工學碩士學位論文

효율적인 RDF 질의 처리를 위한 RDF-Schema Domain과 Range정보 기반의 데이타 탐색 범위 감소 기법

Reducing Search Space Scheme using RDF-Schema Domain and Range Information for Efficient RDF Query Processing

2005年 2月

서울大學校 大學院

전기 • 컴퓨터 工學部

金 成 泰

효율적인 RDF 질의 처리를 위한 RDF-Schema Domain과 Range정보 기반의 데이타 탐색 범위 감소 기법

Reducing Search Space Scheme using RDF-Schema Domain and Range Information for Efficient RDF 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

金 成 泰

金成泰의 工學碩士 學位論文을 認准함

2004年 12月

委 員 長 이 석 호 ㊞

副委員長 김 형 주 ㊞

委 員 이 상 구 ㊞

요 약

웹의 발전 방향인 시멘틱 웹은, 웹(Web)상의 자원(Resource)에 잘 정의된 의미(Semantic)를 제공함으로써 사람뿐만 아니라 컴퓨터도 문서의 의미를 해석할 수 있게 해 준다. RDF(Resource Description Framework)는 웹 자원들에게 의미적 연결성을 제공하며 트리플(Triple)구조를 사용하여 메타 데이타(Meta-data)를 표현하며 RDF-Schema는 리소스의 구조나 리소스들간의 관계를 표현할 수 있는 미리 정의된 어휘(Vocabulary)이다. 본 논문에서는 대용량의 시멘틱 웹 문서를 관계형 데이타베이스 시스템에 저장할 때 RDF-Schema 어휘인 rdfs:domain과 rdfs:range가 가지는 특정 시멘틱 정보를 이용하여 기존의 관계형 데이타베이스시스템에서 트리플 데이타를 저장할 때의 문제점을 해결하는 기법을 제안한다. 또한, 질의 처리시 본 논문에서 제안한 방법을 사용할 경우 불필요한 데이타의 검색을 줄일 수 있으므로 효율적인 질의 수행 결과를 얻을 수 있다.

주요어 : 시멘틱 웹, RDF/RDFS, OWL, Domain, Range, 질의처리.
학 번 : 2003 - 21539

목 차

그림 목차

1. 서론

1989년 팀 버너스-리(Tim Berners-Lee)가 제안한 월드 와이드 웹(World Wide Web)은 여러 기술들과 더불어 현재의 웹을 구성하고 있으며, 이는 컴퓨터 시스템들간의 정보교환을 가능하게 해 주었다. 특히 누구나 쉽게 배울 수 있는 HTML 언어를 사용함으로써 웹에서 다루는 정보의 양은 기하급수적으로 늘어나게 되었다. 그러나 사용자를 위한 간단한 HTML의 표기법은 웹 브라우저를 통해 표현되는 문서를 단지 사람의 눈에 보기 편리하게 할 뿐이지 그 문서의 구조나 의미적인 정보는 표현할 수가 없다. 이러한 결과 웹에서 검색한 결과는 그 정확도의 저하와 함께 컴퓨터 프로그램이 웹 문서의 의미를 자동으로 파악하는 것은 사실상 불가능 하게 되었다. 가령 "약혼 기념일을 위한 식사코스 중, 점심 메뉴의 리스트를 제공하고 각 메뉴에 어울리는 와인을 추천하되, 검은색 빛깔의 와인은 제외하라" 라는 질의는 키워드 방식의 검색을 할 수가 없다. 이런 질의가 가능하기 위해서는 웹 자원의 의미적 기술을 전제 조건으로 한다[1].

이러한 요구사항 아래 W3C가 주도해온 시멘틱 웹(Semantic Web)[2] 활동은 컴퓨터가 웹상의 자원을 이해할 수 있도록 메타 데이타를 통해 웹 자원들을 지식화 하여 찾고자 하는 정보를 정확하고 효율적으로 검색하고 통합 재사용 할 수 있도록 하는데 있다. 시멘틱 웹의 궁극적인 목적은 웹에 있는 데이타를 컴퓨터가 좀 더 잘 이해할 수 있도록 하여 정확한 검색과 추론을 통한 정보의 새로운 정보를 도출해 내는 것이다. 이러한 시멘틱 웹에서 가장 핵심이 되는 것은 온톨로지(Ontology)[3]이다. Gruber는 온톨로지를 "공유된 개념화에 대한 정형화되고 명시적인 명세(Formal explicit specification of a shared conceptualization)" [3]라고 정의하였다. 즉, 온톨로지는 도메인 상에서의 어휘와 개념들 간의 상호관계를 정의하는 것이다.

온톨로지를 표현하기 위해 문법적 구조와 스키마 등을 정의한 언어가

온톨로지 언어(Ontology Language)이다. 현재 RDF/RDFS[4], DAML+OIL[5], 그리고 OWL[6]등이 있다. 가장 기본적인 시멘틱 웹 언어인 RDF/RDFS는 웹 리소스들에 관한 메타정보를 표현하기 위한 언어이다. DAML+OIL은 DAML(DARPA Agent Markup Language) 프로그램의 DAML-ONT[7]와 유럽에서 개발된 OIL(Ontology Interface Layer)[8]의 결합을 통하여 만들어졌으며 OWL은 현재 가장 주목 받고 있는 온톨로지 언어로서 DAML+OIL의 속성과 클래스 이름 그리고 네임스페이스 등을 변경하고 RDF및 RDF-Schema의 변화를 받아들였다. 온톨로지를 잘 활용하기 위해서는 OWL과 같은 언어로 표현된 온톨로지 데이타의 효율적 저장과 질의에 대한 연구가 무엇보다 중요하게 되었으며 현재 이들에 관한 연구가 진행 중에 있다[9,10,11]. 온톨로지를 구축하는데 있어서 그 기본 구조는 트리플 구조를 갖는다. 이를 Statement라고 하며, 주어(Subject), 술어(Predicate), 목적어(Object)로 구성된다. 온톨로지는 트리플 구조를 기반으로 하는 그래프(Graph)로 모델링 된다. 즉, 수많은 트리플들이 서로 연결되어 하나의 큰 그래프를 형성하게 되는 것이다. 그래프로 모델링 되는 온톨로지 데이타를 영구 저장소(Persistent Storage)에 저장하기 위해서는 기본 단위를 이루는 트리플 단위로 저장한다. 현재, 저장구조 시스템으로는 Jena2[10]에서 제공하는 영구 데이타 모델(Persistent Data Model)과 On-To-Knowledge의 Sesame[11]이 있으며 이들은 모두 하부 데이타 저장구조로 관계형 데이타베이스 시스템을 지원한다. 이들 모두 관계형 데이타베이스 시스템에 트리플 데이타를 저장하는 하나의 Statement 테이블을 중심으로 데이타베이스 스키마를 설계하여 사용하고 있다. 온톨로지를 구성하는 모든 트리플 데이타가 하나의 테이블에 저장될 경우 데이타를 검색하고 조인 연산을 하는데 있어 그 비용이 크게 된다. 이러한 문제를 해결하기 위해 우리는 RDF-Schema에서 제공하는 스키마 정보인 rdfs:domain과 rdfs:range를 이용하여 다수(Multiple)의 Statement 테이블 구조를 갖도록 데이타베이스 스키마를 재구성한다. 이러한 스키마를 갖는 데이타베이스 시스템에 질의를 수행하면 해당 조건에

맞는 데이타를 찾는데 있어 전체 트리플 데이타를 검색할 필요 없이 해당 트리플 데이타가 존재할 수 있는 Statement 테이블만 검색함으로써 검색 범위를 줄일 수 있어 보다 효율적인 데이타의 검색을 수행할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해 살펴보고, 3장에서는 본 논문에서 사용되는 데이타 모델과 배경 지식에 대해 설명한다. 4장에서는 제안하는 방법에 대한 동기와 그 기법에 대해서 알아본다. 5장에서는 실험 결과를 설명하고 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

RDF/RDFS와 OWL를 이용하여 구축된 대용량의 온톨로지 데이타를 저장 관리하는데 있어, 현재 가장 많이 사용되는 관계형 데이타베이스 시스템에 효율적으로 저장하기 위한 방법이 활발히 연구 중에 있다. 가장 대표적인 것이 Jena2[10,12]와 Sesame[11]등이 있다. Jena는 HP Lab에서 시멘틱 웹 연구를 위해 개발되었으며 시멘틱 웹 애플리케이션(Application)을 만들기 위한 자바 프레임워크이다. 현재 Jena2.1 버전까지 나와 있으며 Jena2.1에서는 RDF/RDFS, DAML+OIL과 OWL을 모두 지원하고 데이타베이스를 이용한 Persistent Model을 제공한다. Jena1에서는 트리플 데이타를 저장하기위해 정규화된(Normalized) 데이타베이스 스키마를 사용하였다. 데이타인 리소스에 대해 한번씩만 저장되기 때문에 공간(Space)면에서는 효율적이기는 하나 검색 연산시 Statement 테이블과 리터럴(Literal)테이블 그리고 리소스 테이블 간에 다수의(Multiple) 조인 연산을 필요로 한다. 이에 반해, Jena2의 경우 비정규화(Denormalized)된 데이타베이스 스키마를 사용하였다[10]. Statement 테이블에 직접 데이타를 저장하고 의도적으로 비정규화를 시킴으로써 테이블 공간이 커지기는 하나, 데이타의 검색 시간과 조인 연산을 그만큼 줄일 수 있다. 추가적으로, 'jena_gntn_stmt'에 저장될 값들 중 일정 크기 이상의 데이타가 들어갈 경우 테이블 크기가 너무 커지는 것을 막기 위해 별도의 테이블인 'jena_long_uri' 와 'jena_long_lit'을 구성하고 이를 참조하는 형태로 구성되어있다. 또한 공통적으로 사용하는 접두사에 대해서는 'jena_prefix' 테이블에 저장하고 이를 참조하도록 한다. Jena2[10] 구조의 핵심은 노드들의 트리플 집합체인 RDF 그래프이며 'jena_gntn_stmt' 테이블이 온톨로지 그래프상의 모든 트리플 데이타인 Statement를 저장하는 구조를 갖는다. 그림 1은 Jena2[10]에서 데이타베이스 시스템을 이용하여 온톨로지 데이타를 저장할 때 사용하는 데이타베이스 스키마를 보여주고 있다.

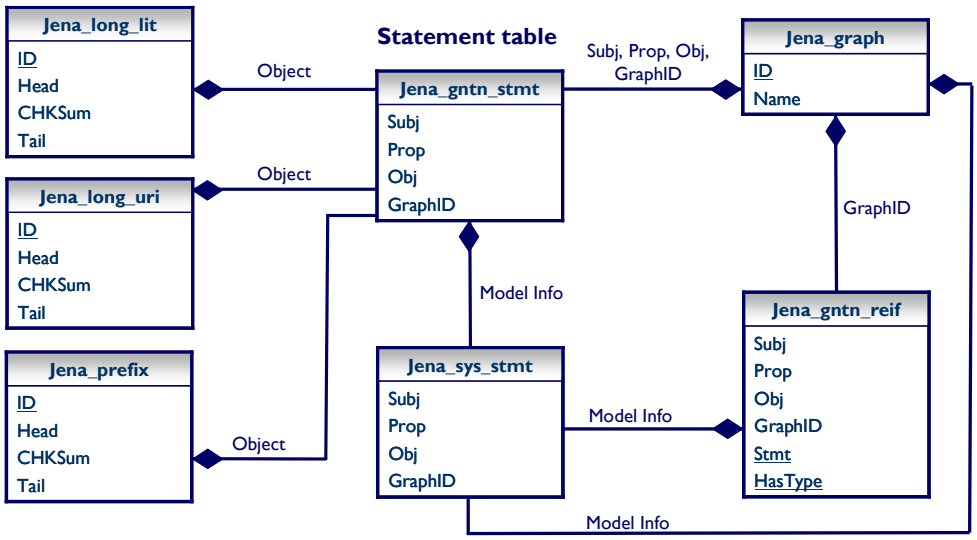

그림 1 Jena2 데이타베이스 스키마

그러나 Jena1과 Jena2 모두 온톨로지 그래프를 구성하고 있는 수많은 트리플을 저장 관리하는데 하나의 Statement 테이블인 'jena_gntn_stmt' 테이블을 사용하였다. 트리플 데이타를 저장하는 테이블을 중심으로 데이타베이스 스키마가 구성되었기 때문에 대용량의 온톨로지를 처리하는데 있어 그 성능이 저하된다. 그림 2는 실제로 Jena2 Database에 온톨로지 데이타가 들어간 상태를 보여준다. Predicate의 값들이 많이 중복되어 있으며 실제 Obj 컬럼의 값은 하나의 문서가 들어갈 수 있는 긴 문자열(Long String) 타입의 값이 올 수도 있다. 또한 어떤 객체에 대한 참조 값이 올 수도 있다. 이러한 특성 때문에 사용자의 질의에 해당하는 튜플(Tuple)을 찾는 연산의 비용은 온톨로지 데이타가 커질수록 증가하게 된다. Jena2[10]에서 'jena_gntn_stmt' 테이블이 새로운 모델이 추가될 때 동적으로 n값이 1씩 증가하여 새로운 테이블이 추가되기는 하나 질의의 종류가 어떠한 하나의 모델에만 국한되지는 않기 때문에 좋은 방법은 아니다. 그러한 결과, 기존의 방식에서는 사용자 질의에 대한 결과를 만들어 내기 위해 'jena_gntn_stmt' 테이블에 대해서 다수의 셀프-조인(Multiple

Self-Join) 연산이 빈번하게 발생한다.

