시맨틱 웹에서의 효율적인 온톨로지 추론을 위한 개선방법에 관한 연구

A Study on Methodology for Efficient Ontology Reasoning in the Semantic Web

홍준석(June Seok Hong)*

초 록

온톨로지를 이용한 시맨틱 웹은 의미 기반의 표현 수단으로써 기존의 웹이 갖는 한계점을 극복할 수 있는 차세대 웹의 표준으로 인식되고 있다. 시맨틱 웹에 표현된 정보를 최대로 활용하기 위해서는 온톨로지에 대한 질의 검색 및 추론 기능이 필요한데, 대부분의 시맨틱 웹 도구들은 RDF 메타데이터 구조에 따른 Triple 기반의 저장 구조를 이용함으로 인해 온톨로지 추론을 위한 의미 단위의 복합 질의를 효율적으로 지원하지 못하고 있다. 본 연구에서는 기술 논리(DL)에 기반하여 온톨로지 데이터 구조와 일치하는 저장 구조를 설계하고, 이를 이용하여 시맨틱 웹 온톨로지에 대한 질의 검색 도구를 개발함으로써 온톨로지 추론을 위한 효율적인 복합 질의 검색을 지원할 수 있는 개선 방법을 제시하고자 한다. 그리고 제안된 방법을 구현한 시스템인 SMART-DLTriple을 기존의 시스템과 비교하여 그 성과를 평가하였다. 개선된 온톨로지 질의 검색 방법은 온톨로지 추론의 성능 향상에 기여하여 실용적인 온톨로지 추론 시스템의 개발에 도움을 줄 것이다.

ABSTRACT

The semantic web is taken as next generation standards of information exchange on the internet to overcome the limitations of the current web. To utilize the information on the semantic web, tools are required the functionality of query search and reasoning for the ontology. However, most of semantic web management tools cannot efficiently support the search for the complex query because they apply Triple-based storage structure about RDF metadata. We design the storage structure of the ontology in corresponding with the structure of ontology data and develop the search system(SMART-DLTriple) to support complex query search efficiently in this research. The performance of the system using new storage structure is evaluated to compare with the popular semantic web management systems. The proposed method and system make a contribution to enhancement of a practical ontology reasoning systems due to improved performance of the ontology search.

키워드 : 시맨틱 웹, RDF, OWL, 기술논리, 저장 구조
Semenatic Web, RDF, OWL, Description Logic, Storage Structure

---

본 연구는 2006학년도 경기대학교 학술연구비(일반연구과제) 지원에 의하여 수행되었음.
* 경기대학교 경영정보학과 부교수
2008년 04월 18일 접수, 2008년 06월 09일 심사완료 후 2008년 06월 16일 게재확정.

1. 서 론

웹을 포함한 인터넷 상에서의 메타데이터 정보를 이용하여 상호운용성을 지원하면서 지식의 표현 및 교환을 목적으로 하는 시맨틱 웹은 메타데이터 표현 언어인 RDF(Resource Description Framework)[4], 이러한 메타데이터의 구조를 표현하기 위한 스키마 언어인 RDFS(RDF Schema)[5], 그리고 메타데이터의 의미를 표현하기 위한 온톨로지 언어인 OWL(Web Ontology Language)[2] 등을 기반 언어로 사용하고 있다. 이러한 언어들을 통해 온톨로지를 활용함으로써 시맨틱 웹은 단순한 연결만을 의미하는 하이퍼링크만을 제공하는 기존의 웹이 갖는 한계점을 극복하여 차세대 웹의 표준으로 인식되고 있다. 시맨틱 웹 상에서 RDF(S)와 OWL 등의 언어로 표현된 정보를 최대로 활용하기 위해서는 이들 언어들로 표현된 메타데이터와 온톨로지를 저장 및 관리하고, 사용자의 요구에 따라 질의에 응답하며, 온톨로지 추론이 수행된 결과까지도 제공할 수 있어야 한다.

시맨틱 웹 환경을 대상으로 이러한 기능을 제공하기 위하여 지금까지 연구되고 개발되어 온 대부분의 시맨틱 웹 도구들은 온톨로지가 갖는 의미 중심의 데이터 구조보다는 온톨로지의 저장 및 관리가 용이하다는 점 때문에 RDF 메타데이터 구조를 그대로 구현한 Triple (또는 N3) 기반의 저장 구조를 이용하고 있다. 온톨로지의 이론적 기반인 기술 논리(Description Logics, DL)도 대부분의 선언식은 단항식 또는 이항식으로 표현되어 Triple 구조에 적합하다[9]. 시스템의 관점에서도 Triple 단위의 저장 구조를 이용하게 되면, 이들 언어로 표현된 온톨로지를 저장 구조로 적재하는 일이 매우 간단하게 되며, 온톨로지를 포함한 메타데이터에 대한 Triple 단위의 질의는 효율적으로 처리가 가능하다. 그러나 온톨로지의 표현과 추론에 자주 사용되는 rdf : List나 owl : Restriction 등과 같이 기술 논리 상에서 하나의 의미 단위가 복수 개의 Triple로 표현될 수밖에 없는 경우에 대하여 검색을 처리하기 위한 복합 질의 처리는 불가능하거나, 가능한 경우에도 매우 낮은 성능을 보일 수밖에 없다. 그로 인해 이와 같은 질의 처리 기능을 주로 이용하는 온톨로지 추론 기능은 매우 복잡하게 설계되거나 추론 속도에 심각한 지장을 초래하게 된다.

본 연구에서는 시맨틱 웹에서의 효율적인 온톨로지 추론을 지원하기 위하여 온톨로지 표현 및 추론의 이론적인 바탕이 되고 있는 기술 논리(DL) 범위의 OWL 온톨로지(OWL-DL)를 대상으로 온톨로지 의미 단위의 데이터 구조와 일치하는 온톨로지 저장 구조를 설계하고, 이를 이용하여 온톨로지 추론을 위한 효율적인 복합 질의를 처리할 수 있도록 온톨로지의 적재 및 저장, 질의 검색 기능 등을 포함한 온톨로지 관리의 개선 방법을 제시하고자 한다. 그리고 제안된 방법을 기존의 온톨로지 관리 도구인 SMART-Triple과 연결하여 기능적으로 향상된 Java 컴포넌트 기반의 온톨로지 관리 시스템 SMART-DLTriple로 구현하고, 이를 지금까지 널리 알려진 온톨로지 관리 시스템인 SMART-Triple, Jena 등의 시스템들과 질의 검색 처리에 대한 효율성을 평가하기 위하여 성능 비교를 수행하였다.

본 논문의 제 2장에서는 시맨틱 웹 관리 도구에 관한 기존 연구들을 정리하고, 제 3장에

서 온톨로지 데이터의 구조를 파악하였다. 이를 이용하여 제 4장에서 새로운 온톨로지 저장 구조를 설계하고, 제 5장에서는 온톨로지 저장 구조를 이용한 온톨로지 질의 검색 시스템 SMART-DLTriple을 설명하였다. 마지막으로 제 6장에서 연구의 결론과 본 연구의 활용 방안에 대해 기술하였다.

