관계형 데이타베이스 기반의 RDF 온톨로지 접근 제어 모델

(An RDF Ontology Access Control Model based on Relational Database)

정 동 원 †

(Dongwon Jeong)

요 약 이 논문에서는 관계형 보안 모델 기반의 RDF 웹 온톨로지 접근 제어 모델을 제안한다. 시맨틱 웹은 차세대 웹으로 인식되고 있으며 RDF는 시맨틱 웹을 실현하기 위한 웹 온톨로지 서술 언어이다. RDF와 관련된 많은 연구들이 진행되었으나 대부분 에디터, 저장소 및 추론 엔진 등의 연구에만 집중되었을 뿐 정보 시스템의 가장 중요한 요구 사항 중 하나인 보안 문제에 대한 연구는 매우 미비하다. RDF 온톨로지 보안에 대한 일부 연구들이 제안되었으나 관련 데이타를 모두 메모리에 로딩해야 하는 오버헤드를 지니며, 현재 대부분의 온톨로지 저장소가 관계형 데이타베이스를 기반으로 개발되고 있는 상황을 고려하지 않고 있다. 이 논문에서는 이러한 문제점을 해결하기 위한 관계형 모델 기반의 새로운 RDF 웹 온톨로지 보안 모델을 제안한다. 제안된 보안 모델은 높은 실용성과 활용성을 제공하며, 또한 관계형 보안 모델의 안정성에 기인한 제안 모델의 안정성 확보가 용이하다.

키워드 : RDF, RDF 스키마, 웹 온톨로지, 보안, 접근 제어, RBAC

Abstract This paper proposes a relational security model-based RDF Web ontology access control model. The Semantic Web is recognized as a next generation Web and RDF is a Web ontology description language to realize the Semantic Web. Much effort has been on the RDF and most research has been focused on the editor, storage, and inference engine. However, little attention has been given to the security issue, which is one of the most important requirements for information systems. Even though several researches on the RDF ontology security have been proposed, they have overhead to load all relevant data to memory and neglect the situation that most ontology storages are being developed based on relational database. This paper proposes a novel RDF Web ontology security model based on relational database to resolve the issues. The proposed security model provides high practicality and usability, and also we can easily make it stable owing to the stability of the relational database security model.

Key words : RDF, RDF Schema, Web Ontology, Security, Access Control, RBAC

1. 서 론

시맨틱 웹(Semantic Web)은 컴퓨터 시스템이 웹 정보를 이해하고 지능적으로 처리할 수 있도록 하기 위한 차세대 웹이라 할 수 있다. 시맨틱 웹은 질적으로 향상된 정보 수집과 새로운 지식 생성을 가능하게 한다. 이를 위해서는 HTML이나 XML과 같은 웹 정보 기술 언어와는 달리 지식을 표현할 수 있는 웹 기술 언어가 요구된다.

RDF(Resource Description Language)는 웹 지식을 표현하기 위한 언어로서 W3C에 의해 개발되었다[1,2]. 지금까지 RDF를 기반으로 한 웹 온톨로지의 구축, 편

· 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-311-D0076)

† 종신회원 : 군산대학교 정보통계학과 교수
djeong@kunsan.ac.kr

논문접수 : 2007년 6월 28일
심사완료 : 2007년 12월 23일

정보과학회논문지: 데이타베이스 제35권 제2호(2008.4)

집, 저장 및 활용을 위한 다양한 도구 및 시스템들이 개발되었으며[3-10], 이러한 연구 결과들은 RDF 웹 온톨로지를 보다 용이하게 생성하고 효율적으로 관리할 수 있도록 해 준다. 그러나 RDF 웹 온톨로지에 대한 안전한 관리를 위해서는 온톨로지에 대한 접근 제어 모델 개발이 필수적으로 요구된다.

지금까지 RDB를 기반으로 한 많은 온톨로지 저장소들이 개발되어 왔으며, 대표적인 저장소로서 Sesame[4], Jena[5], Kowari[6], Redland[7], DLDB[8,9], OWL-JessKB[10] 등이 있다. 그러나 대부분 처리 성능에 집중되었을 뿐 보안 문제에 대한 연구는 이루어지지 않았다. 일부 RDF 온톨로지를 위한 접근 제어 방법들[11-13]이 제안되었으나 여러 가지 문제점을 지닌다. 기존 RDF 온톨로지 보안 모델들이 지니는 문제점을 요약하면 다음과 같다.

- 대부분의 온톨로지 저장소가 RDB를 기반으로 개발된다는 현실 미고려
- 오랜 기간에 걸쳐 안정화 된 RDB 보안 모델의 활용 미비
- 사용자 질의가 주어질 때마다 질의를 평가함으로써 발생하는 성능 저하 문제
- 새로운 접근 제어 정책(규칙) 정의 언어 개발을 위한 비용 문제
- 새로운 보안 모델의 안정성 평가를 위한 오버헤드
- 질의 평가 및 처리를 위한 연산자 구현에 따른 비용 증가

이러한 문제점을 개선하기 위해 이 논문에서는 새로운 관계형 모델 기반의 접근 제어 모델을 제안한다. 제안하는 접근 제어 모델은 RDF 온톨로지에 대한 접근을 보다 세부적(클래스 레벨과 인스턴스 레벨)으로 제어할 수 있다. 이를 위해 관계형 데이타베이스의 보안 모델과 뷰를 이용하여 보안 정책에 따라 유효한 데이타 집합을 사전에 정의한다. 특히 RDF 온톨로지 구축시 이용되는 제약조건(rdfs:subClassOf, rdfs:subPropertyOf)을 사전에 평가하여 유효한 허용 데이타 집합을 생성한다. 따라서 사용자 질의가 주어지면 이에 대한 평가 및 실행 수행 속도를 향상시킬 수 있다. 또한 관계형 데이타베이스 관리 시스템에 의해 평가 연산이 수행되므로 보다 안정된 질의 처리가 가능하다.

이 논문의 구성은 다음과 같다. 제2장에서는 RDF 모델에 대한 정의 및 기존 RDF 보안 모델에 대하여 기술하고 제3장에서는 제약 조건과 논문에서 사용하는 용어 및 표기 등에 대하여 정의한다. 제4장에서는 제안하는 모델의 접근 방법과 정의된 모델에 대하여 기술하고 제5장에서는 제안 모델을 시스템 구조, 주요 프로세스 및 알고리즘에 대하여 기술한다. 제6장에서는 제안 모델의 검증을 위한 실험 결과를 보이고 제7장에서는 결론 및 향후 연구 방향에 대하여 기술한다.

2. 관련 연구

이 장에서는 RDF 모델에 대하여 정의하고 웹 정보 기술 언어 계층에서 RDF 하위 계층에 속하는 XML 관련 보안 모델에 대하여 기술한다. RDF 표현 형식이 XML 형식을 이용한다는 점을 고려하여 RDF 보안 모델 개발을 위한 기존 XML 접근 제어 모델의 적용 가능성과 그 한계성에 대하여 논의한다. 마지막으로 RDF 모델을 고려한 기존 접근 제어 방법에 대하여 기술한다.

2.1 RDF 모델

RDF는 웹 상에 있는 자원들에 대한 정보를 표현하기 위한 언어로서 웹 자원에 대한 메타데이타를 표현하기 위한 목적으로 개발되었다. 이는 웹 자원에 대한 응용 프로그램들 간 단순한 정보 교환이 아닌 의미 손실없이 상호간 교환이 이루어지도록 하기 위한 프레임워크를 제공하는데 그 목적이 있다. RDF는 웹 식별자인 URI (Uniform Resource Identifier)를 이용하여 대상 개체(Things)를 식별하기 위한 것으로 프로퍼티(Property)와 프로퍼티의 값(Property Values)으로 자원을 표현한다.

RDF 모델은 자원과 자원의 프로퍼티 그리고 프로퍼티의 값을 노드와 아크로 표현하는 그래프 모델로서, 자원은 웹 상에서 식별할 수 있는, 즉 참조 가능한 형태로 표현되며 그래프 상에서 노드로 표현된다. RDF 모델은 subject, predicate 그리고 object로 구성되는 문장(Statement)들의 집합으로서 subject와 object는 노드를 구성하고 predicate는 아크로 표현한다. RDF에서 subject는 참조 가능한 노드(URIrefs)이어야만 하며 object는 참조 가능한 노드이거나 상수 값, 즉 리터럴(Literal)일 수 있다[1].

