분산 환경을 고려한 OWL 문서의 저장 및 인덱싱 기법

김용욱[0], 김연희, 임해철
홍익대학교 컴퓨터공학과
{yongwook[0], kyh, lim}@cs.hongik.ac.kr

Storing and Indexing Schemes for OWL Documents in Distributed Environment

Yong Wook Kim[0], Youn Hee Kim, Hae Chull Lim
Dept. of Computer Engineering, Hong Ik University

요 약

정보의 단순한 연결뿐만 아니라 의미적인 연결 관계를 표현하는 시맨틱 웹에서 RDF와 RDFS만으로는 정보간의 다양한 의미적 관계를 나타내기가 힘들다. 따라서 정보간의 의미적 관계를 보다 명확하게 필요로 하는 분산 환경에서는 시맨틱 웹 언어의 표준으로 인식되고 있는 OWL로 표현된 시맨틱 웹 데이터를 위한 저장 구조가 필요하다. 따라서 본 논문에서는 분산 환경을 고려하여 OWL이 표현하는 다양한 의미적 관계를 이용한 질의 처리를 지원하는 저장 구조를 제안한다. 그리고 OWL에 정의된 클래스 간의 계층 관계를 이용한 질의의 효율적인 처리를 위한 인덱스 구조와 처리 전략을 제안한다.

1. 서 론

초기 인터넷 사용자들은 질의를 할 때 단지 많은 정보를 얻으면 만족했다. 하지만, 최근에는 정보의 범람으로 인해 사용자는 정보의 양보다 정확한 정보 검색을 원하게 되었다. 이와 같은 요구에 부합하기 위해 등장한 개념이 시맨틱 웹이다. 시맨틱 웹은 현재 웹의 확장으로서 사람과 컴퓨터가 이해할 수 있는 언어로 표현되어 사람과 컴퓨터가 협력하여 작업할 수 있는 환경을 제공해 준다[1]. 시맨틱 웹을 위해서는 기계가 의미를 이해할 수 있도록 온톨로지를 먼저 구축을 하는 것이 필요하다. 온톨로지는 자원의 개념과 자원간의 개념을 정의해 놓은 일종의 사전으로서 다양한 온톨로지 기술 언어를 이용하여 정의할 수 있다. 현재 가장 일반적인 기술 언어는 RDF(Resource Description Framework)이다. RDF는 메타데이터를 기술하기 위해 W3C에서 제안된 표준으로서, 데이터의 의미를 자원-속성-값이라는 트리플 구조로 표현해 준다[2]. 또한 RDFS는 RDF만으로는 표현할 수 없는 클래스와 클래스 간의 관계를 정의해 주기 때문에 RDF와 같이 사용한다[3]. 그러나 자원간의 동일성과 이질성, 집합 관계나 제한 조건은 RDF와 RDFS의 방법으로는 표현을 할 수 없다. 이에 W3C는 더욱 다양하고 자세한 개념을 표현할 수 있는 기술 언어인 OWL(Web Ontology Language)을 표준 온톨로지 언어로 제안하였다[4]. 하지만 아직까지 질의 처리를 위한 연구는 RDFS를 중심으로 이루어지고 있고 질의 방법도 키워드나 트리플 구조에 기반한 단순 질의 처리를 대상으로 하는 경우가 많다. 따라서 사용자의 다양한 요구에 대답하기 위해서는 키워드나 단순 트리플 질의뿐만 아니라 클래스로 이루어지는 질의를 효율적으로 처리하는 저장과 인덱스 구조가 필요하다. 또한 현재의 인터넷 환경에서는 분산 환경을 고려한 저장이 필요하다. 분산 환경은 네트워크 안에 있는 온톨로지 정보를 내 자신이 가진 정보처럼 사용할 수 있다. 하지만 온톨로지 정보는 다양한 사람이 만들기 때문에 같은 개념을 표현하더라도 사람마다 다르게 표현 할 수 있다. 따라서 분산 환경에서는 다양한 온톨로지를 가져올 때 동음이의어와 이음동의어 처리가 필요하다. 본 논문에서는 클래스와 관련된 질의를 효율적으로 처리하기 위한 OWL 기반의 저장 구조와 함께 인덱스 구조를 제안한다.

본 논문의 구성은 다음과 같다.

