포함관계 추론에서 접근 권한에 대한 효율적 RDF 질의 유효성 검증

(An Efficient RDF Query Validation for Access Authorization in Subsumption Inference)

김 재 훈 † 박 석 ††
(Jaehoon Kim) (Seog Park)

요 약 시맨틱 웹을 위한 하나의 보안연구로, 본 논문에서는, 온톨로지 계층 구조와 RDF 트리플 패턴에 기반한 RDF 접근 권한 명세 모델을 소개한다. 또한 권한 명세 모델을 승인된 접근 권한들에 대한 RDF 질의 유효성 검증 과정에 적용한다. RDF 트리플 패턴을 가지는 대표적 RDF 질의 언어인 SPARQL 또는 RQL 질의는 RDF 트리플 패턴 형식으로 명세된 접근 권한에 따라 실행 거부되거나 인가될 수 있다. 이러한 질의 유효성 검증 과정을 효율적으로 수행하기 위하여 RDF 포함 관계 추론에서의 주요한 권한 충돌 조건들을 분석한다. 다음으로 분석된 충돌조건과 Dewey 그래프 레이블링 기술을 활용하는 효율적 질의 유효성 검증 알고리즘을 제시한다. 실험을 통하여 제시된 검증 알고리즘이 합리적인 유효성 검증 시간과, 데이터와 접근권한들이 증가할 때 확장성을 가짐을 보인다.

키워드 : RDF 질의 유효성, 접근제어, 권한충돌, 데이터베이스 보안, 시맨틱 웹

Abstract As an effort to secure Semantic Web, in this paper, we introduce an RDF access authorization model based on an ontology hierarchy and an RDF triple pattern. In addition, we apply the authorization model to RDF query validation for approved access authorizations. A subscribed SPARQL or RQL query, which has RDF triple patterns, can be denied or granted according to the corresponding access authorizations which have an RDF triple pattern. In order to efficiently perform the query validation process, we first analyze some primary authorization conflict conditions under RDF subsumption inference, and then we introduce an efficient query validation algorithm using the conflict conditions and Dewey graph labeling technique. Through experiments, we also show that the proposed validation algorithm provides a reasonable validation time and when data and authorizations increase it has scalability.

Key words : RDF query validation, access control, authorization conflict, database security, Semantic Web

1. 서 론

시맨틱 웹(Semantic Web)[1]과 관련한 많은 연구와 표준화 노력들은 앞으로의 웹의 발전 방향이 지능적인 웹 서비스에 맞추어져 있음을 보여준다. 이와 관련하여 본 연구진은 시맨틱 웹을 위한 접근 제어(access control) 보안 연구를 수행하고 있다. 이러한 노력의 일환으로 RDF 트리플(triple) 기반의 RDF 접근 제어 모델을 제시하였으며[2], RDF 접근 제어에서의 권한 충돌(authorization conflict) 조건 및 효율적 권한 충돌 발견 알고리즘을 소개하였다[3]. 본 논문에서는 기 분석된 RDF 권한 충돌조건이 본 논문에서 다루고자 하는 RDF 질의 유효성 검증에 또한 효율적으로 활용될 수 있음을

· 본 연구는 한국연구재단을 통해 교육과학기술부의 세계수준의연구중심대학육성사업(WCU)으로부터 지원받아 수행되었습니다(R33-2008-000-10110-0).

† 정 회 원 : 서일대학 정보통신과 교수
jhkimygk@seoil.ac.kr
†† 종신회원 : 서강대학교 컴퓨터공학과 교수
spark@dblab.sogang.ac.kr

논문접수 : 2009년 9월 9일
심사완료 : 2009년 9월 29일

보인다. 질의 유효성 검증(query validation)은 그림 3(b)에서와 같이 사용자로부터 입력된 RDF 질의가 해당 사용자의 접근권한에 따라 거부되거나 인가되는 과정을 말한다. 하지만 이전 연구[2,3]에서는 보안 관리자(security administrator)가 해당 사용자에 대한 새로운 권한을 부여할 경우 기존의 명세된 권한들과의 충돌여부를 조사하는 접근권한 명세(authorization specification) 과정(그림 3(a))에서의 권한 충돌 조건을 분석하였다. 새로이 명세된 권한이 기존 권한과 충돌되면 거부되고 그렇지 않으면 권한 저장소에 저장된다. 본 논문에서는 이전에 분석된 충돌조건과 충돌발견 알고리즘이 신속한 질의 유효성 검증을 위하여 일부 변형되어 활용될 수 있음을 보인다. 또한 유효성 검증시 RDF 질의의 추론에 의한 질의 처리 특성이 고려되어야 함을 설명한다.

RDF 질의 유효성 검증의 하나의 예는 다음과 같다. 그림 1에서 *Music*에 대한 접근이 사용자 *Dave*에게 허용되지 않았음을 가정하자. 그렇다면 아래의 *Pop*을 읽고자 하는 RDF 질의는 인가될 수 있는가? 답은 아니오이다. 왜냐하면 인가된 질의에 의해 *Pop*의 인스턴스들이 *Dave*에게 노출되면, *Pop*의 인스턴스들은 *subClassOf* 추론에 의해 *Music*의 인스턴스들로 해석될 수 있기 때문에, 해당 질의는 거절되어야 한다. 즉 RDF 질의 유효성 검증 과정에서는 RDF 추론에 의한 입력된 질의와 해당 권한 사이에서의 충돌여부 판정이 이루어져야 한다. 아래의 질의는 Music.rdf 문서로부터 팝송의 다운로드 주소를 얻고자 하는 질의이다.

```
prefix ex: <http://example.org/schemas/contents/>
select ?uri
from <http://example.org/data/Music.rdf#>
where { ex:Pop ex:downloadFrom ?uri . }
```

본 논문의 구성은 다음과 같다. 2장에서는 RDF 접근제어와 관련한 몇가지 주요한 연구를 소개한다. 3장에서는 본 연구와 관련한 RDF 질의 언어의 기본적 특성을 설명한다. 4장에서는 본 연구에서 이전에 제안한 RDF 권한 명세 모델을 간략히 소개한다. 다음으로 5장에서는 질의 유효성 검증 과정에서의 권한 충돌 문제를 소개하며, 6장에서는 그래프 레이블링 기술을 이용한 효율적 질의 유효성 검증 알고리즘을 제시한다. 7장에서는 제안된 알고리즘의 성능실험 결과를 소개하며, 8장에서는 본 연구의 결론을 언급한다.

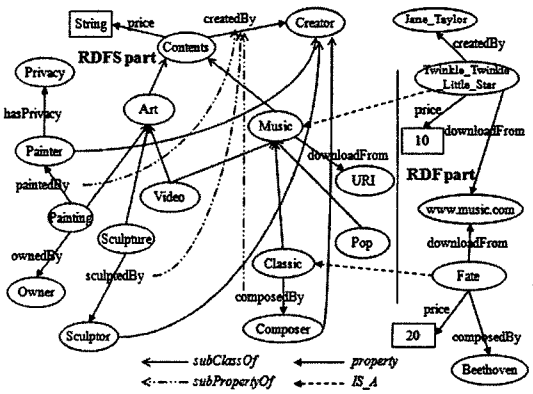


그림 1 RDF 그래프

2. 관련 연구

Jain과 Farkas의 연구[4]는 본 연구의 동기가 되었다. Jain과 Farkas 또한 RDF 트리플에 기반한 접근 제어 모델을 제시하였으며, RDF 데이터의 접근제어에서의 추론에 의한 권한 충돌 문제를 소개하였다. 하지만 그들이 제안한 접근 권한 명세시의 충돌 발견 알고리즘은 본 연구의 이전 논문[3]에서 지적한 바와 같이 매우 단순 소모적인(brute force) 알고리즘을 사용한다. 제안된 알고리즘은 먼저 추론에 의해 모든 RDF 트리플에 대하여 권한 전파(authorization propagation)를 수행한다. 다음으로 모든 RDF 트리플에 대하여 권한 충돌이 있었는지를 조사하고, 만약 있었다면 그 추론은 취소된다. 이러한 방법은 RDF 데이터가 작은 규모가 아닐 때 비효율적이다. 본 논문에서는 RDF 질의 유효성 검증에서의 충돌 조건과 그래프 레이블링 방법을 활용하여 보다 효율적인 충돌 발견 알고리즘을 제시한다.

