SPARQL-to-SQL 변환 알고리즘의 저장소 독립적 활용을 위한 시스템 모델

(A System Model for Storage-Independent Use of SPARQL-to-SQL Translation Algorithm)

손 지 성 † 정 동 원 ††

(Jiseong Son) (Dongwon Jeong)

백 두 권 †††

(Doo-Kwon Baik)

요 약 웹 온톨로지에 대한 연구가 활발해지면서 웹 온톨로지를 저장하기 위한 다양한 형태의 저장소와 질의 언어가 개발되고 있다. SPARQL의 이용이 증가하고 대부분 관계형 데이타베이스 기반의 저장소를 이용함에 따라 SPARQL을 SQL로 변환하는 알고리즘 개발의 필요성이 대두되었다. 지금까지 제안된 변환 알고리즘들은 SPARQL의 일부만을 SQL로 변환하거나 변환 알고리즘이 저장소 구조에 종속적이라는 문제점이 있다. 이 논문에서는 저장소에 독립적으로 특정 변환 알고리즘을 활용할 수 있는 모델을 제안한다.

키워드 : SPARQL, SQL, 변환, 웹 온톨로지

Abstract With active research on Web ontology, various storages and query languages have been developed to store Web Ontology. As SPARQL usage increases and most of storages are based on relational database, the necessity of SPARQL-to-SQL translation algorithm development becomes issued. Even though several translation algorithms have been proposed, there still remain the following problems. They do not support fully SPARQL clauses and they are dependent on a specific storage model. This paper proposes a new model to use a specific translation algorithm independently on storages.

Key words : SPARQL, SQL, Translation, Web Ontology

· 이 연구에 참여한 연구자는 '2단계 BK21 사업'의 지원을 받았음
· 이 논문은 제34회 추계학술대회에서 'SPARQL-to-SQL 변환 알고리즘의 저장소 독립적 활용을 위한 시스템 모델'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 고려대학교 컴퓨터학과
redfunky07@gmail.com
†† 종신회원 : 군산대학교 정보통계학과 교수
djeong@kunsan.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
baikdk@korea.ac.kr

논문접수 : 2007년 12월 7일
심사완료 : 2008년 4월 24일

Copyright@2008 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지 : 컴퓨팅의 실제 및 레터 제14권 제5호(2008.7)

1. 서 론

웹의 발전과 함께 정보의 양이 기하급수적으로 증가하면서 기존의 사용자 중심의 웹은 사용자가 원하는 정확한 정보를 검색하지 못하는 문제점이 있다. 이를 해결하기 위해 시맨틱 웹이 제안되었으며 시맨틱 웹을 구축하기 위해 RDF, RDF Schema, OWL 등의 웹 온톨로지 언어들이 개발되었다[1].

시맨틱 웹 환경에서 웹 온톨로지 정보를 검색하기 위한 RQL[2], RDQL[3], SPARQL과 같은 다양한 질의 언어가 개발되었으며, 이 중에서 SPARQL (SPARQL Protocol and RDF Query Language)[4]은 W3C에 의해 권고안으로 채택된 언어로서 이를 지원하는 다수의 시스템이 개발되고 있다. SPARQL은 기본적으로 SELECT 절과 WHERE 절로 이루어져 있으며 조건문인 WHERE 절은 트리플 형태로 표현된다.

이러한 질의 언어 개발과 더불어 웹 온톨로지 저장소에 대한 연구 또한 활발하게 진행되고 있으며, 특히 관계형 데이타베이스를 기반으로 한 다양한 저장소들이 개발되고 있다[5-10].

대부분의 웹 온톨로지 저장소는 관계형 모델을 기반으로 하고 있으며 웹 온톨로지에 대한 질의 언어로서 SPARQL의 활용성이 높아지고 있다. 그런 이유에 의해 SPARQL을 SQL로 변환하는 알고리즘에 대한 개발이 요구되고 있다.