2장에서는 기존에 제한된 RDF/S의 저장구조와 인덱스 구조를 알아본다. 3장에서는 분산 환경에서 일어날 수 있는 동일성과 이질성의 문제를 해결하기 위해 OWL에서 고려해야 될 요소를 알아본다. 4장에서는 3장에서 제시한 OWL의 요소를 고려하여 저장 구조를 제안하고, 5장에서는 저장 구조와 함께 클래스 질의를 효율적으로 처리를 위한 인덱스 구조를 제안한다. 6장에서는 본 논문에서 제안 저장 구조와 인덱스 구조에 대한 질의 처리 전략을 설명하고 7장에서 이 논문의 결론을 맺는다.

2. 관련 연구

2.1 RDF/S를 위한 저장 기법

XML에서 제안된 구간 기반 베이스로 된 레이블링 스키마 기법들은 (시작위치, 끝 위치) 값을 노드의 순회를 통해 저장하고 계층 질의 시에 노드 간에 두 값을 비교하는 방법으로 계층 정보를 묻는 질의에 대해 간단히 처리를 해줄 수 있다[5]. 하지만, 데이터가 추가 될 때마다 전체적으로 순회를 다시 하여 레이블링을 해야 되기 때문에 자주 정보가 추가 되는 환경에서는 효율 면에서 떨어진다. 비록 공간에 여유를 두고 레이블링을 한다면 이 문제는 해결 될 수도 있지만, 어느 정도로 여유 저장 공

본 연구는 한국과학재단 특정기초연구(과제번호 : R01-2004-000-10586-0(2006))의 지원으로 수행되었음

간을 확보해야 될지 계산하는 일은 쉬운 일이 아니다 [5]. 이에 제안된 방법이 <그림 1>에서 보여 지는 소수 넘버 레이블링 방법이다. 이 방법은 각각의 노드에 고유한 소수 값을 부여하고, 노드와의 관계에 따라 부모의 소수 레이블 값에 자신의 고유한 소수 값을 곱한 값을 소수 레이블 값으로 저장을 하는 방법으로, 두 노드간의 관계 질의 시 단순히 두 노드의 소수 레이블 값만 비교를 하여 나눈 값의 몫이 0이라는 것으로 두 노드가 이어진 노드인지를 구할 수 있다. 이 방법의 장점은 새로운 노드의 삽입과 삭제 시에 다른 노드의 레이블을 변경할 필요 없이 변경된 노드의 레이블 정보만 추가(삭제)하는 방법으로 레이블링이 효율적으로 이루어진다는 것이다. [6]에서는 이를 이용하여 RDF의 트리플 기반의 저장 구조에 적용하였다. 하지만 논문에서 고려했던 리소스의 연결 상태의 질의는 중요한 질의 유형이 아니다. 또한 RDF와 함께 RDFS도 고려해야 시맨틱 웹의 장점을 살릴 수 있지만, [6]에서는 RDFS까지 고려한 질의는 고려하지 않았다는 점에서 한계가 있다. 또한 분산 환경에서는 인스턴스와 클래스의 명칭이 온톨로지마다 다를 수 있기 때문에 인스턴트 노드간의 관계를 이용한 질의 시에 순환적으로 연결된 인스턴스 정보보다는 동음이의어와 이음동의어 등의 의미적 관계 정보를 알려 줄 수 있어야 한다. 따라서 분산 환경과 관련된 OWL요소를 저장하기 위한 새로운 저장 방법과 함께 RDF와 RDFS를 동시에 고려한 질의 유형의 처리가 필요하다.

