RDF 데이터 관리를 위한 프로퍼티 기반 분할 저장 모델

김성완° , 임해철*
상육의명대학 컴퓨터정보과, *홍익대학교 컴퓨터공학과
swkim@syu.ac.kr, lim@cs.hongik.ac.kr

Property-based Decomposition Storage Model for RDF Data Management

Sung Wan Kim° and Hae Chull Lim*
Dept. of Computer Information, Sahmyook College, * Dept. of Computer Eng., Hongik Univ.

요 약

시맨틱 웹의 구현을 위한 수단으로 RDF 및 기타 기반 기술이 사용되고 있다. 이에 따라, 방대한 RDF 데이터의 효율적인 관리를 위한 연구들이 최근 활발하게 국내외에서 진행 중이다. 기존의 많은 연구들은 관계형 데이터베이스 시스템을 이용하여 트리플 형태의 RDF 데이터의 저장하는 방법을 제안하였다. 이러한 방법은 하나의 대규모 테이블상에 RDF 데이터를 저장하므로 데이터 관리측면에서 장점이 있으나 질의 처리 측면에서 볼 때 항상 테이블 전체를 접근해야 하므로 검색 성능이 저하될 수 있는 문제점이 있다. 본 논문에서는 질의 처리 성능을 높이기 위해 프로퍼티를 기반으로 RDF 데이터를 여러 개의 테이블로 분할 저장하는 기법을 제안한다.

1. 서론

W3C의 권고안으로 발표된 RDF는 웹 상의 다양한 리소스들에 대한 의미적 정보를 표현하는 언어[1]로써 향후, 시멘틱 웹의 실현을 위해서는 RDF를 사용하여 작성된 대량의 데이타가 생산되고 사용될 것이다. 시멘틱 웹을 기술적인 측면에서 지원하기 위해서는 RDF 데이타를 효율적으로 관리하는 것이 매우 중요하다.

RDF의 관리를 위한 대부분의 연구들은 데이타 관리를 위해 가장 광범위하게 사용되는 강력한 도구인 관계형 데이타베이스 시스템을 이용하고 있다[2][3]. 특히, 트리플 집합으로 표현될 수 RDF 데이터의 저장을 위해 하나의 대규모 테이블을 이용하고 있다. 이러한 방법은 데이터 관리측면에서 장점이 있으나 질의 처리 측면에서 볼 때 항상 테이블 전체를 접근해야 하므로 검색 성능이 저하된다.

본 논문에서는 질의 처리 성능을 높이기 위해 프로퍼티를 기반으로 RDF 데이터를 여러 개의 테이블로 분할 저장하는 기법을 제안한다. 2장에서는 RDF의 개념 및 기존의 RDF 데이터 관리 기법에 대해서 소개한다. 3장에서는 본 논문에서 제안하는 RDF 저장 모델을 제안한다. 4장에서는 질의 처리 및 성능 평가에 대해서 언급하고, 5장에서는 결론 및 향후 연구에 대해 설명한다.

2. RDF와 RDF 데이터의 관리

RDF(Resource Description Framework)는 웹상의 리소스에 대한 시맨틱 정보를 표현하기 위한 언어로, 각 리소스의 서술을 위해 프로퍼티와 그 값을 사용하여 간단한 문장(statement) 형태로 표현한다. 다시 말하면, RDF 문장은 주어, 서술어, 목적어로 구성되며,주어는 리소스의 URI를, 목적어는 프로퍼티 값으로써 리소스의 URI 또는 리터럴 값을 의미하며, 서술어는 프로퍼티를 의미하며 프로퍼티 역시 URI를 갖는 리소스가 된다. 한편, RDF 문장들은 노드와 아크로 구성된 방향성 그래프 형태로 표현될 수 있다. 여기서, 주어와 목적어는 노드로, 서술어는 주어 노드로 부터 목적어 노드로 연결된 방향성 아크로 표현된다. 즉, 서술어는 주어 노드와 목적어 노드간의 관계성을 나타낸다. 또한, RDF 그래프는 RDF 문장을 의미하는 트리플의 집합으로도 표현할 수 있다. 트리플은 <주어, 서술어, 목적어> 구조로 구성되며, RDF 그래프상의 아크에 해당된다.

RDF의 또 다른 특징은 프로퍼티-중심의 모델이라는 것으로 이는 한 리소스가 특정 프로퍼티들과 연관되어야 한다는 제약이 없다는 것이다. 즉, 언제든지 리소스에 대해 임의의 프로퍼티와 값을 연관시키기 위한 RDF 문장을 서술할 수 있다. RDF 스키마는 RDF 문장 기술에 필요한 어휘정의를 위한 언어로서 프로퍼티, 프로퍼티와 리소스간의 관계 등에 대한 정의를 기술하는 기법을 제공한다. RDF 스키마 역시 RDF로 표현된다.

