OWL 질의 처리를 위한 시그너처 기반 최적화 기법

(An Optimization Technique based on Signatures for OWL Query Processing)

임 동 혁 † 정 호 영 † 김 형 주 ††
(Donghyuk Im) (Hoyoung Jeong) (Hyoung-Joo Kim)

요 약 시맨틱 웹은 차세대 웹으로 연구되고 있으며 시맨틱 웹 상에서는 사람이 아닌 컴퓨터가 이해할 수 있는 정보를 처리해야 한다. 이러한 웹 자원의 내용을 기술하기 위해 온톨로지(Ontology)들을 이용한다. 이러한 온톨로지 중에 현재 W3C에서 제안한 OWL이 부각되고 있다. OWL을 처리하는 데이타 베이스에서 데이타는 그래프 형태로 저장되어 그래프 탐색을 통해 질의 처리를 수행한다. 본 논문에서는 OWL 데이타를 효율적으로 처리하기 위하여 시그니처를 이용한 최적화 기법을 제안한다. 논문에서 제안한 최적화 기법은 질의 수행 시 각 노드의 탐색 회수를 줄여 질의 수행을 빠르게 할 수 있게 한다.

키워드 : 시맨틱 웹, OWL, 질의 처리, 시그니처

Abstract The Semantic Web is being studied as the next step in the evolution of the web. In the environment of the Semantic Web, the information must be understandable computers as well as a just human. So we use ontologies for describing the contents of the web resources. Among such ontologies, OWL is proposed as a recommendation by W3C. OWL data is represented as graph structure and the query is evaluated by traversing each node of the graph. In this paper, we propose the optimization technique based on signature to efficiently process the OWL data. Our approach minimizes traversing each node of the graph in query processing.

Key words : semantic web, OWL, query processing, signature

1. 서 론

시맨틱 웹(Semantic Web)은 차세대 웹으로 연구되고 있으며 이는 웹 자원의 내용에 잘 정의된 의미(Semantic)을 부여함으로써 사람뿐만 아니라 컴퓨터도 쉽게 그 의미를 해석할 수 있도록 한다. 시맨틱 웹에서 가장 핵심이 되는 온톨로지는 “공유된 개념화의 정형화된 명시적 서술(a formal explicit specification of a shared conceptualization)”이라고 할 수 있다[1]. 즉 해당 영역의 개념들과 이들 개념 간의 상호 관계를 정의하는 것을 의미한다.

이러한 온톨로지를 표현하기 위해 스키마와 구문 구조등을 정의한 언어가 온톨로지 언어이며 이러한 온톨로지 언어로 RDF/RDFS[2], DAML+OIL, OWL[3]등을 들 수 있다. 이러한 온톨로지 언어에서는 데이타를 저장하고 검색하기 위한 연구가 활발히 진행되고 있다[4-6]. 이 중에서도 현재 웹 표준화 단체인 W3C에서 제안한 OWL이 부각되고 있다. OWL은 체계적인 온톨로지 구축을 지원하며 그 특징을 살펴보면 클래스와 속성, 클래스 혹은 속성 사이의 관계를 제공한다. 이는 기존의 RDF/RDFS와 같으나 OWL은 RDFS보다 더 복잡한 관계를 나타낼 수 있으며 또한 추론 능력도 더 강력하다. 따라서 이러한 OWL을 보다 효율적으로 사용하기 위해서는 OWL을 저장하고 검색하는 연구가 중요하다. RDF/RDFS와 OWL은 그래프 형태로 표현이 가능하며 이러한 그래프 형태의 모델로 데이타베이스에 문장(statement) 즉 주어(Subject), 술어(Predicate), 목적어(Object)로 구성된 트리플 구조로 저장된다[6]. 이러한

· 본 연구는 BK-21 정보기술 사업단과 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성지원사업(IITA-2005-C1090-0502-0016)의 연구결과로 수행되었음.
† 학생회원 : 서울대학교 컴퓨터공학과
dhlim@oopsla.snu.ac.kr
hyjung@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2004년 12월 22일
심사완료 : 2005년 8월 21일

그래프 모델을 통하여 사용자는 질의를 수행하게 된다. 따라서 질의는 그래프 탐색을 통해서 이루어 지며 이러한 그래프 탐색을 하기 위해서는 문장의 조인 연산이 필요하다. 또한 OWL은 클래스간의 관계를 포함하기 때문에 이러한 클래스 관계를 포함한 결과가 필요하다. 이러한 클래스 관계 역시 데이타베이스에 저장되어 있으므로 조인 연산으로 얻어진다. 이러한 조인 연산은 질의 시스템의 성능을 저하시키는 요인이 된다.