2. 관련 문헌 연구

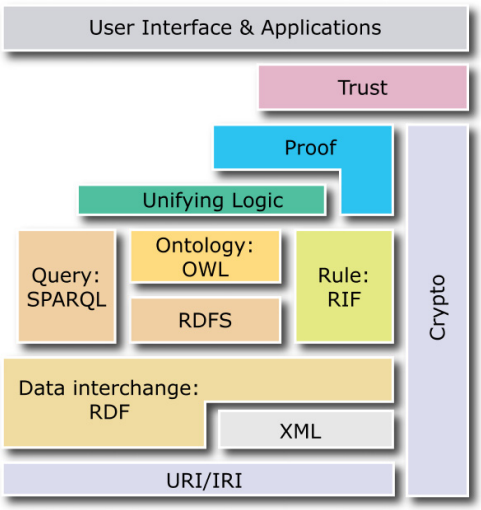
시맨틱 웹을 포함하여 RDF, OWL 등의 온톨로지 언어 표준을 이용하는 응용 시스템의 개발 도구로 사용될 온톨로지 저장 및 질의 처리 시스템은 개발 목적에 따라 크게 2가지 유형으로 구분된다[13]. 첫 번째는 시맨틱 웹 응용 시스템에 사용할 목적으로 메타데이터와 온톨로지에 대한 저장 및 검색 기능과 더불어 시맨틱 웹 개발용 API, 시맨틱 웹 온톨로지 추론 기능 등을 동시에 제공하는 시스템들이다. 가장 널리 알려져 현재 가장 많이 활용되고 있는 HP의 Jena[11]는 소스가 공개된(open-source) 시스템으로 메타데이터의 생성, 추출, 변경, 삭제 기능을 갖는 API를 제공할 뿐만 아니라 시맨틱 웹 데이터에 대한 온톨로지 추론과 규칙 추론 기능을 제공하여 시맨틱 웹 응용 시스템을 개발하는데 필요한 거의 모든 기능을 제공하고 있다. 유럽의 시맨틱 웹 그룹을 중심으로 개발된 KAON2[12]도 시맨틱 웹 응용 시스템 개발을 위해 필요한 메타데이터 사용 기능과 Jena 수준의 추론 기능을 함께 제공하는 시스템이다. 국내에서 개발된 시맨틱 웹 도구로는 RDF 메타데이터의 저장 및 관리, 검색 기능을 제공하는 SMART-Triple[1]을 중심으로 정방향/역방향 추론엔진을 이용한 온톨로지 추론 기능을 제공하는 SMART 프로젝트가 있다. 그 밖에도 시맨틱 웹 데이터를 대상으로 연관성 검색 기능을 주 목적으로 하는 BRAHMS[8], Java 뿐만 아니라 C#, Python, Perl, PHP 등 다중 언어 API를 제공하여 다양한 개발환경을 지원하는 Redland[3] 등과 같은 시스템이 있다.

두 번째 유형은 대용량의 메타데이터를 저장하기 위하여 기존의 데이터베이스 관리시스템을 기반으로 저장 및 검색 기능을 제공하는 시스템들이다. 소스 개방형으로 제공되는 인덱스 구조를 이용하여 저장된 RDF 데이터베이스를 대상으로 메타데이터 검색 기능에 추가적으로 RDF 스키마 추론 기능까지 제공하는 Sesame[6], 별도의 쿼리 언어를 이용하여 Java API를 제공하는 YARS[7], SPARQL 인터페이스와 RDF 스키마 추론 기능을 포함한 AllegroGraph[10] 등이 있으나, 대용량 온톨로지의 안정적인 처리에 초점을 맞추고 있다는 점에서 본 연구와는 무관하다.

3. OWL 온톨로지의 데이터 구조와 저장 구조

3.1 OWL 온톨로지 데이터 구조

온톨로지 데이터는 W3C에서 발표한 시맨틱 웹 프레임워크(<그림 1> 참조)에 따라 URI로 표현된 웹 자원에 대하여 XML 표준 문법을 따르면서 RDF 메타데이터의 표현 구조인 Triple을 기본 구조로 삼고 있다. RDF



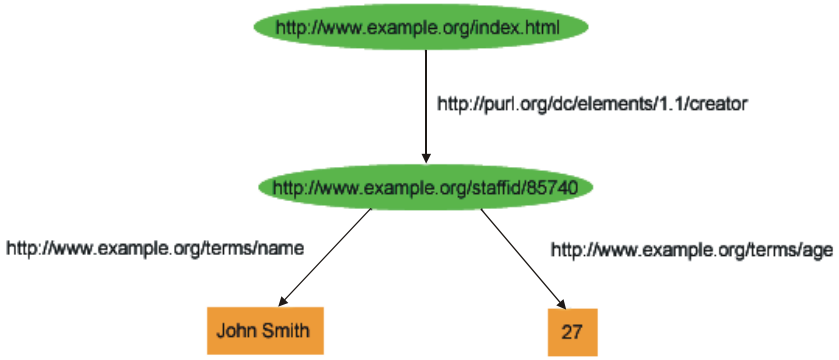
〈그림 1〉 시맨틱 웹 프레임워크〔14〕

메타데이터 데이터는 URI로 식별되는 웹 자원의 속성에 대한 값을 하나의 선언문으로 표현하며, 이는 기술하려는 대상인 주어(subject), 대상의 속성인 술어(predicate), 속성의 값인 목적어(object)의 쌍으로 표현되는 것이다. 예를 들어, 'http://www.example.org/index.html'이라는 웹 자원인 홈페이지는 저작자('http://purl.org/dc/elements/1.1/creator')라는 속성의 값으로 'http://www.example.org/staffid/85740'라는 인스턴스 개체를 갖고, 이 인스턴스 개체는 이름('http://www.example.org/terms/name')과 나이('http://www.example.org/terms/age') 속성의 값으로 각각 'John Smith'와 '27'을 갖는다는 선언문을 다음과 같은 세 개의 Triple로 표현할 수 있다.

(<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740>)
(<http://www.example.org/staffid/85740>
<http://www.example.org/terms/name>
"John Smith")
(<http://www.example.org/staffid/85740>
<http://www.example.org/terms/age> "27")

이러한 선언문들은 <그림 2>와 같이 주어를 노드로, 술어를 아크로, 목적어를 다시 노드로 하는 RDF 그래프로 표현할 수도 있다[4].

