데이터베이스 시스템에 기반한 효율적인 OWL 저장시스템 설계 및 성능분석

조성환[†] · 김성식[††] · 김태영[†††]

요 약

시멘틱 웹은 현재의 웹의 한계에 대한 반성으로 등장하였고, W3C를 주축으로 OWL이라는 온톨로지 표준 기술(description) 언어를 권고하는 수준에까지 이르고 있다. 또한, OWL 데이터에 표현된 정보를 검색하기 위한 Jena, Jess, JTP와 같은 추론기들이 개발되고 있다. 하지만, 아쉽게도 현재까지는 OWL 데이터의 효율적인 저장 및 질의 처리 시스템은 찾아보기 힘들뿐만 아니라, 파일을 기반으로 처리되는 현재 추론기들의 실정상 대용량의 OWL 데이터를 처리하기에는 많은 제약을 가지고 있다. 따라서 온톨로지 상에서의 안정적인 정보 검색을 위해서는 온톨로지 데이터를 효율적으로 저장하고 검색하는 기법이 뒷받침되어야 한다. 이에 본 연구에서는 첫째로, OWL로 기술된 온톨로지 데이터를 데이터베이스에 변환하여 저장하고 데이터베이스 내에서 추론을 지원할 수 있는 모델을 제안하였고, 둘째로, 데이터베이스 시스템에 기반을 둔 OWL 저장 시스템을 설계 및 구현하였으며, 마지막으로, 제안한 시스템을 기존 추론기 시스템과의 성능 차이 실험 비교를 통해 분석하였다.

주제어 : 온톨로지, 시멘틱 웹, 추론, OWL

The Design and Performance Analysis of an Effective OWL Storage System Based on the DBMS

Seong-Hwan Cho[†] · Seong-Sik Kim[††] · TaeYoung Kim[†††]

ABSTRACT

Having observed the restriction of the current Web technology, the Semantic Web has been developed, and it now has grown up with the core help of the W3C to a level where it recommends the OWL Web ontology language. Besides, in order to deduce the information out of OWL data, several inference systems have been developed such as Jena, Jess, and JTP. Unfortunately, however, quite few systems can effectively handle recently developed OWL data, and further, due to the limitation of file-based operation, the current inference systems cannot meet the requirements for handling huge OWL data. An efficient method for storing and searching ontology data is essential for ensuring stable information inference processes. In this study, firstly, we proposed a model based on the database management system to transform and store OWL data and to enable deduction process from the database. Secondly, we designed and implemented an effective OWL storing system based on our model. Finally, we compare our system with the previous inference systems through experimental performance analysis.

Keywords : Ontology, Semantic Web, Inference, OWL

† 정 회 원: 한국교원대학교 컴퓨터교육과 박사과정
†† 종신회원: 한국교원대학교 컴퓨터교육과 교수
††† 종신회원: 한국교원대학교 컴퓨터교육과(교신저자)
논문접수: 2008년 8월 27일, 심사완료 : 2008년 9월 27일

1. 서 론

월드 와이드 웹(World Wide Web)은 10여 년 전에 팀 버너스리에 의해서 제안된 이후 놀랄만한 속도로 발전을 거듭하여 현재는 사람들의 생활의 일부로 자리 잡을 정도로 유용하게 사용되고 있다. 하지만, 사용자가 점점 늘어나고 공유하는 정보의 양이 증가함에 따라서 그에 따른 문제점도 제기되고 있다. 그것은 검증되지 않은 정보의 폭발적인 증가와 그로 인해 사용자가 원하는 정보를 정확히 찾을 수 없으며, 검색 결과 내에서 자신에게 알맞은 사항들을 다시금 걸러내야 하는 2차 검색의 문제를 낳는다는 것이다.

이의 가장 큰 원인은 현재의 웹이 기계가 아닌 사람이 보고 이해할 수 있도록 만들어졌다는 데에 있다고 보고, 많은 연구자들은 웹상의 데이터의 의미를 사람이 아닌 컴퓨터가 이해하고 처리하도록 사용자의 요구를 정확하게 받아들일 수 있도록 하는 시멘틱 웹(Semantic Web)이라 불리는 차세대 웹을 연구하기 시작했다[4].

다시 말해서 시멘틱 웹이란 웹상에 존재하는 자료에 의미를 부가하여, 컴퓨터가 정보의 의미를 분석할 수 있도록 함으로써 사용자가 원하는 정보를 정확히 추출해내는 것이 가능하며, 더 나아가서 인공지능적인 서비스를 제공해줄 수 있는 것이다[11][12]. 시멘틱 웹에 대한 연구와 표준화 작업을 주도하는 W3C에서는 OWL 온톨로지 언어를 기반으로 한 시멘틱 웹을 권고하고 있다.

한편, 기존에 연구된 온톨로지 데이터 개발 시스템으로는 KAON, OntoEdit, Protege2000 등이 있지만, 온톨로지 데이터를 효율적으로 저장하고 관리하는 기술과 검색기능이 뒷받침 되지 못한다는 단점을 가지고 있다[6]. 또한, 온톨로지 데이터의 추론(inference)을 지원하는 Jena[17], Jess, JTP와 같은 기존 추론 엔진은 온톨로지 데이터를 파일에서 읽어 검색해야 하며 데이터의 삽입, 삭제, 갱신을 위해서는 온톨로지 데이터를 프로그래머가 직접 수정하고 갱신시켜야 하는 번거로움이 생긴다. 따라서 이들은 모두 대용량의 온톨로지 데이터의 효율적인 저장 및 질의 처리는 지원하지 못한다는 큰 단점을 가지고 있다.

그러므로 본 연구에서는 대용량 온톨로지 데이터의 처리가 가능하며 데이터의 삽입, 삭제, 갱신이 편리한 RDB(Relational Database System)으로의 변환 및 저장 가능성을 확인하고, 기존의 추론기 기반 검색 시스템의 기능을 RDB에서 지원할 수 있도록 검색시스템을 설계 및 구현하였다. 또한 기존의 Jena 추론기 기반 검색 시스템과 검색 성능을 상호 비교하였다. 이를 자세히 설명하면 아래와 같다.

