한국시뮬레이션학회 논문지
Vol. 17, No. 3, pp. 53-62 (2008. 9)

오라클 VPD 기반의 RDF 웹 온톨로지 접근 제어 모델의 설계 및 구현

정혜진[1] · 정동원[1†]

Design and Implementation of the RDF Web Ontology Access Control Model based on Oracle VPD

Hyejin Jeong · Dongwon Jeong

ABSTRACT

This paper proposes a new implementational model based on the security model of Oracle for Web ontology. Recently, several access control models using relational database security model for access control to Web ontology have been developing, and one of the most representative access control model is the RAC model. However, the RAC model is based on the standard security model, and thus it does not provide a implementational model for practical relational database management systems. In this paper, we propose an implementational model based on Oracle which is widely used and providing various security policies. This paper shows the implementation and experimental evaluation. Especially, the proposed model uses the VPD security model of Oracle and support high application and usability.

Key words : Security, Access control, Web ontology, RDF, Relational database, Oracle, VPD

요 약

이 논문에서는 웹 온톨로지를 위한 오라클 보안 모델 기반의 구현 모델을 제안한다. 최근 웹 온톨로지에 대한 접근 제어를 위해 관계형 데이터베이스 보안 모델을 이용한 접근 제어 모델이 개발되고 있으며 대표적인 접근제어 모델이 RAC 보안 모델이다. 그러나 RAC 보안 모델은 관계형 데이터베이스의 표준 보안 모델에 기반을 두고 있으며 실제 관계형 데이터베이스 관리 시스템의 구현 모델은 개발되어 있지 않다. 이 논문에서는 이러한 RAC 보안 모델의 문제점을 해결하기 위하여 현재 가장 널리 사용되며 다양한 보안 정책을 제공하는 RDBMS인 오라클 기반의 구현 모델을 제안한다. 또한, 제안하는 RAC 구현 모델의 구현 및 평가에 대하여 기술한다. 특히, 제안하는 구현 모델은 오라클에서 제공하는 VPD 보안 모델을 이용하며 높은 실용성과 활용성을 제공한다.

주요어 : 보안, 접근 제어, 웹 온톨로지, RDF, 관계형 데이터베이스, 오라클, VPD

1. 서 론

인류의 생활과 문화를 변화시킨 웹이 최근 들어 '시맨틱 웹'이라는 이름으로 새로운 변화를 준비하고 있다. 초기 웹은 단순한 하이퍼링크 기능만으로도 혁신적이었지만 점차 웹이 보급되고 유통되어 정보의 양이 늘어나 초기 웹의 링크 기술만으로는 감당하기 힘들어졌다. 이에 따라 단순 링크만이 아닌 고도화, 지능화, 자동화된 웹을 지향하게 되었으며, 이러한 목표를 위한 차세대 웹 기술로서 시맨틱 웹(Semantic Web)이 등장하게 되었다[1].

시맨틱 웹을 구현하기 위해서는 다양한 기술이 요구되며, 특히 웹 리소스(웹 문서, 각종 파일, 서비스 등)에 대한 정보와 자원 사이의 관계-의미 정보(Semantics)를 기계(컴퓨터)가 처리할 수 있는 온톨로지의 구축이 필수적으로 요구된다. 현재 시맨틱 웹 온톨로지를 기술하는 표준 언어로 W3C에서 제안한 RDF(Resource Description Framework), RDF-S(RDF Schema), OWL(Web Ontology Language) 등이 있다[2-4]. RDF는 XML의 단순 의미 마크

2008년 6월 30일 접수, 2008년 9월 9일 채택
[1] 국립군산대학교 정보통계학과
주 저 자 : 정혜진
교신저자 : 정동원
E-mail; djeong@kunsan.ac.kr

업 방법을 탈피하여, 정보자원 간의 메타데이터 수준의 의미 구조를 정의하는 언어이며 RDF-S는 RDF 형태로 기술되어진 정보 자원의 데이터 모델을 정의하는 언어이다[5-8]. OWL은 RDF와 RDF-S에 기반을 두고 DAMAL+OIL의 의미 체계를 활용하여 설계된 온톨로지 언어로 정보 자원간의 의미 관계 뿐만 아니라, 도메인의 지식 체계를 표현하기 위해서 설계된 표준 온톨로지 언어다[7-9]. 이러한 웹 온톨로지를 기술하는 표준 언어들 중에서 가장 기초적으로 널리 이용되는 언어가 RDF이다.

최근 동향을 보면, 대부분의 온톨로지 저장소는 관계형 데이터베이스를 기반으로 개발되고 있다[10-14]. 이러한 상황에서, 웹 온톨로지에 대한 접근 제어를 위해 관계형 데이터베이스 보안 모델을 이용한 접근 제어 모델이 개발되었으며 대표적으로 RAC 보안 모델(Relational Database-based RDF Ontology Access Control Model)을 예로 들 수 있다[15]. 그러나 RAC 보안 모델은 관계형 데이터베이스의 표준 보안 모델에 기반을 두고 있으며 실제 관계형 데이터베이스 관리 시스템의 구현 모델은 개발되어 있지 않다. 이는 제안된 보안 모델의 활용성 및 실용성을 약화시킨다.

이 논문에서는 이러한 문제점을 해결하기 위하여 오라클 기반의 RAC의 구현 모델을 제안한다. 다양한 관계형 데이터베이스 관리 시스템이 개발되어 있으나 오라클을 위한 구현 모델을 제안하는 이유는 다음과 같다.

- 오라클은 현재 가장 널리 사용
- 다양한 보안 정책 제공
- 다른 관계형 데이터베이스 관리 시스템에 비해 뛰어난 안정성과 성능 제공

오라클은 표준 관계형 보안 모델은 물론 다양한 보안 정책을 수립할 수 있는 방법을 제공한다. 이 논문에서 오라클에서 제공하는 VPD(Virtual Private Database) 기반 보안 정책을 이용한 RAC의 구현 모델을 제안한다[16,17].