Qin과 Atluri의 연구[5]는 웹 온톨로지 데이터(RDF와 OWL)의 접근제어에서의 추론에 의한 권한 충돌 문제를 분석한 처음의 연구로 사료된다. 그들은 RDF 데이터의 포함 관계에 관련된 subClassOf, subPropertyOf 외에 두 개념 사이의 동등 관계(equivalence relationship), 전체 개념과 부분 개념 사이의 관계(예로, intersection, union), 비 추론적 관계(non-inferable relationship)등 다양한 의미적 관계에 대한 권한 전파를 고려하였다. 하지만 Jain과 Farkas와는 달리 보안 객체로서 XPath와 유사한 RDF path를 사용하였다. 이러한 보안 객체의 정의는 바람직하지 않다. 왜냐하면 OWL과 RDF 모델은 기본적으로 RDF 트리플에 기반하기 때문이다. 따라서 RDF 트리플에 기반을 둔 Jain과 Farkas의 모델이 더 바람직하다고 할 수 있다.

정동원외 2인의 연구[6]에서는 RDF 온톨로지 뷰에 대한 개념을 소개하였다. 사전에 보안 관리자는 데이터베이스에서의 뷰(view) 개념과 같이 각 사용자에 대해 RDF 데이터에 대한 뷰를 정의한다. 따라서 사용자는 자신에게 허용된 뷰에 대해서만 RDF 질의를 수행할 수 있다. 사용자로부터 RDF 질의가 주어질 경우 허용된 뷰로부터의 질의 재작성을 통하여 질의 결과를 제시한

다. 그들은 RDF 인스턴스 수준까지의 뷰 정의 및 RDF 추론에 의한 질의 재작성의 문제들을 소개하였다.

최근 Finin외 6인의 연구[7]에서는 NIST의 RBAC (Role Based Access Control) 모델을 OWL로 기술하는 방법에 대하여 소개하였다. 그들은 비록 OWL이 접근 권한 명세를 위하여 개발된 언어가 아니지만, RBAC 모델을 기술하는 언어로 성공적으로 활용될 수 있음을 보여 주었다.

RDF 데이터는 XML로 기술되기 때문에 XML 접근 제어 메커니즘을 살펴 볼 필요가 있다. 우선 Lee et al. [8] 연구에서는 본 연구와 유사하게 신속한 접근 제어 수행을 위하여, XML 데이터에 대해 이차원 인덱스[9]를 사용하여 접근권한 인덱스를 구성한다. 또한 최 근접 이웃 탐색 기술(nearest neighbor search technique)을 이용하여 명시된 접근권한을 효율적으로 찾을 수 있게 한다. 하지만 본 연구에서는 RDF 데이터에 대한 그래프 상에서의 계승에 의한 명시적 접근권한 전파와 추론에 의한 암시적 접근권한 전파에 따른 질의 유효성 검증을 연구한다. 또한 최근까지의 XML 접근제어에 관한 다양한 연구는 참고문헌[10,11]를 참조할 수 있다. 상기 문헌들에서는 뷰-기반, 레이블링-기반, 오토마타-기반의 XML 접근제어 메커니즘을 관련 연구에서 소개하고 있으며, 하지만 앞서 언급한 대로 이러한 연구들이 RDF에서의 추론을 고려한 접근제어 메커니즘을 고려하지 않음을 알 수 있다.

3. RDF 질의 언어

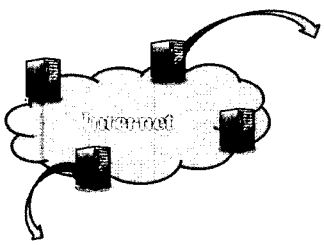
3.1 RDF 트리플

RDF 데이터는 RDF Schema(RDFS)에 정의되어진 온톨로지 개념을 사용하는 RDF 문장(statement)들로 이루어진다. RDFS[12]는 기본적인 온톨로지 어휘 *subClassOf*와 *subPropertyOf*에 의해 클래스와 속성의 계층구조를 정의할 수 있다. 예로, 그림 2에서의 RDFS 문서는 웹 콘텐츠에 관한 클래스와 속성 계층구조를 보여준다. RDF와 RDFS 문장은 [*S*, *P*, *O*]의 트리플 형태로 표현되며, RDF 데이터는 RDF 트리플로 이루어진 그래프 형태로 표현될 수 있다.

정의 1 (RDF 그래프, RDF 트리플).

RDF 그래프는 *RDF* 트리플들의 집합이다. *RDF* 트리플은 [*S*, *P*, *O*]로 표현되며, $S \in SUBJECT$, $P \in PREDICATE$, $O \in OBJECT$ 이다.

- 집합 *SUBJECT*는 *RDFS*에서의 클래스와 속성을 정의하는 *URI* (*Uniform Resource Identifier*)와 *RDF*에서의 인스턴스들을 정의하는 *URI*, 그리고 공백 (*blank*) 노드들을 포함한다.
- 집합 *PREDICATE*는 *RDFS*에서의 속성을 참조하는 *URI* 노드들을 포함한다.
- 집합 *OBJECT*는 *P*에 의해서 연관되어지는 상대방 클래스 혹은 인스턴스의 *URI* 노드와, 공백 노드, 리터럴 (*literal*)들을 포함한다.



```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex="http://example.org/data/contents#">
<ex:Classic rdf:ID="Fate">
  <ex:composedBy rdf:resource="ex:Beethoven"/>
  <ex:price rdf:datatype="&xsd:string">20</ex:price>
  <ex:downloadFrom rdf:resource="http://www.music.com"/>
</ex:Pop>
<ex:Music rdf:ID="Twinkle_Twinkle_Little_Star">
  <ex:createdBy rdf:resource="ex:Jane_Taylor"/>
  <ex:price rdf:datatype="&xsd:string">10</ex:price>
  <ex:downloadFrom rdf:resource="http://www.music.com"/>
</ex:Music>
</rdf:RDF>
```

RDF 문서

```
<rdf:Description rdf:ID = "Contents">
        <rdf:type rdf:resource = "http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:ID = "createdBy">
        <rdf:type rdf:resource = "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
        <rdfs:domain rdf:resource = "ex:Contents"/>
        <rdfs:range rdf:resource = "ex:Creator"/>
</rdf:Description>
<rdf:Description rdf:ID = "Music">
        <rdf:type rdf:resource = "http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource = "ex:Contents"/>
</rdf:Description>
<rdf:Description rdf:ID = "composedBy">
        <rdf:type rdf:resource = "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
        <rdfs:domain rdf:resource = "ex:Classic"/>
        <rdfs:range rdf:resource = "ex:Composer"/>
                <rdfs:subPropertyOf rdf:resource = "ex:createdBy"/>
</rdf:Description>
<rdf:Description rdf:ID = "Classic">
        <rdf:type rdf:resource = " http://www.w3.org/2000/01/rdf-schema#Class "/>
         <rdfs:subClassOf rdf:resource = "ex:Music"/>
</rdf:Description>
```

RDFS 문서

그림 2 웹 콘텐츠에 관한 샘플 RDF와 RDFS 문서

예로, 그림 1의 그래프는 그림 2의 RDF와 RDFS를 표현한다. RDF 트리플 [*Classic*, *composedBy*, *Composer*]는 *S*로서 클래스 URI 상수 *Classic*을 가지며, *P*로서 속성 URI 상수 *composedBy*, *O*로서 클래스 URI 상수 *Composer*를 갖는다. [*Fate*, *Price*, 20]은 *S*로서 인스턴스 URI 상수 *Fate*와 *O*로서 리터럴 20을 갖는다.