이와 같은 메타데이터의 구조를 정의하기 위해 사용하는 온톨로지 데이터는 메타데이터의 표현 구조인 Triple을 기본 구조로 하면서도 사전에 정의된 정형화된 데이터 구조를



〈그림 2〉 RDF 그래프의 예

갖고 있다. 메타데이터의 구조를 정의하기 위한 첫 번째 방법으로 비슷한 속성들을 공유하는 인스턴스 개체들의 모임을 정의하기 위하여 클래스(rdfs : Class or owl : Class)를 선언하고, 클래스의 내용을 정의하기 위하여 구성 개체들을 나열하는 방법(owl : oneOf), 속성에 대한 제약을 이용하는 방법(owl : Restriction), 다른 클래스들의 교집합(owl : intersection)과 합집합(owl : union)을 이용하는 방법을 제공한다. 그 중에서도 속성 제약을 이용하는 방법은 지정된 속성의 값을 범위 또는 개수의 조건에 맞게 갖는 인스턴스 개체들의 모임으로 클래스를 정의하는 방법인데, 속성의 값을 모두 특정 클래스의 인스턴스만 갖는 경우(owl : allValuesFrom), 속성의 값으로 특정 클래스의 인스턴스를 적어도 하나 이상 갖는 경우(owl : someValuesFrom), 특정 인스턴스만을 갖는 경우(owl : hasValue)의 세 가지와 속성의 값의 개수 조건으로 최대 개수(owl : maxCardinality), 최소 개수(owl : minCardinality), 특정 개수(owl : cardinality)로 제한하는 세 가지로 나눌 수 있다. 그리고 이와 같이 정의된 클래스들 간의 관계를 선언하는 방법으로 포함 관계(rdfs : subClassOf), 일치 관계(owl : equivalentClass), 배반 관계(owl : disjointWith), 여집합 관계(owl : complementOf) 등을 제공한다.

개체에 대하여 사용하는 속성(rdf : Property)도 사전적으로 정의하고, 선언된 속성들 간의 포함 관계(rdfs : subPropertyOf), 일치 관계(owl : equivalentProperty), 역 관계(owl : inverseOf) 뿐만 아니라 속성을 적용할 대상의 범위(rdfs : domain)와 속성 값의 범위(rdfs : range)에 대한 정의도 가능하다. 그리고 각 속성의 유형을 특징별로 특정 개체가 해당 속성의 값을 오직 하나만 갖는 경우(owl : FunctionalProperty), 역으로 해당 속성의 값을 동일하게 갖는 개체는 오직 하나 뿐인 경우(owl : InverseFunctionalProperty), 속성에 의한 연관성이 반복적으로 성립되는 경우(owl : TransitiveProperty), 속성에 의한 연관성이 역으로도 성립하는 경우(owl : SymmetricProperty) 등으로 구분할 수 있다.

마지막으로 각 개체에 대하여 개체들 간의 일치 관계(owl : sameAs)와 불일치 관계(owl : differentFrom)를 선언할 수 있으며, 다수의 개체에 대하여 상호 불일치 관계(owl : AllDifferent)를 한 번에 선언하는 것도 가능하다.

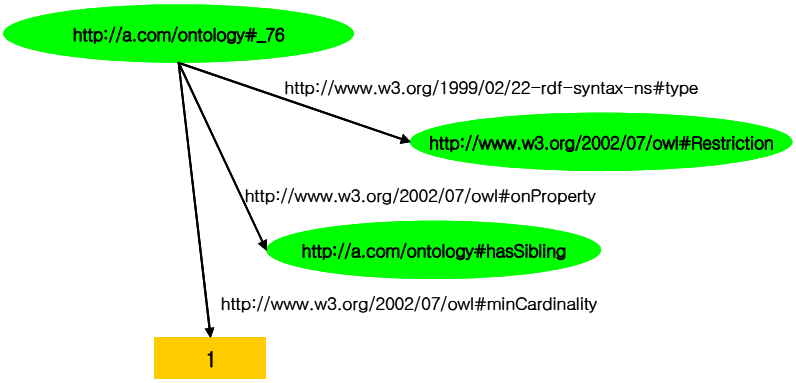
이러한 온톨로지 데이터는 대부분 하나의 의미 단위가 메타데이터의 기본 구조인 하나의 Triple로 표현된다. 예를 들어, 클래스 A가 클래스 B에 포함된다는 온톨로지 데이터는 아래와 같은 하나의 Triple로 표현할 수 있다.

(<http://a.com/ontology#A>
<http://www.w3.org/RDFS#subClassOf>
<http://a.com/ontology#B>)

그러나 온톨로지 추론에 필요한 하나의 의미 단위, 즉 기술 논리(DL)에서는 하나의 식으로 표현되는 경우가 복수 개의 Triple로만 표현되는 경우도 있다. 첫 번째는 RDF 메타데이터의 List를 이용하는 Collection이 사용되는 경우이다. 클래스 중에서 원소를 나열하는 방법의 경우와 교집합 또는 합집합을 이용한 경우에는 하나의 클래스 선언이 복수 개의

Triple로 구성될 수밖에 없고, 다수의 개체에 대해 상호 불일치 클래스를 선언하는 경우도 선언에 사용된 개체 수의 약 2배에 해당하는 Triple에 의해서 표현된다. 두 번째는 속성에 대한 제약으로 클래스를 선언하는 Restriction의 경우이다. <그림 3>과 <그림 4>는 각각 2개의 클래스의 합집합으로 클래스를 정의(DL 식으로는 'A ≡ B∪C')하기 위하여 7개의 Triple이 사용된 경우와 속성 제약으로 Restriction 클래스를 정의(DL 식으로는 '≥ 1hasSibling')하기 위하여 3개의 Triple이 사용된 경우의 예제를 보여주고 있다.