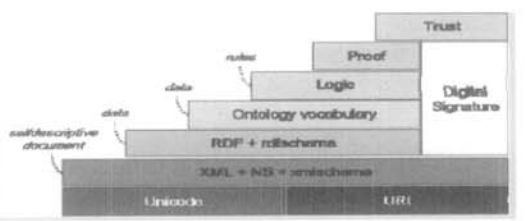
첫째, OWL Lite 구문들인 Class, Property, Instance 관련 Typical Syntax들을 분석하고, 이를 바탕으로 관계형 변환모델을 설계하였다.

둘째, 설계한 관계형 모델에 기반을 둔 온톨로지 데이터 검색 시스템을 구현하기 위하여 초등학교 3학년부터 6학년까지의 과학과 교과서 '생물' 관련 단원과 교사용 지도서를 분석하였고, Ontology Development 101 단계[18]에 기초하여 생물 지식 온톨로지를 설계하였으며, 이를 OWL Lite 구문을 이용하여 구축하였다.

셋째, 기존 Jena 추론기를 기반으로 한 온톨로지 데이터 검색 시스템과 제안한 관계형 데이터베이스 시스템에 기반을 둔 온톨로지 데이터 검색 시스템의 검색 성능을 실험적으로 비교하였다.

2. 시멘틱 웹과 온톨로지

시멘틱 웹은 웹상에 있는 정보에 컴퓨터가 이해하고 처리할 수 있도록 의미를 부여하여, 정보의 정확한 검색과 통합, 공유가 가능하도록 하는 것이다[13]. 현재 시멘틱 웹에 관한 연구와 표준화 작업은 W3C를 중심으로 이루어지고 있으며, 시멘틱 웹을 구축하기 위한 시멘틱 웹 계층구조는 <그림 1>과 같다.



<그림 1> Semantic Web 계층구조[8]

시멘틱 웹에 대한 관심이 증가하면서 시멘틱 웹을 완성하기 위한 기술로 온톨로지(ontology)가 각광을 받게 되었다. 온톨로지란 개념들과 이들을 의미적으로 연결하는 관계들로 구성된 사전으로서 어느 특정 도메인에 관련된 용어들을 계층적 구조로 표현하고 추가적으로 이를 확장할 수 있는 추론 규칙을 포함하는 것이다[14][18].

지금까지 개발된 대표적인 온톨로지 표현 언어들로는 XML 기반의 XOL(Ontology Exchange Layer), OXML(Ontology Markup Language), SHOE(Simple HTML Ontology Extension)등이 있고, RDF, RDFS를 기반으로 하는 OIL, DAML+OIL 등이 있지만[2], 표현력과 사용의 용이성에서 우수함으로 인해 최근 동향으로 OWL(Web Ontology Language)이 제시되고 있으며, W3C에서도 웹 온톨로지 언어로 OWL을 권고하고 있다[1].

3. 기존 연구 분석

OWL은 RDF(S)를 기반으로 하는 온톨로지 언어로서, 단지 사람에게 정보를 표시하는데 그치지 않고 컴퓨터가 정보의 내용을 직접 처리할 수 있는 어플리케이션을 구현하는데 활용될 수 있도록 설계된 언어이다. 즉, OWL은 풍부한 어휘와 형식적 의미론을 포함하고 있기 때문에 기계 해석을 가능하도록 하여 사람이 아닌 컴퓨터가 그 정보의 내용을 직접 처리할 수 있는 어플리케이션의 구현에 이용할 수 있는 언어이다[3].

OWL은 표현력이 서로 다른 세 개의 하위 언어, 즉 OWL Lite, OWL DL, OWL Full로 구성된다. 각 하위 언어는 서로 다른 개발자 및 사용자 층을 대상으로 한다[16].

OWL Lite는 OWL과 DAML+OIL의 공통적이고 유용한 부분들을 간추려 만들어졌고, 기능적인 면에서 볼 때 웹 응용프로그램을 지원하기 위해 간단하면서도 RDFS에 비해 풍부한 표현력을 가지는 언어로 볼 수 있다[15]. OWL DL은 Lite보다 좀 더 논리적인 표현을 위한 온톨로지 언어이다. OWL DL은 어휘의 사용에 있어 몇 가지 제한 사항을 포함한다. OWL Full은 OWL DL과 동일한 어휘로 구성되어 있으나, 표현력에 있어서 가장 풍부하며 RDF의 자유로운 구문을 모두 허용하고 있다[19]. 한편, OWL 온톨로지 정보를 데이터베이스로 변환하려는 관련 연구는 국내외에서 이제 막 시작 단계에 있다고 할 수 있다.

이민정은 특정한 도메인의 온톨로지에서 요구되는 질의 분석을 통하여 제공되어져야 하는 질의 모델을 정리하였고 이러한 모델을 기반으로 데이터베이스 상에서 상품 온톨로지 정보 검색 시스템을 구축하였다[5].

정진완과 린제시(Jiexi Lin)는 OWL에서 정의된 속성(property)을 지원하기 위한 효율적인 추론 알고리즘과 OWL 데이터 변경에 대한 효율적인 업데이트 방법을 제안하였다[7].

우은미는 XML 계층구조와 데이터베이스를 기반으로 OWL로 기술된 데이터에 대하여 OWL-QL 질의를 사용한 검색을 실험하였다[4].

OWL 데이터의 DB 저장에 관한 또 다른 관련 연구로서 A. Gali et. al.이 있다.[9] 이 연구에서는 XML 분해·저장 기법을 응용하여 OWL파일에 표현된 각 클래스(Class)별로 분해하여 속성명들을 컬럼으로 가진 테이블을 생성하고 인스턴스들을 이 테이블들에 저장한다. 즉 루트 클래스로부터 시작하여 깊이(depth)가 3이 될 때까지 파싱하고, 그 때까지 파싱된 클래스-서브클래스 데이터들을 테이블에 저장하며, 이 알고리듬을 계속하여 반복·적용하는 방법을 사용하였다.

이 연구에서 사용된 알고리듬은 XML 트리분해 기법을 응용하였으므로 실험적으로 클래스만을 대상으로 한다. 또한 각 클래스가 각각의 테이블로 표현되므로 테이블이 클래스 이름에 종속(dependent)되고, 클래스 수가 많아짐에 따라 테이블 수도 많아지게 되므로 쿼리처리 시에 조인(join) 연산에 상당한 부담을 주게 된다. 따라서 이러한 XML 분해응용 기법보다는 보다 일반적이고 확장된 OWL 데이터 저장기법이 필요하며, 본 연구에서는 OWL 구문들을 의미적으로 분석하여 Entity-Relationship (E-R) Diagram으로 표현하고 이를 테이블로 변환하는 개별 클래스에 독립적인(independent) 방식을 제안하였다.