따라서 본 논문에서는 데이타베이스를 사용하는 OWL 관리 시스템에서 시그니처[7]를 이용하여 좀 더 효율적으로 질의를 처리하는 최적화 기법을 제안하였다. 객체가 자기가 속한 클래스간의 관계를 나타내주는 클래스 시그니처와 그 클래스가 가지는 Property들의 패턴을 가지는 패턴 시그니처를 가지고 있는 것이다. 사용자 질의문 역시 시그니처를 구하여 객체의 시그니처와 AND 연산 후 그 결과가 질의문의 시그니처와 같으면 해당 객체만 검색하면 되는 것이다. 이는 상위 노드가 하위 노드의 모든 시그니처 값을 가지고 있는 XML에서 제안한 방법[8]과 객체 지향형 데이타베이스에서 제안하였던 시그니처 방법[9]과 가장 큰 차이점을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 설명하고 3장은 데이타 모델 및 질의처리에 대하여 설명한다. 4장은 시그니처를 적용한 OWL 의 저장 기법과 최적화 기법에 대하여 설명하고 5장에서는 시그니처를 이용한 것과 이용하지 않은 것에 대한 성능을 비교하며 마지막으로 결론 및 향후 연구 방향에 대하여 설명한다.

2. 배경 지식 및 관련 연구

RDF/RDFS와 OWL문서를 기존의 데이타베이스에 효율적으로 저장하기 위한 방법은 활발히 연구되고 있다. 이런 방법들은 주로 스키마 생성 방법이 주를 이룬다. 가장 대표적인 것을 보면 Jena2[6,10]와 Sesame[5] 등이 있다. Jena2는 RDF/RDFS, OWL을 모두 지원하며 특히 데이타베이스에 저장하여 Persistent 모델을 제공하고 Logic 처리를 위하여 Reasoner라는 추론시스템을 제공한다. Persistent 모델이란 Jena2에서 제공하는 데이타베이스 기반의 그래프 모델을 의미하며 시스템 내에서 자체적으로 데이타를 트리플 데이타베이스에 저장시킨다. 따라서 사용자는 내부적인 구조에 상관없이 이 Persistent 모델에만 접근해서 데이타를 처리할 수 있게 되는 것이다. Jena2에서는 문장 테이블이 트리플 저장 구조를 가지고 있다. 그림 1은 이러한 jena2의 관계형 데이타베이스에서의 문장 테이블의 트리플 저장 구조를 보여주고 있으며 “http://somewhere/Paper”의 저자가 “Smith”임을 의미한다.

문장 테이블

Subject	Property	Object
http://somewhere/Paper	author	Smith

그림 1 Jena2에서의 문장 테이블의 저장 구조

Jena2는 특히 OWL 처리를 위한 API를 지원하고 있으며 질의 언어로서 RDQL[11]을 지원한다. Sesame는 정규화된 스키마 구조를 가지면서 문장 구조의 테이블은 가지고 있다. 즉 스키마 정보를 별도의 테이블에 저장을 하지만 문장을 나타내는 트리플 테이블을 갖고 있다. RDF/RDFS를 지원하며 OWL을 미약하게 지원한다. 즉 OWL의 모든 표준을 지원하진 않는다. Jena2와 마찬가지로 Persistent 모델을 제공하며 이러한 모델을 질의하기 위해 RQL[12]을 지원해 주고 있다. 시그니처 기법은 필요한 문자열에 대해 비트로 된 해쉬 값을 구하여 이 값에 대한 비트 연산만으로 원하는 데이타를 빨리 찾는 데 사용되었다[7-9]. [9]에서는 객체 지향 데이타베이스에서 OID로 참조하는 객체의 애트리뷰트의 시그니처 값을 구하여 그 피참조 객체의 객체 시그니처를 같이 저장한 후 전진 운행 질의 처리 기법에서 미리 참조하는 객체의 시그니처 값을 비교함으로써 만족하는 값이 있을 경우에만 질의를 처리한다. [8]에서는 트리의 각 노드에 해당 노드의 서브 트리에 대한 시그니처 정보를 트리의 형태로 저장하는 DOM구조로 저장하여 정규 경로식을 가지는 질의문에 대하여 탐색 범위를 줄여 효율적인 XML 문서를 처리한다.

3. 데이타 모델과 질의 처리

본 논문에서 다루는 데이타는 OWL이다. 이것은 그래프 모델의 형태를 가진다. 즉 노드와 간선을 가지며 노드 뿐만이 아니라 간선에도 값을 가진다는 것이 XML과의 차이점을 가진다. OWL 데이타의 구조 형태는 그래프 모델이다. 그림 2는 Gene Ontology[13]의 OWL 파일의 일부분으로 Plastid Chromosome(색소 염색체) 부분을 나타내주고 있다. 이 OWL 문서를 그래프 모델로 표현한 것이 그림 2이다.