〈그림 3〉 합집합 클래스 선언의 예

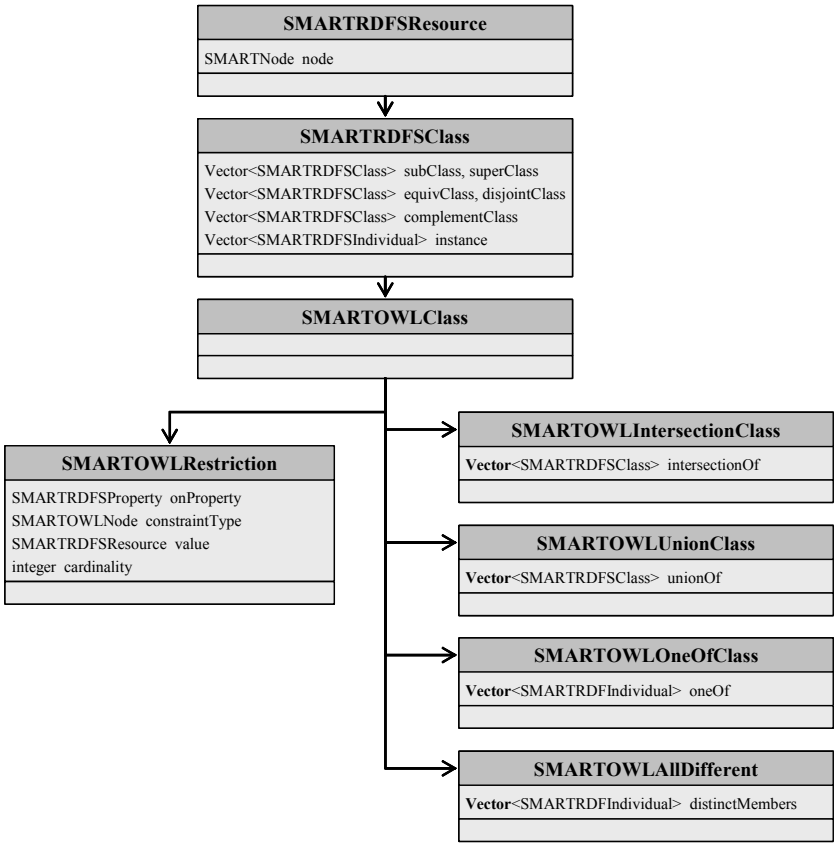


〈그림 4〉 속성 제약 클래스 선언의 예

3.2 OWL 온톨로지의 저장 구조

시맨틱 웹에서 사용되는 메타데이터를 포함하여 온톨로지 데이터를 효과적으로 저장하기 위해서는 앞 절에서 살펴본 온톨로지 데이터의 구조를 반영하고, 온톨로지 추론을 위한 질의 처리에 최적화되도록 저장 구조를 설계하여야 한다. 온톨로지 데이터의 이론적 바탕이 되고 있는 기술 논리(DL)에서는 데이터 유형을 메타데이터 구조 정의에 관련된 TBox(Teminological Box)와 이러한 구조를 따르는 메타데이터 인스턴스에 관련된 ABox(Assertional Box)로 구분하고 있다. 이와 같은 구분을 따라서 본 연구에서도 온톨로지 데이터인 TBox에 해당하는 데이터는 새로이 설계된 저장 구조에 반영하고, 메타데이터인 ABox에 해당하는 데이터는 기존의 메타데이터의 구조인 Triple 저장 구조를 그대로 이용하되 새로운 저장 구조로부터의 인덱스를 활용함으로써 질의 검색 처리에 대한 효율성을 높이고자 하였다.

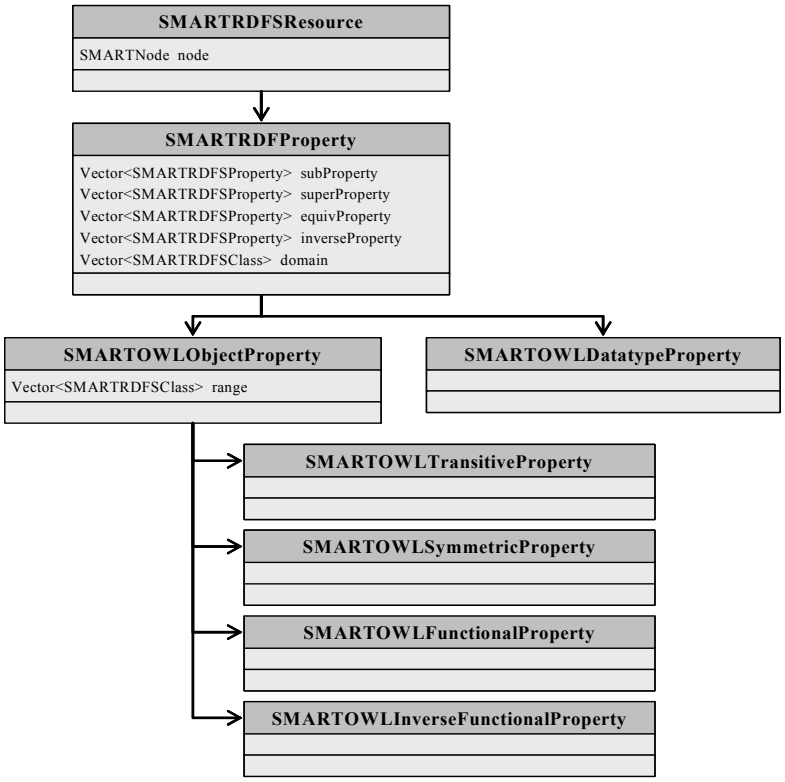
첫 번째로 클래스에 대한 내용을 정의한 데이터를 저장하기 위한 구조로 클래스들의 계층적 구조를 이용하여 <그림 5>와 같은



〈그림 5〉 클래스에 대한 저장 구조

저장 구조를 설계하였다.

가장 넓은 범위의 클래스인 RDFSClass를 최상위로 두고, 이를 상속받아 OWLClass를 만들고, 그 하위에 속성 제약으로 클래스를 정의하는 Restriction, 교집합과 합집합에 의해 클래스를 정의하는 IntersectionClass와 UnionClass, 그리고 원소를 나열하여 클래스를 정의하는 OneOfClass를 두었다. 마지막으로 다수의 개체에 대하여 상호 불일치를 선언하는 AllDifferent도 OWLClass의 하위에 설정하였다. 그리고 각 클래스의 수준에 따라 저장이 필요한 내용을 멤버로 포함시켰는데, 모든 종류의 클래스에서 가능한 포함 관계, 일치 관계, 배반 관계, 보완 관계를 갖는 클래스들을 바로 검색할 수 있도록 RDFSClass에 subClass, equivClass, disjointClass, complementClass의 멤버를 포함시켰고, 역 관계가 동일하게 성립하는 나머지 관계와 구별하여 포함 관계의 역 관계는 superClass의 멤버로 추가하였다. 그리고 하위의 저장 구조에는 각각의 클래스 선언에 필요한 데이터를 저장할 수 있도록 설계하였는데, Restriction은 제약을 적용할 속성(onProperty), 제약의 종류(constraintType), 속성 값의 클래스 또는 값(value)과 속성 값의 개수(cardinality)를 저장할 수 있는 멤버를 포함하고 있고, IntersectionClass와 UnionClass는 클래스 선언에 사용된 클래스들을 벡터로 저장



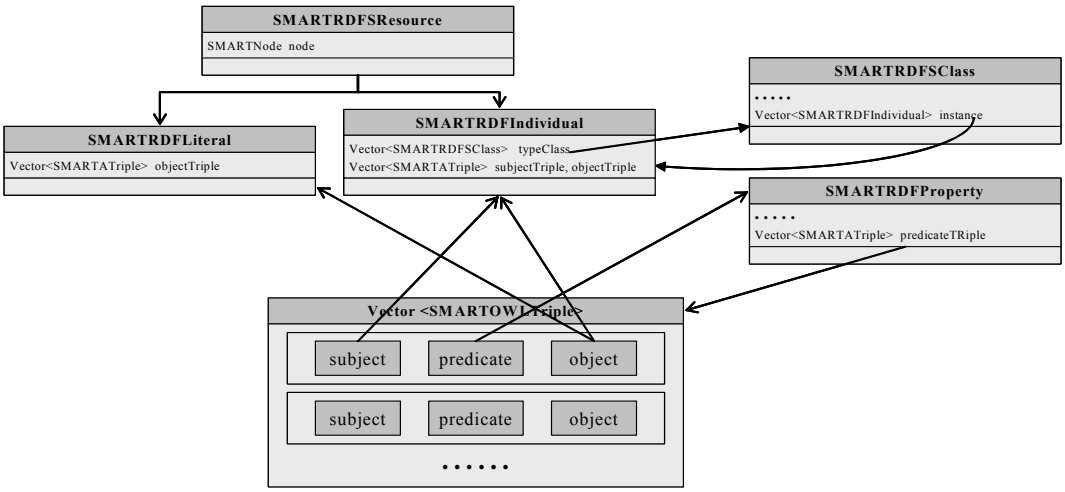
〈그림 6〉 속성에 대한 저장 구조

할 수 있도록 멤버로 포함하고 있다. OneOfClass도 클래스를 구성하는 개체들을 저장하는 벡터를 멤버로 포함시켜 구성하였다. 이와 같이 온톨로지 데이터에 대하여 하나의 의미 단위를 하나의 구조에 저장함으로써 온톨로지 추론에 필요한 단순 질의 및 복합 질의에 효율적으로 대응할 수 있게 된다.

