확장 클래스-속성 뷰기반의 SPARQL-SQL 질의 변환 및 속도 개선

SPARQL-SQL Conversion and Improvement in Response Time based on Expanded Class-Property Views

이승우, 김평, 김재한, 성원경
한국과학기술정보연구원 정보서비스연구팀

Seungwoo Lee, Pyung Kim, Jaehan Kim, Won-Kyung Sung
ISRLab, KISTI

요약

데이터베이스관리시스템(DBMS)이 대용량의 트리플 형식의 지식을 저장하기 위한 도구로 사용되는 것이 일반적인 추세인 상황에서, 보다 효율적으로 트리플 형식의 지식을 저장/관리/추론/질의하기 위해 DBMS에 어떠한 형태로 스키마를 설계하느냐는 여전히 이슈로 남아 있다. 본 논문에서는 효율적인 질의 관점에서 확장 클래스-속성 뷰(ECPV)를 활용하여 질의를 처리하는 방법과 이로 인해 얻어지는 응답 속도의 개선을 소개한다. DBMS기반의 추론 엔진의 응답 시간은 결국 테이블의 크기와 조인 횟수에 비례하게 되며, 질의가 복잡할수록 필요한 조인 횟수도 늘어나므로 응답 시간도 증가하게 된다. ECPV는 바로 조인 횟수를 줄이기 위해 미리 조인 연산을 수행해 둔 것으로, 질의 과정에서 이를 활용하기 위해서는 SPARQL 질의를 ECPV를 사용하는 SQL 질의로 변환해줘야 한다. 본 논문은 이러한 변환 과정과 함께 실험을 통해 응답 속도의 개선 정도를 제시한다.

Abstract

In a general tendency that DBMS is used as a tool for storing large size of triple knowledge, it still remains in issue that which DBMS schema should be designed for storing, managing, inferring, and querying the triple knowledge efficiently. In this paper, we present, in the view point of efficient query process, a method that processes a query using Expanded Class-Property Views (ECPV) and, as a result, improvement in response time. The response time of DBMS-based inference systems is proportioned to table size and the number of table join operations. The more query is complex, the more join operations it requires, and the longer response time it requires. ECPV is a table obtained by processing possible join operations before queries. To use ECPV in the query process, SPARQL queries should be converted into corresponding ECPV-based SQL queries. This paper describes the conversion process and shows the improvement in response time by experiments.

I. 서 론

실용적인 추론 응용 서비스가 되려면, 대용량의 트리플 지식을 대상으로 적절한 저장 공간을 사용하면서 만족스러운 응답 속도를 보일 수 있어야 한다. 그런 점에서 트리플 지식에 대한 효과적인 저장구조가 요구된다. 최근에는 데이터베이스관리시스템(DBMS)이 대용량의 트리플 형식의 지식을 저장하기 위한 도구로 사용되는 것이 일반적인 추세다. 이런 상황에서 추론 엔진의 응답 시간은 결국 트리플을 저장하기 위한 DB 스키마에 영향을 받게 되며, 특히 테이블의 크기와 테이블의 조인 횟수에 비례하게 된다. 더욱이 질의가 복잡할수록 필요한 조인 횟수도 늘어나므로 응답 시간도 증가하게 된다. 따라서 조인의 대상이 되는 테이블 하나의 크기를 줄이고, 또한 질의 처리에 필요한 조인 횟수를 줄일 수 있도록 DB 스키마를 설계하는 것이 중요하다.

