공학석사 학위논문

OWL 질의 처리를 위한 시그니처를 이용한 최적화 기법

2005년 2월

서울대학교 대학원

전기·컴퓨터 공학부

임 동 혁

OWL 질의 처리를 위한 시그니처 기반 최적화 방법

Optimization Technique based on Signature for OWL Query Processing

指導敎授 金 炯 周

이 論文을 工學碩士 學位論文으로 提出함

2004年 10月

서울大學校 大學院

전기·컴퓨터 工學部

任 東 爀

任東爀의 工學碩士 學位論文을 認准함

2004年 12月

委 員 長 이 석 호 印

副委員長 김 형 주 印

委 員 이 상 구 印

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.

클래스 시그니처와 패턴 시그니처로 구성된 시그니처를 가지고 질의 처리시 시그니처를 비교한 후 매칭되는 시그니처만 탐색을 허용한다. 이 기법은 기존의 처리 방식에 비해 뛰어난 성능 향상을 보인다.

주요어 : OWL, 시그니처, 온톨로지, 최적화기법

학 번 : 2003 - 21675

목 차

그림 목차

1. 서론

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 형시적 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. 여기서 개념화(conceptualization)는 어떤 현상에 대해 단순화, 추상화된 관점으로 그 현상에 관련있는 것들을 식별하는 모델로 설명된다. 형식적(formal)인 것은 온톨로지가 사람뿐만이 아니라 기계가 판독할 수 있는 형태이어야 한다는 것을 의미하며 공유된(shared) 의미는 단독으로 개별적으로 사용되는 온톨로지가 아니라 여러 사람에 의해 일치되는 것을 가지고 있어야 한다는 것이다.

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 데이터를 저장하고 검색하기 위한 연구가 활발히 진행되고 있다[4,5,6]. 이 중에서도 현재 웹 표준화 단체인 W3C에서 제안한 OWL이 부각되고 있다. OWL은 체계적인

온톨로지 구축을 지원하며 특징을 살펴보면 클래스와 속성, 클래스 혹은 속성 사이의 관계를 제공한다. 이는 기존의 RDF/RDFS와 같으나 OWL은 RDFS보다 많은 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 데이터베이스에 statement 즉 주어(Subject), 술어(Predicate), 목적어(Object)로 구성된 triple 구조로 저장된다[6]. 이러한 그래프 모델을 통하여 사용자는 질의를 수행하게 된다. 따라서 질의는 그래프 탐색을 통해서 이루어 지며 이러한 그래프 탐색을 하기 위해서는 statement 테이블의 조인 연산이 필요하다. 또한 OWL은 클래스간의 관계를 포함하기 때문에 이러한 클래스 관계를 포함한 결과가 필요하다. 이러한 클래스 관계 역시 데이터 베이스에 저장되어 있으므로 조인 연산으로 얻어진다. 그림 1은 위의 과정을 그림으로 나타내주고 있다. 이러한 조인 연산은 질의 시스템의 성능을 저하시키는 요인이 된다. 따라서 본 논문에서는 데이터 베이스를 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.

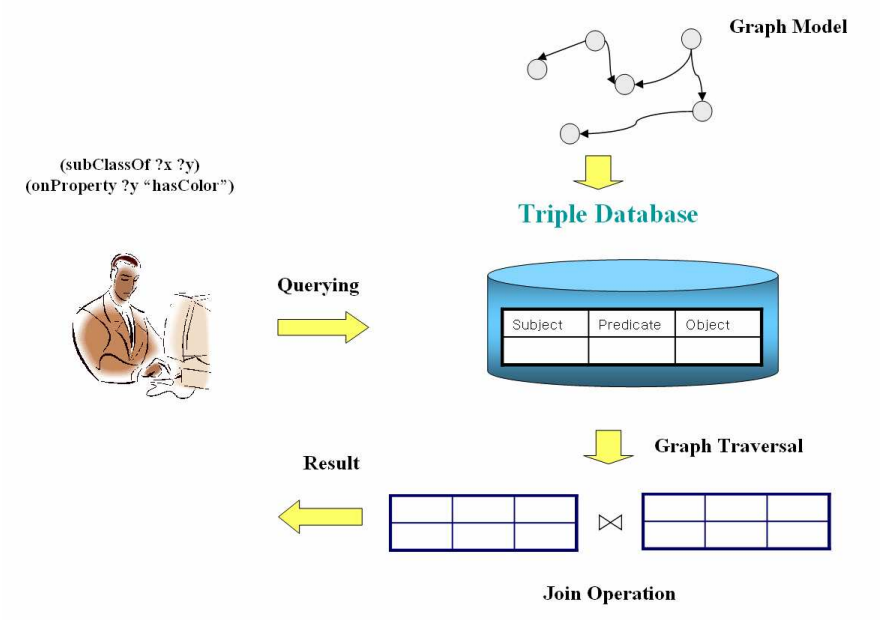


그림 1 OWL의 저장과 질의 처리 과정

이는 상위 노드가 하위 노드의 모든 시그니처 값을 가지고 있는 XML에서 제안한 방법[8]과 객체 지향형 데이터 베이스에서 제안하였던 시그니처 방법[9]과 가장 큰 차이점을 가진다. 가장 큰 차이점이라고 하면 모든 값들이 아닌 특정 Property의 값들만 가진다는 차이점이 된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 설명하고 3장은 데이터 모델 및 질의 처리에 대하여 설명한다. 4장은 시그니처를 적용한 OWL을 Jena2의 triple 데이터베이스에 저장하는 방법과 저장된 시그니처를 사용하여 본 논문에서

제안하고 있는 최적화 기법에 대하여 설명하고 5장에서는 시그니처를 이용한 것과 이용하지 않은 것에 대한 성능과 각 질의에서의 그래프 탐색 횟수 그리고 시그니처를 사용하는데 드는 비용에 대해 설명한다. 마지막으로 결론 및 향후 연구 방향에 대하여 설명한다.

2. 배경 지식 및 관련 연구

2.1 Jena2

RDF/RDFS와 OWL문서를 기존의 데이터 베이스에 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.

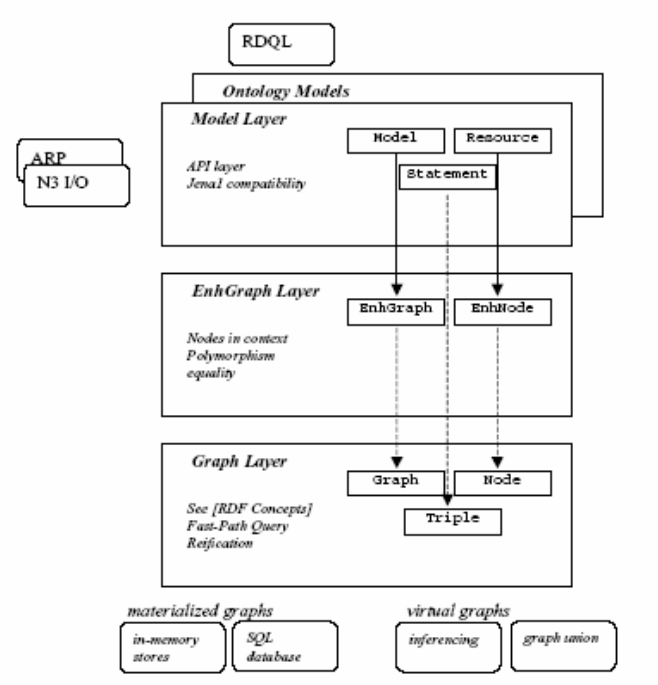


그림 2 Jena2 구조

그림 2는 Jena2의 구조를 나타내주고 있다. Jena2는 시맨틱 웹 기반의 자바 프레임워크로서 RDF/RDFS, OWL을 모두 지원하며 특히 데이터베이스에 저장하여 persistent 모델을 제공하며 Logic 처리를 위한 reasoner라는 추론 기능을 지원해 주며 Jena2에서는 statement의 triple 저장 구조를 가지고 있다.

Statement Table

Subject	Predicate	Object
mylib:doc1	dc:title	Jena2
mylib:doc1	dc:creator	HP Labs - Bristol
mylib:doc1	dc:creator	Hewlett-Packard
mylib:doc1	dc:description	101
201	dc:title	Jena2 Persistence
201	dc:publisher	com.hp/HPLaboratories

Literals Table

Id	Value
101	The description - a very long literal that might be stored as a blob.

Resources Table

Id	URI
201	hp:aResource-WithAnExtreme-lyLongURI

그림 3 Jena2의 관계형 데이타베이스 스키마

또한 OWL 처리를 지원해주는 장점을 가지고 있으며 질의 언어로서 RDQL[11]을 지원한다. 그림 3은 Jena2에서 사용하는 관계형 데이터 베이스 스키마를 나타내주고 있다. Jena에서 사용하는 질의의 형태는 그래프에 매칭되는 triple 패턴들을 찾는 것이다. 즉 질의의 한 형태는 다음과 같다.

(?x P ?y) (?y Q ?z)

위의 예제는 그래프에서 매칭되는 ?x, ?y 그리고 ?z를 찾으라는 질의가 된다. 이러한 질의 형태를 지원하는 것이 RDQL[11]이다. RDQL은 Jena 모델에서의 RDF를 지원하기 위한 질의 언어이다. 그림 4의 RDF 모델에서의 RDQL은 다음과 같다.