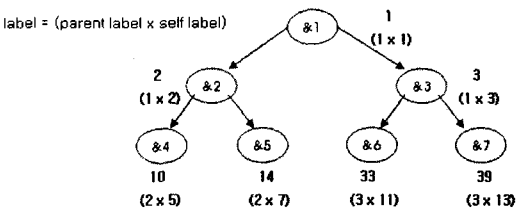

그림 1. 소수 넘버 레이블링의 예

2.2 RDF/S를 위한 인덱싱 기법

기존의 인덱스 방법은 주어, 서술어, 목적어의 구조로 이루어진 질의를 처리하기 위해서 주어와 서술어 목적어를 키로 갖는 인덱스를 각각 만들고 질의를 처리하기 위해서 3가지 인덱스의 결과를 조인하는 방법을 사용하였다. 하지만 이 방법의 경우는 트리플 질의 시 각각의 인덱스로 별로 디스크의 검색이 일어나기에 질의 처리가 효율적이지 못하다. YARS(Yet Another RDF Store)[7]의 검색 구조는 질의를 QUADS구조로 S(주어), P(서술어), O(목적어), C(문서출처) 형태로 조합된 질의 패턴을 처리 하기 위한 인덱스이다. 기본적인 트리플 요소에 문서 출처를 고려한 이유는 분산 환경을 고려하기 위해서이다. <표 1>은 YARS에서 제안한 질의 형태로써 트리플 구조에서 나타날 수 있는 질의 패턴 함께 이를 처리할 수 있는 Index구조를 6가지로 나타냈다. 제안된 인덱스는 B+ Tree형태로 따로 따로 키를 가질 필요 없이 한 번의 검색으로 S, P, O, C의 검색이 가능하다. 하지만 인덱스의 크기가 너무 크기 때문에 자주 쓰이는 인덱스를 줄이는 방법이 필요하다. 또한 RDF 구조로 질의 처리를 하기 때문에 RDFS를 고려하지 않았다. 비록 OWL도 QUADS인덱스로 처리는 가능하다고 하지만, [7]에서는 스키마에 정의된 클래스나 클래스간의 의미적 관계를 고려하지 않고 RDF 인스턴스와 함께 기본적인 트리플 구조를 기반으로만 질의를 처리하는 한계가 존재한다.

표 1. QUAD 인덱스를 위한 질의 형태의 구분

Index	Quary partterns
SPOC	(?, ?, ?, ?), (s, ?, ?, ?), (s, p, ?, ?), (s, p, o, ?), (s, p, o, ?), (s, p, o, c)
CP	(?, ?, ?, c), (?, p, ?, c)
OCS	(?, ?, o, ?), (?, ?, o, c), (s, ?, o, c)
POC	(?, p, ?, ?), (?, p, o, ?), (?, p, o, c)
CSP	(s, ?, ?, c), (s, p, ?, c)
OS	(s, ?, o, ?)

* ?는 구하고자 하는 변수를 의미함

3. 분산 환경을 지원하기 위한 OWL의 요소

본 논문에서는 OWL이 점차 표준화가 되어가고 있고, OWL로 표현할 수 있는 많은 의미적 관계에 대한 연구가 아직 미흡하기에 이를 처리할 수 있는 저장 구조를 제안한다. 분산 환경에서는 스키마 정보간의 의미적 관계를 보다 명확하게 하는 것이 중요하지만 RDF/S로는 부족하다. <표 2>는 OWL의 여러 관계 중 분산 환경에서 고려되어야 할 요소를 정리한 것이다. 따라서 본 논문에서는 OWL의 요소 중 분산 환경에서 여러 온톨로지를 사용할 때 일어날 수 있는 동일성과 이질성의 문제를 해결할 수 있는 요소를 RDF/S의 확장을 위해 고려한다.

표 2. 분산 환경을 위한 OWL의 확장 요소 분류

구분	Property	설명
동일성 (이음동의어 처리)	equivalentClass	다른 온톨로지에서 유래된 두 클래스가 동치임을 표현
	equivalentProperty	다른 온톨로지에서 유래된 두 속성이 동치임을 표현
	sameAs	다른 온톨로지에서 유래된 두 리소스가 동치임을 표현
이질성 (동음이의어 처리)	disjointWith	다른 온톨로지에서 유래된 두 클래스의 구성 요소는 다르다는 것을 정의
	differentFrom	다른 온톨로지에서 유래된 두 리소스가 서로 다르다는 것을 정의
	AllDifferent	다른 온톨로지에서 유래된 여러 개의 리소스가 서로 다르다는 사실을 한 번에 기술

분산 환경에서는 <표 2>와 같은 요소들의 고려가 필요하기에 앞으로 본 논문에서는 저장을 할 때와 질의 처리를 할 때 <표 2>에 나타낸 요소를 고려하겠다.

