내장형 시스템에서 DAML-S 서비스 지원을 위한 효율적인 DAML+OIL 문서 관리 시스템

(Development of the Efficient DAML+OIL Document Management System to support the DAML-S Services in the Embedded Systems)

김 학 수 † 정 문 영 †† 차 현 석 †† 손 진 현 †††
(Hag Soo Kim) (Moon-young Jung) (Hyun Seok Cha) (Jin Hyun Son)

요 약 최근에 시멘틱 웹에 대한 관심과 기대가 증가함에 따라 시멘틱 웹 기반의 웹서비스인 시멘틱 웹서비스에 대한 연구가 활발히 진행되고 있다. 기존의 웹서비스는 XML 기반의 웹서비스 기술 언어인 WSDL을 사용하지만 시멘틱 웹서비스에서는 DAML-S와 같은 온톨로지 언어기반의 웹서비스 기술 언어를 사용한다. 시멘틱 웹서비스에 대한 연구는 웹서비스 검색, 웹서비스 구동, 웹서비스 구성, 웹서비스 실행 모니터링 등의 관점에서 수행되고 있다. 특히, 시멘틱 웹서비스 검색은 시멘틱 웹서비스의 궁극적인 목적을 달성하기 위한 가장 기반이 되는 분야로 기존의 정보 검색과는 다른 특징들을 가지고 있다. 즉, 시멘틱 웹 기술 언어에 적합한 저장 시스템과 검색 방법이 요구된다. 이에 부합하여 관련된 시스템이 많이 개발되고 있으나 지능로봇 응용과 같은 제한된 메모리 공간 및 디스크 공간을 가지는 임베디드 환경에서 기존의 시스템을 적용하기에는 부적합하다. 이와 관련하여, 이 논문에서는 임베디드 시스템 환경에서 시멘틱 웹서비스 검색에 이용될 수 있도록 하기 위해서 DAML-S 서비스 지원을 위한 효율적인 DAML+OIL 문서 관리 시스템을 개발하였다. 또한, 우리는 이 논문에서 소개한 시스템의 특성을 다른 시스템과 비교하여 기술한다.

키워드 : 시멘틱 웹, 시멘틱 웹서비스, 온톨로지, DAML+OIL, DAML-S

Abstract Recently, many researchers have given high attention to the semantic web services based on the semantic web technology. While existing web services use the XML-based web service description language, WSDL, semantic web services are utilizing web service description languages such as DAML-S in ontology languages. The researchers of semantic web services are generally focused on web service discovery, web service invocation, web service selection and composition, and web service execution monitoring. Especially, the semantic web service discovery as the basis to accomplish the ultimate semantic web service environment has some different properties from previous information discovery areas. Hence, it is necessary to develop the storage system and discovery mechanism appropriate to the semantic web description languages. Even though some related systems have been developed, they are not appropriate for the embedded system environment, such as intelligent robotics, in which there are some limitations on memory disk space, and computing power. In this regard, we in the embedded system environment have developed the document management system which efficiently manages the web service documents described by DAML-S for the purpose of the semantic web service discovery, In addition, we address the distinguishing characteristics of the system developed in this paper, compared with the related researches.

Key words : Semantic web, Semantic web service, ontology, DAML+OIL, DAML-S

· 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2004-003-D00327)
† 학생회원 : 한양대학교 컴퓨터공학과
hagsoo@cse.hanyang.ac.kr
†† 비 회 원 : 한양대학교 컴퓨터공학과
myjeong@cse.hanyang.ac.kr
hscha@cse.hanyang.ac.kr
††† 종신회원 : 한양대학교 컴퓨터공학과 교수
jhson@cse.hanyang.ac.kr
논문접수 : 2004년 4월 8일
심사완료 : 2004년 10월 15일