정의 2 (*subClassOf* 관계, *subPropertyOf* 관계). 만약 어떤 클래스 c_i가 다른 클래스 c_j의 하위 클래스이면 ($c_i \subset c_j$), c_i와 그것의 인스턴스들은 c_j의 속성을 계승하며, 추론에 의해 c_i는 c_j로 해석될 수 있다. 만약 어떤 속성 p_i가 다른 속성 p_j의 하위 속성이라면 ($p_i \subset p_j$), p_i는 추론에 의해 p_j로 해석될 수 있다.

예로, 그림 2에서, <*ex:Classic rdf:ID = "Fate"*>는 *subClassOf* 추론에 의해 <*ex:Music rdf:ID = "Fate"*>로 해석될 수 있으며, <*ex:composedBy rdf:resource = "ex:Beethoven"*>은 *subPropertyOf* 추론에 의해 <*ex:createdBy rdf:resource = "ex:Beethoven"*>으로 해석될 수 있다.

3.2 SPARQL과 RQL

대부분의 RDF 질의 언어들은 공통적으로 RDF 트리플에 기반한 질의 형식을 지원한다[13-15]. 먼저 대표적인 RDF 질의 언어로서, W3C에서 승인된 SPARQL 질의 언어[14]를 고려하자. SPARQL로 RDF 데이터를 질의하는 것은 RDF 그래프에 대하여 패턴 매칭을 수행하는 것과 같다. 패턴들은 RDF 트리플로 표현되며, 조인(Join) 연산에 의해서 다른 패턴과 상호 연결될 수 있다. 조인질의 형식은 패턴들 사이에서 공유되는 동일 변수의 선언으로 표현된다. 예로, 질의 *Q*1에서, WHERE 절은 두 가지 트리플 패턴[*?music*, downloadFrom, *?uri*]와 [*?music*, createdBy, *?creator*]를 언급한다. 그리고 두 패턴은 변수 *?music*에 의해서 조인되어진다. SPARQL에서의 변수이름은 $ 혹은 ?로 시작한다. FROM 절은 질의 대상인 RDF 데이터가 위치한 URI 주소를 명시하며, SELECT 절은 최종 질의 결과로서 반환될 변수들을 명시한다. 즉, 질의 *Q*1은 어떤 *creator*에 의해서 만들어진 음악 파일이 어떤 웹 주소로부터 다운로드될 수 있는 경우에 한해서 음악 파일 이름, *creator* 이름, URI 주소의 질의 결과를 반환한다.

Q1: prefix ex: <http://example.org/schemas/contents/>
select *?music ?creator ?uri* from <http://example.org/data/Music.rdf#>
where { *?music* ex:createdBy *?creator* .
?music ex:downloadFrom *?uri* . }

다른 샘플 질의를 살펴보자. Q2는 경로 탐색 질의 유형을 보여준다. 즉, 변수 *?painter*에 의해서 *?painting* $\xrightarrow{paintedBy}$ *?painter* $\xrightarrow{hasPrivacy}$ *?privacy*의 경로 탐색이 이루어진다. Q2는 어떤 그림을 그린 화가의 개인적인 정보를 반환한다.

Q2: prefix ex: <http://example.org/schemas/contents/>
select *?painter ?privacy*
from <http://example.org/data/Art.rdf#>
where { *?painting* ex:paintedBy *?painter* .
?painter ex:hasPrivacy *?privacy* . }

정의 3 (Q-pattern, pt_i(Q)). Q-pattern은 RDF 질의에서의 RDF 패턴을 가리키며, RDF 트리플 [*S*, *P*, *O*]의 형태를 가진다. *S*와 *P*는 임의의 변수로 표현될 수 있으며, 본 연구에서 *O*는 항상 변수 형태로만 존재함을 가정한다(*O*가 인스턴스 URI이거나 리터럴인 경우 실제 데이터 값에 따른 다소 복잡한 접근제어 메커니즘을 요구한다. 본 연구에서는 이를 배제한다). 어떤 주어진 RDF 질의 Q에 대하여, Q가 가지는 일련의 Q-pattern들을 pt_i(Q), i = 1, 2, 3, ..., n으로 표시한다.

주의할 점은 SPARQL은 RDF 추론이 반영된 질의 처리를 지원하지 않는다. 단지 패턴들의 직접적인 매칭에 의한 질의 처리를 지원한다. 예로 Q1의 질의 결과는 Q-pattern [*?music*, creadedBy, *?creator*]와 직접적으로 매치되는 클래스 *Music*에 대하여 단지 계산된다. Q1은 *subClsassOf* 추론과 *subPropertyOf* 추론에 의한 클래스 *Classic*의 매칭 트리풀들은 반환하지 않는다. SPARQL의 또다른 제약사항은 스키마 질의 처리를 지원하지 않는다. 예로, 그림 2의 RDFS 스키마에서 SPARQL은, "클래스 *Classic*에 대해 모든 속성을 검색하기 위한", 혹은 "속성 *hasPrivacy*의 도메인 클래스를 검색" 하기 위한 질의 구문을 제공하지 않는다. 즉, SPARQL은 스키마를 제외한 RDF 데이터에 대해서만 질의 처리를 수행할 수 있다.

또다른 대표적 RDF 질의 언어로서 RDF 트리플과 Q-pattern에 기반한 RQL 질의 언어[15]를 고려할 수 있다. RQL은 SPARQL과 달리 추론에 의한 질의 답변과 스키마에 대한 질의 처리를 지원한다. SPARQL 질의 Q1, Q2를 표현하는 다음 RQL 질의 Q3, Q4를 살펴보자.

Q3: select *?music ?creator ?uri*
from {*?music*}ex:createdBy{*?creator*},
{*?music*}ex:downloadFrom{*?uri*}
using namespace ex =
http://example.org/schemas/contents/

Q4: select *?painter ?privacy*
from {*?painting*}ex:paintedBy{*?painter*}.ex:hasPrivacy{*?privacy*}
using namespace ex = http://example.org/

schemas/contents/

RQL은 질의 결과를 반환할 때 모든 추론된 질의 결과도 포함하여 반환한다. 예로 Q3의 질의 결과는 *Classic* 에 대한 추론 결과를 포함한 {[*Twinkle_Twinkle_ Little_ Star, Jane_Taylor, www.music.com*], [*Fate, Beethoven, www.music.com*]}이다. 만약 RQL에서 어떤 추론된 답변을 원하지 않는다면, RQL 질의에서의 클래스와 속성 이름 앞에 기호 "^"를 붙이면 된다. 또한 RDFS 스키마에 대해서만 질의 결과를 얻고 싶을 때에는, 클래스와 속성 이름 앞에 기호 "$"를 붙이면 된다.

4. 접근 권한 명세

4.1 A-pattern

기 제안된 접근권한 모델[2,3]에서, 보안 객체들은 RDF 트리플이다. 보안 관리자는 보안 객체를 다음 A-pattern으로 간편하게 표현할 수 있다. A-pattern은 Q-pattern과 유사하다.

정의 4 (A-pattern, *pt*(A)).

A-pattern은 정의 5의 접근 권한에서의 *obj*를 가리키며, RDF 트리플 [*S, P, O*]의 형태로 표현된다. *S*와 *P*는 임의의 변수로 표현될 수 있으며, 본 연구에서 *O*는 항상 변수임을 전제로 한다(*O*가 인스턴스 URI이거나 리터럴인 경우 실제 데이터 값에 따른 다소 복잡한 접근제어 메커니즘을 요구한다. 본 연구에서는 이를 배제한다). 그리고 *S*에 공백 노드를 명기할 수 없다. 어떤 접근 권한 A에 대하여, A의 A-pattern을 *pt*(A)로 표시한다.

예로, 그림 1의 RDF 그래프에서, RDF 패턴 *pt*(A) = [*$x, composedBy, $z*]는 {[*Classic, composedBy, Composer*], [*Fate, composedBy, Beethoven*]}의 매칭 트리플들을 가진다. [*Classic, $y, $z*]의 매칭 트리플은 {[*Clasic, price, literal*], [*Classic, createdBy, Creator*], [*Classic, composedBy, Composer*], [*Classic, downloadFrom, URI*]}이다. A-pattern [*$x, $y, $z*]은 그래프의 모든 에지를 매치한다.