기존의 SPARQL-to-SQL 변환 알고리즘으로는 Chebotko 알고리즘[11], Jena의 sparql2sql[12], Harris 알고리즘[7] 등이 있다. Chebotko 알고리즘은 중첩 OPTIONAL 절을 SQL로 변환하는 과정에서 기존에 제안된 변환 알고리즘 중 높은 기능성을 제공하지만 SPARQL의 UNION, FILTER 구문에 대한 변환 기능은 제공하지 않는다. 이 알고리즘은 여러 다른 구조의 저장소들 중에서[13] 가장 간단한 구조인 트리플 저장소를 사용한다. 그러나 알고리즘이 트리플 구조에 대하여 종속적이기 때문에 다른 구조의 저장소는 이 알고리즘을 적용시킬 수 없다. 저장소가 다른 구조로 변경될 때마다 변환 알고리즘 또한 그 구조에 맞게 수정되어야 한다. 예를 들어, 트리플 저장소를 사용하는 Chebotko 알고리즘은 다른 구조를 지닌 Jena 저장소나 Sesame[14] 저장소 등에

서 사용될 경우 각 저장소 구조에 맞게 변경되어야 한다.

Jena의 sparql2sql과 Harris 알고리즘은 SPARQL의 UNION과 FILTER 구문 변환을 지원하지 않고 중첩 OPTIONAL 절 변환에 한계가 있다. 또한 Chebotko 알고리즘과 같이 알고리즘이 물리적 구조에 종속적이라는 문제점을 지니고 있어 저장 구조 변경에 따른 알고리즘의 수정이 요구된다.

이 논문에서는 변환 알고리즘의 기능성이 아닌 활용성 문제에 초점을 둔다. 즉, 저장소의 물리적 구조에 독립적으로 변환 알고리즘을 활용할 수 있는 시스템 모델을 제안한다.

논문의 구성은 다음과 같다. 제2장에서는 제안하는 시스템 모델에 대하여 기술하고 제3장에서는 실험 및 평가 결과를 보인다. 제4장에서는 결론 및 향후 연구에 대해 제시한다.

2. 제안 모델

2.1 접근 방법

특정 변환 알고리즘을 다양한 저장소에 활용하기 위한 접근 방법은 크게 두 가지로 분류할 수 있다. 첫 번째 방법은 저장소에 종속적인 접근 방법으로서, 저장 구조 혹은 변환 알고리즘을 변경하는 경우이다. 그러나 이 접근 방법은 변환 알고리즘 개발에 따른 비용 문제 및 저장 구조 변경에 따른 많은 문제점을 야기한다.

두 번째 접근 방법은 물리적 공간인 저장소와 변환 알고리즘을 독립적으로 유지시키는 것이다. 이 접근 방법은 저장소의 구조가 변경되어도 변환 알고리즘은 수정할 필요가 없으며, 알고리즘을 여러 다른 구조의 저장소에 독립적으로 활용할 수 있기 때문에 높은 실용성을 제공한다.

이 논문의 접근 방법은 두 번째 방법으로서, 뷰를 기반으로 한 시스템 모델을 제안한다. 즉, 가상테이블인 뷰를 저장소와 변환 알고리즘 사이에 위치시킴으로써 상호간 독립성을 보장한다. 제안 방법은 뷰의 저장 구조를 이용하고자 하는 변환 알고리즘에 따라 변환해야 하는 오버헤드를 지닌다. 그러나 물리적인 구조를 변경하지 않음으로써 독립성은 물론 저장소 고유의 특성, 즉 장점을 그대로 활용할 수 있다.

2.2 제안 모델 구조

그림 1은 각 저장소에 종속적인 변환 알고리즘이 활용된 프레임워크를 보여준다. 프레임워크는 SPARQL 계층, Algorithm 계층, SQL 계층, Physical Table 계층 등 총 4개의 계층으로 구성 되어있다. SPARQL 계층은 검색하고자 하는 데이타에 대한 SPARQL이고 Algorithm 계층은 각 저장소에 종속적으로 개발된 SPARQL-to-SQL 변환 알고리즘이다. SQL 계층은 변환 알고리즘을 통해 생성된 SQL이다. 마지막으로 Physical Table 계층은 실제 웹 온톨로지 데이타 값들이 저장되어있는 물리적 테이블로 구성되어있다.