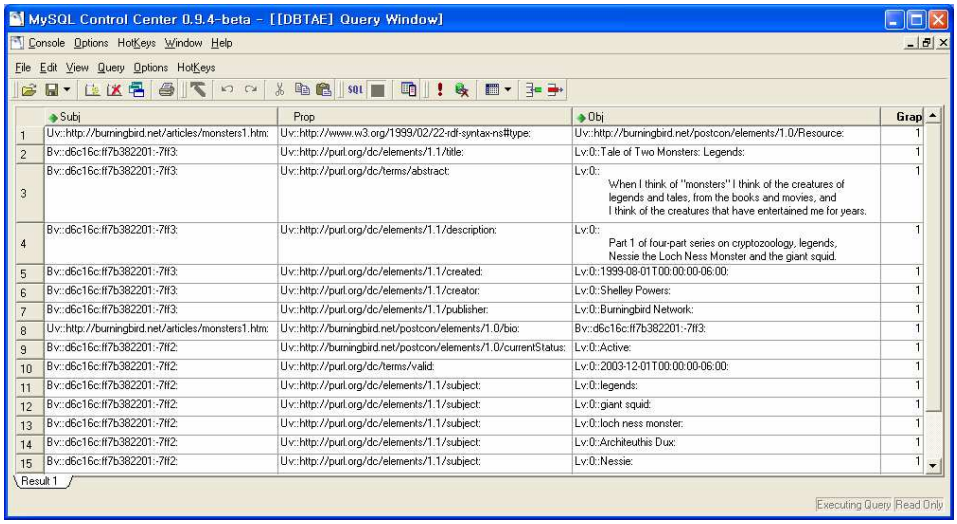

그림 2 Jena2 jena_gntn 테이블

European IST 프로젝트인 On-To-Knowledge의 결과물인 Sesame[11]은 관계형 데이타베이스, 객체지향형 데이타베이스(OODB), 트리플 저장소(Store), 와 같이 다양한 저장 장치에 데이타를 저장할 수 있게 SAIL(Storage And Inference Layer) API를 제공한다. 이는 하부에 어떤 저장소가 사용 되더라도 똑 같은 인터페이스를 제공하는 이점이 있어 저장하려는 데이타의 특성에 최적화 되도록 데이타베이스를 쉽게 바꿀 수 있는 구조를 지원한다.

Sesame[11]은 정규화(Normalization)한 스키마 구조를 가지지만 Jena와 마찬가지로 Statement를 저장하는 Triples 테이블을 중심으로 트리플 데이타가 저장된다. 그림 3은 데이타베이스로 MySQL을 사용할 경우 만들어지는 데이타베이스 스키마이며 트리플 데이타는 하나의 Triples 테이블에 저장되며 이러한 문제로 인해 Jena에서와 같은 문제가 발생한다. 본 논문에서는 기존의 연구에서 트리플 데이타를 저장하기 위해 하나의 테이

블을 사용하던 것을 RDF-Schema를 이용하여 트리플 데아타를 저장하는 데 있어 다수의(Multiple) Statement 테이블을 사용하는 방법을 제안하고 발생하는 문제점의 해결책을 제시한다.

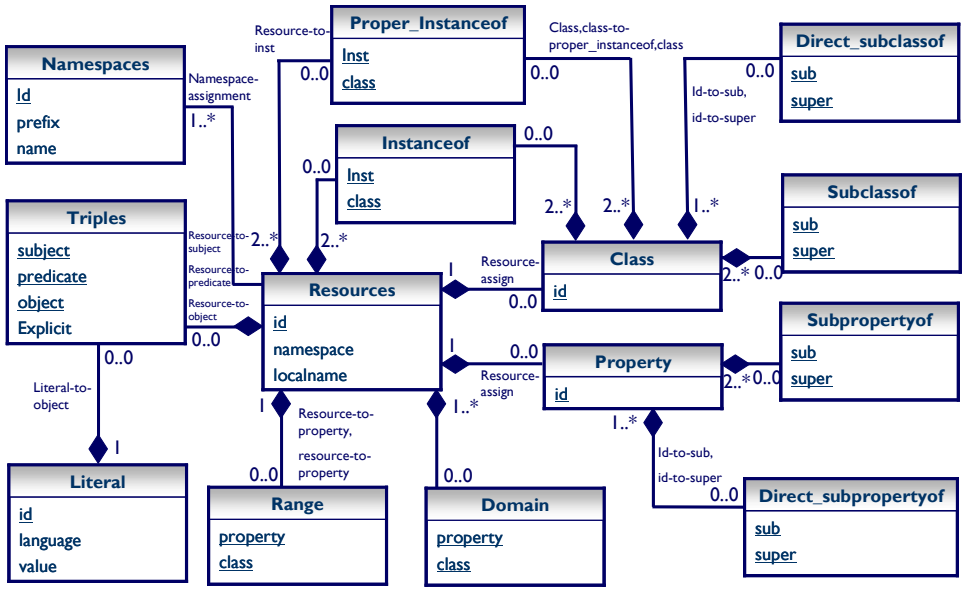

그림 3 Sesame 데이타베이스 스키마

3. 배경 지식

본 장에서는 우리가 사용할 두 온톨로지 데이타의 특성에 대해 설명하고, RDF-Schema에서 제공하는 rdfs:domain과 rdfs:range의 표현 방법과 그 의미에 대해서 알아본다.

3.1 온톨로지 데이타

데이타는 OWL로 구성된 와인 온톨로지[13]와 Gene 온톨로지[14]이다. 와인 온톨로지는 Web Ontology Language(OWL) Reference[13]에서 제공하고 있으며 클래스간의 복잡한 관계를 잘 설정하고 있는 대표적인 온톨로지이다. 그림 4는 와인 온톨로지의 그 일부를 나타낸 것이다.

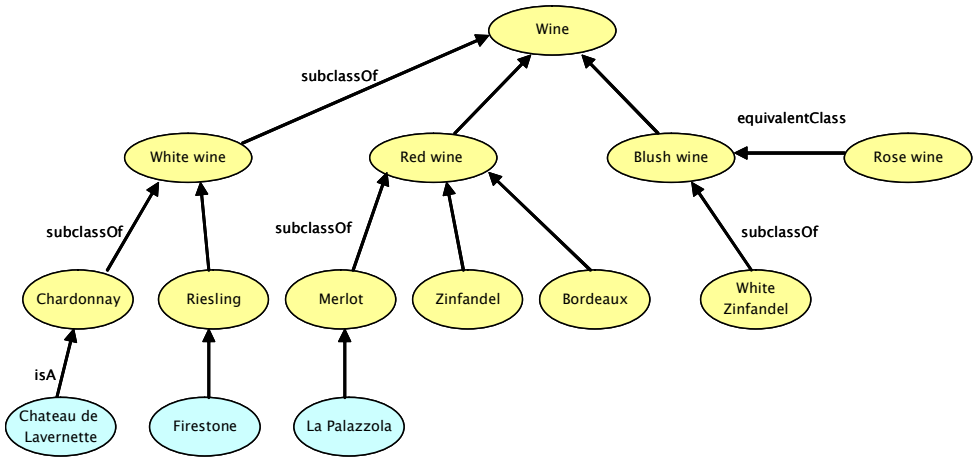

그림 4 Wine 온톨로지 일부

와인은 최상위 클래스 이며 subClassOf로는 Whie 와인, Red 와인 그리고 Blush 와인이 존재하며 Rose 와인은 Blush 와인과 동일한 클래스 범주인

equivalentClass 관계를 갖는다. 또 각각의 와인들은 특정 와인들과 관계를 갖는다. 그림 5는 실제 와인 OWL 문서에 나타나 있는 온톨로지의 일부분이다. 객체형 속성(ObjectProperty)[15]은 클래스 요소들간의 관계를 나타낸다. 첫번째 블록은 객체형 속성인 hasColor에 대한 스키마를 기술하고 있고 중간 블록은 클래스인 WineColor에 대한 스키마를 기술하고 있다. 마지막 아래 3개의 행은 WineColor 클래스에 대한 실제 인스턴스를 나타내며 이들이 나타내는 것은 Wine의 색깔로 "Red", "Rose", "White"가 있다는 것을 의미한다.

```
<owl:ObjectProperty rdf:ID="hasColor">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
    <rdfs:domain rdf:resource="#Wine" />
    <rdfs:range rdf:resource="#WineColor" />
 </owl:ObjectProperty>

  ….

 <owl:Class rdf:ID="WineColor">
    <rdfs:subClassOf rdf:resource="#WineDescriptor" />
    <owl:oneOf rdf:parseType="Collection">
       <owl:Thing rdf:about="#White" />
       <owl:Thing rdf:about="#Rose" />
       <owl:Thing rdf:about="#Red" />
    </owl:oneOf>
 </owl:Class>

 ….

<WineColor rdf:ID="Red" />
<WineColor rdf:ID="Rose" />
<WineColor rdf:ID="White" />
```

그림 5 Wine OWL 문서 일부

Gene 온톨로지는 생물정보 데이타베이스의 주석(Annotation)이 주 목적이다. 이는 생물체를 중심으로 그 안에서 유전자들의 역할에 대한 정보를 표현하며 유전자들이 각 생물체에서 수행하는 역할을 기술하기 위한 구조화된 공통의 어휘(Vocabulary)를 생성한다. Gene 온톨로지의 대략적 스키마는 그림 6과 같다. Term 클래스는 Gene 온톨로지에서 사용되는 용어를 기술하고 Evidence 클래스는 Gene 온톨로지에 대응시킬 때 근거를 명시하며 Dbxref 클래스는 해당 개념이 나타나 있는 외부 DB에 대한 참조를 기술한다.

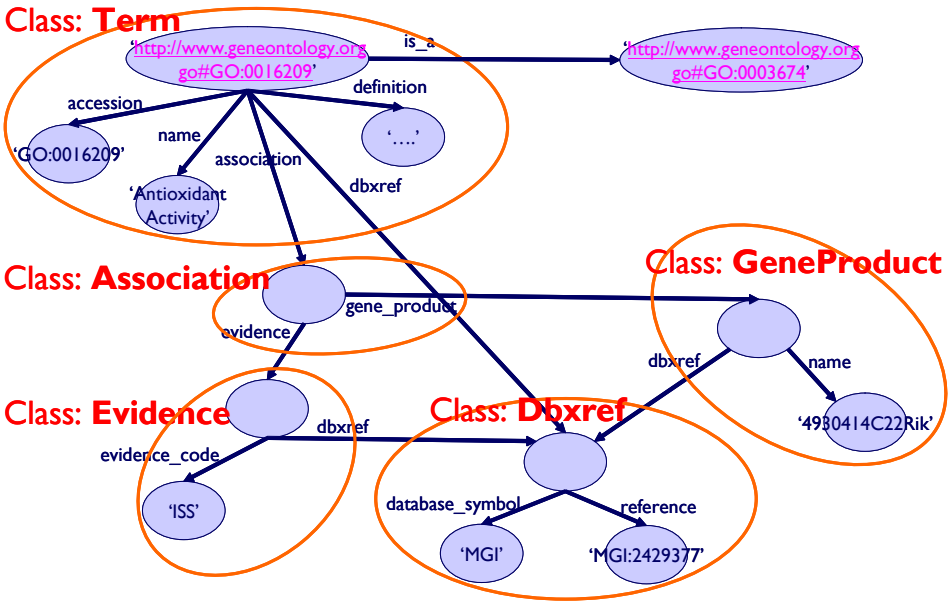

그림 6 Gene 온톨로지 스키마 일부

3.2 Jena 2.0

Jena는 Semantic Web 애플리케이션을 만들기 위한 자바 프레임워크로서 RDF, RDFS, DAML+OIL, OWL 그리고 Rule기반 추론엔진을 포함하는 프로그래밍 환경을 제공한다. 또한 Jena는 오픈소스로서 HP 연구소에서 시멘틱 웹 연구의 일환으로 개발한 툴이며, 현재 2.1버전까지 개발되어 있는 상태이다. 그림 7은 기존의 버전인 Jena1.6.1과 Jena2.0과의 비교 테이블로서 지원하는 문서의 종류와 사용가능한 어휘, 데이타베이스, 추론가능한 언어, Parser 및 데이타터베이스 스키마 구조에 대하여 나타낸다.