그림 3에서 최상위 노드는 색소 염색체 클래스의 URL을 나타낸다. 이 염색체 클래스는 크게 두 부분의 의미를 가진다고 할 수 있다. 하나는 염색체라는 클래스의 하위 클래스라는 것이고 다른 하나는 색소체라는 클래스의 part-of 관계를 가진 클래스의 하위 클래스라는 것이다. 즉 색소체의 일부분이라는 의미를 가진다. 이렇게 OWL에서 하나의 클래스는 클래스와 클래스와의 관

```
<owl:Class rdf:about="http://www.geneontology.org/go#plastid+ chromosome">
<rdfs:label>plastid+ chromosome</rdfs:label>
<rdfs:comment>GO:0009508 A circular DNA molecule containing
plastid encoded genes. definition_</rdfs:comment>
<oiled:creationDate>2003-03-13T11:55:37Z</oiled:creationDate>
<oiled:creator>chris</oiled:creator>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.geneontology.org/go#chromosome"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.geneontology.org/go#part-of"/>
<owl:someValuesFrom>
<owl:Class rdf:about="http://www.geneontology.org/go#plastid"/>
</owl:someValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

그림 2 OWL 파일의 예(Gene Ontology)

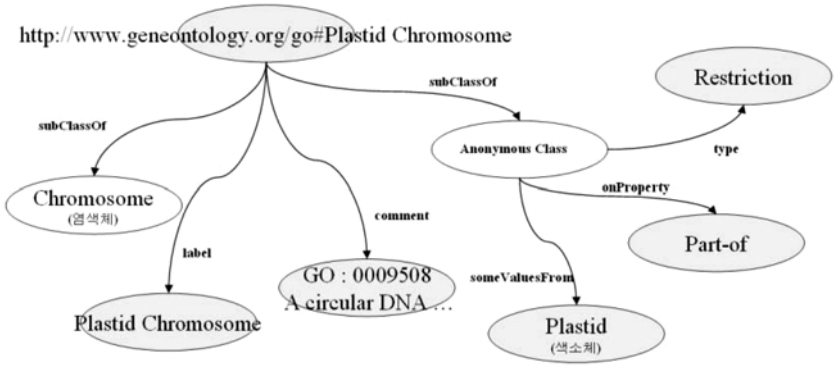


그림 3 그래프 모델(Gene Ontology)

계와 그 클래스가 가지는 property들이 핵심 부분이라고 할 수 있다. 이러한 property들은 OWL에서 restriction이라는 anonymous 클래스로 구성되어 있다. 위의 예제에서 살펴보면 anonymous 클래스는 someValuesFrom의 값으로 색소체를 가지고 onProperty 값으로 part-of의 값을 가지며 타입은 restriction을 갖는다. 이러한 anonymous 클래스를 위의 예제처럼 하나만 가질 수도 있고 여러 개의 anonymous 클래스를 가질 수 있다. OWL은 RDF를 기반으로 구성되기 때문에 그래프 형태로 존재한다. 따라서 질의 처리는 그래프 탐색을 통해서 이루어진다. 색소 염색체의 클래스 URL부터 각각의 노드를 탐색하여 이 클래스의 질의를 수행한다. Jena2에서 제공하는 모델은 위와 같은 그래프 모델이다. 본 논문에서 가정하는 데이타는 OWL 문서가 아닌 데이타베이스에 저장되어 있는 대용량의 OWL 데이타를 가정하고 있다. 데이타베이스에 저장되어 있는 트리플 구조는 노드와 간선 그리고 이 간선이 지시하는 노드로 구성되어 있다. 따라서 그래프 탐색의 처리는 데이타베이스의 조인 연산을 필요로 한다. 따라서 이러한 조인 연산을 줄이는 것이 OWL 질의 처리의 핵심이다.

본 논문에서 다루는 질의 처리는 이러한 그래프 탐색을 줄여 효율적인 질의 처리를 할 수 있게 하는 최적화 기법이다. 즉 시그니처를 이용하여 탐색 횟수를 줄여 빠르게 OWL을 처리 하는 것이 본 논문의 목적이다.

4. 시그니처(Signature) 기법을 적용한 OWL 질의 처리

4.1 시그니처 기법