이 논문의 구성은 다음과 같다. 제2장에서는 본 연구에 필요한 RAC 보안 모델과 오라클 보안 모델인 VPD에 대하여 기술한다. 제3장에서는 오라클의 VPD 기반의 RAC 구현 모델 즉, 제안모델에 대하여 기술한다. 제4장에서는 실험을 통한 구현 모델의 평가 결과를 기술하고 제5장에서는 결론 및 향후 연구 내용에 대하여 기술한다.

2. 관련 연구

이 장에서는 RAC 보안 모델에 대하여 간략하게 소개하고 RAC 보안 모델을 구체화하기 위한 오라클 VPD 기반 보안 모델에 대하여 기술한다.

2.1 RAC 보안 모델

RAC 보안 모델은 관계형 데이터베이스를 기반으로 하며, RDF 온톨로지를 위한 저장소로서 관계형 데이터베이스를 이용한다. 또한 관계형 데이터베이스의 보안 모델을 이용하여 RDF 데이터에 대한 보안 정책을 수립하게 되며 질의가 주어질 경우, 결과 생성을 위한 질의에 대한 접근 제어 평가 또한 관계형 데이터베이스에 의해 수행된다. 관계형 데이터베이스는 RBAC(Role-Based Access Control) 기반의 보안 모델을 사용하며 물리적으로 GRANT 연산을 이용하여 보안 정책을 수립하게 된다. 특히 관계형 데이터베이스에서는 보다 세부적인 데이터에 대한 보안 정책 수립을 위해 뷰를 활용할 수 있다. 임의의 물리적인 테이블 전체에 대한 접근 제어가 아닌 테이블 내의 데이터에 대한 접근 제어를 위해 뷰를 생성하고 이를 통해 접근을 통제할 수 있다[11].

그림 1은 RAC 개념 모델을 보여준다. 주어진 RDF 온톨로지는 파서를 거쳐 관계형 데이터베이스 내에 저장되며, 초기에 RDF 온톨로지는 테이블에만 존재하게 된다. 저장된 RDF 데이터에 대해 보안 정책을 수립하고 수립된 정책에 따라 롤(Roles)을 먼저 정의한다. 이를 위해 GRANT 연산을 이용하여 각 롤에 권한(Authority)을 부여한다. 정의된 롤은 특정 사용자에 부여되어 접근 제어 정책을 생성한다.

RDF 온톨로지는 크게 클래스와 그에 해당하는 실제 인스턴스로 구성된다. 앞서 기술한 일반적인 절차는 클래스 레벨과 인스턴스 레벨에 모두 동일하게 적용되나 인스턴스에 대한 절차는 조금 더 복잡한 과정을 지닌다. 이는 클래스와는 달리 인스턴스는 테이블에 직접 접근 제어 규칙을 정의하는 경우와 뷰를 생성한 후 뷰에 대해서 규칙

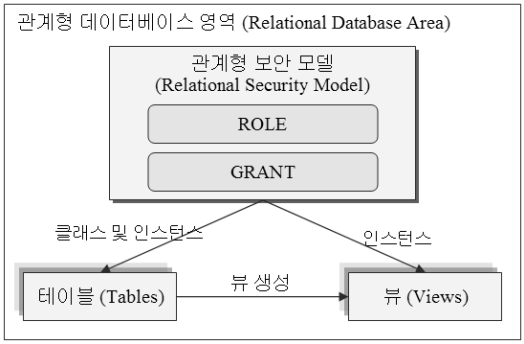

그림 1. RAC 개념 모델

을 생성하는 경우로 분류되기 때문이다. 따라서 전체 인스턴스에 대한 접근권한은 테이블을 대상으로 설정되지만 특정 부분에 대한 권한부여를 위해서는 먼저 뷰를 생성하고 생성된 뷰에 대한 접근권한을 부여하는 과정으로 수행된다. RAC모델의 한계점은 클래스인 경우에는 테이블에 해당하는 모든 인스턴스를 저장한다는 가정이다. 이 논문에서는 이와 같은 RAC모델의 가정을 전제한 제안 모델을 구현한다.

2.2 오라클 VPD

VPD(Virtual Private Database)는 데이터베이스 차원의 FGCA(Fine Grained Access Control)와 "secure application context"를 통해 보안 정책을 구현함으로써 이루어진다.

VPD(Virtual Private Database)는 행과 열 레벨에서 데이터베이스 접근 제어를 위한 보안 정책 생성을 가능하게 한다. 기본적으로 VPD는 임의의 VPD 보안 정책을 적용한 테이블, 뷰와 같은 객체를 액세스하는 SQL문에 동적인 WHERE 절을 추가한다. 보안 정책을 데이터베이스 객체에 직접 부여하고 사용자들이 데이터를 액세스할 때마다 보안 정책들을 자동적으로 적용하기 때문에 데이터베이스 객체(테이블, 뷰와 같은)에 대한 세밀한 레벨까지의 보안을 강화할 수 있다.

사용자가 오라클 VPD 보안 정책에 의해 보호된 테이블이나 뷰와 같은 객체를 직·간접적으로 접근하고자 할 때, 오라클 데이터베이스는 사용자의 SQL 문을 동적으로 수정한다. 이러한 변경은 보안 정책을 구현한 함수에 의해 반환된 WHERE 절, 즉 Predicate를 생성한다. 그러므로 SQL문에 동적인 WHERE절을 추가하기 위해서는 그림 2에서 보는 것과 같이 Policy Function을 먼저 정의하고, 생성된 Policy Function을 정책에 첨부해야 한다.