	Jena1.6.1	Jena2.0
Document	RDF, RDFS, DAML+OIL	RDF, RDFS, DAML+OIL, OWL
Vocabulary	DC, RDF, RDFS, RSS, VCARD, DAML+OIL	DC, RDF, RDFS, RSS, VCARD, DAML+OIL, OWL, XSD
Database	MySQL, Oracle, Postgresql, Interbase, Berkeley DB	MySQL, Oracle, Postgresql
Interface's Document	bit of DAML+OIL	RDFS, OWL Lite
Parser	ARP parser(RDF/XML)	ARP parser(RDF/XML), OWL Syntax Checker
Query	RDQL	RDQL
Database Schema	rdf_literals	jena_long_lit
	rdf_statements	jena_gntn_stmt
	rdf_resources	jena_graph
	rdf_namespaces	jena_gntn_reif
	rdf_models	jena_long_uri
	rdf_layout_info	jena_prefix
		jena_sys_stmt

그림 7 Jena1.6.1, Jena2.0 비교테이블

Jena2.0의 데이타베이스 스키마는 이전 버전과 다른 구성을 가지고 있다.

3.3 Sesame

Sesame는 RDF데이타와 RDFS 정보의 반영구적인 저장과 그 정보에 순차적인 질의를 할 수 있도록 하기위한 아키텍져이다. Sesame는 European IST 프로젝트 ON-To-Knowledge의 Repository Facility이다. Sesame는 클라이언트(Client)와 통신을 위해 HTTP, Remote Method Invocation(RMI) 또는 Simple Object Access Protocol(SOAP)을 제공한다. 이러한 Protocol의 제어를 위해 HTTP Protocol Handler와 SOAP Protocol Handler가 있다. 클라이언트의 요청을 Request Router는 해당하는 모듈에게 전달하게 된다. 이에 해당하는 모듈에는 RDF데이타/RDFS 정보를 추가하거나 저장소를 비우는 역할을 하는 Admin Module, RQL Query를 처리하는 Query Module, 다른 RDF tool들과의 결합을 위해 제공하는 Export Module들이 있다. 그림 8은 Sesame의 데이타베이스 테이블로서 20개로 구성되어 있다.

	Sesame
Document	RDF, RDFS, DAML+OIL
Database	PostgreSQL, MySQL, Oracle, Memory based model
Interface's Document	DAML+OIL
Query	RQL
Database Schema	Resources
	Namespaces
	Triples
	Allnewtriples
	newtriples
	Addedtriples
	Literals
	Range
	Domain
	InstanceOf
	Class
	Property
	Direct_subclassof

그림 8 Sesame 데아타베이스 특징

Sesame은 Jena와 달리 많은 테이블이 존재한다. Jena2에서와 같이 단순히 트리플 구조만을 저장하는 것이 아니라 트리플 구조가 갖는 여러 의미에 대해서도 저장하기 때문이다. Sesame는 먼저 문서가 들어왔을 때 리소스들을 추출하여 트리플 형태로 변환하여 Triples테이블에 저장하게 된다. 여기에서 트리플은 Subject, Predicate, Object로 구성이 되며 Resources테이블과 Literal테이블을 참조한다. 이때 리소스의 특징으로 리소스가 어디에 쓰일 수 있는지에 대한 domain 과 리소스의 값이 될 수 있는 범위에 대한 range대 대한 정보를 Domain, Range테이블에 넣는다. 그리고 리소스를 인스턴스(Instance), 클래스(Class), 속성(Property)으로 분류하고 클래스에 대해서는 SubClassof와 DirectSubClassof의 정보를 추출하여 저장하고, 속성에 대해서는 SubPropertyof와 DirectSubPropertyof의 정보를 추출 저장하게 된다. Sesame는 많은 테이블을 두어 질의 처리

시 추론을 하지 않고도 테이블에 나와 있는 정보를 통해 처리할 수 있다.

3.4 RDF-Schema rdfs:domain, rdfs:range

RDF-Schema는 RDF의 시멘틱 확장으로서 RDF의 단점을 해결하여 연관된 리소스들간에 그룹(Group)을 기술하거나 리소스들간에 관계(Relation)를 기술 가능하게 한다. 여기서 우리는 우리가 이용할 RDF-Schema 어휘인 rdfs:domain과 rdfs:range에 대해서 알아본다.

3.4.1 rdfs:domain

W3C 콘소시엄(Consortium)[16]에서 제공하는 RDF Vocabulary Description Language 1.0: RDF Schema 문서에 나와있는 rdfs:domain 정의를 풀어 쓰면 다음과 같다. Predicate이 domain값을 가지면 해당 Predicate은 domain값에 나타난 클래스의 인스턴스만이 Subject로 나타날 수 있다는 것이다. 예를 들어 설명하면 그림 9과 같다.

```
<owl:ObjectProperty rdf:ID="nickName">
    <rdfs:domain rdf:resource="Person" />
<owl:ObjectProperty>
```

그림 9 rdfs:domain 예

RDF혹은 OWL 문서에서 위와 같이 객체형속성(owl:ObjectProperty)[15]이 nickName이고 rdfs:domain 값으로 Person으로 지정되어 있다면 객체형속성 nickName은 Person 클래스의 속성으로만 사용될 수 있다. 즉, Predicate의 Subject로서 Person 클래스의 인스턴

스들 만이 나올 수 있는 제약사항을 표시한다.

3.4.2 rdfs:range

rdfs:range값은 rdfs:domain값과 유사하나 트리플 관계에서 Predicate과 Subject에 관한 제한이 아니라 Predicate과 Object에 관한 제약 사항이다. 이 또한 그림 10의 예를 들어 설명하겠다.

```
<owl:ObjectProperty rdf:ID="postCode">
    <rdfs:range rdf:resource="postCodeFormat" />
<owl:ObjectProperty>
```

그림 10 rdfs:range 예

RDF혹은 OWL 문서에서 위와 같이 객체형속성이 postCode이고 rdfs:range값으로 postCodeFormat 클래스로 지정되어 있다면 postCode란 Predicate은 Object로서 postCodeFormat 클래스의 인스턴스들 만이 나올 수 있다는 것을 알려준다. 실제 우편번호 값은 000-000 형식을 사용하므로 postCode의 값은 이러한 형태의 포맷 형식을 갖는 구성원이 될 것이다.

4. 스키마 정보를 이용한 데이타 탐색범위 축소

본 장에서는 앞장에서 설명한 RDF-Schema의 rdfs:domian과 rdfs:range를 사용하게 된 동기와 이 정보들을 이용한 데이타베이스 스키마에 대해서 설명한다. 그 다음, 우리가 제시한 알고리즘을 소개하고 질의 처리하는 과정을 예제를 통해 살펴본다. 또한, 제시한 방법이 갖는 문제점에 대한 분석과 그 해결책을 제시한다.

4.1 동기(Motivation)

온톨로지를 구성하고 있는 수많은 트리플을 저장 관리하는데 있어 관계형 데이타베이스를 사용하는 기존의 방식들[10,11]은 트리플 데이타를 저장하는 하나의 Statement 테이블을 사용하였다. 즉, Jena2[10]의 경우 'jena_gntn_stmt' 테이블에 저장 하였고 Sesame[11]의 경우 'Triples' 테이블을 사용하였다. 두 경우 모두 트리플 데이타를 저장하는 이들 테이블을 중심으로 데이타베이스 스키마가 구성되어 있다. 트리플 데아타를 저장하는 테이블에 인덱스가 생성되어 있지만 각 컬럼에 저장되는 실제 데이타 값에, 많은 중복이 발생하고 그 값으로 긴 문자열이 들어가거나 객체에 대한 참조가 그 값으로 올 수 있기 때문에 인덱스 효과가 적다. Jena2[10]에서 'jena_gntn_stmt' 테이블이 새로운 모델이 추가될 때 동적으로 n값이 1씩 증가하여 새로운 Statement 테이블이 추가되기는 하나 질의의 종류가 어떠한 하나의 모델에만 국한되지는 않기 때문에 좋은 방법은 아니다[17]. 그러한 결과, 기존의 방식에서는 사용자 질의에 대한 결과를 만들어 내기 위해서 모든 트리플 데이타가 들어가 있는 'jena_gntn_stmt' 테이블에 대해서 질의에 맞는 트리플 데아타를 검색하고 질의처리 과정에서 셀프-조인(Self -Join) 연산이 빈번하게 발생할 수 밖에 없다. 그림 11은 트리플 데이타베이스에 온톨로지 데아타가 저장된

후 사용자 질의 처리를 보여주는 과정이다.

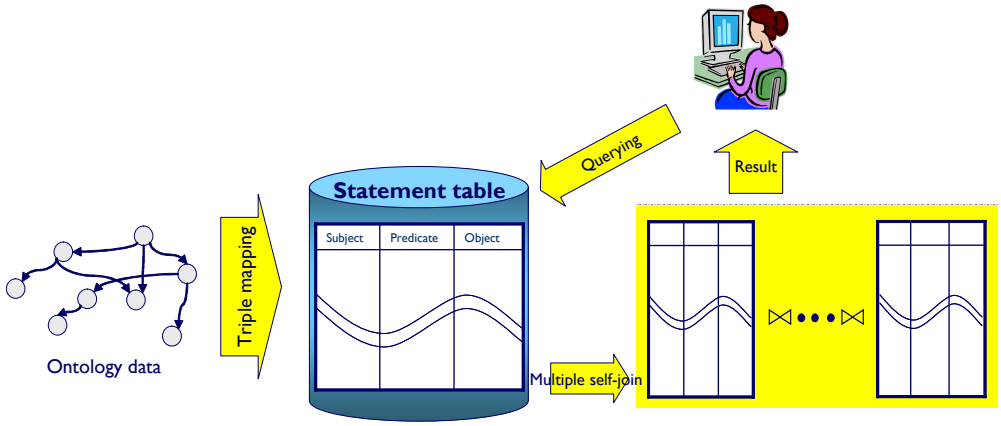

그림 11 온톨로지 데이타 저장과 질의처리 과정

이러한 문제점을 해결하기 위해 RDF-Schema에서 제공하는 rdfs:domain과 rdfs:range를 이용할 수 있도록 데이타베이스 스키마를 바꾸어 주면 조인 연산 수행 시, 조인에 참여하는 데이타에 대하여 보다 효율적으로 데이타를 검색할 수 있다.

4.2 재구성된 데이타베이스 스키마

RDF나 OWL에 나타난 rdfs:Class 및 owl:Class는 클래스간의 텍사노미(Taxonomy) 관계와 클래스 요소간의 관계를 나타내준다. 이를 따르는 트리플들은 클래스의 구성원(Member)들에 관한 사실과 구성원들 간의 관계를 기술하게 된다. 앞서 제시한 rdfs:domian과 rdfs:range를 이용할 수 있게 데이타베이스 스키마를 설계함으로써 관련된 트리플 데이타들을 각각의 클래스 Statement 테이블에 저장함으로써 질의 처리시 데이타의 검색 스페이스(Space) 축소로 인한 빠른 결과를 얻을 수 있다. 즉 RDF 혹은

OWL 문서상에 나와있는 rdfs:domain과 rdfs:range의 값에 나타나 있는 클래스에 대해서는 별도로 트리플을 저장하는 Statement 테이블들을 만드는 다수의(Multiple) Statement 테이블 구조를 갖도록 데이타베이스 스키마를 재구성하는 것이다. 아래 그림 12는 이 논문에서 제시하는 방법의 트리플 데아타의 저장방법과 질의 과정을 보여주는 시스템 플로우(System flow)이다.

