시맨틱 웹에서 RDF 데이터 저장구조들의 성능비교

김경호, 백우현, 손지은, 김경창
홍익대학교 컴퓨터공학과
e-mail : gotaneraseunavez@gmail.com, qordngus88@nate.com, taijiun@naver.com,
kckim@hongik.ac.kr

Comparison of Storage Structures for RDF Data in Semantic Web.

KyungHo Kim, WooHyoun Back, JiEun Son, KyungChang Kim
Dept. of Computer Engineering Hongik University

요 약

RDF(Resource Description Framework)는 시맨틱 웹의 기초로서 웹 사용자에게 정보를 보다 정확하고 효율적으로 접근하는 표준이다. RDF 데이터를 효율적으로 저장하고 접근하는 필요성이 날로 증가하고 있다. RDF 데이터를 저장하고 검색하는 기본 저장 구조는 관계형 데이터베이스를 이용하는 것이다. 최근에는 RDF 데이터가 엄청나게 증가하고 있는 시점에 대용량 database 의 질의(단순 조회)에 최적화된 칼럼-지향(column-oriented) 데이터베이스가 대안으로 제안되었다. 본 논문에서는 RDF 데이터의 저장 구조로서 관계형 데이터베이스와 칼럼-기반 데이터베이스를 비교분석 하고자 한다. Berlin SPARQL Benchmark 를 이용한 성능분석 결과 RDF data 의 저장 구조로서 칼럼-기반 데이터베이스의 효율성을 입증하였다.

1. 서론

시맨틱 웹은 웹을 인간이 읽는 형태에서 기계가 처리하는 형태로 변환하는 시도이다. 시맨틱 웹의 목적은 여러 다른 응용들과 기관에 퍼져 있는 웹 데이터를 통합하고 공유하는 것이다. RDF(Resource Description Framework)는 시맨틱 웹의 기초로서 웹 사용자에게 정보를 보다 정확하고 효율적으로 접근하는 표준이다. RDF 에서는 얻고자 하는 resource 를 추려내기 위해서 해당 resource 의 subject, property, object 에 해당하는 특성들을 뽑아내어 xml 언어의 형태로 기술한다 [1]. 이 형태의 데이터에 대해서 사용자가 원하는 작업을 하는 데에는 성능적 혹은 관리적인 측면에서 여러 가지 장애가 따르게 된다. 이러한 점을 보완하기 위해서 RDF Triple 형태(subject, property, object)를 관계형 데이터베이스에 저장하는 시도가 많이 있었다.

최근 화두가 되고 있는 빅 데이터(Big Data)와 같이 대용량 data 를 빠른 시간 안에 수집하고 저장, 분석, 사용하기에는 튜플(tuple)-기반 형태인 관계형 데이터베이스 저장방식은 질의 실행 시간을 비추어 봤을 때 많은 한계와 제한이 있다. 최근에는 RDF 데이터를 튜플-기반이 아닌 칼럼-기반으로 저장하는 칼럼-지향(column-oriented) 데이터베이스가 대안으로 떠오르고 있다. 본 논문에서는 RDF 데이터를 효율적으로 저장하는 저장 구조로서 관계형 데이터베이스와 칼럼-기반 데이터베이스를 비교 분석 하고자 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 언급하고, 3 장에서는 RDF 저장구조에 대해 살펴보고, 4 장은 성능 비교 분석을 실시하고 5 장에서는 결론을 맺는다

2. 관련 연구

RDF 데이터의 가장 기본적인 저장 구조로는 RDF Triple 형태(subject, property, object)를 1 개의 관계형 테이블에 저장하는 것이다 [2]. RDF Triple 형태는 RDF 질의를 처리하기 위하여 같은 correlated 테이블을 여러 번 self-join 해야 하기 때문에 성능이 저하되는 단점을 지니고 있다.

이러한 성능 단점을 보안하기 위한 구조로 RDF 데이터를 1 개의 테이블에서 기존의 관계형 스키마를 저장하는 형태인 여러 관계형 테이블로 저장한다. 이를 위해 RDF Triple 형태(subject, property, object)에서 같이 정의된 property 집합을 찾아서 테이블 칼럼(column)으로 생성한다 [3,4].