사용자는 SPARQL 계층에서 검색하고자 하는 데이타에 대한 SPARQL을 입력한다. Algorithm 계층의 변환 알고리즘은 Physical Table 계층의 저장소 구조에 따라 서로 다른 형태를 지닌다. SQL 계층에서는 Algorithm 계층에서 적용하는 알고리즘에 따라서 각각의 다른 형태의 SQL문이 생성된다. 이 SQL문은 Physical 계층을 대상으로 연산을 실행한다. Physical Table 계층은 각 저장소의 물리적인 테이블 구조의 집합으로 서로 다른 저장 구조를 지닌다.

그림 1을 보면, 알고리즘1이 기본 알고리즘으로 트리플 저장소를 사용한다. 그러나 저장소 구조가 Jena 또는 Sesame로 변경될 경우 알고리즘1 또한 알고리즘 2,3으로 수정되어야 한다. 따라서 시스템들이 동일한 데이타를 저장하고 있다고 해도 이를 검색하기 위한 SQL 문은 각 저장소에 따라 다르게 작성된다. 이와 같이 알고리즘을 변경할 경우, 개발 비용과 변경된 알고리즘에 대한 검증 비용 등이 요구된다.

특정 변환 알고리즘을 변경하지 않고 이용한다면, 변환 알고리즘에서 전제하는 저장 구조에 맞게 각 저장소의 구조를 변경해야 한다. 이 또한 각 저장소의 특성을 약화시키거나 손실하는 문제점과 함께 저장소와 관련된 다양한 모듈 (예를 들어, 추론 엔진, 파서)과의 연동이 불가능하게 되는 문제점을 초래한다.

그림 2는 제안 모델을 위한 프레임워크를 보여준다. 그림 2에서 알 수 있듯이, 제안 프레임워크는 그림 1과 동일한 예제를 포함하는 매우 유사한 구조이다. 상호 다른 물리적 테이블 저장 구조를 지니는 저장 시스템인 Triples, Jena, Sesame가 최하위 계층에 존재한다. 그러나 그림 1의 기본 구조를 기반으로 SQL 계층과 Physical Table 계층 사이의 View 계층을 추가로 정의한다. View 계층은 물리적 테이블 저장 구조와 변환 알

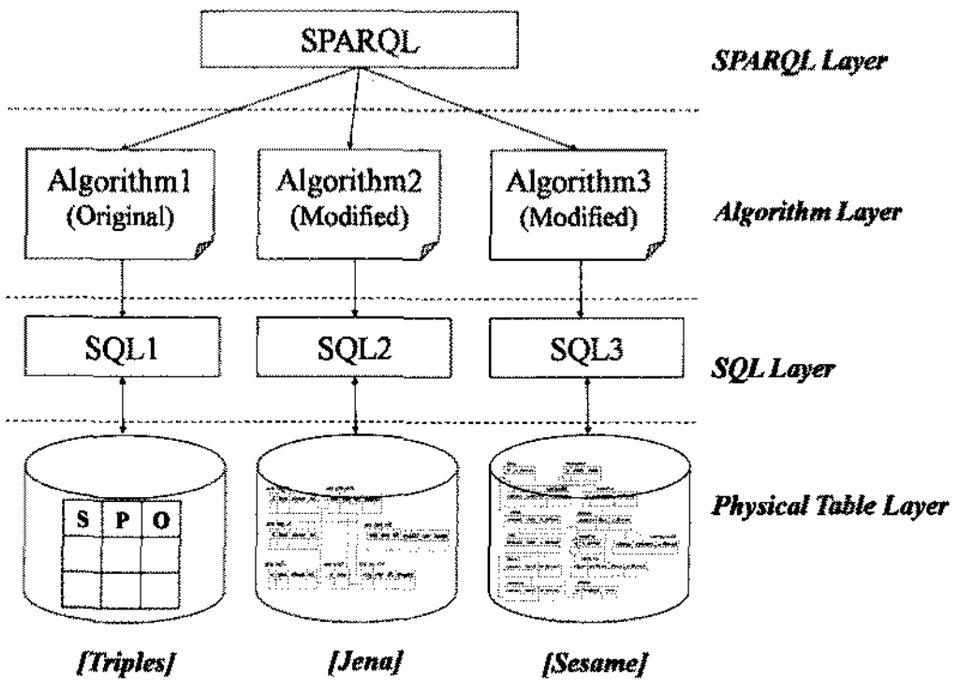

그림 1 저장소에 종속적인 프레임워크

고리즘을 독립적으로 유지시켜준다.

