대용량 RDF 데이터의 처리 성능 개선을 위한 효율적인 저장구조 설계 및 구현

A Design and Implementation of Efficient Storage Structure for a Large RDF Data Processing

문현정(Mun Hyeon Jeong)*, 성정환(Sung Jung Hwan)**,
김영지(Kim Young Ji)***, 우용태(Woo Yong Tae)****

초 록

본 논문에서는 대용량 RDF의 효율적인 저장을 위하여 관계 정보와 데이터 정보를 분리한 새로운 방식의 저장 구조를 제안하였다. 제안 방식은 기존의 저장 방식에 비해 데이터의 중복을 최소화하여 대량의 RDF 데이터를 효율적으로 저장할 수 있다. 또한 본 논문에서 제안한 저장 방식을 이용하여 트리플 형태의 관계 정보 릴레이션과 데이터 정보 릴레이션에서 필요한 데이터를 분리 검색하여 결합하는 방식에 의해 RDF 데이터에 대한 질의 성능을 개선할 수 있다. 본 연구 결과는 RDF 데이타를 이용한 전자상거래, 시맨틱 웹, 지식관리 등과 같은 응용 분야에서 대량의 RDF 데이터의 효율적인 관리를 통하여 질의 성능을 개선할 수 있는 기반 기술로 사용할 수 있다.

ABSTRACT

We design and implement an efficient storage technique to improve query processing for a large RDF data set. The proposed techniques can minimize data redundancy compared to the existing techniques by splitting relation information and data information from triple formatted RDF data. Also, we can enhance query processing speed separating and connecting the entire query steps by relation and data based on the proposed storage technique. The proposed technique can be applied to the areas, such as e-Commerce, semantic web, and KMS to store and retrieve a large RDF data set.

키워드 : RDF 저장 구조, 질의처리, 온톨로지
RDF Storage Structure, Query Processing, Ontology

이 논문은 2004년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2004-206-D00039).

* 창원대학교 연구교수
** 키코테크 연구원
*** 하이브레인넷 책임연구원
**** 창원대학교 컴퓨터공학과 교수

1. 서 론

최근에 웹 문서의 급격한 증가와 전자거래의 활성화에 따라 W3C의 주도아래 기계가 이해할 수 있는 시맨틱 웹에 대한 연구가 활발하게 진행되고 있다[12]. 이러한 시맨틱 웹의 핵심 기술의 하나인 RDF는 '공유된 개념화의 정형화된 명시적 서술(a explicit specification of a shared conceptualization)'이라고 정의할 수 있다[20]. 즉, 해당 영역의 개념들과 이들 개념간의 상호 관계를 명시적으로 정의하기 위한 기술을 의미한다. 이러한 RDF 언어의 종류는 RDF(Resource Descri-ption Framework), RDFS(RDF Schema), DAML+OIL, OWL(Web Ontology Language) 등이 있다. RDF는 subject, predicate, object로 구성된 트리플 구조로 RDF 문서를 기술하기 위한 언어이다. RDFS는 RDF 문서의 subject, predicate, object에 대한 타입과 관계를 정의하기 위한 언어로 RDF와 함께 사용된다[8,11]. DAML+OIL은 미국에서 개발한 DAML ONT와 유럽에서 개발한 OIL (Ontology Interface Layer)을 결합한 RDF 언어이다[9]. OWL은 RDFS와 DAML+OIL을 확장한 언어로 자원들에 대한 다양한 개념과 관계를 기술할 수 있고, 추론 기능을 통하여 강력한 표현력을 지원한다[10].

최근에 관계형 데이터베이스를 이용하여 대량의 RDF 데이터를 효율적으로 관리하기 위한 연구가 활발하게 진행되고 있다. ICS-FORTH에서는 속성별 릴레이션 저장 방식과 수직분할 저장 방식에 기반을 둔 RDF 스토리지인 RDFSuite를 제안하였다[18]. HP사에서는 수직적 트리플 구조를 이용한 Jena2 스토리지 시스템을 제안하였다[15]. 독일의 Karlsruhe 대학에서는 확장된 수직적 트리플 저장 구조를 이용하는 Kaon 시스템을 제안하였다[17]. On-to-Knowledge 프로젝트인 Sesame에서는 RDF 스키마에 정의된 클래스와 속성별로 독립적인 릴레이션을 생성하여 RDF 데이터의 정보를 저장하는 방식을 사용하였다[13]. 국내에서도 데이터베이스를 이용하여 대규모 RDF 데이터를 저장하기 위한 연구가 활발하게 진행되고 있다[2]. KISTI에서는 지식기반 정보유통 플랫폼의 추론시스템 구현을 위해 DBMS 기반의 RDF 확장 모델인 OntoFrame-K®을 개발하였다[3, 4]. 김연희 등은 시맨틱 웹 기반의 메타데이터와 RDF 정보를 인덱싱하기 위한 기법을 제안하였다[1]. 탁경현 등은 Sesame 시스템의 스키마를 기반으로 효율적인 RDF 저장 스키마를 제안하였다[6]. 하지만, RDF 데이터를 저장하기 위한 기존의 연구 기법은 데이터가 중복하여 저장되는 문제로 인해 대용량의 RDF 데이터에 대한 효율적인 관리가 어렵다. 또한 중복 저장된 데이터로 인해 불필요한 조인 연산이 대량으로 발생하여 질의 성능이 저하되는 문제점이 있다.

본 논문에서는 RDF 데이터를 관계 정보와 데이터 정보로 분리하여 저장하는 새로운 방식을 제안하였다. 제안 방식은 기존의 저장 방식에 비해 데이터의 중복을 최소화하여 대량의 RDF 데이터를 효율적으로 저장할 수 있다. 또한 이 방식은 RDF 검색을 위하여 트리플 형태의 관계 정보 릴레이션

과 데이터 릴레이션에서 필요한 데이터를 분리 검색하여 결합하는 방식에 의해 질의 성능을 개선할 수 있다. 본 연구에서 제안한 기법의 효율성을 검증하기 위하여 Oracle 10g Spatial Resource Description Framework에서 예제로 제공하는 Family RDF 스키마를 이용하여 실험용 RDF 데이터를 생성하였다[7]. 그리고 Family RDF를 대상으로 제안 기법과 기존의 RDF 저장 기법간의 저장 공간과 질의 성능에 대한 비교 실험을 통하여 제안 기법의 효율성을 입증하였다.