시그니처는 일반적인 텍스트 문서에서 검색을 빠르게 하기 위한 방법으로 제안되었다. 문자열에 대한 해쉬값으로서 본 논문에서는 [7]의 SC 방법으로 시그니처를 만들었다. 예를 들어 그림 4에서와 같이 각각의 문자열에 대한 시그니처를 구했다고 하면 이 문자열을 포함하는 문서의 블록은 이 문자열의 시그처들의 OR연산을 가지고 있는다. 이때 이 문서가 'Wine'을 포함하는지 질의하면 'Wine'의 시그니처와 블록의 시그니처를 AND 연산을 하여 그 결과가 Wine의 시그니처가 나온다면 이 블록내에 'Wine'이 있을 확률이 높은 것이다. 다시 말하여 AND 연산의 결과가 'Wine' 시그니처가 나오지 않는다면 이 블록에는 'Wine'이라는 문자열을 포함하지 않고 있으므로 블록내에 있는 문자열을 검색하지 않아도 된다. 즉 탐색 범위를 줄일 수 있는 것이다. 이러한 시그니처 기법은 관계형 데이타베이스, 객체 지향 데이타베이스, XML 데이타베이스등에서 인덱스 기법으로 많이 사용되어 왔다.

4.2 OWL에서의 시그니처를 적용

본 논문에서 제안하고 있는 OWL에서의 시그니처 방법은 2가지의 시그니처 구조를 가진다. 즉 클래스들간의 시그니처인 클래스 시그니처와 property들로 이루어진 시그니처인 패턴 시그니처로 구성되어 있다. 클래스들의 관계를 나타내는 클래스 시그니처는 계층 정보를 의미한다. 즉 상속 관계, 집합 연산(intersectionOf,unionOf), 동등관계(equivalentOf) 등이 포함된다. 집합 연산은 OWL inference rule에 의해 subclassOf 관계로 나타낼 수 있다. 즉 'WhiteWine'이 'Wine'과 'hasColor'와 'White'의 intersectionOf으로 구성되면 이는 'Wine'의 subclassOf로 나타낼 수 있다. unionOf와 equivalentOf 또한 바꾸어 줄 수 있다. 클래스에서 property는 그래프의 패턴을 의미할 수 있다. 즉 restriction으로 구성되어 있는 property와 그 값이 그 클래스의 패턴을 의미해 준다. 이러한 특성을 반영하기 위해 패턴 시그니처를 별도로 두어서 사용하자는 것이다. 이렇게 두 개의 시그니처를 별도로 두어서 사용하는 것이 본 논문에서 제안한 방법인 것이다. 그림 5는 이러한 시그니처 구조를 표현한다.

예를 들어 색소 염색체의 클래스 시그니처는 염색체 클래스를 포함할 수 있도록 염색체의 시그니처와 OR 연산을 한 결과 값을 자기의 시그니처로 가져야 한다. 즉 색소 염색체는 염색체 클래스에 포함된다고 할 수 있다. 또한 색소 염색체의 property에 해당하는 part-of가 색소체인 것은 패턴 시그니처에 part-of의 시그니처와 색소체의 시그니처의 OR 연산 결과로 한다. 즉 restriction이 하나의 패턴 시그니처를 가지고 있으며 한 클래스는 자기에게 속한 모든 restriction들의 시그니처 값들을 OR 한 결과를 가지게 된다. 이렇게 전처리된 시그니처 값들을 가지고 질의문을 수행할 때 효율적으로 처리할 수 있다. 즉 질의문을 수행할 때 시그니처를 우선 검색하여 해당 조건에 만족하는 객체들을 우선 찾아

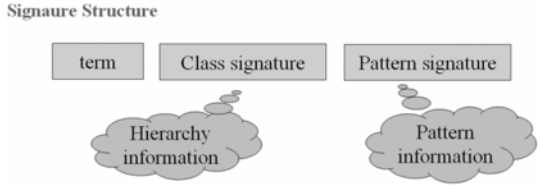


그림 5 OWL에서의 시그니처 구조

String	Signature
Wine	01000000
Food	00100001
Grape	10001001

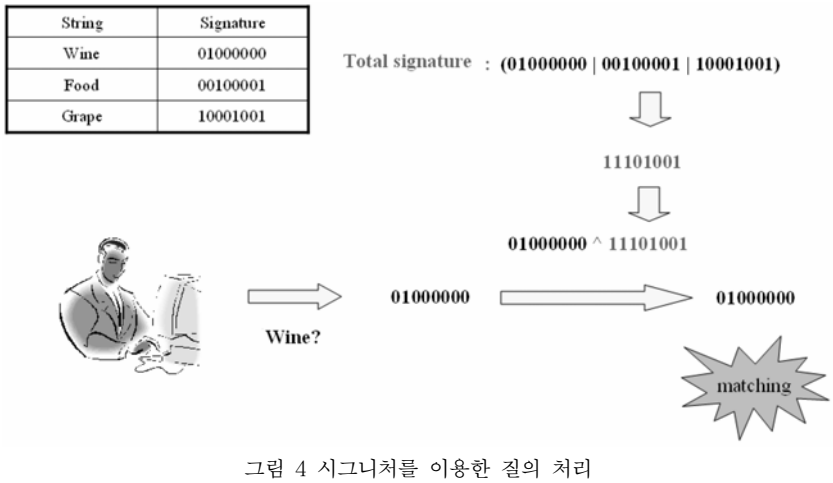


그림 4 시그니처를 이용한 질의 처리

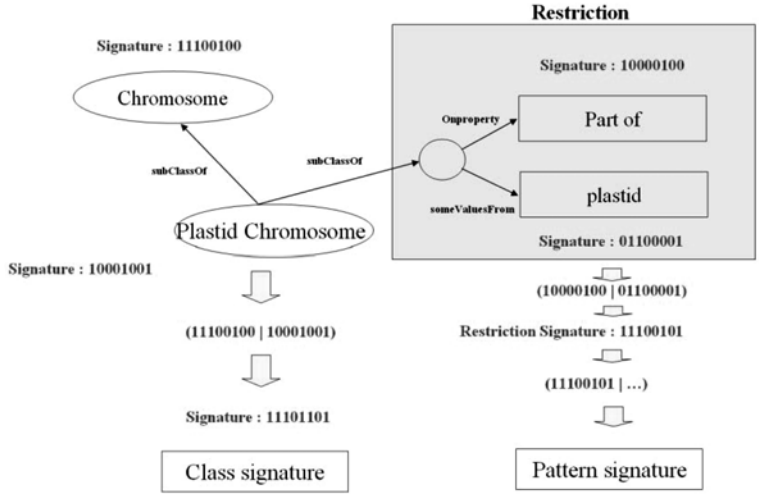
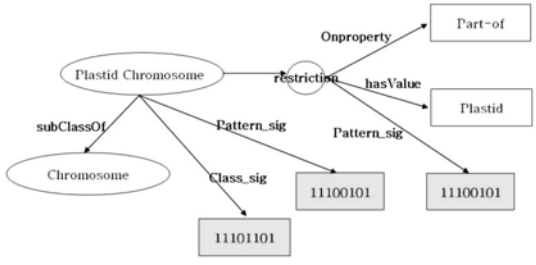