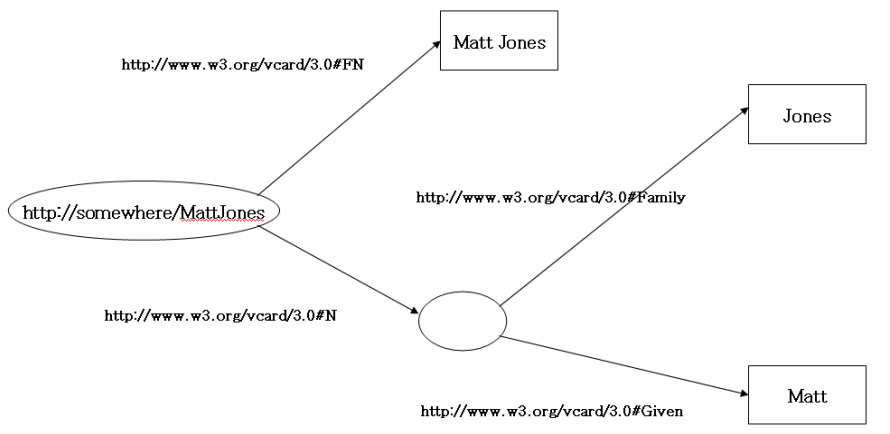


그림 4 RDF 모델의 예

SELECT ?x WHERE

(?x, <http://www.w3.org/vcard/3.0#FN>, “Matt Jones”)

이 RDQL은 “Matt Jones”를 http://www.w3.org/vcard/3.0#FN 의 값으로 갖는 노드를 찾는 질의이다. 이 질의에 대한 결과는 “http://somewhere/MattJones”의 노드가 된다. 즉 그림 2의 RDF

모델을 그래프로 보았을 때 이 그래프의 triple 패턴을 찾는 질의가 된다. 다음은 조금 더 복잡한 형태의 질의이다.

SELECT ?z WHERE
(?x, <http://www.w3.org/vcard/3.0#N>, ?y)
(?y, <http://www.w3.org/vcard/3.0#Given>, ?z)

이 질의는 연속된 triple 패턴을 찾는 것으로 결과는 “Matt” 노드가 된다.

2.2 Sesame

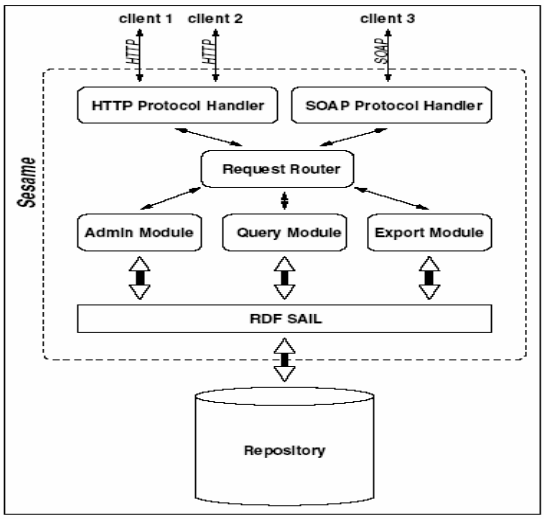


그림 5 Sesame 구조

Sesame은 On-To-Knowledge 프로젝트에서 수행한 저장 시스템으로서 정규화된 스키마 구조를 가지지만 Statement 구조의 테이블은 가지고 있다. 그림 5는 Sesame의 구조를 나타내 주고 있으며 그림 6은 Sesame에서 mysql을 사용하여 저장한 데이타베이스 스키마를 나타낸다. Sesame는 RDF/RDFS를 지원하며 OWL을 약하게 나마 지원해 주고 있다. Jena2와 마찬가지로 persistent 모델을 제공하며 이러한 모델을 질의하기 위해 RQL[12]을 지원해 주고 있다.

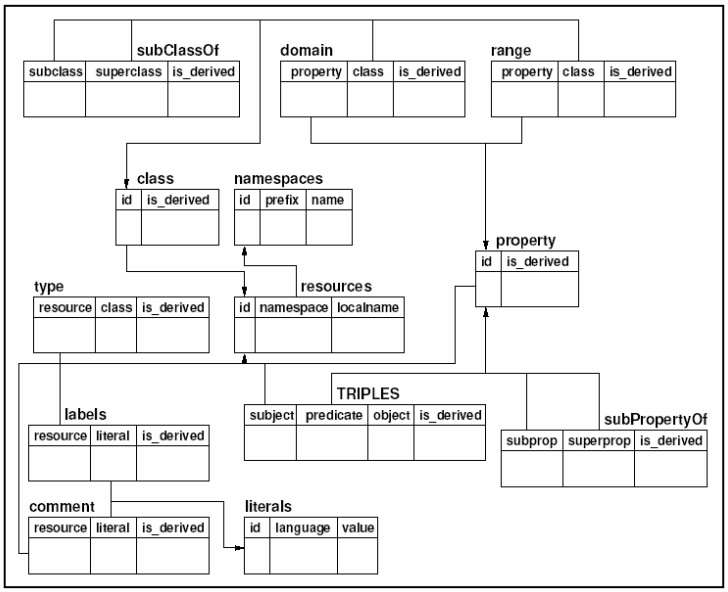


그림 6 Sesame의 관계형 데이터베이스 스키마

RQL[12] 역시 RDQL과 마찬가지로 RDF 모델을 대상으로 삼고 있다. 하지만 RDQL과 다르게 스키마와 데이터에 대한 질의를 분리하여 사용하고 있다. 그림 7은 RQL을 설명하기 위한 RDF 모델로서 예술에 대한 웹 포탈을 나타내고 있다.

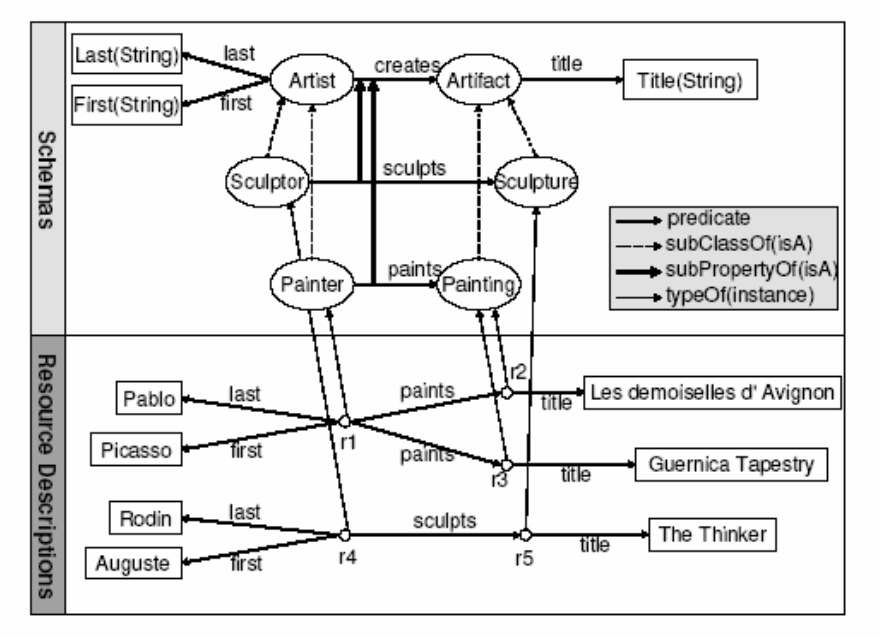


그림 7 예술 웹 포탈에 대한 RDF 모델

그림 7의 RDF 모델에서 위의 부분은 스키마 부분을 나타내며 아랫부분은 데이터 즉 인스턴스에 해당된다. “Artist”는 “Artifact”에 “creates”의 간선으로 연결되어 있다. 이는 예술가는 예술 작품을 만드는 것을 의미한다. 또한 스키마에서는 클래스간의 상속이 가능하며 간선간에도 즉 property 간에도 상속 관계가

성립하게 된다. 클래스 상속 관계는 “Artist”와 “Sculptor” 즉 예술가는 조각가의 상위 클래스가 되며 “creates”는 “sculpts”의 상위 술어가 된다. 그림 7의 모델을 가지고 RQL을 설명하면 다음과 같다.

SELECT x, y FROM Class{$x}creates{$y}

즉 “creates”라는 간선으로 연결된 두 개의 노드를 찾는 질의이다. 어떤 클래스가 “creates”라는 property의 정의역과 공역으로 나타나는지를 나타낸다. 정의역은 주어(Subject)가 가질 수 있는 값의 범위이며 공역은 목적어(Object)가 가질 수 있는 값의 범위를 나타낸다.

SELECT $x FROM Artist{$x}

위의 질의는 “Artist” 클래스의 하위 클래스를 찾는 질의어이다. 조금 더 복잡한 질의어를 살펴보면

SELECT z, w
FROM Sculptor.creates{y}.exhibited{z}, {z}title{w}

위의 질의는 “Sculptor”에 의해 “creates” 된 “exhibited”로 연결된 리소스의 “title”을 찾으라는 질의어이다. 이처럼 RQL은 다양한 형태의 질의를 지원하며 이런 다양한 형태의 지원으로

인하여 많은 RDF 질의어로 사용된다.