전통적인 DBMS 시스템에서는 데이터를 해당 응용에 특화시켜 잘 짜인 스키마 형태로 저장하여 변경이나 확장에 있어서 유연성이 결여되는 반면 추론 시스템에서는 지식을 트리플 형태의 작은 단위로 분산시켜 저장하여 응용에 따라 단편적인 지식을 조합하여 새로운 지식으로 확장하기 쉽도록 하는 유연함을 제공한다. 그러나 이러한 추론 시스템의 유연함은 장점이기도 하지만, 질의를 처리하는 과정에서 매번 단편적인 지식들을 조합해야 함으로써 발생하는 응답 속도 문제는 단점이기도 하다. 이를 극복하기 위한 한 가지 방법으로 [4]에서는 확장 클래스-속성 뷰(Expanded Class-Property View; ECPV)를 제안하였다. [4]에서는 트리플을 저장하는 기본 구조는 스키마-인지(schema-aware)[1] 방식을 따르고 질의에 대한 응답을 처리하는 과정에서 요구되는 조인의 횟수를 줄이기 위해 ECPV를 보조적인 저장 구조로 사용한다. 또한, 온톨로지 스

키마로부터 자동으로 ECPV를 생성하는 방법을 설명하였다. 본 논문에서는 ECPV를 기반으로 SPARQL 질의를 SQL 질의로 변환하는 과정을 설명하고, 실험을 통해 실제로 응답속도가 어느 정도 향상되는지를 살펴보고자 한다.

▶▶ 그림 1. 국가과학기술 R&D 기반정보 온톨로지 (일부)

Ⅱ. 확장 클래스-속성 뷰

확장 클래스-속성 뷰(ECPV)는, 기존의 트리플 형태의 지식은 주된 저장 구조로서 유지하면서 질의 처리를 보다 효율적으로 수행하기 위한 보조적인 저장 구조로서, 특정 클래스로부터 직·간접적으로 관계를 갖는 속성 및 속성 값의 쌍들로 구성된다[4]. 예를 들어, 국가과학기술 R&D 기반정보 온톨로지[2]에서 Person 클래스를 살펴보자. 그림 1에서 볼드체로 표시된 속성(property)들은 Functional Property 혹은 Inverse Functional Property[1]인 경우이다. ECPV는 바로 이런 유형의 속성들을 대상으로 하며 Person 클래스에 대해 생성된 ECPV는 그림 2와 같은 구조를 갖는다. 사람의 소속기관은 하나로 가정하고 있기 때문에 소속기관(Institution 클래스)에 대한 정보를 Person 클래스의 ECPV를 통해, 조인 과정 없이, 바로 접근할 수 있게 된다.

subj	c1	c2	c3	c4	c5	c6	c7	c8
	engNameOfPerson	*hasInstitutionOfPerson*	*hasInstitutionOfPerson_engNameOfInstitution*	*hasInstitutionOfPerson_korNameOfInstitution*	*hasInstitutionOfPerson_latitudeOfInstitution*	*hasInstitutionOfPerson_longitudeOfInstitution*	*korNameOfPerson*	*INVstandForSameAsGroupOf*

▶▶ 그림 2. Person 클래스에 대한 ECPV 스키마

1) *standForSameAsGroupOf*가 Inverse Functional Property에 해당한다.

Ⅲ. SPARQL-SQL 변환

SPARQL은 온톨로지 질의언어 중의 하나이다. 온톨로지의 트리플 지식이 내부적으로는 DBMS에 저장되기 때문에 SPARQL을 지원하기 위해서는 SPARQL 질의를 SQL로 변환하는 과정을 거쳐야 한다. 트리플이 저장되는 내부 구조가 스키마-무관(schema-oblivious) 방식인 경우의 SPARQL-SQL 변환에 대해서는 [3]에 설명되어 있다. 현재 OntoFrame®[6]은 트리플의 내부 저장구조로 스키마-인지(schema-aware) 방식을 따르는데, 이 경우에도 [3]의 변환 과정은 동일하게 적용될 수 있다. 다만, 속성(property)에 따라 참조하는 테이블이 달라진다는 점이 차이점이다. 즉, 스키마-무관 방식에서는 속성에 상관없이 모든 트리플이 하나의 테이블에 저장되는 반면, 스키마-인지 방식에서는 속성별로 테이블을 나눠서 저장하기 때문에 테이블을 참조할 때도 속성별 테이블을 참조해야 하는 것이다.