RDF 데이터의 관리를 위한 기존 대부분의 연구는 RDF 그래프를 트리플 집합으로 인식하고 관계형 데이터베이스 시스템의 테이블로 저장한다[2][3]. 기본적인 저장 스키마는 하나의 대규모 트리플 테이블, 리소스 테이블, 리터럴 테이블 등 3개의 기본 테이블외 기타 테이블로 구성된다. 리소스 테이블은 프로퍼티를 포함한 모든 리소스를 저장하며, 리터럴 테이블은 모든 리터럴 값을 저장한다. 트리플 테이블은 주어, 서술어, 목적어 등의 필드로 구성되며, 트리플 형태의 RDF 문장을 저장한다. 또한, 리소스 테이블과 리터럴 테이블을 참조한다. 이러한 방법은 리소스와 리터럴 값이 한번만 저장되고, 단일

의 트리플 테이블을 유지하므로 관리 측면에서 장점을 가진다. 그러나, 질의 처리시 테이블 전체를 접근해야 하며, 테이블간의 조인 연산이 요구되므로 질의 처리 성능이 저하될 수 있다.

한편, 전통적인 관계형 데이터베이스 시스템에서 한 데이터 객체는 테이블의 행으로 표현된다. [4]의 연구에서는 분할 저장 모델을 이용한 데이터 객체 저장 방법을 소개하였다. 이 기법은 전통적인 관계형 데이터베이스의 테이블을 애트리뷰트 수 만큼의 바이너리 테이블들로 분할한다. [5]에서는 분할 저장 기법이 전통적인 테이블 표현보다 우수한 검색 성능을 보임을 실험을 통해 소개하였다. 또한, <객체식별자, 애트리뷰트명, 애트리뷰트값>으로 구성된 3-ary 수직 테이블 구조에 대해 소개하였다. 아래 그림은 관계형 데이터베이스의 수평적 테이블 구조와 3-ary 수직 테이블 구조 그리고 n개의 바이너리 테이블 구조를 비교한 것이다. 3-ary 수직 테이블과 바이너리 테이블에는 null 값이 저장되지 않는 특징을 가지고 있다.

HT

Oid	A1	A2	A3
1	a	b	null
2	null	c	d
3	null	null	a

<그림 1> 수평 테이블

VT

Oid	Key	Val
1	A1	a
1	A2	b
2	A2	c
2	A3	d
3	A3	a

<그림 2> 3-ary 수직 테이블

A1

Oid	val
1	a

A2

Oid	val
1	b
2	c

A3

Oid	val
2	d
3	a

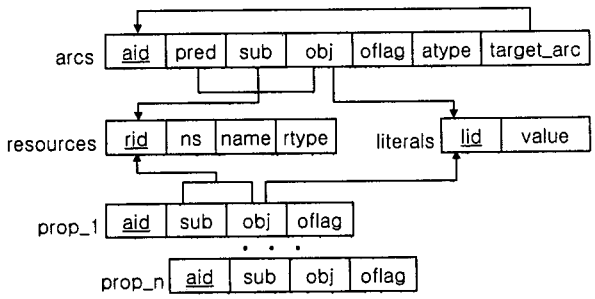
<그림 3> 바이너리 테이블

트리플 집합으로 표현되는 RDF 데이터의 저장을 위해 많이 사용되는 트리플 테이블은 구조는 3-ary 수직 테이블 구조를 응용한 것으로 볼 수 있다.

3. 프로퍼티 기반의 분할 저장 모델

본 논문에서는 다음의 두 가지 개념에 기반하여 RDF 데이터 저장 모델을 설계한다. 첫째, RDF가 프로퍼티-중심의 데이터 모델이라는 것이다. 일반적인 데이터베이스 모델링 과정에서 개체 뿐만 아니라 관계에 대해서도 별도의 독립된 테이블을 생성하여 유지하는 것과 같이 RDF 데이터상의 동일 프로퍼티들에 대해 각각 독립적으로 표현하여 관리한다. 둘째, 이를 위한 실제적인 적용을 위해 [4]에서 소개한 분할 저장 모델의 개념을 이용하여 프로퍼티를 기반으로 분할된 테이블들로 구성된 RDF 데이터 저장 구조를 설계한다. <그림 4>는 전체적인 RDF 데이터 저장을 구조 중 본 논문의 주제와 직접적으로 관련된 몇 가지 중요한 테이블들만을 선별하여 표현한 것이다.