2.3 시그니처를 이용한 기법들

시그니처 방법은 필요한 단어들의 해쉬 값을 구하여 간단한 비트 연산만으로 원하는 데이터를 빨리 찾기 위하여 사용되었다[7,8,9]. [7]에서는 시그니처를 만드는 방법들과 그 방법들의 성능을 비교하고 있다. 가장 대표적인 방법으로는 WC(word signature) 방법과 SC(superimposed coding) 방법이 있다. WC 방법은 그림 8과 같다.

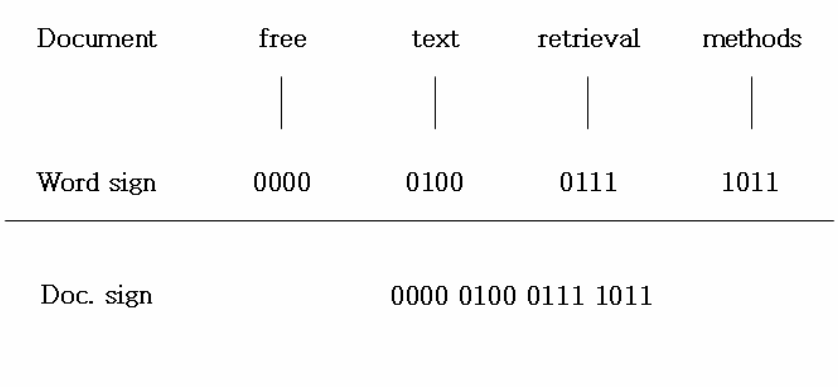


그림 8 Word Signature 방법

WC 방법은 문서내의 있는 문자열을 길이 f(그림 8에서는 f의 값이 4가 된다)로 비트 해쉬 함수로 가져서 모든 값들을 연결시켜 문서가 시그니처 값을 갖게 되는 방법이다. SC 방법은 WC 방법과

약간의 차이를 가지게 된다. 그림 9는 SC 방법을 나타내 주고 있다. 각 단어의 해쉬 값을 구하는 것은 같으나 문서가 가지는 시그니처 값은 시그니처 값들을 연결해서 가지는 것이 아니라 OR 연산을 해서 가지게 되는 것이다.

Word	Signature
free	001 000 110 010
text	000 010 101 001
block signature	001 010 111 011

그림 9 Superimposed Coding 시그니처 방법

그림 9에서 블록은 2개의 단어를 가지며 시그니처의 크기는 12 비트이며 1로 세팅되는 값은 4이다. 즉 각 단어에서 1로 세팅되는 개수를 의미한다. [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.

2.4 OWL

OWL[3]에 대한 연구는 현재 활발히 진행 중이라고 할 수 있다. OWL은 RDF/RDFS보다 많은 관계를 표현할 수 있으며 크게 3종류의 sublanguage로 구성된다. 이는 각각 자기가 필요한 부분만 접근해서 사용할 수 있도록 하기 위함이다.

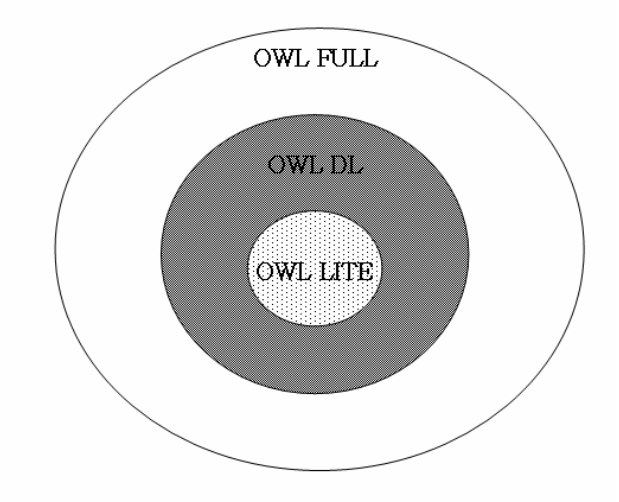


그림 10 OWL의 sublanguage

OWL은 OWL LITE, OWL DL, OWL FULL로 분류된다. 각각의 특징을 보면 다음과 같다. OWL Lite는 기본적인 기능만을 제공

한다. OWL DL은 제약 사항에서만 사용할 수 있는 제한을 가지며 OWL FULL은 자유로운 표현력을 가진다. 그림 10은 이러한 OWL 관계를 나타내 주고 있다.

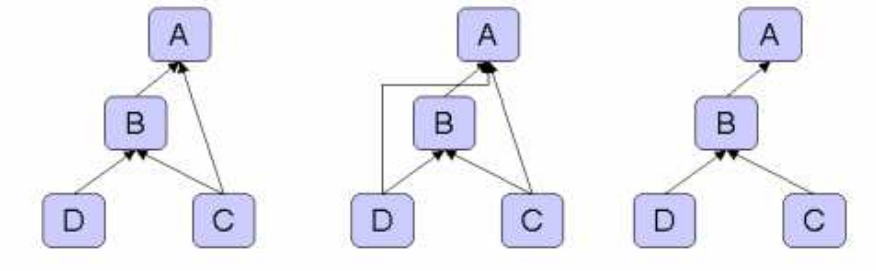
OWL은 시맨틱 웹에서 핵심적인 역할을 할 것이라고 여겨지고 있다. 현재 W3C의 Recommendation으로 많은 연구가 진행중이다. W3C에서는 wine 온톨로지를 가지고 OWL을 설명하고 있다. 그림 11은 wine 온톨로지에서 white wine 클래스에 대한 정의를 나타내주고 있는 부분이다.

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor" />
      <owl:hasValue rdf:resource="#White" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

그림 11 white wine 클래스

white wine 클래스를 살펴보면 우선 클래스를 정의할 때 intersectionOf 관계를 볼 수 있다. 이는 white wine 클래스는 wine 클래스이면서 “hasColor”가 “White” 값을 갖는 클래스인 것을 의미하게 된다. 즉 교집합의 관계를 기술하고 있는 것이다. 이러한 set 관계를 지원해 주는 것도 OWL 만이 가지는 특징이다.

intersectionOf 뿐만이 아니라 unionOf, complementOf 등의 관계도 기술 할 수 있다. 이렇게 다양하면서도 복잡한 관계를 나타낼 수 있는 것이다. wine 온톨로지의 사용을 조금 더 자세히 살펴보면 Wine Ontology에 있어서 손님이 어떠한 음식을 주문한다고 하면 시맨틱 웹 환경의 Wine 에이전트는 이 손님이 시킨 음식에 맞는 Wine을 제공해 주게 된다. 즉 어떤 음식을 주문하면 이 음식이 가지는 특성과 어울리는 wine을 찾아서 에이전트가 스스로 처리하는 것이다. 이러한 추론 능력을 가지게 끔 하는 것이 OWL이 추구하고자 하는 바이다. 추론 능력은 시맨틱 웹에서의 가장 핵심적인 역할이 된다. OWL에서 제공하는 대표적인 추론 기능은 이행적 관계의 기술과 혹은 set 관계에서 나타나게 된다.



a) 실제 관계 b) 모든 이행 관계 c) 직접 관계

그림 12 추론 기능의 예

그림 12는 OWL에서 사용하는 이행적 추론 기능에 대한 설명을

나타내 준다. 그림 12의 a)는 실제 클래스들의 관계를 나타내 준다. a)의 관계에서 가능한 모든 이행 관계를 고려하면 b)의 관계가 된다. 하지만 추론 기능이 가능하다면 c)의 관계만으로 b)의 모든 관계를 계산 할 수 있게 된다. 이러한 기능을 하는 것이 시맨틱 웹에서의 추론 기능이 되는 것이다. 이 추론 기능이야말로 시맨틱 웹에서의 목표인 “machine understandable”을 실현하는 방안이 되게 되는 것이다. 예를 들면 Jena2에서는 Reasoner라는 시스템을 구현하여 이러한 모든 이행 관계에 대해 메모리에 올려서 사용하고 있다. 따라서 한 번의 계산으로 이러한 이행 관계를 구하면 사용자 질의를 구할 시 계속 계산하는 것이 아니라 이미 구해서 메모리에 올려진 결과를 사용하게 되므로 효율적으로 처리하게끔 한다.

3. 데이터 모델과 질의 언어

본 논문에서 다루는 데이터는 OWL이다. 이것은 그래프 모델의 형태를 가진다. OWL이 RDF 기반의 구조를 가지고 있기 때문이다. 즉 노드와 간선을 가지며 노드 뿐만이 아니라 간선에도 값을 가진다는 것이 XML과의 차이점을 가진다.