ECPV를 기반으로 하는 SPARQL-SQL 변환을 설명하기에 앞서, ECPV를 사용하지 않는 경우의 SPARQL-SQL 변환을 예를 들어 살펴보자. OntoFrame®에서 제공하는 주제별 연구전문가 분석 서비스는 특정 주제(topic)를 연구하는 연구자(?person)와 그의 소속기관(?institution) 정보를 얻고, 후처리 과정에서 해당 주제에 대한 연구 성과물 건수를 기준으로 전문도를 계산하여 어떤 연구자가 해당 주제에 대한 전문가인지를 분석하는 서비스이다. 여기서 특정 주제에 따른 연구자와 소속기관 정보를 얻기 위한 SPARQL 질의는 [그림 3]과 같이다. 연구자에 대한 이름과 소속기관 정보는 없는 경우에도 결과로 얻을 수 있도록 OPTIONAL 절로 처리되었다. 'standForSameAsGroupOf'는 동일인으로 확인된 사람들 중 대표 URI(?rep)를 가리키는 속성으로, 역시 값이 없는 경우도 있기 때문에 OPTIONAL 절로 처리되었다.

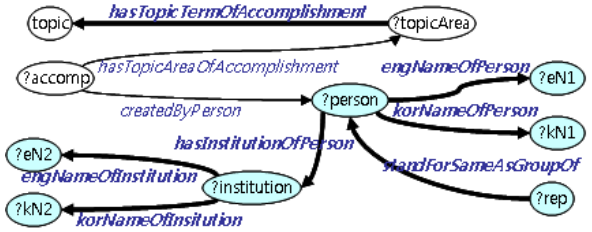
```
SELECT ?person ?rep ?eN1 ?kN1 ?institution ?eN2  ?kN2
WHERE {
?topicArea hasTopicTermOfAccomplishment <topic> .
?accomp hasTopicAreaOfAccomplishment ?topicArea  .
?accomp createdByPerson ?person  .
OPTIONAL { ?rep standForSameAsGroupOf ?person  . }
OPTIONAL { ?person engNameOfPerson ?eN1  . }
OPTIONAL { ?person korNameOfPerson ?kN1  . }
OPTIONAL { ?person hasInstitutionOfPerson ?institution  . }
OPTIONAL { ?institution engNameOfInstitution ?eN2  . }
OPTIONAL { ?institution korNameOfInstitution ?kN2  . }
}
ORDER BY ?eN1 ?kN1
```

▶▶ 그림 3. 주제별 연구전문가 분석을 위한 질의 – SPARQL

```
SELECT T2.obj [person], T3.subj [rep], T4.obj [eN1], T5.obj [kN1],
T6.obj [institution], T7.obj [eN2], T8.obj [kN2]
FROM [hasTopicTermOfAccomplishment] T0
 INNER JOIN [hasTopicAreaOfAccomplishment] T1
        ON T0.subj=T1.obj
 INNER JOIN [createdByPerson] T2 ON T1.subj=T2.subj
 LEFT JOIN [standForSameAsGroupOf] T3 ON T2.obj=T3.obj
 LEFT JOIN [engNameOfPerson] T4 ON T2.obj=T4.subj
 LEFT JOIN [korNameOfPerson] T5 ON T2.obj=T5.subj
 LEFT JOIN [hasInstitutionOfPerson] T6 ON T2.obj=T6.subj
 LEFT JOIN [engNameOfInstitution] T7 ON T6.obj=T7.subj
 LEFT JOIN [korNameOfInstitution] T8 ON T7.subj=T8.subj
WHERE T0.obj='topic'
ORDER BY T4.obj, T5.obj
```