4.2 접근 권한

RDF 접근 권한은 다음과 같은 형식으로 기술된다.

정의 5 (접근 권한).

접근 권한은 <*subj, obj, act, sign, type*>의 다섯 개 속성으로 표현된다.

- *subj*는 접근 권한이 인가될 사용자를 가리킨다.
- *obj*는 A-pattern이다.
- *act*는 *obj*에 대하여 수행될 연산을 가리킨다. 본 논문에서는 SPARQL과 RQL 질의 언어와 관련하여 읽기 (read) 연산만을 고려한다. SPARQL과 RQL은 쓰기 (write) 연산을 지원하지 않는다.
- *sign*은 접근이 허용되면, (+), 접근이 허용되지 않을 때는, (-) 값을 갖는다.
- *type*은 *subClsassOf* 와 *subPropertyOf* 관계에 의하여 하위 클래스와 하위 속성으로 접근 권한이 전파될 경우 R, 접근 권한이 전파되지 않을 경우 L 값을 갖는다. 5.1절에서 보다 자세한 사항을 살펴볼 것이다.

본 논문에서는 승인된 접근 권한들에 대하여 입력된 RDF 질의를 허용하거나 거절하는 질의 유효성 검증에 대하여 살펴볼 것이다. 그림 3의 접근 제어 시스템에서 <b>의 처리과정은 질의 유효성 검증을 보여 준다. 입력된 질의가 질의 유효성 검증기(query validator)에 의하여 접근 권한을 위반하지 않는 것으로 판단될 경우, 질의는 RDF 질의 처리기(query processor)에게 넘겨진다.

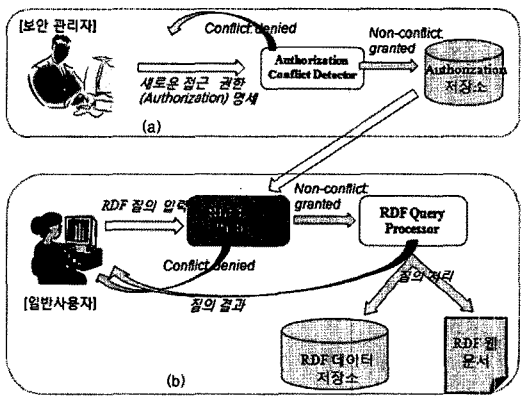


그림 3 RDF 접근 제어 시스템

5. RDF 질의 유효성 검증

5.1 명시적 접근 권한 전파에 따른 유효성 검증

접근권한의 *type* 값이 R일 때 접근 권한은 계승(inheritance)에 의해서 하위클래스 혹은 하위 속성에 영향을 준다. 따라서 A-pattern들에 의해서 입력된 RDF 질의의 Q-pattern들이 영향을 받을 때, 그 질의는 거부될 수 있다. 본 절에서는 *type* = R일 때의 질의 유효성 검증을 설명한다.

정의 2에서 설명한 것처럼, 만약 $c_i \subset c_j$ 이면, c_i는 c_j의 속성들을 계승한다. 따라서 c_j의 p_k에 대하여 명세된 어떤 접근 권한 A_j는 c_i의 속성 p_k에도 영향을 준다. 이러한 접근 권한 전파를 *명시적 권한 전파*라고 한다.

예 1. A_j = <*Dave*, [*Music, downloadFrom, $z*], read, -, R>가 명세될 때, A_j는 명시적 권한 전파 정책에 의하여 {<*Dave*, [*Pop, downloadFrom, $z*], read, -, R>, <*Dave*, [*Classic, downloadFrom, $z*], read,

-, R>, <*Dave*, [*Video, downloadFrom, $z*], read, -, R>}의 접근 권한들을 유도해 낸다. 따라서 Dave가 어디에서 음악 파일들을 다운로드할 수 있는 지를 질의하고자 할 때, 유도된 접근 권한 <Dave, [*Classic, downloadFrom, $z*], read, -, R>에 의해 다음 질의는 거절된다.

prefix ex: <http://example.org/schemas/contents/>
sekect ?uri
from <http://example.org/data/Music.rdf#/>
where { ex:Classic ex:downloadFrom ?uri . }

A_j = <*Dave*, [*Art, $y, $z*], *read*, -, *R*>이 명세될 때, $P(A_j)$ = $y 이기 때문에, A_j는 *Art*의 모든 속성을 계승하는 하위 클래스들에도 동일하게 적용된다. 따라서 다음 Q-pattern들을 포함하는 RDF 질의와 권한 충돌 관계를 갖는다: {[*Painting, price, $z*], [*Painting, createdBy, $z*], [*Painting, paintedBy, $z*], [*Painting, ownedBy, $z*], [*Painting, $y, $z*], [*Sculpture, price, $z*], [*Sculpture, createdBy, $z*], [*Sculpture, sculptedBy, $z*], [*Sculpture, $y, $z*], [*Video, price, $z*], [*Video, createdBy, $z*], [*Video, $y, $z*]).

마찬가지로 *subPropertyOf* 관계에 있어서도, 만약 $p_i \subset p_j$ 이면, p_j에 대한 접근 권한 A_j는 속성 p_i에 대해서도 적용된다. 따라서 RDF 질의가 입력될 때 *subPropertyOf* 계층 구조에 대해서도 명시적 권한 전파에 의한 질의 유효성 검증이 이루어 져야 한다. 예로 A_j =<Dave, [*Art, createdBy, $z*], read, -, R>는 <Dave, [*Sculpture, sculptedBy, $z*], read, -, R>를 유도하기 때문에, 다음 SPARQL 질의는 거절되어야 한다.

prefix ex: <http://example.org/schemas/contents/>
select ?sculptor
from <http:example.org/data/Art.rdf#>
where { ex:Sculpture ex:sculptedBy ?sculptor . }

A_j = <Dave, [*Art, $y, $z*], read, -, R>에 관하여, Art의 모든 속성의 모든 하위 속성들과 매치되는 Q-pattern을 가지는 RDF 질의는 거절된다. 어떤 클래스 인스턴스 i_l를 S 값으로 가지는 RDF 질의는 i_l의 클래스에 대하여 정의된 접근 권한 A를 따른다. 또한 i_l이 어떤 속성 p_k를 가질 때, RDF 질의는 p_k에 대하여 정의된 접근 권한 A를 따른다.

5.2 암시적 접근 권한 전파에 따른 유효성 검증

명시적 권한 전파가 계승에 의한 것이라면, 암시적 권한 전파는 RDF 추론에 의한 권한 전파를 의미한다. 본 절에서는 암시적 권한 전파에 의하여 RDF 질의가 거절되는 것을 선언한다. 정의 2에서 설명한 것처럼, 하위 클래스와 속성들은 *subClsassOf*와 *subPropertyOf* 추론에 의하여 상위 클래스와 속성들로 해석될 수 있으므로, 다음 네 가지 형태의 질의와 권한 사이에서의 관계성을 갖는다.