최근에는 RDF 데이터를 효율적으로 처리하기 위하여 RDF 데이터를 관계형 스키마가 아닌 칼럼-지향 데이터베이스 구조로 저장하는 추세이다 [5, 6]. 칼럼-지향 데이터 구조는 RDF 데이터를 관계형 테이블에서 튜플-기반 혹은 행-기반(row-oriented)이 아닌 칼럼-기반으로 저장하는 구조이다.

3. RDF 저장구조

RDF 데이터를 저장하기 위한 기본적인 방법은 단순히 subject, property, object 라는 3 개의 column 을 지닌 1 개의 table 를 생성하는 것이다. 이에 대한 스키마는 <표 1>과 같으며 이는 일반적으로 RDF Triple 이라고 표현된다. 아래 각 표에서 사용되는 table instance 들은 Berlin SPARQL Benchmark 에서 인용되었다.

<표 1> RDF Triples 구조

-RDF Triples table-

SUBJ	PROP	OBJ
ProductFeature1	type	ProductFeature
ProductFeature1	label	Biographies
ProductFeature1	publisher	StandardizationInstitution1
ProductFeature1	date	2000-06-22
ProductFeature1	comment	Impoverishment reliers
ProductFeature2	type	ProductFeature
ProductFeature2	label	Yodeler
ProductFeature2	date	2000-07-06
ProductFeature2	Publisher	StandardizationInstitution1
ProductFeature3	type	ProductFeature
ProductFeature3	date	2000-06-21

<표 1>과 같은 형식은 관계형 DBMS 에서의 질의 처리를 위해 전혀 최적화되지 않은 스키마라 할 수 있다. 이를 관계형 DBMS 에서의 일반적인 table 구성 형식으로 최적화시켜 바꾼다면 <표 2>와 같이 property, left-over 두 table 로 나타내어 질 수 있다.

<표 2> 관계형 데이터베이스 구조.

-property table-

SUBJ	date	type
ProductFeature1	2000-06-22	ProductFeature
ProductFeature2	2000-07-06	ProductFeature
ProductFeature3	2000-06-21	ProductFeature
ProductFeature4	2000-06-27	ProductFeature

-left-over table-

SUBJ	PROP	OBJ
ProductFeature1	label	Biographies
ProductFeature1	publisher	StandardizationInstitution1
ProductFeature1	comment	Impoverishment reliers
ProductFeature2	label	Yodeler
ProductFeature2	Publisher	StandardizationInstitution1

RDF 데이터의 또 다른 저장 구조로서 칼럼-기반 데이터베이스를 사용할 수 있다. <표 2>와 같은 관계형 table 을 column-oriented 데이터베이스에 맞게 변경하면 <표 3>과 같이 subject, object 의 2 개의 column 을 지진 복수의 table 들로 구성할 수 있다.

<표 3> 칼럼-기반 데이터베이스 구조.

-type table-

SUBJ	OBJ
ProductFeature1	ProductFeature
ProductFeature2	ProductFeature
ProductFeature3	ProductFeature
ProductFeature4	ProductFeature

-date table-

SUBJ	OBJ
ProductFeature1	2000-06-22
ProductFeature2	2000-07-06
ProductFeature3	2000-06-21
ProductFeature4	2000-06-27

-label table-

SUBJ	OBJ
ProductFeature1	Biographies
ProductFeature2	Yodeler

-comment table-

SUBJ	OBJ
ProductFeature1	Impoverishment reliers

-publisher table-

SUBJ	OBJ
ProductFeature1	StandardizationInstitution1
ProductFeature2	StandardizationInstitution1

이와 같은 저장구조 전환에 대한 자세한 정보는 [6]를 참조하기 바란다.

4. 성능분석

이 장에서는 앞서 설명한 RDF 데이터에 대한 여러 저장구조를 바탕으로 실험을 진행하였다. 본 실험 및 성능 분석은 Berlin SPARQL Benchmark [7,8]를 사용하였다.

4.1 실험 환경

먼저 실험에 사용된 하드웨어 및 소프트웨어 사양은 다음과 같다.