VPD를 이용하여 데이터에 대한 보안 정책을 수립할 때 먼저 사용자의 구별이 이루어져야 한다. 사용자를 구별하기 위해서는 먼저 동일한 접근권한에 따라 집단화하여 그룹에 정책을 적용하는 방법과 그룹을 생성하지 않고 사용자에게 정책을 적용하는 방법이 있다. 첫 번째 접근

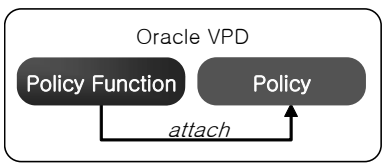

그림 2. Oracle VPD 구조

방법에서, 사용자를 동일한 접근권한에 따라 정책을 적용할 경우 그룹을 생성하기 위해서는 dbms_rls.create_policy_group을 이용한다. 그룹별로 정책을 적용할 경우에는 사용자들이 데이터베이스에 접속을 할 때, 각 사용자에 대하여 그룹을 지정해 주어야 하기 때문에 트리거를 생성해야한다. 그룹 생성이 이루어지면 테이블을 위한 VPD를 구현한다. 전체 VPD 기반구조는 DBMS_RLS PL/SQL 패키지에 의해 제어된다. 어떤 인스턴스를 표시할 것인지에 관한 Role은 정책을 통해 제어된다. 이 정책을 통해 테이블에 대한 질의에 WHERE절이 적용되며, 그 결과로 접근이 제한된다. 즉, WHERE 조건은 Policy Function에 의해 생성된다. 따라서 Policy Function을 생성한 후, Policy Function을 grouped_policy에 적용하여 그룹에 대한 정책을 추가한다. 이로써 VPD를 이용하여 보안 정책을 수립하게 된다. 두 번째 접근 방법에서, 사용자에 대하여 정책을 적용할 경우에는 execute dbms_rls.add_policy를 사용하며 정책을 생성하며 "object_schema"속성을 이용하여 테이블을 소유하고 있는 사용자에게만 정책을 적용할 수 있다. 그 이유는 이 속성의 값으로는 사용자명이 들어가기 때문이다. 그리고 "object_name"속성의 값으로는 "object_ schema"의 값으로 쓰인 사용자가 가지고 있는 테이블명이나 뷰, 또는 syntax를 값으로 가질 수 있다. 그렇기 때문에 사용자에 대하여 정책을 적용할 경우에는 테이블을 소유하고 있는 사용자에게만 정책을 적용할 수 있다.

오라클 VPD 정책에서는 권한부여를 따로 생성하지 않아도 된다는 장점이 있다. 이는 정책을 생성할 경우 "statement_types"속성으로 접근에 대한 권한부여를 할 수 있기 때문이다. 권한부여의 요소로는 SELECT, INSERT, UPDATE, INDEX 및 DELETE 문을 적용할 수 있다. 그러나 정책에 "sec_relevant_cols_opt ⇒ dbms_rls.all_rows" 속성을 사용하였을 때에는 SELECT 문만 적용할 수 있으며, ALTER TABLE 문과 같은 DDL의 필터링을 위해서는 사용될 수 없다.

3. RAC을 위한 구현 모델

이 장에서는 RAC을 구현하기 위한 오라클 VPD 기반 구현 모델에 대하여 기술한다.

3.1 제안 모델 구조

그림 3은 전체적인 RAC을 위한 구현 모델을 보여준다. 제안 모델은 RAC 보안 모델에서 이용한 표준 관계

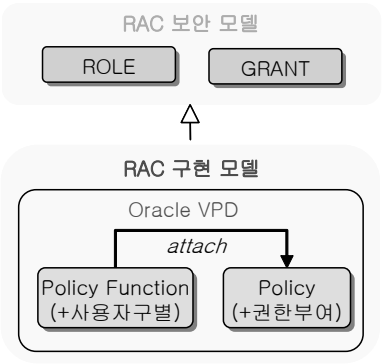

그림 3. 제안 모델 구조

형 보안 모델을 확장한 모델로 RAC 보안 모델에 오라클 VPD를 추가하여 보다 다양한 접근 제어 규칙을 생성한다. 접근 제어 규칙은 사용자에 따라 1)전체 인스턴스, 2)일부 필드, 3)일부 튜플, 4)일부 필드의 일부 튜플, 5)임의의 데이터의 5가지 경우로 접근을 제어할 수 있다. 제안 모델인 RAC을 위한 구현 모델은 전체 인스턴스에 대한 접근 제어 규칙을 생성할 경우에는 RAC 보안 모델에서 이용한 표준 관계형 보안 모델의 GRANT연산을 이용하여 접근 제어를 한다. 그 외 2)일부 필드, 3)일부 튜플, 4)일부 필드의 일부 튜플, 5)임의의 데이터의 경우에는 오라클 VPD 보안 모델을 이용하여 접근 제어를 한다.

오라클 VPD로 접근 제어를 하기 위해서는 앞서 기술하였듯이 먼저 동적 WHERE 절을 생성 가능하도록 하는 Policy Function을 만들어야 한다. 생성된 Policy Function은 특정 객체에 대한 접근 제어 규칙을 생성하기 위하여 정책 생성 시 정책에 첨부되며, 생성된 정책은 사용자에게 적용된다. 정책을 적용하는 방법은 group_policy를 생성하여 여러 명의 사용자에게 정책을 적용하는 방법과 object_schema 즉, 테이블을 소유한 사람에게 정책을 적용하는 방법이 있다. 본 논문에서는 사용자별로 여러 가지 접근 제어 규칙을 정의하여 접근 제어를 하는 것에 목적을 두고 있다. 따라서 group_policy를 생성하여 여러 명의 사용자에게 정책을 적용해야 한다. 이를 위하여 먼저 동일한 접근권한을 가진 사용자에 대하여 집단화해야 한다. 또한, 사용자에 대한 그룹화를 위하여 트리거와 프로시저를 정의해야 한다. 따라서 본 논문에서는 이러한 번거로움을 해결하기 위하여 Policy Function에서 If 문을 추가한다. 사용자의 구분을 위해 “sys_context('userenv', 'SESSION_USER')”문을 이용한다. 예를 들어 사용자가 user1인지 아닌지에 대한 구분을 할 경우 “sys_context