1. 서 론

1989년에 Tim Berners-Lee에 의해 제안된 월드 와이드 웹은 현재 널리 알려진 클라이언트 - 서버 개념과 누구나 쉽게 익힐 수 있는 HTML 언어를 이용하여 편리성을 추구한 덕분에 일반 사용자 누구나 쉽게 정보를 접근하거나 게시할 수 있게 되었고, 결과적으로 폭발적인 정보의 증가를 가져왔다. 사용자들은 단순히 URL을 입력하거나 하이퍼링크를 따라가면서 원하는 정보를 찾거나 검색엔진의 도움으로 정보를 찾는다. 이러한 단순함이 현재 웹의 성장을 가져온 중요한 열쇠이다. 그러나 폭발적인 정보의 증가와 더불어 사용자가 원하는 정보를 찾는 것은 어렵게 되었다. 지금의 검색엔진은 단순히 단어 매칭을 통한 빈도수, 확률을 계산하여 문서의 랭킹을 매겨서 사용자에게 보여준다. 이것은 HTML의 단점을 확연히 보여주는 예이다. 왜냐하면, HTML에서는 단순히 디스플레이, 레이아웃 기술로 설계되었기 때문에 단어의 의미를 표현하는데 있어서 한계를 드러내고 있다. 즉, HTML이 가지고 있는 한계점으로 인해 더 이상의 기능적인 성장을 기대하기 어려운 상황이다. 또한, 사람뿐만 아니라 소프트웨어, 에이전트가 자동으로 정보를 추출하는 것 또한 어렵다. 이에 지금의 웹보다 더 많은 기능성과 상호 운용성을 지닌 새로운 웹인 시멘틱 웹이 등장하게 되었다. 시멘틱 웹[1-4]은 전혀 다른 웹이 아닌 지금의 웹의 확장이며 전체를 하나의 통합된 형태로 바라보고 있는 시멘틱 웹은 자연히 지금의 웹이 갖고 있는 많은 요소들을 하나로 통합시키고 있다. 시멘틱 웹의 궁극적인 목적은 컴퓨터가 웹에 있는 정보를 이해할 수 있도록 도와주는 표준과 기술을 개발하여 시멘틱 검색, 시멘틱 웹서비스, 데이타 통합, 네비게이션, 작업의 자동화 등을 지원하는 것이다. 이 논문에서는 특히, 지금의 웹상에서 이루어지던 웹서비스를 시멘틱 웹으로 통합시킨 시멘틱 웹서비스의 저장시스템에 대해서 다룬다. 그것을 다루기 전에, 시멘틱 웹을 실현하기 위한 기반요소들을 먼저 알아보겠다.

시멘틱 웹에서는 웹을 하나의 커다란 유기체로 바라본다. 그리고 웹 문서들은 이 커다란 유기체의 일부분을 기술하게 된다. 문서가 기술하는 것은 웹이 가지고 있는 지식일 수도 있고, 웹이 가지고 있는 기능일 수도 있다. 웹 문서가 기술하는 내용은 지금의 웹과 같이 문서 내부에 한정되지 않고 전체 시멘틱 웹을 대상으로 하며 어디에서든지 참조, 이용할 수 있다. 이런 시멘틱 웹과 지금의 웹의 가장 큰 차이점 중의 하나는 웹을 사용하는 주체가 사람뿐만 아니라 기계도 포함된다는 사실이다. 즉, 에이전트가 능동적으로 웹을 사용할 수 있게 한다. 사람이 보는 데에 최적화된 기존의 웹과는 달리 시멘틱 웹은 웹에 존재하는 지식을 기계가 이해하고 이용할 수 있도록 만들어지며 이를 위해 온톨로지라는 개념을 사용하고 있다.

온톨로지[5,6]는 시멘틱 웹에서 사람과 기계 모두가 동일하게 해석하는 개념의 집합이다. 시멘틱 웹 기술 언어에서 사용하는 단어나 문법들은 모두 정해진 규격에 따라 만들어지며, 나타내고자 하는 사실이나 지식, 표현하고 싶은 기능에 대해서 공통된 개념을 이용하여 표현할 수 있게 된다. 이런 공통된 개념은 기계도 사람이 이해하는 방식과 같은 의미로 해석할 수 있게 만들어 준다. 온톨로지는 과거에 철학분야에만 국한되어 사용되었으나 근래에는 컴퓨터 공학 분야에 적용되어 널리 사용되고 있다. 특히 최근에는 지식공학, 지식표현, 데이타베이스 디자인, 정보 모델링, 정보통합/관리/조직, 에이전트 기반 시스템 등 다양한 분야에 적용되고 있다. 인공지능에 있어서 온톨로지는 "개념화의 명세(specification of a conceptualization)"로 정의된다. 이는 "공학 인공물(engineering artifact)"로써 어떤 사실을 기술하기 위해 필요한 객체(object)의 집합인 "단어(vocabulary)"와 이의 객체들 간의 관계(relation)와 기능(function)들의 집합으로 이루어진다. 다른 말로표현하면 온톨로지는 객체의 집합과 객체들 간의 관계의 정의에 어떤 사실이나 상태를 표현하고자 하는 지식 표현 기법이다. 이러한 온톨로지 개념을 적용하는 시멘틱 웹이 발전하기 위해서는 RDF[7]와 같은 기술로 모델링 되어야 하는데, 온톨로지를 모델링 하기에는 RDF 자체만으로는 의미적인 면을 표현하는 데 있어서 한계성이 나타나고 있다. 그래서 RDF 기반으로 확장된 언어들이 개발되고 있다.

