RDF 온톨로지 접근 제어를 위한 3 계층 온톨로지 뷰 보안 모델

(A Three-Layered Ontology View Security Model for Access Control of RDF Ontology)

정 동 원 † 징 이 신 †† 백 두 권 †††
(Dongwon Jeong) (Yixin Jing) (Dook-Kwon Baik)

요 약 RDF 온톨로지는 XML 트리 모델을 이용하여 표현할 수 있다. 그러나 XML 문서를 보호하기 위해 개발된 XML 보안 모델을 RDF 온톨로지에 적용하는 방법은 부적합하다. RDF는 그래프 모델로서 추론 기능을 제공하므로 새로운 보안 모델의 개발이 요구된다. 이 논문에서는 RDF 온톨로지 접근 제어를 위한 새로운 질의 지향 모델을 제안한다. 제안 모델은 3 계층 온톨로지 뷰를 이용하여 사용자 질의를 재작성한다. 이를 통해 제안 모델은 추론 규칙에 따라 추론 모델을 생성하는 기존 접근 방법의 문제점을 해결한다. 사용자가 방문할 수 있는 접근 가능한 온톨로지 개념들과 인스턴스들을 각각 온톨로지 뷰로서 정의하며, 또한 추론 질의에 대한 제어를 위해 정의한 추론 뷰를 통해 사용자의 추론 기능을 제어할 수 있다. 이 논문에서는 3 계층 뷰를 정의하고 이에 따라 질의를 재작성하는 알고리즘에 대하여 기술한다. 시스템 구조와 구현된 프로토타입에 대하여 기술한다. 마지막으로, 제안 모델과 기존 접근 방법에 대한 실험 및 평가 결과에 대하여 기술한다.

키워드 : RDF, 온톨로지, 접근 제어, 데이타 보안, 온톨로지 뷰

Abstract Although RDF ontologies might be expressed in XML tree model, existing methods for protection of XML documents are not suitable for securing RDF ontologies. The graph style and inference feature of RDF demands a new security model development. Driven by this goal, this paper proposes a new query-oriented model for the RDF ontology access control. The proposed model rewrites a user query using a three-layered ontology view. The proposal resolves the problem that the existing approaches should generate inference models depending on inference rules. Accessible ontology concepts and instances which a user can visit are defined as ontology views, and the inference view defined for controlling an inference query enables a controlled inference capability for the user. This paper defines the three-layered view and describes algorithms for query rewriting according to the views. An implemented prototype with its system architecture is shown. Finally, the experiment and comparative evaluation result of the proposal and the previous approach is described.

Key words : RDF, Ontology, Access control, Data security, Ontology view

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

† 종신회원 : 군산대학교 정보통계학과 교수
djeong@kunsan.ac.kr
†† 학생회원 : 고려대학교 컴퓨터학과
jing@software.korea.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
baik@software.korea.ac.kr

논문접수 : 2007년 7월 3일
심사완료 : 2007년 10월 23일

정보과학회논문지 : 데이타베이스 제35권 제1호(2008.2)

1. 서 론

RDF(Resource Description Framework)는 기계가 읽을 수 있는 형태(Machine-Readable Format)로 지식을 표현하기 위해 개발된 언어로서, 온톨로지 탐색을 통해 보다 나은 추론 기능을 제공할 목적으로 개발된 인코딩 방법이다[1]. 지식 정보 관리를 위한 방법으로서의 온톨로지에 대한 인식이 확산되고 있으며 RDF를 이용한 온톨로지 구축이 증가하고 있다. 이에 따라 온톨로지의 안전한 접근 제어를 위한 보안 모델 개발의 필요성이 대두되고 있다. 비록 온톨로지가 지식 공유를 목적으

로 구축되고 있지만 온톨로지에 대한 다양한 접근성으로 인해 유효한 접근 제어 평가에 대한 연구가 요구된다. 예를 들어, 영화 등급 평가 시스템을 위해 구축된 온톨로지의 경우, NC-17(개념)로 분류된 영화(인스턴스)에 대한 접근이 17세 이하의 사용자에게는 허용되지 않아야 한다. 이와 같이 민감한 정보가 허가되지 않는 사용자에게 노출되지 않도록 하기 위해서는 온톨로지 접근 제어 모델 개발이 요구된다.

지금까지 XML 문서 보안에 대한 많은 연구들이 진행되어 왔다[2-7]. RDF 온톨로지에 대한 접근 제어는 XML 문서에 대한 접근 제어와 비교하여 몇 가지 공통적인 특성을 공유한다. 즉, 클래스는 물론 인스턴스에 대한 보호와 다른 접근 권한을 지닌 다른 사용자에 대한 권한 부여 등과 같은 공통점을 지닌다. 그러나 RDF 온톨로지 접근 제어는 XML 접근 제어와는 다른 특성과 기능이 고려되어야 하며 이는 다음과 같은 RDF의 특성에서 기인한다.

- 트리 모델과 그래프 모델: XML 문서는 트리 모델이다. 그러나 RDF는 개념과 인스턴스들 간의 관계로 연결된 그래프 모델이다. 따라서 XML을 위한 전통적인 XPath 기반의 접근 제어 방법을 RDF 온톨로지에 적용하는 것은 적합하지 않다.
- 추론 기능: RDF 온톨로지는 지식 추론 기능을 제공하는 반면, XML은 추론 기능을 제공하지 않는다. 따라서 XML 문서에 대한 접근 제어와는 달리 RDF 온톨로지에 대한 접근 제어는 추론에 따른 접근 제어 문제를 고려해야 한다.

지금까지 RDF 온톨로지에 대한 질의 처리 및 접근 제어를 위한 다양한 연구들이 진행되어 왔으며 사용자 질의에 따른 온톨로지 뷰 정의 방법, RDF 뷰 언어, 순회 뷰를 이용한 온톨로지 질의 처리 방법 등이 제안되었다[8-14]. 그러나 이러한 접근 방법들은 단순히 RDF 온톨로지 순회 방법에 초점을 두고 있으며 추론 기능을 고려하지 않는다는 문제점을 지닌다. 기존 접근 방법들의 문제점을 요약하면 다음과 같다.

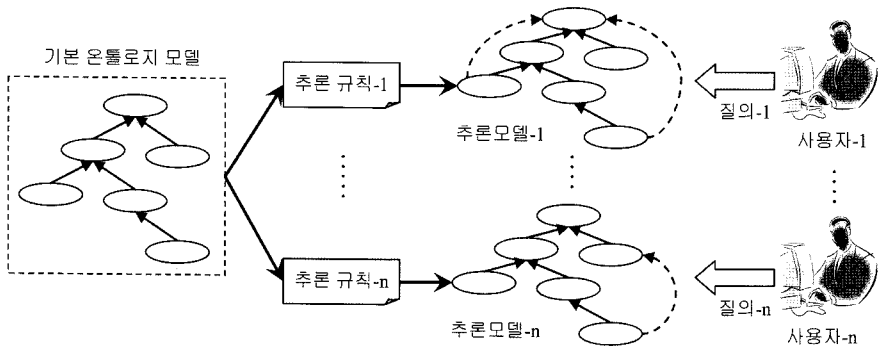
- 세밀하고 다양한 접근 제어 정책 수립의 어려움
  Volz[8], Magkanaraki[9], Noy[10]는 경량의 온톨로지 질의 처리를 위한 온톨로지 뷰 개발, 가상의 온톨로지를 생성할 수 있는 RDF 뷰 언어 및 향상된 온톨로지 순회를 위한 순회 뷰 개념을 각각 제안하였다. 그러나 이들 접근 방법은 온톨로지에 대한 질의 처리와 순회 성능에 초점을 두고 있으며 세부적인 온톨로지에 대한 접근 제어 정책을 제공하지 않는다.
- 개념 및 인스턴스 레벨에서의 접근 제어 기능 미비
  Qin과 Atluri[11]는 개념 레벨 접근 모델(CAC, Concept-Level Access Control)을 제안하였다. 그러나 개념 레벨에서의 접근 제어 기능을 완전하게 제공하지 않으며 인스턴스에 대한 접근 제어 기능은 지원하지 않는다.
- RDF 추론 기능을 위한 접근 제어 방법 부재
  Reddivari와 Finin[12]은 개념과 인스턴스 레벨에 대한 접근 제어 모델을 제안하고 있으나 아직 초기 단계이며 구현되지 않은 상태이다. 무엇보다 추론에 따른 접근 제어 문제에 대해 고려하고 있지 않다.

앞서 기술한 기존 연구의 문제점들 중에서, 추론 영역에 대한 접근 제어 문제를 해결하기 위한 연구들이 진행되었다[15,16]. 이 연구들은 온톨로지 추론 영역의 접근 제어를 위해 백 엔드 온톨로지 추론 모델을 기초로 하고 있다. Jena[15]와 Sesame[16]에 적용된 방법들은 추론 규칙을 이용하여 RDF 온톨로지 기본 모델을 해석한다. 추론이 요구되는 질의에 대한 결과를 생성하기 위해 온톨로지의 소스 문장이나 파생된 문장들이 추론 모델로서 메모리 혹은 데이타베이스 내에 저장된다(그림 1(a)). 이러한 접근 방법은 온톨로지 추론 규칙이 수정될 때마다 해당 추론 모델을 다시 생성해야 하는 문제점을 지닌다. 즉, 접근 제어 메카니즘이 추론 모델에 종속되어 있기 때문에 추론 모델이 변경될 때마다 접근 제어 정책이 수정되어야 한다.