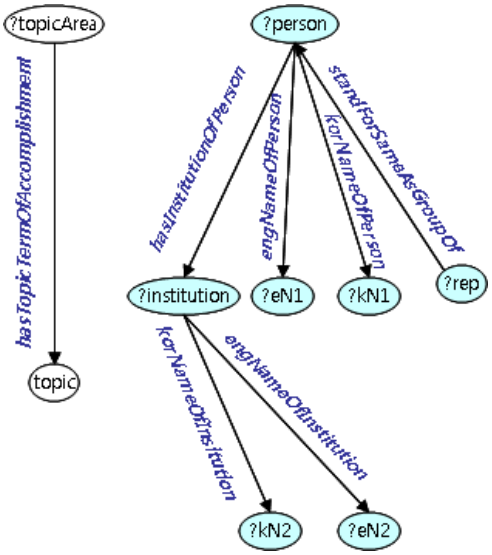
▶▶ 그림 4. 주제별 연구전문가 분석을 위한 질의 – SQL

[그림 3]의 SPARQL 질의를 스키마-인지 방식의 저장구조에 따라 SQL로 변환하면 [그림 4]와 같이 된다. SPARQL의 WHERE절에 나타나는 각 트리플 조건들 하나하나는 SQL에서는 FROM절의 테이블에 대응되고, 트리플 조건 사이에 공유되는 변수는 테이블 사이의 조인조건으로 매핑된다. 테이블 사이의 조인은 기본적으로 내부조인(inner join)이며, OPTIONAL 절인 경우는 왼쪽 조인 (left join)에 대응된다. 트리플 조건에 나타나는 상수(예를 들면, <topic>)나 FILTER 절은 SQL에서는 WHERE 절의 조건으로 변환된다. 이와 같이 변환된 SQL 문은 SPARQL 문에서의 트리플 조건의 수보다 하나 작은 개수의 조인 연산을 요구한다. 트리플 조건의 수가 9개 이므로 [그림 4]에서 8회의 조인이 필요함을 확인할 수 있다.



▶▶ 그림 5. 주제별 연구전문가 분석을 위한 질의 – 그래프 표현

다음으로, SPARQL 질의를 ECPV 기반의 SQL로 변환하는 과정을 설명하기 위해, 우선 [그림 3]의 SPARQL 질의를 그래프 형태로 표현하면 [그림 5]와 같다. 여기서, '?'로 시작하는 노드는 변수를 가리키고, '?'가 없는 노드(예를 들면, topic)는 상수를 가리킨다. 노드와 노드를 연결하는 선은 속성(property 혹은 predicate)을 가리키며 화살표의 시작이 주어(subject)에, 끝이 목적어(object)에 해당한다. 연결선 중에서 굵은 선으로 표시한 것은 Functional Property 혹은 Inverse Functional Property를 가리킨다.



▶▶ 그림 6. 트리 표현

즉, 이들 속성은 ECPV 테이블에 포함될 수 있는 것들이다. 그러면 이제 이 속성들이 어느 ECPV 테이블로 대체될 것인지를 결정해야 한다. 이를 위해, 이 속성들을 트리형태로 표현할 수 있는데, Functional Property의 경우는 주어를 부모로, 목적어를 자식으로 하고, Inverse Functional Property의 경우는 그 반대로 하여 표현하면 [그림 6]과 같이 두 개의 트리를 얻을 수 있다. 각 트리의 루트(root) 노드가 속하는 클래스를 결정하면 그 클래스의 ECPV 테이블이 바로 찾고자 하는 ECPV 테이블이며 해당 트리 전체를 대신할 수 있다. 루트 노드가 속하는 클래스는 결정하는 방법은 온톨로지 스키마에서 루트 노드에 연결된 속성들의 *rdfs:domain* 혹은 *rdfs:range* 값을 읽어오는 것이다. 예를 들어, *hasTopicTermOfAccomplishment*의 *rdfs:domain*은 TopicAreaOfAccomplishment 클래스이고, *standForSameAsGroupOf*의 *rdfs:range*는 Person 클래스이므로 두 트리는 각각 TopicAreaOfAccomplishment 클래스의 ECPV 테이블(ECPV_TopicAreaOfAccomplishment)와 Person 클래스의 ECPV 테이블 (ECPV_Person)로 대체될 수 있다. 다만, 왼쪽 트리의 경우는 후손 노드가 하나뿐이므로 ECPV 테이블을 사용하는 목적인 조인 횟수를 줄이는 효과를 기대할 수 없다. 따라서, 이 경우에는 ECPV 테이블을 사용하지 않고, 트리플 테이블을 그대로 참조하는 것이 바람직하다.

```
SELECT T2.obj [person], E1.c8 [rep], E1.c1 [eN1], E1.c7 [kN1], E1.c2
[institution], E1.c3 [eN2], E1.c4 [kN2]
FROM [hasTopicTermOfAccomplishment] T0
 INNER JOIN [hasTopicAreaOfAccomplishment] T1
         ON T0.subj = T1.obj
 INNER JOIN [createdByPerson] T2 ON T1.subj=T2.subj
 INNER JOIN [ECPV_Person] E1 ON T2.obj=E1.subj
WHERE T0.obj='topic'
ORDER BY E1.c1, E1.c7
```