4. 분산 환경을 고려한 OWL의 저장 구조

클래스 이름으로 질의를 할 때 고려되어야 점은 상위 클래스에 질의를 할 때 이와 연결된 하위 클래스에도 같은 질의를 수행해야 된다는 점이다. <그림 2>는 클래스의 계층 정보가 표현된 OWL 예제이다. 예를 들어 "Bevarage에 StEmilion이 속하는가?"와 같이 클래스 Beverage를 통한 질의를 했을 때, Bevarage의 인스턴스 중에는 질의를 만족하는 인스턴스는 없지만, 간접적으로 와인의 인스턴스 정보도 Bevarage에 포함한다고 말할 수 있다. 따라서 질의를 처리 할 때에서 연결된 서브클래스도 고려해주어야 한다.

```
<owl:Class rdf:ID="Beverage" />
<owl:Class rdf:ID="liquor" >
  <rdfs:subClassOf rdf:resource="# Beverage ">
</owl:Class>
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="#liquor">
</owl:Class>
<owl:Class rdf:ID="WineColor" />
<owl:Class rdf:ID="Red" />
<owl:ObjectProperty rdf:ID="hasColor">
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineColor" />
</owl:ObjectProperty>

<Wine rdf:ID="StEmilion ">
  <hasColor rdf:resource="#Red" />
</Wine>
```

그림 2. OWL 문서 예제

name	selfLabel	primeLabel
Beverage	1	1
Liquor	2	2
Wine	3	6

그림 3. 소수 레이블을 이용한 클래스 구조

보통의 저장 구조에서는 이를 처리해주기 위해서 상속관계를 subclassof와 subpropertyof로 따로 테이블을 만들어서 처리하지만, 상속 관계가 이행적인 특성을 가지고 있기에 계층 관계를 이용한 질의 처리 속도가 상대적으로 느렸다. 예를 들어 <그림 2>의 예제에서 Beverage의 서브 클래스를 구한다고 생각해 보자. 이를 구하기 위해서는 먼저 Beverage의 서브클래스에 liquor가 있다는 것을 구하고 다시 liquor의 서브클래스는 Wine임을 구하는 식으로 반복적인 과정을 거쳐야 한다. 따라서 이를 간단하게 위해 <그림 3>에서 볼 수 있듯이 상속관계가 일어날 수 있는 클래스와 속성에 소수 레이블링 기법을 적용한다면 Beverage의 소수 레이블링 값으로 나눠지는 클래스가 Liquor와 Wine이므로 직접 간접적으로 연결된 하위 클래스를 테이블 검색 한번으로 구할 수 있다.

이와 같이 제안된 저장 방법에서는 계층질의를 효율적 처리를 위해 소수레이블링 기법을 이용하고, 저장의 효율성을 위해서 <그림 3>에서와 같이 클래스마다 배정된 고유한 소수 값을 클래스, 속성에 대한 식별자로 <그림 5>와 같이 이용한다. 또한 domain과 range 질의와 [표 2]에서 나타난 동음이의어를 효율적으로 처리해 주기 위한 고려되어야 될 요소 위해 테이블을 각각 만들었다.

단, 이음동의어의 경우 사용자가 질의 시 계층 질의와 같이 고려해주어야 되는 요소이기 때문에 식별자를 동일하게 부여해주고, 같은 식별자를 가진 값은 네임스페이스를 함께 이용하여 각각을 구분해준다.

```
<rdf:RDF
   xmlns    = "http://www.example.org/wine#"
   xmlns:vin = "http://www.example.org/wine2#">

  <owl:Class rdf:ID="Wine">
    <owl:equivalentClass rdf:resource="&vin;Wines ">
  </owl:Class>

</rdf:RDF>
```

그림 4. equivalentClass속성을 표현한 OWL 문서

<그림 4>는 실제로 equivalentClass의 속성을 OWL로 표현한 예이다. 예에서는 클래스 이름이 Wine과 Wines로 다르고 네임스페이스가 다른 구분되는 연관관계가 없어 보이는 두 클래스가 실재로는 같다는 정보를 나타내준다. <그림 5>는 소수 레이블을 이용한 클래스 테이블 구조에 equivalentClass속성을 나타내기 위해 저장한 예제이다. 이를 통해 검색 시에 단순히 cid만을 비교함으로써 Wine클래스와 Wines클래스가 같다는 것을 알 수 있다. <그림 6>은 네임스페이스 테이블의 저장 예이다. 네임스페이스는 문서의 출처에 따라 식별자를 고유하게 부여하기 때문에 클래스와 속성과 리소스를 출처에 따라 고유하게 구분할 수 있는 기능을 지원해주도록 해 준다.