그러므로 기존 연구에서는 특정 도메인에 한정되어 있다거나, 온톨로지 데이터가 갖고 있는 클래스, 프로퍼티, 인스턴스의 Triple 구조 중 속성(프로퍼티)에 한정적이며, XML 구조를 이용하여

OWL-QL 질의를 실험하는 등 사용자에게 친숙한 인터페이스 채택과 일반적인 관계형 데이터베이스의 사용이라는 면에서 제약을 갖고 있다고 하겠다.

본 연구에서는 OWL Lite에서 제공하는 기본적인 Typical Syntax에 기반을 두고, Class, Property, Instance 구문이 갖는 특징을 분석하여, 이를 기반으로 효율적인 OWL 데이터 저장을 위한 일반화된 관계형 모델을 설계하였다. 그리고 이를 적용하여 실제로 OWL Lite로 작성된 생물 온톨로지 데이터를 관계형 데이터베이스로 전환하였다. 기존 파일시스템과의 성능비교를 위해 기존의 추론기, 즉 Jena 기반 온톨로지 데이터 검색 시스템을 구축하여 두 시스템간의 동일조건 검사를 통해 검색 시간을 비교하였다.

4. 관계형 DB 변환 모델

4.1 설계전략

온톨로지 데이터를 데이터베이스화 하는 데는 상속(inheritance)이라는 개념과 클래스(Class), 프로퍼티(Property), 인스턴스(Instance)라는 Triple 구조 등으로 인해, 직접적인 데이터베이스 설계방법을 적용하기에는 많은 제한이 생긴다.

이에 본 연구에서는 온톨로지 데이터에서 요구되는 개체들을 파악하여, 이들 간의 함수적 종속(functional dependency)을 분석하고, 이를 사용하여 테이블들을 추출하여 나가는 전략을 이용하였다.

4.2 클래스 변환

클래스는 기본적으로 클래스 번호(CID)와 클래스 이름(C_Name)을 갖는다. 그리고 OWL Lite 구문 속에서 클래스 관련 Typical Syntax들이 갖는 특징을 분석해보면(<표 1> 참조), 특정 클래스는 다른 클래스들과 하위클래스(rdfs:subClassOf), 동치클래스(rdfs:equivalentClass), 및 교차클래스(owl:intersectionOf) 관계(relationship)들을 갖는다. (밑줄 참조)

이러한 관계들을 관계형 모델로 변환하면 동치관계의 경우, 동치클래스 번호(EquivalentNum)와 동치인 클래스(Equivalent_Value)들을 갖고, 하위클래스 관계의 경우 계층클래스 번호(CIDNum)와 하위클래스(SubClass_Value)들을 갖게 된다. 또한, 교차클래스 관계를 가질 경우 교차클래스 번호(IntersectionNum)와 교차할 클래스(Intersection_Value)들을 갖는다.

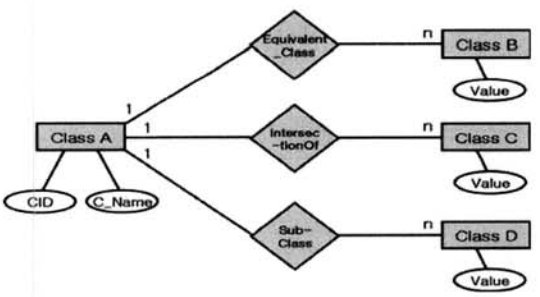
<표 1> 클래스 관련 OWL Typical Syntax

```
<owl:Class rdf:ID="SubClassName">
  <rdfs:subClassOf rdf:resource="#SuperClassName"/>
</owl:Class>
```

```
<owl:Class rdf:ID="class1">
  <rdfs:equivalentClass rdf:resource="class2"/>
</owl:Class>
```

```
<owl:Class rdf:ID="ClassName">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="AnotherClass1"/>
    <owl:Class rdf:about="AnotherClass2"/>
  </owl:intersectionOf>
</owl:Class>
```

이를 E-R Diagram으로 도식하였고(<그림 2> 참조), 필요한 Table들을 도출하였다.(<표 2> 참조)



<그림 2> 클래스 E-R Diagram

<표 2> 클래스 모델에서 추출된 Table

■ Class Table

CID	C_Name
Integer	String

■ Equivalent_Class Table

EquivalentNum	Equivalent_Value
Integer	Integer

■ Sub_Class Table

CIDNum	SubClass_Value

Integer	Integer

■ IntersectionOf Table

IntersectionNum	Intersection_Value
Integer	Integer

4.3 프로퍼티 변환

프로퍼티는 Property 번호(PID)와 Property 이름(P_Name), Property 타입(P_Type), Domain, Range, 속성을 갖는다. 그리고 OWL Lite 구문 속에서 프로퍼티 관련 Typical syntax들이 갖는 특징을 분석해보면(<표 3> 참조), 프로퍼티는 특정 인스턴스와 InverseOf, Transitive, Symmetric, Restriction 관계를 갖는다.

이러한 관계들을 관계형 모델로 변환하면 InverseOf 관계의 경우, InverseOf 번호(InverseOfNum)와 인스턴스(INID), Property(PID), 목적 인스턴스(INID2)의 속성을 갖게 된다.

Transitive 관계의 경우 Transitive 번호(TransitiveNum), 인스턴스(INID), 1차 목적 인스턴스(INID2), 2차 목적 인스턴스(INID3) 속성을 갖는다.

Symmetric 관계의 경우, Symmetric 번호(SymmetricNum), 인스턴스(INID), 목적 인스턴스(INID2) 속성을 갖는다.

Restriction 관계의 경우 Restriction 번호(RestrictionNum)와 Restriction Type(R_Type) 및 allValuesFrom, someValuesFrom, minCardinality, maxCardinality, cardinality에 관한 값(Value) 속성을 갖는다.