리소스 테이블은 리소스 식별자, 네임스페이스, 리소스 이름, 리소스 타입 필드들로 구성되며 URI를 가지는 모든 리소스들을 저장한다. 아크 테이블은 프로퍼티로 표현된 RDF 문장들을 저장한다. 여기에는 아크 식별자, 프로퍼티, 주어, 목적어, 목적어 구분 플래그, 아크 타입, 그리고 아크 타입이 컨테이너, 컬렉션, 구체화 등일 경우 현재의 아크가 실제로 참조하는 아크 식별자 값을 가지는 타겟아크 필드로 구성된다.



<그림 4> RDF 데이터 저장 구조

프로퍼티 기반의 테이블들은 RDF 그래프상의 동일한 프로퍼티를 기반으로 표현된 아크들 즉, RDF 문장들에 대한 대한 주어와 목적어를 저장한다. 프로퍼티 테이블은 아크 식별자, 주어, 목적어, 목적어가 리소스인지 리터럴인지 구별해주는 플래그 필드를 포함한다.

프로퍼티 기반의 테이블은 RDF 그래프상에 자주 사용되는 프로퍼티들 또는 질의에 자주 사용되는 프로퍼티들에 대해 선택적으로 생성될 수 있다. 일반적으로 특정 도메인에 사용되는 프로퍼티의 개수는 제한적이며 실제 자주 사용되는 프로퍼티의 개수 역시 제한적이다. 참고로, [6]에서는 성명, 메일박스, 홈페이지 URL, 친구들 등 개인 정보 서술을 위한 어휘를 제공하며 최근 가장 많이 사용되는 온톨로지인 FOAF(Friend-of-a-Friend)의 활용에 대한 분석을 하였다. 특히, FOAF 온톨로지 어휘를 사용한 FOAF 문서들을 대상으로 사용 빈도가 높은 프로퍼티를 조사하였다. 5200개의 FOAF 문서 파일과 foaf:Person 클래스에 대한 약 260,000개의 인스턴스들에 대한 분석 결과 foaf:name, foaf:mbox_sha1sum, foaf:homepage, foaf:knows 등 몇 개의 프로퍼티만이 50% 이상의 활용률을 가지는 것으로 분석되었다.

이 처럼 활용 빈도가 높은 프로퍼티들에 대해 선택적으로 각각 프로퍼티 기반의 테이블들을 생성하여 RDF 문장들을 프로퍼티에 따라 분할하여 저장 관리하도록 한다. 상대적으로 중요도가 적은 프로퍼티를 사용한 RDF 문장들은 아크 테이블에 유지한다. 프로퍼티 기반의 테이블에 저장된 RDF 문장은 다른 테이블에는 중복하여

저장하지 않는다. 결국, 특정 프로퍼티와 그 값을 포함하는 리소스를 검색하는 질의 처리에 대해 모든 RDF 문장들을 접근하는 것이 아니라 특정 프로퍼티 테이블만 접근하여 처리하므로 질의 처리 성능의 향상을 얻을 수 있게 된다.

4. 질의 처리 및 성능 평가

트리플 형태로 RDF 데이터를 저장하는 방법에서는 기본적으로 트리플 패턴을 기반으로 구성된 질의 처리를 지원한다. 본 논문에서는 트리플 패턴에 기반 한 기본 질의 처리에 대해서만 고려하기로 한다. 트리플의 각 구성 요소는 변수 혹은 특정 값이 될 수 있으므로 총 8가지의 질의 조합이 가능하다. 8가지의 질의 유형 중 4가지는 특정 프로퍼티가 주어진 경우이다.

단일의 트리플 테이블 저장 구조는 모든 질의에 대해 단일 테이블 전체를 접근하여 처리해야 한다. 제안 기법에서는 프로퍼티가 주어진 경우, 특히 프로퍼티 기반의 테이블을 대상의 질의 처리의 경우 해당 프로퍼티 테이블만을 접근하여 처리하면 된다. 경우에 따라서는 단일 테이블 저장 구조와 마찬가지로 모든 테이블 전체를 접근하여 처리해야 한다.