그림 6 OWL 데이타에 대한 시그너처 전처리

서 검색을 수행 하는 것이다. 그림 6은 전처리 과정에 해당하는 것을 보여준다.

4.3 OWL에서의 시그너처를 저장

이렇게 전처리 계산된 시그너처는 데이타 베이스 시스템에 저장되어야 한다. 그래서 질의가 수행될 시 이 저장된 시그너처를 이용해야 한다. 시그너처를 별도의 데이타베이스 시스템에 저장할 수 있으나 이는 효율적인 수행에 있어서 문제점을 가지게 된다. 그래서 본 논문에서는 시그너처를 별도로 다른 데이타 베이스에 저장하는 것이 아니라 그래프 모델 내에 데이타와 같이 직접 저장하는 방식을 택했다. 즉 전처리 된 결과를 하나의 노드와 간선을 만들어서 그 노드에 추가 삽입시키는 것이다. 이는 Jena2에서 제공하는 addProperty란 메서드를 통해서 쉽게 사용할 수 있다. 즉 class_sig는 클래스 시그너처 값을 가지는 간선이 되고pattern_sig는 패턴 시그너처 값을 가지는 간선이 되는 것이다. 이는 각각의 전처리 과정에서 구해진 결과가 곧바로 데이타베이스에 저장되는 것이다. 우선 restriction에 property들의 시그너처값들의 OR 연산을 한 결과가 패턴 시그너처에 저장이 되고 이러한 restriction들이 자기가 속한 클래스의 패턴 시그너처 값으로 저장이 된다. 이때 클래스가 여러 개의 restriction을 가지고 있으면 이 여러 개의 restriction들의 시그너처 값들은 OR 연산을 하여 클래스가 가지고 있는다. [8]에서 사용한 시그너처의 기법과 유사하다고 할 수 있으나 [8]에서는 상위 노드가 하위 노드의 모든 시그너처 값들을 가지고 있는 것이고 본 논문에서는 클래스가 restriction의 시그너처 값들을 가지는 것이고 이 restriction들의 패턴 시그너처는 하위 노드들의 값을 가지는 것이므로 [8]과의 가장 큰 차이점



Statement Table

Subject	Property	Object
#Plastid Chromosome	Class_sig	11101101
#Plastid Chromosome	Pattern_sig	11100101

그림 7 시그너처의 저장

이 된다. 이런 차이점은 [8]은 XML의 트리 모델에서 엘리먼트의 관계에 초점을 두는 반면에 본 논문에서는 OWL 모델에서 클래스와 클래스의 관계 그리고 그 클래스가 가지는 restriction에 초점을 두기 때문에 나타난다. 즉 OWL 특성에 맞게 시그너처를 저장하는 것이다. 따라서 그림 7은 이러한 시그너처의 저장을 나타내며 트리플 데이타베이스에 저장되는 방식을 나타내 주고 있다.

4.4 OWL에서의 시그너처를 이용한 질의 처리

만약에 색소체를 part-of로 가지는 클래스를 검색하라는 질의를 처리한다고 하자. 우선 클래스를 찾아서 그 클래스의 패턴 시그너처를 검사한다. 이 클래스의 패턴 시그너처의 값과 색소체와 part-of의 OR 연산한 결과를 AND 연산을 한다. 결과가 색소체와 part-of의 OR 연산 결과와 같다면 이 클래스가 답이 될 수 있으므로 탐색을 허용하고 그렇지 않을 경우 없다는 것을 보장하

므로 탐색을 하지 않고 다음 클래스를 검사하게 된다. 클래스와 property의 복합적인 구성으로 이루어진 질의에 있어서도 마찬가지이다. 염색체의 하위 클래스이면서 색소체를 part-of의 값으로 가지는 클래스를 검색한다고 하면 우선 part-of를 색소체 값으로 갖는 클래스를 패턴 시그니처를 이용하여 검색하고 이 클래스들의 클래스 시그니처를 이용해서 매칭되는 값에 해당하는 클래스만 탐색을 허용한다. 이러한 복잡한 질의 처리에 있어서 검색하는 알고리즘은 그림 8과 같다.