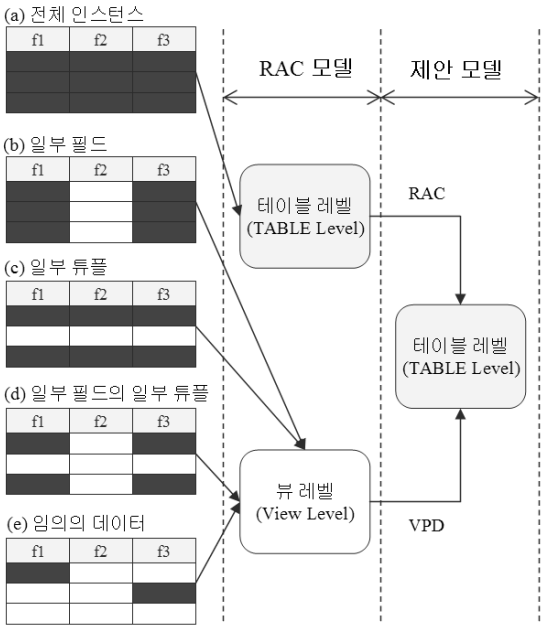

그림 4. RAC과 제안 모델 간 관계성

('userenv','SESSION_USER')= =user1"라는 구문을 사용한다. 사용자의 구분을 한 후 사용자에 맞는 동적인 WHERE절을 반환할 수 있도록 return문을 기술한다. 이로써 Policy Function이 생성되면 Policy를 생성할 때 Policy에 Policy Function을 첨부해주면 사용자에 맞는 접근 제어 규칙이 생성되며 접근 제어가 가능하다. 따라서 일부 데이터에 대한 접근을 제어하고자 할 경우, 가상의 테이블인 뷰를 생성하지 않고도 테이블에 직접 다양한 접근 정책을 적용할 수 있다.

또한, Policy를 생성할 때 “statement_types”속성을 사용하여 사용자에 대한 권한부여를 정의할 수 있기 때문에 GRANT 연산자를 이용하여 각각의 사용자에게 권한부여를 하지 않아도 된다는 장점이 있으며 “sec_relevant_cols_opt ⇒ dbms_rls.all_rows” 속성을 적용하지 않은 Policy에 대하여는 UPDATE, DELETE 권한을 사용자에게 부여할 수 있다는 장점도 있다.

3.2 RAC과 제안 모델 간 관계성

그림 4는 RAC 보안 모델에서의 접근 제어 방법과 제안하는 RAC 구현 모델 간 연관성을 보여준다.

RAC 보안 모델의 경우, 저장된 RDF 온톨로지에 대한 접근 제어를 할 때 테이블과 뷰에 접근권한을 생성한다. 즉 특정 테이블의 일부 데이터에 대한 접근권한을 생성하

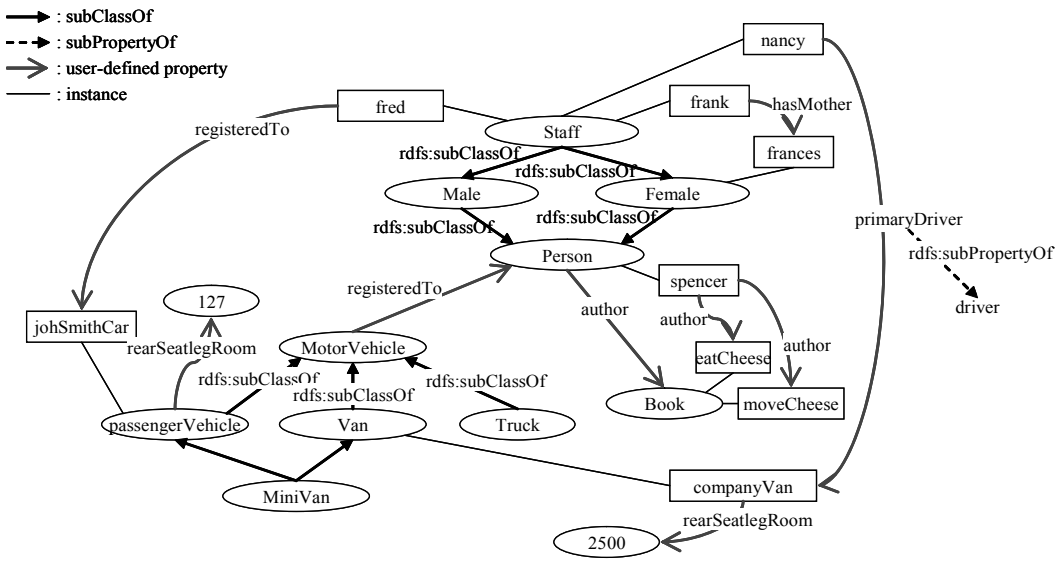

그림 5. RDF Primer 예제

기 위해 먼저 뷰를 생성하고 이에 대한 접근을 허용한다. 이는 하나의 테이블에 대하여 여러 명의 사용자가 접근을 하게 될 경우 사용자에 따라 1)전체 인스턴스, 2)일부 필드, 3)일부 튜플, 4)일부 필드의 일부 튜플, 5)임의의 데이터의 5가지 경우로 접근을 제어할 수 있도록 하기 위한 것이다. 즉, 전체 인스턴스에 대하여 접근 제어를 할 때에는 GRANT 연산자를 이용하여 전체 인스턴스에 대한 권한을 부여한다. 그러나 2), 3), 4), 5)의 경우에는 그에 맞는 접근을 제어하기 위하여 각각의 경우에 맞게 뷰를 먼저 생성해야한다. 그리고 접근 제어를 할 경우 뷰에 대하여 GRANT 연산자를 이용하여 접근을 허용한다.