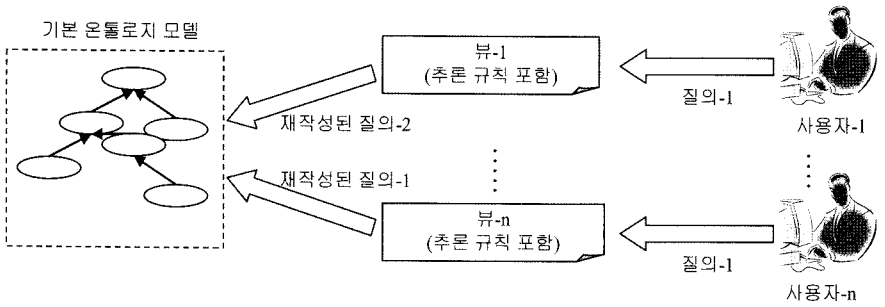
이 논문에서는 앞서 언급한 문제점들을 해결하기 위한 새로운 RDF 온톨로지 접근 제어 모델을 제안한다. 이 논문의 목적은 (1) 클래스 레벨 및 인스턴스 레벨에서의 접근 제어 기능을 제공하고 (2) RDF가 제공하는 추론 기능을 고려하여 추론 영역에 대한 접근을 제어할 수 있도록 하며 (3) 마지막으로, 추론 규칙에 따른 추론 모델에 독립적인 접근 제어 기능을 제공하는데 있다. 이를 위해 온톨로지를 3 계층으로 분류하고 이를 추론 메카니즘과 결합시킨다. 따라서 단순 질의는 물론 추론 가능한 질의에 대해 동일한 접근 제어 방법을 이용할 수 있다. 이를 위해 내부적으로 온톨로지 뷰를 이용한 질의 재작성 연산이 실행된다. 이러한 방법을 통해서 기본 온톨로지 모델을 재사용할 수 있으며 각 사용자를 위한 추론 모델을 생성할 필요가 없다(그림 1(b)).

이 논문의 구성은 다음과 같다. 제2장에서는 사전 정의에 대하여 기술하고 제3장에서는 제안하는 온톨로지 접근 제어 모델에 대하여 기술한다. 제4장에서는 온톨로지 뷰에 대하여 정의하고 제5장에서는 질의 재작성 알고리즘에 대하여 기술한다. 제6장에서는 구현 시스템 및 실험을 통해 제안된 모델과 기존 접근 방법을 비교 평가하고 제7장에서는 결론 및 향후 연구 내용에 대하여 기술한다.

2. 질의 언어



(a) 기존 접근 방법



(b) 뷰에 따른 질의 재작성

그림 1 기존 추론 제어 방법 및 제안 모델의 구조

온톨로지 질의는 임의의 온톨로지 내에 있는 특정 내용에 대한 요청이다. XPath를 이용하는 XML 질의 처리 방법과는 달리 온톨로지 질의 처리를 위해서는 다른 방법이 이용된다. 온톨로지 질의 처리에 대한 해결책은 언어와 온톨로지 모델을 사상시키는 것이다. 이러한 개념에서 RDF 모델에 대한 질의를 위해 RQL[17,18], SPARQL[19] 등이 개발되었다. 이들 언어는 기본적인 질의는 물론 온톨로지 모델을 고려하여 보다 정확한 결과 생성을 가능하게 한다. 복잡한 질의는 기본 패턴을 적용하여 작성하게 되며 이러한 기본 패턴은 트리플 모델을 기반으로 한다. 일반적인 개념에 따라 질의 트리플 패턴을 의미 손실 없이 정의하면 다음과 같다.

정의 1. 질의 트리플 패턴(Query Triple Pattern)

I, B, L을 각각 IRI(Internationalized Resource Identifiers), 공백 노드(Blank Nodes), RDF 리터럴(RDF Literals)이라 하고 T를 I∪B∪L, V를 변수의 유한 집합이라 할 때, 질의 트리플 패턴 pa는 <subject, property, object>의 형식으로 pa∈(I∪B∪V)×(I∪V)×(T∪V)이다. 질의 트리플 패턴은 변수의 개수와 그 위치에 따라 다음과 같이 6 가지 유형을 지닌다.

<$?v_1$, predicate, object>, <subject, $?v_1$, object>, <subject, predicate, $?v_1$>, <$?v_1$, $?v_2$, object>, <subject, $?v_1$, $?v_2$>, <$?v_1$, predicate, $?v_2$>

위 정의된 유형에서, 첫 번째 유형인 <v_1, predicate, object>에 대한 트리플 질의 예로서 <?class, name, "James">를 들 수 있다. 예로 주어진 질의 트리플은 이름이 "James"인 모든 사람을 검색하는 질의이다.

위 질의 트리플 패턴 정의를 기반으로 사용자 질의는 다음과 같이 정의된다.

정의 2. 사용자 질의 (User Query)

사용자 질의는 3-튜플 모델로서 q=(O, π, stmt)이며 O는 대상 온톨로지를 의미한다. π는 최종적으로 반환되는 결과를 나타내는 변수의 집합으로 $\pi \subseteq \{v_i | v_i$는 stmt 내의 변수$\}$로 정의된다. 따라서 stmt는 pa의 집합으로 stmt=([∧|∨]pa)*이다.

$$q\ (O) = SELECT\ \pi \\ WHERE\ \text{stmt}$$

질의는 SELECT 절과 WHERE 절로 표현된다. SELECT 절에서, π는 변수들의 집합이며 사용자에게 반환될 결과를 의미한다. WHERE 절의 stmt는 검색 조건을 기술하며 트리플인 pa의 집합으로 구성된다. 질의 결과는 WHERE 절의 조건을 만족하는 변수들의 값

이 된다. 질의 재작성이 이루지는 동안 접근 제어를 구현하기 위해서 SELECT 절과 WHERE 절에 대한 수정이 요구된다.

위 정의에 대한 예로서, 성(lastName)이 "Jeong"인 사람들의 직업을 검색하는 질의를 생각해 보자. 성이 "Jeong"인 사람을 선택하기 위한 트리플은 <?person, lastName, "Jeong">이고 모든 사람들의 직업을 선택하는 트리플은 <?person, hasJob, ?job>이 된다. 질의에 대한 최종적인 결과는 두 트리플 간의 조인 연산을 통해서 얻을 수 있으므로 stmt는 <?person, lastName, "Jeong">∧<?person, hasJob, ?job>이 된다. 또한 사용자 질의에서 최종적으로 얻고자 하는 값은 직업이므로 π = {job}이 된다. 따라서 전체 질의는 SELECT ?job WHERE <?person, lastName, "Jeong">∧<?person, hasJob, ?job>이다.

3. 접근 제어 모델

이 장에서는 RDF 온톨로지 접근 제어 제안 모델에 대하여 기술한다. 온톨로지 뷰는 사용자가 방문하고자 하는 RDF 온톨로지의 접근을 제한하기 위해 이용된다. 뷰는 온톨로지 관리자에 의해 생성되고 관리된다. 온톨로지 뷰는 인스턴스 레벨 및 개념 레벨에서 사용자가 액세스할 수 있는 내용들을 각각 관리한다. 또한 온톨로지 추론에 따른 영역에 대한 접근 제어를 위해 추가적인 뷰를 정의한다. 제안하는 접근 제어 모델은 다음과 같이 정의된다.

정의 3. 접근 제어 모델은 6-튜플로서 제안 모델 M = (O, V, R, U, ϕ, μ)이다.

- O는 질의 대상이 되는 온톨로지를 의미한다.
- V=(V_C, V_I, V_F)는 RDF 온톨로지 뷰의 집합으로서, V_C, V_I 및 V_F는 각각 개념, 인스턴스 및 추론에 대한 접근 제어를 위해 정의된 뷰를 의미한다.
- R은 온톨로지 사용자 유형 집합으로서 롤(Roles) 집합을 의미한다.
- U는 온톨로지 사용자들의 집합을 의미한다.
- ϕ = R→2^U로서 특정 사용자가 수행해야 하는 롤에 따라 온톨로지 관리자가 정의하며, 하나의 롤은 여러 사용자에게 부여될 수 있다. 따라서 롤:사용자 = 1:n의 관계를 지닌다.
- μ = V→R로서 뷰와 롤 간의 할당 관계를 나타낸다. 롤의 권한에 따라 온톨로지 관리자는 각 롤에 하나의 온톨로지 뷰를 부여하고 사용자의 온톨로지에 대한 접근 권한을 제어하게 된다.

개념 뷰(Concept View)인 V_C는 개념 레벨에서 온톨로지에 대한 접근을 제어하기 위해 정의된다. 개념은 많은 인스턴스와 연관성을 지닌다. 따라서 개념에 대한 제어는 개념과 관련된 모든 인스턴스에 대한 접근 제어를 의미한다. 예를 들어, 인스턴스 집합 I_j = {I_1, I_2, I_3, ..., I_n}가 개념 C_1의 인스턴스 집합이라고 가정해 보자. 이 때, C_1에 대한 접근이 제한될 경우, 해당 인스턴스들에 대한 접근도 함께 제한된다.

인스턴스 뷰(Instance View)인 V_I는 V_C에 기반을 두며 인스턴스 레벨에서의 접근 제어를 담당한다. V_C와 V_I 온톨로지의 추론에 따른 영역에 대한 접근 제어 문제에 대해서는 고려하지 않는다. V_I와 V_C를 기반으로 추론에 대한 접근 제어를 위해 추론 뷰(Inference View, V_F)가 정의되며 V_I와 V_C 두 뷰의 상위에 위치한다. V_F에 대한 접근 제어를 통해 질의 평가가 이루어지고, 허용된 적절한 추론 결과(Controlled Inference Results)만이 반환된다.

4. 3 계층 온톨로지 뷰

사용자 질의가 주어지면 뷰 집합에 따라 온톨로지 질의를 재작성함으로써 RDF 온톨로지 접근을 제어할 수 있다. 임의의 질의에 대해 사용자에게 부여된 뷰에 따라 질의를 수정하는 재작성 메카니즘이 동작하게 된다. 이러한 방법을 통해 온톨로지 관리자는 온톨로지 접근과 추론을 제어할 수 있다.