- $S(pt_i(Q)) \subset S(pt(A))$, $sign(A)$ = -, $type(A)$ = L, $Q+ \Rightarrow A'+$ (충돌): Q를 RDF 질의, A를 접근 권한이라고 하자. 그림 4(a)에서처럼, $S(pt_i(Q)) \subset S(pt(A))$를 만족하는 Q-pattern pt_i가 존재할 경우, $pt_i(Q)$에 매칭되는 RDF 트리플들은 *subClassOf* 추론에 의하여 $pt(A)$에 매칭되는 RDF 트리플로 해석될 수 있다. 따라서 질의 Q를 허용하는 것 (= $Q+$)은 $sign(A)$ = +를 요구한다. 이러한 암시적 권한 전파를 $Q+ \Rightarrow A'+$로 표시한다. 하지만 $sign(A-) \neq sign(A'+)$이므로, 이것은 충돌이다. 예로, A = <Dave, [*Music, downloadFrom, $z*], read, -, L>이 명세될 경우 다음 RQL 질의는 거절되어야 한다. 이것은 $pt(A)$가 $pt(Q)$로부터 추론될 수 있기 때문이다. 주목해야 할 점은 $type$(A) = L이므로 이것은 명시적 권한 전파에 해당되지 않는다는 것이다.

  select ?uri
  from ex:Classic.ex:downloadFrom{?uri}
  using namespace ex = http://example.org/schemas/congtents/
- $S(pt_i(Q)) \subset S(pt(A))$, $sign(A)$ = +, $type(A)$ = L, $Q+ \Rightarrow A'+$ (비충돌): 이러한 경우에는, $sign(A+) \equiv sign(A'+)$ 이기 때문에, 충돌이 아니다.
- $S(pt(A)) \subset S(pt_i(Q))$, $sign(A)$ = -, $type(A)$ = L/R, $A- \Rightarrow Q'-$ (충돌): 그림 4(b)에서처럼, $pt(A)$에

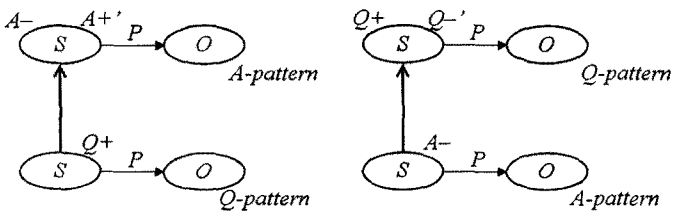


그림 4 *subClassOf* 추론에서의 암시적 권한 충돌

대한 매칭 트리플이 *subClassOf* 추론에 의하여 $pt_i(Q)$의 매칭 트리플로 해석되는 것을 고려하자. 즉, 3.2절에서 설명된 것처럼, RQL과 같은 RDF 질의 언어는 추론된 질의 답변을 지원한다. 이러한 경우에 암시적 권한 전파 $A^- \Rightarrow Q'^-$가 고려되어야 한다. $sign(Q^+) \neq sign(Q'^-)$이기 때문에, 이것은 충돌이다. 예로, 3.2절의 Q3는 *Music*을 접근하기 때문에, Q3는 <Dave, [Classic, downloadFrom, $z], read, -, L>과 충돌 관계를 갖는다. $S(pt(A)) \subset S(pt_i(Q))$이기 때문에 명시적 권한 전파는 일어나지 않는다. 따라서 $type(A)$의 값에 상관없이 암시적 권한 전파가 고려되어야 한다.

- $S(pt_i(A)) \subset S(pt_i(Q))$, $sign(A) = +$, $type(A) = L/R$, $A^+ \Rightarrow Q'^+$ (비충돌): 이러한 경우에, $sign(Q^+) \equiv sign(Q'^+)$ 이기 때문에, 비충돌이다.

  subClassOf 추론과 유사하게 *subPropertyOf* 추론에서도 두 가지 충돌 조건이 존재한다.

- $P(pt_i(Q)) \subset P(pt(A))$, $sign(A) = -$, $type(A) = L$, $Q^+ \Rightarrow A'^+$ (충돌): A_1^- = <Dave, [*Art, createdBy, $z*], read, -, L>과 A_2^- = <Dave, [*Art, price, $z*], read, -, L>에 대한 다음 RQL 질의를 고려하자.

  select ?*sculptor*
  from ex:Sculpture.ex:sculptedBy{?*sculptor*}
  using namespace ex = http://example.org/schemas/contents/

  비록 위의 질의는 A_2^-와 충돌 관계가 아니지만, 접근 권한 A_1^-에 의해서 거절되어야 한다. 이것은 A-pattern [*Art, createdBy, $z*]이 *subPropertyOf* 추론에 의해서 Q-pattern [*Sculpture, sculptedBy, $z*]로부터 추론되기 때문이다.

- $P(pt(A)) \subset P(pt_i(Q))$, $sign(A) = -$, $type(A) = L/R$, $A^- \Rightarrow Q'^-$ (충돌): 다음 RQL 질의는 *subPropertyOf* 추론에 의해 접근 권한 <Dave, [*Classic, composedBy, $z*], read, -, L>와 충돌 관계를 가진다.

  select ?*creator*
  from ex:Music.ex:createdBy{?*creator*}
  using namespace ex = http://example.org/schemas/contents/

6. 효율적 질의 유효성 검증 알고리즘

그림 4에서처럼 입력된 RDF 질의와 충돌되는 (-) *sign*을 갖는 접근 권한이 있을 경우, 사실 이것은 항상 충돌이다라고 말할 수 없다. 몇가지 예외 상황이 있다. 본 절에서는 정의 1, 3, 4에서의 S와 P값의 유형에 따라 이러한 예외 상황을 분석한다. : $S \in$ {클래스 URI 상수, 인스턴스 URI 상수}, $P \in$ {$y, 속성 URI 상수}. S = $x의 경우에는 속성 P를 가지는 최상의 클래스의 URI 상수로 대체할 수 있다. 따라서 S 유형에 대해서는 클래스와 인스턴스 URI 상수만을 고려한다. 예로 RDF 패턴 [$x, *price*, $z]는 속성 *price*를 가지는 모든 클래스와 인스턴스와 매치되기 때문에, 최상위 클래스 *Contents*로 $x를 대체할 수 있다: [*Contents, price*, $z]. 그림 5의 충돌 판정 테이블 (CDT)는 (S, P) 값들의 쌍에 의해서 충돌의 가능성을 정리한다. 'O'는 충돌을 의미하고, 'X'는 비충돌을 의미하며, '?'은 충돌이 있는지를 더 조사해 보아야 하는 것을 의미한다. 다음 예를 통하여 CDT에 관하여 설명하도록 한다.

예 2. 그림 6의 접근 권한들에 대하여 다음 SPARQL 질의 Q5를 고려하자.

Q5: *select ?y, ?z*
from <http://example.org/data/Music.rdf#>
where { Classic ?y ?z . }

우선 *Music*은 *Classic*의 조상 클래스이고, R1은 (-) *sign*을 갖고 L *type*이기 때문에, R1은 Q5와 암시적 권한 전파에 의해서 충돌 관계를 가질 수 있다. S(R1) $\in$ 클래스 URI, S(Q5) $\in$ 클래스 URI, P(Q5) = ?y, P(R1) = $y이기 때문에, CDT의 규칙 1에 의해서 최종적인 판정은 충돌이다. 이러한 판정의 근거는 "P(Q5) = ?y "가 *Music*으로부터 계승된 모든 속성을 포함하기 때문

ancestor RDF pattern(-) / descendant RDF pattern(+)		p=$y or ?y		p=URI	
		s=class URI	s=instance URI	s=class URI	s=instance URI
p=$y or ?y	s=class URI	1) O	2) X	3) O	4) X
	s=instance URI	5) O	6) X	7) O	8) X
p=URI	s=class URI	9) ?	10) X	11) ?	12) X
	s=instance URI	13) ?	14) X	15) ?	16) X

(a) *subClassOf* 관계

ancestor RDF pattern(-) / descendant RDF pattern(+)		p=$y or ?y		p=URI	
		s=class URI	s=instance URI	s=class URI	s=instance URI
p=$y or ?y	s=class URI	17) O	18) X	19) O	20) X
	s=instance URI	21) O	22) X	23) O	24) X
p=URI	s=class URI	25) O	26) X	27) O	28) X
	s=instance URI	29) O	30) X	31) O	32) X

(b) *subPropertyOf* 관계

그림 5 CDT (충돌 판정 테이블): O 충돌, X 비충돌, ? 추가적 검증

```
R1: <Dave, [Music, $y, $z], read, -, L>
R2: <Dave, [Art, price, $z], read, -, L>
R3: <Dave, [Twinkle_Twinkle_Little_Star, $y, $z], read, -, R>
```

그림 6 RDF 저장소에 저장된 샘플 접근 권한

이다. 유사하게 [*Contents*, *?y*, *?z*]와 매칭되는 트리플들을 접근하는 RQL 질의는 규칙 1에 의해서 R1과 충돌이다.