그러나 앞서 기술하였듯이, RAC 보안 모델은 테이블 전체에 대한 접근 제어 규칙을 제외한 부분적인 데이터에 대한 접근 제어(2), 3), 4), 5) 의 경우)를 위해 뷰를 생성한다. 따라서 부분적인 데이터에 대한 접근 제어 규칙이 증가할수록 뷰의 수 또한 증가한다. 이는 보안 관리자에게 관리의 복잡성과 이에 따른 오버헤드를 초래한다.

그러나 제안 모델에서는 뷰를 생성하여 특정 테이블의 일부 데이터 즉, 일부 필드, 일부 튜플, 일부 필드의 일부 튜플, 임의의 데이터에 대한 접근권한의 생성하던 것을 VPD의 Policy Function을 이용하여 동적인 WHERE절을 가능하게 하는 Policy를 사용자에게 적용시켜 뷰를 생성하지 않고 테이블 레벨에서 직접 이에 대한 접근을 허용한다. 그러므로 가상의 테이블인 뷰를 생성하지 않으므로 테이블 레벨과 뷰 레벨에서의 접근 권한 생성을 테이블 레벨에서 여러 가지 접근 제어(2), 3), 4), 5)의 경우를 포함한)를 위한 접근 권한 생성을 가능하게 한다.

따라서 보안 관리자에게 관리의 복잡성과 이에 따른 오버헤드를 초래한다는 문제점을 보완한다.

4. 구현 및 평가

이 장에서는 실제 웹 온톨로지 예제를 정의하고 사용자를 생성한다. 생성된 사용자에 대하여 권한을 부여하고, 사용자에 맞는 접근 제어 규칙을 정의한다. 또한, Policy Function을 생성하고 생성된 Policy Function을 정책에 추가하여 사용자에게 적용한다. 사용자에게 정책이 잘 적용되었는지 확인하기 위하여 질의를 정의하는 평가가 이루어진다.

4.1 웹 온톨로지 예제 정의

그림 5는 W3C 표준 스펙에 있는 RDF Primer 예제이다[12]. 이 예제는 RAC 보안 모델의 예제로 쓰인 예제와 동일하며, 하나의 테이블로 저장하여 실험한다. 즉, RDF와 RDF-S를 TRIPLE 구조로 정의한다. 예제의 테이블명은 RDF_graph로 정의하며 subject, predicate, object 3개의 필드로 구성된다.

4.2 사용자 및 권한부여 예제

표 1은 실험을 위해 정의한 사용자별 접근 제어 규칙

표 1. 사용자별 접근 제어 규칙

사용자	접근 제어 규칙
user1	전체 인스턴스 접근 금지
user2	predicate 필드 접근 금지
user3	predicate 필드의 값이 rdfs:subClassOf 이 아닌 튜플 접근 금지
user4	predicate 필드의 값이 rdfs:subClassOf 이 아닌 subject, object 필드 접근 금지
user5	subject와 object의 Staff 접근 금지
user6	전체 인스턴스 접근 허용

표 2. 질의 예제

	질의
Q1	select * from RDF_graph;
Q2	select * from RDF_graph where Predicate='rdf:type';
Q3	select * from RDF_graph where subject='Staff';
Q4	select Predicate from RDF_graph;

이다. 사용자는 총 6명으로 user1과 user6은 전체 인스턴스에 대한 접근을 금지하거나 허용하며, user2는 predicate 필드에 대한 접근을 금지하여 앞에서 기술한 일부 필드에 대한 접근을 제어한다. user3은 predicate 필드의 값이 "rdfs:subClassOf"이 아닌 튜플에 대하여 접근을 금지하므로 앞에서 기술한 일부 튜플에 대한 접근을 제어를 한다. 그리고 user4는 predicate 필드의 값이 "rdfs:subClassOf"이 아닌 subject, object 필드에 대하여 접근을 금지하므로 앞에서 기술한 일부 필드의 일부 튜플에 대하여 접근을 제어한다. 마지막으로 user5는 subject필드와 object필드의 값이 "Staff"인 셀에 대하여 접근을 금지하므로 앞에서 기술한 임의의 데이터에 대한 접근을 제어한다.

표 2는 실험을 위해 정의한 질의 예제이다. Q1은 테이블의 전체 데이터를 검색하며 두 번째 Q2는 테이블에서 predicate 필드의 값이 "rdf:type"인 데이터를 검색하는 질의이다. Q3은 테이블의 subject 필드의 값이 "Staff"인 데이터에 대하여 전체 데이터를 검색하며 Q4는 테이블의 predicate 필드에 대한 데이터를 검색하는 질의이다.

4.3 구현 및 실험 결과

앞서 정의한 사용자별 권한부여를 위한 함수 및 정책을 구현한다. 다음은 user3을 위해 구현한 Policy Function 및 정책을 보여준다.