cid	ns	name	primeLabel
2	1	Wine	2
2	2	Wines	2

그림 5. equivalentClass 테이블의 저장 예

nid	namespace
1	http://www.example.org/wine#
2	http://www.example.org/wine2#

그림 6. namespace 테이블의 저장 예

그리고 테이블에서 클래스와 속성과 리소스가 쓰이는 경우 각각이 이름을 통해서는 구분이 되지 않기 때문에 네임스페이스 값을 같이 저장해준다. 추가적으로 리소스 테이블에는 타입정보를 저장하도록 해서, 선언 리소스가 속하는 클래스 정보를 알 수 있도록 했다.

마지막으로 리소스 레벨에서의 트리플 관계를 저장하기

위한 statement테이블이 존재한다.

<그림 7>는 4장에서 표현한 테이블들을 표현한 관계형 데이터베이스 스키마이다.

하지만 <그림 7>와 같은 관계형 데이터베이스 스키마에서는 리소스 간의 관계를 질의할 때 statement를 전부 찾아야 되기 때문에 질의 처리의 효율이 떨어진다. 또한 클래스이름을 통해 만족하는 리소스를 구할 때 리소스 테이블을 순회해야 되는 문제가 생긴다. 따라서 다음 장에서는 클래스 이름을 통한 질의 시 효율적인 질의처리를 위한 인덱스 구조를 제안한다.

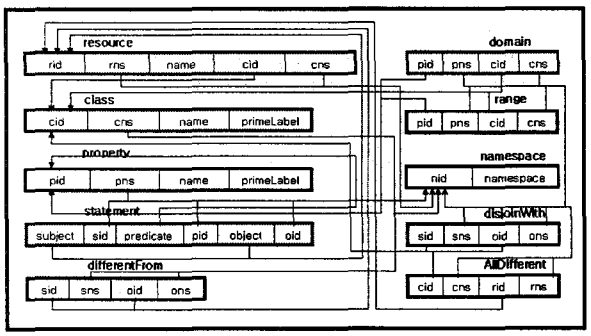

그림 7. OWL 문서를 위한 제안 저장 구조

5. 클래스를 이용한 질의 처리를 지원하는 인덱스 구조

4장의 저장 구조는 스키마 레벨에서의 계층 질의와 분산 환경에서 필요한 OWL요소와 관계된 질의를 효율적으로 처리하기 위해서 제안되었다면, 5장은 클래스를 통해 발생할 수 있는 질의를 효율적으로 처리하기 위해 필요한 두 가지 형태의 인덱스 구조를 제안한다.

이를 위해 YARC에서 제안된 인덱스를 변형이 필요하다. 왜냐하면 YARC의 경우 단순히 RDF 인스턴스와 스키마를 구분하지 않고, 인덱스로 만들었기 때문에 인덱스의 크기가 상대적으로 크고, 인덱스의 크기가 큰 만큼 처리속도도 느리기 때문이다. 그리고 YARC 인덱스에 고려한 문서 정보의 경우 이미 제안된 저장 구조의 네임스페이스를 통해 고려되었기 때문에 인덱스에서는 고려하지 않아야한다. 따라서 YARC의 인덱스가 스키마 레벨에서 효율적인 질의 처리와 제안된 저장 구조에 적용이 가능하기 위해서 YARC의 변형을 통한 확장이 필요하다. 본 논문에서는 트리플을 기반으로 한 질의 유형을 처리하기 위한 인덱스를 <표 3>과 같이 3가지로 정의하고, 정의된 인덱스 방법을 기반으로 리소스 레벨에서 필요한 인덱스와 클래스와 리소스의 연관 질의에 필요한 인덱스를 제안한다.

먼저 리소스 레벨에서 선언된 트리플구조를 통해 SPO, OS, PO인덱스를 구축한다. 이를 통해 RDF 인스턴스 레벨에서의 질의를 효율적으로 처리할 수 있고, 리소스 레벨에서 선언된 트리플구조만을 인덱스로 만들기 때문에 YARC에 비해 인덱스 크기가 작다. 또한 인덱스의 키가 리소스와 속성과 클래스의 식별자로 이루어지기 때문에 식별자가 같은 이음동의어의 처리가 가능하다는 장점이 있다.