<표 3> 프로퍼티 관련 OWL Typical Syntax

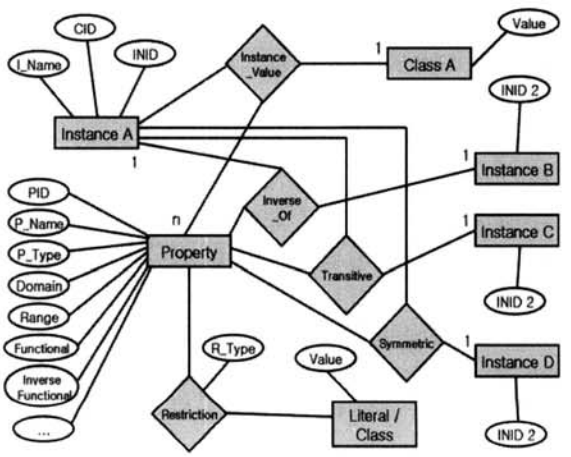
```
<owl:ObjectProperty rdf:ID="propertyName">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="property1">
  <owl:inverseOf rdf:resource="property2"/>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="propertyName"/>
```

```
<owl:SymmetricProperty rdf:ID="propertyName"/>
```

```
<owl:Restriction>
  <owl:onProperty rdf:resource="propertyName"/>
  <owl:allValuesFrom rdf:resource="targetClass"/>
  <owl:someValuesFrom rdf:resource="targetClass"/>
  <owl:minCardinality rdf:datatype="&xsd;
    nonNegativeInteger">cardinalityValue
  </owl:minCardinality>
  <owl:maxCardinality rdf:datatype="&xsd;
    nonNegativeInteger">cardinalityValue
  </owl:maxCardinality>
   <owl:cardinality rdf:datatype="&xsd;
    nonNegativeInteger">cardinalityValue
   </owl:cardinality>
</owl:Restriction>
```

이를 E-R Diagram으로 도식하였고(<그림 3> 참조), 필요한 Table들을 도출하였다.(<표 4> 참조)



<그림 3> 프로퍼티 E-R Diagram

<표 4> 프로퍼티 모델에서 추출된 Table

■ Property Table

PID	P_Name	P_Type	Domain	Range
Int	String	String	Integer	Integer

■ InverseOf Table

InverseOfNum	INID	PID	INID2
Integer	Integer	Integer	Integer

■ Symmetric Table

SymmetricNum	INID	INID2

Integer	Integer	Integer

■ Transitive Table

TransitiveNum	INID	INID2	INID3
Integer	Integer	Integer	Integer

■ Restriction Table

RID	R_Type	value
Integer	String	Integer

4.4 인스턴스 변환

인스턴스는 인스턴스 번호(INID), 인스턴스 이름(I_Name), 속한 클래스 이름(CID) 속성을 갖는다. 그리고 OWL Lite 구문 속에서 인스턴스 관련 Typical Syntax들이 갖는 특징을 분석해보면(<표 5> 참조), 특정 인스턴스는 특정 클래스에 속하며, 다른 인스턴스와 동치(owl:sameAS) 혹은 비동치(owl:differentFrom)의 관계, 그리고 Instance_Value 관계를 갖는다. (밑줄 참조)

이러한 관계들을 관계형 모델로 변환하면 동치 관계의 경우, 동치번호(sameAsNum)와 동치 인스턴스(sameAs_Value)를 갖게 되고, 비동치 관계의 경우, 비동치 번호(DifferentFromNum)와 비동치인 인스턴스(DifferentFrom_Value)를 갖게 된다.

또한, Instance_Value 관계를 가질 경우, Instance_Value 번호(INID)와 클래스 번호(CID), 속성(PID), 속성값(Value)을 갖는다.

<표 5> 인스턴스 관련 OWL Typical Syntax

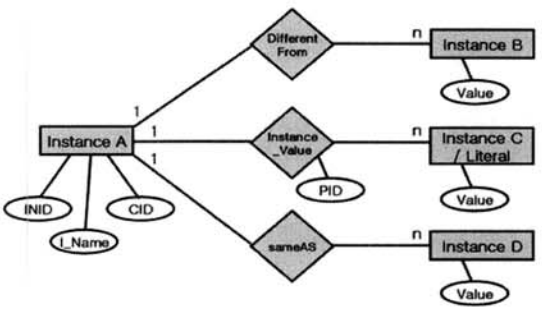
```
<ClassName rdf:ID="identifier">
    instanceProperties
</ClassName>
```

```
<ClassName
  rdf:about="subjectIndividualURIref">
  <owl:sameAs
    rdf:resource="equivIndividualURIref"/>
</ClassName>
```

```
<ClassName
  rdf:about="subjectIndividualURIref">
  <owl:differentFrom
    rdf:resource="differentIndividualURIref"/>
</ClassName>
```

```
<ClassName>
  rdf:about="subjectIndividualURIref">
  <owl:hasIndividuals
    rdf:resource="differentIndividualURIref"/>
</ClassName>
```

이를 E-R Diagram으로 도식하였고(<그림 4> 참조), 필요한 Table들을 도출하였다.(<표 6> 참조)