(1) Policy Function 생성

```
create function func3(p_schema varchar2,
p_obj varchar2)
return varchar2 as
  r_val varchar2(1000):='';
begin
  if (sys_context('userenv',
  'SESSION_USER')='USER3') then
    r_val:='Predicate=''rdfs:subClassOf'';
  end if; return r_val; end;
```

(2) Policy 생성

```
execute dbms_rls.add_policy(
        object_schema ⇒ 'manager',
        object_name ⇒ 'RDF_graph',
        policy_name ⇒ 'policy3',
        policy_function ⇒ 'func3',
        statement_types ⇒ 'select' );
```

(1), (2) 에서는 Policy Function 부분에서 사용자에 대한 식별이 이루어져, 사용자에 맞는 동적인 WHERE 절의 반환이 이루어진다. 그리고 Policy Function 적용 예제를 보면, object_name에는 접근 제어 규칙이 적용될 테이블명이, object_schema에는 테이블을 소유하고 있는 사용자명, 그리고 policy_name 에는 정책이름, 그리고 policy_function에는 정책에 추가될 Policy Function명을 기술해 주며, statement_types에는 허용되는 권한에 대하여 기술한다. statement_types의 요소로는 앞에서 기술한 바와 같이 select, insert, delete, update가 있으며 이 논문에서는 select 권한만을 주기로 한다. 이는 RAC 보안 모델에서도 select 권한만을 주었기 때문이다.

그림 6은 일부 필드에 대한 접근을 제어 받는 user2의 계정에서 테이블의 전체 데이터에 대한 검색을 하는 질의 Q1을 한 결과이다. user2는 predicate 필드에 대하여 접근권한이 없으므로 predicate 필드의 데이터가 보이지 않는 것을 확인할 수 있다.

그림 7은 일부 튜플에 대한 접근이 제한된 사용자 user2의 계정에서 Q1을 한 결과이다. user3은 predicate 필드의 값이 rdfs:subClassOf인 튜플에 대해서만 접근이 허용된다. 따라서 그림 7에서 보는 바와 같이 predicate 필드의 값이 rdfs:subClassOf인 튜플에 대해서만 접근이 허용되는 것을 확인 할 수 있다. 즉, 그림 6와 그림 7은 서로 다른 사용자가 같은 질의에 대하여 실행하였을 때 사용자의

```
SUBJECT         PREDICATE    OBJECT
--------------- ------------ --------------
nancy                        Staff
Staff                        Female
spencer                      moveCheese
Male                         Person
companyVan                   2500
frances                      Female
eatCheese                    Book
MiniVan                      PassengerVehic
johnSmithCar                 fred
fred                         Staff
Truck                        MotorVehicle
PassengerVehic               127
Female                       Person
MotorVehicle                 Person
Person                       Book
PassengerVehic               MotorVehicle
spencer                      Person
Staff                        Male
spencer                      eatCheese
moveCheese                   Book
Van                          MotorVehicle
MiniVan                      Van
johnSmithCar                 PassengerVehic
companyVan                   Van
frank                        frances
nancy                        companyVan
frank                        Staff
primaryDriver                driver

28 rows selected
```

그림 6. user2에 대한 Q1의 질의 결과

```
SUBJECT         PREDICATE       OBJECT
--------------- --------------- --------------
Staff           rdfs:subClassOf Female
Male            rdfs:subClassOf Person
MiniVan         rdfs:subClassOf PassengerVehic
Truck           rdfs:subClassOf MotorVehicle
Female          rdfs:subClassOf Person
PassengerVehic  rdfs:subClassOf MotorVehicle
Staff           rdfs:subClassOf Male
Van             rdfs:subClassOf MotorVehicle
MiniVan         rdfs:subClassOf Van

9 rows selected
```

그림 7. user3에 대한 Q1의 질의 결과

권한에 적합한 결과를 보여준다.

4.4 평가

앞서 실제 구현과 실험에 대하여 기술하였으며, 이 절에서는 사용자에 적용된 정책이 제대로 수행되어 접근 제어가 잘 이루어져 있는지에 대한 실험 평가를 하며, 기존 RAC 모델과의 평가를 한다.

표 3은 실험에 대한 평가표이다. 표 3에서 보는 바와 같이 전체 데이터에 대하여 접근권한이 없는 user1에 대하여는 모든 접근이 거부됨을 알 수 있으며 전체 데이터에 대하여 접근권한이 있는 user6의 경우에는 모든 접근이 허용됨을 알 수 있다. 일부 필드 즉, predicate 필드에 대한 user2에 대하여는 predicate 필드를 제외한 일부 튜플에 대하여 접근이 허용되며, 일부 튜플에 대하여 접근

표 3. 실험에 대한 결과

질의	user1	user2	user3	user4	user5	user6
Q1	D	P_F	P_T	P_R	P_R	P_A
Q2	D	D	D	P_R	P_R	P_A
Q3	D	P_F	P_A	P_A	D	P_A
Q4	D	P_F	P_T	P_A	P_A	P_A

- D : 접근불가
- P_T : 일부 튜플 접근허용
- P_A : 전체 접근허용
- P_F : 일부 필드 접근허용
- P_R : 임의의 데이터 접근허용

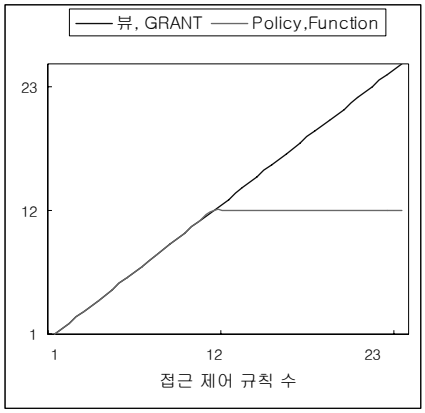

그림 8. 생성 개수

권한이 있는 user3은 일부 튜플에 대하여 접근이 허용된다. 일부 필드의 일부 튜플에 대하여 접근권한이 있는 user4는 일부 필드에 대하여 규칙적으로 접근이 허용되며, 임의의 데이터에 대하여 접근권한이 있는 user5는 임의의 데이터에 대하여 접근이 허용된다. 그러므로 사용자별 접근 제어 규칙에 따라 질의에 대한 적절한 접근을 제어하는 것을 확인 할 수 있다.