```
Function Retrieval_Sig
input : OWL Class C (user query) , OntModel model

for all classes (type x Class) ∈ model    do
  if pattern_sig value of x ∧ signature of c ≡ signature of c then
  if class_sig value of x ∧ signature of c ≡ signature of c then
       traversal property node of x
       check x to satisfy condition of c
       return x
  endif
  endif
```

그림 8 시그니처를 이용한 검색 알고리즘

5. 실험 결과

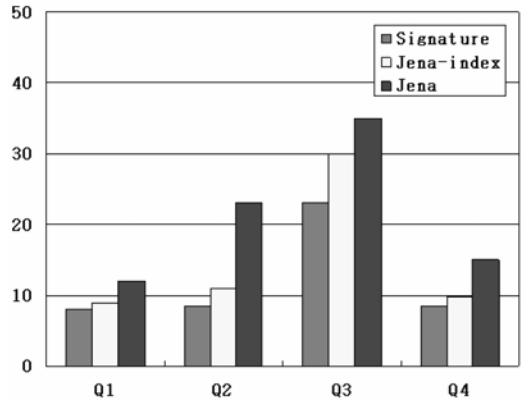
본 논문의 실험은 Pentium Ⅲ 803MHz CPU, 512MB memory, Window 2000 operating system 그리고 Java로 코딩된 Jena2와 mysql을 저장시스템으로 사용하는 Persistent 모델을 이용하여 수행하였다. 실험 데이타로는 Gene Ontology[13]의 termdb가 사용되었다. 실험에 사용된 데이타는 10Mbytes의 크기를 가지며 Gene Ontology는 현재 생물 정보학에서 가장 활발히 사용되는 ontology이다. 원래 Gene Ontology는 RDF 문서로 작성되어 있지만 현재 GONG Project[14]에 의해 DAML+OIL로 바꾸는 작업이 진행 중이며 이 DAML+ OIL 문서를 OWL 컨버터 프로그램[15]을 사용해서 OWL 문서로 바꾸어서 실험하였다. 실험에서 사용한 질의는 표 1의 4개의 질의를 사용하였다. 단순 질의 즉 클래스간의 관계가 없는 것 혹은 restriction을 가지지 않는 질의는 포함하지 않았다.

첫 번째 질의는 하나의 restriction을 포함하는 클래스를 찾는 질의이고, 두 번째는 다수의 restriction을 포함하는 클래스를 찾는 질의어다. 세 번째 질의는 클래스간의 관계를 묻는 질의다. 마지막 질의어는 클래스와 클래스간의 관계와 restriction을 포함하는 클래스를 찾는 질의이다. 즉 앞의 세 질의는 패턴 시그니처에 대한 성능을 시험하는 것이고 마지막 질의는 클래스 시그니처의 성능에 대한 실험이다. 실험에 사용한 질의는 표에서처럼 트리플로 나타내어진다. 즉 하나의 트리플 패턴으로 표현이 되는 것이다. 실험은 위의 질의를 본 논문에서 제안한 시그니처 기법과 Jena2에서 지원하는 Naïve한 방법, jena2에 B+-tree 인덱스를 사용하는 방법을 비교하였다. 시그니처의 크기는 32 비트로 사용하였다.

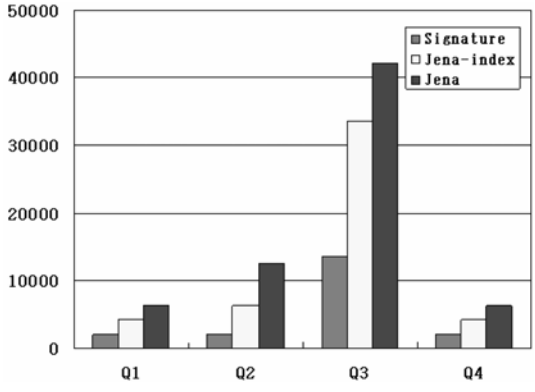
그림 9(a)는 Gene Ontology에 위의 질의를 사용한 수행 시간을 나타내 주고 (b)는 각각의 질의에 대한 노드 탐색 횟수를 나타낸다. (a)의 실험 결과에서 모든 질의에 있어 시그니처 기법을 사용한 것이 더 좋은 성능을 나타내는 것을 알 수 있다. 이러한 결과를 (b)를 통해 분석해 보면 시그니처를 사용한 방법이 가장 많이 노드 탐색을 줄임을 알 수 있다. 따라서 이러한 노드 탐색의 범위를 줄인 것은 더 좋은 성능을 보여준다. Jena2-index보다 더 좋은 성능을 나타내는 것도 jena2가 인덱스를 사용하여 restriction 노드에서부터 하위 노드를 탐색하는 반면에 시그니처를 사용한 기법은 시그니처 값이 일치하는 노드에서만 탐색을 하기 때문이다. Q1과 Q3를 비교해 보면 수행 시간은 많이 차이가 나지만 노드 탐색 비율은 비슷한 결과를 나타낸다. 이에 반해 Q2가 가장 효율적인 노드 탐색 횟수를 보여준다. 이는 질의에 있어 restriction이 더 많은 영향을 끼친다고 볼 수 있다. 따라서 다수의 restriction에 대해 시그니처 기법이 더 좋은 성능을 보여준다. Q4는 클래스간의 관계를 포함하는 질의이며 이 질의 실험에 클래스 시그니처가 사용되었고 마찬가지로 클래스 시그니처를 사용한 것이 시그니처를 사용하지 않은 것보다 성능이 더 좋게