2. 관련 연구

2.1 RDF 형식

RDF는 1999년에 W3C에서 표준화한 시맨틱 웹 언어로서 웹상에 존재하는 다양한 자원에 대한 메타데이터를 기술하기 위한 언어이다. RDF는 의미의 손상 없이 응용시스템간에 정보를 교환할 수 있도록 공통의 프레임을 제공하기 위한 형식으로 RDF에서는 웹상에 있는 모든 자원을 표현하기 위해 속성(property)과 속성 값(value)을 저장한다. RDF 형식은 자원의 특성이나 다른 자원과의 관계를 〈subject, predicate, object〉로 구성된 트리플 형식으로 표현한다[3, 4]. RDF 스키마는 애플리케이션에서 사용할 클래스와 속성을 RDF 어휘로 기술한 것이다. 즉, 웹 자원을 기술하기 위한 메타데이터의 구조를 RDF 어휘로 기술한 것이다. RDF 스키마는 다양한 사물을 클래스로 기술하며, 클래스는 트리플 형식에서 subject로 rdfs: Class와 rdfs: Resource를, predicate으로 rdf: type과 rdfs: subClassOf를 사용하여 기술한다.

2.2 기존의 RDF 데이터 저장 방식

기존의 RDF 데이터를 저장하는 방식은 크게 세 종류로 나누어진다. 전통적인 파일 시스템을 이용하는 방식, XML 형식의 Large Object 타입을 사용하는 방식 그리고 관계형 데이터베이스를 이용하여 RDF 데이터를 저장하는 방식이다. 최근에 널리 사용되는 관계형 데이터베이스를 이용한 저장 방식은 수직분할 저장 방식, 속성별 릴레이션 저장 방식, 수평분할 저장 방식, 그리고 혼합형 저장 방식이 있다[13, 15, 17, 18].

2.2.1 수직분할 저장 방식

수직분할 저장 방식은 RDF 데이터를 관계형 데이터베이스에 저장하는 방법중에서 가장 널리 사용되는 방식이다. 이 방식은 RDF 문서를 〈subject, predicate, object〉형태의 트리플 구조로 단일 릴레이션에 RDF 스키마와 데이터를 함께 저장하는 방식이다. 이 방식은 RDF 데이터 전체를 단일 릴레이션에 저장하기 때문에 관리가 용이하다. 그리고 RDF의 구조 변경시에도 데이터만 추가되기 때문에 RDF의 구조 변화를 쉽게 반영할 수 있다

하지만 이 방식에서는 클래스가 가진 속성의 수에 비례하여 subject 값의 중복이 발생하고, 인스턴스 수에 따라 subject와 predicate 값의 중복이 발생한다. 또한 질의

수행시 전체 릴레이션을 스캔해야하는 문제점이 발생한다. 그리고 복잡한 질의 수행시에 반복적인 셀프조인이 발생하여 질의 성능이 저하될 수 있다. 이 방식을 이용한 기존 시스템은 Jena, KAON 등이 있다[15, 17].

2.2.2 속성별 릴레이션 저장 방식

속성별 릴레이션 저장 방식은 'decomposition storage model'이라고 불리며 속성별로 릴레이션을 할당하여 저장하는 방식이다[21]. 하나의 속성 릴레이션은 2개의 subject와 object 애트리뷰트를 가진다. 이 방식은 속성별로 릴레이션을 생성하므로 릴레이션의 수는 증가하지만, 각 릴레이션의 크기는 줄어든다. 즉, 속성 이름이 릴레이션 이름으로 생성되어 predicate에 해당하는 값이 수직 분할 방식과는 다르게 중복 저장되지 않는다.

하지만, 이 방식은 RDF 스키마에서 속성이 변경되거나 확장되는 경우에는 릴레이션 이름이 변경되거나 릴레이션 수가 증가하는 단점이 있다. 또한 동시에 여러 개의 속성값을 검색하는 경우에 조인 연산이 많이 발생하여 질의 성능이 저하될 수 있다. 이 방식을 이용한 시스템은 PARK, DLDB 등이 있다[14, 21].

2.2.3 수평분할 저장 방식

수평분할 저장 방식은 수직분할 저장 방식처럼 단일 릴레이션에 저장하고, RDF 스키마에 정의된 모든 속성을 애트리뷰트로 정의한다. 즉, 클래스 인스턴스별로 하나의 튜플로 저장하는 방식이다[16, 21]. 이 방식은 subject와 predicate 값이 중복되는 수직분할 방식의 단점과 다중 릴레이션이 생성되는 속성별 릴레이션 저장 방식의 단점을 보완한 방식이다. 따라서 이 방식은 여러 개의 속성을 조회하는 경우에 다른 방식에 비해 조인 과정이 불필요하게 되어 질의 속도가 빠르다.

하지만 이 방식은 속성이 다중 속성값을 가지고 있는 경우에 하나의 값만 저장할 수 있는 제약 사항이 있다. 또한, 속성이 증가하는 경우에는 애트리뷰트를 추가해야 하므로 릴레이션의 구조가 변경되는 문제점이 발생한다. 이 방식을 이용한 모델은 OntoFrame-K®가 있다[3].

2.2.4 혼합형 저장 방식

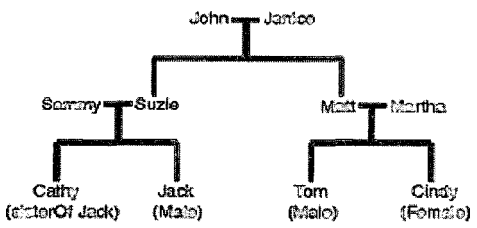
혼합형 저장 방식은 속성별 릴레이션 저장 방식과 수직분할 저장 방식을 결합한 방식이다. 이 방식은 각 클래스와 속성별로 별도의 릴레이션이 생성되며, 클래스 릴레이션에는 인스턴스 고유의 ID값과 인스턴스 값이 저장된다. 그리고 속성 릴레이션에는 관련된 클래스들의 인스턴스 ID값이 저장된다[1, 18, 21]. 따라서 이 방식은 수직분할, 수평분할, 속성별 릴레이션 저장 방식보다 저장 용량이 감소하여 대규모 RDF 데이타를 효과적으로 저장할 수 있다[16, 21].