예 3. 그림 6의 접근 권한들에 대하여 SPARQL 질의 Q6를 살펴보자.

Q6: *select ?y, ?z*
from <http://example.org/data/Art.rdf#>
where { Painting ?y ?z . }

Q6는 CDT의 규칙 3에 의해서 R2와 충돌이다: $sign(Q6) = +$, $sign(R2) = -$, $S(Q6) \in$ 클래스 URI, $S(R2) \in$ 클래스 URI, $P(Q6) = ?y$, $P\ (R2) \in$ 속성 URI. "$P(Q6) = ?y$"는 Q6가 *Art*로부터 계승된 *price*를 접근하는 것을 의미한다. 유사하게 [*contents*, *createdBy*, *?z*]에 매치되는 트리플들을 접근하는 RQL 질의는 규칙 3에 의해서 R1과 충돌이다.

예 4. 다음 SPARQL 질의 Q7를 고려하자.

Q7: *select ?z*
from <http://example.org/data/Music.rdf#>
where { Pop downloadFrom ?z . }

$sign(R1) = -$, $sign(Q7) = +$, $S(R1) \in$ 클래스 URI, S(Q7) $\in$ 클래스 URI, P(R1) = $y, P(Q7) $\in$ 속성 URI이기 때문에, CDT의 규칙 9에 의해서 추가적인 충돌검증 작업이 요구된다. 상기 예에서는 추가적 검증에 의한 최종 판정은 충돌이다. 왜냐하면 속성 *downloadFrom*이 *Music*으로부터 계승되기 때문이다. 하지만, 만약 *downloadFrom*이 계승되지 않았다면, 이것은 비충돌이다. 유사하게, 규칙 11의 경우에서도 두 속성 URI들이 동일 URI라면 충돌이고, 그렇지 않으면 비충돌이다. 예 4는 규칙 13과 규칙 15에도 적용되어진다.

예 5. R1에 대하여 SPARQL 질의 Q8를 고려하자.

Q8: *select ?y, ?z*
from <http://example.org/data/Music.rdf#>
where { Fate ?y ?z . }

$sign(R1) = -$, $sign(Q8) = +$, $S(R1) \in$ 클래스 URI, S(Q8) $\in$ 인스턴스 URI, P(R1) = $y, P(Q8) $\in$ $y이기 때문에, CDT의 규칙 5에 의해서 이것은 무조건 충돌이다. 왜냐하면, 인스턴스 *Fate*는 *Music*으로부터 계승하기 때문이다. 상기 예는 CDT의 규칙 7에 또한 적용된다. 다음으로, Q8과 R3를 고려하자. 이 경우에, $Fate \neq Twinkle\_Twinkle\_Little\_Star$, 즉, 두 인스턴스가 서로 다른 객체이므로, 이것은 비충돌이다. 더욱 일반화하여, URI (Uniform Resource Identifier)에 의해서 표현되는 RDF 인스턴스는 해당 웹상에서 유일하기 때문에, S에 대하여 인스턴스 URI를 가지는 두 접근 권한은 무조건 비충돌이다. 정리1은 질의 유효성 검증 과정을 보다 단순화 시키는 인스턴스 URI의 이러한 특성을 정리한다.

정리 1. 인스턴스 URI i를 S 값으로 가지는 RDF 접근 권한 혹은 질의는 i가 속하는 클래스 c_i와 c_i의 조상 클래스 c_j ($c_i \subset c_j$)를 S 값으로 가지는 RDF 접근 권한 혹은 질의와 단지 충돌이다. 그 외의 경우는 비충돌이다.

증명: (1) RDF 인스턴스의 URI 값은 웹에서 유일하기(unique) 때문에, S에 대하여 다른 인스턴스 URI 값을 가지는 접근 권한과 질의는 비충돌이다. (2) 어떤 RDF 인스턴스는 *subClassOf* 추론에 의해서 단지 상위의 클래스로 추론되므로, 하위 클래스를 S 값으로 가지는 접근 권한 혹은 질의와는 비충돌이다. (1)과 (2)에 의해서, 인스턴스 URI를 S 값으로 가지는 RDF 접근 권한 혹은 질의는 인스턴스가 해당하는 클래스 혹은 조상 클래스를 S 값으로 갖는 접근 권한 혹은 질의와만 충돌 관계를 갖는다. □

증명 (1)에 의해서, CDT의 규칙 6, 8, 14, 16은 비충돌이며, 증명 (2)에 의해서 규칙 2, 4, 10, 12는 비충돌이다.

예 6. 그림 5(b)의 *subPropertyOf* CDT에서, 정리1에 해당하는 규칙들을 제외한 모든 규칙들은 충돌이다. *subPropertyOf* CDT는 예 5에서처럼 계승되지 않은 속성에 대하여 비충돌인 경우를 가지지 않는다. 이것은 *subPropertyOf* 관계가 속성들 사이에서의 포함관계를 직접적으로 정의하기 때문이다. 예로 다음 SPARQL 질의 Q9을 살펴보자.

Q9: *select ?z*
from <http://example.org/data/Music.rdf#>
where { Clssic composedBy ?z . }

subClassOf CDT의 규칙 9에 의해서 R1에 대하여 유효성 검증이 요구된다. 하지만, *subPropertyOf* CDT의 규칙 25에 의해서 위 경우는 무조건 충돌이다. 왜냐하면 S(R1)=$y이지만, *subPropertyOf* 관계에서 $y는 속성 *createdBy*를 의미하기 때문이다(그림 1의 RDF 그래프를 참조). 다음 SPARQL 질의를 또한 살펴보자.

Q10: *select ?x, ?z*
from <http://example.org/examples/Art.rdf#>
where { ?x price ?z . }

***Q10*은** *subPropertyOf* CDT에 대하여 검증될 필요가 없다. 단지 *subClassOf* CDT에 대하여 검증되기만 하면 된다. 왜냐하면 P(Q10) = *price*는 *subPropertyOf*

관계에 포함되지 않기 때문이다.

부록의 알고리즘은 그림 5의 CDT를 반영한다. 먼저 라인 1에서는 주어진 RDF 질의 Q에서의 모든 Q-pattern pt_i에 대하여 접근권한과의 충돌 여부를 조사한다. *subClassOf* 관계에 대하여, 라인 2부터 12까지의 코드는 Q에 대한 명시적 혹은 암시적 권한 전파에 의한 상위 클래스를 S로 가지는 접근 권한과의 충돌 여부를 확인한다. 라인 12에서 22사이의 코드는 암시적 권한 전파에 의한 하위 클래스를 S로 가지는 접근 권한과의 충돌 여부를 체크한다. 라인 5와 16은 예 4의 추가적인 조사를 필요로 하는 예외 상황을 반영한다. 라인 8과 19는 정리 1의 비충돌의 예외 상황을 반영한다. 다음으로 *subPropertyOf* 관계에 대하여 라인 23에서 37사이의 코드가 질의 유효성 검증을 수행한다. 라인 26과 34에 의해서 확인되는 정리 1의 경우를 제외한 나머지 모든 경우는 충돌이다. S와 P에 대한 클래스와 속성의 조상/후손관계를 효율적으로 파악하기 위하여 Dewey 그래프 레이블링 기술[16]을 사용하였다. 이전 연구[3]에서는 그래프 레이블링 방법으로 소수 그래플 레이블링을 사용하였지만, 소수 그래프 레이블링의 계속적인 레이블 곱셈에 의한 레이블 변수의 오버플로우 현상으로 인하여 이의 사용이 적절치 못하다는 것을 결론지었다. 따라서 가장 일반적이며 효율적인 방법으로 생각되는 Dewey 그래프 레이블링을 사용하여 알고리즘을 구현하였다.