<그림 4> 인스턴스 E-R Diagram

<표 6> 인스턴스 모델에서 추출된 Table

■ Instance Table

INID	I_Name	CID
Integer	String	Integer

■ sameAs Table

sameAsNum	sameAS_Value
Integer	Integer

■ differentFrom Table

differentFromNum	differentFrom_Value
Integer	Integer

■ Instance_Value Table

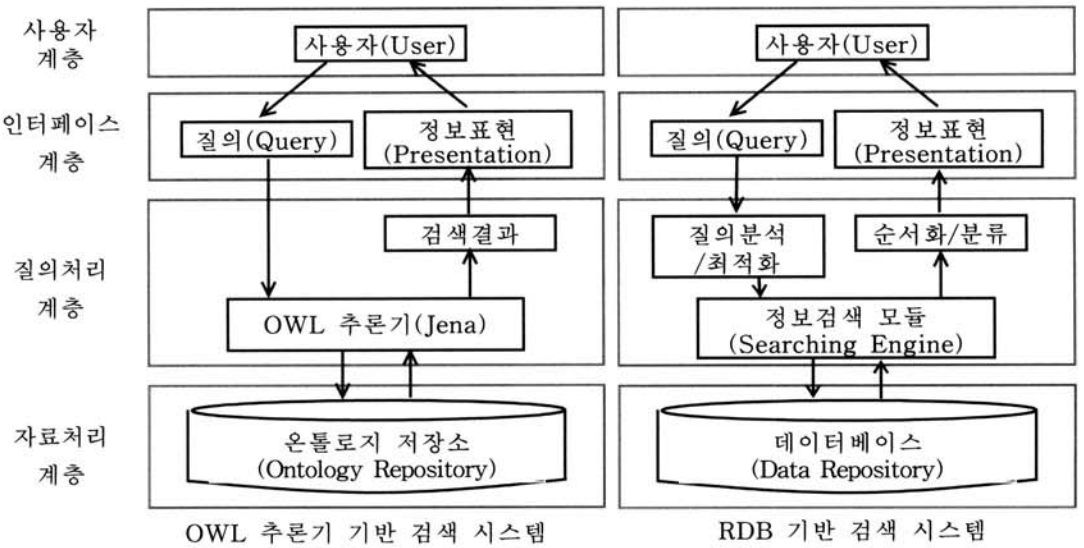
INID	CID	PID	Value
Integer	Integer	Integer	Integer

5. 온톨로지 데이터 검색 시스템

5.1 시스템 개요

관계형 데이터베이스에 기반한 OWL 데이터의 효율적인 변환을 검증하기 위해 본 연구에서는

기존의 OWL 추론기, 즉, 파일을 기반으로 한 OWL 데이터 검색 시스템과 본 연구에서 제안된



<그림 5> 전체 시스템 제어 흐름

데이터베이스 기반의 검색 시스템을 각각 구현하였다. 전체 시스템 개요는 <그림 5>와 같다.

① 사용자 계층 : 사용자가 탐색하길 희망하는 생물정보를 키워드로 입력하는 단계.

② 인터페이스 계층 : 사용자 입력 사항을 온톨로지나 관계형 DB 환경에 맞추어 변환하는 단계.

③ 질의처리 계층 : OWL 기반 검색 시스템의 경우는 사용자 질의를 분석하여 Jena 추론기를 통해 온톨로지 저장소를 접근하는 단계이고, RDB 기반 검색 시스템의 경우는 일반적인 관계형 정보검색 모듈을 이용하여 DB를 접근한다.

④ 자료처리 계층 : OWL 기반 검색 시스템의 경우는 생물 정보가 OWL 파일로 저장되어 있고, RDB 기반 검색 시스템의 경우는 테이블 형태로 생물 정보가 저장되어 있다.

결론적으로 추론기 기반의 파일 검색 시스템은 사용자가 질의한 사항을 Jena 기반의 추론기를 통해 분석하고, 이를 OWL 언어로 작성된 파일 형태의 온톨로지 저장소로부터 데이터를 추출하여 사용자에게 제공한다.

한편, RDB 기반 검색 시스템에서는 사용자의 질의에 대한 분석을 토대로 OWL 데이터가 저장된 데이터베이스로부터 해당 자료를 검색하여 사용자에게 제공한다.

본 연구에서 구현한 검색 시스템에서는 검색하는 방법을 다음의 세 가지로 분류하였다.

■ 클래스 검색

사용자가 입력한 검색어가 클래스 이름일 경우에 입력한 클래스의 하위 클래스들을 출력한다. 우선, 사용자가 입력한 클래스 이름이 온톨로지 구축과정에서 선정된 클래스에 존재하는지 유무를 확인한 후에, 존재한다면 입력한 클래스의 계층관계 클래스들을 출력한다.

■ 프로퍼티 검색

클래스의 속성(프로퍼티)으로 정의된 용어를 사용자가 검색어로 입력하는 경우의 검색방법이다. 우선, 사용자가 입력한 프로퍼티 이름이 온톨로지 구축과정에서 선정된 프로퍼티에 부합하는지 유무를 확인하고, 프로퍼티가 존재한다면 프로퍼티가 갖는 특징들이 출력된다.

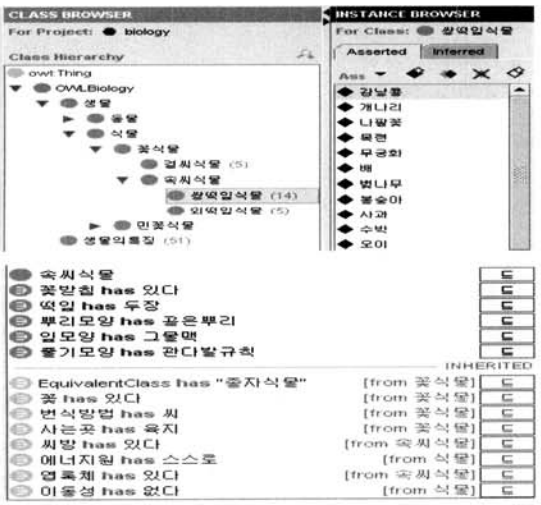
즉, 프로퍼티의 Type이 ObjectProperty인지 혹은 DatatypeProperty인지와 이 프로퍼티를 가질 수 있는 클래스의 범위를 나타내는 Domain, 그리고 이 프로퍼티의 값이 가질 수 있는 값의 범위인 Range값들이 출력된다. 또한, 사용자가 질의한 프로퍼티를 갖는 클래스들의 종류와 그 클래스가 갖는 Value값들이 함께 출력된다.

■ 인스턴스 검색

인스턴스로 정의된 용어를 사용자가 검색어로 입력하였을 경우의 검색방법이다. 마찬가지로 온톨로지 구축단계에서 정의된 인스턴스가 사용자가 질의한 인스턴스 이름과 일치하는지를 검사하고, 입력한 인스턴스 이름이 존재한다면, 인스턴스가 갖는 속성과 속성값, 그리고 입력한 인스턴스가 속하는 클래스 계층관계(hierarchy)를 출력한다.

5.2 테이블 저장

다음은 생물-식물-꽃식물-속씨식물-쌍떡잎식물-강낭콩의 생물 온톨로지 정보가 관계형 데이터 베이스 테이블에 어떻게 변환되어 저장되는지 살펴보도록 하겠다. 우선 온톨로지 개발 툴인 Protégé를 이용하여 강낭콩과 관련된 클래스, 프로퍼티, 인스턴스 들을 <그림 6>과 같이 개발한다.