두 번째로 속성에 대한 내용을 정의하는 데이터를 저장하기 위한 저장 구조도 RDFProperty를 최상위로 하고, 그 하위에 OWLObjectProperty와 OWLDatatypeProperty로 구분한 다음에, 속성의 종류에 따라 OWLTransitiveProperty, OWLSymmetricProperty, OWLFunctionalProperty, OWLInverseFunctionalProperty로 나누어 설계하였다. 그리고 속성들 간의 관계를 저장하기 위한 멤버로 모든 속성에 적용되는 포함 관계, 일치 관계, 역 관계, 개체 범위에 해당하는 subProperty, equivProperty, inverseProperty, domain을 포함시켰고, OWLObjectProperty와 그 하위의 속성에만 해당하는 값 범위는 range 멤버로 추가하였다. 속성에 대한 계층적 저장 구조와 각각의 멤버는 위의 <그림 6>과 같다.

3.3 메타데이터의 저장 구조 및 인덱스 구조

앞 절에서 설명한 바와 같이 메타데이터의 저장 구조는 주어, 술어, 목적어의 쌍인 Triple 저장 구조를 그대로 선택하였고, 이를 위하여 인스턴스 개체(Individual)와 값(Literal)에 대한 저장 구조를 설계하였다. 다만, 메타데이터 중에서 인스턴스 개체의 소속 클래스를 표시하는 rdf : type 속성에 관한 데이터는 이미 존재하는 클래스와 인스턴스 개체의 저장 구조에 추가적인 멤버(instance와 typeClass)로 포함시킴으로써 질의 검색의 효율성을 높였다. 인스턴스 개체의 저장 구조와 속성에 대한 저장 구조를 쌍으로 묶은 문장을 저장할 수 있는 Triple 구조와 이러한 Triple



〈그림 7〉 메타데이터의 저장 구조

에 대한 검색을 빠르게 수행할 수 있도록 각 개체 및 속성의 저장 구조에 연결된 인덱스를 아래의 <그림 7>과 같이 설계하였다.

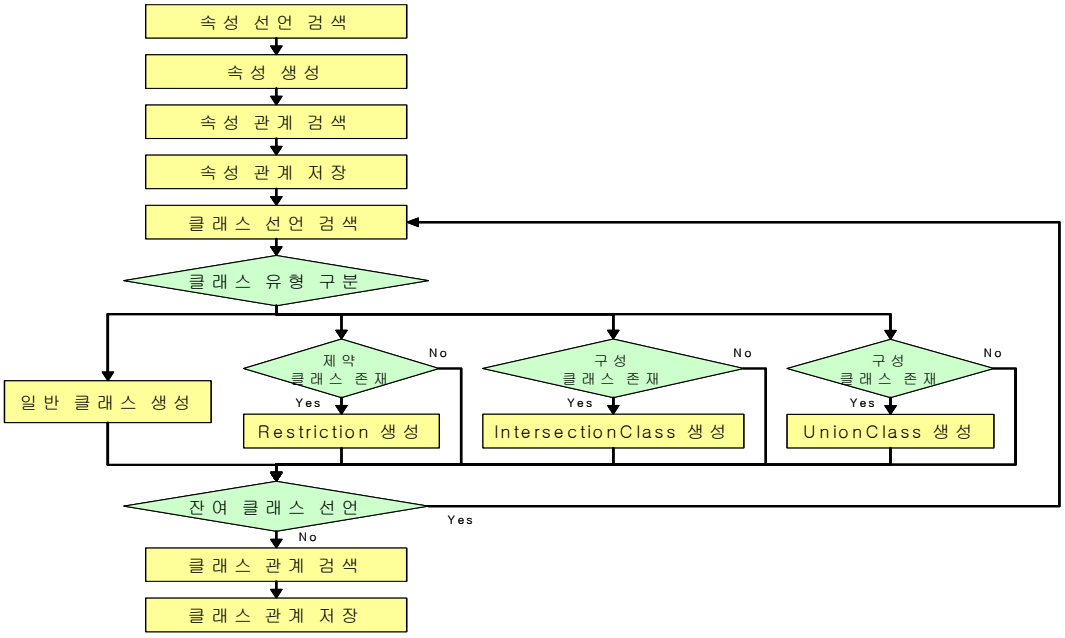
메타데이터가 저장된 각 Triple들을 검색하는 경우에 전체 메타데이터 Triple을 대상으로 하는 것이 아니라 주어, 술어, 목적어로 사용된 개체 또는 속성의 저장 구조로부터의 인덱스를 이용하여 제한적으로 검색함으로써 효율적인 질의 검색이 가능하게 되므로, 각 개체 또는 속성에 대한 빠른 검색을 통해 효율적인 질의 검색을 가능하게 해준다.

4. 개선된 온톨로지 질의처리 방법

4.1 OWL 온톨로지 적재 알고리즘

온톨로지 데이터를 새로이 설계한 저장 구조에 적재하기 위해서는 온톨로지 개체의 구성 요소에 따라 필요한 구성 요소가 다 적재된 후에 각 개체를 적재해야 하므로 아래와 같은 순서를 따라야 한다.

클래스를 정의하는데는 클래스의 종류에 따라 서로 다른 구성 요소를 필요로 한다. 먼저 Restriction은 속성을 반드시 필요로 하며, 경우에 따라서는 다른 클래스를 필요로 한다. 그리고 IntersectionClass와 UnionClass는 클래스 정의에 다른 클래스를 필요로 하고, OneOfClass는 인스턴스 개체를 이용하여 클래스를 정의한다. 따라서 다른 구성 요소를 필요로 하지 않는 속성에 대한 정의와 속성들 간의 관계를 가장 먼저 저장 구조에 적재하고, 다른 클래스를 필요로 하지 않는 클래스 정의를 저장한 후에 이를 이용하여 Restriction, IntersectionClass, UnionClass 등을 저장 구조에 적재한다. 그러나 이들은 서로를



클래스 정의에 이용하는 경우도 있으므로 반복적으로 수행할 필요가 있다. 그 다음에 OneOfClass는 필요한 인스턴스 개체들을 적재한 후에 클래스를 저장 구조에 저장한다.

온톨로지 데이터의 적재가 모두 끝나고 나면 그 다음으로 메타데이터를 저장하는데, 메타데이터 중에서 유일하게 저장 구조에 반영된 인스턴스 개체와 소속 클래스 간의 관계를 우선적으로 저장한 다음에 마지막으로 나머지 메타데이터를 Triple 구조에 저장하고 이들에 대한 인덱스를 생성하면 된다.