따라서 W3C에서 정의한 RDF 모델은 다음과 같이 정의된다.

RDF = (N, B, E, L),

- N: URI를 포함하는 참조 가능한 자원인 노드들의 집합
- B: 공백 노드의 집합
- E: 아크(간선, Edge) 집합, 즉 프로퍼티의 집합으로 방향성을 지님
- L: URI를 포함하지 않는 리터럴의 집합

그림 1은 간단한 RDF 그래프의 예를 보여 준다.

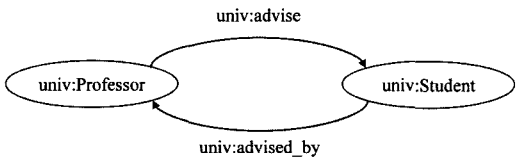


그림 1 RDF 그래프의 예

그림에서, univ는 접두어로서 실제 임의의 URI를 나타내며 위 그래프는 다음과 같은 두 개의 RDF 문장으로 구성되어 있다.

(univ:Professor, univ:advise, univ:Student)

(univ:Student, univ:advised_by, univ:Professor)

RDF 문장은 3-튜플(트리플, Triple) 모델로서 표현되며(subject, predicate, object) 형식이다. subject는 노드나 공백 노드이며 predicate는 프로퍼티이다. object는 노드, 공백 노드 및 리터럴이 될 수 있다. 따라서 subject, predicate 및 object를 다음과 같이 정의할 수 있다.

subject $\in$ (NUB)

predicate $\in$ (E)

object $\in$ (NUBUL)

(subject, predicate, object) $\in$ (NUB) $\times$ (E) $\times$ (NUBUL)

2.2 XML과 RDF 모델의 접근 제어

웹 정보를 기술하기 위해 개발된 HTML의 가장 큰 문제점은 단순히 표현 중심의 정보 생성만이 가능하다는 점이다. 이는 웹 페이지 내의 정보에 대한 의미를 표현할 수 없기 때문에 상호 전달되는 정보의 의미를 정확하게 이해할 수 없다. 이러한 문제점을 보완하고 해결하기 위해 개발된 언어가 XML(Extensible Markup Language)이다[14].

XML은 정보의 의미까지 표현함으로써 보다 정확한 의미 교환을 가능하게 한다. 이러한 의미를 표현한 메타데이타를 요소(Element)라고 정의하며 이에 따라 XML 문서를 작성하게 된다. XML의 장점으로 인해 현재 널리 활용되고 있으며 XML과 관련된 많은 연구들이 진행되어 왔다. 이러한 연구들 중에서 XML 데이타에 대한 안전한 관리를 위해 XML 보안 모델이나 XML 보안 정책을 기술할 수 있는 보안 기술 언어(XML Access Control Description Language) 등이 개발되었다[15-18].

RDF는 기본적으로 XML 형식으로 표현된다. 이러한 특성은 기존 XML을 위해 개발된 보안 모델이나 접근 제어 규칙 정의 언어를 RDF 접근 제어 모델 개발에 적용할 수 있다는 가능성을 제공한다. XML과 RDF는 보다 의미있는 웹 구축이라는 측면에서 그 개발 목적과 지향점이 동일하다고 할 수 있으나 엄격하게 두 언어는 서로 다른 개념에서 출발한다. 즉, RDF가 XML 형식으로 기술되지만 기본적으로 두 언어 사이에는 구조적인 차이점이 존재한다.

먼저 XML은 기본적으로 정해진 XML 구문에 따라 문서를 생성하게 된다. 그러나 RDF는 정해진 구문을 따르지 않는다. 이는 하나의 동일한 RDF 문장이 다양한 방식으로 표현될 수 있음을 의미한다. RDF는 XML 형식, 즉 트리 모델 형식으로 표현되지만 실질적으로는 그래프 모델이다.

무엇보다 RDF는 지식을 표현하는 언어이다. 즉 추론에 의해 새로운 문장(지식)을 생성할 수 있다. 이러한 RDF의 특성은 보안 규칙 정의가 XML에 비해 보다 복잡함을 의미한다. 따라서 XML의 기존 보안 모델이나 XML을 위해 개발된 보안 기술 언어는 RDF 접근 제어 모델 개발에 부적합하며 RDF를 위한 새로운 접근 제어 모델이 요구된다.

2.3 RDF 웹 온톨로지를 위한 접근 제어 모델

Kaushik는 온톨로지에 대한 접근 허용 여부를 결정하기 위해 로직 프로그래밍에 기반한 보안 정책 언어를 제안하였다[12]. Jain은 다중 보안 레벨을 RDF 패턴에 할당하여 RDF 데이타를 제어하는 프레임워크를 제안하였다. 그러나 RDF의 저장에 대한 접근 제어를 위한 프로토타입만을 부분적으로 개발하였을 뿐 추가적인 연산(추가, 삭제 등)에 대해서는 개발되지 않은 상태이다[13].

기존 RDF 접근 제어 모델들은 그래프 모델을 기반으로 하며 사용자 질의가 주어졌을 때 이에 대한 평가를 위해 관련 데이타가 메모리에 로딩되어야 한다. 이는 저장소에 무관하게 모든 데이타 혹은 관련 데이타만을 선별하여 그래프 모델로 변환하여 처리해야 하는 문제점을 지닌다. 이는 구현의 복잡성과 함께 성능 문제를 초래한다. 무엇보다 저장소 모델의 개발과 관련하여 현실적인 상황을 고려하지 않고 있다. 현재까지 대부분의 RDF 웹 온톨로지의 저장 관리를 위한 저장소 모델은 관계형 모델을 기반으로 개발되어 왔다[4,5,7-10].

이러한 배경에서, 관계형 데이타베이스의 안정된 보안 모델을 이용한 RDF 데이타에 대한 접근 제어 방법의 개발이 요구되며 이를 통해 관계형 데이타베이스의 많은 장점을 활용할 수 있다. 즉 추가적인 보안 정책 기술 언어가 요구되지 않으며 새로운 보안 기술 언어 혹은 보안 모델에 대한 안정성 확보가 용이하다. 또한 모든 데이타를 메모리 상에 로딩할 필요가 없으며 관계형 데이타베이스의 최적화 기능을 활용함으로써 전체적인 처리 성능을 개선할 수 있다.

3. 사전 제약 조건, 용어 및 표기 정의

이 장에서는 논문의 제약 조건을 기술하고 논문에서 사용하는 주요 표기 및 기호 등에 대하여 정의한다.

3.1 사전 제약 조건 정의

RDF 온톨로지에 대한 접근 제어는 기본적인 RDF 구문을 고려한 접근 제어 문제와 함께 추론에 의해 생성되는 새로운 지식에 대한 접근 제어 문제까지 고려해야 한다. 그러나 현재 이 논문에서는 RDF에서 풍부한 지식 표현을 위해 제공하고 있는 구문, 즉 개념 레벨에서 제약조건을 표현할 수 있는 형식 표현(예를 들어, rdfs:subClassOf)과 실제 생성된 인스턴스 레벨에 대해

서만 다룬다.

또한 이 논문에서는 추론과 관련된 모듈이나 추론 모듈과의 연계성에 대해서는 다루지 않는다. RDF 온톨로지는 관계형 데이타베이스 내에서 관리되며, RDF 데이타에 대한 접근 제어 평가 또한 관계형 데이타베이스에 의해 이루어진다. 따라서 추론과 관련된 모듈은 데이타베이스 상위 레벨에 존재하게 되며 추론 엔진은 최종적으로 SQL을 통해서 원하는 데이타를 액세스하게 된다. 추론 엔진은 원하는 데이타를 액세스하기 위해 SPARQL [19]과 같은 RDF를 위해 개발된 질의 언어를 이용할 수 있으나 최종적으로 SQL로 변환되어 데이타베이스 관리 시스템에 전달된다.

제안하는 접근 제어 모델의 구현 알고리즘 개발과 모델 구현의 용이성을 고려하여 각 클래스에 해당하는 모든 인스턴스를 개별적인 테이블로 관리하는 저장 구조를 이용한다. 실질적으로 RDF 온톨로지 관리를 위해 매우 다양한 형태의 저장 구조 정의가 가능하며 저장 구조가 달라질 경우 이 논문에서 제안하는 보안 모델의 세부 알고리즘 또한 변경되어야 한다.