그림 8은 접근 제어 규칙의 수에 따른 뷰, GRANT와 Policy, Function 의 개수에 대한 그래프이다. 접근 제어 규칙의 수가 증가함에 따라서 GRANT, 뷰의 개수는 접근 제어 규칙에 비례한다. 그러나 Policy, Function의 개수는 접근 제어 규칙의 수가 12개 이상일 경우에는 12개로 고정이 된다.

이는 Policy의 경우에는 'sec_relevant_cols' 속성의 값과 'sec_relevant_cols_opt=>dbms_rls.ALL_ROWS' 옵션의 유무가 접근 제어 규칙을 생성할 때 영향을 주기 때문이다. 본 논문에서 테이블의 구조는 TRIPLE 구조로 subject, predicate, object 필드로 구성 되어 있다. 따라서 'sec_relevant_cols' 속성의 값으로 들어갈 수 있는 개수

표 4. 생성 최대 개수 계산 모델

RAC 보안 모델	GRANT	n
	View	
RAC 구현 모델 (제안 모델)	Function	$\sum_{i=1}^{2} 3C_i$
	Policy	

는 다음과 같다.

- 한 필드를 값으로 가질 경우 : $3C_1$ = 3가지
- 두 필드를 값으로 가질 경우 : $3C_2$ = 3가지
- 세 필드를 값으로 가질 경우 : $3C_3$ = 1가지

그러나 세 필드를 가지는 경우는 전체 인스턴스에 대하여 접근 제어를 하게 되는 경우이므로 표준 보안 모델을 사용하여 GRANT 연산만을 이용하기 때문에 제외하여 6가지이다.

또한 'sec_relevant_cols_opt⇒dbms_rls.ALL_ROWS' 속성의 유무를 고려하면 6×2=12이다. 표 4는 GRANT, 뷰와 Function, Policy 생성의 최대 개수를 수식으로 표현한 것이다. 여기에서 n은 접근 제어 규칙의 수를 나타내며, i는 자연수를 뜻한다.

5. 결론 및 향후연구

이 논문에서는 RAC 보안 모델의 활용성과 응용성 향상을 위한 구현 모델을 제안하였다. 제안한 구현 모델은 현재 가장 널리 사용되고 있는 가장 대표적인 관계형 데이터베이스 관리 시스템인 오라클을 기반으로 한다. 즉 오라클 VPD 기반의 RAC 구현 모델을 제안하였다.

이 논문에서는 RAC 모델과 오라클 VPD에 대한 관련 연구를 기반으로 RAC 모델을 위한 오라클 VPD 기반의 구현모델에 대하여 기술하고 RAC 모델과 제안 모델간의 관계성에 대하여 기술하였다. 또한 실제 웹 온톨로지 예제와 접근 제어 규칙을 정의하였으며 Policy Function 및 정책을 생성하여 이를 통한 구현과 실험평가를 수행하였다. 실험 및 평가 결과에서, 제안 모델은 RAC 보안 모델과 달리, 별도의 뷰를 생성하지 않고 직접적으로 테이블에 대한 접근 제어 정책을 생성하여 보안 관리자에게 관리의 용이성을 제공한다. 또한 RAC 보안 모델의 실용성, 적용성 및 활용성을 향상시킨다.

이 논문에서는 RAC을 위한 오라클 VPD 기반의 구현 모델을 제안하였다. 따라서 향후에는 오라클 11g에 추가된 기능인 OLS(Oracle Label Security) 기반 구현 모델에 대한 연구와 다른 종류의 관계형 데이터베이스 보안 모델을 이용한 RAC 모델 구현에 관한 연구가 이루어져야 한다. 또한, 현재 RAC 모델은 웹 온톨로지를 기술하는 표준 언어들 중에서 가장 기초적으로 널리 이용되는 언어인 RDF 웹 온톨로지 접근 제어 모델이다. 그러므로 RDF/RDF-S에 기반을 두고 DAML+OIL의 의미 체계를 활용하여 도메인의 지식 체계를 표현 할 수 있는 표준 온톨로지 언어인 OWL 웹 온톨로지 접근 제어 모델의 구현에 관한 연구가 이루어져야 한다.

참 고 문 헌

1. T. Berners-Lee, "The World Wide Web: Past, Present and Future," IEEE Computer special issue, 1996.
2. B. Motik, A. Maedche, and R. Volz, "A conceptual modeling approach for semantics-driven enterprise applications,". In Proceedings of the First International Conference on Ontologies, Databases and Applications of Semantics, Irvine, USA, 2002.
3. D. Fensel, C. Bussler, and A. Maedche, "Semantic Web enabled Web Services," In Proceedings of the International Semantic Web Conference, LNCS, Springer, 2002.
4. W3C, Resource Description Framework, http://www.w3.org/RDF/.
5. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF schema. Recommendation 10 February 2004, W3C (2004).
6. Horst, H.J. ter: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics 3(2–3), 2005, pp. 79–115.
7. A. Graves, C. Gutierrez, "Data representations for WordNet: A case for RDF," 3rd. International WordNet Conference, Jeju Island, Korea, January 2006.
8. 김성아, "OWL을 이용한 공간 온톨로지 구현방법론의 기초적 연구," 대한건축학회논문집 계획계 21권 6호(통권200호), 2005년 6월, pp. 51-58.
9. W3C, Web Ontology Language, http://www.w3.org/2004/OWL/.
10. Z. Pan and J. Heflin, "DLDB: Extending relational databases to support Semantic Web queries," In Workshop on Practical and Scalable Semantic Web Systems, 2nd International Semantic Web Conference (ISWC2003), Springer-Verlag, Lecture Notes in Computer Science,