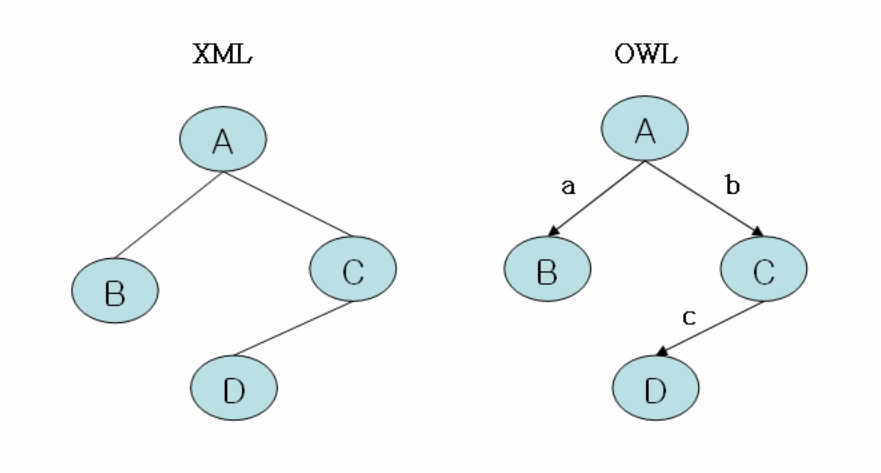


그림 13 XML 구조와 OWL 구조

그림 13은 XML의 구조와 OWL의 차이점을 나타내주고 있다. XML은 트리 구조이다 즉 노드와 간선으로 이루어져 있어도 간선에 방향성과 레이블이 없다. 하지만 OWL은 간선에 방향성이 존재하며 레이블이 있는 것이 차이점을 보인다.

<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 14 OWL 파일의 예 (Gene Ontology)

OWL 데이터의 구조 형태는 그래프 모델이다. 그림 14는 Gene Ontology[13]의 OWL 파일의 일부분으로 Plastid Chromosome(색소 염색체) 부분을 나타내주고 있다. 이 OWL 문서를 그래프 모델로 표현한 것이 그림 15이다.

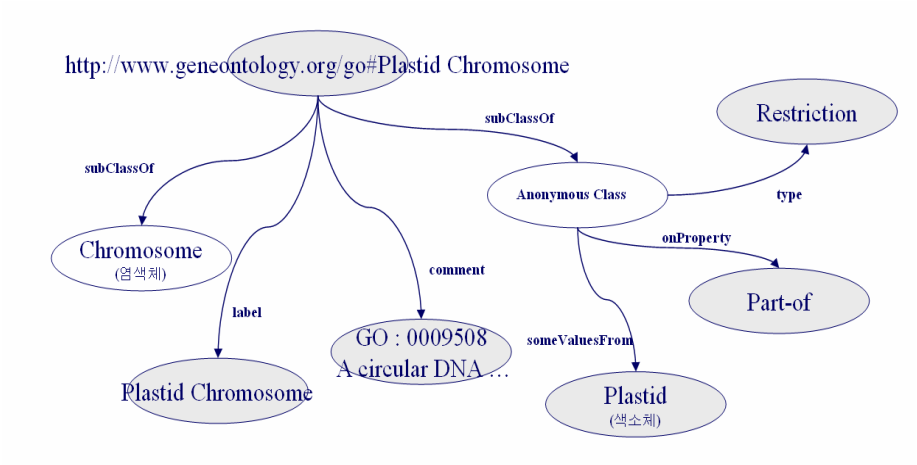


그림 15 그래프 모델 (Gene Ontology)

그림 15에서 최상의 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 관계와 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 데이터 베이스에 저장되어 있는 대용량의 OWL 데이터를 가정하고 있다. 즉 Persistent 모델로 triple 구조로 관계형 데이타베이스에 저장되어 있는 것이다. 데이터베이스에 저장되어 있는 Triple 구조는 노드와 간선 그리고 이 간선이 지시하는 노드로 구성되어 있다. 따라서 질의 또한 그래프의 노드를 찾거나 혹은 서브 그래프 또는 그래프 경로를 찾는 것을 질의어로 갖게 된다. 이러한 질의를 처리하기 위해서는 그래프 탐색을 해야만 한다. 그래프 탐색의 처리는 데이터베이스의 조인 연산을 필요로 한다. 데이타베이스에 triple 구조로 저장이 되어 있기 때문에 각각의 그래프 탐색이 조인 연산이 되는 것이다. 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 시그니처에 매칭되는 클래스에 속한 노드만 탐색하겠다는 것이다. 매칭되지 않는 클래스에 속한 노드는 탐색하지 않고 다음 클래스로 넘어감으로서 이와 같은 방법으로 탐색 횟수를 줄여 빠르게 OWL을 처리 하는 것이 본 논문의 목적이다.

4. 시스템 구현

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.

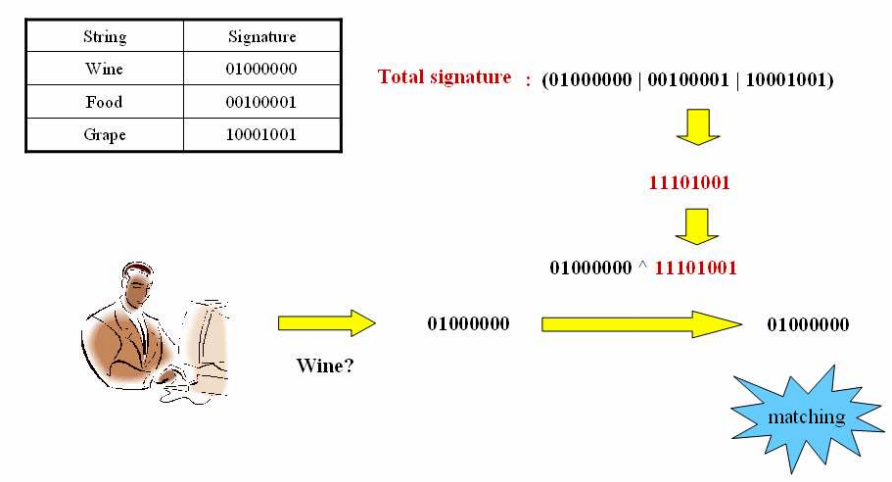


그림 16 시그니처를 이용한 질의 처리

예를 들어 그림 16에서와 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 Pattern 시그니처를 별도로 두어서 사용하자는 것이다. 이렇게 두 개의 시그니처를 별도로 두어서 사용하는 것이 본 논문에서 제안한 방법인 것이다. 그림 17은 이러한 시그니처 구조를 표현한다.

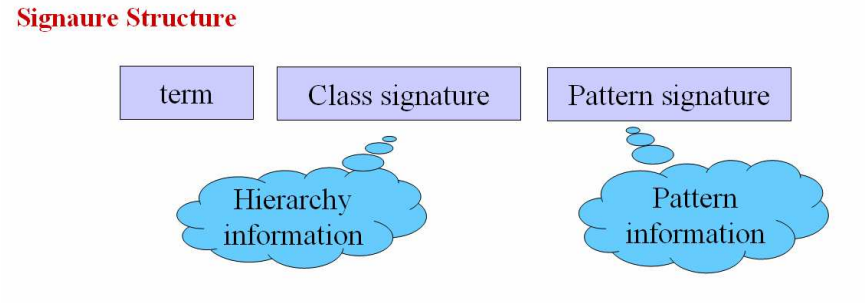


그림 17 OWL에서의 시그니처 구조

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 수행시 효율적으로 처리할 수 있다. 즉 질의문을 수행할 시 시그니처를 우선 검색하여 해당 조건에 만족하는 객체들을 우선 찾아서 검색을 수행 하는 것이다. 그림 18은 전처리 과정에 해당하는 것을 보여준다.

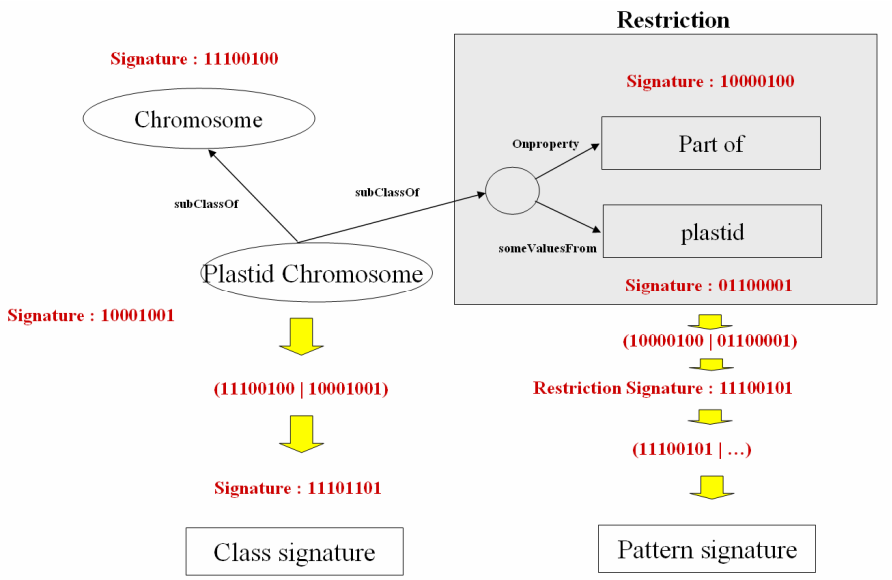
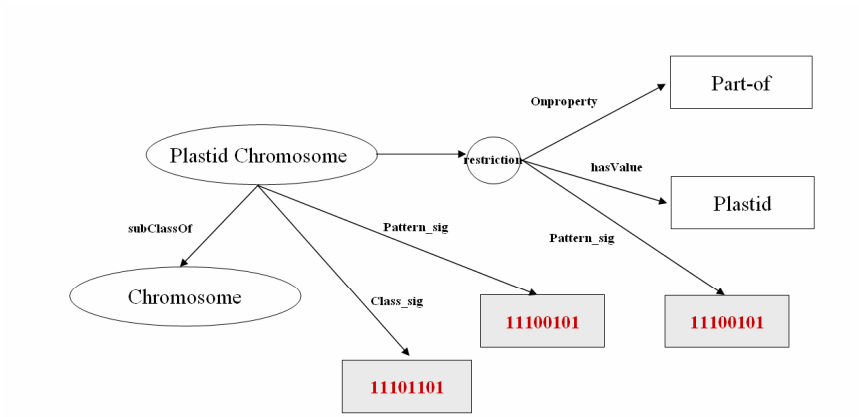


