존 권한이 있다면 결합하여 저장한다.
end

그림 4 강성 권한 허가 연산의 알고리즘

② $Aset(A_1)$와 $Aset(A_2)$가 동일하고 $Mset(A_1)$와 $Mset(A_2)$가 동일한 경우
A_{11} = (s, o, m_1, as_1, at_1, p_1 AND p_2, $aset_1$, $mset_1$) ,
A_{13} = (s, o, m1, as_1, at_1, toCNF(p_1 AND NOT p_2), $aset_1$, $mset_1$)의 두 개의 권한으로 분할된다.

③ 그 외의 경우
A_{11} = (s, o, m_1, as_1, at_1, p_1 AND p_2, $aset_1 \cap aset_2$, $mset_1 \cap mset_2$) ,
A_{12} = (s, o, m_1, as_1, at_1, p_1 , $aset_1 - aset_2$, $mset_1 - mset_2$),
A_{13} = (s, o, m_1, as_1, at_1, toCNF(p_1 AND NOT p_2), $aset_1$ $aset_2$, $mset_1$ $mset_2$)의 세 개의 권한으로 분할된다.

강성 권한 허가 연산의 알고리즘은 그림 4와 같다.

4.1.2 약성 권한 허가 연산

강성 권한 허가 연산은 허가 요청한 권한을 기존 권한과 모순이 생기지 않는 한에서 최대한 허가할 수 있는 장점이 있지만 추가 권한이나 기존 권한이 권한 허가자의 의도와 달리 분할될 수 있고 이에 따라 저장되어야 할 명시적 권한의 수가 늘어날 수 있는 단점이 있다. 권한의 분할과 부분 성공을 허용하지 않는 약성 권한 연산은 추가 권한이 모든 기존 권한들과 실 권한 객체가 무관한 관계이거나 실 권한 객체가 동일하고 기존 권한보다 추가 권한의 우선 순위가 높은 경우에만 허가한다. 권한의 분할이 일어나지 않으므로 한 번의 권한 허가 연산으로 저장될 수 있는 권한은 최대 하나가 된다. 약성 권한 허가 연산의 알고리즘은 매우 단순하므로 언급하지 않기로 한다.

4.2 권한 취소 연산

기존의 내용 독립적 모델에서의 권한 취소 연산의 의미에 따르면 취소하려는 권한과 동일한 명시적 권한이 존재할 때만 연산이 성공하고 그 외에는 실패로 간주하여야 한다. 이것을 본 모델에 적용하려면 권한 취소자가 취소하려는 권한의 술어, 애트리뷰트 집합, 메소드 집합을 정확히 기억하고 일일이 명시해 주어야 한다. 뿐만 아니라 어떤 사용자의 특정 클래스에 대한 모든 권한을 취소해야 할 경우는 그 사용자의 해당 클래스에 대한 모든 권한들에 대해 위의 절차를 반복해 주어야 하는 불편이 있다.