▶▶ 그림 7. 주제별 연구전문가 분석 – ECPV기반 SQL

이와 같이 대체할 ECPV 테이블을 결정한 후에는 ECPV 테이블에 포함되지 않는 나머지 트리플 조건들은 기존의 SPARQL-SQL 변환에서와 동일하게 내부 조인 혹은 왼쪽 조인으로 변환하고, 여기에 ECPV 테이블을 조인하면 된다. [그림 7]은 최종적으로 얻어지는 ECPV 기반의 SQL을 보여준다. 굵은 글씨체로 표시된 부분이 ECPV 테이블이 대체되어 들어간 부분이다. 그림에서 알 수 있듯이, 변환된 ECPV기반의 SQL에서는 조인 연산이 3회로 줄었음을 알 수 있다.

ECPV 기반의 SQL 변환과정에서 한 가지 주의할 점은 ECPV 테이블은 생성과정에서 subj 컬럼을 중심으로 왼쪽 조인 연산을 수행하여 얻어졌다는 점이다. 따라서, SPARQL 질의에서 OPTIONAL 절로 표현되지 않는 속성이 ECPV 테이블에서 참조되는 경우에는 변환된 SQL 문에서는 WHERE 절에 해당 속성의 값이 NULL이 아니라는 조건을 추가해야 한다.

Ⅳ. 실 험

추론 시스템의 응답 속도에 있어서 확장 클래스-속성 뷰(ECPV)의 효과를 살펴보기 위해 OntoFrame®의 추론 서비스를 대상으로 조사하였다. OntoFrame®은 연구자의 R&D 전 주기 과정을 지원하기 위한 플랫폼[6]으로, IT 분야 시소러스와 과학기술 온톨로지, 추론 시스템을 사용하여 다양한 학술연구정보 분석 서비스[5]를 제공한다.

[5]에 설명된 서비스 중 ECPV가 적용될 수 있는 21개의 서비스에 대해 ECPV를 적용한 경우(w/)와 그렇지 않은 경우(w/o)를 비교 실험한 결과를 [표 1]에 정리하였다. '응답 건수' 열은 각 서비스에 대해 실험에 사용된 파라미터에 의해 결과로 얻어지는 레코드의 수를 가리킨다. 조인 횟수에서 괄호 안의 숫자는 실제 테이블에 레코드가 없는 경우의 조인을 제외한 횟수를 가리킨다. 예를 들어, 실험에 사용된 지식 데이터는 사람에 대해 한글 이름을 갖지 않기 때문에 *korNameOfPerson* 속성은 실제로 거의 값을 갖지 않는데, 이런 경우를 제외한 횟수이다. 즉, 응답 속도에 영향을 미치는 조인 횟수는 괄호 안의 숫자라고 볼 수 있다. 각 서비스의 응답 시간은 각 서비스를 5회 수행하여 평균을 낸 값으로 밀리초(ms) 단위로 표시하였고, 마지막 열은 ECPV를 적용한 경우(w/)에서 ECPV를 적용하지 않은 경우(w/o)를 뺀 값을 가리킨다.