제안 프레임워크를 사용할 경우 사용자는 SPARQL 계층을 통해 검색하고자 하는 데이타를 질의하는 SPARQL을 입력한다. Algorithm 계층에는 Chebotko 알고리즘과 같은 변환 알고리즘들로 구성된다. 이 알고리즘들은 사용자의 입력으로 주어진 SPARQL을 SQL로 변환하는 역할을 담당한다. 이 계층에 임의의 알고리즘이 주어졌을 때, 변환된 SQL 질의문은 다수의 저장소에 전달된다. 이 때, 특정 저장소는 주어진 알고리즘에서 전제한 저장 구조와 동일할 수 있으나 대부분은 다른 저장 구조를 지니게 된다. SQL 계층은 변환 및 전달된 SQL 질의문을 각 저장소에 전달하는 역할을 수행한다. 알고리즘에서 전제한 저장 구조와 동일한 저장 구조를 지닌 저장소인 경우에는 테이블을 직접 액세스하도록 SQL 질의문을 전달하게 되며, 동일하지 않은 저장 구조를 지닌 저장소에는 뷰를 대상으로 질의문을 전달하게 된다. View 계층은 알고리즘과 다른 저장 구조를 지니는 저장소를 위해 생성된 뷰들의 집합이다. 따라서 알고리즘과 상이한 저장 구조를 지닌 저장소에 대한 검색은 View 계층을 대상으로 연산이 수행된다. 마지막으로, Physical Table 계층은 실제 웹 온톨로지 데이타 값들이 저장되어 있는 물리적 테이블의 집합이며 알고리즘과 동일한 저장 구조를 지니는 저장소에 대한 검색은 이 계층을 대상으로 연산이 수행된다.

그림 2에서, 변환 알고리즘과 동일한 구조를 지니는 Triples 저장소를 제외한 저장소는 View 계층에서 트리플 구조의 뷰를 생성한다. 뷰가 생성되면 변환 알고리즘에 의해 생성된 동일한 하나의 SQL 질의문을 이용하여 뷰를 대상으로 검색 연산이 수행된다.