권한 취소 연산은 두 가지 입장이 있을 수 있다. 첫째로 취소 요청한 권한과 실 권한 객체를 제외한 나머지 부분(권한 주체, 권한 객체, 접근 모드, 권한 타입, 권한 부호)이 동일하고 실 권한 객체가 무관하지 않은 기존 권한이 존재할 경우 이 권한과 취소 요청한 권한의 실 권한 객체들의 교집합에 속하는 객체들에 대해서만 권

```
procedure new_revoke (revoke_request, result)
input  : revoke_request (s, o, m, as, at, p, aset, msset)
output : result ∈ {True, False, PartialTrue}
begin
step 1. 취소 요청한 권한 revoke_request와 권한 주체, 권한 객체, 접근 모드, 권한 부호, 권한 타입이 동일한 기존
        권한들을 구한다. 그러한 기존 권한이 없다면 result에 False를 리턴하고 끝낸다.
step 2. 위의 step 1에서 구한 기존 권한들에 대해 취소 요청한 권한 revoke_request와의 관계에 따라 다음을
        수행한다.
        (a) 두 권한의 실권한 객체 사이의 관계가 무관할 경우 : 무시하고 다른 기존 권한과 비교한다.
        (b) 두 권한의 실권한 객체가 동일할 경우 : 기존 권한을 취소하고 result에 True를 리턴하고 끝낸다.
        (c) 기존 권한의 실권한 객체가 취소 요청한 권한의 실 권한 객체를 포함하는 경우 : 기존 권한을 분할하여
            취소 요청 권한과 무관한 부분은 유지하고 교차하는 부분을 취소하며 result에 Partial True를 리턴하고 끝낸다.
        (d) 취소 요청 권한의 실 권한 객체가 기존 권한의 실 권한 객체를 포함하거나 교차하는 경우 : 취소 요청
            권한을 분할하여 분할된 권한들에 대해 new_grant를 반복한다. 모든 취소 연산 결과가 True이면 result에 True를
            리턴하고 끝내고 모든 취소 연산 결과가 False이면 result에 False를 리턴하고 끝내고 그 외의 경우는
            result에 PartialTrue를 리턴하고 끝낸다.
end
```

그림 5 강성 권한 취소 연산의 알고리즘

한을 취소하고 교집합에 속하지 않는 부분에 대한 권한은 그대로 유지하는 방법이 있다. 두 번째로 취소 요청한 권한의 실 권한 객체에 포함되는 범위의 기존 권한들을 모두 취소하는 방법이 있는데 이것은 한 번의 권한 취소 연산으로 여러 권한을 동시에 취소할 수 있는 편리함이 있지만 원치 않는 권한들까지 취소될 수 있는 side effect가 있다. 본 논문에서는 권한 취소자의 의도에 따라 적절한 방법을 선택할 수 있도록 위의 두 가지 입장을 모두 지원해 주기로 하며 전자를 약성 권한 취소 연산, 후자를 강성 권한 취소 연산이라 하였다. 그림 5는 강성 권한 취소 연산의 알고리즘이다.

그림 6은 강성 권한 취소 연산의 요청과 그 결과를 나타내 준다. 어떤 사용자의 어떤 클래스에 대한 권한 B를 취소하려 할 때 그 사용자가 그 클래스에 대해 권한 B와 실 권한 객체만 다른 A_1, A_2, A_3의 기존 권한을 가지고 있는 경우이다. 오른쪽 그림은 그 결과를 나타낸다. 취소 요청한 권한에 실 권한 객체가 포함되는 권한 A_1은 완전히 취소되고 실 권한 객체가 무관한 권한 A_3는 취소 되지 않는다. 취소 요청 B와 교차하는 A_2는 분할되어 교차하는 부분은 취소되고 교차하지 않는 부분은 취소되지 않아 결과적으로 A_2는 부분 취소된다.

4.3 권한 검사 연산

사용자 프로그램이나 질의 언어로부터의 접근 요청은 그 요청이 타당한가를 판단하기 위해 명시적으로 저장되어 있는 권한들과 비교되어야 하는데 이것은 권한 검사 연산에 의해 수행된다[2]. 내용 독립적 모델에서와 같이 접근 요청을 암시하는 긍정 권한이 존재할 경우 접근 요청을 허락하며 접근 요청을 암시하는 부정 권한이 있을 때에는 접근 요청을 거부한다[2]. 그러나 본 모델에서의 암시 규칙은 두 권한의 실 권한 객체 사이의

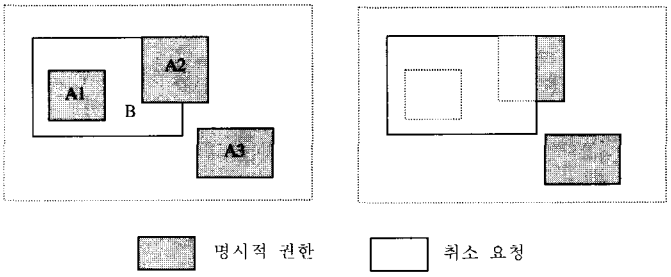


그림 6 강성 권한 취소 연산과 그 결과

관계를 고려하여 확장되어야 한다[8]. 본 논문에서 권한의 암시는 권한 객체와 접근 모드의 두 차원을 따라 일어난다. 권한 주체의 그룹이나 역할(role)계층에 따른 권한의 암시는 고려하지 않았으나 충분히 확장 가능하다. 암시 규칙은 다음과 같다.