그림 18 OWL 데이터에 대한 시그니처 전처리

4.3 OWL에서의 시그니처를 저장

이렇게 전처리 계산된 시그니처는 데이타 베이스 시스템에 저장되어야 한다. 그래서 질의가 수행될 때 이 저장된 시그니처를 이용해야 한다. 시그니처를 별도의 데이타 베이스 시스템에 저장할 수 있으나 이는 효율적인 수행에 있어서 문제점을 가지게 된다. 그래서 본 논문에서는 시그니처를 별도로 다른 데이터 베이스에 저장하는 것이 아니라 그래프 모델 내에 데이터와 같이 직접 저장하는 방식을 택했다. 즉 전처리된 결과를 하나의 노드와 간선을 만들어서 그 노드에 추가 삽입시키는 것이다. 이는 Jena2에서 제공하는 addProperty란 메서드를 통해서 쉽게 사용할 수 있다. 즉 class_sig는 클래스 시그니처 값을 가지는 간선이 되고 pattern_sig는 패턴 시그니처 값을 가지는 간선이 되는 것이다. 이는 각각의 전처리 과정에서 구해진 결과가 곧바로 데이터 베이스에 저장되는 것이다. 우선 Restriction에 Property들의 시그니처값들의 OR 연산을 한 결과가 pattern 시그니처에 저장이 되고 이러한 Restriction들이 자기가 속한 클래스의 pattern 시그니처 값으로 저장이된다. 이때 클래스가 여러 개의 Restriction을 가지고 있으면 이 여러 개의 Restriction들의 시그니처값들은 OR 연산을 하여 클래스가 가지고 있는다. [8]에서 사용한 시그니처의 기법과 유사하다고 할 수 있으나 [8]에서는 상위 노드가 하위 노드의 모든 시그니처 값들을 가지고 있는 것이고 본 논문에서는 클래스가 Restriction의 시그니처 값들을 가지는 것이고 이 Restriction들의 pattern 시그니처는 하위 노드들의 값을 가지는 것이므로 [8]과의 가장 큰 차이점이 된다. 그림 19는 이러한 시그니처의 저장을 나타내며 Triple 데이터

베이스에 저장되는 방식을 나타내 주고 있다.



Statement Table

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

그림 19 시그니처의 저장

4.4 OWL에서의 시그니처를 이용한 질의 처리

시그니처가 데이터베이스에 저장이 되면 이 시그니처를 질의 처리시 사용해야 한다. 시그니처를 저장해 두는 이유가 질의 처리시 사용해야 하기 때문이다. 본 논문에서 가정하는 질의어의 형태는 클래스의 어떤 속성을 찾는 단순 질의가 아니라 어떤 속성을 가지는 클래스를 찾는 형태로 한다. 예를 들면 색소 염색체는 어떤 클래스에 Part-of 관계를 가지는 질의가 아니라 어떤 클래스에

Part-of 관계를 가지는 클래스를 찾는 다는 의미가 된다. 실제 시맨틱 웹에서 사용하는 질의는 클래스의 속성을 찾는 것보다 제한적인 속성을 가지는 클래스를 찾는 질의가 주로 사용되기 때문이다. 이는 추론 기능과 같이 사용 될 때 많이 사용되기도 한다. 만약에 색소체를 Part-of로 가지는 클래스를 검색하라는 질의를 처리한다고 하자. 일단 사용자 질의에서 클래스와 사용자가 찾고자 하는 속성의 시그니처를 구한다. 즉 색소체의 시그니처 값과 Part-of의 시그니처 값을 구해서 이 시그니처를 가지고 질의를 처리하는 것이다.