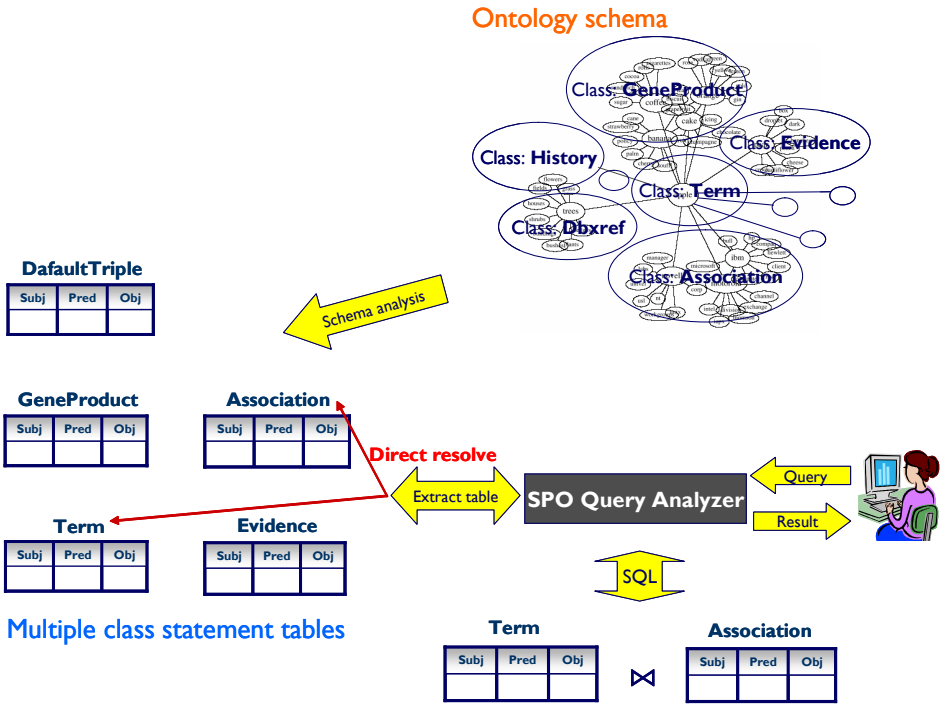

그림 12 새로 제시한 시스템 플로우

전처리(Preprocessing) 단계에서 스키마 분석을 통하여 두 객체형속성에 나타난 클래스에 대해서는 클래스 Statement테이블을 생성한다. 파싱과정을 통해 각각의 클래스 Statement 테이블에 트리플 데이타를 저장하며 분리 가능하지 않은 트리플 데이타는 'DefaultTriple' 테이블에 저장한다.

이후 사용자 질의는 SQP Query Analyzer에서 Predicate을 검사하고 이에 해당하는 클래스 Statement테이블을 선택하여 조인연산에 참여시키고 그 결과를 사용자에게 반환한다. 또한, Jena의 'jena_gntn_stmt' 테이블에 저장되어 있는 튜플 데이타들을 확인해 보면 네임스페이스(Namespace)[18]에 해당하는 값들이 중복되어 나타나는 것을 확인할 수 있다. 물리적인 데이타베이스 공간을 줄이기 위해 네임스페이스에 해당하는 부분을 별도로 테이블에 저장하였고 이를 다수의 Statement 테이블에서 참조하는 형식으로 데이타베이스 스키마를 재구성하여 Statement 테이블의 크기를 상당히 줄일 수 있다.

와인 온톨로지 문서의 일부인 그림 5에 나와 있는 것처럼 hasColor의 range값은 WineColor 클래스 갖는다. WineColor 클래스의 인스턴스는 'White', 'Rose', 'Red' 중 하나를 가질 수 있으므로 결국 hasColor 속성은 위의 세 값 범위 내에서만 값을 가진다. 이렇듯 Predicate hasColor는 문서내의 다른 연관 정보가 없는 한 기타 다른 클래스들과는 전혀 관계가 없다.

그러나 기존의 방법에서는 데이타를 저장할 때 이러한 정보를 전혀 이용하지 않고 모든 트리플 데이타를 하나의 Statement 테이블에 저장하였다. 그 결과로서 데이타 검색에 대한 불필요한 데이타 검색 사이즈 증가를 초래하였다. 우리는 앞서 설명한 rdfs:domain과 rdfs:range 정보를 트리플 데이타를 저장하는데 이용함으로써 효과적인 검색을 할 수 있다. 그림 13은 이러한 정보를 이용하기 위한 DomainRange 테이블로서 와인 온톨로지상에 나타나는 Predicate에 대해서 rdfs:domain 정보와 rdfs:range 정보를 보여주고 있다. 즉 Predicate 값으로 adjacentRegion이 나오면 그것의 Subject로는 Region 클래스의 인스턴스만이 나올 수 있으며 Object 값 또한 Regoin 클래스의 인스턴스만을 가질 수 있다. 또 다른 예로 Predicate 값으로 hasVintageYear가 나오면 Subject로는 Vintage 클래스의 인스턴스 만이 나올 수 있으며 Object값으로는 VintageYear 클래스의 인스턴스만이 나올 수 있음을 보여주고 있다.

Domain	Porp	Range
…	…	…
Region	adjacentRegion	Region
Vintage	hasVintageYear	VintageYear
Wine	hasWineDescriptor	WineDescriptor
Wine	hasColor	WineColor
…	…	…

그림 13 Wine 온톨로지에서의 rdfs:domain 과 rdfs:range

사용자 질의상에 Predicate값이 adjacentRgion이 나오면 우리는 단지 Region 클래스 Statement 테이블만 참조함으로써 해당 데이타를 찾을 수 있어 기존의 전체 트리플 데이타를 검색하는 것에 비해 관련되지 않은 상당히 많은 양의 트리플 데이타를 검색하지 않아도 된다. 그러나 여러 온톨로지가 한 문서상에 저장될 수 있고 그에 따라 여러 클래스에서 같은 Predicate 이름을 사용함으로써 이에 대한 중복이 발생할 수 있다. 이런 경우 발생하는 문제점과 그 해결 방법에 대해서 알아보자.

4.3 Edge reverse tracing

Gene 온톨로지에서 Term의 name이 'antioxidant activity(항산화 작용)'이고 이와 관련된 GeneProduct의 name이 'T14G11.8'인 Term을 찾고자 할 때 우리는 이 질의를 RDQL 혹은 RQL형식의 일련의 트리플 패턴인 $Pattern_n$(Subject, Predicat, Object)으로 그림 14와 같은 입력 인터페이스를 갖는다. 여기서 임의 변수에 '?' 붙은 값은 트리플 구조인 Subject, Predicate, Object에서 우리가 알고자 하는 값이나 온톨로지 상에서 그래프의 경로를 이동하는 패스(Path)를 나타낸다.

Pattern 1 (?X , name, 'antioxidant activity')
Pattern 2 (?X , association, ?Y)
Pattern 3 (?Y , gene_product, ?Z)
Pattern 4 (?Z , name, 'T14G11.18')

그림 14 트리플 패턴 질의

이런 트리플 패턴의 질의를 질의 트리(Query Tree)로 그림 15과 같이 나타낼 수 있다.

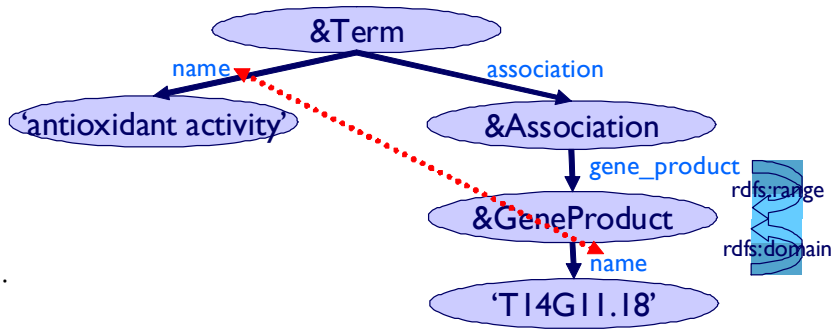

그림 15 질의 트리

여기서 &Term과 같이 &가 붙은 node들은 아직 결정 되지 않은 노드를 나타낸다. 그림 16에 나와 있는 ProDuplicate 테이블은 온톨로지 상에 나타난 모든 Predicate 값을 가지고 있다. 그 값이 1이면 해당 Predicate는 중복되었다는 것을 나타낸다. 그림 17의 DomainRange 테이블을 확인해 봄으로써 name이란 Predicate이 domain값으로 Term 클래스와 GeneProduct 클래스를 가질 수 있다는 것을 알 수 있다. 따라서 어느 클래스의 Statement table에서 데이타를 검색해야 하는지 결정할 수가 없거나 최악의 경우 이 두 클래스 Statement table을 검색해야 한다. 이러한 경우 우리는 질의 트리 상에서 방향성 있는 간선(Directed Edge)를 역방향 추적(Reverse tracing)을 함으로써 Predicate name의 domain 값을 결정할 수 있다. 즉, 그림 15에서 Predicate name의 domain은 Predicate

gene_product의 range 값과 같다는 것을 이용하는 것이다. 이 값은 그림 17의 DomainRange 테이블에서 확인 할 수 있으며 그 결과로 predicate name에 대해서 GeneProduct 클래스에서 검색해야 함을 결정할 수 있다.

prop	dupli
.....	.....
name	1
gene_product	0
.....	.....

그림 16 Gene온톨로지 PropDuplicate 테이블 #1

Domain	Pred	Range
.....	.....	.....
Term	name	null
Association	gene_prdouct	GeneProduct
GeneProduct	name	null
.....	.....	.....

그림 17 Gene온톨로지 DomainRange 테이블 #1

이 방법을 확장하여 Reverse tracing한 간선도 역시 중복된 Predicate value를 갖는 간선일 경우 스택(Stack)을 이용한 방법으로 문제를 해결할 수 있다.

4.4 Multiple edge reverse tracing

앞의 예제 질의에서와 같은 질의 트리에서 Predicate인 gene_product 역시 다른 클래스와 같이 사용되어 중복 된 경우를 생각해 보겠다. 즉 PropDuplicate 테이블은 그림 18과 같이 gene_product 값이 1로서 중복되

었음을 나타낸다. 또한 이때의 DomainRange 테이블은 그림 19와 같다.

prop	dupli
……	……
name	1
gene_product	1
association	0
……	……

그림 18 Gene온톨로지 PropDuplicate 테이블 #2

domain	pred	Range
……	……	……
Term	name	null
Association	gene_prdouct	GeneProduct
GeneProduct	name	null
Term	association	Association
……	……	……

그림 19 Gene온톨로지 DomainRange 테이블 #2

다수의 간선(Multiple Edge)에 대해 Predicate이 중복된 문제는 그림 20의 스택을 사용함으로써 해결할 수 있다. 즉 Predicate name이 중복되어 있으므로 이에 해당하는 domain과 Predicate 값의 쌍(Pair)을 스택에 넣는다(Push). 이 값을 알기 위해 상위 간선에 해당하는 Predicate gene_product을 검사하게 되는데 이 값 역시 중복되어 있으므로 domain 값과 predicate쌍을 역시 스택에 넣는다. 상위 간선인 association은 중복되어 있지 않으므로 이 값의 range 값인 Association을 반환한다. 이 값이 &y의 값이므로 그림 19에서 해당 쌍의 range값은 GeneProduct인 것을 알 수 있다. 결국 Predicate name이 GeneProduct 클래스에 속해 있음을 알 수 있다.

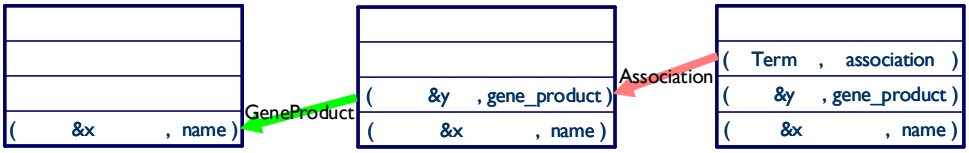

그림 20 스택을 이용한 해결 방법

질의 트리 상에서 상위 간선에 해당하는 값이 중복되어 Reverse tracing을 할 수 없는 경우 Predicate을 가질 수 있는 클래스 Statement 테이블의 리스트를 유지함으로써 모든 테이블을 검색해야 하는 최악의 경우를 피할 수 있다.

5. 성능 평가

본 장에서는 질의를 수행하여 Jena2에서 기존에 사용하던 하나의 Statement 테이블을 사용하는 방법과 논문에서 제시하는 rdfs:domain과 rdfs:range에 나타난 클래스에 대해서 다수의 statement 테이블을 사용했을 때의 처리 시간에 대한 평가와 데아타베이스 사이즈에 대한 측정으로 성능 평가를 보인다.

5.1 실험 환경

본 실험을 위한 장비로는 CPU Pentium4 1.6GHz 와 RAM 1GB를 사용하였다. 데이타 베이스 시스템으로는 MySQL 4.0을 사용 하였으며 데이타는 Open Biological Ontologies[12]에서 제공하는 Gene 온톨로지 데아타인 termDB를 사용하였고 실험 시스템은 Java로 구현 하였다. 실험 질의 구성은 그림 21과 같다.

Q1	Find term whose accession is 'GO:0016209' and related evidence code value is 'ISS'
Q2	Find Q1 term and that is related with database symbol with 'PMID'
Q3	Find parent term whose child term's definition is containing 'amino acid'
Q4	Find term whose name is 'antioxidant' and related with GeneProduct whose name is 'T14G11.18'

그림 21 실험 질의 구성

Q1질의의 경우 온톨로지 상에서 Term 클래스의 Predicate이 accession이고 그 값으로 'GO:0016209'를 가지며 관련된 Evidence 클래스의 Predicate인 evidence_code 값으로 'ISS'를 갖는 노드를 찾으라는 질의이다. 이 질의는 Association 클래스를 거쳐 2번의 조인 연산이 발생한다.