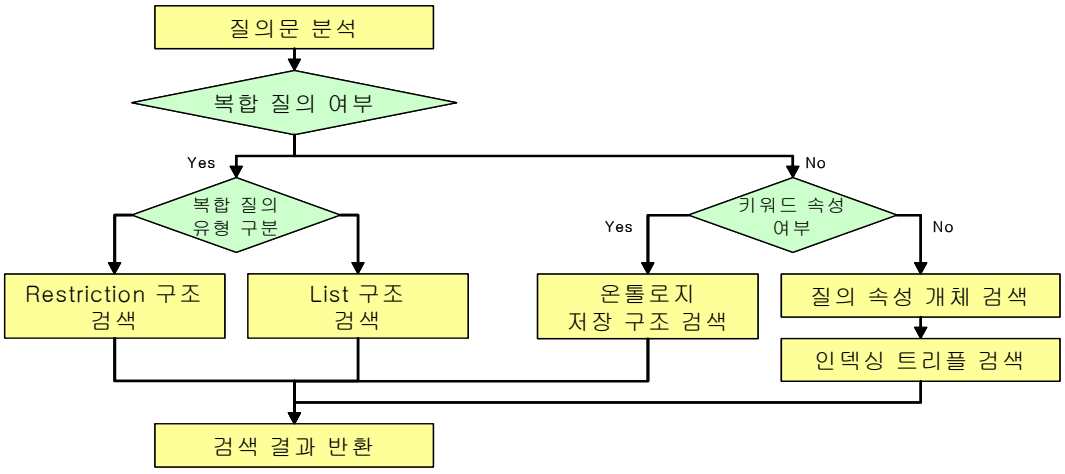
4.2 OWL 온톨로지 검색 알고리즘

저장된 온톨로지 데이터에 대한 검색은 저장 구조를 활용하기 위하여 질의에 따라 온톨로지 데이터에 대한 것인지 또는 메타데이터에 대한 것인지를 구별해야만 질의에 맞는 결과를 쉽게 검색할 수 있다. 모든 데이터는 술어를 기반으로 구성되므로 술어가 온톨로지를 구성하는 속성인 경우(즉, OWL-DL의 키워드 속성인 경우)에는 해당 온톨로지 데이터가 저장된 구조를 활용하고, 그렇지 않은 경우에는 질의의 술어에 해당하는 속성의 저장 구조로부터 인덱싱되어 있는 Triple들을 검색함으로써 효율적인 Triple 검색이 가능하다. 또한 다수의 Triple이 결합되어 하나의 의미를 갖는 List나 Restriction 등에 대한 복합 질의의 경우에는 해당 저장 구조를 이용함으로써 효율적인 질의 검색을 수행할 수 있다. 이러한 검색 과정은 아래와 같은 순서로 구성된다.

5. 온톨로지 검색 시스템 : SMART-DLTriple

5.1 시스템 구조

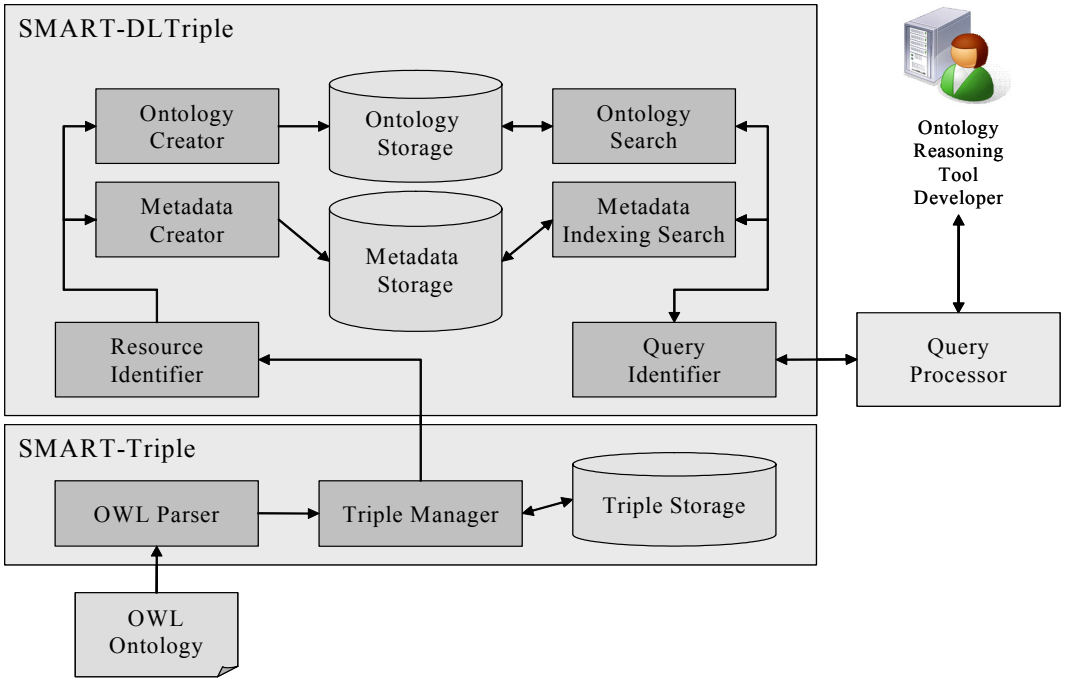
기술 논리(DL)에 기반한 온톨로지 검색 시스템인 SMART-DLTriple은 앞 절에서 설계한 저장 구조에 따라 온톨로지 추론을 위한 복합 질의에 효율적으로 대응할 수 있도록 시맨틱 웹 개발 도구인 SMART 프로젝



트의 구성 요소 중 하나인 RDF 검색 시스템 SMART-Triple을 기반으로 하여다음과 같은 구조로 개발되었다.

SMART-DLTriple 시스템은 OWL 온톨로지를 적재하기 위하여 Triple 기반의 저장 구조를 활용할 수 있는 SMART-Triple 시스템을 이용한다. Triple 구조의 저장소로 적재된 OWL 온톨로지를 자원의 유형에 따라 클래스, 속성 등에 관한 온톨로지 데이터는 Ontology Creator를 통해 구분에 맞게 설계된 저장 구조를 이용하여 Ontology Storage에 저장하고, 그 밖의 메타데이터는 Metadata Creator를 통해 온톨로지 저장 구조에 인덱스로 연결된 Triple 구조의 Metadata Storage에 저장한다. Ontology Creator는 제 4장의 적재 알고리즘에서 설명한 바와 같이 클래스와 속성의 종류에 따라 적절한 저장 구조를 생성하여 관련 데이터를 저장하는 기능과 생성된 클래스 및 속성 개체들에 대해 연관성 데이터를 저장하는 기능을 가지고 있다. Metadata Creator는 필요한 인스턴스 개체를 생성하고, 이미 생성된 클래스 및 속성 개체들과 인스턴스 개체를 이용하여 새로 생성된 메타데이터 Triple을 인덱싱하는 기능을 수행한다.