다음으로 스키마 요소 중 클래스와 리소스의 연관 질의를 위한 인덱스를 제안한다. 클래스와 리소스의 연관질의 인덱스는 트리플구조 중 속성의 값이 type인 요소만 고려하기에 인덱스의 크기도 작고, 클래스와 인스턴스의 연관질의 시에 질의 처리속도를 효율적으로 처리할 수 있기에 필요하다. 인덱스 형태와 특성은 리소스 레벨에서의 인덱스 같지만 속성 값이 type인 트리플 구조로만 인덱스를 만들기 때문에 속성을 질의하는 PO인덱스는 사용하지 않는다.

제안된 인덱스는 공통적으로 리소스와 속성과 클래스 이름으로 이루어 진 B+ 트리로 인덱스 셑을 만들고, 순차 셑은 인덱스 셑의 정보에 추가적으로 각각의 네임스페이스 정보를 가지고 있어 같은 식별자를 가지고 있는 있더라도 네임스페이스에 따라 구분이 가능하도록 해줬다.

이처럼 제안된 인덱스 구조는 클래스와 연관된 질의 시 저장구조와 함께 사용되어 테이블을 직접 찾을 때 발생할 수 있는 조인 연산을 줄일 수 있다. 다음 장에서는 4장과 5장에서 살펴본 제안된 저장구조와 인덱스 구조를 클래스 질의에 활용할 때 필요한 질의 처리 전략에 대해 알아본다.

표 3. 질의 형태와 인덱스 구조

Index	Quary partterns
SPO	(?,?,?), (s,?,?), (s,p,?), (s,p,o)
OS	(?,?,o), (s,?,o)
PO	(?,p,?), (s,p,o)

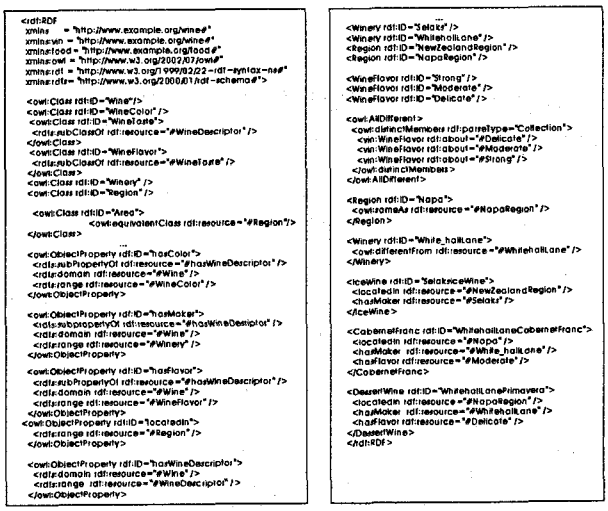

그림 8. OWL로 작성된 와인 문서

6. 질의 처리 전략

이 장에서는 제안된 저장구조와 인덱스를 이용해 클래스와 인스턴스가 연결된 질의 형태와 처리 방법을 알아본다. 실제 문서를 본 논문에서 제안한 저장 구조와 인

덱스 구조에 적용하기 위해서 <그림 8>와 같이 OWL로 정의된 와인 문서를 이용했다.

<그림 9>, <그림 10>, <그림 11>은 <그림 8>의 와인 문서를 제안한 저장 구조와 클래스 이름으로 연결된 속성과 리소스를 질의할 때 질의 처리를 효율적으로 하기 위해 필요한 2개의 인덱스이다. 제안된 인덱스는 기존의 인덱스보다 트리플 질의 시에 조인을 줄여주는 장점이 있다. 예를 들어, "WhitehallLaneCabernetFranc의 맛은 어떤가?"라는 질의를 처리 원한다면, 이전의 인덱스 구조에서는 S가 WhitehallLaneCabernetFrance인 요소를 모두 구하고, P가 hasFavor인 요소를 모두 구한다음 조인을 해야 한다. 하지만 제안된 방법은 질의의 형태를 저장 구조의 테이블을 통해 s,p,o의 형태로 (7,5,?)로 나타낸 뒤 <그림 10>의 B+인덱스를 이용해 간단하게 (7,5,10)를 구할 수 있다. 이처럼 제안된 인덱스는 간단한 질의에도 효율적인 수행속도를 보여준다.