Q2질의는 Q1과 같은 Term 찾고 DBxref 클래스의 database_symbol Predicate의 값이 'PMID'를 갖는 Term을 찾는 질의이다. Q3은 Q1과 Q2와는 다르게 온톨로지 상에서 상속관계인 is_a관계를 찾는 질의로서 상위 Term의 Predicate인 definition이 그 값으로서 'amino acid'를 포함하는 개념을 갖는 Term 클래스의 인스턴스를 찾는 질의이다. Q4는 Predicate의 중복으로 인해 스택 연산(Stack Operation) 수행을 통해 질의 트리의 상위 Predicate을 검사해야 하는 경우의 질의이다.

5.2 실험 결과

실험은 크게 두 가지 항목을 비교하였다. 첫 번째 실험은 문서상에 나타나 있는 스키마 정보인 rdfs:domain과 rdfs:range를 사용하여 질의를 수행한 것과 해당 정보를 이용하지 않고 질의를 했을 때의 Jena2 영구 모델(Persistent Model)과 그 성능을 비교하였다. 우리는 데이타베이스의 사이즈를 줄일 수 있도록 allNameSpace라는 테이블에 네임스페이스를 저장관리하고 참조하는 데아타베이스 스키마를 제안하였다. 이것을 바탕으로 데아타베이스 사이즈를 비교 평가하였다.

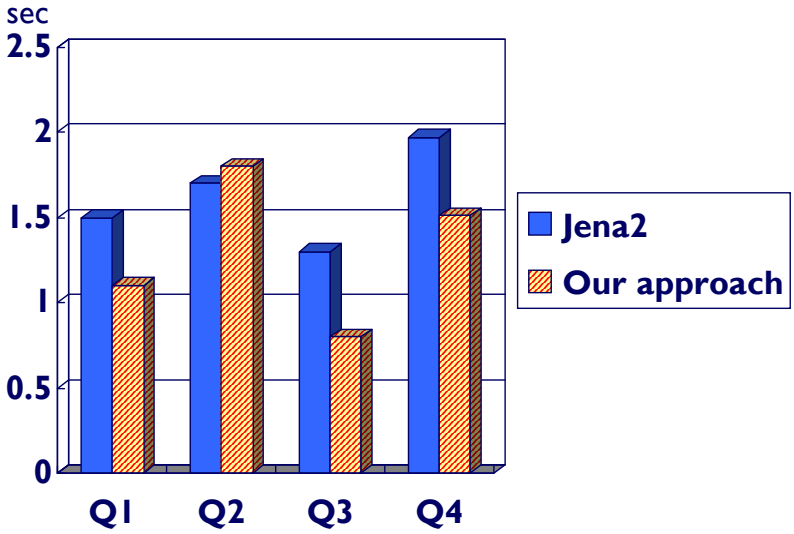

그림 22 질의 처리 시간

그림 22는 실험 결과를 보여준다. 실험 결과에서 우리가 제시한 방법에서 좀 더 나은 성능을 보여준다. Q1질의에서는 본 논문에서 제시한 RDF-Schema를 이용하는 경우이다. 질의상에 Predicate을 검사하고 해당 Predicate이 클래스별 Statement 테이블에 속해 있는 Predicate값이면 해당 테이블만 검색함으로서 전체 트리플 데이타를 검색할 때 보다 좋은 성능을 낸다. 결과적으로 Q1은 세 개의 테이블인 Term 클래스 Statement 테이블과 Association 클래스 Statement 테이블 그리고 Evidence 클래스 Statement 테이블만 조인에 참여하면 된다. 그렇기 때문에 Gene 온톨로지에 속하는 전체 트리플 데아타에 대해서 세 번 조인하는 것보다 좀 더 나은 성능을 발휘한다. 질의 Q2의 경우 Predicate인 'database_symbol'은 rdfs:domain과 rdfs:range에 해당하는 정보가 나타나 있지 않은 경우를 실험 한 경우이다. 이 경우 논문에서 제시한 방법이 약간 떨어지는 성능이

나왔는데 이는 분할되어 있는 테이블들을 검색해야 하는 비용이 추가되었기 때문이다. 그러나 시멘틱 웹이 잘 구축 된다면 rdfs:domain과 rdfs;range 정보는 충실히 제공되어 진다고 가정할 수 있을 것이다. 질의 Q3 역시 질의 1과 같이 좋은 성능이 나왔다. 질의 Q4의 경우 그림 10에서의 예제 트리와 같은 질의를 하였다. 즉 중복된 Predicate을 가지는 경우의 질의 수행 시간 결과를 나타낸다. 이 경우에도 역시 우리가 제시한 방법이 기존의 방법보다 조금 더 우수함을 보여주고 있다. 이러한 이유는 온톨로지상에서 Predicate이 차지하는 부분은 극히 작기 때문이다. 왜냐하면, 트리플 중에서 Subject와 Object의 값은 클래스의 인스턴스에 따라 다른 값을 가질 수 있지만 Predicate은 한번 온톨로지 스키마가 정해지면 변하지 않는다. 그렇기 때문에 Predicate에 대해서는 캐쉬(Cache)를 사용함으로써 불필요한 디스크 I/O를 줄일 수 있고 해당 Predicate의 클래스 Statement 테이블을 찾을 수 있는 것이다. 이 경우 역시 전체 트리플 데이타가 아닌 일부 클래스의 트리플 데이타 중에서 검색을 하기 때문에 기존의 방법보다 좋은 성능을 보여준다.

그림 23은 데이타베이스 사이즈에 대한 평가이다. 본 논문에서 제시한 데이타베이스 스키마를 사용함으로써, 전체 데이타베이스의 사이즈가 20%정도 감소 한 것을 알 수 있다. 역시 데이타베이스 용량 면에서도 더 나은 성능을 보여준다.

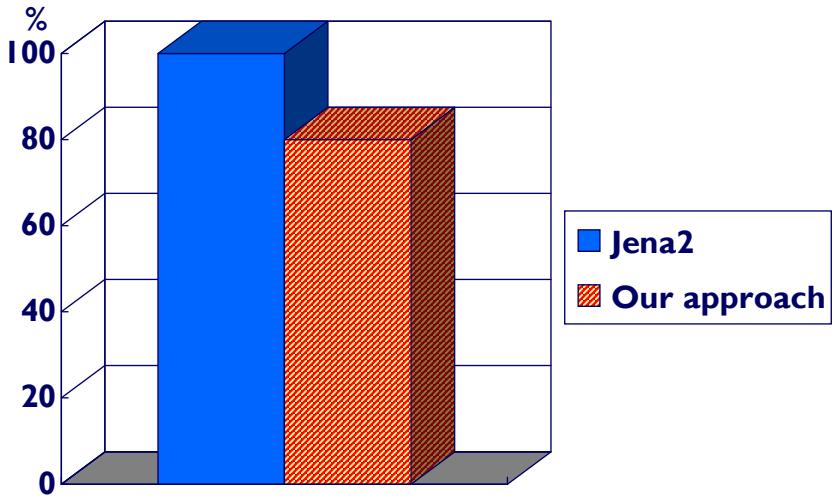

그림 23 데이타베이스 용량 변화

6. 결론 및 향후 연구 방향

우리는 본 논문에서 OWL 혹은 RDF 문서상의 스키마 데이타인 rdfs:domain과 rdfs:range의 정보를 이용하여 데이타베이스 스키마를 설계하였고 그에 따라 트리플 데이타를 저장하여 관련된 데이타는 한 테이블에 클러스터링 되는 효과를 얻을 수 있었다. 그렇게 함으로써 관련되지 않은 데이타에 대한 불필요한 검색 연산을 피할 수 있었고 찾고자 하는 데이타의 신속한 결과를 얻을 수 있었다. 이런 기법을 제공하기 위해서는 다수의 Statement를 생성하는 전처리(Preprocessing) 과정과 rdfs:domain 값과 rdfs:range 값을 유지하는 DomainRange 테이블그리고 Predicate이 중복이 되어있는지를 나타내는 PropDuplicate 테이블 2개만을 유지하면 된다. 또한 이 테이블을 유지하는 비용은 앞서 말했듯이 온톨로지 데이타에 비해 아주 작은 비중을 차지하므로 그 자료구조를 유지하는 비용은 크지 않고 검색을 수행할 경우에도 효율적으로 이용할 수 있다.

향후 연구에서는 RDF 혹은 OWL문서를 생성할 때, 반자동으로 의미정보인 rdfs:domain과 rdfs:range 데이타가 작성될 수 있는 방법에 대해 연구를 진행할 계획이다.

참고 문헌

[1] 오삼균, Web Ontology Language와 그 활용에 관한 고찰, 데아타베이스연구회지, vol. 18 no. 03.pp. 0063 - 0080 2002. 9

[2] Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001

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

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

[5] D. Connly, F. Harmelen "DAML+OIL (March 2001) Reference Description", W3C Note 18 December 2001

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

[7] Debora L. McGuinness et al., "DAML-ONT: An Ontology Language for the Semantic Web", http://www.daml.org/2000/10/daml-ont.html.

[8] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Debora, "OIL: An Ontology Infra-structure for the semantic Web", IEEE Intelligent System, vol.16, no.2, March/April, 2001, pp.38-45.

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

[10] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds, "Efficient RDF Storage and Retrieval in Jena2", Proceedings of SWDB'03.

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

[12] Jena-A Semantic Web Framework for Java, http://jena.sourceforge.net/.

[13] WineOntology.
http://www.w3.org/TR/2002/WD-owl-guide-20021104/wine.owl

[14] Gene Ontology (GO): http://www.geneontology.org

[15] Dean, M., D. Connolly, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P.Patel-Schneider, and L. Stein. 2002. "Web Ontology Language (OWL) Reference Version 1.0." http://www.w3.org/TR/owlref/

[16] W3 Consortium (W3C): http://www.w3c.org

[17] Database Schema Design and Analysis for the efficient OWL Semantic information processing Kyung-Hyen Tak, Hag-Soo Kim, Hyun-Seok Cha, Jin-Hyun son Dept. of Computer Science & Engineering, Hanyang University KDBC 2004

[18] Bray, T., D. Hollander, and A. Layman. 1999. "Namespaces in XML." <http://www.w3.org/TR/REC-xml-names/>