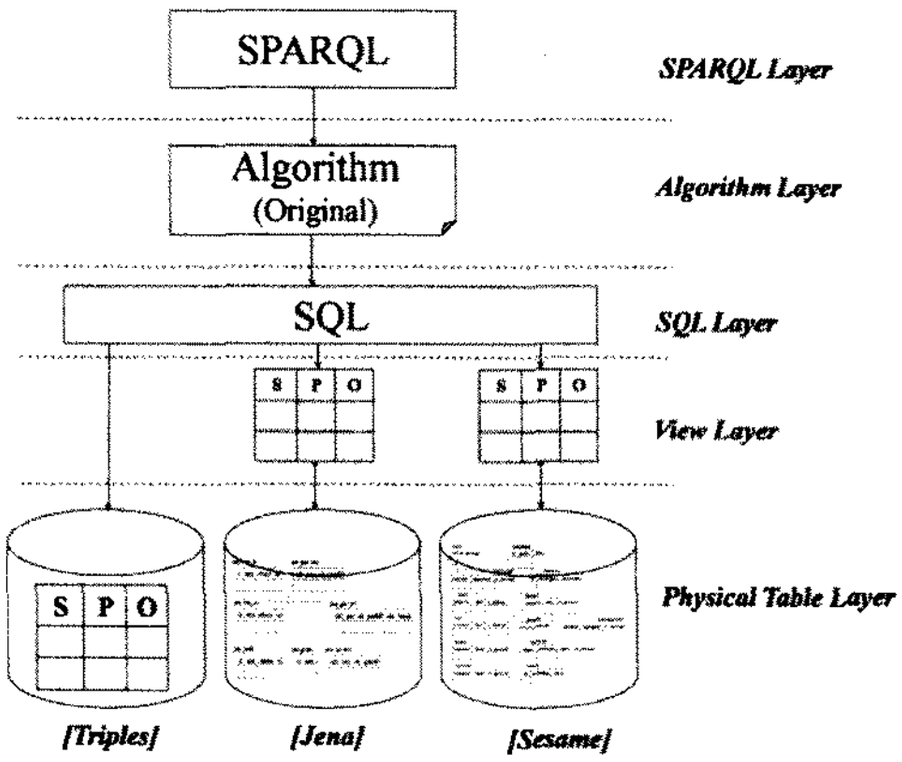

그림 2 제안하는 프레임워크

2.3 대상 저장소를 위한 뷰 생성

View 계층에 생성된 뷰는 각 시스템의 저장소 구조가 다를지라도 주어진 알고리즘이 사용하는 저장소와 모두 같은 구조로 생성된다. Chebotko 알고리즘을 사용하는 제안 모델에서는 구현 및 실험평가로 Triples 모델과 Jena 모델을 사용한다.

Chebotko 알고리즘은 트리플 구조를 저장소로 사용하므로 뷰는 모두 트리플 구조이다. 그러나 Triples 모델의 경우 저장 구조가 이미 트리플 구조이므로 뷰를 생성해 줄 필요가 없다.

반면, Jena 모델의 경우에는 트리플 구조가 아니므로 트리플 구조의 뷰를 생성해야 한다. 생성된 뷰는 SUBJECT, PREDICATE, OBJECT로 구성된다.

Jena 저장소를 구성하는 총 7개의 테이블 중에서, 실질적으로 데이타를 저장하는 테이블은 jena_gntn_stmt 이다. 그러나 데이타의 길이가 256바이트가 넘을 경우, Literal이 데이타 타입인 데이타는 jena_long_lit 테이블에 저장되고 URI는 jena_long_uri 테이블에 데이타가 저장된다. 질의 결과로 추출해야 할 컬럼들은 jena_gntn_stmt 테이블의 SUBJ, PROP, OBJ이다. 그러나 jena_long_lit, jena_long_uri 테이블의 경우에는 데이타를 추출할 때 데이타의 길이가 256바이트가 넘을 경우에만 추출한다.

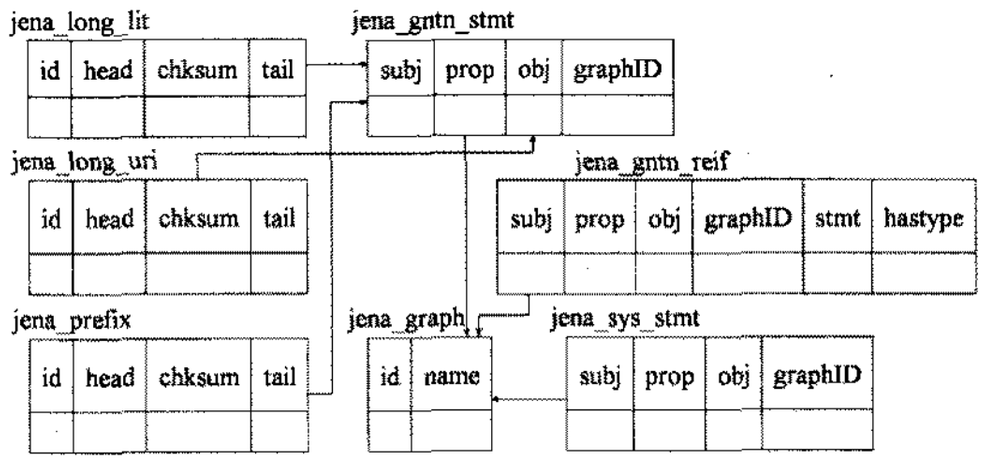

그림 3 Jena 저장 구조

3. 실험 및 평가

3.1 실험 환경

실험을 위한 프로토타입 구현 환경은 다음과 같다. (1) CPU: Pentium 4 (3.00 GHz), (2) 메모리 사이즈 (RAM): 0.99GB, (3) 하드디스크 용량: 104GB, (4) 플랫폼: Windows XP Professional OS, (5) 언어: Java JDK 1.6.0, (6) DBMS: Oracle 9i Enterprise Edition Release 9.2.0.1.0, (7) 실험 저장소: Triples 저장소와 Jena 저장소.