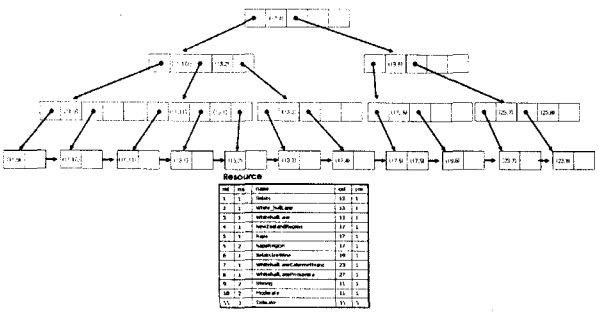

그림 9. 와인 문서에 적용한 확장된 클래스-리소스 연관 OS 인덱스

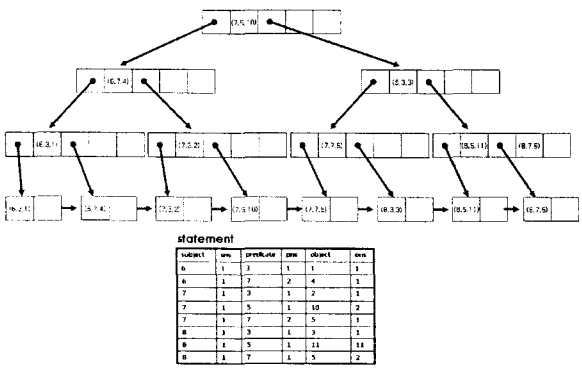

그림 10. 와인 문서에 적용한 리소스 레벨의 확장된 SPO 인덱스

클래스 이름으로 질의할 수는 가장 복잡한 형태인 연결된 속성의 리소스를 묻는 질의 처리과정을 정리하면 다음과 같다.

1단계: 소수레이블링을 이용하여 서브클래스와 서브속성이 있는지를 알아보면서 클래스와 속성를 보고 동음이의어를 제외한 나머지 집합을 뽑아낸다.(동음이의어 고려)

2단계: 클래스-리소스 정보로 이루어진 OS 인덱스를 통해 클래스의 인스턴스들을 뽑아낸다. (이음동의어 고려)

3단계: 리소스 레벨의 (S,P,O)인덱스를 이용하여 클래스와 관련된 리소스와 속성으로 연결된 리소스를 뽑아낸다. (이음동의어 고려)

Class

cid	cns	name	primeLabel
2	1	Wine	2
3	1	WineColor	30
5	1	WineDescriptor	10
7	1	WineTaste	70
11	1	WineFlavor	770
13	1	Winery	13
17	1	Region	34
17	2	Area	34
19	1	IceWine	38
23	1	CabernetFranc	46
27	1	DessertWine	54
29	1	RedWine	58

Resource

rid	rns	name	cid	cns
1	1	Selaks	13	1
2	1	White_hallLane	13	1
3	1	WhitehallLane	13	1
4	1	NewZealandRegion	17	1
5	1	Napa	17	1
5	2	NapaRegion	17	1
6	1	SelaksIceWine	19	1
7	1	WhitehallLaneCabernetFranc	23	1
8	1	WhitehallLanePrimavera	27	1
9	2	Strong	11	1
10	2	Moderate	11	1
11	2	Delicate	11	1

Property

pid	pns	name	Prime-Label
2	1	hasColor	22
3	1	hasMaker	33
5	1	hasFlavor	55
7	1	locatedIn	7
7	2	placedIn	7
11	1	hasWineDescriptor	11

statement

subject	sns	predicate	pns	object	ons
6	1	3	1	1	1
6	1	7	2	4	1
7	1	3	1	2	1
7	1	5	1	10	2
7	1	7	2	5	1
8	1	3	1	3	1
8	1	5	1	11	11
8	1	7	1	5	2

namespace

nid	namespace
1	http://www.example.org/wine#
2	http://www.example.org/wine2#
3	http://www.w3.org/2002/07/owl#
4	http://www.w3.org/1999/02/02-rdf-syntax-ns#
5	http://www.w3.org/2000/01/rdf-schema#

disjoinWith

sid	sns	oid	Ons
17	1	27	1

AllDifferent

cid	cns	rid	rns
11	1	9	1
11	1	10	1
11	1	11	1

domain

pid	pns	cid	cns
2	1	2	1
3	1	2	1
11	1	2	1

range

pid	pns	cid	cns
2	1	3	1
3	1	13	1
5	1	11	1
7	1	17	1
7	2	17	1
11	1	5	1

differentFrom

sid	sns	oid	ons
2	1	3	1

그림 11. 와인 문서를 실제로 저장한 저장 구조

위와 같은 처리과정을 고려한 질의 처리과정의 예를 살펴보면 다음과 같다.