하지만 이 방식은 각 클래스와 속성별로 별도의 릴레이션을 생성하므로 릴레이션 수가 많아지고 다수의 조인이 발생하는 문제점이 발생한다. 그리고 RDF에서 클래스간, 속

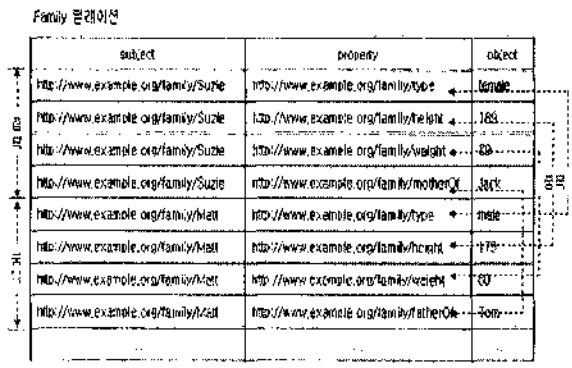
성간 관계를 표현하는 subClassOf, subPropertyOf와 같은 관계를 표현하기 어렵다.

2.3 기존 저장 방식의 문제점

기존에 RDF 데이터를 트라플 형식으로 관리하는 대표적인 방법은 수직분할 저장 방식과 수평분할 저장방식이 있다. 본 절에서는 기존 방식의 문제점을 분석하고 제안 방식의 타당성을 검증하기 위하여 Oracle Spatial 10g에서 샘플로 제공하는 Family RDF 예제를 대상으로 비교하였다. 〈그림 1〉은 본 논문에서 RDF 예제로 사용한 Family 구조를 나타낸 그림이다. 〈그림 2〉는 Family 구조중에서 Suzie 데이타를 RDF 형식으로 표현한 그림이다. 먼저, 수직분할 방식에서는 RDF 트리플을 하나의 릴레이션에 저장하므로 subject와 predicate 값의 중복이 많이 발생한다. 〈그림 3〉은 수직분할 저장방식에 따른 Family 릴레이션을 나타낸 그림이다. 그림에서처럼 Suzie와 Matt 데이터의 subject 속성 수만큼 중복되며, 인스턴스의 수만큼 type, height, weight, motherOf, fatherOf 애 트리뷰트가 중복됨을 알 수 있다.



〈그림 1〉 Oracle Spatial RDF 10g의 Family RDF 예제[7]



〈그림 3〉 수직분할 방식으로 저장된 Family RDF

```
〈rdf: Description rdf: about = "#Person"〉
  〈rdf: type rdf: resource = "http://www.w3.org/2002/07/owl#Class"/〉
〈/rdf: Description〉

〈rdf: Description rdf: about = "#Suzie"〉
  〈type rdf: datatype = "http://www.w3.org/2001/XMLSchema#integer"〉female〈/weight〉
  〈weight rdf: datatype = "http://www.w3.org/2001/XMLSchema#integer"〉89〈/weight〉
  〈height rdf: datatype = "http://www.w3.org/2001/XMLSchema#decimal"〉189〈/height〉
  〈rdf:type rdf: resource = "#Person"/〉
  〈motherOf rdf: resource = "#Jack"/〉
〈/rdf:Description〉
```

〈그림 2〉 RDF 데이터로 표현한 Family 구조의 일부

Family 릴레이션

name	type	height	weight	motherOf	fatherOf	?
http://www.example.org/family/Suzie	female	189	89	Jack		
http://www.example.org/family/Matt	male	179	80		Tom	
...	...	...	...	...	...	...

널 값 — 칼럼 추가 가능성

〈그림 4〉 수평분할 방식으로 저장된 Family RDF

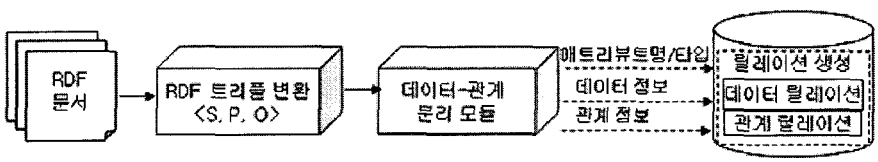
수평분할 방식에서는 속성별로 릴레이션을 분리하므로 데이터의 중복은 발생하지 않는다. 하지만, 검색 결과에 따라 조인 수와 애트리뷰트가 변경되어 질의 처리가 비효율적이다. 〈그림 4〉는 수평분할 저장방식에 따른 Family 릴레이션을 나타낸 그림이다. 그림에서처럼 motherOf와 fatherOf 애트리뷰트는 해당 값이 없는 경우에는 널 값을 가지며, RDF 스키마의 변경으로 인해 애트리뷰트가 추가될 가능성이 있다.

3. 관계 정보와 데이터 정보를 분리한 RDF 데이터 저장 구조

본 연구에서는 대량의 RDF 데이터의 효율적인 저장을 위해 RDF의 관계 정보와 데이터 정보를 분리하여 관계형 데이터베이스에 저장하는 새로운 형태의 저장구조를 제안하였다. 먼저, 트리플 형식으로 변환된 RDF 데이터는 본 논문에서 제안한 관계-데이터 분리 알고리즘에 의해 관계 정보와 데이터 정보로 분리된다. 이때 관계 정보는 트리플 형식으로 저장되며, 데이터 정보는 수평분할 저장방식처럼 릴레이션의 애트리뷰트로 저장된다.

3.1 RDF 트리플 변환

먼저, RDF 데이터는 Jena API를 이용하여 트리플 형식으로 변환하였다. 트리플 형식으로 변환된 RDF 데이터는 본 연구에서 제안한 관계-데이터 분리 알고리즘에 의해 관계 릴레이션과 데이터 릴레이션으로 생성하였다. 실험용 RDF 데이터는 Oracle Spatial 10g에서 제공하는 Family RDF 샘플을 확장하여 사용하였다.