실험 목적은 제안 모델을 바탕으로 동일한 데이타 셋과 질의문을 사용하여 Triples 모델과 트리플 구조의 뷰를 생성한 Jena 모델 간에 질의 결과가 동일하게 나타나는지를 측정하는 것이다.

실험을 위한 데이타 셋으로는 Lehigh 대학에서 제공되는 UBA (Univ-Bench Artificial Data Generator)를 이

표 1 실험을 위한 질의 정의

질의	질의 설명
Q1	University0에서 GraduateCourse0를 가르치는 교수
Q2	University0에서 FullProfessor0 교수와 Lv:0::FullProfessor0@Department1.University0.edu와의 관계
Q3	University0의 FullProfessor0이 가르치는 과목
Q4	Lv:0::Research1에 대한 정보
Q5	University0에서 지도 관계를 가지는 교수와 학생
Q6	University0 에 대한 관계와 정보
Q7	GraduateStudent이면서 GraduateCourse0을 수강하는(takesCourse)하는 학생

용하여 생성한 OWL 데이타 셋을 사용한다[15]. OWL 데이타 셋은 대학, 교수, 학생 등의 정보로 구성되어 있다.

표 1은 실험을 위해 OWL 데이타 셋을 기반으로 7개의 질의문을 정의하였다. 이 질의문은 조인연산이 포함된 질의문과 조인연산이 포함되지 않은 질의문으로 분류할 수 있다.

표 2는 표 1에서의 질의를 실제 SPARQL 질의문으로 정의한 예를 보여준다. 이 질의문은 조건절의 개수로 크게 두 가지로 나누어 진다. Q1, Q2, Q3, Q4, Q5, Q6은 WHERE절에서 조인연산을 하지 않는다. Q1, Q2, Q3은 한 개의 변수를 지니고 Q4, Q5, Q6은 두 개의 변수를 지닌다. Q7은 ?x를 공통변수로 사용하고 있으므로 그에 따른 자연조인 연산을 수행한다. 제안한 시스템 모델이 정확한 변환 알고리즘을 수행하는지 평가하기 위해 변수의 개수를 다르게 정의해 준다.

3.2 실험 결과

이 절에서는 표 2에서 정의한 7개의 질의문을 실험 평가한다. 그림 4는 구현한 프로토타입의 실행 화면을 보여준다. 실험은 Query List에서 총 7개의 질의문 중 하나를 선택한다. Query문에서 선택한 질의문은 Original SPARQL Query에서 SPARQL 질의문 형태로 출력된다. Translation버튼을 클릭하면 SPARQL 질의문은 구현된 알고리즘에 의해서 SQL문으로 변환된다. 변환된 SQL 질의문은 Translated SQL에서 출력된다. 실험에서 사용한 저장소는 트리플 구조와 Jena 구조이므로 Triples ExecuteQuery 버튼을 클릭할 경우 트리플 구조에서 SQL 질의문에 해당하는 결과 값을 Query results의 Triples에 출력하게 되고 Jena ExecuteQuery 버튼을 클릭하면 미리 생성해준 트리플 구조의 뷰에 의해서 질의문의 결과값을 Query results의 Jena에 출력하게 된다.

그림 4는 실험을 위해 구현한 프로토타입을 이용하여 Q6에 대한 각 저장소로부터 반환된 질의 결과를 보여준다.

표 3은 논문에서 정의한 7개 질의문을 이용하여 각 저장 시스템을 대상으로 실험한 결과이다. 표 3의 질의 결과를 통하여 Triples 모델과 Jena 모델이 동일한 개수의 결과 값을 지님을 확인할 수 있다. 결론적으로 서로 다른 구조를 가진 두 저장소이지만 Jena 모델에 트리플 구조의 뷰를 생성함으로써 두 저장소가 하나의 질의문을 실행하여 동일한 결과를 나타난다는 것을 알 수 있다. 따라서, Jena 모델뿐만 아니라 다른 모델의 저장소에도 트리플 구조의 뷰를 생성한다면 단일 질의문으로 동일한 질의 결과를 추출해 낼 수 있다.