4.1 개념 및 인스턴스 뷰

V_C와 V_I는 각각 개념 레벨과 인스턴스 레벨에서의 온톨로지 접근 제어 평가를 위해 정의한 뷰이다. 즉, 추론이 이루어지는 질의에 대한 접근 제어 평가를 수행하지 않는다. 따라서 이 뷰들은 각각 사용자가 접근 가능한 온톨로지 개념과 인스턴스의 집합이다. 논문에서, 표기 a.b를 a에 속하는 b라는 의미로 정의한다.

정의 4. 개념 뷰(Concept View, V_C)

개념 뷰인 V_C는 개념들의 집합으로서 다음과 같은 절차를 통해 생성된다.

(1) <C_i, rdf:type, rdfs:Class> 조건을 만족하는 클래스의 집합 C_i를 선택하고 선택된 C_i를 시드 클래스 (Seed Classes) 라고 정의한다.

(2) 각 선택된 클래스에 대하여 <C_j, rdfs:subClassOf, C_i> 조건을 만족하는 클래스인 C_j를 판별하여 V_C에 추가한다.

(3) 각 C_j에 대해서 반복적으로 (2)의 프로세스를 수행하면서 하위 클래스를 지니지 않을 때까지 연산을 반복한다.

결과적으로, 하위 클래스와 함께 시드 클래스들이 V_C에 추가된다. V_C는 클래스 계층 트리의 집합으로 구성되며 각 트리의 루트 클래스는 시드 클래스가 된다. 이러한 구조는 특정 개념을 방문할 수 있는 사용자가 그 하위 클래스까지 방문하여 정보를 검색할 수 있게 해

준다.

예제로서, Professor 클래스의 하위 클래스를 FullTimeLecturer, AssistantProfessor, AssociateProfessor, FullProfessor라고 하고 각 하위 클래스들은 더 이상의 하위 클래스를 지니지 않는다고 가정하자. 이 때, C_i로서 Professor 클래스가 주어지면 <C_i, rdf:type, rdfs:Class> 조건을 만족하므로 시드 클래스가 된다. 다음 단계는 이에 대한 하위 클래스를 검색하는 단계로서, 조건 <C_j, rdfs:subClassOf, C_i>는 <C_j, rdfs:subClassOf, Professor>와 동일한 트리플로 표현된다. 나머지 개념들에 대해 위 조건을 만족하는 모든 클래스를 판단하게 되며, 따라서 C_j는 다음과 같은 집합을 얻게 된다.

C_j={FullTimeLecturer, AssistantProfessor, AssociateProfessor, FullProfessor}

이러한 연산을 반복적으로 수행하면서 FullTimeLecturer, AssistantProfessor, AssociateProfessor, FullProfessor의 하위 클래스가 존재하는지를 확인한다. 위 예제의 경우, FullTimeLecturer, AssistantProfessor, AssociateProfessor 및 FullProfessor의 하위 클래스가 존재하지 않으므로 뷰 생성을 완료하게 된다.

정의 5. 인스턴스 뷰(Instance View, V_I)

인스턴스 뷰, V_I는 (V_C, I)로서 정의되며 I는 다음 조건을 만족한다.

- $\forall I_i \in I$에 대해서 <I_i, rdfs:type, C_i>이고 $C_i \in V_I.V_C$
- <I_j, owl:sameAs, I_i>인 조건을 만족하는 $\forall I_j$에 대해서 $I_i \in I$

인스턴스는 특정 클래스에 속하기 때문에 V_I를 정의하기 위해서는 먼저 해당하는 클래스에 대한 접근이 허용되어야 한다. 만일 V_I의 접근 가능한 인스턴스 집합이 널이면 해당 V_C와 동일하게 된다. 질의가 주어지면 V_C와 V_I를 질의에 대한 접근 제어 평가에 이용한다. 따라서 V_C와 V_I에 포함된 클래스와 인스턴스만이 결과로서 사용자에게 반환된다.

4.2 추론 뷰(Inference View)

추론은 RDF 온톨로지의 특징 중 하나로서 추론 뷰는 온톨로지에 대한 추론 기능에 대한 접근을 제한하거나 허용할 수 있는 메카니즘을 제공한다. 추론 뷰는 두 가지 사항을 고려하여 정의해야 한다.

- 질의가 추론을 요구하는지를 어떻게 판단할 것인가에 대한 문제
- 추론이 요구되는 질의인 경우, 어떻게 제어할 것인가에 대한 문제

첫 번째 고려 사항에서, 질의가 추론 가능한지를 판단하기 위해 추론 규칙을 정의한다. 추론 질의 제어를 위해 추론 뷰는 사용자 추론 기능을 제한할 수 있도록 전이 범위(Transitive Range)를 정의한다. 즉, 추론으로 인해 전이할 수 있는 노드를 제한할 수 있도록 정의한다. 이를 통해 제한된 범위 내에서만 추론이 이루어지도록 한다.

기본적으로 전체 추론 규칙은 조건부와 결론부로 구성된다. 조건을 만족하는 트리플이 결론부에 기술된 트리플 형태로 반환된다. 온톨로지 관리자 측면에서 온톨로지 질의가 추론 조건을 만족하면 결론부에 정의된 추론 결과를 반환하도록 해 주어야 한다. 이 논문에서는 다양한 추론 규칙 중 하나인 트리거 규칙(Trigger Rules)을 다음과 같이 정의한다.

트리거 규칙의 조건은 두 개의 트리플로 구성되며 각 트리플은 속성(Property)을 기술한다. 속성은 추론 기능을 포함하는 추론 속성(Inferable Property)과 추론을 트리거하는 트리거 속성(Trigger Property)으로 분류된다. 두 개의 트리플은 상호 추론 가능한 논리 곱 연산자로 연결된다. 추론 방향에 따라 트리거 규칙은 PTR(Positive Trigger Rule)과 NTR(Negative Trigger Rule)로 분류된다.

정의 6.1. PTR(Positive Trigger Rule)

PTR은 <n_1, $predicate_{t1}$, n_2>∧<n_2, $predicate_i$, n_3>→<n_1, $predicate_{t1}$, n_3>과 같은 형식으로 표현되며, $predicate_i$는 온톨로지 개발자에 의해서 정의되거나 RDF 내에 정의된 추론 속성(Inferable Property)을 의미한다. $predicate_{t1}$은 $predicate_i$로 진행되는(순방향으로 진행되는) 추론을 트리거하기 때문에(즉, n_2로부터 n_3로) PTR(Positive Trigger Rule)이라고 정의하며 이러한 속성을 PTP(Positive Trigger Property)라고 정의한다. 만일 $predicate_{t1}$이 $predicate_i$와 동일하다면 이 때 $predicate_i$를 전이 속성(TP, Transitive Property)이라고 정의한다.

NTR은 앞선 PTR 정의와 유사하지만 전이적인 특성, 즉 전이 속성이 아니라는 측면에서 차이가 있으며 정의는 다음과 같다.

정의 6.2. NTR(Negative Trigger Rule)

NTR(Negative Trigger Rule)은 <n_1, $predicate_{t2}$, n_2>∧<n_3, $predicate_i$, n_2>→<n_1, $predicate_{t2}$, n_3>와 같은 형식으로 정의된다. $predicate_i$는 온톨로지 개발자에 의해 정의되거나 RDF 내에 정의된 추론 속성이다. $predicate_{t2}$는 순방향이 아닌 역방향(Negative Direction)으로 추론을 트리거하므로 (즉, n_3에서 n_2 방향으로) NTP(Negative Trigger Property)라고 정의한다.

다음은 앞서 정의한 트리거 규칙들에 대한 예를 보여준다.

예제 1. 만일 온톨로지 개발자가 추론 속성으로서 locatedIn이라는 속성을 정의한다면 다음과 같은 트리거 규칙을 정의할 수 있다.

- 임의의 상품이 특정 장소 (place_a)에서 제조되고, 제

조했던 장소가 다른 장소 (place_b) 내에 위치할 때, 상품이 place_b에서 제조되었다고 말할 수 있다. 따라서 <product, madeIn, place_a>∧< place_a, locatedIn, place_b>→<product, madeIn, place_b>와 같은 트리거 규칙을 정의할 수 있으며 madeIn은 PTP가 된다.

- 특정 법률이 임의의 장소(place_a)에서 발효되어 유효하다면 그 법률이 유효한 장소 place_a에 속해 있는 다른 장소인 place_b에서도 유효하다고 말할 수 있다. 따라서 이는 <law, appliedTo, place_a>∧<place_b, locatedIn, place_a>→<law, appliedTo, place_b>와 같이 정의되며 appiedTo 트리거 속성은 NTP(Negative Trigger Property)가 된다.
- <place_a, locatedIn, place_b>∧< place_b, locatedIn, place_c>→<place_a, locatedIn, place_c>에서 알 수 있듯이, locatedIn 또한 전이 속성이다.

위 예제에서, 질의 <?x, madeIn, place_b>를 고려해 보자. 이 질의에 대한 술어(predicate)는 예제 1의 첫 번째 트리거 속성과 일치하므로 place_a에서 제조된 상품이 질의 결과로서 반환된다. 이와 같은 결과를 생성하기 위해 주어진 질의 <?x, madeIn, place_b>를 재작성해야 한다. 트리거 규칙에 따라 질의를 재작성하는 세부적인 절차와 알고리즘은 5장에서 기술한다.

표 1은 RDF/S에 기술된 트리거 규칙들의 패턴을 보여준다.