〈그림 5〉 관계 정보와 데이터 정보를 분리하는 RDF 데이터 저장 방식의 개요도

3.2 RDF 문서에 대한 관계-데이터 분리 알고리즘

관계- 데이타 분리 알고리즘은 RDF 데이터로부터 관계 릴레이션과 데이타 릴레이션을 분리하여 생성하기 위한 알고리즘이다. 여기서 관계 릴레이션은 수직분할 방식으로 관리하고, 데이터 릴레이션은 수평분할 방식으로 관리한다. 트리플 형식의 RDF 데이터에서 property가 'type'이고 object가 'Object-Property'인 트리플은 관계 릴레이션의 데이터로 관리한다. 또한 predicate이 'type'이고 object가 'DataProperty'인 subject는 데이터 릴레이션의 애트리뷰트명으로 구분한다. <그림 6>은 본 연구에서 제안한 RDF 데이터로부터 관계 정보와 데이타 정보를 분리하여 저장하기 위한 알고리즘이다.

```
// RDF 데이타를 〈subject, predicate, object〉 형식의 트리플로 변환
WHILE i 〉 RDF statement 수
        Get subject to set triple[i][0] from RDF statement;
        Get predicate to set triple[i][1] from RDF statement;
        Get object to set triple[i][2] from RDF statement;

// 관계 정보를 저장하기 위한 릴레이션 생성
Create relation table with subject, predicate and object column;

// 데이터 정보 릴레이션의 애트리뷰트명과 데이터타입 결정
IF  triple[i][1] = "range" THEN
        IF  triple[i][2] = "string" THEN
                column_name[count][0]  =  triple[i][0];
                column_name[count][1]  =  "VARCHAR2(200)";

        ELSE IF triple[i][2] = "decimal" OR triple[i][2] = "integer" THEN
                column_name[count][0]  =  triple[i][0];
                column_name[count][1]  =  "NUMBER(7, 2)";
        END IF
END IF

//관계 정보 데이터를 관계 릴레이션에 입력
IF  triple[i][2] NOT EQUAL string AND  triple[i][2] NOT EQUAL decimal
    AND triple[i][2] NOT EQUAL integer THEN
        Insert triple[i] into relation table;
END IF
END WHILE
// 데이터 정보를 저장하기 위한 릴레이션 생성
CREATE data table using column_name

// 데이터 정보를 데이터 릴레이션에 입력
WHILE j 〉 RDF 트리플 수
      FOR k = 1 TO count
        IF column_name[k][0]  =  triple[j][1] THEN
                IF triple[j-1][0] NOT EQUAL triple[j][0] THEN
                        Insert triple[j][0] into subject column of data table;
                ELSE
                        Update data table
                        set triple[j][2]  =  column_name[k][0]
                        where subject  =  triple[j][0];
                ENDIF
        ENDIF
        INCREMENT k
    END FOR
ENDWHILE
```

〈그림 6〉 RDF 데이터에 대한 관계-데이타 분리 알고리즘

3.3 관계-데이터 정보 분리를 위한 데이터베이스 구조

본 논문에서 RDF 데이터는 관계-데이터 분리 알고리즘에 의해 관계 정보와 데이터 정보로 분리된다. 여기서 관계 정보는 RDF 트리플 형식을 그대로 저장하는 수직분할 저장방식을 사용한다. 그리고 데이터 정보는 수평분할 저장방식을 사용하여 저장한다. 관계 정보와 데이터 정보를 위한 릴레이션은 RDF 데이터의 스키마 정보를 분석하여 생성된다. 제안 방식은 기존의 수평분할 저장방식에서 대량의 널 값을 가지는 애트리뷰트 생성 문제와 속성 추가에 따라 릴레이션의 구조가 변경되는 문제를 개선할 수 있다. 즉, 데이터 속성과 객체 속성을 분리 저장하여 널 값을 가지는 애트리뷰트가 대폭적으로 감소하고, 새로운 객체가 추가되어도 릴레이션 구조 변경이 발생하지 않는 장점을 가진다.

〈그림 7〉은 본 연구에서 제안한 관계 정보와 데이터 정보를 분리한 데이터베이스의 구조를 나타낸 그림이다.

3.4 새로운 관계 추론을 위한 규칙 정의

본 연구에서는 RDF 데이터에 기술된 관계 정보를 이용하여 새로운 관계를 추론하기 위하여 규칙을 정의하였다. RDFS의 추론 방법은 전방향 추론(forward chaining)과 후방향 추론(backward chaining)이 있다. 먼저, 전방향 추론은 정의된 사실에 적용 가

subject	data_property1	data_property2	...	data_propertyN
http://www.example.org/subject1	value1	value2		valueN
http://www.example.org/subject2	value1	value2	...	valueN
...	...	...	...	...

(a) 데이터 정보를 저장하기 위한 릴레이션 구조

subject	property	object
http://www.example.org/subject1	object_property1	value1
http://www.example.org/subject1	object_property2	value2
...	...	...

(b) 관계 정보를 저장하기 위한 릴레이션 구조

〈그림 7〉 관계 정보와 데이터 정보를 분리한 데이터베이스 구조

능한 규칙을 적용하여 최종 목표에 도달하는 방식이다. 이 방식은 추론의 결과를 미리 저장하므로 질의 속도는 빠르나 저장 공간이 많이 소모되는 문제점이 있다. 후방향 추론은 목표에서 시작하여 이를 지지할 수 있는 사실들을 찾는 방식이다. 이 방식은 질의 처리 시간에 추론하는 관계로 질의 속도는 느리다.

본 연구에서는 관계형 데이터베이스에 기반하여 전방향 추론을 위한 규칙을 정의하였다. 기존의 전방향 추론에서 추론의 결과를 미리 저장하는 방식과는 달리 질의 처리시에 추론할 수 있는 방법을 제안하였다. 새로운 관계를 정의하기 위한 추론 규칙은 기존의 관계 정보를 이용한 방식, 관계 정보와 데이터 정보를 이용한 방식 그리고 규칙을 중첩으로 사용한 방식을 사용하여 정의하였다. 규칙은 관계형 데이터베이스에서 제공하는 함수를 이용하여 생성하였다.