부 록

A. UML Diagram

Use Case Diagram 1

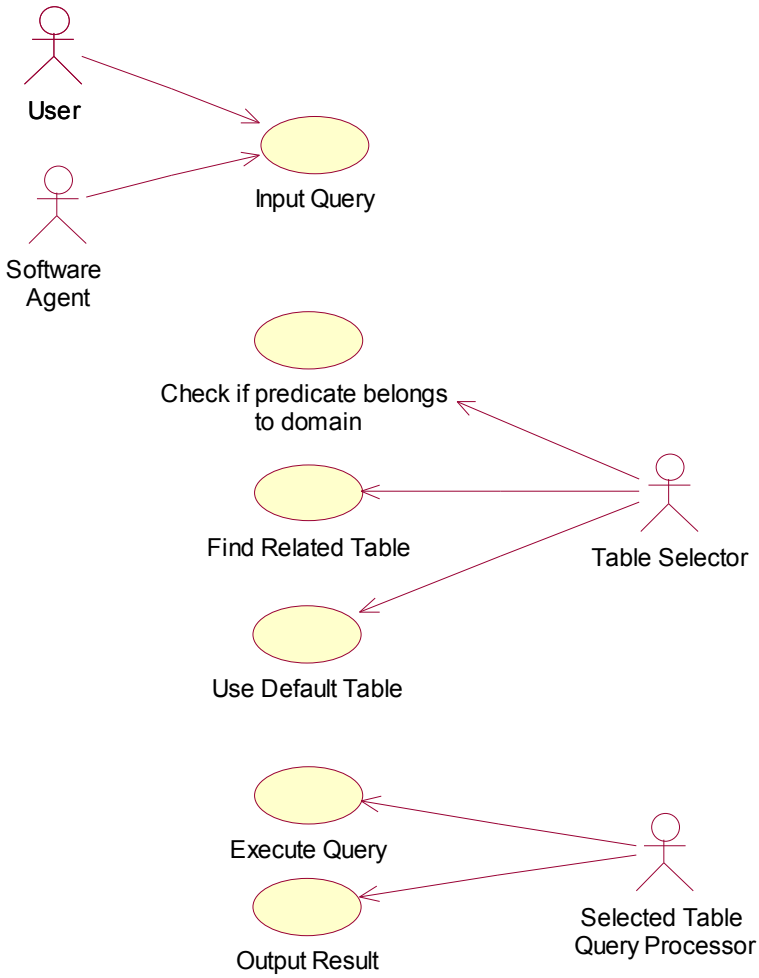

그림 24 사용자 및 프로세스간의 유스 케이스 다이어그램

Use Case Diagram 2

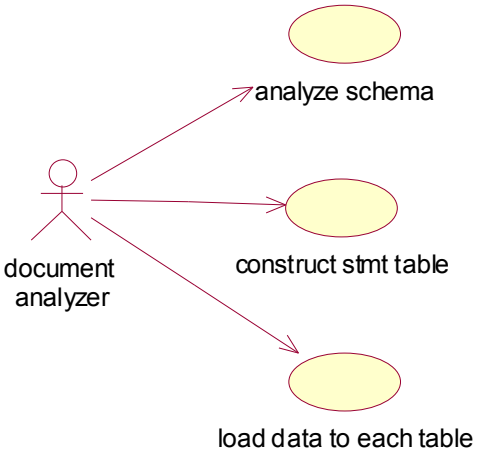

그림 25 Multiple Statement 테이블 데이타베이스 생성 유스 케이스 다이어그램

Class Diagram 1

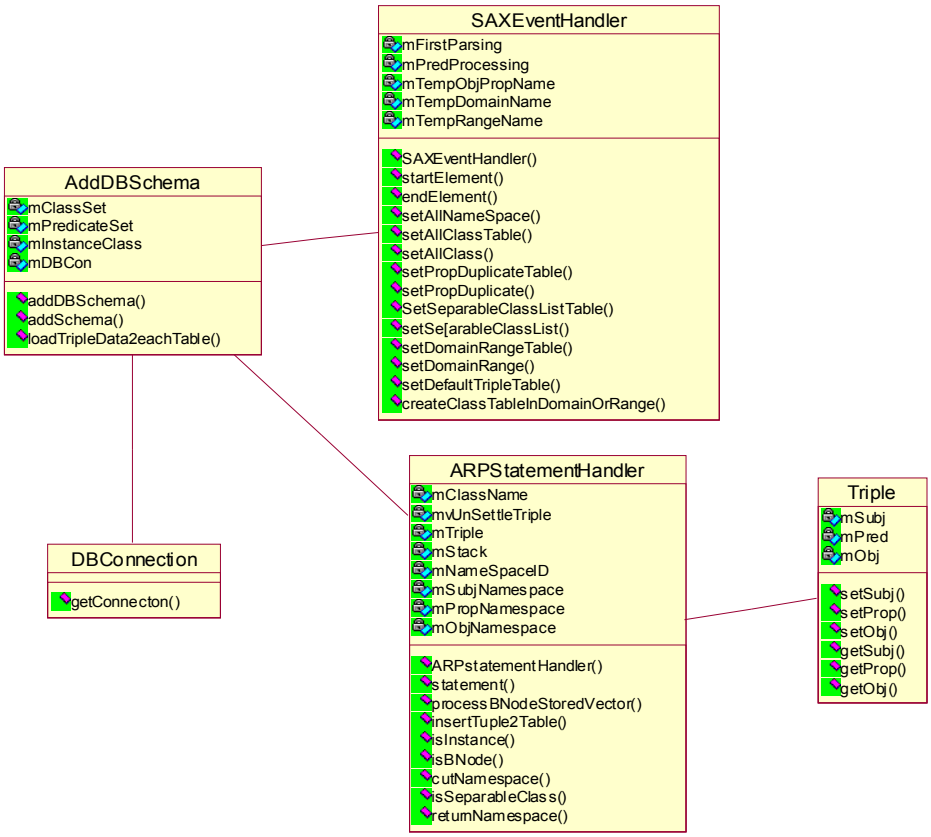

그림 26 Multiple Statement 테이블 데이타베이스 생성 클래스 다이어그램

Class Diagram 2

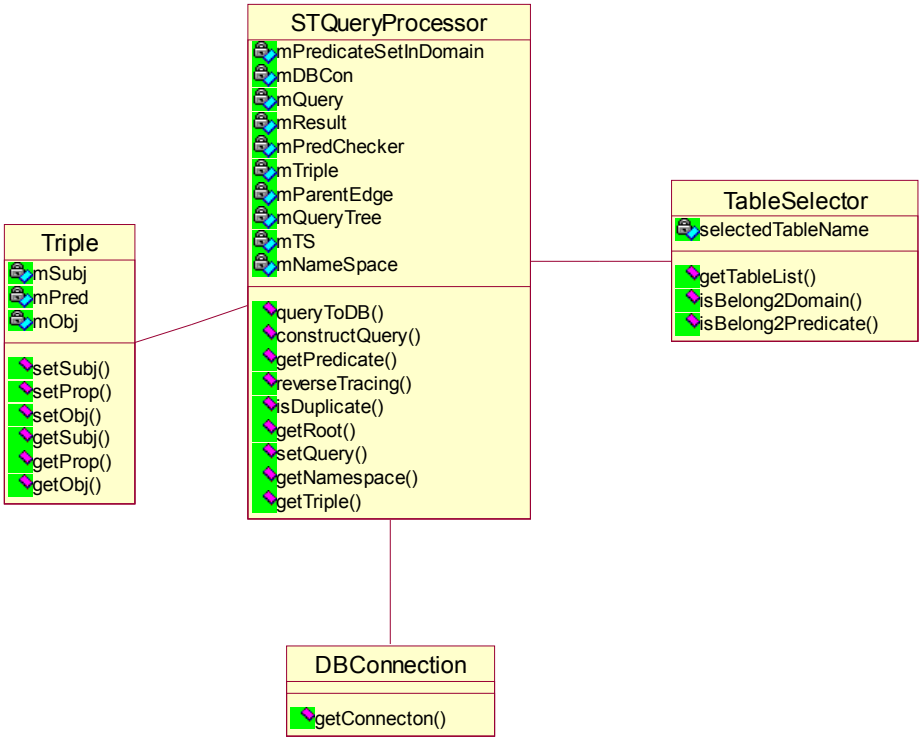

그림 27 질의 실행 클래스 다이어그램

Sequence Diagram 1

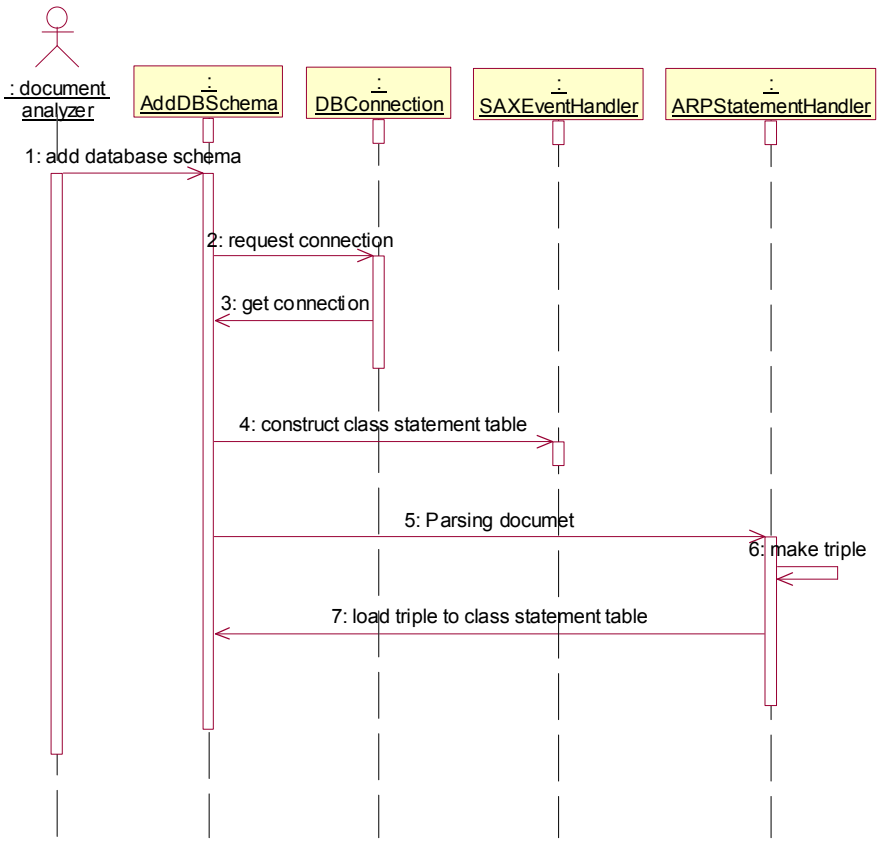

그림 28 Multiple Statement Table 생성 시퀀스 다이어그램

Sequence Diagram 2

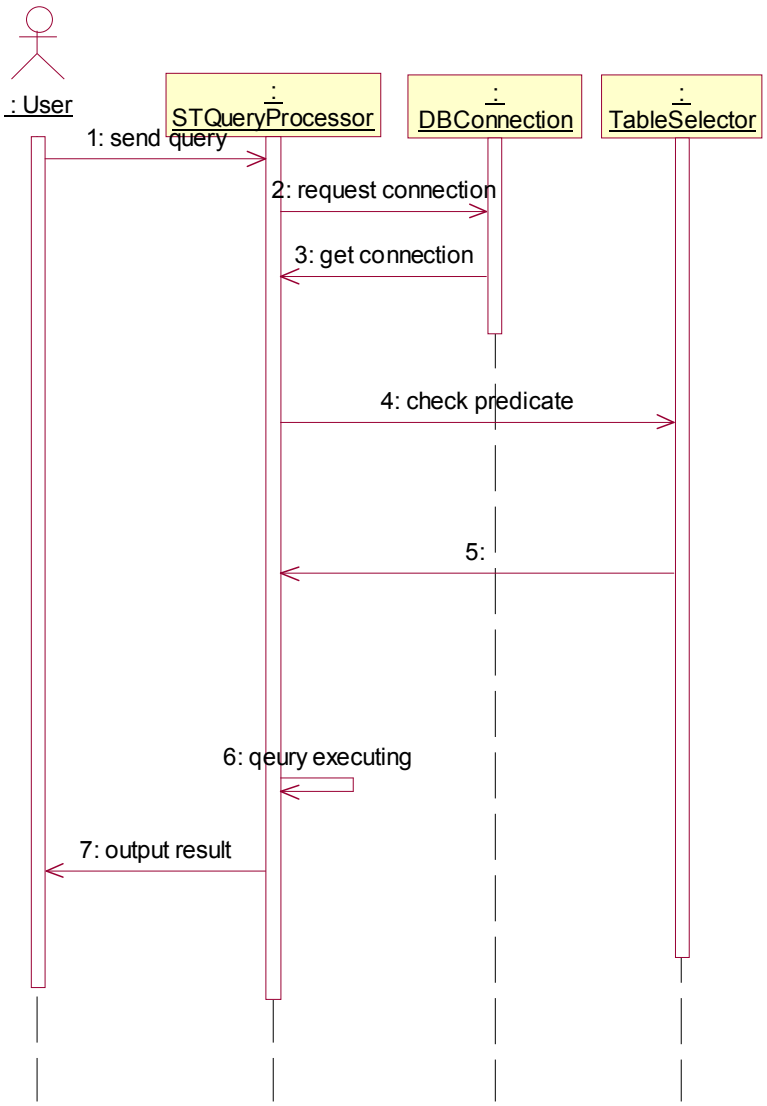

그림 29 Multiple Statement 질의 처리기 시퀀스 다이어그램

B. Jena2 Architecture Component

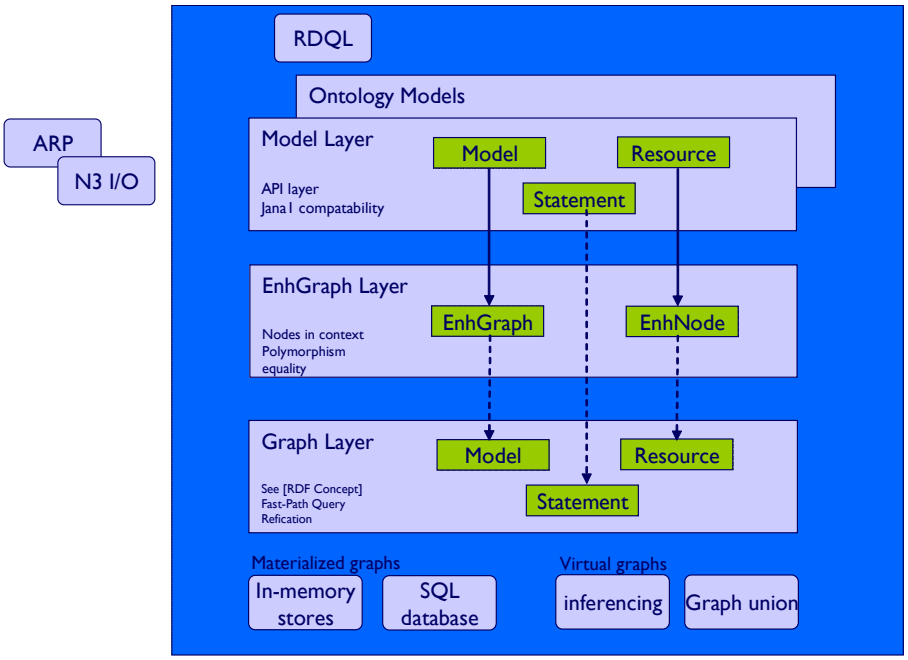

그림 30 Jena2 Architecture

ARP	RDF/XML로 작성된 문서에 대한 parser로서 RDF Core Working Group의 최종 결정에 완전하게 따르는 것을 목적으로 하고 있다. ARP가 문서를 파싱하면 그 결과는 node-edge-node의 트리플 집합(set)을 형성하며 이들 각각은 subject, predicate, object를 의미한다.
Persistence	Storage로는 MySQL, Oracle, PostgreSQL등의 데이타베이스를 지원한다.
Resoning subsystem	추론엔진은 OWL Lite 를 제공한다. Jena2 내부에는 Transitive 추론엔진, RDFS 추론엔진을 포함하며 외부의 추론엔진을 plug-in 형식으로 Jena에서 사용할 수 있다.
Ontology Subsystem	Jena Ontology API는 RDF기반 온톨로지 작업을 지원하며 OWL, DAML+OIL 그리고 RDF-Schema를 지원한다.