추론 가능한 질의들이 식별되면 추론 기능 제어 방법을 고려해야 한다. 일단 트리거 규칙이 재작성 되면 질의를 통해 추론 결과를 얻을 수 있다. 그러나 추론 결과를 반환하기 전에, 정보 노출 방지를 위한 접근 제어 평가가 선행되어야 한다. 이 논문에서 추론 뷰인 VF는 전이 범위 (Transitive Range)를 정의함으로써 추론 범위, 즉 허용되는 유효한 질의를 제어할 수 있다.

정의 7. 추론 뷰(Inference View, V_F)

추론 뷰 V_F (O) = (V_I(O), T, H_p)로서 정의된다.

- V_I(O): 대상 온톨로지의 해당 인스턴스 뷰
- T: t∈T에 대하여 t=(Trigger_Property$_{pos}$(p_i), Trigger_Property$_{neg}$(p_i), [N_s(p_i), N_e(p_i)])이다.
  - p_i는 추론 가능한 속성을 의미한다.
  - Trigger_Property$_{pos}$(p_i)와 Trigger_Property$_{neg}$(p_i)는 p_i에 대한 PTP와 NTP의 집합을 의미한다.
  - [N_s(p_i), N_e(p_i)]는 선택 부분으로서 만일 p_i가 전이 속성 (TP, Transitive Property)이면, N_s(p_i) = {n_{s1}, n_{s2}, …, n_{sm}}, N_e(p_i) = {n_{e1}, n_{e2}, …, n_{en}}, n_{si}, n_{ej}: $s_1 \le s_i \le s_m$, $e_1 \le e_j \le e_n$은 p_i를 트래킹하여 접근할 수 있는 노드들을 의미한다.
- H_p는 (P, $\prec$)로서 표현되며 부분적으로 순서화 된 속성 집합을 의미한다. 각 $p_i \in P$에 대하여 집합 {$p_j \in P$: $p_j \prec p_i$}는 subPropertyOf에 의해 순서화 된다. 즉, (P, $\prec$)는 온톨로지의 속성 계층에 대한 정보를 지닌다.

전이 추론(Transitive Inference)을 제어하기 위해서는 추론에 대한 허용 여부와 함께 추론에 따라 전이되는 단계, 즉 전이 범위를 정의해야 한다. 따라서 V_F는 전이 추론을 제어하기 위해 정의된 전이 범위를 포함한다. 위 정의에서, N_s와 N_e는 전이 범위를 정의하기 위한 노드들의 집합이다. N_s는 p_i의 전이가 시작되는 노드를 정의하며 N_e는 p_i의 전이가 종료되는 노드를 정의한다.

다음은 온톨로지 뷰의 사용 예를 보여준다.

예제 2. 그림 2는 온톨로지 뷰를 이용하여 동일한 질의에 대한 사용자에 따라 접근을 제어하는 예를 보여준다. 온톨로지 내에 있는 place3에 대해 유효한 법규 검색을 위한 동일한 하나의 질의(<?x, appliedTo, place3>)가 여러 사용자에 의해 주어졌다고 가정하자. 또한 주어진 네 개의 사용자 롤에서, 단지 관리자만이 law2에 대한 결과를 얻을 수 있으며 선임 관리자(Senior Manager)나 VIP 고객만이 추론 질의를 수행하여 그 결과를 얻을 수 있다고 가정한다.

이러한 가정하에서, 유효한 접근 허용을 위해 관리자

표 1 트리거 규칙들에 대한 패턴

트리거 규칙	트리거 속성	추론 속성
<x, rdfs:subPropertyOf, y>∧<y, rdfs:subProeprtyOf, z> → <x, rdfs:subPropertyOf, z>	rdfs:subPropertyOf (PTP, TP)	rdfs:subPropertyOf
<x, rdfs:subClassOf, y>∧<y, rdfs:subClassOf, z> → <x, rdfs:subClassOf, z>	rdfs:subClassOf (PTP, TP)	rdfs:subClassOf
<x, rdf:type, y>∧<y, rdfs:subClassOf, z> → <x, rdf:type, z>	rdf:type (PTP, !TP)	rdfs:subClassOf
<x, rdfs :domain, y>∧<y, rdfs:subClassOf, z> → <x, rdfs:domain, z>	rdfs:domain (PTP, !TP)	rdfs:subClassOf
<x, rdfs:range, y>∧<y, rdfs:subClassOf, z> → <x, rdfs:range, z>	rdfs:range (PTP, !TP)	rdfs:subClassOf

※ PTP: Positive Trigger Property; NTP: Negative Trigger Property; TP: Transitive Property

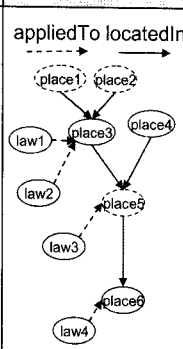
롤 (Role)	초기 질의 (Original)	온톨로지	온톨로지 뷰			결과
			V_C	V_I	V_F	
일반 고객	Select ?x Where <?x, appliedTo, place3>		Law, Region	law1,law3	Null	law1
VIP 고객			Law, Region	law1,law3	$Trigger\_Property_{neg}$ (locatedIn)={appliedTo}, Ns={place1, place2}, Ne = {place5} Hp = Ø	law1, law3
관리자			Law, Region	law1, law2, law3	Null	law1, law2
선임 관리자			Law, Region	law1, law2, law3	$Trigger\_Property_{neg}$ (locatedIn)={appliedTo}, Ns={place1, place2}, Ne = {place5} Hp = Ø	law1, law2, law3

그림 2 온톨로지 뷰와 접근 권한 정의를 위한 온톨로지 뷰와 실행 예제

는 서로 다른 롤에 서로 다른 뷰를 부여해야 한다. 그림 2에서, law2에 대한 접근을 제어하는 V_I를 일반 고객(General Client)에게 할당함으로써 law2에 대한 접근을 방지한다. 반면, 선임 관리자나 VIP 고객에게는 추론 결과까지 얻을 수 있는 V_F를 할당한다. V_F는 appliedTo가 locatedIn의 NTP임을 선언하게 되고 전이 범위가 (place1, place2)와 place5 사이에 존재하도록 제한한다.

결론적으로, VIP 고객과 선임 관리자의 질의에 대해서, place3이 place5에 위치해 있기 때문에 law3을 포함하는 질의 결과를 얻을 수 있도록 질의를 재작성한다. 또한 이들 사용자 그룹은 place6이 전이 범위를 벗어나기 때문에 law4를 액세스할 수 없게 된다. 예제에서 알 수 있듯이, 온톨로지 뷰는 롤의 권한을 정의하기 위한 방법을 제공한다. 질의가 해당 뷰에 의해 평가된 후, 동일한 질의에 대해 사용자 별로 다른 결과가 반환된다.

5. 질의 재작성 알고리즘

질의가 주어지면, 먼저 사용자 롤에 부여한 온톨로지 뷰를 검색하여 평가한다. 질의 재작성 연산을 통해 사용자에 의해 주어진 질의를 재작성하고 이를 실행한다. 복잡한 질의는 논리곱과 논리합 그리고 트리플 패턴을 이용하여 변환하며 논리합으로 연결된 각 트리플 패턴은 독립적으로 처리된다[20]. 따라서 이 논문에서는 WHERE 절의 stmt가 하나의 트리플로 구성되어 있는 단순한 질의에 대한 재작성 과정에 대해서만 기술한다.

추가적으로, 트리플 패턴에 두 개의 변수가 있을 경우, 먼저 각 변수의 값을 구하는 질의를 수행한다. 해당 변수에 값을 할당하고 다른 변수를 배제시킴으로써 하나의 변수만을 포함하는 질의를 재작성한다. 예를 들어, 예제 <subject, $?v_1$, $?v_2$>가 주어졌을 때, 먼저 변수 $?v_1$과 $?v_2$에 대한 값을 구한다. 이 때, 두 변수의 값들을 l, k라 할 때, 주어진 사용자 질의는 다음과 같이 두 개의 질의 (<subject, l, $?v_2$>, <subject, $?v_1$, k>)로 재작성된다.

5.1 개념 뷰와 인스턴스 뷰 강화 알고리즘

질의에 대한 V_C와 V_I의 강화(Enforcement) 절차는 각각 그림 3, 그림 4와 같다. 알고리즘은 서술의 용이성과 직관적인 이해를 돕기 위해 트리플을 이용하여 의사코드로 표현한다. 그러나 이를 구현하기 위해서는 선택된 질의 언어에서 허용하는 어휘를 이용하여야 한다.

Concept_Control() 함수는 사용자 질의 결과에 개념 뷰를 이용하여 제어할 수 없는 부분이 포함되지 않도록 논리곱 연산자 (∧)를 추가하여 질의를 재작성한다. 만

```
Algorithm 1. Enforcing V_C on a query
q = Select ?v Where stmt
Function Concept_Control(Query q, V_C con_view)
    If (?v is instance)
        For each c∈ con_view
            q.stmt ← q.stmt∧(∨<?v,typeOf, c>);
        End For
    End If
    If (?v is concept)
        For each c ∈ con_view
            q.stmt ← q.stmt∧(∨<?v,isSameAs, c>);
        End For
    End If
    Return q;
EndFunction
```

그림 3 질의에 대한 VC 강화 알고리즘

```
Algorithm 2. Enforcing V_I on a query
q = SELECT ?v WHERE stmt
Function Instance_Control(Query q, V_I ins_view)
    If (?v is instance)
        For each i ∈ ins_view.I
            q.stmt ← q.stmt∧(∨<?v,isSameAs, i>);
        End For
    End If
    If (?v is concept)
        q ← Concept_Control(q, ins_view.V_C)
    End If
    Return q;
End Function
```

그림 4 질의에 대한 VI의 강화 알고리즘

일 변수 ?v의 값이 인스턴스인 경우, 개념 뷰 내에 있는 임의의 개념 타입으로 제한하여 논리합 연산자 (∨)를 이용하여 질의를 재작성한다. 변수 ?v의 값이 클래스인 경우, 개념 뷰에 있는 임의의 개념 c와 동일하도록 제한한다. 이러한 제한 또한 논리합 연산자를 이용하게 되며, 마지막으로 알고리즘은 재작성된 질의를 반환하게 된다.