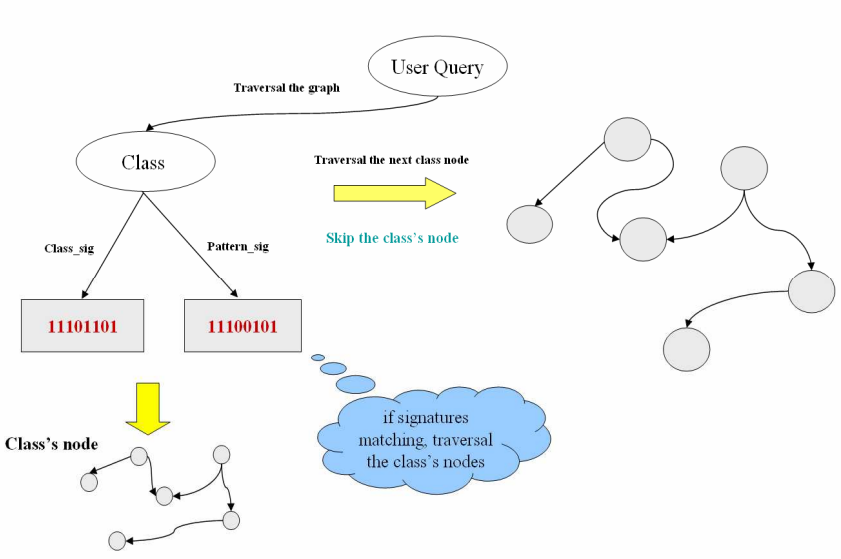


그림 20 시그니처를 이용한 질의 처리 과정

우선 클래스를 찾아서 그 클래스의 pattern 시그니처를 검사한다. 이

클래스의 pattern 시그니처의 값과 색소체 시그니처 값과 Part-of 시그니처 값의 OR 연산한 결과를 AND 연산을 한다. 결과가 색소체와 Part-of의 시그니처를 OR 연산한 결과와 같다면 이 클래스가 답이 될 수 있으므로 탐색을 허용하고 그렇지 않을 경우 없다는 것을 보장하므로 탐색을 하지 않고 다음 클래스를 검사하게 된다. 그림 20은 이와 같은 질의 처리 과정을 나타내 주고 있다.

```
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 21 시그니처를 이용한 질의 처리 알고리즘

클래스와 Property 의 복합적인 구성으로 이루어진 질의에 있어서도 마찬가지이다. 염색체의 하위 클래스이면서 색소체를 Part-of의 값으로 가지는 클래스를 검색한다고 하면 우선 Part-of를 색소체 값으로 갖는 클래스를 pattern 시그니처를 이용하여

검색하고 이 클래스들의 클래스 시그니처를 이용해서 매칭되는 값에 해당하는 클래스만 탐색을 허용한다. 이러한 복잡한 질의 처리에 있어서 검색하는 알고리즘은 그림 21과 같다.

5. 실험 결과

5.1 실험 환경

본 논문의 실험은 Pentium Ⅲ 803MHz CPU, 512MB memory, Window XP operating system 그리고 Java로 코딩된 Jena2와 mysql을 저장시스템으로 사용하는 Persistent Model을 이용하여 수행하였다. 실험 데이타로는 Gene Ontology[13]의 termdb가 사용되었다. 실험에 사용된 데이터는 10Mbytes의 크기를 가지며 Gene Ontology는 현재 가장 활발히 사용되는 ontology이다.

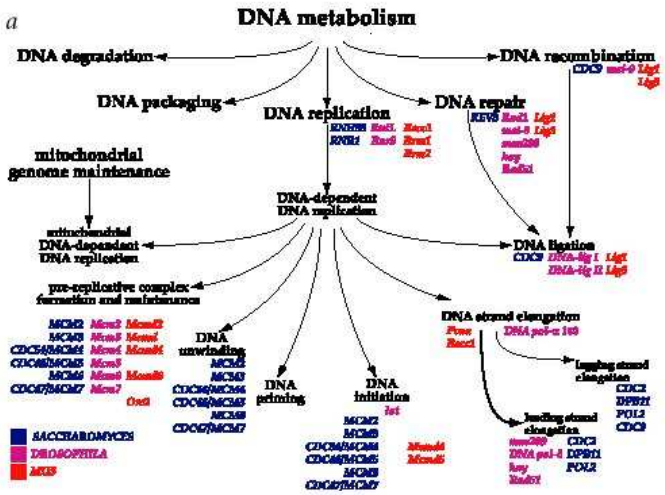


그림 22 Gene Ontology 구조의 예

그림 22는 Gene Ontology의 구조를 나타내주고 있다. Gene Ontology는 생물 정보 데이터베이스의 주석이 주된 목적이다. 하지만 Gene Ontology는 외부 연구 기관이나 표준화 기관에서 제안된 것이 아니라 데이터베이스 그룹 안에서 시작하여 형성되었다. 또 Gene Ontology는 데이터베이스 그룹들에 포함될 수 있는 모든 분자 생물 지식을 표현하려고 하지 않고, 하나의 생물체를 중심으로 그 안에서 유전자들의 역할에 대한 정보만을 표현하려고 하기 때문에, 광범위하지 않고 비교적 좁다. Gene Ontology는 유전자의 기능을 크게molecular function, biological process, cellular component 의 범주로 나누고 각각의 범주에 계층적인 controlled vocabulary를 확립하였다. 이들 범주는 서로 배타적인 것이 아니며, 한 개의 유전자를 묘사하기 위한 특징들을 나누는 범주이다. 원래 Gene Ontology는 RDF 문서로 작성되어있지만 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. 즉 GONG 프로젝트에서 만든 DAML+OIL 문서를 OWL 문서로 바꾸어서 실험을 하였다. 실험에서 사용한 질의는 그림 23의 4개의 질의를 사용하였다. 본 논문에서 가정하였듯이 단순 질의 즉 클래스간의 관계가 없는 것 혹은 Restriction을 가지지 않는 질의는 포함하지 않았다.

Q1	Find class that is ‘part-of’ relationship with ‘secretory+ pathway’
	(type ?x class)(subClassOf ?x ?y) (onProperty ?y “Part-of”)(type ?y restriction) (someValuesFrom ?y “secretory+ pathway”)