규칙 4.3 (암시 규칙)

(1) 강성 권한의 암시 규칙

① 두 강성 긍정 권한 A_1과 A_2에 대해 만약 $S(A_1) = S(A_2)$, $O(A_1) \geq O(A_2)$, $M(A_1) \geq M(A_2)$이고 $AO(A_1)$가 $AO(A_2)$를 포함하거나 서로 동일하다면, $A_1 \rightarrow A_2$ 이라 표시하고 강성 긍정 권한 A_1이 강성 긍정 권한 A_2를 암시한다고 한다.

② 두 강성 부정 권한 A_1과 A_2에 대해 만약 $S(A_1) = S(A_2)$, $O(A_1) \geq O(A_2)$, $M(A_2) \geq M(A_1)$이고 $AO(A_1)$가 $AO(A_2)$를 포함하거나 서로 동일하다면, $A_1 \rightarrow A_2$ 이라 표시하고 강성 부정 권한 A_1이 강성 부정 권한 A_2를 암시한다고 한다.

(2) 약성 권한의 암시 규칙

① 두 약성 긍정 권한 A_1과 A_2에 대해 만약 $S(A_1) = S(A_2)$, $O(A_1) \geq O(A_2)$, $M(A_1) \geq M(A_2)$ 이고 $AO(A_1)$가 $AO(A_2)$를 포함하거나 서로 동일하고 A_2와 권한 타입만 다른 강성 권한 ($S(A_2)$, $O(A_2)$, $M(A_2)$, $AS(A_2)$, st, $P(A_2)$, $Aset(A_2)$, $Mser(A_2)$)을 암시하는 강성 권한이 권한 집합에 존재하지 않고 그러한 강성 권한과 권한 타입만 다른 약성 권한이 권한 집합에 존재하지 않는다면, $A_1 \rightarrow A_2$ 이라 표시하고 약성 긍정 권한 A_1이 약성 긍정 권한 A_2를 암시한다고 한다.

② 두 약성 부정 권한 A_1과 A_2에 대해 만약 $S(A_1) = S(A_2)$, $O(A_1) \geq O(A_2)$, $M(A_2) \geq M(A_1)$ 이고 $AO(A_1)$가 $AO(A_2)$를 포함하거나 서로 동일하고 A_2와 권한 타입만 다른 강성 권한 ($S(A_2)$, $O(A_2)$, $M(A_2)$, $AS(A_2)$, st, $P(A_2)$, $Aset(A_2)$, $Mser(A_2)$)을 암시하는 강성 권한이 권한 집합에 존재하지 않고 그러한 강성 권한과 권한 타입만 다른 약성 권한이 권한 집합에 존재하지 않는다면, $A_1 \rightarrow A_2$ 이라 표시하고 약성 부정 권한 A_1이 약성 부정 권한 A_2를 암시한다고 한다.

본 모델에서의 사용자 프로그램이나 질의 언어로부터의 접근 요청은 (s, o, m, p, aset, mset)의 형태이다. s는 접근 요청한 의 경우 본 모델은주체, o는 접하려는 객체, m은 접근 모드, p는 임의의 CNF형태의 술어, aset은 애트리뷰트 집합, mset은 메소드 집합을 나타낸다. 접근 요청한 객체 o가 클래스가 아니라면 p는 항상 True이고 aset과 mset은 상수 'All'의 값을 가지며 아무 의미가 없다. 이 경우 사용자 s가 객체 o를 접근모드 m으로 접근하려 함을 의미한다. o가 클래스라면 사용자 s가 술어 p를 만족하는 클래스 o의 aset에 속하는 애트리뷰트들과 mset에 속하는 메소드들을 접근모드 m으로 접근하기를 요청함을 나타낸다. 내용 독립적 모델에서와 같이 접근 요청을 암시하는 긍정 권한이 있을 때에는 접근 요청을 허용하며 접근 요청을 암시하는 부정 권한이 있을 때에는 접근 요청을 거부한다. 접근 요청을 암시하는 기존 권한이 있는 지를 판단하기 위해 실 권한 객체들 사이의 관계를 고려하여 확장된 규칙 4.3의 암시 규칙을 적용하도록 한다.