3.2 노테이션 및 기호 정의

논문에서 사용하는 주요 노테이션 및 기호는 표 1과 같다.

4. RAC 모델

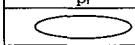
이 장에서는 제안한 RDF 웹 온톨로지 접근 제어 모델에 대하여 기술한다. 제안하는 모델은관계형 데이타베이스 보안 모델을 기반으로 한 RDF 온톨로지 접근 제어 모델로서 RAC(Relational Database-based RDF Ontology Access Control) 모델이라고 명명한다.

4.1 RAC 모델의 접근 방법

RAC 모델은 관계형 데이타베이스를 기반으로 하며, 따라서 RDF 온톨로지를 위한 저장소로서 관계형 데이타베이스를 이용한다. 또한 관계형 데이타베이스의 보안 모델을 이용하여 RDF 데이타에 대한 보안 정책을 수립하게 되며 질의가 주어질 경우, 결과 생성을 위한 질의에 대한 접근 제어 평가 또한 관계형 데이타베이스에 의해 수행된다. 관계형 데이타베이스는 RBAC(Role-Based Access Control) 기반의 보안 모델을 사용하며 물리적으로 GRANT 연산을 이용하여 보안 정책을 수립하게 된다. 특히 관계형 데이타베이스에서는 보다 세부적인 데이타에 대한 보안 정책 수립을 위해 뷰를 활용할 수 있다. 임의의 물리적인 테이블 전체에 대한 접근 제어가 아닌 테이블 내의 데이타에 대한 접근 제어를 위해 뷰를 생성하고 이를 통해 접근을 통제할 수 있다.

RDF는 3-튜플(subject, predicate, object) 구조의 문장들로 구성된다. subject는 표현하고자 하는 대상(Things)을 식별할 수 있도록 URI와 프레그먼트 식별자(Fragment Identifier)를 이용하여 표현한다. 이를 URIS(URI References)라고 하며 예를 들어, http://www.example.org/univ#Professor"에서 http://www.example.org/univ는 URI에 해당하며 "Professor"는 프레그먼트 식별자이다. 일반적으로 프레그먼트 식별자는 클래스(개

표 1 노테이션 및 기호 정의

노테이션/기호	설 명
N	노드들의 집합으로서 URI를 포함하며 subject와 object로 정의됨
E	간선(아크)의 집합으로서 RDF 그래프에서 predicate에 해당
L	URI를 포함하지 않는 상수 값을 지니며 RDF 그래프의 object에 해당함
S	subject들의 집합
s_i	임의의 subject로서 $s_i \in S$
O	object들의 집합
o_i	임의의 object로서 $o_i \in O$
P	predicate들의 집합
p_i	임의의 predicate로서 $p_i \in P$
	RDF 그래프의 노드로서 클래스나 리터럴을 의미
	RDF 그래프의 인스턴스(Instance, Individual)을 나타냄
T	테이블들의 집합으로 $T = \{t_1, t_2, ..., t_n\}$
t_i	임의의 테이블로서 $t_i \in T$
$t_i.f_j$	임의의 테이블 t_i에 있는 특정 필드 f_i
V	뷰들의 집합으로 $V = \{v_1, v_2, ..., v_n\}$
R	롤들의 집합으로 $R = \{r_1, r_2, ..., r_n\}$
U	사용자 집합으로 $U = \{u_1, u_2, ..., u_n\}$
C	클래스의 집합으로 $C = \{c_1, c_2, ..., c_n\}$
c_i	임의의 클래스로서 $c_i \in C$
I	인스턴스의 집합으로 $I = \{i_1, i_2, ..., i_n\}$
i_k	임의의 클래스로서 $i_k \in I$

념, Concept)를 정의하기 위한 부분이다. predicate는 subject를 표현하기 위한 프로퍼티(Property)이며 프로퍼티에 대한 값, 즉 대상이 나타내고자 하는 의미를 표현하는 부분이 object이다[1]. object는 URIS로 표현되지 않고 단순한 상수 값인 리터럴로 표현될 수 있다.

접근 제어 규칙은 개념(클래스) 레벨과 인스턴스 레벨에서 정의될 수 있다. 특정 클래스에 대한 접근 제어는 해당하는 모든 인스턴스들을 제어하도록 한다. 이를 위한 가장 간단한 방법은 각 클래스에 대한 인스턴스를 개별적인 테이블에 관리하고 관계형 데이타베이스의 보안 규칙 정의 방법을 이용하여 이에 대한 접근 권한을 부여하는 방법이다. 이 외에도 다양한 접근 제어 메카니즘이 가능하며 이는 RDF 온톨로지 저장소의 구조에 따라 달라질 수 있다. 이 논문에서는 각 클래스에 해당하는 모든 인스턴스를 개별적인 테이블로 관리하는 저장 구조를 이용한다.

그림 2는 접근 제어를 위한 기본적인 개념을 보여주기 위한 RDF 그래프 예이다. 그림 2는 클래스와 인스턴스 그리고 프로퍼티를 모두 포함하는 예제로서, FullProfessor에 해당하는 인스턴스는 {professor-1, ..., professor-n}이다. 만일 특정 사용자에게 FullProfessor 클래스에 대한 접근을 허용하지 않을 경우, 이에 해당하는 모든 인스턴스에 대한 접근이 거부된다.

그림 2의 RDF 그래프에서, FullProfessor와 GraduateStudent 그리고 각 클래스의 인스턴스들은 그림 3과 같이 저장된다. Class_Table에 있는 각 클래스에 해당하는 인스턴스들을 위해 개별적인 테이블이 생성된다. Professor 클래스의 경우, 해당하는 모든 인스턴스들이 FullProfessor_Instance 테이블에 저장된다. 따라서 FullProfessor 클래스에 해당하는 모든 인스턴스 정보에 대한 보안 정책을 관계형 데이타베이스의 보안 정책 수립 방법을 통해 구체화할 수 있다.

인스턴스에 대한 접근 제어 정책은 매우 다양하게 생성될 수 있다. 특정 사용자에게 일부 데이타 집합에 대

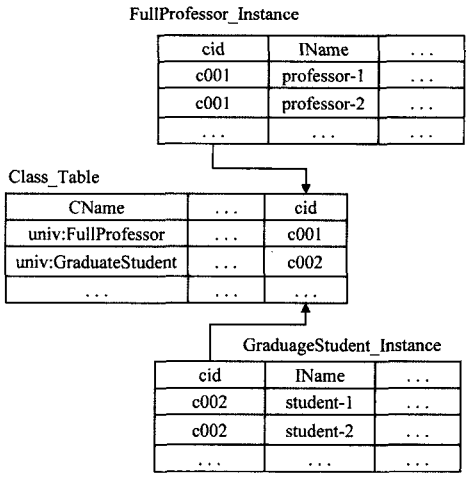


그림 3 RDF 그래프를 위한 실제 물리 저장 구조

한 접근만을 허용할 수 있기 때문에 물리적인 테이블 구조를 기반으로 한 접근 제어 정책 수립에 한계가 있다. 따라서 인스턴스에 대한 부분적인 접근을 제어하기 위해 관계형 데이타베이스의 뷰를 이용한다. 먼저 허용하고자 하는 데이타 집합을 뷰로 생성하고 이 뷰에 대한 접근 권한을 할당한다.

인스턴스에 대한 보안 정책 수립시 제어하고자 하는 접근 영역(부분 데이타 집합)에 따라 접근 패턴을 그림 4와 같이 정의할 수 있다. 관계형 데이타베이스의 기능에 따라 다를 수 있으나 일반적인 검색 연산에 대한 제어는 그림 4(a)를 제외한 나머지 경우에 대해서는 뷰를 생성하여 뷰에 대한 접근 권한을 부여해야 한다.