Instance_Control() 함수는 인스턴스 뷰를 통해 허용되지 않는 인스턴스의 접근을 제어하기 위한 알고리즘이다. 따라서 유효 권한을 지니지 않은 사용자가 임의의 인스턴스에 접근하고자 할 때, Instance_Control() 알고리즘에 의해 허용 여부가 평가된다. 이 알고리즘 또한 앞서 기술한 개념 강화 알고리즘과 매우 유사하다. 만일 변수 값이 인스턴스 일 경우, 인스턴스 뷰 내에 있는 특정 인스턴스 i와 동일하도록 제한된다.

5.2 추론 뷰 강화 알고리즘

그림 5는 추론 뷰를 강화하기 위한 알고리즘을 보여준다. 그림 5의 알고리즘에서, ins_view_enforced 플래그는 인스턴스 뷰가 강화되었는지를 나타내며 temp_r은 중간 결과를 저장하기 위해 이용된다. 일반적으로 질의 재작성은 두 단계로 이루어진다.

```
Algorithm 3. Enforcing V_F on a query
Function Inference_Control(Query q, V_F  inf_view)
 1  Boolean ins_view_enforced ← false;
 2  Set temp, temp_r,  result;
 3  Case  q.stmt = <?v_1, predicate, object>
 4    If ( predicate∈(P, ≺) )            // predicate has sub properties.
 5      For(∀sub_pre∈(P,≺) AND sub_pre≺predicate)
 6        q.stmt←q.stmt∨<?v_1, sub_pre, object>;
 7      End For
 8      Instance_Control(q, V_F.V_I) ;  //enforce inference view on rewritten query
 9      ins_view_enforced ← true;
10    End If
11    If (predicate∈ Trigger_Property_pos(predicate))      //predicate is transitive
12        If (ins_view_enforced = false)
13            q← Instance_Control(q, V_F.V_I);
14            ins_view_enforced ← true;
15        End If
16        temp_r←Execute q;
17        result ←result ∪temp_r          //store intermediate results
18        q.stmt← φ
19        For(∀r∈temp_r AND r∈[N_s(predicate), N_e(predicate)])
20            q.stmt← q.stmt∨<?v_1, predicate, r>
21        End For
22    End If
23    If(predicate∈ Trigger_Property_neg(predicate'))
24        If(predicate' is NOT transitive)
25          q.stmt←q.stmt∨(<?v_1,predicate, ?v_2>∧<object,predicate',?v_2>);
26        Else   /*predicate' is transitive*/
27        temp ← Execute "Select ?x Where <object, predicate',?x>"
            /*seek all successors of object tracing predicate'.*/
28        For(∀r∈temp AND r∈[N_s(predicate'), N_e(predicate')])
29            q.stmt← q.stmt∨<?v_1, predicate, r>;
30        End For
31        If (ins_view_enforced = false)
32            q←Instance_Control(q, V_F.V_I);
```

```
33              ins_view_enforced ← true;
34          End If
35      End If
36      EndIf
37      If(predicate∈ Trigger_Property_pos(predicate')AND(predicate'≠predicate))
38        ..... /* similar to (predicate∈ Trigger_Property_neg(predicate') */
39      End If
40      result ← Execute q
41  End Case
42  Case  <subject, predicate, ?v1>
43      If ( predicate∈(P, ≺) ) .....          /*similar to case <?v1, p, o>*/
44      If(predicate∈Trigger_Property_pos(predicate))...  /*similar to case <?v1, p, o>*/
45      If(predicate∈ Trigger_Property_pos(predicate') AND predicate'≠predicate)
46            .....
47          temp_r← Execute q;
48          result ←result ∪temp_r                    //store intermediate results
49          q.stmt← φ
50          For(∀r∈temp_r AND r∈[Ns(predicate'), Ne(predicate')])
51              q.stmt← q.stmt∨<r, predicate', ?v1>
52          End For
53          result ←result∪Execute q
54      End If
55      If(predicate∈ Trigger_Property_neg(predicate'))
56          ..... /* similar to (predicate∈ Trigger_Property_pos(predicate') */
57  End Case

58  Case <subject, ?v1, object>
59      r← Execute q;
60      result←result∪{r} ;
61      If r∈(P, ≺)
62          result←result∪{r'∈P: r'≺r} ;
63      End If
64  End Case
65  return result;
End Function
```

그림 5 질의에 대한 VF 강화 알고리즘

첫 번째 단계에서, 입력된 질의를 분석하여 트리플 패턴을 식별한다. 트리플 패턴 <?v_1, predicate, object>를 예로 들어보자. 만일 predicate이 H_p 내에 존재하면, 하위 술어(sub_pre)를 포함하는 모든 트리플을 생성하여 q.stmt에 추가한다(라인 4~10). 앞선 연산이 수행된 후, predicate이 TP, PTP 또는 NTP인지에 따라 세 가지 경우의 수가 존재한다.

먼저, predicate이 TP인 경우(라인 11), q를 실행하여 그 중간 결과를 temp_r에 저장한다. predicate의 전이 범위(즉, [N_s(predicate), N_e(predicate)]) 내에 있는 중간 결과는 새로운 트리플을 생성하기 위해 이용된다.

predicate이 predicate'의 NTP, 즉 Trigger_Property$_{neg}$인 경우(라인 23), 또 다른 두 가지의 경우가 발생한다. 만일 predicate'이 !TP(Not Transitive Property)이면 q.stmt는 새로운 트리플을 추가하게 된다. 그렇지 않을 경우, 즉 predicate'이 TP이면 모든 후위 속성들을 탐색하여 추가하게 된다(라인 26). predicate'의 전이 범위 (즉, [N_s(pre'), N_e(pre')]) 내에 있는 모든 후위 속성들은 새로운 트리플을 생성하는데 이용된다.

마지막으로, predicate이 predicate'의 PTP, 즉 Trigger_Property$_{pos}$인 경우에도 재작성 절차는 predicate이 NTP인 경우와 유사하다(라인 37). 함수의 마지막 단계에서, 재작성된 q는 최종 결과를 사용자에게 반환하기 위하여 실행된다.

예제 3. 그림 6은 그림 2의 예제를 이용하여 알고리즘 3을 명시적으로 보여주기 위한 예제이다. 질의 예제로서 st=<?v_1, appliedTo, place3>이 주어졌다고 가정해 보자. VIP 고객의 질의가 예제 2에 표현된 온톨로지를 대상으로 할 때, 고객에게 부여된 추론 뷰가 검색된다.

st는 <?v1, predicate, object>의 형태로 식별된다. $V_F.H_p$ = ∅이고 appliedTo∈ Trigger_Property$_{neg}$(loca-

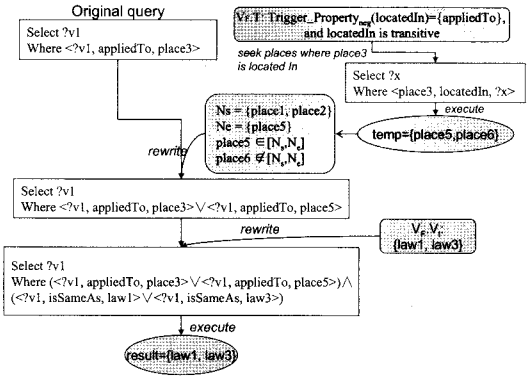


그림 6 질의 재작성 예제

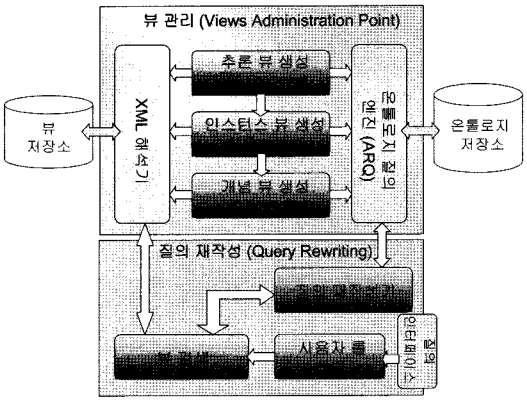


그림 7 OACE 구조

tedIn) = {appliedTo}이므로 locatedIn 특성에 따라 st를 처리하게 된다. locatedIn은 TP이므로 locatedIn을 거치는 place3의 전위 노드들을 미리 획득하게 된다. 즉, 질의 <place3, locatedIn, ?x>가 실행되어 반환된 결과인 {place5, place6}이 temp에 임시 저장된다. [Ns(locatedIn)={place1, place2}, Ne(located)={place5}]에 의해 평가가 이루어지며, 임시 결과인 {place5, place6}에서 단지 place5만이 범위 내에 존재하게 된다. 따라서 st←st∨<?v1, appliedTo, place5>이며 결과적으로, st=<?v1, appliedTo, place3>∨<?v1, appliedTo, place5>가 된다.

다음 단계에서, st는 VF.VI에 의해 처리되며 st는 (<?v1, appliedTo, place3>∨<?v1, appliedTo, place5>)∧(<?v1, isSameAs, law1>∨<?v1, isSameAs, law3>)과 같은 질의로 재작성된다. 마지막으로, 질의가 실행되고 그 실행 결과로서 {law1, law3}이 반환된다.

6. 프로토타입 시스템 구현 및 평가

이 장에서는 시스템 프로토타입 시스템 구현을 위한 시스템 아키텍처를 보이고 3 계층 뷰 정보 관리를 위한 XML 스키마 구조에 대하여 기술한다. 또한 제안 모델과 기존 접근 방법에 대한 실험을 위한 예제 온톨로지 및 질의 패턴을 정의하고 실험 결과 및 평가 결과에 대하여 기술한다.