만약 기존 권한의 집합에 접근 요청을 암시하는 긍정 권한이나 부정 권한이 존재하진 않지만 그러한 권한들과 실권한 객체가 무관하지 않은 권한이 있을 경우에는 권한 허가 연산과 권한 취소 연산과 마찬가지로 역시 두 가지 입장이 있을 수 있다. 첫 번째로 부분적으로 접근 요청의 허용이나 금지를 나타내는 기존 권한이 있을 경우 접근 요청을 분할하여 접근을 요청한 사용자가 가지고 있는 권한 만큼의 데이타를 정확히 접근할 수 있도록 해주는 방법이 있을 수 있다. 이러한 방법은 권한 검사 연산의 보안성과 정확성을 동시에 최대로 만족시켜준다. 즉 어떤 사용자가 시스템의 권한부여 정책에 의해 접근해서는 안 되도록 명시된 객체들에 대해서는 접근을 금지시킬 수 있게 하여 권한 검사 연산의 보안성을 보장하며 접근 할 수 있도록 허용된 객체들에 대한 접근 요청이 거절되지 않도록 하여 과잉 보안이 되지 않고 정확한 권한 검사가 이루어지도록 한다. 그러나 이러한 방법은 사용자의 접근 요청이 분할될 수 있으며 이것은 때로는 사용자가 원치 않는 결과를 초래할 수가 있는 부작용이 있다. 두 번째 방법은 부분 접근을 허용하지 않고 내용 독립적 모델에서의 권한 검사 연산의 의미를 그대로 적용하는 것이다. 즉 접근 요청을 암시하는 긍정 권한이 기존 권한의 집합에 존재할 경우에만 접근을 허용하고 그 이외의 경우에는 접근을 금지하는 입장이다. 이러한 방법은 권한부여 정책에서 어떤 사용자가 접근하지 못하도록 설정된 객체에 대한 접근은 막아줌으로써 보안성은 보장하나 접근 요청을 부분적으로 만족하는 기존 권한이 있더라도 접근 요청을 완전히 거절함으로써 사용자가 접근할 수 있도록 명시된 객체에 대한 접근도 금지할 수 있게 되어 과잉 보안이 이루어질 수 있는 단점이 있다. 즉 보안성은 만족하나 정확성은 떨어지게 된다. 본 논문에서는 전자의 입장을 강성 권한 사 연산이라 하고 후자를 약성 권한 검사 연산이

```
procedure new_check (access_request, result)
input  : access_request (s, o, m, p, aset, msset)
output : modified_requset_set
begin
step 1. modified_request_set = {}, sub_request_set = {}
step 2. access_requset를 암시하는 강성 긍정 권한이 있다면,
        access_request는 모두 허용되며 modified_requset_set = {access_request}를 저장하고 끝낸다.
step 3. access_requset를 암시하는 강성 부정 권한이 있다면,
        access_request는 모두 금지되며 modified_requset_set = {}를 저장하고 끝낸다.
step 4. 접근하려는 객체 o가 클래스일 경우,
        step 4-1. access_requset를 모든 기존 강성 권한들에 대해 무관, 동일 또는 포함 관계가 될 때까지 분할하여
                  sub_request_set에 저장한다.
        step 4-2. sub_request_set에 속하는 접근 요청들에 대해,
                  만약 해당 접근 요청을 암시하는 강성 긍정 권한이 권한 집합에 존재한다면,
                  그 접근 요청을 sub_request_set에서 삭제하고 modified_request_set에 삽입한다.
                  만약 해당 접근 요청을 암시하는 강성 부정 권한이 권한 집합에 존재한다면,
                  그 접근 요청을 sub_request_set에서 삭제한다.
        step 4-3. sub_request_set이 공집합이라면 modified_requset_set을 리턴하고 끝낸다.
        step 4-4. sub_request_set에 속하는 접근 요청들에 대해,
                  기존 약성 권한들과 무관, 동일 또는 포함 관계가 될 때까지 분할하여 sub_requset_set에
                  저장한다.
        step 4-5. sub_requset_set에 속하는 접근 요청들에 대해,
                  만약 해당 접근 요청을 암시하는 약성 긍정 권한이 권한 집합에 존재한다면,
                  그 접근 요청을 sub_request_set에서 삭제하고 modified_request_set에 삽입한다.
                  만약 해당 접근 요청을 암시하는 약성 부정 권한이 권한 집합에 존재한다면,
                  그 접근 요청을 sub_request_set에서 삭제한다.
        step 4-6. sub_request_set이 공집합이라면 modified_requset_set을 리턴하고 끝낸다.
step 5. access_requset를 암시하는 약성 긍정 권한이 있다면,
        access_request는 모두 허용되며 modified_requset_set = {access_request}를 저장하고 끝낸다.
step 6. access_requset를 암시하는 약성 부정 권한이 있다면,
        access_request는 모두 금지되며 modified_requset_set = {}를 저장하고 끝낸다.
end
```