마지막으로 접근 제어를 위해 고려해야 할 사항은 RDF에서 제공하는 rdfs:subClassOf와 rdfs:subPropertyOf의 처리 문제이다. rdfs:subClassOf와 rdfs:subPropertyOf는 계층을 형성하며 질의 평가시 이에 대한 이행성을 고려해야 한다. 예를 들어, "모든 Professor를 검색하시오."라는 질의에 대해서 Professor 클래스의 인

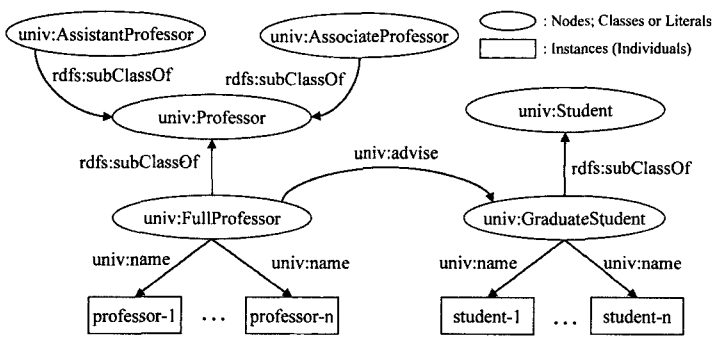


그림 2 인스턴스 및 프로퍼티 포함 예제

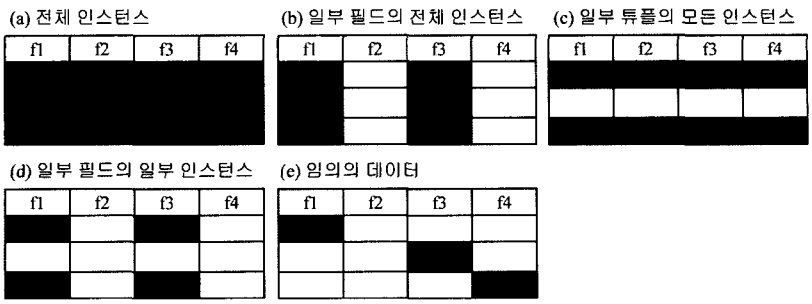


그림 4 인스턴스 접근 영역에 대한 분류

스턴스만을 검색할 수 있고 또는 Professor 클래스의 하위 클래스인 FullProfessor, AssociateProfessor 등의 클래스들에 해당하는 인스턴스를 검색할 수 있다. 즉 이행성을 고려한 결과를 반환할 수도 있고 이행성이 반영되지 않은 결과를 반환할 수 있다. 따라서 보안 정책 수립시 이에 대한 부분을 고려하여 접근 제어 규칙을 정의할 수 있도록 해야 한다. 만일 이행성을 고려한다면 다시 하위 클래스가 접근 대상이 되어 클래스 레벨 혹은 인스턴스 레벨상의 접근 제어 규칙을 정의하게 된다.

4.2 RAC 개념 모델

그림 5는 앞서 언급한 내용을 기반으로 전체적인 RAC의 개념 모델을 보여준다. 주어진 RDF 온톨로지는 변환기(Translator)에 의해 사전에 정의된 테이블로 RDF 데이타를 저장하게 된다. 초기에 저장된 데이타는 테이블에만 존재하게 된다. 저장된 RDF 데이타에 대한 보안 정책을 수립하고 수립된 정책에 따라 롤(Roles)을 먼저 정의하고 GRANT 연산을 이용하여 각 롤에 권한(Authority)을 부여한다. 정의된 롤은 특정 사용자에 부여되어 접근 제어 정책을 생성하게 된다.

이러한 일반적인 절차는 클래스 레벨과 인스턴스 레벨에 모두 동일하게 적용되나 인스턴스에 대한 절차는 조금 더 복잡한 과정을 지닌다. 앞서 기술하였듯이, 인스턴스는 테이블에 직접 접근 제어 규칙을 정의하는 경우와 뷰를 생성한 후 뷰에 대해서 규칙을 생성하는 경우로 분류되기 때문이다. 따라서 전체 인스턴스에 대한

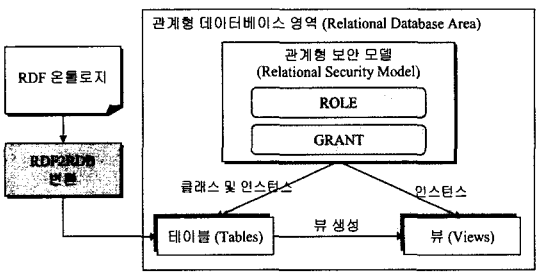


그림 5 RAC의 개념 모델

접근 권한은 테이블을 대상으로 설정되지만 특정 부분에 대한 권한 부여를 위해서는 먼저 뷰를 생성하고 생성된 뷰에 대한 접근 권한을 부여하는 과정으로 수행된다.

4.3 RAC 모델 정의

앞서 서술한 내용을 기반으로 RAC 모델은 다음과 같이 정의된다.

RAC =(T, V, R, U, f)

- T: 테이블의 집합. 입력된 RDF 그래프를 RDB에 저장한 데이타 집합
- V: RDF 데이타 뷰의 집합으로서, 인스턴스의 부분집합
- R: 롤의 집합
- U: 사용자 집합
- f : 롤을 정의하는 연산(f_{ROLE}), 정의된 롤을 특정 사용자들에게 부여하는 연산($f_{AUTHORITY}$), 뷰를 생성하는 연산(f_{VIEW}) 및 질의에 대한 권한을 평가하는 연산(f_{EVAL})의 집합으로, 모든 연산들은 실질적으로 관계형 데이타베이스 관리 시스템에 의해 수행되며 단지 롤을 정의하는 인터페이스와 데이타베이스 시스템 간 연동 연산들만이 추가 모듈에 의해 수행

사용자의 접근 제어 대상은 클래스, 인스턴스 혹은 프로퍼티가 될 수 있다. 그러나 일반적으로 사용자가 접근하고자 하는 최종적인 대상은 인스턴스이다. 이 논문에서는 최종적으로 사용자가 얻고자 하는 결과 대상을 인스턴스로 한정하며 클래스나 프로퍼티에 대한 접근 제어는 다루지 않는다. 따라서 클래스나 프로퍼티는 이행성을 고려하여 전체적인 정보에 대한 접근 제어 규칙 정의하도록 하며 인스턴스에 대해서만 부분 정보에 대한 접근 제어를 정의할 수 있도록 한다.

연산들의 집합인 f에서, f_{ROLE} 연산은 제어하기 위해 선택한 클래스나 프로퍼티와 이행성 반영 여부를 고려하여 인스턴스들을 생성하는데 이용하고 이 결과에서 부분적인 인스턴스 선택을 가능하도록 규칙(관계형 데이타베이스에서의 권한 부여 대상)을 생성하게 된다. 따라

서 f_{ROLE} 연산은 사용자 인터페이스를 통해 관리자가 정의한 사항을 관계형 데이타베이스가 처리할 수 있는 스크립트(CREATE Role과 GRANT)를 생성하는 역할을 수행한다.

5. RAC 시스템 구조 및 주요 프로세스

이 장에서는 제안한 RDF 웹 온톨로지 접근 제어 모델을 위한 시스템 구조와 관계형 데이타베이스 기반 보안 정책 수립을 위한 주요 프로세스 및 알고리즘에 대하여 기술한다.

5.1 RAC 모델을 위한 시스템 구조

그림 6은 RAC 모델을 구현하기 위한 시스템 구조를 보여준다. 그림에서, 임의의 RDF 그래프가 입력으로 주어지게 되면 RDF2RDB(RDF 파서)에 의해 정의된 저장소 모델에 따라 해당 테이블에 RDF 데이타들이 분류되어 저장된다. RDF 데이타가 관계형 데이타베이스 내로 로딩되면 보안 관리자는 보안 규칙을 정의할 수 있는 UI를 통해 접근을 제어하고자 하는 대상을 선택하게 된다. RDB2RDF에 의해 관계형 모델에 저장되어 있던 RDF 데이타 그래프 모델로 역파싱되고 이는 뷰어(Viewer)에 의해 사용자에게 보여지게 된다. 이는 사용자가 RDF 온톨로지의 전체적인 구조를 보다 쉽게 이해하고 보다 정확한 보안 정책 수립할 수 있도록 해 준다. 그러나 현재 이 논문에서는 관계형 데이타베이스에 직접 클래스 정보나 해당 인스턴스 등의 정보를 SQL 문을 통해 액세스하여 보여주는 형태로 구현한다. 즉, RDB를 RDF로 변환하여 이를 그래프로 표현하는 모듈이 구현되어 있지 않은 상태이나 이는 관리자의 관리의 편의성과 RDF 온톨로지 구조에 대한 이해를 위한 것으로 보안 규칙 정의나 이에 대한 접근 제어 평가와는 무관하다.