온톨로지 추론을 위한 질의 검색의 경우에도 제 4장의 검색 알고리즘에서 설명한 바와 같이 온톨로지 데이터에 대한 질의인지 메타데이터에 대한 질의인지를 판정하기 위하여 Query Identifier를 이용한다. 그 결과에 따라 온톨로지 데이터에 대한 질의인 경우에는 Ontology Storage에 대하여 적절한 저장 구



〈그림 8〉 SMART-DLTriple 시스템 구조도

조를 검색하는 기능을 갖고 있는 Ontology Search 컴포넌트를 이용하고, 메타데이터에 대한 질의인 경우에는 관련 속성 및 인스턴스 개체를 찾아 그 인덱스를 이용하여 결과 Triple을 검색하는 Metadata Indexing Search 컴포넌트를 이용한다. Ontology Search에서 검색할 저장 구조를 구분하는 방법은 검색 알고리즘에서 설명한 바와 동일하다.

이 시스템은 Java 컴포넌트를 이용하여 SMART-Triple을 포함한 SMART 프로젝트의 다른 시스템들과 호환가능하도록 개발되었다.

5.2 온톨로지 질의 검색의 성능 비교

SMART-DLTriple 시스템의 성능을 평가하기 위하여 일반적인 온톨로지 및 메타데이터에 대한 검색 성능과 함께 기술 논리(DL) 상에서의 복합 질의에 대한 검색 성능을 유사 기능을 가진 시스템들과 비교측정하였다. 대표적인 RDF 관리 도구인 SMART-Triple의 경우에는 복합 질의에 대한 검색을 지원하지 않고 있으나, 역방향 추론엔진을 이용하여 SPARQL로 표현된 복합 질의를 처리하고 있으며, Jena의 경우에는 SPARQL 처리 기능을 내장하고 있어 복합 질의 처리 기능을 직접 수행할 수 있다. 이러한 성능 비교를 위해 정보통신부에서 2005년도에 발주한 시맨틱 웹 에이전트 프로젝트를 통해 구축된 영화 온톨로지를 대상으로 하였는데, 이 온톨로지는 전체 37,000여 개의 트리플로 구성되어 있으며, 1588개의 클래스와 471개의 속성을 포함하고 있다.

먼저, 이와 같은 온톨로지를 대상으로 온톨로지를 각자의 저장 구조에 적재(loading)하는 시간을 비교하였는데, 각각 10회씩 반복하여 테스트한 결과가 다음의 <표 1>과 같다.

〈표 1〉 온톨로지 적재 성능의 평가 결과

	SMART-DLTriple	SMART-Triple	Jena
소요 시간(sec)	8.078	2.156	2.315

테스트 온톨로지를 적재하는데 SMART-Triple과 Jena는 비슷하게 2초 정도의 시간이 소요되었으나, SMART-DLTriple은 약 4배에 가까운 8초 정도의 시간이 소요되었다. 이는 Triple 기반의 저장 구조를 이용하는 두 시스템과는 다르게 복잡한 저장 구조를 이용함으로써 동일한 온톨로지를 적재하기 위하여 더 많은 시간이 필요한 것으로 판단된다. 그러나 이후에서 보는 바와 같이 질의 검색에서 훨씬 우수한 성능을 보임으로써 이러한 단점은 감수할만 하다.

테스트 온톨로지에 대한 질의 검색은 4가지 유형으로 나누어 수행하였다. 먼저 단순 질의의 종류로 메타데이터에 대한 질의 검색과 온톨로지 데이터에 대한 질의 검색을 수행하였고, 복합 질의의 종류로는 속성 제약 클래스에 대한 질의 검색과 List를 이용하는 교집합 정의 클래스에 대한 질의 검색을 수행하였는데, 특히 교집합 정의 클래스에 대한 질의 검색은 List의 구성 요소가 2개, 3개, 4개인 경우에 대하여 각각 비교평가하였다.

메타데이터에 대한 질의와 온톨로지 데이터에 대한 질의에 해당하는 SPARQL 문장은 아래와 같으며, 각각에 대한 검색 성능을

500회 반복 수행의 평균 시간으로 평가한 결과가 <표 2>와 같다.

```
Meta.D :  SELECT ?x
          WHERE
          {  ?x  base : shownTheater
          base : Theater_11. }
```

```
Ont.D :   SELECT ?x
          WHERE
          {  ?x  rdfs : subClassOf
          base : Card. }
```

SMART-DLTriple 시스템은 단순 질의 검색의 경우에도 인스턴스 개체 및 클래스에 대한 저장 구조의 인덱스를 이용함으로써 다른 두 시스템에 비해 약간 우수한 성능을 보이고 있다.

속성 제약 클래스에 대한 질의는 3개의 Triple을 복합적으로 검색해야 하는 복합 질의로써 Triple 단위로 저장하고 이러한 저장 구조에 대해 검색하는 두 시스템에 비해 하나의 저장 구조(Restriction)에 관련 데이터를 모두 저장하는 SMART-DLTriple이 월등하게 우수한 성능을 보이고 있다. 속성 제약 클래스에 대한 테스트 질의의 SPARQL 문장은 아래와 같다.

```
Rest. :   SELECT ?x ?y
          WHERE
          {  ?x  rdf : type wl : Restriction.
             ?x  owl : onProperty
             base : affiliatedCard.
             ?x  owl : maxCardinality  ?y. }
```

속성 제약 클래스에 대한 질의 검색의 500회 반복 수행의 평균 시간이 SMART-Triple과 Jena는 모두 1msec에 가까운 반면에 SMART-DLTriple은 클래스 검색을 위한 최소한의 시간(약 0.08msec)만이 소요되었다. 이러한 결과는 다수의 Triple을 대상으로 한 검색이 아니라 제한된 클래스 저장 구조에 대한 검색을 수행함으로써 가능한 것이다. 위와 같은 성능의 차이는 List를 이용하는 교집합 정의 클래스에 대한 검색 성능에서 더욱 확연한 차이를 보이고 있다. 교집합 정의 클래스에 대한 질의가 얼마나 많은 Triple에 대한 검색을 필요로 하는지는 4개의 구성 요소를 갖는 교집합 정의 클래스에 대한 질의문의 예제를 통해서 알 수 있다. 따라서 Triple 저장 구조에 대한 검색의 경우에 검색 성능이 질의에 포함된 Triple의 개수에 영향을 받는다는 것을 알 수 있고, 온톨로지 데이터와 일치하는 저장 구조를 갖는 SMART-DLTriple은 질의 Triple의 개수가 아닌 질의 온톨로지 식의 개수에만 영향을 받는다는 것을 알 수 있다.

```
Inter(4) : SELECT ?x ?y1 ?z1 ?y2 ?z2 ?y3
           ?z3 ?y4 ?z4
           WHERE
           {  ?x  owl : intersectionOf  ?y1.
              ?y1  rdf : type  rdf : List.
              ?y1  rdf : first  ?z1.
              ?y1  rdf : rest  ?y2.
              ?y2  rdf : type  rdf : List.
              ?y2  rdf : first  ?z2.
              ?y2  rdf : rest  ?y3.
              ?y3  rdf : type  rdf : List.
              ?y3  rdf : first  ?z3.
              ?y3  rdf : rest  ?y4.
              ?y4  rdf : type  rdf : List.
              ?y4  rdf : first  ?z4.
              ?y4  rdf : rest  rdf : nil. }
```

속성 제약 클래스에 대한 검색 성능과 교

집합 정의 클래스에 대한 검색 성능의 평가 결과는 <표 2>과 같다. SMART-DLTriple 시스템이 다른 두 시스템에 비해 우수한 성능을 보이고 있다는 차이 이외에도 다른 두 시스템은 복합 질의에서 검색하는 Triple의 개수가 증가함에 따라 검색 시간이 증가하지만, SMART-DLTriple 시스템은 복합 질의를 구성하는 Triple의 개수와는 무관하게 하나의 저장 구조(IntersectionClass)만을 찾아가게 되므로 검색 시간에 거의 차이가 없다는 장점도 갖고 있다.