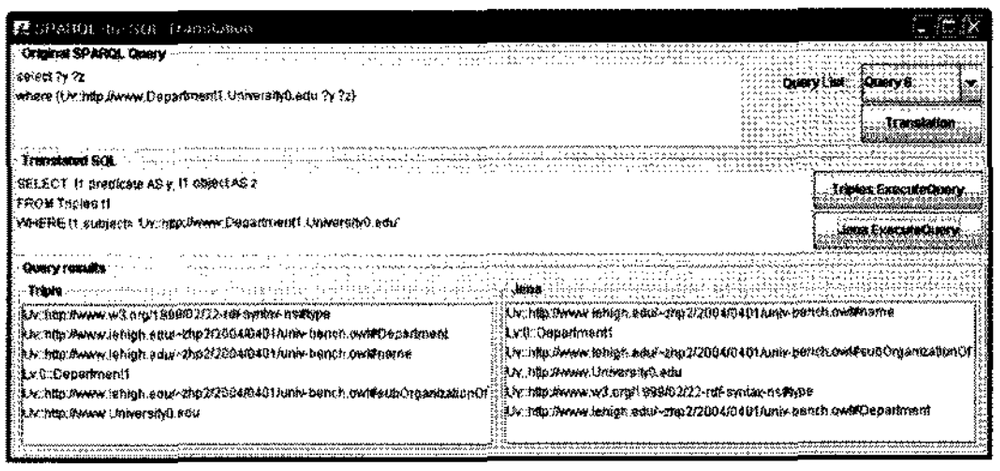

그림 4 질의 6에 대한 실험결과

표 2 실험을 위해 정의한 SPARQL 질의문

BGP 개수	SPARQL 질의문	변수 개수	조인 여부
1	**Q1** : select ?x where {?x Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#teacherOf Uv::http://www.Department1.University0.edu/GraduateCourse0}	1	0
	Q2 : select ?y where {Uv::http://www.Department1.University0.edu/FullProfessor0 ?y Lv:0::FullProfessor0@Department1.University0.edu}	1	
	Q3 : select ?z where {Uv::http://www.Department1.University0.edu/FullProfessor0 Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#teacherOf ?z }	1	
	Q4 : select ?x ?y where {?x ?y Lv:0::Research1}	2	
	Q5 : select ?x ?z where {?x Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#advisor ?z}	2	
	Q6 : select ?y ?z where {Uv::http://www.Department1.University0.edu ?y ?z}	2	
2	**Q7** : select ?x where {?x Uv::http://www.w3.org/1999/02/22-rdf-syntax-ns#type Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#GraduateStudent ?x Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#takesCourse Uv::http://www.Department1.University0.edu/GraduateCourse0 }	1	1

표 3 각 질의에 따른 저장모델들의 결과

질 의	질의 결과 (개수)	
	TRIPLES	Jena
Q1	1개	1개
Q2	1개	1개
Q3	4개	4개
Q4	38개	38개
Q5	5692개	5692개
Q6	6개	6개
Q7	4개	4개

표 4 비교 평가

비교항목	종속적 모델	제안 모델
개발 비용 (수정)	높음	낮음
검증 비용	높음	낮음
저장소 특성	손실	유지
관련 모듈과의 연동	불가능/가능	가능
활용성	낮음	높다
실용성	낮음	높다

표 4에서는 제안 모델과 저장소 또는 알고리즘에 종속적인 모델을 정성적으로 비교 평가한다. 저장소에 종속적인 모델은 저장소가 변경될 경우, 이용하고자 하는 저장소에 맞도록 알고리즘의 수정이 요구된다. 또한 구조에 따른 알고리즘의 수정은 정확성 검증에 필요한 추가적 시간과 비용을 요구한다. 변환 알고리즘의 변경 없이 저장소 구조를 변경할 경우, 저장소가 지니는 고유의 특성 (장점)을 잃게 되며 또한 편집, 파싱 및 추론 등과 같은 관련 모듈과의 연동이 어렵게 된다. 따라서 종속적 모델은 알고리즘의 활용성과 실용성이 낮고 추가적인 시간과 많은 비용을 요구한다.