C. Sesame Architecture

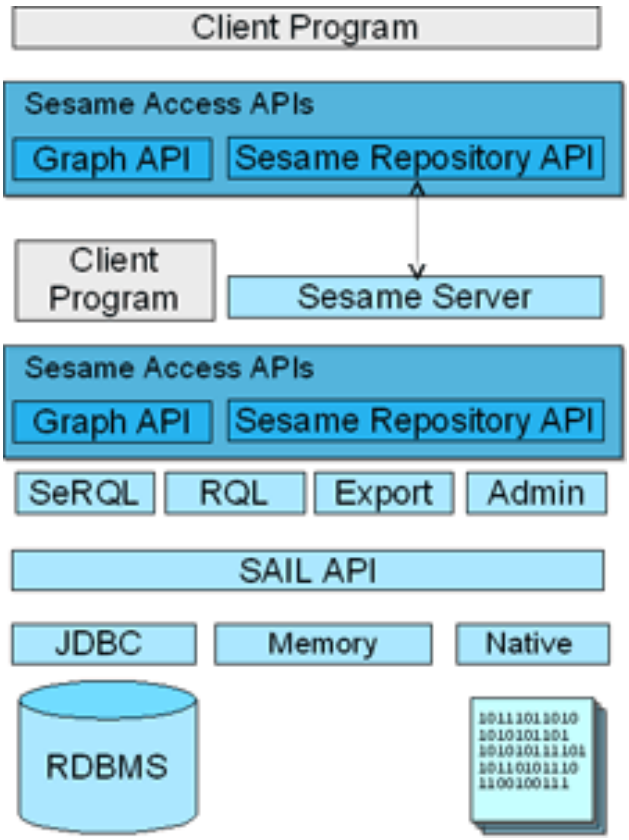

그림 31 Sesame Architecture

Sesame의 아키텍처는 기본적으로 확장성과 적응성을 고려하여 디자인 되었다. 때문에 추가적인 모듈(Module) 또는 프로토콜 핸들러(Protocol Handler)를 추가하면 다른 종류의 저장소를 사용할 수 있다. Sesame은 클라이언트와의 통신을 위해 HTTP, RMI(Remote Method Invodcation)과 SOAP(Simple Object Access Protocol)등 다양한 프로토콜을 제공한다. 이러한 프로토콜을 위해 각각의 핸들러 모듈이 존재한다. RQL 모듈은

Sesame에서 제공하는 Query 모듈로서 RQL 파서를 거쳐 Query 모델을 생성하고 Query Optimizer가 Optimized Query 모델을 생성하고 앞서 말한 SAIL API를 통해 저장소에 질의를 하게 된다. Admin 모듈은 RDF 데이타와 Schema 정보를 저장소(Repository)에 입력가능하게 한다. Export 모듈은 간단한 모듈로서 저장소에 저장된 데이타를 XML/RDF 형태로 내보낼 수(Export) 있게 한다.

D. Wine Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
     <!ENTITY vin  "http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#" >
     <!ENTITY food "http://www.w3.org/TR/2003/CR-owl-guide-20030818/food#" >
     <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
     <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
   ]>

<rdf:RDF
  xmlns      = "http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#"
  xmlns:vin = "http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#"
  xml:base  = "http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#"
  xmlns:food= "http://www.w3.org/TR/2003/CR-owl-guide-20030818/food#"
  xmlns:owl = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">

  <owl:Ontology rdf:about="">
    <rdfs:comment>An example OWL ontology</rdfs:comment>
    <owl:priorVersion>
      <owl:Ontology
rdf:about="http://www.w3.org/TR/2003/WD-owl-guide-20030331/wine"/>
    </owl:priorVersion>
    <owl:imports
rdf:resource="http://www.w3.org/TR/2003/CR-owl-guide-20030818/food"/>
    <rdfs:comment>Derived from the DAML Wine ontology at
      http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml
      Substantially changed, in particular the Region based relations.
    </rdfs:comment>
    <rdfs:label>Wine Ontology</rdfs:label>
  </owl:Ontology>

  <owl:Class rdf:ID="Wine">
    <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasMaker" />
          <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
```

Wine Ontology(계속)

```
 <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasMaker" />
          <owl:allValuesFrom rdf:resource="#Winery" />
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#madeFromGrape" />
          <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasSugar" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasFlavor" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasBody" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasColor" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#locatedIn"/>
        <owl:someValuesFrom rdf:resource="&vin;Region"/>
```

Wine Ontology(계속)

```
    </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="en">wine</rdfs:label>
    <rdfs:label xml:lang="fr">vin</rdfs:label>
  </owl:Class>
<owl:Class rdf:ID="Vintage">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasVintageYear"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="WineGrape">
    <rdfs:subClassOf rdf:resource="&food;Grape" />
  </owl:Class>

  <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>

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

그림 32 Wine Ontology Schema 일부

Wine Ontology Instance

```
<WineBody rdf:ID="Full" />

<WineBody rdf:ID="Medium" />

<WineBody rdf:ID="Light" />

<WineColor rdf:ID="Red" />

<WineColor rdf:ID="Rose" />

<WineColor rdf:ID="White" />

<WineFlavor rdf:ID="Strong" />

<WineFlavor rdf:ID="Moderate" />

<WineFlavor rdf:ID="Delicate" />

<WineSugar rdf:ID="Dry" />

<WineSugar rdf:ID="OffDry">
  <owl:differentFrom rdf:resource="#Dry"/>
  <owl:differentFrom rdf:resource="#Sweet"/>
</WineSugar>

<WineSugar rdf:ID="Sweet">
  <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>

<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red" />
    <vin:WineColor rdf:about="#White" />
    <vin:WineColor rdf:about="#Rose" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

Wine Ontology Instance(계속)

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <vin:WineBody rdf:about="#Light" />
      <vin:WineBody rdf:about="#Medium" />
      <vin:WineBody rdf:about="#Full" />
    </owl:distinctMembers>
  </owl:AllDifferent>

  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <vin:WineFlavor rdf:about="#Delicate" />
      <vin:WineFlavor rdf:about="#Moderate" />
      <vin:WineFlavor rdf:about="#Strong" />
    </owl:distinctMembers>
  </owl:AllDifferent>

  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <vin:WineSugar rdf:about="#Sweet" />
      <vin:WineSugar rdf:about="#OffDry" />
      <vin:WineSugar rdf:about="#Dry" />
    </owl:distinctMembers>
  </owl:AllDifferent>

  <Region rdf:ID="AlsaceRegion">
    <locatedIn rdf:resource="#FrenchRegion" />
  </Region>

  <Region rdf:ID="AnjouRegion">
    <locatedIn rdf:resource="#LoireRegion" />
  </Region>

  <Region rdf:ID="ArroyoGrandeRegion">
    <locatedIn rdf:resource="#CaliforniaRegion" />
  </Region>

  <Winery rdf:ID="Beringer" />

  <Winery rdf:ID="Bancroft" />
```

그림 33 Wine Ontology Instance 일부

E. Gene Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:oiled="http://img.cs.man.ac.uk/oil/oiled#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#">
    <daml:Ontology rdf:about="">
        <dc:title>&amp;quot;Gene Ontology&amp;quot;</dc:title>
        <dc:date>February 2003</dc:date>
        <dc:creator>Gene Ontology Consortium</dc:creator>
        <dc:description>Copyright 1999-2002 Gene Ontology Consortium. Permission to
use the information contained in this database was given by the researchers/institutes who
contributed or published the information. Users of the database are solely responsible for
compliance with any copyright restrictions, including those applying to the author
abstracts. Documents from this server are provided "AS-IS" without any warranty,
expressed or implied.</dc:description>
        <dc:subject>Gene Products</dc:subject>
        <daml:versionInfo>February 2003</daml:versionInfo>
    </daml:Ontology>
    <daml:Class rdf:about="http://www.geneontology.org/go#glycolytic+fermentation">
        <rdfs:label>glycolytic+fermentation</rdfs:label>
        <rdfs:comment>GO:0019660 </rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:08Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class rdf:about="http://www.geneontology.org/go#fermentation"/>
        </rdfs:subClassOf>
    </daml:Class>
    <daml:Class rdf:about="http://www.geneontology.org/go#kappa-opioid+receptor">
        <rdfs:label>kappa-opioid+receptor</rdfs:label>
        <rdfs:comment>GO:0004987 </rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:29Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class rdf:about="http://www.geneontology.org/go#opioid+receptor"/>
        </rdfs:subClassOf>
    </daml:Class>
```

Gene Ontology(계속)

```
    <daml:Class rdf:about="http://www.geneontology.org/go#post-pollination">
        <rdfs:label>post-pollination</rdfs:label>
        <rdfs:comment>GO:0009856 Post-pollination involves interactions
            between the female tissues (stigma, style and ovary) and the
            male gametophyte or the pollen tube cell, which contains the
            sperm cells. definition_</rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:31Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class
rdf:about="http://www.geneontology.org/go#physiological+ processes"/>
        </rdfs:subClassOf>
    </daml:Class>
    <daml:Class
rdf:about="http://www.geneontology.org/go#debranching+ enzyme">
        <rdfs:label>debranching+ enzyme</rdfs:label>
        <rdfs:comment>GO:0005957 OBSOLETE (was not defined before being
            made obsolete). definition_</rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:17Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class
rdf:about="http://www.geneontology.org/go#obsolete+ %2B"/>
        </rdfs:subClassOf>
    </daml:Class>
    <daml:Class
rdf:about="http://www.geneontology.org/go#UDP-N-acetylglucosamine+ 1-carboxy
vinyltransferase">

<rdfs:label>UDP-N-acetylglucosamine+ 1-carboxyvinyltransferase</rdfs:label>
        <rdfs:comment>GO:0008760 Catalysis of the reaction:
            phosphoenolpyruvate + UDP-N-acetyl-D-glucosamine = phosphate
            + UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine.
definition_</rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:05Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class
rdf:about="http://www.geneontology.org/go#transferase%2C+ transferring+ alkyl+ o
r+ aryl+ groups%2C+ other+ than+ methyl+ groups"/>
        </rdfs:subClassOf>
    </daml:Class>
```

Gene Ontology(계속)