3.4.1 기존의 관계 정보를 이용한 추론 규칙 정의

기존의 관계 정보를 이용한 규칙은 의미적인 계층 관계를 이용하여 정의한 규칙이다. 즉, 아버지 관계를 이용하여 할아버지 관계를 추출하기 위한 규칙이다. 즉, 할아버지는 아버지의 아버지를 찾는 경우로, 아버지 관계를 저장하고 있는 관계 릴레이션을 셀프 조인(self-join)하여 찾을 수 있다. 〈그림 8〉은 관계형 데이터베이스의 함수를 사용하여 정의한 추론 규칙으로 할아버지와 할머니를 찾기 위한 추론 규칙의 예이다.

```
CREATE FUNCTION grandParentOf (v_obj in VARCHAR2, v_pred in
VARCHAR2)RETURN VARCHAR2
  IS
        ln_error_no  NUMBER      := 0;
        ln_subj      VARCHAR2(100)  := '';

  BEGIN
        SELECT  p.subject into ln_subj
        FROM relationtab p, relationtab c
        WHERE c.predicate = 'http://www.example.org/family.rdf#' ||v_pred
        AND p.predicate = 'http://www.example.org/family.rdf#' ||v_pred
        AND c.object =v_obj
        AND p.object = c.subject;

  RETURN ln_subj;

END grandParentOf;
```

〈그림 8〉 관계 정보를 이용한 추론 규칙의 구성 예

3.4.2 관계 정보와 데이터 정보를 이용한 추론 규칙 정의

기존의 관계 정보와 데이터 정보를 이용한 규칙은 의미적인 계층 관계와 데이터 정보가 연결하여 새로운 관계를 정의하는 규칙이다. 예를 들어, 아버지의 여자 형제인 고모 또는 어머니의 여자 형제인 이모를 찾는 경우이다. 아버지와 어머니 관계는 관계 정보이고 성별은 데이터 정보이므로 관계 정보와 데이터 정보를 동시에 이용한다. 형제 정보는 RDF 데이터에서 제공하지 않으므로, 아버지 또는 어머니의 부모를 찾아서 같은 부모를 가지는 자식들은 형제로 추론하면 된다. <그림 9>는 관계 정보와 데이터 정보를 이용한 추론 규칙에서 사용한 SQL 구성 예이다.

3.4.3 중첩 추론 규칙 정의

새로운 관계에 대한 추론은 사전에 정의한 규칙을 중첩하여 적용하면 가능하다. 예를 들어, 증조 할아버지는 아버지를 찾는 규칙(parentOf)과 할아버지를 찾는 규칙(grandParentOf)을 중첩하여 적용하면 추론할 수 있다. 추론 결과를 미리 저장하는 기존 방식에서는 새로운 규칙을 정의할 때마다 추론 결과를 저장하는 관계로, 규칙을 중첩하여 새로운 규칙을 정의하기 어렵다.

3.5 관계-데이터 분리 방식에서 질의 처리

제안 방식의 저장 구조에서 질의 처리 방법은 단순 데이터만 질의하는 경우, 단순 관계만 질의하는 경우, 관계와 데이터를 결합한 질의, 규칙과 데이터를 결합한 질의, 데이터와 중첩 규칙을 결합한 질의로 구분할 수 있다. 여기서, 단순 관계나 단순 데이터 조회는 해당 릴레이션에 대해 SQL문에 의해 검색할 수 있다.

3.5.1 관계와 데이터를 결합한 질의 처리

관계와 데이터를 결합한 질의의 경우에는 인스턴스간의 관계를 이용하여 속성을 검색하는 경우이다. 예를 들어, 'John의 아버지의 키와 몸무게'를 찾는 질의는 관계와 데이터를 결합한 질의이다. 즉, John의 아버지 관계에 있는 사람을 찾는 경우는 관계 질의이고,

```
SELECT subject into l_subj
FROM   datatab
WHERE  subject IN (SELECT subject
                   FROM relationtab
                   WHERE subject = (SELECT grandParentOf(v_obj, v_pred)
                                    FROM dual))
                   AND sex = v_sex;
```

<그림 9> 관계 정보와 데이터 정보를 이용한 추론 규칙의 SQL 구성 예

질의 1) John의 아버지의 키와 몸무게는?

```
SELECT height, weight
FROM datatab
WHERE subject = ( SELECT subject
                  FROM relationtab
                  WHERE object = 'John'
                        AND  predicate = 'fatherOf' )
```

〈그림 10〉 관계와 데이터에 대한 질의가 결합된 질의 예

아버지의 키와 몸무게를 찾는 경우는 데이터 질의이다. 〈그림 10〉은 관계와 데이터가 결합된 질의를 처리하는 과정이다.

3.5.2 규칙과 데이터를 결합한 질의 처리

규칙과 데이터를 결합한 질의는 기존의 관계를 이용하여 새로운 관계를 검색하기 위한 규칙을 질의에 사용한 경우이다. 본 연구에서는 추론 결과를 미리 저장하는 기존의 방식과는 다르게 질의 처리시에 규칙을 SQL문으로 변환하여 처리하였다.

〈그림 11〉은 규칙과 데이터가 결합된 질의를 처리하는 과정이다. 그림에서처럼 John의 할아버지의 키를 검색하는 질의는 관계정보 릴레이션에서 John의 할아버지를 서브쿼리로 검색한 다음, 데이터정보 릴레이션에서 John의 할아버지 키를 검색하면 된다.

3.5.3 중첩 규칙과 데이터에 대한 질의가 결합된 경우

관계, 규칙, 데이터가 결합된 질의는 관계와 규칙관계에 있는 인스턴스를 검색한 다음, 인스턴스의 속성을 검색하면 된다. 예를 들어, 'John 어머니의 할아버지 키'를 검색하는 질의는 어머니 관계(motherOf)인 사람을 서브쿼리로 검색한 후, 조건을 만족하는 어머니와 할아버지 규칙 관계인 사람을 다

질의 2) John의 할아버지의 키는?

```
SELECT height
FROM   datatab
WHERE subject =
      (SELECT grandParentOf( 'fatherOf', 'http://www.example.org/family.rdf#john' )
       FROM   dual);
```

〈그림 11〉 규칙과 데이터에 대한 질의가 결합된 질의 예

질의 3) John 어머니의 할아버지 키는?

```
SELECT height
FROM datatab
WHERE subject =
        (SELECT grandParentOf( 'fatherOf',
                               parentOf( 'motherOf', 'http://www.example.org/family.rdf#john' )
        FROM dual);
```

〈그림 12〉 데이터와 중첩 규칙에 대한 질의가 결합된 질의 예

시 서브쿼리로 검색한 후, 그 사람의 키를 검색하면 된다. 〈그림 12〉는 관계, 규칙, 데이터에 대한 질의가 결합된 질의를 처리하는 과정이다.