그림 7 강성 권한 검사 연산의 알고리즘

라 하여 모두 지원해 주도록 한다. 약성 권한 검사 연산의 알고리즘은 암시규칙이 규칙 4.3과 같이 수정되어야 할 뿐 내용 독립적 모델에서의 알고리즘과 동일하다. 강성 권한 검사 연산의 알고리즘은 그림 7과 같다.

5. 결 론

본 논문은 객체지향 데이타베이스를 위한 내용 의존적 권한부여 모델을 제안하였다. 기존의 내용 독립적 권한부여 모델에서의 권한의 정의에 객체의 값에 대한 조건을 나타내는 술어 필드를 추가함으로써 해당 권한이 유효한 조건을 명시할 수 있도록 하였으며 애트리뷰트 집합과 메소드 집합 필드를 추가하여 술어필드를 만족하는 인스턴스들의 집합 중 권한이 유효한 애트리뷰트들과 메소드들을 명시할 수 있게 하였다. 즉 술어 필드는 그 술어를 참으로 만드는 인스턴스들의 집합을 나타내는 수평적 제한 조건을 나타낼 수 있고, 애트리뷰트 집합과 메소드 집합은 그러한 술어를 만족하는 인스턴스들 중 특정 애트리뷰트 집합과 메소드 집합에 대해서만 권한을 부여할 수 있는 수직적 제한 조건을 표현할 수 있다.

본 논문에서 제안한 권한부여 모델은 ORION의 내용 독립적 권한부여 모델을 기초로 하여 확장하였는데, 클래스에 대한 권한에 제한 조건을 명시함으로써 권한의 적용 대상이 그 클래스의 인스턴스 전체가 아니라 명시된 조건을 만족하는 임의의 부분 집합이 될 수 있다. 이로 인해 ORION의 내용 독립적 권한부여 모델에서의

암시 규칙과 연산의 의미와 알고리즘을 그대로 적용할 수가 없게 된다. 본 논문에서는 권한이 실제로 적용되는 대상을 실 권한 객체라 정의하고 권한 주체와 권한 객체가 동일한 임의의 두 권한 사이의 관계를 그들의 실 권한 객체사이의 관계를 토대로 분류하였으며 권한 연산의 의미를 재정의하였다.