[표 1]에서 알 수 있듯이, 21개 서비스 중 13개의 경우에 ECPV에 의한 응답속도 개선을 확인할 수 있었고, 평균적으로도 약 506ms 정도의 속도 개선이 있었다. 그러나, ECPV를 적용함으로써 조인 횟수가 줄어듦에도 불구하고 응답속도가 더 느려진 경우도 꽤 눈에 띄는데, 그중 15, 16, 17, 20번 서비스의 경우는 응답 건수가 비교적 적어서 조인 횟수가 응답속도에 영향을 적게 미침으로 하여 ECPV에 의한 효과가 나타나지 않은 것으로 판단된다. 또한 2, 8, 10번 서비스는 ECPV에 의한 실질적인 조인 횟수 감소가 없었던 경우이다. 나머지 9번 서비스의 경우는 원인을 좀 더 분석할 필요가 있다.

[표 1] ECPV의 효과

서비스 번호	응답 건수	조인 횟수		응답시간(ms)		
		w/o	w/	w/o	w/	차이
1	41	5(3)	3	675	622	**-53**
2	356	10(6)	6	7222	8613	1391
3	63	6(5)	3	297	125	**-172**
4	100	6(5)	4	1750	1213	**-537**
5	1067	8(5)	3	2106	544	**-1562**
6	247	10(7)	6	797	587	**-210**
7	285	14(10)	6	8106	6197	**-1909**
8	585	14(10)	10	5097	6447	1350
9	863	14(10)	6	4422	5081	659
10	3354	14(10)	10	3591	3909	318
11	585	6(5)	4	1337	375	**-962**
12	1236	9(7)	3	3125	1231	**-1894**
13	247	6(5)	3	719	209	**-510**
14	57	6(5)	3	1809	125	**-1684**
15	9	11(7)	2	87	150	63
16	7	11(7)	2	72	109	37
17	20	5(4)	2	56	63	7
18	38979	13(10)	6	18809	14388	**-4421**
19	252	8(5)	3	166	138	**-28**
20	107	8(5)	3	131	141	10
21	100	5(4)	2	1559	1041	**-518**
평균	-	9(6.4)	4.2	2949	2443	**-506**

Ⅴ. 결론 및 향후 연구

지금까지 DBMS 기반의 추론 시스템의 응답 속도 향상을 위해 조인 연산에 드는 비용을 줄이는 ECPV 기반의 SPARQL-SQL 변환 과정을 설명하였다. 또한 이를 추론 서비스에 실제로 적용하였을 때, 다수의 경우에 응답속도가 개선

됨을 실험을 통해 확인할 수 있었다. 하지만, 일부 서비스에 대해서는 오히려 응답속도가 느려지는 경우도 발생하였는데, 이를 개선하기 위해서는 좀 더 세밀한 ECPV의 적용을 고려해야 할 것이다.

▌참 고 문 헌▐

[1] Theoharis, Y., V. Christophides and G. Karvounarakis, "Benchmarking Database Representations of RDF/S Stores," in Proceedings of ISWC2005 (LNCS3729), pp .685-701, 2005.

[2] 강인수, 정한민, 이승우, 김평, 성원경, "국가과학기술 R&D 기반 정보 온톨로지와 추론 모델링", 2006 한국컴퓨터종합학술대회 발표논문집, 2006.

[3] 이승우, 정한민, 성원경, "R-DBMS기반 추론 서비스인 OntoThink-K에서의 SPARQL 질의 지원", 2006 한국정보과학회 추계 학술발표논문집 Vol.33, No.2(B), pp.223-227, 2006.

[4] 이승우, 김평, 성원경, "확장 클래스-속성 뷰를 이용한 추론시스템의 응답 속도 개선", 2007 한국콘텐츠학회 춘계종합학술대회 논문집, 2007.

[5] 이승우, 김평, 강인수, 정한민, 이미경, 성원경, "학술연구정보 분석을 위한 OntoFrame 추론 서비스 질의 설계", KOSTI 2007/한국콘텐츠학회 추계종합학술대회논문집, 2007 (게재예정).

[6] 정한민, 이미경, 성원경, 박동인, "OntoFrame-K: 연구자 간 협업 지원 서비스를 위한 시멘틱 웹 기반 정보 유통 플랫폼", 2006 한국컴퓨터종합학술대회 발표논문집, 2006