〈표 2〉 단순 및 복합 질의 검색 성능의 비교

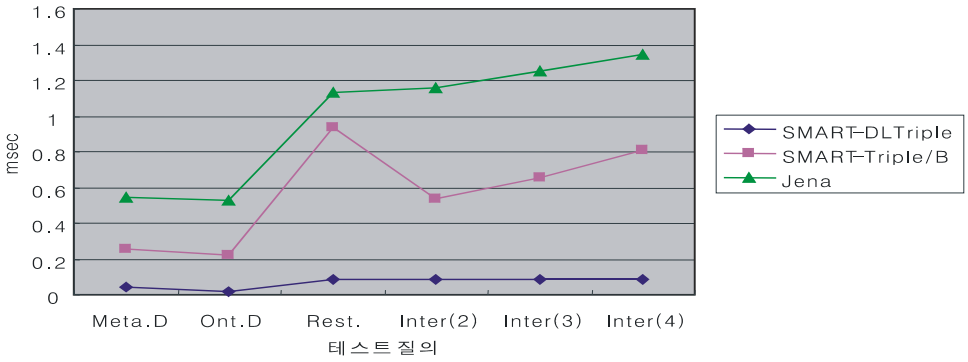
(단위 : msec)

	SMART-DLTriple	SMART-Triple	Jena
Meta.D	0.0438	0.254	0.546
Ont.D	0.0152	0.218	0.528
Rest.	0.0842	0.936	1.128
Inter(2)	0.0814	0.532	1.156
Inter(3)	0.0844	0.658	1.250
Inter(4)	0.0874	0.812	1.344

이상의 세 시스템 간의 질의 검색 성능에 대한 비교 평가 결과를 그래프로 비교해 보면 <그림 9>과 같다.

6. 결 론

본 논문에서는 차세대 웹으로 자리잡아 가고 있는 시맨틱 웹 환경에서의 온톨로지 데이터에 대한 질의 검색 기능을 효율적으로 수행하기 위하여 기존의 시맨틱 웹 도구들이 선택하고 있는 Triple 기반의 저장 구조 대신에 온톨로지 데이터 구조에 적합한 저장 구조를 제안하고 이를 구현한 온톨로지 검색 시스템 SMART-DLTriple을 개발하였다. 새로이 제안한 온톨로지 저장 구조의 효율성을 평가하기 위하여 기존의 대표적인 시맨틱 웹 도구들과 적재 시간 및 질의 검색 시간을 평가해 본 결과, 온톨로지 적재 시간은 많이 걸렸으나 온톨로지에 대한 단순 질의 검색에서 우수한 성능을 보였을 뿐만 아니라 복합 질의 검색에서는 질의가 복잡해질수록 더욱 효율적인 성능을 보여주고 있다. 본 연구를 통해 개발된 온톨로지 검색 시스템은 향후 효율적인 온톨로지 추론엔진을 개발하는 구성 요소로



〈그림 9〉 질의 검색 성능 비교 평가

써 매우 유용하게 활용될 수 있을 것이다.

참 고 문 헌

[1] 박지형, 이명진, 홍준석, “시맨틱 웹에서의 메타데이터 검색을 위한 RDF 저장 구조 및 시스템의 개발에 관한 연구”, 한국전자거래학회지, 제12권, 제2호, 2007, pp. 291-304.

[2] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Bell Labs Research, and Lynn Andrea Stein, OWL Web Ontology Language Reference, W3C Recommendation, February 10, 2004, http://www.w3.org/TR/owl-ref/.

[3] David Beckett, “The Design and Implementation of the Redland RDF Application Framework,” Computer Networks, Vol. 39, No. 5, 2002, pp. 577-588.

[4] Dave Beckett, RDF/XML Syntax Specification (Revised), W3C Recommendation, February Vol. 10, 2004, http://www.w3.org/TR/rdf-syntax-grammar/.

[5] Dan Brickley and R. V. Guha, RDF Vocabulary Description Language 1.0 : RDF Schema, W3C Recommendation, February Vol. 10, 2004, http://www.w3.org/TR/rdf-schema/.

[6] Jeen Broekstra, Arjohn Kampman and Frank van Harmelen, “Sesame : A Generic Architecture for Storing and Querying RDF and RDF Schema,” International Semantic Web Conference 2002, Sardinia Italy, 2002.

[7] Andreas Harth and Stefan Decker, “Optimized Index Structures for Querying RDF from the Web,” the Third Latin American Web Congress, Argentina, October 2005.

[8] Maciej Janik and Krys Kochut, “BRAHMS : A WorkBench RDF Store And High Performance Memory System for Semantic Association Discovery,” the Fourth International Semantic Web Conference ISWC 2005, Galway, Ireland, November 2005.

[9] Nardi, D., Brachman, R. J., “An Introduction to Description Logics,” in the Description Logics Handbook, edited by F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, Cambridge University Press, 2002, pp. 5-44.

[10] AllegroGraph, http://www.franz.com/products/allegrograph/.

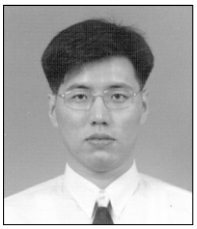
[11] Jena, http://jena.sourceforge.net/.

[12] KAON2, http://kaon2.semanticweb.org/.

[13] Semantic Web Development Tools, http://esw.w3.org/topic/SemanticWebTools, W3C Wiki.

[14] W3C Semanic Web LayerCake Diagram, http://www.w3.org/2007/03/layerCake.png.

저 자 소 개



홍준석	(E-mail : junehong@kyonggi.ac.kr)
1989년	서울대학교 경영학사 (경영학과)
1991년	KAIST 경영과학 석사 (경영과학과)
1997년	KAIST 경영공학 박사 (테크노경영대학원)
1998년	한국전자거래표준원 선임연구원
1999년~2002년	인제대학교 경영학과 조교수
2003년~현재	경기대학교 경영정보학과 부교수
관심분야	시맨틱 웹, 온톨로지 추론, 지능형에이전트, 자동협상시스템, 전자상거래