```
<daml:Class rdf:about="http://www.geneontology.org/go#iprodione+ amidohydrolase">
        <rdfs:label>iprodione+ amidohydrolase</rdfs:label>
        <rdfs:comment>GO:0018748 </rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:32Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class
rdf:about="http://www.geneontology.org/go#hydrolase%2C+ acting+ on+ carbon-nitrogen+ %28
but+ not+ peptide%29+ bonds%2C+ in+ linear+ amides"/>
        </rdfs:subClassOf>
    </daml:Class>
    <daml:Class
rdf:about="http://www.geneontology.org/go#aryl+ hydrocarbon+ receptor+ nuclear+ translocat
or">
        <rdfs:label>aryl+ hydrocarbon+ receptor+ nuclear+ translocator</rdfs:label>
        <rdfs:comment>GO:0005061 </rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:27Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class
rdf:about="http://www.geneontology.org/go#receptor+ signaling+ protein"/>
        </rdfs:subClassOf>
    </daml:Class>
    <daml:Class rdf:about="http://www.geneontology.org/go#response+ to+ cold">
        <rdfs:label>response+ to+ cold</rdfs:label>
        <rdfs:comment>GO:0009409 A change in state or activity of an
            organism (in terms of movement, secretion, gene expression,
            enzyme production, etc.) in response to temperatures below
            the optimal temperature for that organism. definition_</rdfs:comment>
        <oiled:creationDate>2003-03-13T11:55:32Z</oiled:creationDate>
        <oiled:creator>chris</oiled:creator>
        <rdfs:subClassOf>
            <daml:Class rdf:about="http://www.geneontology.org/go#response+ to+ stress"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <daml:Class
rdf:about="http://www.geneontology.org/go#response+ to+ temperature"/>
        </rdfs:subClassOf>
    </daml:Class>
```

그림 34 Gene Ontology 일부

Three categories of GO

GO에 대한 주제를 논의할 때, 가장 문제가 되는 것은 “Function”이 갖는 의미가 모호하다는 것이다. 그렇기 때문에 Function, Role, Process 라는 용어 사이에 구분을 명확히 할 필요가 있었다.

Example)

function -	“transcription factor”, “transporter”, “enzyme”
Role -	“transcription factor regulation HOXA1”, “transporting sucrose”
Process -	“catabolism of carbohydrates”

그러나 관련 토론이 계속되어짐에 따라, Role 이란 용어는 필요하지 않다고 결론을 내렸다. 이런 Function에 대한 모호성을 확실히 하기 위해, GO에서는 세 가지의 독립적인 온톨로지 제시하였다. Molecular Function, Biological Process, 그리고 Cellular Component 가 그것이다. 뒤에서 자세히 설명하기로 하고, 이 세가지를 설명하기에 앞서서 우선 Gene Product에 대한 정의를 명확하게 알아보자.

Gene product는 물리적인 것(Physical Thing)에 국한한다. Protein 또는 RNA 등일 것이다. 예를 들면 alpha-globin , small ribosomal RNA등이다. 복합체(Complex)를 이루어 작용하는 Gene Products를 Gene Product Groups이라 한다. Gene Product Group은 Gene Products 뿐만 아니라, Heme과 같은 작은 Molecules도 포함한다.

Molecular Function은 Gene Product가 잠재적으로 갖는 능력을 뜻한다. 즉 실제 어디서 언제 쓰이는지에 대한 구체화 없이, 무엇을 할 수 있는지에 대한 내용을 갖게 된다. Broad Functional Term을 예로 들면

“enzyme”, “transporter”, “ligand” 등이 있을 수 있고, Narrower Functional Terms을 예로 들면, “adenylate cyclase” 또는 “Toll receptor ligand” 등이 있을 수 있다.

Gene Product와 그것의 Molecular Function는 같은 단어가 쓰이는 경우가 많기 때문에 의미상 혼란이 올 경우가 있다. 예를 들면 “alcohol dehydrogenase”를 E-tube에 적어놓았다고 하면, 이것은 E-tube에 무엇을 넣어 놓았는지에 대한 표기이지만, 그 내용물이 갖는 기능을 기술해 놓은 것도 된다. 그러나 엄밀히 말하면, “Product”와 “Molecular Function”은 다른 점이 있다. “alcohol dehydrogenase” Function을 갖는 Gene은 많다. 그 중에는 “alcohol dehydrogenase”라는 이름을 가진 것도 있지만, 아닌 것이 많다. 또한 “alcohol dehydrogenase”의 기능을 가지면서 “acetaldehyde dismutase”기능을 갖는 것도 있다. 그림 35는 Molecular Function에 대한 것이다.

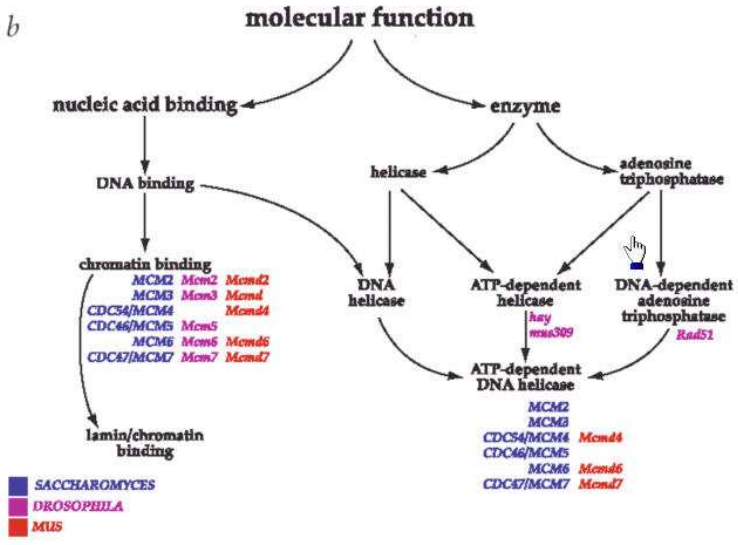

그림 35 Molecular Function

Biological Process : 생물학 목적(Biological Objective)을 뜻한다. 하나 또는 그 이상의 Molecular Functions이 순서대로 조합되어 이루어 내는 것이다. Broad Biological Process Terms으로 예를 들면, “cell growth and maintenance” 또는 “signal transduction”이 있고, Specific Terms을 예로 들면 “pyrimidine metabolism”, “cAMP biosynthesis” 등이 있다.

Biological Process는 molecular function과 의미상으로 구분하는 것은 어려울 수 있다. GO에서는 1개 이상의 구별되는 step이 있어야만 process로 인정하도록 일반적인 규칙을 정하고 있다.

그림 36은 DNA metabolism을 표현하는 biological process의 일부를 표시한 예시이다.

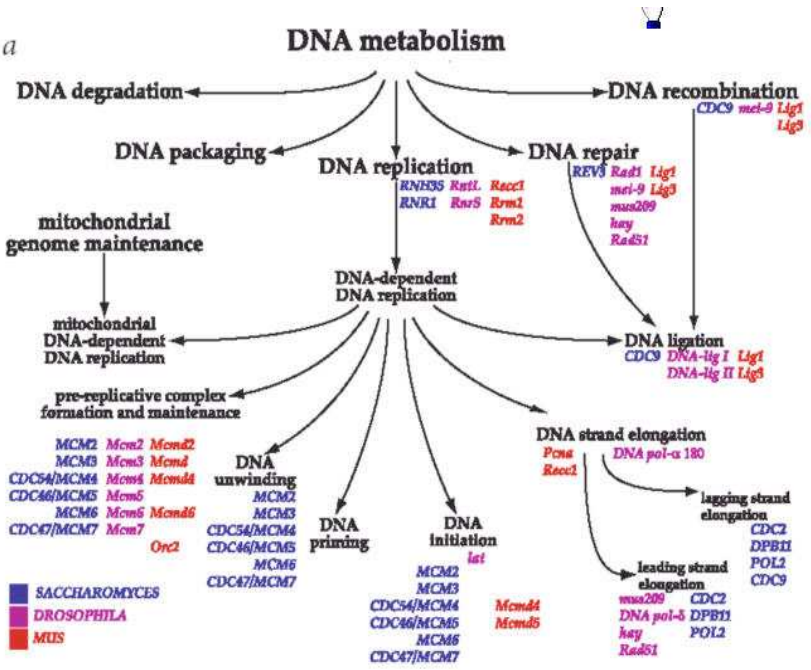

그림 36 DNA Metabolism

Cellular Component : Cell의 구성원(Component)를 말하는 것으로써 anatomical structure에 대한 명명이 주가 될 것이다. “rough endoplasmic reticulum” 또는 “nucleus” 등이 그것이고, gene product group의 예를 들면 “ribosome”,”proteasome” 등등이 될 것이다.

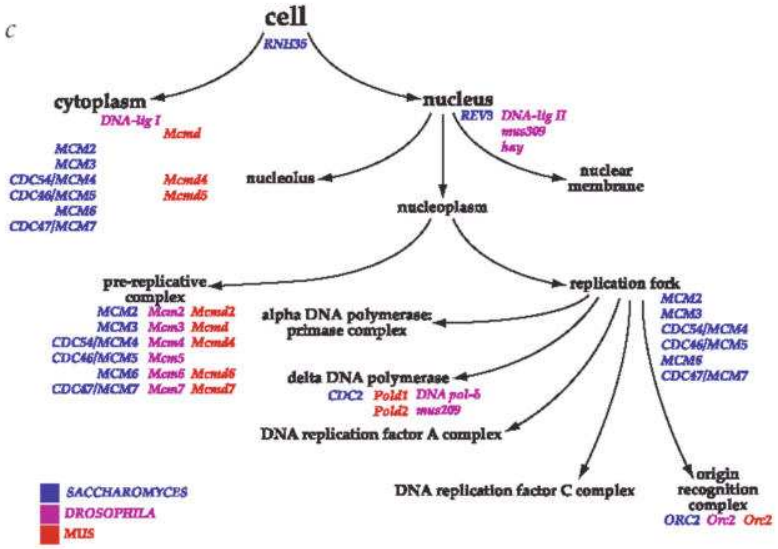

그림 37 Cellular Component

Data Representation

Function, Process, Component 등은 DAGs(directed acyclic graphs)로 표현된다. DAG(방향성 비순환 그래프)는 계층구조(Hierarchy)와는 다른 점이 있다. DAG는 Child가 여러 개의 Parent를 가질 수 있는 반면, Hierarchy Structure는 하나의 Child가 하나의 Parent를 가질 수 밖에 없기 때문이다. 하나의 Child Term은 자신의 Parent Term에 대한 instance이거나(isa relationship), 또는 component 이다(part-of relationship). Child Term은 하나 이상의 Parent를 갖고 , 서로 다른

Parent에 대해서 다른 클래스의 관계를 갖게 될 것이다. 새로운 DATA Format으로써 각광을 받고 있는 XML 역시 Hierarchy구조는 표현 가능하지만, DAG 는 표현하기 힘들다. 그렇기 때문에 GO는 새로운 Data Format을 제공한다. 이 파일들은 Text File 형태로 제공된다. 또한 GO 홈페이지에서는 새로운 문법으로 표현된 온톨로지를 쉽게 알아볼 수 있고, 검색도 가능한 Browser들을 제공한다. XML version도 제공하는데 , 여기서는 isa relationship , part-of relationship을 표현하기 위해, rdf link 라는 새로운 표현을 사용한다. GO homepage에서는 XML version의 온톨로지 파일(File)과 DTD 파일을 제공한다.

Abstract

Future of current Web is the Semantic Web which gives a well defined semantics to web resources. So, such things give not only human but also computer ability to understand semantics of document. RDF(Resource Description Framework) supports semantic connection among web resources and uses a triple structure to express meta-data. RDF-Schema is a predefined vocabulary to express resource's structure or relation between resources. This paper describes the problem with storing triple data to relational database system and proposes problem solving way by using semantic information of rdfs:domain and rdfs:range of RDF-Schema vocabulary. Moreover, query processing by using our approach is more efficient in that data search space is reduced.

Keywords: Semantic Web, RDF/RDFS, OWL, domain, range, Query Processing.

Student Number : 2003 - 21539

감사의 글

대학원에 입학하여 OOPSLA 연구실에 들어온 것이 엊그제 같은데 벌써 졸업이 다가왔습니다. 연구실 선배님으로부터 2년 석사기간은 눈 깜짝할 사이에 지나간다고 들었는데, 그때 그 의미를 지금에 와서야 알 것 같습니다. 지금 돌이켜 보면 연구실 생활을 좀더 열심히 했더라면 더욱 좋은 논문이 나오지 않았을까 하는 후회도 하지만 그동안의 노력으로 미흡하나마 그동안의 연구 결과로서 이제 작은 결실을 보게 되었습니다. 정보화 본부장으로서 늘 바쁘신 와중에도 2년 동안 언제나 변함없이 격려해 주시고 지도해 주신 김 형주 교수님께 감사드립니다. 무엇보다도 제가 낯선 시멘틱 웹 연구 분야에 접근 할 수 있도록 많은 도움을 주신 정민이형, 경섭이형, 상원이형, 호영이 형에게 진심으로 감사의 마음을 전합니다.

연구실에서 2년 동안 같이 고생했던 저의 동기인 종남이형, 윤호, 동혁, 기성에게도 고마운 마음을 전하고 싶습니다. 저의 동기들이 늘 주위에서 도와주웠기 때문에 2년이란 생활을 즐겁게 한 것 같습니다. 또한 연구 분야는 달랐지만 여러 가지로 도움을 주신 살아있는 지식엔진 형동이형, 일환이형, 홍래형, 지열이형 그리고 준원에게도 모두 감사를 드립니다. 그리고 굳은 일을 도맡아서 석사 논문에 집중할 수 있게 해준 태휘, 형우, 강표에게도 고마움을 표시합니다. 또한 지금은 졸업을 하였지만 연구실에서 여러 가지 조언을 해 준 종익형에거도 깊은 감사를 드립니다.

항상 저의 든든한 후원자인 누나, 매형에게 깊은 감사의 마음을 표시하며, 끝으로 오늘날의 제가 있게 해 주신 존경하고 사랑하는 부모님께 이 논문을 바칩니다.