CPU : Pentium(R) Dual-Core CPU E6600 @ 3.06GHz
Memory : DDR3 4GB
HDD : SATA 500G (WDC WD5000AAKS-0)

OS : Oracle Enterprise Linux 4 update 8 - 32 bit [9]
DBMS : Oracle 10g release2 [10]

Oracle DBMS 의 환경설정에서 디스크 I/O 에 직접적으로 영향을 주는 db_block_size (디스크 I/O 의 단위가 되는 block 의 크기) 와 db_file_multiblock_read_count (한번 I/O 가 발생할 때 읽어 들이는 Block 의 개수)에 대한 설정은 다음과 같다.

db_block_size = 8192 (8k)
db_file_multiblock_read_count = 1

본 실험에서 사용되는 Berlin SPARQL Benchmark 에서 Scale-Factor 1000 에서 RDF Triples 형태를 relation 형태로 나타내었을 때의 tuple 의 개수는 27,886 이다.

다음은 Oracle DBMS 안의 스키마 환경 구성 및 실험 방법이다. RDF Triples, RDF row oriented, 그리고

RDF column-oriented 저장구조를 객관적으로 비교하기 위하여 전부 테이블로 각각 구성한다. 성능 비교에서 column-oriented 데이터베이스 저장 구조 부분에 대해서만 Oracle 이 아닌 다른 column-oriented DBMS 를 사용하게 된다면 DBMS 내부 최적화 방식, 환경변수, 아키텍쳐, 처리 메커니즘 등에 따라서 그 객관성을 잃어 버릴 수 있다. 따라서 본 논문에서는 모든 저장 구조를 같은 DBMS 인 oracle 10g 을 이용하여 테이블 구성을 하고 실험을 진행하도록 한다. 특히 칼럼-기반 데이터베이스를 emulate 하기 위하여 2 column table 구성으로 하였다. 이는 실제로 칼럼-기반 데이터베이스의 저장 방식과 유사하다고 할 수 있다.

모든 저장 구조에 대한 테이블들은 Berlin SPARQL Benchmark 제공하는 data 를 삽입한다. 여기에 같은 내용을 출력하게 하는 query 들을 앞에 언급된 세 개의 테이블 형태에 각각 맞게 수정하여 질의 실행 시간을 측정한다. 실험에 사용될 query 들은 SQL 튜닝이 가능함을 고려하여 각 table 에서 가장 최적화된 query 들로 객관화시켜 적용하되, 복수의 query 들을 5 번씩 실행시켜 구한 평균 실행 시간을 성능비교의 지표로 삼는다. 이하 query 들의 유형은 Q1~Q5 라고 칭하고 각 유형에는 세 저장 방식에 따른 query 들 3 개를 포함한다. 각 저장 구조에 대한 relation 스키마는 다음과 같다.

--RDF Triples 스키마
TRIPLES (subj:varchar2, prop:varchar2, obj:varchar2)

--RDF 관계형 데이터베이스 스키마
PROPERTY (subj:varchar2, ttype:varchar2, ddate:varchar2)
LEFT_OVER(subj:varchar2, prop:varchar2, obj:varchar2)

--RDF 칼럼-기반 데이터베이스 스키마
COMMENT (subj:varchar2, obj:varchar2)
COUNTRY (subj:varchar2, obj:varchar2)
DDATE (subj:varchar2, obj:varchar2)
HOMEPAGE (subj:varchar2, obj:varchar2)
LABEL (subj:varchar2, obj:varchar2)
MBOX_SHA1SUM (subj:varchar2, obj:varchar2)
NAME (subj:varchar2, obj:varchar2)
PUBLISHER (subj:varchar2, obj:varchar2)
SUBCLASSOF (subj:varchar2, obj:varchar2)
TTYPE (subj:varchar2, obj:varchar2)

실험에 사용된 query 는 해당 link 를 참조하기 바란다.[11]