제안한 모델은 뷰를 지원하지 않는 객체지향 데이타베이스 시스템에서도 데이타베이스의 내용에 기반한 접근제어를 할 수 있도록 하며 뷰를 이용한 대부분의 권한 부여 기법들과 달리 부정 권한을 지원한다. 또한 권한의 충돌의 해결을 권한 허가 연산 시에 해결하도록 함으로써 권한 검사 연산의 성능을 향상시켰다. 권한에 명시된 조건을 만족하는 객체들을 집단화하여 하나의 권한을 부여할 수 있게 함으로써 각각의 객체에 권한을 허가하고 저장해야하는 수고를 덜고 다양한 보안 요구조건을 나타낼 수 있는 강력한 표현력을 제공하였다.

참 고 문 헌

[1] W.Kim. Authorization, *Introduction to Object-Oriented Databases*. The MIT Press, 1990.

[2] F.Rabitti, E.Bertino, W.Kim, and D.Woelk. A Model of Authorization for Next-Generation Data base Systerms. *ACM Transactions on Database Systems*, Vol. 16, No. 1. Mar. 1991.

[3] E.Fernandez, R.Summers and C.Wood. *Database Security and Integrity*, Addison-Wesley Publi shing Company. 1981.

[4] E.Bertino, P.Samarati, and S.Jajodia. An Extended Authorization Model for Relational Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 1. Mar. 1997.

[5] P.P.Griffiths and B.W.Wade. An Authorization Mechanism for Relational Database System. *ACM Transactions on Database Systems*, Vol. 1, No. 3, pp 242-256. Sep. 1976.

[6] W.Kim. On View Support in Object-Oriented Database Systems, *Modern Database Systems: The Object Model, Interoperability, and Beyond*, pp130-145. ACM Press, 1995.

[7] E.Bertino, F.Origgi, and P.Samarati. A New Authorization Model for Object-Oriented Data bases. *Database Security VIII:Status and Prospects, Elsevier Science* B.V. 1994.
In Proceedings of the 3rd International Confe rence on Extending Database Technology EDBT 92, LNCS, Vol

[8] V.Atluri and A.Gal. An Authorization Model for Temporal and Derived Data: Securing Information Portals. *ACM Transactions on Information and System Security*, Vol.5, No. 1, pp 62-94, Feburary 2002.

[9] A.Gal and V.Atluri. An Authorization Model for Temporal Data. In *Proceedings of the Seventh ACM Conference on Computer and Communi cation Security*, pp 144-153, November 2000.

[10] A Content-Based Authorization Model for Digital Libraries. *IEEE Transactions on Knowledge and Data Engineering*, pp. 296-315, 2002.

[11] R.Ahad, P.Lyngbaek and E.Onuebge. Supporting access control in an object-oriented database language. 2580, pp184-200, Vienna, March. 1992.

[12] E.Gudes, H.Song and E.Fernandez. Evaluation of Negative, Predicate, and Instance-based Autho rization in Object-Oriented Databases. *Database Security IV:Status and Prospects*, Elsevier Science B.V. 1991.

홍 성 림
1997년 이화여자대학교 전자계산학과 학사. 1999년 한국과학기술원 전산학과 석사. 1999년 LG 종합기술원 정보기술연구소. 2002년 (주)데이콤, 천리안. 2002년 (주)데이콤 멀티미디어 인터넷. 관심분야는 객체지향데이타베이스,권한부여

박 창 원
1995년 2월 서강대학교 전자계산학과 (학사). 1997년 2월 한국과학기술원 전산학과 (석사). 2002년 8월 한국과학기술원 전자전산학과 전산학전공 (박사). 현재 LG전자기술원 정보기술연구소 선임연구원. 관심분야는 XML, XQuery, XSLT, GML, LBS, Databases

정 진 완
1973년 서울대학교 공과대학 전기공학과 (학사). 1983년 University of Michigan 컴퓨터공학과(박사). 1983년 ~ 1993년 미국 GM 연구소 전산과학과 선임연구원 및 책임연구원. 1793년 ~ 현재 한국과학기술원 전산학과 부교수 및 교수. 관심분야는 XML, 멀티미디어 데이타베이스, GIS, 웹 정보검색, 객체지향 데이타베이스