4. 실험 및 결과

본 논문에서 제안한 대량의 RDF 데이터에 대한 저장 방식과 질의 성능의 효율성을 검증하기 위하여, Oracle Spatial RDF 10g에서 샘플로 제공하는 Family RDF 데이터 스키마와 관계 정보를 이용하여 실험용 RDF 데이터를 생성하였다. 실험을 위하여 1대 할아버지부터 14대 자손까지 총 32,766명에 대한 족보를 Family RDF 데이터로 생성하였다. 생성된 데이터 수는 32,771건의 관계 정보 데이터, 32,766건의 가족에 대한 기본 정보의 데이터로 총 65,545건이다. 개인의 기본 정보는 이름, 성별, 키, 몸무게, 이메일, 홈페이지, 휴대전화번호, 주소로 구성하였다. 관계 정보는 아버지 관계인 fatherOf와 어머니 관계인 motherOf만 정의하였다.

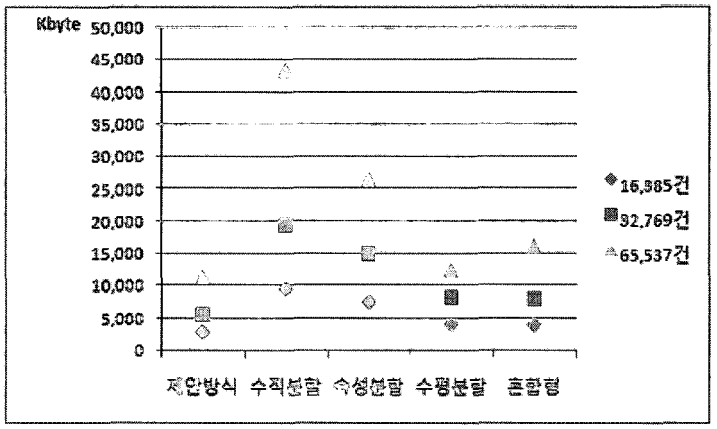
본 논문에서 제안한 RDF 데이터의 저장 방식과 기존의 저장 방식의 성능을 비교하기 위하여 저장 용량과 질의 처리 속도를 비교 실험하였다.

4.1 저장 용량 비교 실험

본 논문에서 제안한 방식과 기존의 저장 방식의 저장 용량을 비교하기 위하여, 데이터 정보가 관계 정보보다 많은 경우, 데이터 정보가 관계 정보보다 적은 경우로 구분하여 저장 용량을 비교하였다.

〈그림 13〉는 데이터 정보가 관계 정보보다 많은 경우에서 제안 방식과 기존의 수직분할, 속성별 테이블, 수평분할, 혼합형 저장 방식의 저장 용량을 비교한 그래프이다. 그림에서처럼 제안한 방식이 기존의 저장 방식보다 데이터 저장 공간을 적게 차지하는 것을 알 수 있다. 특히, 제안 방식은 데이터가 많을수록 기존 방식보다 중복 저장을 최소화하여 효율적으로 RDF 데이터를 저장할 수 있음을 알 수 있다.

〈그림 14〉는 데이터 정보가 관계 정보보다 적은 경우에서 제안 방식과 기존 저장 방식의 저장 용량을 비교한 그래프이다. 데이터 정보가 관계 정보보다 적은 경우는 각



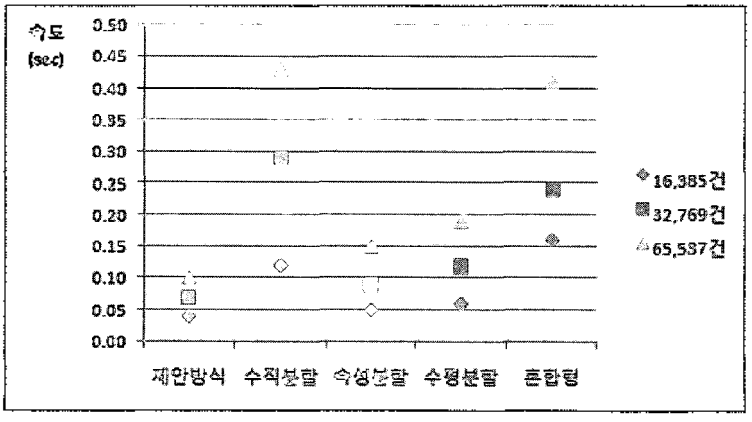
〈그림 13〉 데이터 정보가 관계 정보보다 많은 경우의 저장 용량 비교 결과

구성원의 관계인 fatherOf, motherOf 정보가 각 구성원이 가지고 있는 개인 정보보다 더 많은 경우이다. 제안한 방식이 기존의 저장 방식보다 저장 공간을 적게 차지하는 것을 알 수 있다.

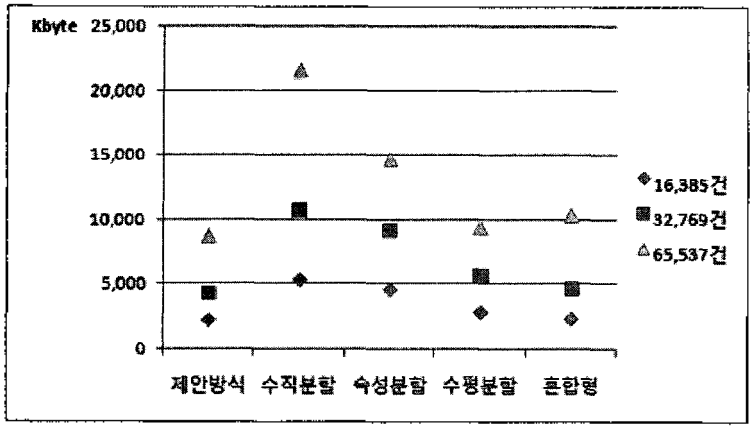
4.2 질의 성능 측정 실험

본 논문에서 제안한 방식의 질의 성능의 효율성을 검증하기 위하여 데이터 정보가 관계 정보보다 많은 경우, 데이터 정보가 관계 정보보다 적은 경우로 구분하여 질의 성능을 비교하였다.