정의된 보안 규칙은 규칙 변환기(Rule Converter)에 의해 관계형 데이타베이스가 이해할 수 있는 SQL 문으로 변환되게 된다. 접근 제어 방법을 정의한 SQL 문은 실행기(Executor)에 의해 실행되어 롤을 생성하고 사용자에게 해당 테이블에 대한 권한을 부여한다. 또한 인스턴스에 대한 뷰를 생성하고 뷰에 대한 접근 권한을 사용자에게 할당한다.

위와 같은 과정이 완료되고 사용자로부터 질의가 입력되면, 질의 평가 및 접근 제어 정책에 따라 유효한 결과가 사용자에게 반환된다. 질의는 궁극적으로 SQL 문으로 작성되며 SPARQL[19]과 같은 RDF 질의 언어로 표현될 수 있다. SPARQL을 이용하여 RDB 데이타를 접근하기 위해서는 RDF 전용 질의 언어를 SQL 문으로 변환하는 모듈이 추가되어야 한다. 이와 관련한 많은 연구가 이루어져 왔으며 다수의 구현 모듈이 공개되어 있다[5,20,21]. 그러나 이 논문에서는 이에 대하여 상세하게 다루지 않는다.

5.2 접근 제어 규칙 정의 프로세스

RAC 모델은 관계형 데이타베이스 보안 모델을 기반으로 하기 때문에 접근 제어 규칙 정의 과정 또한 관계형 데이타베이스의 보안 규칙 생성 과정과 유사하다. 관계형 데이타베이스에서의 접근 제어는 롤 생성, 권한 부여, 사용자에게 정의한 롤을 할당하는 과정을 거치게 된다. 그러나 앞서 언급하였듯이, RDF 온톨로지는 관계형 데이타와는 다른 특성을 지니고 있기 때문에 보다 세분화된 절차가 요구된다.

그림 7은 접근 제어 규칙 정의를 위한 전체적인 과정을 보여준다. 그림에서 알 수 있듯이, RDF 온톨로지 접근 제어 규칙 정의 프로세스는 관계형 보안 모델에서의 프로세스와 거의 동일하다. 그러나 롤의 생성 과정은 기존 관계형 모델에 비해 보다 세분화된다. 롤 생성 과정은 크게 (1)클래스에 대한 직접적인 권한 부여, (2)클래스에 해당하는 인스턴스들에 대한 권한 부여, (3)클래스에 대한 이행 레벨에 따른 하위 클래스에 대한 권한 부여 (4)이행 레벨에 따른 하위 클래스에 해당하는 인스턴

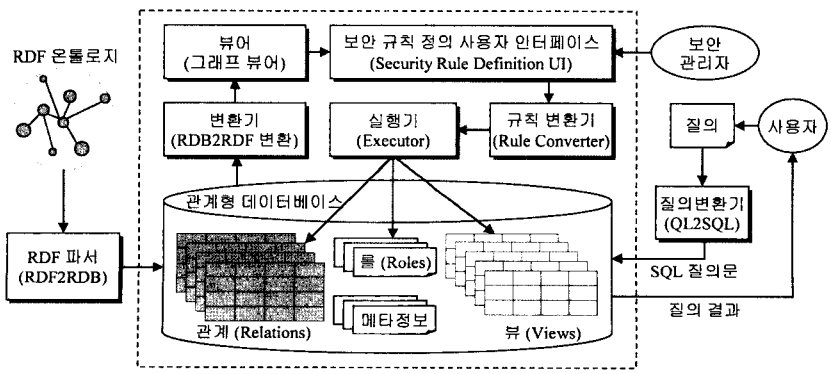


그림 6 RAC 모델 구현을 위한 시스템 구조

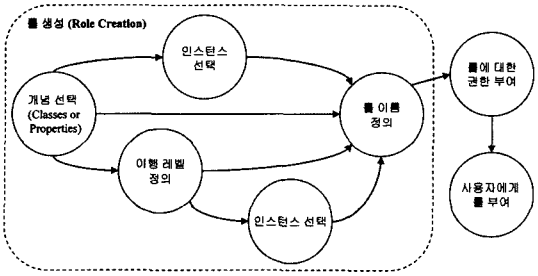


그림 7 접근 제어 규칙 정의 과정(보안 정책 생성 프로세스)

스들에 대한 권한 부여, (5)프로퍼티에 대한 권한 부여, (6)프로퍼티의 이행 레벨에 따른 하위 프로퍼티에 대한 권한 부여 등으로 분류된다. 이행 레벨은 하위 클래스 혹은 하위 프로퍼티에 대한 깊이를 나타낸다. 예를 들어, 깊이가 2인 경우, 기준 클래스의 하위-하위 클래스를 의미한다. 또한 관계형 보안 모델과는 달리 제어 대상이 먼저 선정된 후, 롤이 정의되고 권한이 부여된다. 그러나 내부적인 처리는 관계형 보안 모델을 그대로 이용하므로 동일한 프로세스로 동작하게 된다. 선택 대상이 프로퍼티인 경우에는 해당 프로퍼티와 함께 문장을 생성하는데 이용되는 모든 subject와 object들에 대해서만 접근 제어가 이루어진다.

5.3 관계형 보안 모델로의 변환 알고리즘

그림 7을 통해 기술한 프로세스에 따라 보안 관리자가 정의한 접근 제어 규칙은 관계형 데이타베이스 관리 시스템이 처리할 수 있는 정의 언어로 변환된다. 다음은 정의한 접근 제어 규칙을 관계형 데이타베이스가 이해하고 처리할 수 있는 형식으로 변환하는 알고리즘이다.

```
START Algorithm (IN: selection; OUT: sql)
01:  if selection ∈ C then  //클래스 집합 C의 원소일 경우
02:    if UI.getInstance() is NULL && UI.getTransitionLevel is NULL then  //클래스만 선택
03:        String role_name = UI.getRoleName();
04:        sql = "CREATE Role "+role_name+";";
05:        authorities = UI.getAuthorities();
06:        sql += "GRANT "+authorities+" ON "+selection+ "_Instance TO "+role_name";";
07:    else if UI.getInstance() is NOT NULL && UI.getTransitionLevel is NULL then
08:        sql = UI.getSelectInstances(); //뷰 생성 질의
09:        String role_name = UI.getRoleName();
10:        sql = "CREATE Role "+role_name+";"
11:        authorities = UI.getAuthorities();
12:        sql += "GRANT "+ authorities +" ON "+ view_name + " TO "+ role_name";";
13:    else if UI.getInstance() is NULL && UI.getTransitionLevel is NOT NULL then
14:        int level = UI.getTransitionLevel();
15:        Array subClasses = searchSubClasses(selection);
16:        String role_name = UI.getRoleName();
17:        sql = "CREATE Role "+role_name+";";
18:        authorities = UI.getAuthorities();
19:        String tList = null;
20:        for (int i=0, i<subClasses.length(), i++)
21:            if i == subClasses.length()-1 then
22:                tList += subClasses+ '_Instance";
23:            else
24:                tList += subClasses+ '_Instance,";
25:        sql += "GRANT + authorities + " ON " + tList + " TO " + role_name";";
26:    else if UI.getInstance() is NOT NULL && UI.getTransitionLevel is NOT NULL then
```

```
27:         int level = UI.getTransitionLevel();
28:         Array subClasses = searchSubClasses(selection);
29:         sql = UI.getAllSelectInstances(); //모든 뷰 생성 질의
30:         String role_name = UI.getRoleName();
31:         sql = "CREATE Role "+role_name+";";
32:         authorities = UI.getAuthorities();
33:         sql += "GRANT "+authorities+" ON "+ all_view_names + " TO "+role_name";";
34:         /** 대상이 프로퍼티인 경우 */
35:   else if selection ∈ P then  //프로퍼티 집합 P의 원소일 경우. 즉 predicate
36:     if UI.getTransitionLevel() is NULL then
37:         sql += "CREATE VIEW view_name
38:                 AS   SELECT * FROM Property_Table
39:                     WHERE Property_Table.name = 'selection';";
40:         String role_name = UI.getRoleName();
41:         sql = "CREATE Role "+role_name+";";
42:         authorities = UI.getAuthorities();
43:         sql += "GRANT +authorities+" ON "+ view_name + " TO "+ role_name;
44:     else
45:         int level = UI.getTransitionLevel();
46:         Array subProperties = searchSubProperties(selection);
47:         sql = UI.getAllSelectInstances(); //모든 뷰 생성 질의
48:         String role_name = UI.getRoleName();
49:         sql = "CREATE Role "+role_name+";";
50:         authorities = UI.getAuthorities();
51:         sql += "GRANT "+authorities+" ON "+ all_view_names + " TO " + role_name";"
52:   /** 사용자에게 롤 할당 */
53:   sql += "GRANT" + role_name + "TO" + UI.getUserList();
54:   return sql;
END Algorithm
```

알고리즘에서, 라인 1~33은 선택 대상이 클래스인 경우에 대한 처리로서 순차적으로 클래스, 클래스에 해당하는 인스턴스, 하위 클래스, 하위 클래스의 인스턴스에 대한 접근 권한을 제어하기 위한 알고리즘을 정의한 부분이다. 라인 34~51은 대상이 프로퍼티인 경우를 처리하기 위한 알고리즘이다. 51번째 라인까지의 연산을 통해 롤 생성 및 권한 부여를 위한 최종적인 SQL 문이 생성된다. 변환이 완료되면 사용자에게 롤을 부여하게 된다. 정의한 알고리즘은 규칙 변환기에 의해 수행되며 생성된 SQL 문장은 관계형 데이타베이스와의 연동을 통해 실행기에 의해 처리된다.