제안된 알고리즘의 시간 복잡도는 다음과 같다. 주어진 RDF 질의 Q에 대하여 질의 유효성 검증을 수행할 접근권한의 수를 a라고 하면, 제안된 알고리즘은 라인 2, 13, 23, 31 에서의 레이블링 방법에 의하여 a번 만에 해당 질의에 관련된 접근권한들을 찾아낼 것이다. 또한 라인 4, 15, 25, 33에서는 찾아진 권한들 ($<< a$)에 대하여 질의 유효성 검증을 수행한다. 따라서 시간 복잡도는 최악의 경우 $O(a)$이다. 이러한 과정이 라인 1에서처럼 질의 Q의 Q-pattern의 수만큼 수행되므로, 주어진 질의 Q에서의 Q-pattern의 수를 q라고 할 경우, 시간 복잡도는 $O(q \times a)$이다.

7. 실 험

조사한 바에 의하면 본 논문에서 고려한 RDF 질의 유효성 검증과 관련하여 직접적으로 비교 가능한 대상이 없었다. 하지만 이전 연구[3]에서와 같이 Jain과 Farkas가 제안한 방법[4]과의 비교 수행을 통하여 제안된 방법의 효율성을 가늠하였다. 앞서 설명한 것처럼 본 연구에서는 이전 연구와 달리 Dewey 그래프 레이블링을 사용하였으며, 소수 그래프 레이블링을 사용하는 경우와 수행 성능의 큰 차이가 없음을 확인하였다.

7.1 실험 환경

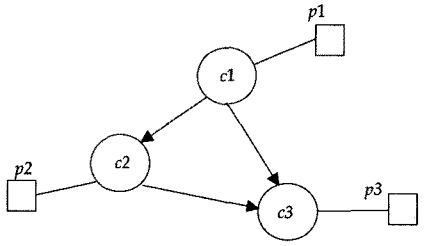


그림 7 실험 DAG의 임의 생성

표 1 실험 인자

실험인자	값의 범위	설명
#C	100 to 1,000	하나의 DAG 에서의 클래스의 수
#P	1 to 5	하나의 DAG 에서의 각 클래스에 대한 속성의 수
#S	1 to 5	하나의 DAG 에서의 각 클래스에 대한 *subClassOf* 의 평균수
#A	10 to 500	RDF 질의와 충돌 검사될 접근 권한들의 수

실험을 위한 RDF 데이터로 그림 7에서와 같이 임의의 DAG (Directed Acyclic Graph)가 실험 인자 *#C*, *#P*, *#S*에 따라 랜덤하게 생성된다. 표 1에서 *#C*는 그래프의 클래스의 수이며, *#P*는 속성의 수이며, *#S*는 하나의 클래스 노드에 대한 *subClassOf* 관계의 평균수이다. 예로 그림 7에서 *c3*는 *c1*과 *c2*의 하위 클래스이므로 *#S*는 2이다. 실험 RDF 질의는 DAG에 대하여 아래와 같이 랜덤하게 생성된다.

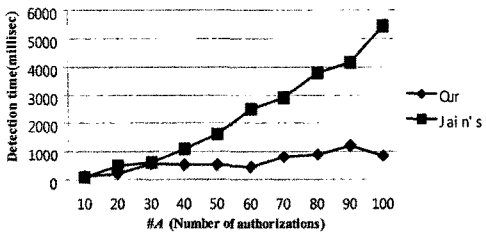
```
select ?x from <test.rdf>
where { c2 p2 ?x . }
```

제안된 알고리즘의 평가에 있어 하나의 RDF 질의가 여러 개의 Q-pattern $pt_i(Q)$를 가질 경우, 각각의 Q-pattern에 대하여 접근 권한들과의 충돌 여부를 조사한다. 하지만 실험에서는 하나의 RDF 질의가 단일 Q-pattern을 가지는 것으로 실험하였다. 이는 RDF 질의가 가지는 Q-pattern의 수를 예측할 수 없기 때문이다. 하지만 하나의 RDF 질의가 여러 개의 Q-pattern을 가지는 경우, 본 절에서 측정된 실험값에 Q-pattern의 수만큼을 곱하면 해당 유효성 검증 시간을 예측할 수 있다.

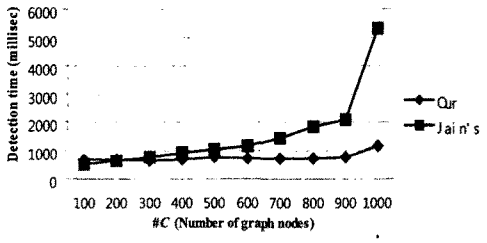
모든 실험은 1GB 메모리와 1.86 GHz Intel(R) Core(TM) 2 CPU를 가진 윈도우 서버 2008 엔터프라이즈 컴퓨터에서 실험되었으며, 모든 프로그램은 Java 언어로 구현되었다.

7.2 실험 결과

*#C*와 *#S*의 값을 크게 하는 것은 DAG의 크기가 커지는 것을 말한다. 즉, 실험대상의 RDF 스키마와 데이터가 더 커지며 더 복잡해지는 것을 나타낸다. *#A*의 값을

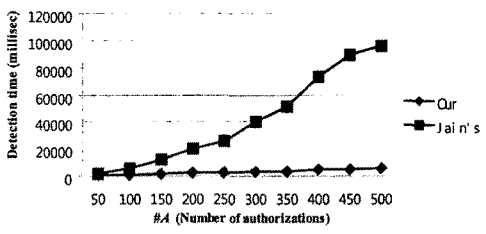


(a) #C = 1000, #S = 3

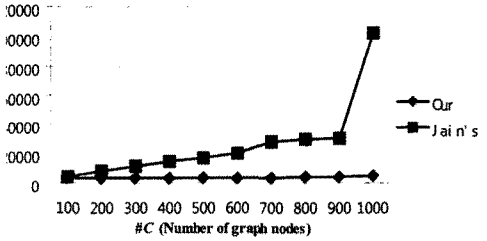


(b) #A = 100, #S = 3

그림 8 질의 유효성 검증 시간 비교



(a) #C = 1000, #S = 5



(b) #A = 500, #S = 5

그림 9 추가적 실험 결과

크게 하는 것은 입력된 RDF 질의에 대하여 검사될 접근 권한의 수가 많다는 것을 의미한다. 그림 8(a)의 그래프는 $\#C$ = 1,000, $\#S$ = 3일 때 $\#A$에 따른 유효성 검증 시간을 보여준다. 제안된 질의 유효성 검증은 Jain과 Farkas의 방법과 비교하여 적은 수행 시간을 갖는다. 또한 $\#A$에 따른 유효성 검증 시간의 증가율이 낮다는 것을 확인할 수 있다.

다음으로 그림 8(b)의 그래프는 $\#A$ = 100, $\#S$ = 3일 때 $\#C$에 따른 질의 유효성 검증 시간을 보여준다. 역시 낮은 질의 유효성 검증 시간과 낮은 증가율을 확인할 수 있다. 그림 9의 그래프들은 추가적인 실험 결과를 보여준다. 두 그래프 모두 데이터와 접근 권한의 수가 증가할 때 확장성(scalability)을 가짐을 보여 준다.

이러한 실험결과는 참고문헌[3]에서 분석한 바와 같이 Jain과 Farkas의 방법이 모든 인스턴스 데이터에 대하여 보안 레이블을 할당하는 일종의 전체 데이터 스캔을 요구하기 때문이다. 하지만 본 연구의 제안 방법은 5장과 6장에서 분석한 충돌조건을 접근권한의 수준에서 검증하며, 보다 신속한 조상/후손 관계의 파악을 위하여 Dewey 레이블링을 사용하고 있다.

8. 결론 및 향후 연구

본 논문에서는 기 연구된 RDF 접근권한 충돌 문제[2,3]를 RDF 질의 유효성 검증 문제에 적용하였다. 또한 실험을 통하여 제안된 질의 유효성 검증 알고리즘이 RDF 데이터 규모와 접근권한의 수가 증가할수록 낮은 유효성 검증 시간의 증가율을 보이는 것을 확인하였다. 본 논문에서의 연구 결과는 이전의 연구 결과와 더불어 5장에서의 접근 권한 충돌 조건의 개념과 6장에서의 권한 충돌 알고리즘이 RDF 접근 제어 시스템에서 핵심적으로 활용될 수 있음을 보여주었다.