Q2	Find class that is ‘part-of’ relationship with ‘nuclear+ membrane’ and ‘nuclear+ envelope -endoplasmic+ reticulum+ network’
	(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 “nuclear+ envelope -endoplasmic+ reticulum+ network’”)
Q3	Find all super classes of ‘plastid+ chromosome’
Q4	Find class that is subclass of ‘cell+ growth+ and%2For+ maintenance’ and is ‘part-of’ relationship with ‘cell+ growth’

그림 23 실험에 쓰인 질의

첫번째 질의는 하나의 Restriction을 포함하는 클래스를 찾는 질의이고 , 두번째는 다수의 Restriction을 포함하는 클래스를 찾는 질의어다. 세번째 질의는 클래스간의 관계를 묻는 질의다. 마지막 질의어는 클래스와 클래스간의 관계와 Restriction을 포함하는 클래스를 찾는 질의이다. 즉 앞의 세 질의는 pattern 시그니처에 대한 성능을 시험하는 것이고 마지막 질의는 class 시그니처를 시험하는 실험이다. 실험에 사용된 질의는 OWL-QL의 형태를 가진다. 그림 23의 Q1, Q2는 OWL-QL[16] 형태로 표현이 된 것이다. OWL-QL은 DQL[17]을 OWL에 맞게 개선시킨 질의어이다. Q3와 Q4 역시 OWL-QL[16]의 형태로 표현되나 간략한 형식으로 표현하였다. OWL-QL은 실험은 위의 질의를 본 논문에서 제안한 시그니처 기법과 Jena2에서 지원하는 Naïve한 방법, jena2에 인덱스를 사용하는 방법을 비교하였다. 인덱스는 관계형 데이터베이스에서 사용하는 B+-tree 인덱스를 Restriction에 사용한 것을 나타낸다. Naïve 방법에서는 Jena2에서 지원하는 Subject, Predicate, Object 컬럼에 사용하는 인덱스를 사용하는 것을 의미한다. 즉 관계형 데이타베이스 시스템에서 사용하는 인덱스를 역시 의미한다.

5.2 실험 결과

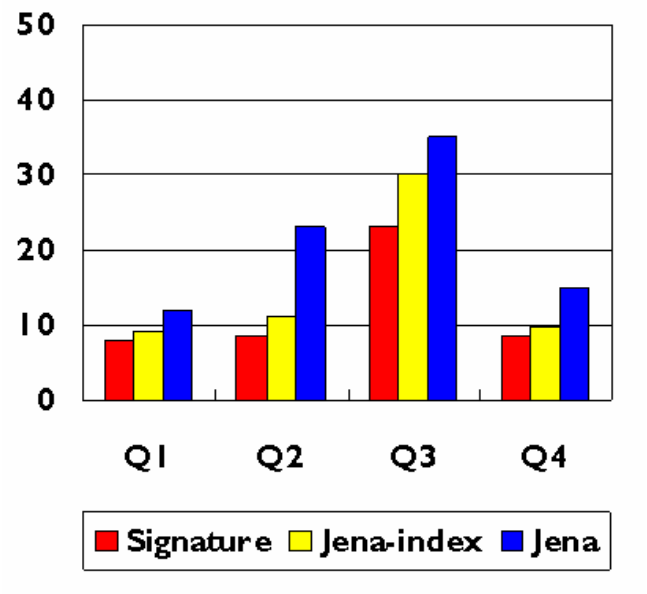


그림 24 수행 시간 성능 비교

그림 24는 Gene Ontology에 위의 질의를 사용한 수행 시간을 나타내 주고 그림 25는 각 질의의 그래프 탐색된 노드의 수를 나타내 준다. 그림 26은 그림 25의 결과에 기반하여 Naïve한 방법에 대한 시그니처와 jena2+index의 노드 탐색 비율을 나타낸다. 그림 11의 실험 결과에서 모든 질의에 있어 시그니처 기법을 사용한 것이 더 좋은 성능을 나타내는 것을 알 수 있다. 이러한 결과를 그림 12를 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.

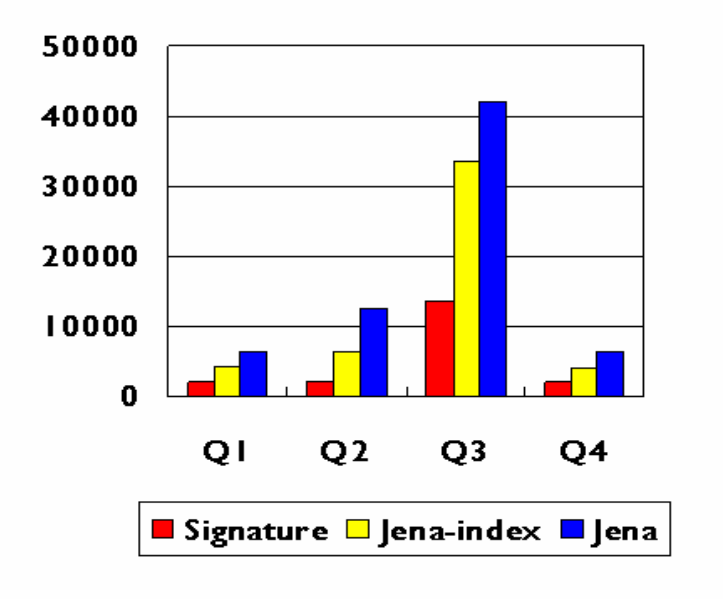


그림 25 그래프 탐색 노드 탐색 횟수

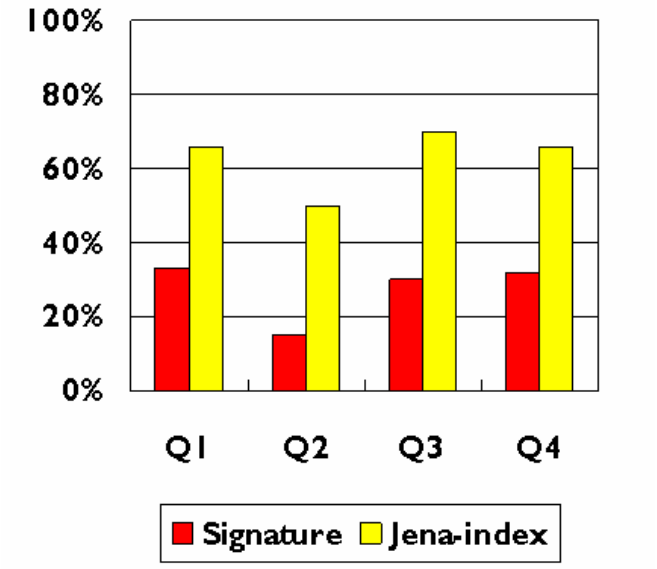


그림 26 노드 탐색 횟수 비율 성능 비교

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 사용되었다. 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 두가지로 볼 수 있다. 하나는 전처리를 필요로 한다는 것이고 두번째는 데이터 그래프의 노드가 증가한다는 것이다.

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

라 하면 노드의 증가율은 다음과 같다.

노드의 증가율 I = (Ni+ Ci*2+ Ri) / Ni 가 된다.

즉 클래스는 클래스 시그니처와 pattern 시그니처 즉 한 개의