<그림 6> 강낭콩 생물 온톨로지 정보

개발된 온톨로지 정보는 OWL 파일 형태로 <표 7>과 같이 저장된다.

<표 7> 강낭콩 OWL Syntax

```
<owl:Class rdf:ID="쌍떡잎식물">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#떡잎"/>
   </owl:onProperty>
   <owl:hasValue>
    <생물의특징 rdf:ID="두장"/>
   </owl:hasValue>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#꽃받침"/>
   </owl:onProperty>
   <owl:hasValue rdf:resource="#있다"/>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#잎모양"/>
   </owl:onProperty>
   <owl:hasValue>
    <생물의특징 rdf:ID="그물맥"/>
   </owl:hasValue>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#줄기모양"/>
   </owl:onProperty>
   <owl:hasValue rdf:resource="#관다발규칙"/>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#뿌리모양"/>
   </owl:onProperty>
   <owl:hasValue rdf:resource="#곧은뿌리"/>
  </owl:Restriction>
 </rdfs:subClassOf>
                < 중략 >
</owl:Class>
```

이 온톨로지 정보는 <그림 7>과 같은 테이블 형태의 DB에 저장된다.

IID	Instance Name	CID	PID 1	PID 2	PID 3	PID 4	PID 5	PID 6	PID 7	PID 8	PID 9	PID 10	PID 11	PID 12	PID 13	PID 14	PID 15	PID 16	PID 17
60	소나무	4	2	29	23	1	23	3	23	29			49		27			23	20
61	소철	4	2	29	23	1	23	3	23	29			49		27			23	20
62	은행나무	4	2	29	23	1	23	3	23	29			49		27			23	20
63	잣나무	4	2	29	23	1	23	3	23	29			49		27			23	20
64	전나무	4	2	29	23	1	23	3	23	29			49		27			23	20
65	강낭콩	6	2	29	29	5	29	3	12	29			49		27			23	20
66	개나리	6	2	29	29	5	29	3	12	29			49		27			23	20
67	나팔꽃	6	2	29	29	5	29	3	12	29			49		27			23	20
68	목련	6	2	29	29	5	29	3	12	29			49		27			23	20
69	무궁화	6	2	29	29	5	29	3	12	29			49		27			23	20
70	배	6	2	29	29	5	29	3	12	29			49		27			23	20
71	벚나무	6	2	29	29	5	29	3	12	29			49		27			23	20

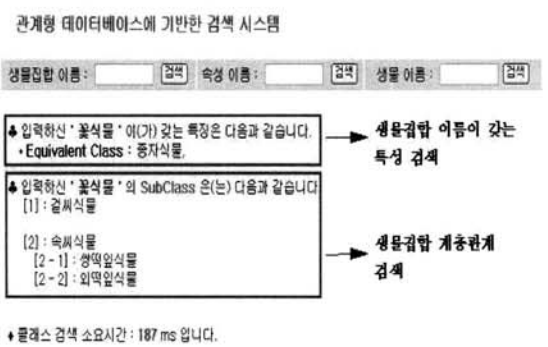
<그림 7> 관계형 DB 테이블

5.3 시스템 구현

■ 생물 집합(클래스)에 의한 검색

포유류, 강장동물, 척추동물, 거미류 등 생물의 계층 관계를 알기 위한 검색 방법이다. 즉, "척추동물은 포유류, 양서류, 파충류, 어류, 조류로 나누어진다."와 같이 생물의 계층적인 지식을 추출하기 위한 검색 방법이다.

<그림 8>에서 보는 바와 같이, 키워드로 입력한 생물집합 이름에 대해 OWL에 기술된 동의어와 해당 생물의 계층관계가 위계적으로 검색된다.



<그림 8> 생물 집합에 의한 검색

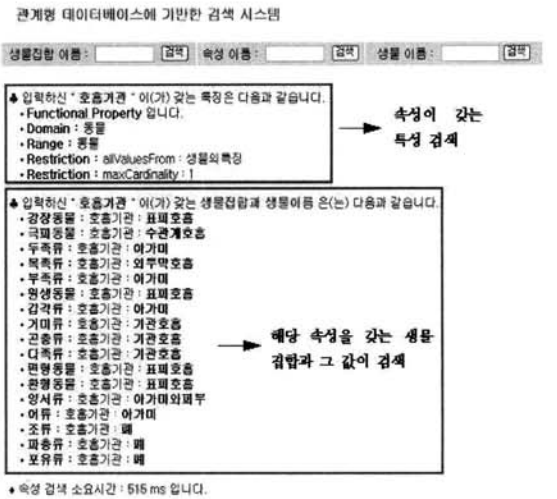
■ 생물 속성(프로퍼티)에 의한 검색

생물이 갖는 속성은 무엇이며, 그 속성이 어떠한 값을 갖는지를 검색하기 위한 방법이다. 즉, "포유류의 수정방법은 체내수정이다"와 같이 '수정방법' 이라는 속성을 갖는 생물과 그 속성에 대해 '체내수정'이라는 값을 갖는 정보를 추출하기 위한 검색 방법이다.

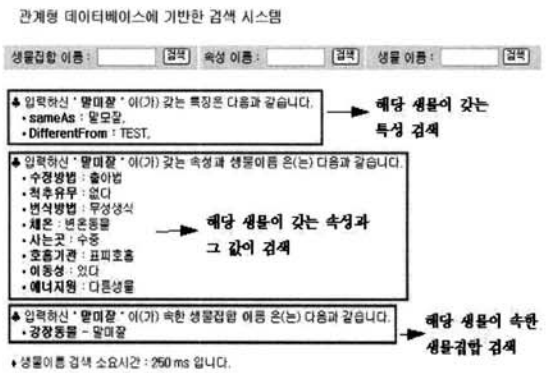
<그림 9>에서 보는 바와 같이, 키워드로 입력한 속성이름에 대해 OWL에 기술된 정보들이 나타나고, 그 속성과 관련된 생물이름과 속성값이 검색된다.

■ 생물 이름(인스턴스)에 의한 검색

특정 생물은 어떠한 생물집합에 속하며, 그 생물은 어떠한 속성과 속성 값을 갖는지를 검색하기 위한 방법이다. 즉, "호랑이는 포유류에 속하며, 호랑이가 갖는 특징으로 사는 곳은 육지이고, 호흡은 허파로 하며, 번식방법은 태생이다."와 같



<그림 9> 생물 속성에 의한 검색



<그림 10> 생물 이름에 의한 검색

이 '호랑이'라는 특정한 생물이름을 통해 '호랑이'가 갖는 속성들과 그 속성들에 대해 어떠한 값을 갖는지의 지식을 추출하기 위한 검색 방법이다.