6. 실험 및 평가

이 장에서는 제안 모델에 대한 실험 및 평가 결과에 대하여 기술한다. 실험은 제안 모델의 접근 제어에 대한 정확성 문제에 초점을 둔다. 따라서 기존 관련 접근 방법과의 처리 성능에 대한 실험은 향후 연구 과제로 남겨 두며, 단지 평가 결과에서 이에 대해 정성적인 결과만을 기술한다.

6.1 실험을 위한 RDF 그래프 예제

실험은 RAC 모델에서 RDF 온톨로지 접근 제어를 정확하게 수행하는지를 확인하기 위한 것으로, 실험을 위한 접근 제어 유형은(1)클래스에 대한 직접적인 접근 제어, (2)대상 클래스의 인스턴스들에 대한 접근 제어, (3)클래스의 이행 레벨에 따른 하위 클래스에 대한 접근 제어, (4)이행 레벨에 따른 하위 클래스의 인스턴스들에

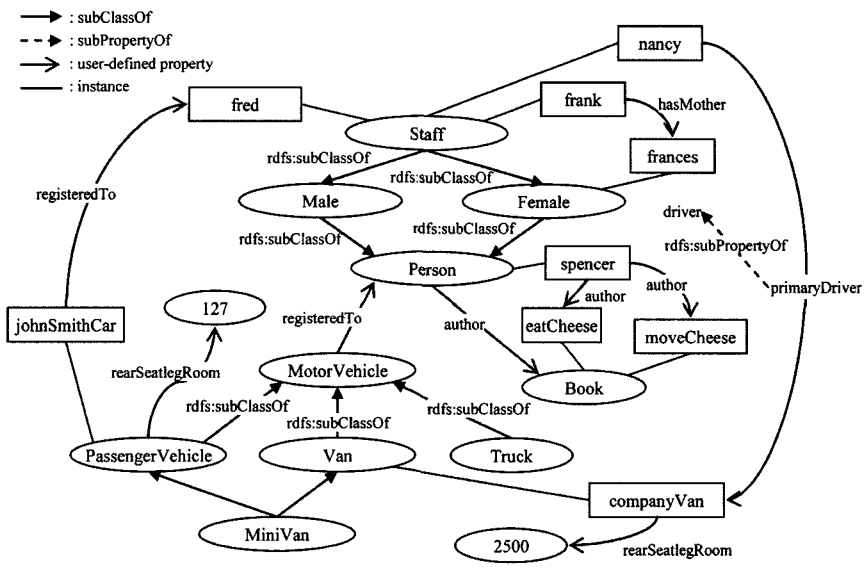


그림 8 실험을 위한 RDF 그래프

대한 접근 제어, (5)선택 프로퍼티에 대한 접근 제어 및 (6)프로퍼티의 이행 레벨에 따른 하위 프로퍼티에 대한 접근 제어 등이다.

실험은 W3C의 표준 문서[1]에 있는 예제를 이용하여 RDF 그래프를 생성하며, 이용된 RDF 그래프는 그림 8과 같다. 주어진 RDF 그래프는 클래스 계층, 프로퍼티 계층 및 인스턴스 등을 포함하는 예제로서, 이 논에서 제안하는 RDF 접근 제어 모델에 대한 평가에 필요한 모든 구성 요소들을 포함한다. RDF 그래프에 대한 XML 표현이나 이에 대한 상세한 내용을 표준 문서를 참조하기 바란다[1].

6.2 규칙 및 질의 정의

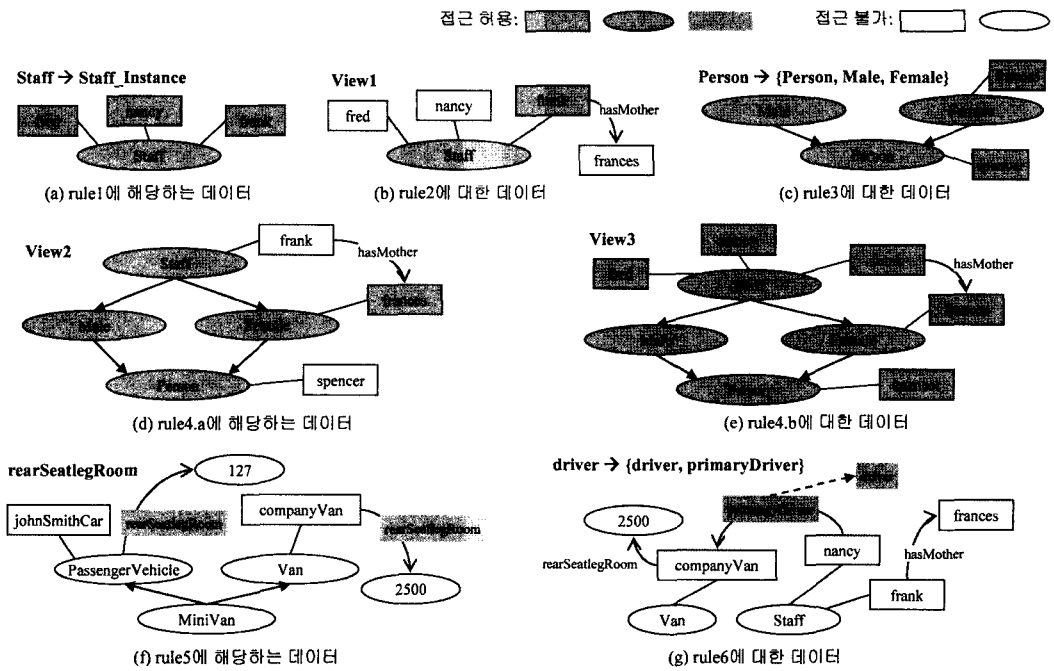
접근 제어 유형별 실험을 위한 보안 규칙은 다음과 같다.

rule1 = (Staff, null, {select}, +, {u1})
rule2 = (View1, null, {select}, +, {u2})
rule3 = (Person, 1, {select}, +, {u3})
rule4.a = (View2, 1, {select}, +, {u3, u4})
rule4.b = (View3, 2, {select}, +, {u4})
rule5 = ({rearSeatlegRoom}, null, {select}, +, {u5})
rule6 = ({driver}, 1, {select}, +, {u6})

기술한 규칙 구문은 <target, transitionLevel, action, sign, user>와 같은 형식으로 표현된다. target은 클래스, 프로퍼티 혹은 특정 뷰가 되며 target이 클래스인 경우에는 해당하는 인스턴스를 관리하는 테이블이 된다. 따라서 target = $\{t_i, v_i, p_i\}$로 정의된다. transitionLevel은 접근이 가능한 하위 객체에 대한 깊이를 의미하며 action = {select, update, insert, delete}이다. sign={+, -}로서 각각 접근 허용과 접근 불가를 의미한다. 마지막으로 user는 해당하는 사용자 리스트를 의미하며 따라서 user = $\{u_i \mid u_i \in U\}$로 정의된다.