클래스당 2개의 노드가 증가하게 된다. 또한 Restriction의 개수만큼 pattern 시그니처를 가지게 되므로 위와 같은 증가율이 나타나게 된다. 실제로 Gene Ontology의 termDB인 경우 실제 데이타베이스의 크기가 약 21MByte 이지만 전처리 단계를 거친 후에는 약 1.1MByte가 증가한다. 이는 데이터 크기에 비하면 5%에 해당되는 작은 크기이다. 시그니처를 사용하는 가장 큰 장점은 적은 공간을 가지고서 효율적으로 탐색을 줄일 수 있는 것이다.

6. 결론 및 향후 연구 방향

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 데이터베이스에서 Join연산으로 이루어지기 때문에 성능이 저하된다. 따라서 본 논문에서 제안한 질의 처리 기법은 대용량의 OWL을 다루는 질의 처리 시스템에서 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 데이터들은 깊이가 그리 깊지 않은 형태로 되어있다[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. 즉 모든 추론 관계까지 고려해서 시그니처를 계산하여 저장하여 이용하도록 하는 것이다. 또한 패턴 매칭 알고리즘과 결합시켜 유사도 검색이나 온톨로지의 병합과

같은 새로운 문제에도 본 논문에서 제안한 기법이 활용 될 수 있을 거라고 생각한다.

참고 문헌

[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]Richard Fikes, Pat Hayes, Ian Horrocks, “OWL-QL: A Language for Deductive Query Answering on the Semantic Web”, KSL Technical Report 03-14, Stanford University.

[17] A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, “Benchmarking RDF Schema for the Semantic Web”, ISWC, 2002

부 록

A. UML Diagram

Use Case Diagram

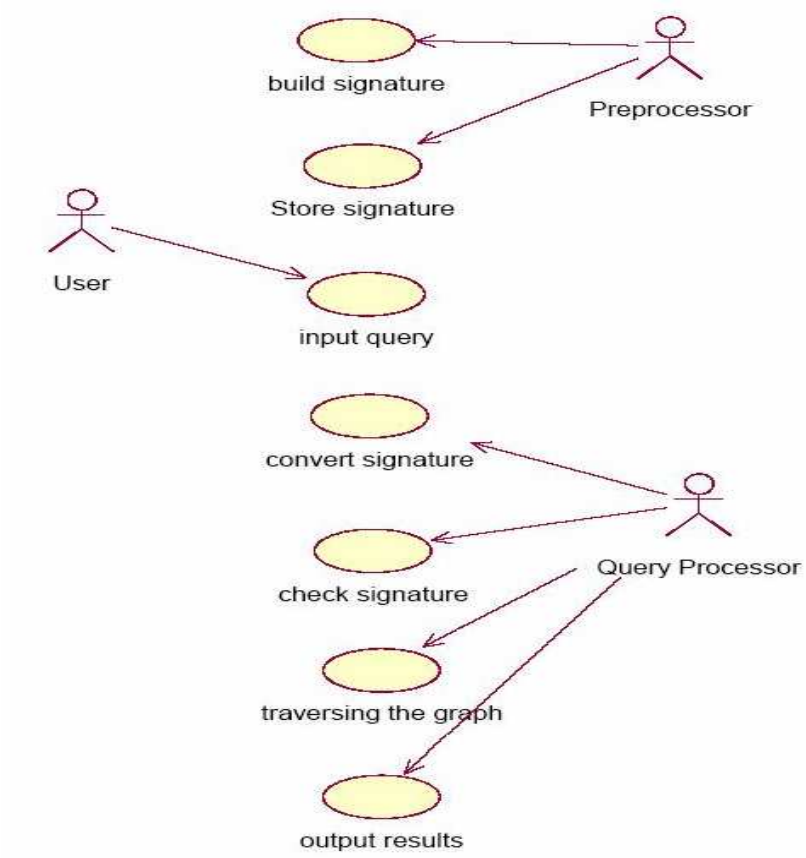


그림 27 유스 케이스 다이어그램

Class Diagram 1

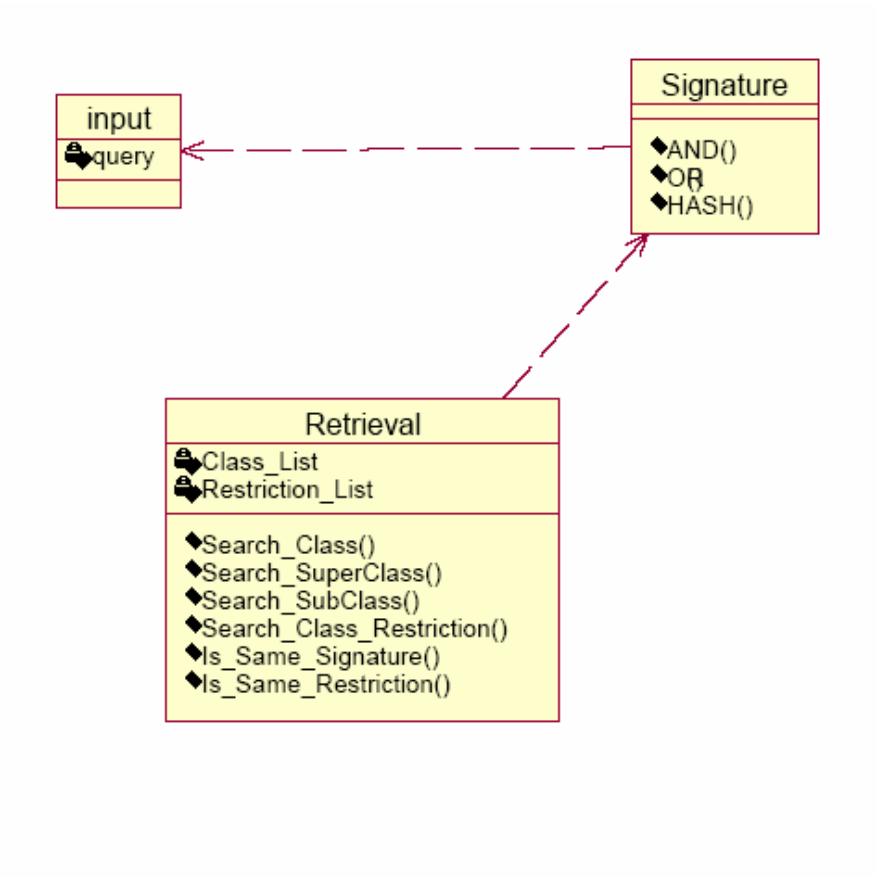


그림 28 시그니처 질의 처리 클래스 다이어그램

Class Diagram 2

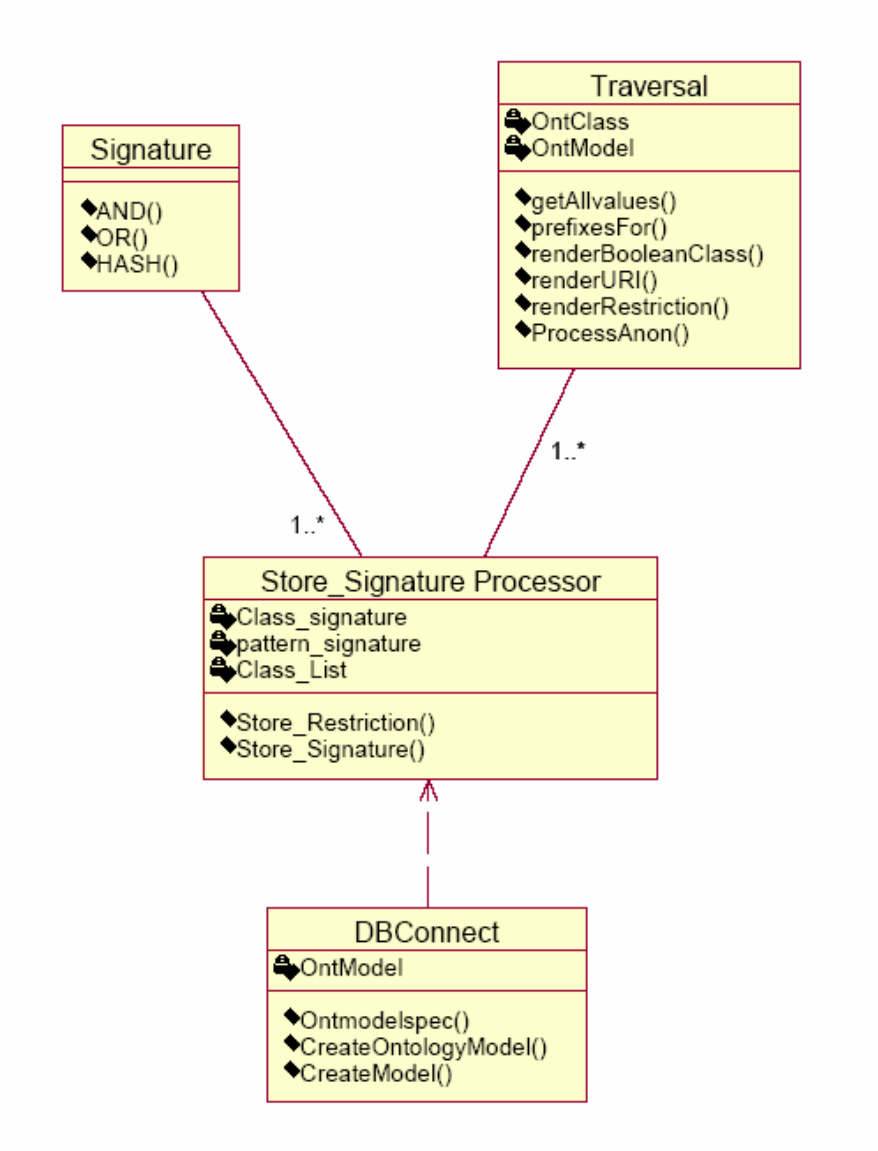


그림 29 시그니처 전처리 클래스 다이어그램

Sequence Diagram 1

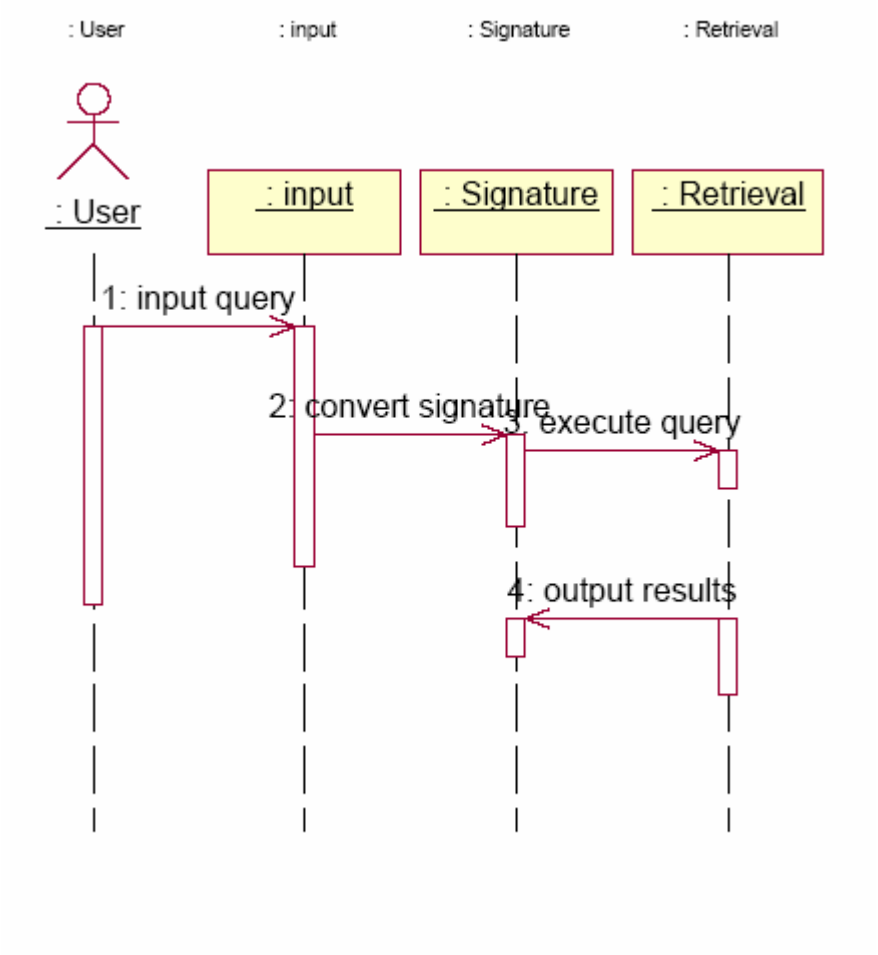


그림 30 사용자와 프로세서간의 시퀀스 다이어그램

Sequence Diagram 2

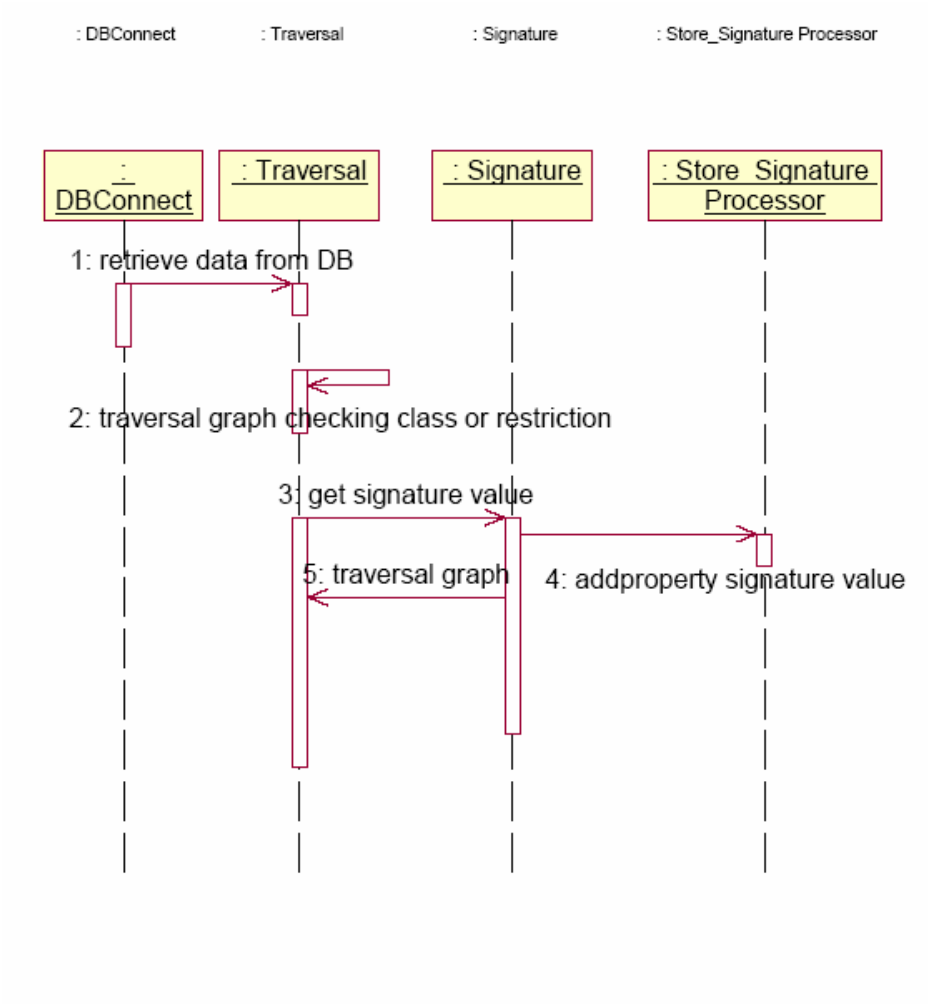


그림 31 시그니처 전처리 시퀀스 다이어그램

Sequence Diagram 3

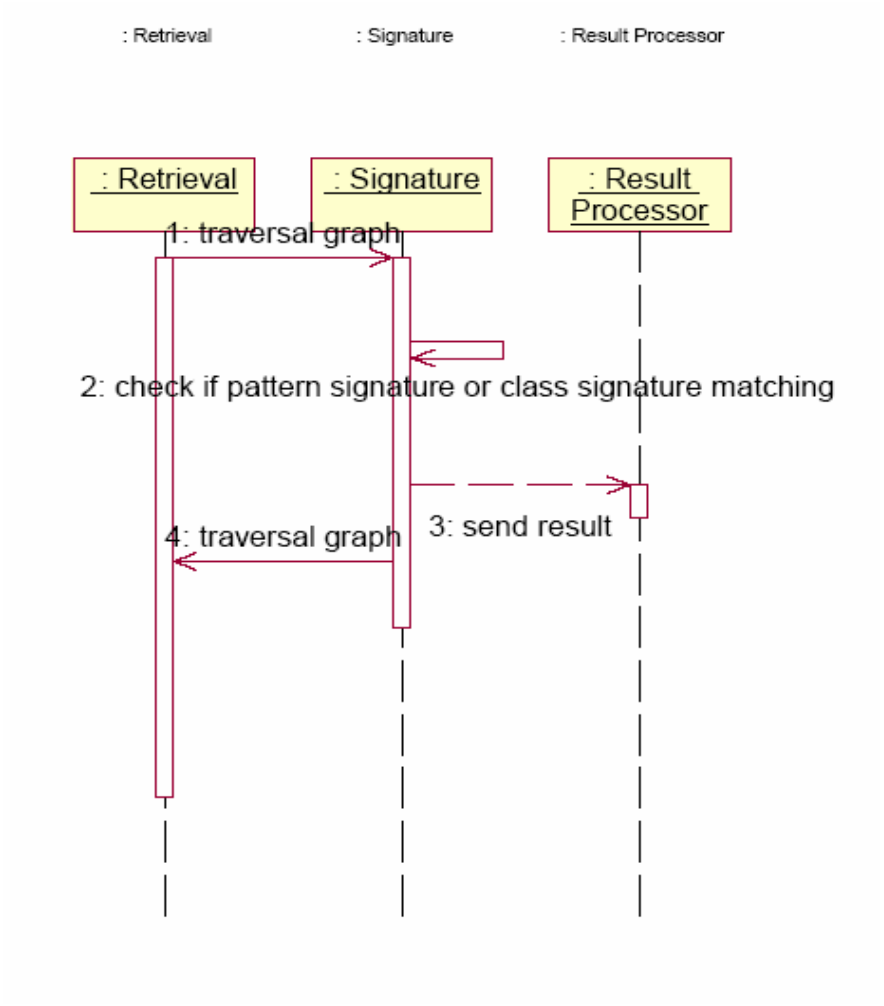


그림 32 시그니처 질의 처리 시퀀스 다이어그램

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.

Using signatures composed of the class signature and the pattern signature, we compare signatures for query processing and we can traversal the graph when signatures is matching. By this technique, the oveall performance is greatly improved for the previous approach.

Keywords : OWL, Signature, Ontology, Optimization

Student Number : 2003 - 21675