6.1 시스템 구현

이 절에서는 앞서 기술한 RDF 접근 제어 모델을 기반으로 프로토타입 시스템의 구현 결과에 대하여 기술한다. 구현된 프로토타입 시스템을 OACE(Ontology Access Control Environment)라 명명한다. 그림 7은 구현된 프로토타입 시스템인 OACE의 구조를 보여준다.

구현된 프로토타입 시스템, OACE는 온톨로지 질의 엔진으로서 ARQ[21]를 이용하며 ARQ는 Jena에서 SPARQL 질의를 처리할 수 있도록 구현하여 제공하는 엔진이다. 온톨로지 관리자는 VAP(뷰 관리, Views Administration Point)와의 상호작용을 통해서 다양한 뷰를 생성하고 롤에 뷰를 할당한다. 이러한 프로세스는 주로 개념과 속성 계층을 처리하기 위해 Jena에서 제공하는 프로시저를 호출함으로써 처리된다. 개념 계층 처리를 위한 프로시저는 개념 뷰를 구성하기 위해 이용되며 속성 계층 처리를 위한 프로시저는 추론 뷰의 생성에 이용된다.

뷰는 전달되는 질의의 롤에 따라 사전에 생성 및 분류되어 XML 파일로 저장된다. 질의가 주어지면 URI-dentification(사용자 롤 식별, User Role Identification)은 사용자의 롤을 식별한다. VRetrieval(뷰 검색, Views Retrieval)은 VRepository(뷰 저장소, Views Repository)로부터 해당 뷰들을 검색하고 초기에 제공된 사용자 질의와 함께 뷰를 QRewriter(질의 재작성기, Query Rewriter)에 전달한다. QRewriter는 WHERE 절의 stmt로 기술된 문장에서 트리플 패턴을 분석하여 질의를 재작성한다. 중간 결과와 최종 질의에 대한 결과를 얻기 위해 QRewriter와 온톨로지 질의 엔진인 ARQ 간 상호작용이 지속적으로 이루어진다.

추론 뷰의 XML 스키마의 구조는 그림 8과 같다.

추론 뷰의 정의에 따라 스키마는 세 부분으로 구성된다. Instance_View는 Concept_View를 내포하게 되며 접근 가능한 인스턴스를 정의한다. Trigger_Rule 부분은 각 추론 가능한 속성에 대한 트리거 속성 (NTP와 PTP)과 [Ns, Ne]를 정의한다.

시스템 프로토타입은 Intel Pentium D CPU 2.66GHz, 512M DDR2 메모리 환경하에서 구현되었으며 평가를 위한 실험 또한 동일한 환경에서 이루어졌다. 질의 성능 평가를 위한 벤치 마크 온톨로지는 온톨로지 생성 도구인 UBA(Univ-Benchmark Artificial Data Generator)를 이용하여 생성한다. UBA는 Lehigh 대학[22]에서 자

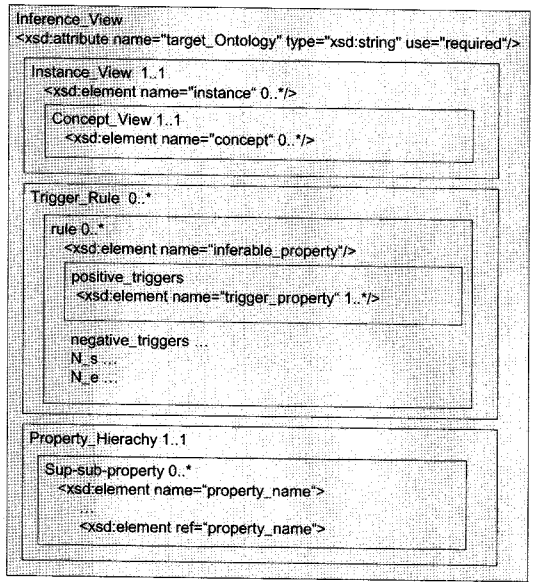


그림 8 온톨로지 뷰를 위한 XML 스키마 구조

체 프로젝트(SWAT 프로젝트)의 결과에 대한 성능 평가를 위해 개발한 온톨로지 생성 도구이다.

그림 9는 평가에 이용한 온톨로지 일부분을 보여준다. 온톨로지 스키마 "univ-bench"와 "University0_0"에 해당하는 인스턴스를 임포트하였다. 실험에 이용된 온톨로지 규모는 43개의 개념과 1,657개의 인스턴스로 구성된다.

표 2는 실험을 위해 정의한 질의 예제를 보여준다. 총 5개의 질의를 정의하여 실험에 이용하였으며, 질의 패턴은 검색 대상(개념, 인스턴스)과 추론 기능의 포함 여부에 따라 분류된다.

6.2 평가 항목 및 실험 결과

실험을 통한 평가는 개념 뷰, 인스턴스 뷰 및 추론 뷰를 대상으로 실험을 실시한다. 기존 연구들은 일부 뷰에 대한 부분적인 기능만을 제공하지 않는다. 무엇보다 구체적인 시스템을 제공하고 있지 않기 때문에 특정 시스템을 대상으로 직접적인 실험이 불가능하다. 따라서 이 논문에서는 특정 비교 대상 시스템을 선정하지 않고 제안 모델과 뷰를 제공하지 않는 접근 방법으로 분류하여 실험을 실시한다. 이에 따라 실험은 세 가지 유형으로 분류되며, 실험 III은 추론 뷰를 통해 질의를 재작성하는 접근 방법과 각 추론 모델을 통해 접근하는 방법을 구분하여 실험한다. 즉, 실험 I, II에서와 같이 뷰의 제공 여부에 따라 실험 대상을 분류하고 추가적인 조건을 부

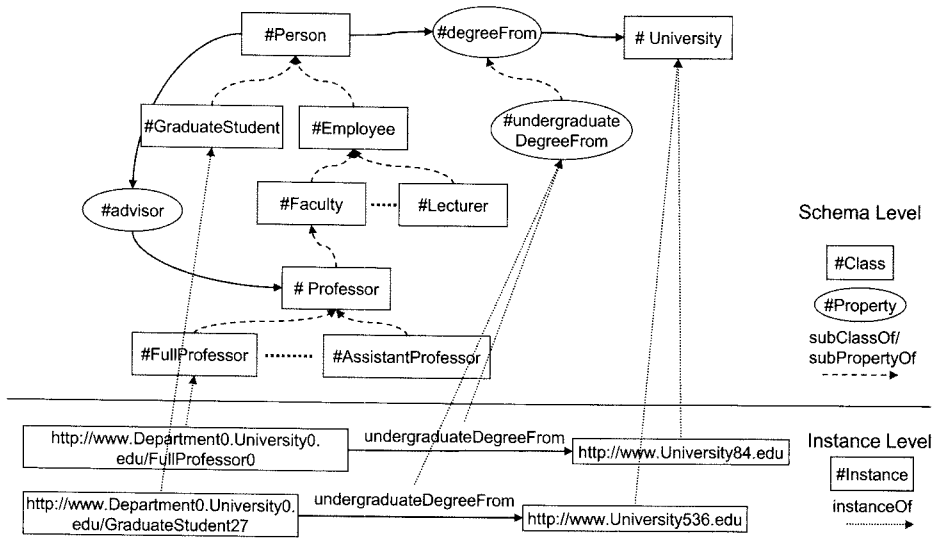


그림 9 실험에 이용한 온톨로지의 부분 예제

표 2 실험을 위해 이용된 질의 예제

#(패턴)	질의 예제
1(1)	Select ?v Where <?v, #degreeFrom, http://www.University84.edu>
2(1)	Select ?v Where <?v, rdfs:subClassOf, #Employee>
3(2)	Select ?v Where <#advisor, rdfs:range, ?v>
4(3)	Select ?v Where <http://www.DEpartment0.University0.edu/GraduateStudent27, ?v, http://www.University536.edu>
5(4)	Select ?v Where <#Professor, ?v1, ?v2>

여한 실험이 이루어짐을 의미한다. 각 실험에 대한 평가 내용은 다음과 같다.

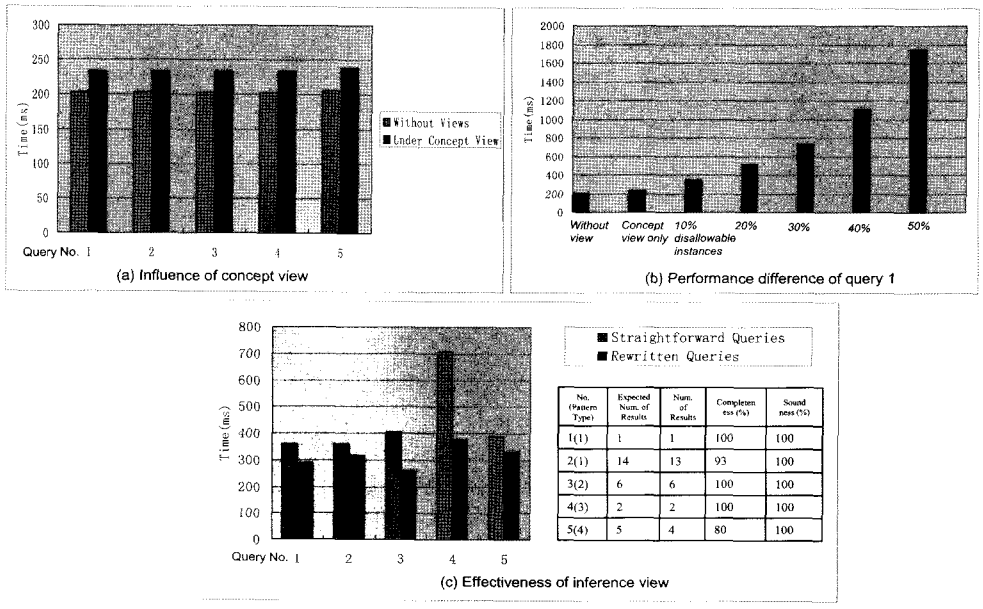
실험 I. 첫 번째 실험은 개념 뷰 강화를 위해 발생하는 지연 시간에 대해 평가한다. 실험은 개념 뷰를 지원하지 않는 접근 방법과 개념 뷰를 지원하는 접근 방법(제안 모델)에 대해서 정의된 질의를 실행함으로써 이루어진다. 평가를 위한 개념 뷰를 모든 개념에 대해 접근이 가능하도록 정의한다. 이를 통해 주어진 질의를 재작성함으로써 최장으로 확장되는 최악의 상황까지 평가할 수 있다.