그림 9는 위에서 정의한 규칙에 따라 사용자가 접근 가능한 데이타 집합을 보여준다.

rule1은 선택 클래스 레벨에서 정의된 규칙으로 사용자 u1은 Staff 클래스에 해당하는 모든 인스턴스에 대한 접근이 허용되며, 따라서 물리적인 Staff_Instance 테이블 전체에 대한 접근을 허용한다는 의미이다(그림 9(a)). rule2는 선택 클래스에 해당하는 일부 인스턴스만을 허용할 수 있도록 u2에게 허용한다는 의미로서 이를 위해 생성된 뷰가 View2이므로 이를 기반으로 접근을 제어하게 된다(그림 9(b)). rule3은 Person 클래스의 하위 클래스들 중에서 깊이가 1인, 즉 바로 하위의 클래스에 해당하는 모든 인스턴스에 대한 접근을 허용한다는 보안 규칙을 정의한다(그림 9(c)). rule4.a는 선택 클래스와 그 바로 하위 클래스에 해당하는 인스턴스의 부분 집합에 대한 접근을 제어하기 위해 생성된 View2에 대해서만 접근을 제어하기 위한 규칙이다(그림 9(d)). rule4.b는 rule4.a와 동일하나 깊이가 2인 하위 클래스의 인스턴스 일부까지의 접근을 허용하도록 정의한 규칙이다(그림 9(e)). rule5와 rule6은 프로퍼티에 대한 접근 제어 규칙으로서, rule5는 프로퍼티가 rearSeatlegRoom인 경우에 대해서만 접근을 허용한다는 의미이다. 따라서 rearSeatlegRoom 프로퍼티와 연관된 subject와 object에 접근을 허용하게 된다(그림 9(f)). rule6은 rule5와



(a) rule1에 해당하는 데이터 (b) rule2에 대한 데이터 (c) rule3에 대한 데이터

(d) rule4.a에 해당하는 데이터 (e) rule4.b에 대한 데이터

(f) rule5에 해당하는 데이터 (g) rule6에 대한 데이터

그림 9 각 규칙에 따른 접근 제어 가능 데이타 집합

동일하나 하위 프로퍼티에 대한 접근까지 허용하도록 정의한 규칙이다(그림 9(g)).

표 2는 보안 규칙에 따라 접근 제어가 정확하게 평가되어 유효한 데이타에 대한 접근만을 허용하는지를 평가하기 위해 정의한 질의 예제를 보여준다. 정의된 질의 예제는 RDF를 위해 개발된 질의 언어인 SPARQL 형식으로 표현한다. 그러나 실질적인 실험에서는 SQL 문으로 변환되어 관계형 데이타베이스에 전달된다.

6.3 실험 결과 및 평가

6.2절에서 정의한 접근 제어 규칙과 질의 예제를 이용하여 수행한 실험 결과는 표 3과 같다. 표 3의 결과에서, 사용자 u1은 rule1을 통해 Staff 클래스의 모든 인스턴스를 검색할 수 있도록 롤을 부여했기 때문에 q1.1에 대한 유효한 결과를 얻을 수 있으나 q1.2에 대해서 접근이 허용되지 않아야 한다. 실험 결과에서도 q1.1에 대한 결과로서 {fred, nancy, frank}를 반환하였다. 그러나 사용자 u1이 질의 q1.2를 통해 접근하려고 할 때 이에 대한 접근이 허용되지 않았다.

사용자 u2는 rule2를 통해 그림 9(b)와 같이 Staff 클래스의 일부 인스턴스에 대해서만 접근할 수 있도록 롤이 부여되었기 때문에 q2.1에 대해서는 올바른 결과인 {frank}를 얻는다. 그러나 u2의 q2.2에 대한 접근이 허용되지 않았음을 알 수 있다.

사용자 u3은 rule3을 이용하여 Person 클래스를 기준으로 바로 하위 클래스인 자식 클래스의 인스턴스에 대한 접근이 허용되었으므로 q3.1에 대한 검색 결과, {frances}를 얻게 된다. 그러나 Staff 클래스로부터 깊이 2인 하위 클래스(Staff 클래스)에 대한 접근은 허용되지 않았으므로 q3.2을 이용하여 Staff 클래스의 인스턴스를 액세스하고자 했을 때 접근이 제한되어야 하며 실험 결과에서도 동일하게 접근이 허용되지 않았음을 알 수 있다.

사용자 u3과 u4는 rule4.a를 이용하여 그림 9(d)와 같은 데이타를 포함하는 뷰에 대해서만 접근이 가능하도록 롤이 부여되었다. 즉, Person 클래스와 자식 클래스(Male, Female)의 인스턴스들에 대한 접근은 가능하다. 따라서 u3과 u4 모두 q4.1에 대한 유효한 결과를 얻게 된다. q4.2와 같이 Person 클래스의 하위-하위 클래스(깊이가 2인)인 Staff 클래스의 인스턴스에 대한 접근이 시도될 경우, u3에게는 접근이 허용되지 않는 반면, rule4.b를 통해 깊이가 2인 하위 클래스에 대한 접근이 허용된 u4에 대해서는 접근이 허용된다. 또한 사용자 u4는 q4.3에 대해서도 접근이 허용되어 유효한 결과를 얻을 수 있다.

사용자 u5는 rule5를 통해 모든 rearSeatlegRoom 프로퍼티에 대한 접근이 허용된다. 따라서 질의 q5.1에 대한 유효한 결과를 얻을 수 있다. 프로퍼티는 subject와 object를 연결해 주는 predicate이므로 검색 결과는

표 2 실험을 위한 질의 예제

질의 예제	설 명
q1.1 = SELECT ?staff WHERE { ?staff rdf:type :Staff .}	Staff 클래스의 모든 인스턴스 검색
q1.2= SELECT ?van WHERE { ?van rdf:type :Van .}	Van 클래스의 모든 인스턴스 검색
q2.1 = SELECT ?staff WHERE { ?staff rdf:type :Staff . ?staff :hasMother frances .}	Staff 클래스의 인스턴스 중에서 frances가 모친인 직원 검색
q2.2 = SELECT ?staff WHERE { ?staff rdf:type :Staff . johnSmithCar :registeredTo ?staff .}	johnSmithCar를 소유하고 있는 직원 검색
q3.1 = SELECT ?female WHERE { ?female rdf:type :Female .}	모든 여성 검색
q3.2 = SELECT ?staff WHERE { ?staff rdf:type :Staff .}	모든 직원 검색
q4.1 = SELECT ?female WHERE { ?female rdf:type :Female . OPTIONAL {?male :hasMother ?female .} }	여성 중에서 자녀가 있는 사람 검색
q4.2 = SELECT ?staff WHERE { ?staff rdf:type :Staff . OPTIONAL {?car :registeredTo ?staff .} }	직원 중에서 차를 소유하고 있는 직원 검색
q4.3 = SELECT ?staff WHERE { ?staff rdf:type :Staff . johnSmithCar :registeredTo ?staff .}	johnSmithCar를 소유하고 있는 직원 검색
q5.1 = SELECT ?car WHERE { ?car :rearSeatlegRoom ?room .}	rearSeatlegRoom 프로퍼티를 지닌 subject 검색
q5.2 = SELECT ?person ?car WHERE { ?person :primaryDriver ?car .}	primaryDriver 프로퍼티를 지니는 subject와 object 검색
q6 = SELECT ?person ?car WHERE { ?person :driver ?car .}	driver 프로퍼티를 지니는 subject와 object 검색

표 3 접근 제어 실험 결과

질의	사용자	실험결과	질의	사용자	실험결과
q1.1	u1	접근허용	q1.2	u1	접근불가
q2.1	u2	접근허용	q2.2	u2	접근불가
q3.1	u3	접근허용	q3.2	u3	접근불가
q4.1	u3	접근허용	q4.1	u4	접근허용
q4.2	u3	접근불가	q4.2	u4	접근허용
q4.3	u3	접근불가	q4.3	u4	접근허용
q5.1	u5	접근허용	q5.2	u5	접근불가
q5.2	u6	접근허용	q6	u6	접근허용

subject 혹은 object가 된다. 그러나 q5.2에 대해서는 primaryDirver에 대한 접근이 허용되도록 보안 규칙이 적용되지 않기 때문에 접근이 제한되어 유효한 결과를 얻을 수 없다.

사용자 u6은 rule6을 통해 driver 프로퍼티와 깊이가 1인 하위 프로퍼티까지 접근할 수 있는 롤이 부여되었다. 따라서 q6의 driver 프로퍼티에 대한 직접적인 접근은 물론 q5.2에서와 같이 primaryDriver 프로퍼티(driver 프로퍼티의 하위 프로퍼티)에 대해서도 접근할 수 있다.