〈그림 15〉는 데이터 정보가 관계 정보보다 많은 경우에 제안 방식과 기존 방식과 질의 성능을 비교한 결과로, 제안 방식이 데이터가 많아질수록 기존 방식보다 질의 성능이 우수한 것을 알 수 있다. 수직분할 방



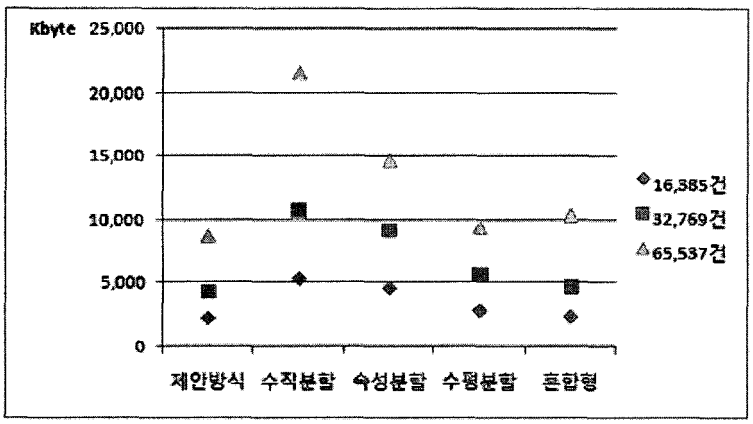
〈그림 14〉 데이터 정보가 관계 정보보다 적은 경우의 저장 용량 비교 결과



〈그림 15〉 관계와 데이터 질의가 결합된 질의의 성능 비교

식은 하나의 릴레이션에 RDF 관계 정보와 데이터 정보까지 저장하는 방식이기 때문에 상대적으로 질의 성능이 떨어졌다. 특히, 데이터가 많을수록 질의 성능은 더 차이가 나는 것을 알 수 있다. 혼합형 방식은 클래스별, 속성별로 릴레이션이 분리되어 조인이 많이 발생하기 때문에 데이터의 중복은 적지만 질의 성능은 떨어진다.

〈그림 16〉은 데이터 정보가 관계 정보보다 적은 경우에 제안 방식과 기존 방식과 질의 성능을 비교 실험한 결과이다. 실험 결과에서 제안 방식과 수평분할 방식과 속성별 테이블 분할방식이 비슷한 성능을 보였다. 그 이유는 제안방식과 수평분할 방식은 상대적으로 적은 조인 연산이 발생하고, 데이터 중복이 적기 때문이다. 그리고 속성별 수평분할 방식은 조인할 테이블은 상대적으로 많으나, 개별적인 테이블의 크기가 작으



〈그림 16〉 규칙과 데이터 질의가 결합된 질의의 성능 비교

므로 조인 연산은 많이 발생하지 않는다. 수직분할 방식은 전체 테이블 스캔을 여러 번 수행하는 관계로 질의 성능이 가장 비효율
적이다. 혼합형 방식은 데이터의 중복은 적지만, 조인 연산이 많이 발생하므로 질의 성능이 떨어진다. 특히 데이터가 많을수록 혼합형 방식과 수직분할 방식은 질의의 성능이 급격하게 저하됨을 알 수 있다.

5. 결 론

최근에 대용량의 RDF 데이터가 늘어남에 따라 효율적인 RDF 저장 방식과 RDF 데이터에 대한 질의 성능 개선이 필요하다. 하지만 RDF 데이터를 저장하기 위한 기존의 연구 기법은 데이터가 중복하여 저장되는 문제로 인해 대용량의 RDF 데이터에 대한 효율적인 관리가 어렵다. 또한 기존 기법은 중복 저장된 데이터로 인해 불필요한 조인 연산이 대량으로 발생하여 질의 성능이 저하되는 문제점이 발생한다.

본 논문에서는 대용량 RDF의 효율적인 저장을 위하여 관계 정보와 데이터 정보를 분리한 저장 구조를 제안하였다. 이를 위하여 트리플 형식의 RDF 문서를 관계 정보와 데이터 정보로 자동적으로 분류하여 관계형 데이터베이스에 저장하기 위한 알고리즘을 제시하였다. 제안 방식은 기존의 저장 방식에 비해 데이터의 중복을 최소화하여 대량의 RDF 데이터를 효율적으로 저장할 수 있었다.

그리고 새로운 관계를 정의하기 위하여 관계형 데이터베이스를 이용한 추론 규칙을 정의하였다. 제안 규칙은 기존의 전방향 추론에서 추론 결과를 미리 저장하는 방식과 달리 질의 처리시에 추론할 수 있는 방법이다. 규칙은 관계형 데이터베이스에서 제공하는 함수를 이용하여 생성하였다. 특히, 본 논문에서 제안한 저장 방식을 이용하여 트리플 형태의 관계 정보 릴레이션과 데이터 정보 릴레이션에서 필요한 데이터를 분리 검색하여 결합하는 방식에 의해 RDF 데이터에 대한 질의 성능을 개선할 수 있었다.

본 연구에서 제안한 기법의 효율성을 검증하기 위하여 Oracle 10g Spatial Resource Description Framework에 소개된 Family RDF 스키마를 이용하여 실험용 RDF 데이터를 생성하였다. 그리고 Family RDF를 대상으로 제안 기법과 기존의 RDF 저장 기법간의 저장 공간과 질의 성능에 대한 비교 실험을 통하여 제안 기법의 효율성을 입증하였다.

실험 결과, 본 논문에서 제안한 방식의 저장 용량은 가장 널리 쓰이는 Jena, Kaon의 수직분할 저장 방식보다 평균 65% 정도 적은 용량을 차지하였고, 데이터의 크기가 증가할수록 저장 효율이 더욱 개선되었다. 질의 성능에 대한 비교 실험에서도 제안 방식은 기존 방식보다 우수한 성능을 나타내었다. 특히 데이터가 증가할수록 기존 방식보다 더욱 우수한 질의 성능을 보였다. 이는 제안 방식의 저장 구조가 데이터 중복을 최소화하여 질의 처리의 부하를 최소화할 수 있기 때문이다.