RDF 질의 유효성 검증과 관련한 향후 연구 계획은 접근 권한에 따른 RDF 질의 재작성이다. RDF 질의 재작성은 접근 권한에 따라 일부 RDF 데이터가 접근 불허되지만, 허용되는 데이터는 접근 가능하도록 입력된 질의를 수정하여 질의 처리기에서 수행하는 것이다. 즉, 사용자가 어떤 RDF 질의를 입력할 경우, 비록 RDF 질의가 접근 권한과 충돌되어 거부되어야 하지만, 자신이 볼 수 있는 일부 RDF 데이터가 있다면 이를 볼 수 있도록 허용하는 것이다. 본 연구에서는 제안된 알고리즘에 의해 발견된 충돌 Q-pattern을 삭제하고, 비충돌 Q-pattern을 이용하여 질의 재작성하는 방법을 연구하고 있다.

참 고 문 헌

[1] W3C Semantic Web Activity, http://www.w3.org/2001/sw/

[2] J. Kim, S. Park, "Analysis of Access Authorization Conflict for Partial Information Hiding of RDF Web Document," *Journal of KIISC*, vol.18, no.2, pp.49-63, April 2008. (in Korean)

[3] J. Kim, S. Park, "Efficient Authorization Conflict Detection Using Prime Number Graph Labeling in RDF Access Control," *Journal of KIISE : Databases*, vol.35, no.2, pp.112-124, April 2008. (in Korean)
[4] A. Jain, C. Farkas, "Secure resource description framework: an access control model," *Proc. of 11th ACM Symposium on Access Control Models and Technologies*, 2006, pp.121-129.
[5] L. Qin, V. Atluri, "Concept-level Access Control for the Semantic Web," *Proc. of ACM Workshop on XML Security*, 2003, pp.94-103.
[6] D. Jeong, Y. Jing, D. Baik, "A Three-Layered Ontology View Security Model for Access Control of RDF Ontology," *Journal of KIISE : Databases*, vol.35, no.1, pp.29-43, Feb. 2008. (in Korean)
[7] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Sinsborough, B. Thuraisingham, "ROWLBAC - representing role based access control in OWL," *Proc. of SACMAT 2008*, pp.73-82.
[8] J. Lee, K. Whang, W. Han, I. Song, "The Dynamic Predicate: Integrating Access Control with Query Processing in XML Databases," *VLDB Journal*, vol.16, no.3, pp.371-387, July 2007.
[9] V. Gaede, O. Gunther, "Multidimensional access methods," *ACM Comput. Surv.*, vol.30, pp.170-231, 1998.
[10] J. Kim, S. Park, "A Circle Labeling Scheme without Re-labeling for Dynamically Updatable XML Data," *Journal of KIISE : Databases*, vol.36, no.2, pp.29-43, April 2009. (in Korean)
[11] C. Byun, S. Park, "An Efficient Query-based XML Access Control Enforcement Mechanism," *Journal of KIISE : Databases*, vol.34, no.1, pp.1-17, Feb. 2007. (in Korean)
[12] RDF Semantics, W3C Recommendation, http://www.w3.org/TR/rdf-mt/
[13] G. Karvounarakis, S. Alexaki, M. Scholl, V. Christophides, D. Plexousakis, "RQL: a declarative query language for RDF," *Proc. of WWW 2002*, pp.592-603.
[14] SPARQL Query Language for RDF, W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/
[15] T. Furche, B. Linse, F. Bry, D. Plexousakis, G. Gottlob, "RDF querying: language constructs and evaluation methods compared," *Proc. of Reasoning Web 2006*, pp.1-52.
[16] V. Christophides, G. Karvounarakis, D. Plexousakis, M. Scholl, S. Tourtounis, "Optimizing taxonomic semantic web queries using labeling schemes," *Journal of Web Semantics*, 11(1) (2003) 207-228.

부록. 질의 유효성 검증 알고리즘

Algorithm Method1
Input: 입력된 RDF 질의 Q
Output: Q와 충돌 관계를 갖는 접근 권한들의 집합 *Conflict_Set*

```
1 foreach pt_i ∈ Q do   /* Q의 모든 Q-pattern pt_i에 대하여 수행 * /

    /* 명시적 & 암시적 권한 전파: pt_i와 subClassOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 상위 접근 권한들을 조사 * /
2   Dewey 그래프 레이블을 이용하여 pt_i와 subClassOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 상위 접근 권한들 검색하여
RS에 저장;
3   if RS is not empty then
4       foreach rs_i ∈ RS do
5           if ((P(pt_i) ∈{속성 URI} ∧ P(rs_i) = $y ∧ P(pt_i) ∉ properties(S(rs_i))) ∨ (P(pt_i), P(rs_i) ∈속성 URI ∧P(pt_i) ≠
P(rs_i))) then     /* 이것은 예 4의 추가적인 검사를 수행 */
6               Conflict-free; Continue;
7           else
8               if S(rs_i) ∈인스턴스 URI then
9                 Conflict-free; Continue;  /* this checks the case of Theorem 1 */
10              else
11                  Conflict;
12                  Conflict_Set := Conflict_Set ∪ rs_i.authorization_ID

    /* 암시적 권한 전파: pt_i와 subClassOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 하위 접근 권한들을 조사 */
13  Dewey 그래프 레이블을 이용하여 pt_i와 subClassOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 상위 접근 권한들 검색하여
RS에 저장;
14  if RS is not empty then
15      foreach rs_i ∈ RS do
16          if ((P(rs_i) ∈{속성 URI} ∧ P(pt_i) = $y ∧ P(rs_i) ∉ properties(S(pt_i))) ∨ (P(pt_i), P(rs_i) ∈속성 URI ∧P(pt_i)      ≠
P(rs_i)))   /* 이것은 예 4의 추가적인 검사를 수행 */
17              Conflict-free; Continue;
```

```
18              else
19                 if S(Q) ∈instance URI then Conflict-free; Continue;  /* 이것은 정리 1의 예외 상황을 검사 */
20                  else
21                        Conflict;
22                        Conflict_Set := Conflict_Set ∪ rsi.authorizationID

    /* 명시적 & 암시적 권한 전파: pti와 subPropertyOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 상위 접근 권한들을 조사 * /
23    Dewey 그래프 레이블을 이용하여 pti와 subClassOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 상위 접근 권한들 검색하여
RS에 저장;
24    if RS is not empty then
25       foreach rsi ∈ RS do
26            if S(rsi) ∈인스턴스 URI then
27               Conflict-free; Continue;  /* 이것은 정리 1의 예외 상황을 검사 */
28            else
29               Conflict;
30               Conflict_Set := Conflict_Set ∪ rsi.authorizationID

    /* 암시적 권한 전파: pti와 subPropertyOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 하위 접근 권한들을 조사 */
31    Dewey 그래프 레이블을 이용하여 pti와 subPropertyOf 관계에 있어 포함 관계를 갖는 (-) sign 값을 갖는 하위 접근 권한들 검색하
여 RS에 저장;
32    if RS is not empty then
33       foreach rsi ∈ RS do
34            if S(pti) ∈인스턴스 URI then Conflict-free; Continue; /* 이것은 정리 1의 예외 상황을 검사 */
35          else
36             Conflict;
37               Conflict_Set := Conflict_Set ∪ rsi.authorizationID

38    return Conflict_Set;
```

김 재 훈

1997년 건국대학교 전자계산학과 공학사 1999년 건국대학교 컴퓨터·정보통신공학과 공학석사. 2005년 서강대학교 컴퓨터공학과 공학박사. 2005년 3월~2006년 9월 삼성전자 정보통신총괄 통신연구소 책임연구원. 2006년 9월~2009년 2월 서강대학교 컴퓨터공학과 연구교수. 2009년 3월~현재 서일대학 정보통신과 전임강사. 관심분야는 유비쿼터스 컴퓨팅 환경에서의 상황정보(Context) 관리, Data Privacy, Data Stream Management System(DSMS), 데이터베이스, 데이터베이스 보안 임

박 석

정보과학회논문지 : 데이타베이스
제 36 권 제 3 호 참조