<그림 10>에서 보는 바와 같이, 키워드로 입력한 생물이름에 대해 OWL에 기술된 정보들이 나타나고, 그 생물이름과 관련된 속성들과 속성값들이 검색된다.

6. 시스템 성능 평가

6.1 성능 평가 환경

OWL를 사용하여 작성한 생물 온톨로지 데이터에 대하여 파일 기반의 OWL 추론기 검색 시스템

과 데이터베이스 기반 검색 시스템에서 동일한 키워드를 입력하여 각각의 시스템에서 수행시간을 비교하였다. 시스템 개발 환경은 <표 8>과 같다.

<표 8> 시스템 개발 환경

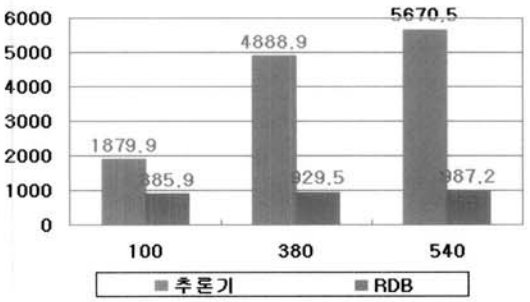
웹 서버	하드웨어	• PentiumIV PC
	웹서버 S/W	• Apache Tomcat 5.0.28
	웹 프로그래밍 언어	• JSP
온톨로지 구축도구	온톨로지 언어	• OWL (W3C 2004년 2월 Recommendation)
	온톨로지 개발 툴	• Protégé 3.1
추론기 검색모듈	프로그래밍 언어	• Java 1.5.0_01
	Inference Engine	• Jena 2.2
	프로그램 개발 툴	• JBuilder X
RDB 검색모듈	DBMS	• Mysql 5.0

키워드는 클래스 검색을 위해 '생물' 키워드를 입력하고, 프로퍼티 검색을 위해 '수정방법' 키워드를 입력하며, 인스턴스 검색을 위해 '말미잘' 키워드를 입력하여 비교하였다. 수행시간의 오차를 고려하여 각각의 시스템에 동일한 키워드를 12회 반복하여 입력하고, 최고값과 최저값을 제외한 나머지 10회의 수행시간의 평균을 비교대상으로 선정하였다. 또한, OWL 데이터의 증가에 따른 검색 성능을 점검하기 위해 생물 온톨로지 데이터의 크기를 증가시켜 실험하였다. 따라서, 생물 이름이 100개, 380개, 540개인 경우로 나누어서 각각의 검색성능을 비교하였다.

시스템 수행시간을 측정하기 위한 방법으로는 Benchmark Tag library 1.0을 사용하였으며, 이 library는 자바에 대한 개방형 소스 솔루션을 만들고 유지하는 Jakarta project의 일환으로 JSP 소스 코드내의 작업 수행 시간을 ms(mili-sec.) 단위로 반환하는 library이다[10].

6.2 클래스 검색 분석

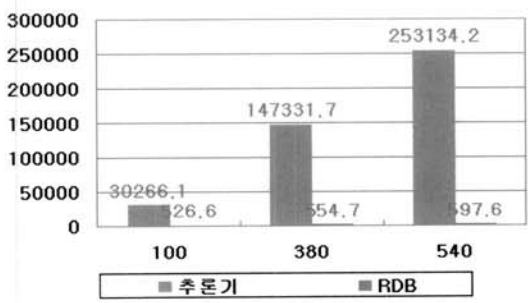
클래스 검색은 <그림 11>에서 보는 바와 같이, 동일한 검색 결과를 추출하는데 기존 추론기 기반 검색 시스템보다 데이터베이스 기반 검색 시스템의 검색 시간이 월등히 뛰어나며, 기존 추론기 기반 시스템은 OWL 데이터의 양이 증가함에 따라 검색 성능이 현저히 저하되고 있음을 알 수 있다. 이러한 부분에서 데이터베이스 기반의 OWL 데이터 검색 시스템의 필요성을 다시 한 번 인식 할 수 있다.



<그림 11> 클래스 검색 시간 비교

6.3 프로퍼티 검색 분석

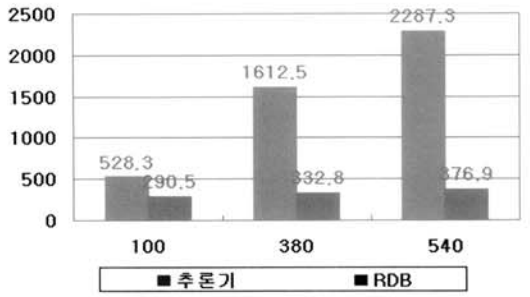
프로퍼티 검색은 <그림 12>에서 보는 바와 같이 클래스나 인스턴스 검색에 비해 기존의 추론기 검색 성능이 현저히 떨어지는 부분이다. 이는 기존 추론기가 제공하는 method의 한계 때문이다. 기존 추론기에서 제공하는 method는 특정 속성에 연관되는 클래스를 root 클래스부터 leaf 클래스까지 모두 검색해야 하고, 그 각각의 클래스의 인스턴스들을 모두 검색하여 특정 속성을 갖는지를 검색한다. 따라서, 인스턴스의 양이 증가함에 따라 고려해야 할 클래스의 숫자와 속성의 숫자가 급격히 증가하며, 따라서 검색 성능도 현저히 떨어지게 된다.



<그림 12> 프로퍼티 검색 시간 비교

6.4 인스턴스 검색 분석

인스턴스 검색도 <그림 13>에서 보는바와 같이 추론기 기반 검색 시스템은 데이터베이스 기반의 검색 시스템에 비해 검색 성능이 떨어졌고, OWL 데이터의 증가에 따라 검색 시간도 또한 크게 증가하고 있음을 알 수 있다.