반면, 제안 모델은 알고리즘과 저장소가 독립적이기 때문에 알고리즘의 수정 없이 다른 구조를 지닌 저장소에 적용할 수 있으므로 활용성과 실용성이 높고 추가적 시간과 비용이 적다.

4. 결론 및 향후 연구

이 논문에서는 다양한 저장소에 특정 변환 알고리즘을 독립적으로 활용하기 위해서 시스템 모델을 제안한다. 제안 모델은 저장소와 변환 알고리즘 간 독립성을 유지하기 위해 그 사이에 뷰를 생성한다. 또한 기존 알고리즘 중에서 기능성이 가장 높다고 판단한 Chebotko 알고리즘을 변환 알고리즘으로 선택하여 프로토 타입을 구현하였다.

구현된 프로토타입을 이용하여 제안 모델의 정확성 검증을 위해 실험을 하였고 그 결과 및 비교 평가에 대하여 기술하였다. 결과적으로, 제안 모델은 종속적 접근 방법에 비해 효율성, 활용성 및 실용성 측면에서 보다 나은 접근 방법임을 알 수 있다.

향후 연구로서, 사용자가 저장소를 위한 뷰를 보다 용이하게 생성할 수 있는 모듈 개발이 요구된다. 이는 이 논문에서 제안하는 모델의 활용성 및 사용자 관리의 용이성을 보다 향상시킬 수 있다. 추가적으로, 더 높은 기능성을 위하여 SPARQL의 UNION과 FILTER 구문 변환에 대한 알고리즘 개발이 요구된다.

참 고 문 헌

[1] Tim Berners-Lee, James Hendler, and Ora Lassila, "The Semantic Web," Scientific American, Vol. 284, No. 5, pp. 34-43, May 2001.

[2] The RDF Query Language (RQL), http://139.91.183.30:9090/RDF/RQL/.

[3] RDQL-A Query Language for RDF, W3C, 9 January 2004.

[4] SPARQL Query Language for RDF, W3C, 4 October 2006.

[5] Sesame: RDF schema querying and storage, http://www.openrdf.org/.

[6] Zhengxiang Pan, Jeff Heflin, "DLDB: Extending relational databases to support Semantic Web queries," Workshop on Practical and Scalable Semantic Web Systems, ISWC 2003, pp. 109-113, 2003.

[7] Steve Harris, "SPARQL query processing with conventional relational database systems," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 3807, pp. 235-244, 2005.

[8] OWLJessKB: A Semantic Web Reasoning Tool, http://edge.cs.drexel.edu/assemblies/.

[9] Jena Semantic Web Framework, http://jena.sourceforge.net/.

[10] Dongwon Jeong, Myounghoi Choi, Yang-Seung Jeon, Youn-Hee Han, Laurence T. Yang, Young-Sik Jeong, and Sung-Kook Han, "Persistent Storage System for Efficient Management of OWL Web Ontology," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 4611, pp. 1089-1097, July 2007.

[11] Artem Chebotko, Shiyong Lu, Hasan M. Jamil, and Farshad Fotouhi, "Semantics Preserving SPARQL-to-SQL Query Translation for Optional Graph Patterns," Technical Report TR-DB- 052006-CLJF, May 2006, Revised, November 2006.

[12] sparql2sql - a query engine for SPARQL over Jena triple stores, http://jena.sourceforge.net/.

[13] Yannis Theoharis, Vassilis Christophides, and Grigoris Karvounarakis, "Benchmarking database representations of RDF/S stores," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 3729, pp. 685-701, 2005.

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

[15] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin, "LUBM: A Benchmark for OWL Knowledge Base Systems," Journal of Web Semantics, Vol. 3, No. 2, pp. 158-182, July 2005.