RDF 데이터의 저장 및 질의 처리 모듈의 구현을 위해 MySQL, PHP, 아파치 웹서버로 구성된 APM_Setup 5 for Win32을 사용하였다. 실험용 데이터 셋으로는 FOAF 온톨로지[7]를 기반으로 FOAF 프로젝트 개발자 모임(http://rdfweb.org)에서 생성한 단일의 RDF 파일을 사용하였다. 실험용 RDF 데이터의 분석 결과 상이한 프로퍼티는 약 50개 정도가 사용되었으며, 앞 절에서 언급한 바와 같이 foaf:name, foaf:mbox_sha1sum, foaf:knows, foaf:thumbnail 등의 프로퍼티가 가장 많이 사용되었으며 전체 사용빈도의 약 50%를 차지하였다. RDF 파서는 CARA 파서[8]를 이용하였으며 실험 데이에서 추출된 트리플 수는 약 10만개이다. 실험 환경은 Pentium III 866MHz상의 256MB 메인 메모리와 윈도우 2000 서버를 사용하였다.

성능 평가를 위해 우선 트리플 저장을 위한 단일 테이블 구조를 가지는 기존의 방법과, <그림 4>와 같이 사용빈도가 높은 4개의 프로퍼티 기반의 테이블과 기타 테이블 1개로 구성된 제안 방법을 구현하여 실험하였다. 공간 사용량 측면에서는 단일 테이블 방법과 제안 방법이 각각 57MB와 약 46MB의 저장 공간을 사용하였다. 제안 방법이 적은 공간 사용량을 보인 것은 4개의 프로퍼티 기반의 테이블에 프로퍼티 필드를 저장하지 않기 때문이다. 검색 시간을 측정하기 위해 다음과 같은 3가지의 트리플 패턴 질의를 테스트 질의로 사용하였다. 첫째, 전체 트리플을 검색하는 경우 (?, ?, ?)와 사용 빈도가 높은 프로퍼티를 질의에 사용한 경우 (?, foaf:name, ?), 그리고 사용 빈도가 낮은 프로퍼티를 질의에 사용한 경우 (?, foaf:givenname, ?)이다. 아래 표는 3가지 테스트 질의에 대해, 최초 검색 시간과 최초 검색 이후에 대한 평균 검색 시간을 각각 나타낸 것이다.

질의	구분	단일 테이블 기법(초)	제안 기법(초)
Q1	최초	41.20	33.25
	평균	30.79	11.98
Q2	최초	33.05	5.60
	평균	1.86	1.20
Q3	최초	33.33	8.10
	평균	0.41	0.19

캐쉬 효과로 인해 두 기법 모두 평균 검색 시간이 최초 검색 시간보다 우수한 것으로 측정되었다. 3가지 질의 유형에 대해서 제안 기법이 우수한 것으로 평가되었으며, 특히, 예상한 바와 같이 프로퍼티가 주어진 경우 매우 우수한 것으로 평가되었다.

5. 결론

시맨틱 웹의 실현을 위해 RDF 데이터의 효율적인 관리가 중요시 되고 있다. 본 논문에서는 프로퍼티를 기반으로 분할된 독립적인 테이블들을 이용한 RDF 저장 모델을 소개하였으며, 간단한 실험을 통해 제안 저장 모델의 우수성을 보였다. 현재, RDF 데이터의 여러 특성을 효율적으로 표현할 수 있는 최적화된 저장 모델의 연구가 진행되고 있으며, 다양한 측면에서 성능 평가를 하고자 한다.

* 본 연구는 한국과학재단 기초과학연구사업의 지원을 받아서 작성되었음(과제번호:R01-2004-000-10586-0(2004)).

<참고문헌>

[1] W3C, RDF Primer, 2004. 2 (http://www.w3c.org)
[2] S. Melnik, "Storing RDF in a Relational Database" (http://www-db.stanford.edu/~melnik/rdf/db.html)
[3] J. Broekstra et al., "Sesame : A Generic Architecture for Storing and Querying RDF and RDF Schema", Proc. of the 1st Int'l Semantic Web Conference, pp. 54-68, 2002
[4] G. Copeland and S. Khoshafian, "A Decomposition Storage Model", In Proc. of the ACM SIGMOD Inter'l Conf. on Management of Data, pp. 268-279, 1985
[5] R. Agrawal, A. Somani, Y. Xu: "Storage and Querying of E-Commerce data", 27th Int'l Conf. on Very Large Data Bases (VLDB), pp. 149-158, 2001
[6] Li Ding et al., "How the Semantic Web is Being Used:An Analysis of FOAF", In Proc. of the 38th Hawaii Int'l Conf. on System Sciences, 2005
[7] FOAF project (http://www.foaf-project.org)
[8] CARA RDF Parser (http://cara.sourceforge.net/)