표 1 실험에 쓰인 질의

Q1	(type ?x class)(subClassOf ?x ?y)(onProperty ?y “Part-of”) (type ?y restriction)(someValuesFrom ?y “secretory+pathway”)
Q2	(type ?x class)(subClassOf ?x ?y)(onProperty ?y “Part-of”) (type ?y restriction)(someValuesFrom ?y “nuclear+membrane”) (subClassOf ?x ?z))(onProperty ?z “Part-of”)(type ?z restriction) (someValuesFrom ?z “nuclearenvelope-endoplasmic+reticulum+network”)
Q3	(type ?x class)(subClassOf ?x “plastid+chromosome”)
Q4	(type ?x class)(subClassOf ?x “cell+growth+and%2For+maintenance”) (subClassOf ?x ?y)(type ?y restriction) (onProperty ?y “Part-of”)(someValuesFrom ?y “cell+growth”)



(a) 수행 시간 (단위:초)



(b) 노드 탐색 횟수

그림 9 성능 비교

나왔다. 위의 성능 비교 결과에서 알 수 있듯이 클래스 시그니처를 이용한 기법이 다른 방법들 보다 더 효율적으로 수행하고 노드 탐색에 있어서도 많은 노드 탐색을 줄이는 것을 알 수 있다. 시그니처 기법을 사용하는데 드는 비용은 크게 두 가지로 볼 수 있다. 하나는 전처리를 필요로 한다는 것이고 두 번째는 데이타 그래프의 노드의 증가 즉 시그니처를 저장하는 노드가 증가한다는 것이다.

· Ni : 그래프 모델 M에서의 전체 노드의 개수
· Ci : 그래프 모델 M에서의 클래스의 개수
· Ri : 그래프 모델 M에서의 restriction의 개수

라 하면 시그니처를 저장하는데 드는 추가 노드의 비용은 다음과 같다.

노드의 비용 I = (Ni+Ci*2+Ri) / Ni

즉 클래스는 클래스 시그니처와 패턴 시그니처 즉 한 개의 클래스당 2개의 노드가 증가하게 된다. 또한 restriction의 개수만큼 패턴 시그니처를 가지게 되므로 위와 같은 비용이 나타나게 된다. 실제로 Gene Ontology의 termDB인 경우 실제 데이타베이스의 크기가 약 21MByte 이지만 전처리 단계를 거친 후에는 약 1.1MByte가 증가한다. 즉 32비트의 시그니처의 크기를 설정하여 5%에 해당되는 작은 크기의 인덱스로 효율적으로 탐색을 줄일 수 있는 것이다.

6. 결론 및 향후 연구과제

우리는 본 연구에서 OWL 질의 처리를 효율적으로 처리하기 위한 시그니처를 이용한 질의 최적화 기법을 제안하였다. OWL에서 가장 핵심적인 것은 클래스들의 관계와 property들의 관계를 잘 처리하느냐에 달려있다. 이러한 질의 처리에는 수많은 그래프 탐색을 요구하며 이러한 그래프의 탐색은 결국 데이타베이스에서 조인 연산으로 이루어지기 때문에 성능이 저하된다. 따라서 본 논문에서 제안한 질의 처리 기법은 대용량의 OWL을 다루는 질의 처리 시스템에서 적은 공간을 가지고서 효과적인 성능을 보일 수 있다.

하지만 클래스의 상속 관계의 깊이가 깊어 질수록 시그니처의 값은 1로 세팅이 되어 효율적이지 못하다. 이것은 시그니처를 늘리거나 혹은 파티션을 이용하는 기법등으로 해결될 수 있으며 대부분의 온톨로지 데이타들은 깊이가 그리 깊지 않은 형태로 되어있으며 Gene Ontology의 경우 최고 깊이가 12이다[16]. 따라서 이러한 시그니처를 이용한 질의 처리는 매우 효율적으로 작용할 수 있다.

온톨로지의 검색에 있어 추론은 많은 부분을 차지한다. 이러한 추론과 연결된 질의 시스템은 필수적이다. 앞으로 이런 추론과 연결된 질의 시스템에 시그니처를 이용하면 더 효율적으로 검색할 수 있을 거라고 생각된다.

참 고 문 헌

[1] T. R. Gruber, "A Translation Approach to Portable Ontologies," Knowledge Acquisition, 5(2):199-220, 1993.