본 연구 결과는 RDF를 이용한 전자상거

래, 시맨틱 웹, 지식관리 등과 같은 분야에서 대량의 RDF 데이터를 효율적으로 저장 관리할 수 있는 기반 기술로 사용할 수 있으리라 기대된다. 또한 본 연구 결과는 실시간 검색을 요구하는 전자상거래나 정보 검색 분야에서 RDF 데이터에 대한 질의 성능을 개선할 수 있는 우수한 기법이라 사료된다.

참 고 문 헌

[1] 김연희, 시맨틱 웹 환경에서 메타데이터와 RDF 정보의 저장 및 인덱싱 기법, 홍익대학교 박사학위논문, 2006.

[2] 오삼균, "국가지식정보자원관리를 위한 시맨틱웹 설계 및 정책 방향에 관한 연구," 한국비블리아학회, 제15권, 제1호, pp. 43-67, 2004.

[3] 이미경, 김평, 정한민, 성원경, "OntoThink-K®: DBMS 기반의 RDF 확장 모델," 한국정보과학회, 제33권, 제2호 (B), pp. 284-288, 2006.

[4] 정한민, 강인수, 이미경, 이승우, 성원경, "OntoThink-K®: DBMS기반 추론 서비스," 한국정보과학회, 제33권, 제2호 (B), pp. 200-204, 2006.

[5] 정호영, 김정민, 정준원, 김종남, 임동혁, 김형주, "RDF 문서의 저장소와 RDQL 질의 처리기의 설계 및 구현," 한국정보과학회, 제33권, 제4호, pp. 363-371, 2006.

[6] 탁경현, 김학수, 차현석, 손진현, "효율적인 OWL 시맨틱 정보 처리를 위한 데이터베이스 스키마 설계 및 분석," Proc. Korean Database Conference, 제20권, 제2호, pp. 44-51, 2004.

[7] C. Murray, "Oracle® Spatial Resource Description Framework (RDF) 10g Release 2 (10.2) Manual," Oracle Corporation, 2005.

[8] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-schema-20040210, 2004.

[9] D. Connolly, F. Hamelen, I. Horrocks, D. McGuinness, P. Patel-Schneider and L. Stein, "Annotated DAML+OIL Ontology Markup," W3C, http://www.w3.org/TR/daml+oil-walkthru, 2001.

[10] D. L. McGuinness and F. Harmelen, "OWL Web Ontology Language Overview," W3C Recommendation, http://www.w3.org/TR/owl-features, 2004.

[11] F. Manola and E. Miller, "RDF Primer," W3C Recommendation, http://www.w3.org/TR/rdf-primer, 2002.

[12] Ivan Herman, "Semantic Web," W3C Semantic Web Activity, http://www.

w3.org/2001/sw, 2001.

[13] J. Broekstra, A. Kampman and F. Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," Proc. Int'l Conf. on ISWC, Vol. 2342, pp. 54-68, 2002.

[14] K. Stoffel, M. Taylor and J. Hendler, "Efficient Management of Very Large Ontologies", Proc. Int'l Conf. on AIC, pp. 442-447, 1997.

[15] K. Wilkinson, C. Sayers, H. Kuno and D. Reynolds, "Efficient RDF Storage and Retrieval in Jena2," Proc. VLDB Workshop on Semantic Web and Databases, pp. 131-150, 2003.

[16] R. Agrawal, A. Somani and Y. Xu, "Storage and Querying of E-Commerce Data," Proc. VLDB, pp. 149-158, 2001.

[17] R. Volz, D. Oberle, B. Motik and S. Staab, "KAON SERVER-A Semantic Web Management System," Proc. Twelfth World Wide Web Conference, pp. 20-24, 2003.

[18] S. Alexaki, V. Christophides, G. Karvounarakis and D. Plexousakis, "On Storing Voluminous RDF Description: The case of Web Portal Catalogs," Proc. ACM SIGMOD Workshop on the Web and Databases Conference, pp. 24-25, 2001.

[19] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis and K. Toll, "The RDFSuite: Managing Voluminous RDF Description Bases," Technical Report, ICS-FORTH, 2001.

[20] T. R. Gruber, "A Translation Approach to Portable Ontologies," Technical Report, Knowledge Acquisition, 1993.

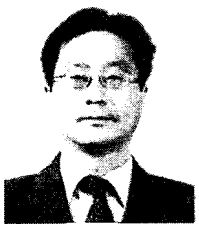
[21] Z. Pan and J. Heflin, "DLDB: Extending Relational Databases to Support Semantic Web Queries," Proc. Int'l Conf. on ISWC, pp. 109-113, 2003.

저 자 소 개

문현정 (E-mail : mun@changwon.ac.kr)
1994. 한국방송대학교 전자계산학과 졸업 (이학사)
1996. 창원대학교 일반대학원 전자계산학과 졸업 (이학석사)
2003. 창원대학교 일반대학원 컴퓨터공학과 졸업 (공학박사)
2004 ~ 현재 창원대학교 연구교수
관심분야 KDD, 온톨로지마이닝, 시맨틱웹

성정환 (E-mail : jhsung@kicotech.com)
2003. 창원대학교 불문학과 졸업 (문학사)
2007. 창원대학교 일반대학원 컴퓨터공학과 졸업 (공학석사)
2007 ~ 현재 키코테크 연구원
관심분야 온톨로지 언어, 온톨로지 저장 구조

김영지 (E-mail : yjkim@hibrain.net)
1997. 창원대학교 전자계산학과 졸업 (이학사)
1999. 창원대학교 일반대학원 전자계산학과 졸업 (이학석사)
2004. 창원대학교 일반대학원 컴퓨터공학과 졸업 (공학박사)
2004 ~ 2006. 고신대학교 초빙교수
2007 ~ 현재 하이브레인넷 책임연구원
관심분야 온톨로지마이닝, 추천모델, e-Learning



우용태 (E-mail : ytwoo@changwon.ac.kr)
1982. 경북대학교 전자공학과 졸업 (공학사)
1984. 경북대학교 일반대학원 전자공학과 졸업 (공학석사)
1995. 경북대학교 일반대학원 전자공학과 졸업 (공학박사)
1987 ~ 현재 창원대학교 컴퓨터공학과 교수
관심분야 데이터마이닝, 온톨로지마이닝, 시맨틱웹