실험 II. 이 실험은 인스턴스 뷰가 질의 응답에 미치는 영향을 평가하기 위한 실험이다. 이 실험은 인스턴스 크기에 따른 성능 및 반환되는 결과를 측정한다. 실험 I에서의 실험 대상 분류와 동일하게 인스턴스 뷰를 지원하는 경우와 지원하지 않는 경우에 대하여 실험을 실시하며, 추가적으로 개념 뷰와 인스턴스 뷰를 모두 제공하지 않는 경우에 대해서도 실험을 실시한다.

실험 III. 이 실험은 질의 재작성이 추론 가능한 결과에 미치는 영향에 대하여 평가하기 위한 실험이다. 실험은 추론 뷰를 통해 질의를 재작성하는 접근 방법과 백엔드 데이타를 추론 모델을 통해 직접 액세스 하는 접근 방법(추론 뷰를 제공하지 않는 접근 방법)을 대상으로 실험을 실시한다. 두 접근 방법에 따라 실행된 질의는 모든 개념에 대한 접근을 허용하는 동일한 개념 뷰와 모든 인스턴스에 대한 접근을 허용하는 동일한 인스턴스 뷰를 지니도록 한다.

그림 10은 앞서 정의한 실험 평가 항목에 대한 모든 실험 결과를 전체적으로 보여준다. 먼저 그림 10(a)는 첫 번째 실험(실험 I)에 대한 실험 결과이다. 그림에서 알 수 있듯이, 개념 뷰를 지원하지 않는 접근 방법의 경우, 각 질의에 대한 처리 비용이 거의 동일하다. 이는 SPARQL 질의 응답이 RDF 모델 트리플과 질의 트리플 간의 사상 연산을 통해 이루어지기 때문이다. 즉, 하나의 트리플의 사상을 위한 응답 시간이 동일하다. 개념 뷰에 의해 발생되는 지연 시간은 질의에 추가되는 절에 따라 달라지게 된다. 각 질의가 동일한 길이로 확장된다면 지연되는 시간은 항상 동일하다. 개념 뷰를 이용하는 접근 방법은 개념 뷰를 이용하지 않는 접근 방법과 비교하여 약 15%의 추가적인 시간을 요구한다. 따라서 단순히 개념에 대한 접근 제어만을 요구하는 어플리케이션의 경우, 처리 성능 측면에서 개념 뷰를 정의하지 않는 접근 방법이 보다 나은 적용 방법이라 할 수 있다.

그림 10(b)는 두 번째 실험에 대한 평가 결과를 보여준다. 그래프는 query1, 즉 표 2의 첫 번째 질의에 대한 질의 처리 성능을 보여준다. 실험 결과에서, 개념 뷰와 인스턴스 뷰를 모두 지원하지 않는 접근 방법이 가장 나은 성능 보였으며 개념 뷰와 인스턴스 뷰를 모두 지원하는 경우 가장 낮은 성능을 제공하였다. 개념 뷰와 인스턴스 뷰를 모두 제공하는 접근 방법(제안 모델)에서, 전체 인스턴스 대비, 접근 가능한 인스턴스의 비율이 높은 경우, 불가능한 인스턴스만을 검사함으로써 최상의 효율성을 얻을 수 있다. SPARQL을 이용하면 이



No. (Pattern Type)	Expected Num. of Results	Num. of Results	Completeness (%)	Soundness (%)
1(1)	1	1	100	100
2(1)	14	13	93	100
3(2)	6	6	100	100
4(3)	2	2	100	100
5(4)	5	4	80	100

그림 10 실험 평가 결과

러한 기능을 구현할 수 있다. 즉 접근이 불가능한 인스턴스 집합($\neg I_i$)은 모든 온톨로지 인스턴스 집합에서 V_I,I를 제거함으로써 얻을 수 있다. 질의 <?v, isSameAs, I_i>는 <?v, isNotSameAs, $\neg I_i$>로 변경된다.

그래프에서, 인스턴스 뷰에 접근이 불가능한 인스턴스의 수가 증가할수록 비용이 증가함을 알 수 있다. 최악의 경우는 접근 가능한 인스턴스와 불가능한 인스턴스의 비율이 5:5인 경우로서, 뷰를 제공하지 않는 접근 방법과 비교하여 약 9배 이상 질의 응답 시간을 요구한다. 따라서 제안 모델의 경우, 접근 가능한 혹은 접근이 불가능한 인스턴스의 비율이 20% 이하일 때 성능 측면에서 실용적인 접근 제어가 가능함을 알 수 있다.

마지막으로, 그림 10(c)는 추론 뷰를 지니는 경우와 추론 뷰를 제공하지 않는 접근 방법을 비교한 실험 결과를 보여준다. 실험 결과에서, 추론 뷰를 이용한 접근 방법이 성능 측면에서 보다 나은 결과를 보인다. 재작성된 질의는 직접적으로 질의를 추론 모델에 보내는 접근 방법보다 낮은 처리 시간을 요구한다.

앞선 실험들을 종합해 볼 때, 단순한 개념 혹은 인스턴스에 대한 접근 제어의 경우, 제안 모델이 낮은 성능을 보인 반면, 추론이 요구되는 질의에 대한 접근 제어를 위한 연산에서는 보다 나은 성능을 보였다. RDF는 추론 기능을 제공하며 이는 추론 질의에 대한 접근 제어 기능을 반드시 제공해야 함을 의미한다. 이러한 측면에서, 제안 모델이 보다 적합한 접근 방법이라 할 수 있다. 기존 시스템들은 개념 및 인스턴스에 대한 부분적인 접근 제어 기능만을 제공한다. 그러나 제안 모델은 모든 레벨에 대한 접근 제어를 가능하게 한다. 마지막으로, 제안 모델은 추론 규칙과 추론 모델 간의 독립성을 제공한다. 정확하게 추론 규칙에 대한 평가를 통해 유효한 액세스를 수행하는 질의를 재작성함으로써 관리의 편의성은 물론 기존 접근 방법보다 나은 처리 성능을 제공한다.

추가적으로, 이 논문에서는 제안된 모델에 대한 정확성(Soundness)과 완전성(Completeness)에 대하여 실험하였다. 그림 10(c) 내에 있는 표는 이에 대한 실험 결과를 보여준다.

완전성이란 예상되는 검색 결과와 실제 검색된 결과와의 비율을 의미한다. 만일 특정 질의의 실제 반환되어야 하는 결과 개수를 N_{all}이라고 하고 실제 검색된 결과 개수인 질의 실행 결과를 N_{query}라고 할 때, 완전성 계산 모델은 $N_{query}/N_{all} * 100$이 된다. 실험 결과에서 질의 2와 질의 5를 제외한 나머지 질의 모두 100%의 완전성을 보였다.

정확성이란 실제 검색된 결과가 올바른 결과인가를 평가하는 요소이다. 만일 검색된 결과의 개수를 N_{query}라 하고 검색 결과 중에서 올바른 결과 개수를 N_{sound}라고 할 때, 정확성 계산 모델은 $N_{sound}/N_{query}*100$이 된다. 실험 결과에서 제안 모델은 실험에 이용된 모든 질의에 대해서 100%의 정확성을 보였다.

6.3 관련 연구

온톨로지 뷰에 대한 개념은 오랫동안 사용되어 왔으며 기존 온톨로지 뷰는 사용자의 관심에 따라 가상의 온톨로지를 제공하기 위한 목적으로 사용되었다. Volz [8]는 경량의 웹 온톨로지를 위한 질의 기반 온톨로지 뷰에 대하여 연구하였고 Magkanaraki[9]는 시맨틱 웹 상에서, 유효한 자원을 서술하고 클래스에서 가상의 RDF/S 스키마와 속성을 생성할 수 있는 RDF 뷰 언어(RDF View Language, RVL)를 제안하였다. Noy와 Musen[10]은 온톨로지 순회 뷰(Traversal View) 개념을 개발하였다.

그러나 제안된 기존 온톨로지 뷰들은 질의 혹은 순회를 기반으로 하며 방문하고자 하는 온톨로지 일부분 혹은 가상의 스키마를 제공하는데 초점을 두었다. 비롯 일부 온톨로지 보안에 대한 문제를 다루었지만 접근 제어 문제를 핵심적으로 다루고 있지 않다. 이는 온톨로지 보안 측면에서 여러 가지 문제점을 지닌다. 검색이 단순히 백 엔드 온톨로지 모델의 일부분으로 제한되고 추론과의 연계성에 대해서는 고려하지 않고 있기 때문에 허용되지 않는 결과를 반환해 준다. 추가적으로, 백 엔드 온톨로지에서 추론 기능을 지원하지 않을 경우, 유도되는 온톨로지에 대한 검색 결과를 제공하지 않음으로써 RDF가 지니는 기능을 활용할 수 없다.