[2] O. Lassila, R. Swick, "Resource Description Framework(RDF) Model and Syntax Specification," W3C Recommendation, World Wide Web Consortioum, 1999.

[3] M. Dean, G. Schreiber, OWL Web Ontology Language Reference, http://w3c-.org/TR/owl-ref.

[4] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Toll, "The RDFSuite: Managing Voluminious RDF Description Bases," Semantic Web Workshop 2001.

[5] J. Broekstra, A. Kampman, F. Harmelen, "Sesame: An Architecture for Storing and Querying RDF Data and Schema Information," International Semantic Web Conference 2002.

[6] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds, "Efficient RDF Storage and Retrieval in Jena2," Proceedings of SWDB'03.
[7] Chris Faloutsos, "Signature files: Design and Performance Comparison of Some Signature Extraction Methods," SIGMOD, 1985.
[8] Sangwon Park, Hyoung-Joo Kim, "A New Query Processing Technique for XML Based on Signature," DASFAA, 2001.
[9] Hwan-Seung Yong, Suckho Lee, Hyung-Joo Kim, "Applying Signatures for Forward Traversal Query Processing in Object-Oriented Databases," ICDE, 1994.
[10] Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/.
[11] RDQL - A Query Language for RDF, http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.
[12] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, "RQL: A Declarative Query Language for RDF," WWW2002.
[13] The Gene Ontology Consortium, "Gene Ontology: tool for the unification of biology," nature genetics, 2000.
[14] Gene Ontology Next Generation (GONG), http://gong.man.ac.uk/index.shtml
[15] DAML+OIL to OWL Conversion, http://www.daml.org-/2003/06/owlConversion/
[16] A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, "Benchmarking RDF Schema for the Semantic Web," ISWC, 2002.

임 동 혁
2003년 고려대학교 컴퓨터교육과(학사) 2005년 서울대학교 컴퓨터공학부(석사) 2005년~현재 서울대학교 컴퓨터공학부 박사과정 재학 중. 관심분야는 데이타베이스, XML, 시맨틱웹

정 호 영
2000년 한국외국어대학교 컴퓨터공학과(학사). 2002년 한국외국어대학교 컴퓨터공학과(석사). 2003년~현재 서울대학교 컴퓨터공학부 박사과정 재학 중. 관심분야는 시멘틱웹, 온톨로지, XML, 데이타베이스

김 형 주
정보과학회논문지 : 데이타베이스
제 32 권 제 5 호 참조