질의 예제) "IceWine을 만드는 회사를 검색하라"

질의 예제는 클래스이름과 속성 이름을 가지고 연결된 목적 리소스를 구하는 질의이다. 먼저 제안된 저장 구조의 클래스 테이블과 속성 테이블을 통해 소수 레이블링을 이용하여 서브클래스와 서브프로퍼티가 존재하는지의 여부를 확인할 수 있다. 예제 질의에서는 질의에서 사용된 IceWine, hasmake의 서브클래스와 서브프로퍼티 관계를 만족하는 것이 없음을 소수 레이블의 비교를 통해

알 수 있다. 이 과정에서 만약 서브클래스나 서브프로퍼티 정보가 있다면 조합의 집합을 함께 속성과 클래스 이름을 해당하는 식별자로 변환시켜주고 서브 관계를 만족하는 집합이 없다면 해당하는 속성과 클래스를 식별자로 바꿔준다. 다음으로 클래스 값으로 제안된 이용하여 연결된 리소스 값을 구한다. 여기서 클래스 IceWine의 리소스 값으로 SelaksIceWine이 있다는 것을 알 수 있다. 다음으로 리소스 레벨에서의 인덱스 구조를 통해 IceWine을 만드는 회사는 Selaks이라는 것을 구할 수 있다.

7. 결론

시맨틱 웹에서 RDF와 RDFS만으로는 정보간의 다양한 의미적 관계를 나타내기가 힘들다. 정보의 단순한 연결뿐만 아니라 의미적인 연결 관계를 표현하는 시맨틱 웹에서 RDF와 RDFS만으로는 정보간의 다양한 의미적 관계를 나타내기가 힘들다. 따라서 정보간의 의미적 관계를 보다 명확하게 필요로 하는 분산 환경에서 필요한 시맨틱 웹 언어의 표준으로 인식되고 있는 OWL로 표현된 시맨틱 웹 데이터를 위한 저장 구조를 제안했다. 또한 클래스와 연관된 질의 처리를 효율적으로 하기 위해서 소수 레이블링을 이용하여 클래스와 속성의 상속 질의를 수행하도록 디자인하였고, 클래스와 연관된 인스턴스 질의를 효율적으로 하기 위해서 YARS에서 제안된 인덱스 구조를 확장시켜 제안된 저장 구조에 적용시켰다. 향후에는 보다 실용적인 온톨로지에 문서에 제안한 저장구조와 인덱스 구조를 적용한 OWL 저장 시스템을 구축하는 연구가 필요하다.

참고문헌

[1] T.Berners-Lee, J. Hender, O. Lassila. "The Sematic Web", Scientific American, May 2001.

[2] W3C, "RDF Primer", 2004. 2.

[3] RDF Vocabulary Description Language 1.0 : RDF Schema, http://www.w3.org/TR/rdf-schema.

[4] W3C, OWL Web Ontology Language Overview, http://www.W3C.org/TR/owl-features/

[5] Xiaodong Wu, Mong Li Lee, Wynne Hsu, "A Prime Number Labeling Scheme for Dynamic Ordered XML Trees", Proceedings of the 20th International Conference on Data Engineering(ICDE'04), 2004.

[6] 김선영, 권동섭, 이석호, "소수 레이블을 이용한 RDF/RDFS 인덱스 구조", 한국컴퓨터종합학술대회 논문집, 32권 1호, 2005.

[7] Andreas Harth, Stefan Decker, "Optimized Index Structures for Querying RDF from the Web", http://sw.deri.org/2005/02/dexa/yars.rdf.

[8] 고명석, 김연희, 김병곤, 임해철, "OWL 문서의 저장과 질의 형태에 관한 연구", 한국정보과학회 가을 학술발표논문집, 31권, 2호, 2004.