이러한 문제점을 해결하기 위해 RDF 접근 제어에 초점을 둔 많은 연구 결과들이 발표되었다. Qin과 Atluri [11]는 CAC (Concept-Level Access Control) 모델을 제안하였다. 이 모델은 온톨로지에 정의된 개념들을 통해 권한을 명세화하고 개념들간의 관계성에 기반한 전파 (Propagation) 기능으로 데이타 인스턴스를 강화한다. 그러나 이 모델은 개념 레벨에서 접근 제어에 대해서 미비하고 다루고 있을 뿐 개념과 인스턴스 레벨 모두에 대해서는 다루지 않고 있다.

Reddivari와 Finin [12]은 RDF 저장소 관리를 위한 프레임크워에 기반한 보안 정책을 제안하였다. 이들은 RDF 문서에 대한 관리 권한을 통해 정책 기반 제어를 기술하는 방법을 제안하였다. 제안한 정책 정의 방법은 아직 초기 단계로서 구현을 통해 평가되지 않은 상태이다. 온톨로지 추론 제어에 대해서도 다루고 있으나 추론 제어가 온톨로지 질의가 아닌 권한 전파를 위해서만 이용되는 한계성을 지닌다.

이 외에도 Kaushik[13]은 온톨로지 제어를 위해 로직 기반의 정책 언어를 제안하였다. 제안된 방법을 RDF/S

데이타베이스에 적용할 수 있지만 RDF/S 함의를 고려하지 않고 있다. Jain과 Farkas[14]는 RDF 패턴과 보안 분류를 정의하였다. 보안은 RDF 문장을 RDF 패턴에 사상시킴으로써 결정된다. 그러나 RDF 패턴은 RDF 문장이 방대할 경우, 모든 패턴을 사상시켜야 하는 문제점을 초래한다. 추가적으로, 하나의 RDF 문장이 여러 가지 패턴을 지닐 수 있고 또한 문장의 조합으로 다른 문장을 생성할 수 있기 때문에 보안 분류 할당 관리가 어렵다는 문제점을 지닌다.

7. 결 론

이 논문에서는 허용되지 않은 접근으로부터 RDF 온톨로지를 보호하기 위한 접근 제어 모델을 제안하였다. 제안된 모델은 사용자 질의를 재작성하기 위해 3 단계 온톨로지 뷰 집합을 정의하였다. 개념과 인스턴스 뷰는 사용자가 접근할 수 있는 개념과 인스턴스들을 정의하며 추론 뷰는 사용자가 특정 범위 내의 추론 결과에 대해서만 허용하기 위해 정의된다. 이 논문에서는 각 뷰의 정의 방법에 대하여 정의하고 질의에 대한 뷰 강화 알고리즘에 대하여 기술하였다. 또한 실험을 위한 질의를 정의하고 실제 온톨로지를 이용한 실험 결과를 보였다.

실험 및 평가에서, 제안된 접근 방법이 기존 접근 방법에 비해 RDF의 특징을 최대한 반영하고 있으며 보다 나은 기능성과 관리의 효율성을 제공함을 알 수 있다. 단순한 개념이나 인스턴스 검색 실험에서, 제안 모델이 뷰를 지원하지 않는 접근 방법에 비해 낮은 성능을 보였으나 추론이 요구되는 질의에 대해서는 나은 성능을 보였다. 또한 추론 규칙에 따라 추론 모델을 재생성해야 하는 기존 접근 방법과는 달리 질의 재작성을 통해 접근 제어를 수행함으로써 관리의 용이성을 제공한다.

제안한 모델이 기존 모델에 비해 여러 장점을 제공하지만 다음과 같은 개선 사항을 지닌다. 우선 이 논문에서는 Leigh 대학에서 제공하는 온톨로지 생성 도구를 통해 생성한 온톨로지를 실험에 이용하였다. 그러나 향후 실제 응용 분야에 적용함으로써 제안된 모델에 대한 검증 및 확인 작업이 요구된다. 또한 OWL(OWL Web Ontology)[23] 웹 온톨로지에 대한 접근을 제어할 수 있는 확장된 모델 개발이 요구된다.

참 고 문 헌

[1] Beckett, D., McBride, B., RDF/XML Syntax Specification, W3C Recommendation, 10 February 2004.

[2] Hada, S. and Kudo, M., XML access control language: Provisional authorization for XML documents, http://www.trl.ibm.com/projects/xml/xacl/xacl-spec.html.

[3] Oasis, eXtensible Access Control Markup Language (XACML), http://www.oasis-open.org/committees/xacml.

[4] Bertino, E., Castano, S., Ferrari, E., "On Specifying Security Policies for Web Documents with an XML based Language," ACM SACMAT'01, 2001.

[5] Bertino, E., Castano, S., Ferrari, E., "Securing XML documents with Author-X," IEEE Internet Computing, Vol.5, No.3, pp. 21-31, May/June 2001.

[6] Fan, W., Chan, C.-Y., Garofalakis, M., "Secure XML Querying with Security Views," SIGMOD 2004, Paris, France, June 2004.

[7] Geuer-Pollmann, C., "XML Pool Encryption," ACM Workshop on XML Security, Fairfax VA, USA, November 2002.

[8] Volz, R., Oberle, D., and Studer, R., "Implementing Views for light-weight Web Ontologies," IDEAS'03, pp. 160-170, Hong Kong, July 16-18, 2003.

[9] Magkanaraki, A., Tannen, V., Christophides, V., and Plexousakis, D., "Viewing the semantic web through RVL lenses," Springer-Verlag, ISWC'03, Vol. LNCS 2870, pp. 96-112, 2003.

[10] Noy, N. F. and Musen, M. A., "Specifying ontology views by traversal," ISWC'04, Vol. 3298, pp. 713-725, November 2004.

[11] Qin, L., Atluri, V., "Concept-level access control for the Semantic Web," XMLSEC'03, pp. 94-103, Fairfax, Virginia, USA, October 2003.

[12] Reddivari, P., Finin, T., and Joshi, A., "Policy based access control for a rdf store," WWW2005, pp. 78-83, Chiba, Japan, May 2005.

[13] Kaushik, S., Wijesekera, D., and Ammann., P., "Policy-based dissemination of partial web-ontologies," ACM Press, SWS'05, pp. 43-52, New York, NY, USA, 2005.

[14] Jain, A. and Farkas, C., "Secure Resource Description Framework: an Access Control Model," SACMAT'06, Lake Taoe, California, USA, June 7-9, 2006.

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

[16] Broekstra, J., Kampman, A., and van Harmelen, F., "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," Springer Verlag, ISWC 2002, Vol. LNCS 2342, pp. 54-68, 2002.

[17] Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis D., and Scholl, M., "RQL: A Declarative Query Language for RDF," WWW02, Honolulu, Hawaii, USA, May 2002.

[18] Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., Tolle, K., Amann, B., Fundulaki, I., Scholl, M., and Vercoustre, A.-M., "Managing RDF metadata for community webs," WCM'00, Salt Lake City, Utah, USA, pp. 140-151, October 2000.

[19] W3C, W3C Candidate Recommendation, SPARQL Query Language for RDF, April 2006, http://www.w3.org/TR/rdf-sparql-query/
[20] Horrocks, I. and Tessaris, S., "A conjunctive query language for description logic aboxes," AAAI-00, 2000, Austin, Texas, USA, July 30-August 3, 2000.
[21] Hewlett-Packard Development Company, ARQ - A SPARQL Processor for Jena, http://jena.sourceforge.net/ARQ/.
[22] Guo, Y., Pan, Z., and Heflin. J., "An Evaluation of Knowledge Base Systems for Large OWL Datasets," Springer Verlag, Third International Semantic Web Conference, Vol. LNCS 3298, pp. 274-288, 2004.
[23] Patel-Schneider, P.F., Hayes, P., and Horrocks, I., "OWL Web Ontology Language Semantics and Abstract Syntax," W3C Recommendation, February 2004.

정 동 원

1997년 군산대학교 컴퓨터과학과 이학사 1999년 충북대학교 전산과 이학석사 2004년 고려대학교 컴퓨터학과 이학박사 1998년 전자통신연구원 위촉연구원. 1999년~2000년 ICU 부설 한국정보통신교육원 GIS 분원 전임강사. 2000년~2001년 지구넷 부설 연구소 선임연구원. 2002년~2005년 라임디미어 테크놀로지 연구원. 2004년~2005년 고려대학교 정보통신기술연구소 연구조교수. 2005년 Pennsylvania Stat University PostDoc. 2002년~현재 TTA 표준화위원회-메타데이타표준화 프로젝트 그룹 (PG406) 특별위원. 2005년~현재 군산대학교 정보통계학과 교수. 2006년~현재 ISO/IEC JTC1/SC32 전문위원회 위원. 관심분야는 데이타 공학, 시맨틱 웹, 시맨틱 GIS, 유비쿼터스 컴퓨팅, 정보보안

장 이 신

2001년 Wuhan University 이학사. 2004년 Wuhan University 이학석사. 2005년~현재 고려대학교 컴퓨터학과 박사과정. 관심분야는 데이타 공학, 시맨틱 웹, 정보보안, 유비쿼터스 컴퓨팅

백 두 권

1974년 고려대학교 수학과 졸업. 1977년 고려대학교 대학원 산업공학석사. 1983년 Wayne State University 전산학석사 1986년 Wayne State University 전산학박사. 1986년~현재 고려대학교 정보통신대학 교수. 1989년~현재 한국정보과학회 이사/평의원/부회장. 1991년~현재 한국시뮬레이션학회 이사/감사/부회장/회장/고문. 1991년~현재 ISO/IEC JTC1/SC32 전문위원회 위원장. 1997년~1998년 고려대학교 정보전산원 원장. 2002년~2004년 고려대학교 정보통신대학 초대학장. 2003년~2004년 한국정보처리학회 부회장. 관심분야는 데이타 공학, 소프트웨어 공학, 모델링과 시뮬레이션