추가적으로, 그래프 모델을 기반으로 한 기존 접근 방법의 경우, 접근 제어 평가를 위해 관련된 모든 데이타 메모리에 로딩해야 한다. 따라서 데이타를 로딩을 위한 많은 비용을 요구한다. 또한 기존 방법은 그래프 상에서의 순회를 통해 접근 가능한 데이타를 판별한다. 따라서 이 논문의 접근 방법, 즉 접근 제어 규칙에 따라 접근 가능한 데이타를 정의해 놓은 뷰를 이용하는 방법과 비교하여 보다 많은 처리 비용을 요구한다. 현재까지 실험을 위해 제공되는 오픈 시스템이 제공되지 않기 때문에 이에 대한 실험 평가는 향후 연구 과제로 남겨 둔다.

7. 결 론

시맨틱 웹과 함께 이를 위한 온톨로지 기술 언어로서 개발된 RDF에 대한 많은 연구가 이루어져왔다. 그러나 정보 관리에서 가장 중요한 요소 중 하나인 보안에 대한 연구는 매우 미비한 상황이다. 일부 보안 모델이 제안되었으나 대부분의 데이타가 관계형 데이타베이스에 저장, 관리되고 있다는 현실적인 상황과 함께 오랜 기간의 연구를 통해 안정된 관계형 데이타베이스의 보안 모델을 활용할 수 없다는 문제점을 지닌다. 이는 보안을 위한 새로운 접근 제어 모델 개발과 이를 위한 새로운 모듈이 탑재되어야 하는 문제점을 야기한다. 또한 추가적인 보안 규칙(접근 제어 규칙) 정의를 위한 언어 개발

이 요구된다.

이러한 기존 문제점과 현실적인 상황을 고려하여 관계형 데이타베이스 기반의 RDF 웹 온톨로지 보안 모델인 RAC 모델을 제안하였다. RAC의 개념 모델을 통해 접근 방법에 대하여 기술하였고 시스템 구조와 주요 프로세스 및 관계형 보안 모델 생성 알고리즘에 대하여 기술하였다. 또한 가능한 보안 규칙 유형을 정의하고 실험을 위해 각 유형에 따른 접근 제어 규칙과 질의 예제를 정의하였다. 실험은 RAC 모델의 질의에 대한 평가의 정확성을 판단하기 위한 것으로, 실험을 위한 데이타는 W3C의 RDF 표준 문서에 기술된 예제를 이용하였다. 실험 결과에서, RAC 모델은 각 유형별로 정의된 접근 제어 규칙에 따라 주어진 질의에 대해 정확하게 평가하여 유효한 접근에 대해서만 올바른 결과를 반환해 주었다.

이 논문에서 제안한 RAC 모델은 다음과 같은 장점을 지닌다. 우선 현재 대부분의 RDF 온톨로지 저장소가 관계형 데이타베이스를 기반으로 개발되었으므로 제안한 RAC 모델 적용이 용이하며 추가적인 보안 모델 개발이 요구되지 않는다. 또한 안정된 관계형 보안 모델을 이용함으로써 보안 모델의 안전성에 대한 검증이 용이하다. 아울러 관계형 데이타베이스 보안 모델에서 제공하는 연산자와 정의 언어를 이용함으로써 추가적인 접근 제어 규칙 정의 언어나 이를 위한 연산자 개발이 요구되지 않는다.

이 논문에서는 클래스와 인스턴스를 위한 물리 테이블 간 1:1인 저장 구조를 이용하고 있다. 이는 모델과 이를 위한 구현의 용이성을 고려한 것으로 최적의 처리 성능을 제공하지 않는다. 따라서 향후에는 이에 대한 벤치마킹을 통해 보다 개선된 저장 구조 개발이 요구된다. 또한 이 논문에서는 클래스와 프로퍼티를 기준으로 인스턴스를 대상으로 한 접근 제어만을 고려하였다. 이러한 개념은 클래스와 프로퍼티가 온톨로지를 구축하기 위해 일차적으로 정의되어야 하는 대상이라는 측면에서 접근 제어 규칙 정의시 순차적인 제어가 가능하다는 장점을 지닌다. 그러나 임의의 인스턴스를 기준으로 제어 규칙을 정의하거나 접근 대상을 클래스 혹은 프로퍼티로 정의할 수 없다는 문제점을 지니며 보다 다양하고 세분화 된 제어 정책 수립을 어렵게 한다. 또한 추론을 통해 새롭게 생성된 정보에 대한 접근 제어 문제에 대해서도 고려하고 있지 않다. 따라서 향후에는 이러한 문제를 해결할 수 있도록 모델 확장이 요구된다. 마지막으로, 보다 다양한 예제를 이용한 추가적인 정확성 평가와 함께 전체적인 처리 성능에 대한 연구가 이루어져야 한다.

참 고 문 헌

[1] Frank Manola and Eric Miller, RDF(Resource Description Framework) Primer, W3C Recommendation, http://www.w3.org/TR/rdf-primer/, February 10, 2004.

[2] Dan Brickley and R.V. Guha, RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004.

[3] Protégé, http://protege.stanford.edu/.

[4] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," Springer Verlag, Lecture Notes in Computer Science, Vol. 2342, pp. 54-68, 2002.

[5] Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/.

[6] Kowari, http://sourceforge.net/projects/kowari/.

[7] David Beckett, "The Design and Implementation of the Redland RDF Application Framework," Elsevier, Computer Networks, Vol.39, No.5, pp. 577-588, August 2002.

[8] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin, "LUBM: A Benchmark for OWL Knowledge Base Systems," Journal of Web Semantics, Vol.3, No.2, 2005.

[9] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin, "An Evaluation of Knowledge Base Systems for Large OWL Datasets," Vol. LNCS 3298, pp. 274-288, 2004.

[10] OWLJessKB: A semantic Web Reasoning Tool, http://edge.cs.drexel.edu/assemblies/software/owljesskb/.

[11] Pavan Reddivari, Tim Finin, and Anupam Joshi, "Policy-Based Access Control for an RDF Store," 20th International Joint Conference on Artificial Intelligence(IJCAI-07), Hyderabad, India, January 6-12, 2007.

[12] Saket Kaushik, Duminda Wijesekera, and Paul Amman, "Policy-Based Dissemination of Partial Web-Ontologies," In 2005 ACM Workshop on Secure Web Services(SWS 2005), Fairfax, Virginia, USA, pp. 43-62, November 11, 2005.

[13] Amit Jain and Csilla Farkas, "Secure Resource Description Framework: an Access Control Model," ACM Symposium on Access Control Models and Technologies, Lake-Tahoe, California, USA, pp. 121-129, June 7-9, 2006.

[14] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan, W3C, XML 1.1(Second Edition), W3C Recommendation, August 2006.

[15] Dongwon Lee, Wang-Chien Lee, Peng Liu, "Supporting XML Security Models using Relational Databases: A Vision," XML Database Symposium(XSym), Berlin, Germany, September 2003.

[16] Elisa Bertino, Silvana Castano, Elena Ferrai, and Marco Mesiti, "Specifying and enforcing access control policies for xml documents and sources," Springer, In World Wide Web, Vol.3, No.3, pp. 139-151, November 2000.
[17] Damiani, De Capitani et al., "A Finegrained Access Control System for XML Documents," ACM Transaction on Information and System Security (TISSEC), Vol.5, No.2, pp. 169-202, 2002.
[18] Tim Moses, eXtensible Access Control Markup Language(XAML) Version 2.0, OASIS Standard, February 2005.
[19] Eric Prud'hommeaux and Andy Seaborne, SPARQL Query Language for RDF, W3C Candidate Recommendation, June 2007.
[20] Stephen Harris and Nigel Shadbolt, "SPARQL Query Processing with Conventional Relational Database Systems," Springer-Verlag, Lecture Notes in Computer Science, Vol. 3807, pp. 235-244, 2005.
[21] Artem Chebotko, Shiyong Lu, Hasan M. Jamil, and Farshad Fotouhi, "Semantics Preserving SPARQL-to-SQL Query Translation for Optional Graph Patterns," Technical Report TR-DB-052006-CLJF, May 2006, Revised November 2006.

정 동 원
정보과학회논문지 : 데이타베이스
제 35 권 제 1 호 참조