4.2 비교분석

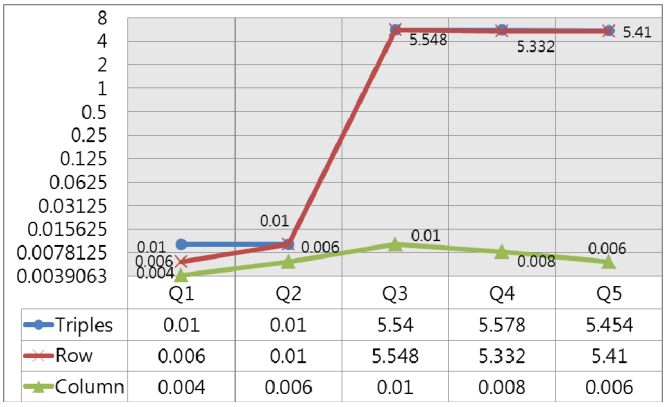
저장 구조에 대한 RDF 데이터의 성능 비교를 위하여 본 실험에서는 5 개의 질의를 실행하였다. Q1, Q2 의 경우 two-way join, Q3, Q4 는 three-way join, Q5 는 four-way join 이다. 모든 질의의 SELECT 절에는 대표적인 집계 함수인 count 함수를 넣었다.
<표 4>는 각 질의들의 실행시간을 각각 5 번씩 돌려 평균 실행 시간을 구한 것이다.

<표 4> 질의 평균 실행 시간

	Triples	Row	Column
Q1	0.01	0.006	0.004
Q2	0.01	0.01	0.006
Q3	5.54	5.548	0.01
Q4	5.578	5.332	0.008
Q5	5.454	5.41	0.006

<표 5>에서 보여지는 바와 같이 실험에 사용된 모든 질의의 유형에 대해서 column oriented 데이터베이스 저장방식이 RDF 데이터 처리에서 우세하다는 것을 볼 수 있으며 그 차이는 multi-way 의 깊이가 깊어질 수록 더욱 심화된다는 것을 알 수 있다. 현재 널리 사용되는 관계형 데이터베이스 저장 방식이 multi-way join 으로 갈수록 단순 Triples 저장 방식과 비교하여 크게 성능차이가 나지 않는다. 이는 RDF 가 주로 사용이 되는 시맨틱 웹, 센서 네트워크와 같은 환경에서 단순 조회의 질의가 많이 사용되는 경우 column-oriented 저장 방식으로 DB 를 구성함이 얼마나 효율적인지를 보여주고 있다. 실제 실험 결과의 화면 캡쳐는 링크를 참조하기 바란다.[12]

<표 4> RDF 저장 구조들의 평균 질의 실행 시간



5. 결론

RDF 는 시맨틱 웹의 기초로서 웹 사용자에게 정보를 보다 정확하고 효율적으로 접근하는 표준이다. 시맨틱 웹 환경에서 데이터 생산 속도 및 이용빈도가 급속도로 높아지고 있는 현실에 맞춘 RDF 데이터의 저장 방식 변화는 매우 가치 있고 중요한 일이다. RDF 데이터의 저장 구조로는 기존의 RDF Triple 형태(subject, property, object)와 관계형 데이터베이스 및 최근에는 칼럼-지향 데이터베이스가 있다.

본 논문에서는 RDF 데이터를 저장하고 검색하는 이들 세가지 저장 구조를 Berlin SPARQL Benchmark 를 이용하여 비교하고 질의 처리 성능 분석을 실시하였다. 실험 결과 RDF 데이터를 효율적으로 저장하고 검색하는 저장 구조로 칼럼-기반 데이터베이스의 우수성을 관찰하였다.

참고문헌

[1] O Lassila, RR Swick. Resource description framework (RDF) model and syntax specification. In W3C Working Draft, 1998.

[2] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An Efficient SQL-based RDF Querying Scheme. In VLDB, pages 1216-1227, 2005.

[3] K. Wilkinson. Jena property table implementation. In SSWS, 2006.

[4] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient RDF Storage and Retrieval in Jena2. In SWDB, pages 131-150, 2003.

[5] D.J. Abadi. Column stores for wide and sparse data. In CIDR, 2007.

[6] DJ Abadi, A Marcus. Scalable Semantic Web Data Management Using Vertical Partitioning. In VLDB '07 Proceedings of the 33rd international conference on Very large data bases. Pages 411-422. 2007.

[7] C Bizer, A Schultz. The berlin sparql benchmark. In International Journal on Semantic Web and Information Systems(IJSWIS). 2009.

[8] Berlin SPARQL Benchmark (BSBM) website.
http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

[9][10] Oracle website.
http://www.oracle.com/index.html

[11] 실험에 사용된 query 들
http://223.194.70.141/~oodb/queries.sql

[12] 실제 실험 결과 화면 캡쳐
http://223.194.70.141/~oodb/Query-test.html