<그림 13> 인스턴스 검색 시간 비교

7. 결 론

OWL 언어로 표현된 온톨로지를 사용하면 의미적 표현이 가능하고 개념 모델의 변화에 대해 유연하게 대처할 수 있으므로 OWL 추론기의 도움을 통해 지능적 검색과 지식 추론을 기대할 수 있다. 그러나 현재의 OWL 추론기들은 파일을 기반으로 처리되기 때문에, 대량의 데이터를 처리하기에는 데이터의 접근 속도와 성능을 보장할 수 없다는 제한점이 있다[6].

이에 본 연구에서는 OWL Lite에서 제공되는 구문들을 OWL Typical Syntax에 기반하여 관계형 모델로의 변환을 점검해 보았고, 기존 추론기 기반 검색 시스템과 설계한 데이터베이스에 기반한 검색 시스템을 각각 구현하여 동일한 키워드에 대해 검색 수행 시간을 비교하였다. 또한, 온톨로지 데이터 양의 증가에 따른 시스템 검색 성능을 비교하기 위해 검색의 대상이 되는 온톨로지 데이터의 양을 점점 증가시키며 검색 속도를 비교하였다.

시스템 성능 평가 결과, 클래스, 프로퍼티, 인스턴스의 동일한 결과물을 검색하는데 있어서 데이터베이스에 기반한 검색시스템의 성능이 뛰어났으며, 기존 추론기를 기반으로 하는 검색 시스템은 파일을 기반으로 처리되는 특성상 온톨로지 데이터의 양이 증가함에 따라 검색 수행시간이 급격히 증가함을 알 수 있었고, 데이터베이스를 기반으로 하는 검색 시스템은 OWL 데이터의 양의 증가에 따른 검색 수행시간의 변화가 크지 않았다.

따라서 본 연구를 통해 대량의 데이터를 처리할 수 있으며, 성능이 보장된 관계형 데이터베이스를 이용하여 기본적인 OWL 기반 온톨로지 정보를 저장, 검색함으로써 검색 성능을 향상시킬 수 있다고 사료된다. 한편으로는 OWL Lite 구문을 넘어서는 OWL DL과 OWL Full에서 제공하는 지능적 검색과 지식 추론에 대한 추후의 연구가 필요하다고 하겠다.

참 고 문 헌

[1] 고메즈 페레즈(Gómez-Pérez 외), Ontological Engineering, Springer, pp. 1-197, 2004

[2] 김현주, 지식기반 교수 학습을 위한 온톨로지 적용 검색 시스템 설계 및 구현, 한국교원대학교 석사학위논문, 2005

[3] 박현근, OWL 시멘틱 웹 기반 온톨로지 상에서의 규칙-사실 생성에 관한 추론, 중앙대학교 석사학위논문, 2004

[4] 우은미, 효율적인 OWL 질의 처리를 위한 저장 스키마 구축 방법, 한국과학기술원 석사학위논문, 2004

[5] 이민정, 상품 온톨로지 모델과 관계형 데이터베이스로의 설계 및 구현, 숙명여자대학교 석서학위논문, 2005

[6] 정복문, 온톨로지 추론 기반의 논문 검색 시스템 설계 및 구현, 한국교원대학교 석사학위논문, 2006

[7] 정진완, 린제시, Reasoning with OWL Properties based on Relational Databases, 데이터베이스연구, 22(2), pp. 17-32, 2006

[8] 최중민, 시멘틱 웹의 개요와 연구동향, 정보과학회지 21(3), pp. 4-10, 2003

[9] A. Gali, C. Chen, K. Claypool, R. Uceda-Sosa, From Ontology to Relational Databases, Lecture Notes in Computer

Science, Vol-No.3289, pp. 279-289, 2004.
[10] Bayern, S., Jakarta Project Benchmark Tag Library 1.0, [online] available: <http://jakarta.apache.org/taglibs/doc /benchmark-doc/intro.html>
[11] Berners-Lee, T., Semantic Web-XML, [on line] available: <http://www.w3.org/2000/Talks/ 1206-xml2k-tbl>
[12] Berners-Lee, T., Hendler, J., & Lassila, O., The Semantic Web, Scientific American, 2001
[13] Cost, R. S., et. al., ITTALKS: A Case Study in the Semantic Web and DAML+OIL, IEEE Intelligent System, 17(1), pp. 40-47, 2002
[14] Gruber, T., A translation approach to portable ontologies, Knowledge Acquisition, 5(2), pp. 199-220, 1993
[15] McGuinness, D. L., & Harmelen, F. V., Feature Synopsis for OWL lite and OWL W3C working draft 29, [on line] available: <http://www.w3.org/TR/2002/WD-owl-features-20020729/>
[16] McGuiness, D. L., Deborah & Harmelen, van, Frank, OWL Web Ontology Language Overview, [online] available: <http://www/w3.org/TR/owl-features/>
[17] _____ , Jena Documentation, [online] available: <http://jena.sourceforge.net/documentation.html>.
[18] Noy, N. F., & McGuiness, D. L., Ontology Development 101: A Guide to Creating Your First Ontology, [on line] available: <http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ ontology101-noy-mcguinness.html>
[19] W3C, OWL Web Ontology Language Overview, [online] available: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

조 성 환

1997 춘천교육대학교 (교육학학사)
2007 한국교육대학교 컴퓨터교육과(교육학석사)
2007～현재 한국교원대학교 컴퓨터교육과 박사과정
관심분야 : 컴퓨터 교육, 정보통신윤리 교육, 프로그래밍 교육
E-Mail : 74csh@hanmail.net

김 성 식

1977 고려대학교 경영학과 (경영학사)
1986 미국 카톨릭대학교 전산학과(이학사)
1988 오리곤 주립대학교 전산학과(이학석사)
1992 고려대학교 컴퓨터과학과(이학박사)
1992～현재 한국교원대학교 컴퓨터교육과 교수
관심분야: 컴퓨터교육, 원격 교육, 정보통신윤리교육
E-Mail : seongkim@knue.ac.kr

김 태 영

1985 한양대학교 산업공학과
1990 Texas A&M University 컴퓨터과학과 (공학석사)
1994 Texas A&M University 컴퓨터과학과 (공학박사)
1994～1994. 8. 삼성 SDS(주) 정보기술 연구소 선임연구원
1994～현재 한국교원대학교 컴퓨터교육과 교수
관심분야: 데이터베이스, 지식처리, 컴퓨터교육
E-Mail : tykim@knue.ac.kr