Vol. 2870, Sundial Resort, Sanibel Island, Florida, USA, October 20-23, 2003, pp. 109-113.
11. J. Broekstra, A. Kampman, and F. v. Harmelen, “Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema,” Springer-Verlag, Lecture Notes in Computer Science, Vol. 2342, June 2002, pp. 54-68.
12. OWLJessKB: A Semantic Web Reasoning Tool, http://edge.cs.drexel.edu/assemblies/software/owljesskb/.
13. Jena Semantic Web Framework, http://jena.sourceforge.net/, 2007.
14. D. Jeong, M. Choi, Y.-S. Jeon, Y.-H. Han, L.T. Yang, Y.-S. Jeong, and S.-K. Han, “Persistent Storage System for Efficient Management of OWL Web Ontology,” Springer-Verlag, Lecture Notes in Computer Science, Vol. 4611, July 2007, pp. 1089-1097.
15. 정동원, “관계형 데이타베이스 기반의 RDF 온톨로지 접근 제어 모델,” 한국정보과학회, 정보과학회논문지:데이타베이스, 제35권, 제2호, 2008년 4월, pp. 155-168.
16. Pete Finnigan, “Using Oracle VPD in The Real World,” UKOUG Unix SIG, January 22, 2008.
17. Oracle® Database Security Guide 11g Release 1 (11.1), February 2008.
18. Frank Manola, Eric Miller, “RDF Primer W3C Recommendation”, February 10, 2004.

부 록

(1) user2를 위한 Policy Function 및 Policy생성

• Policy Function 생성

```
create function func2(p_schema varchar2,p_obj varchar2)
return varchar2 as r_val varchar2(1000):='';
begin
if (sys_context('userenv','SESSION_USER')='USER2')
then r_val:='Predicate is null';
end if; return r_val; end;
```

• Policy 생성

```
execute dbms_rls.add_policy(object_schema⇒'manager',
object_name ⇒ 'RDF_graph', policy_name ⇒ 'policy2',
policy_function ⇒ 'func2', statement_types ⇒ 'select',
sec_relevant_cols ⇒ 'Predicate',
sec_relevant_cols_opt⇒dbms_rls.ALL_ROWS);
```

(2) user4를 위한 Policy Function 및 Policy생성

• Policy Function 생성

```
create function func4(p_schema varchar2,p_obj varchar2)
return varchar2 as r_val varchar2(1000):='';
begin
if (sys_context('userenv','SESSION_USER')='USER4')
then r_val:='Predicate="rdfs:subClassOf"';
end if; return r_val; end;
```

• Policy 생성

```
execute dbms_rls.add_policy(object_schema⇒'manager',
object_name ⇒ 'RDF_graph', policy_name ⇒ 'policy4',
policy_function ⇒ 'func4', statement_types ⇒ 'select',
sec_relevant_cols ⇒ 'Subject, Object',
sec_relevant_cols_opt⇒dbms_rls.ALL_ROWS);
```

(3) user5를 위한 Policy Function 및 Policy생성

• Policy Function 생성

1) create function func5_1(p_schema varchar2,p_obj varchar2) return varchar2 as r_val varchar2(1000):='';

```
begin
if (sys_context('userenv','SESSION_USER')='USER5')
then r_val:='Subject!="Staff"';
end if; return r_val; end;
```

2) create function func5_2(p_schema varchar2,p_obj varchar2) return varchar2 as r_val varchar2(1000):='';

```
begin
if (sys_context('userenv','SESSION_USER')='USER5')
then r_val:='Object!="Staff"';
end if; return r_val; end;
```

• Policy 생성

1) execute dbms_rls.add_policy(

```
object_schema⇒'manager', object_name ⇒ 'RDF_graph',
policy_name ⇒ 'policy5_1', policy_function ⇒ 'func5_1',
statement_types ⇒ 'select',
sec_relevant_cols ⇒ 'Subject',
sec_relevant_cols_opt⇒dbms_rls.ALL_ROWS);
```

2) execute dbms_rls.add_policy(

```
object_schema⇒'manager', object_name ⇒ 'RDF_graph',
policy_name ⇒ 'policy5_2', policy_function ⇒ 'func5_2',
statement_types ⇒ 'select', sec_relevant_cols ⇒ 'Object',
sec_relevant_cols_opt⇒dbms_rls.ALL_ROWS);
```

정 혜 진 (xhyejin86x@kunsan.ac.kr)

2008 군산대학교 수학정보통계학부 정보통계학전공(이학사)
2008～현재 군산대학교 정보통계학과(석사과정)

관심분야 : 모델링, 정보보안, 데이터베이스, 시맨틱 웹

정 동 원 (djeong@kunsan.ac.kr)

1997 군산대학교 컴퓨터과학과(이학사)
1999 충북대학교 전산학과(이학석사)
2004 고려대학교 컴퓨터학과(이학박사)
1999～2000 ICU 부설 한국정보통신교육원(전임강사)
2000～2001 지구넷 부설 연구소(선임연구원)
2002～2004 TTA 표준화 위원회-데이터연구반 SG08.02(특별위원)
2004 고려대학교 정보통신기술연구소(연구조교수)
2005 펜실베니아 주립대학(PostDoc.)
2004～현재 TTA 메타데이터 프로젝트 그룹, PG 606(위원)
2006～현재 ISO/IEC JTC1/SC32 국내위원회(전문위원)
2008～현재 ISO TC211 지리정보 전문위원회(전문위원)
2005～현재 군산대학교 수학정보통계학부 정보통계학전공(교수)

관심분야 : 데이터베이스, 시맨틱 웹, 시맨틱 그리드, 시맨틱 GIS, 유비쿼터스 컴퓨팅, 보안