기존의 웹에서 HTML로는 표현할 수 없었던 시멘틱 웹의 기능성과 통합성 때문에 새로운 시멘틱 웹 기술 언어가 만들어졌다. 시멘틱 웹이 제시되기 이전부터 이질적인 환경에서 상호 운용성을 위한 데이터형식으로 XML[8]이 사용되었고, 시멘틱 웹 기술 언어도 이 XML을 기반으로 만들어졌다. 그림 1은 XML을 기반으로 하는 시멘틱 웹 언어의 계층구조를 보여주고 있다. 이 계층구조에서 보면 가장 하위 레벨에서는 웹 프로토콜에서 자원을 지칭하기 위한 주소지정(addressing) 방법인 URI가 밑받침되고 이를 기반으로 XML과 Namespace, RDF와 RDF 스키마, 온톨로지의 순서로 연구가 진행되고 있으며 그 위의 계층인 Logic에 대해서는 인공지능의 추론연구를 밑받침으로 일부 연구가 시작되었다. 또한 보다 더 상위 계층인 Proof와 Trust는 시멘틱 웹 정보의 신뢰성과 보안에 관한 내용으로서 아직 개념 정도만 얘기되고 있으며 차후 연구과제로 제시되고 있다. 각 계층은 시멘틱 웹 기술 언어가 가지고 있는 표현력과

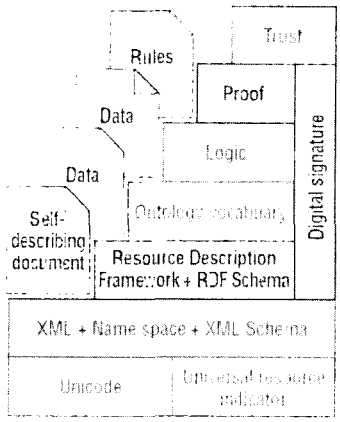

그림 1 시멘틱 웹 계층 구조

기능성에 의해 나뉘어 진다. 그리고 상위 언어들은 하위 언어들을 기반으로 정의되고 만들어져 있다. 시멘틱 웹 언어 계층에서는 문서의 범위를 제한하지 않는다. 즉, 문서의 위치와 크기에 관계없이 시멘틱 웹 전체를 대상으로 시멘틱 웹의 요소를 표현하고 설명할 수 있다. 대표적인 시멘틱 웹 기술 언어는 RDF[7]와 RDF Schema[9], 그리고 DAML+OIL[11]이 있다.

시멘틱 웹서비스는 기존의 WSDL이 갖고 있던 문제점을 개선하고 좀더 의미 있는 서비스 기술을 가능케 하며 WSDL 대신에 DAML-S를 사용한다. 이러한 시멘틱 웹서비스 기술언어를 통해 수많은 회사에서 자신들의 서비스를 제공하고, 그 사실을 알리기 위해서 시멘틱 웹서비스 문서를 발간했을 때, 자연히 사용자는 검색 엔진을 사용하여 비즈니스 정보를 검색하게 될 것이다. 이와 관련하여 사용자가 원하는 서비스, 사용자가 만족할 수 있는 서비스를 찾고 결합시켜주는 검색 엔진이 필요하게 된다. 시멘틱 웹 정보 검색은 기존의 정보 검색과는 다른 형태의 검색이 될 것이며, 기존의 문서 내 단어의 존재 유무나 빈도수와 같은 휴리스틱한 방법을 적용하기는 매우 어렵다. 그래서 시멘틱 웹 기술 언어에 알맞은 새로운 검색 방법이 필요하며, 그러한 시멘틱 웹 기술 언어를 저장할 수 있는 저장 시스템이 필요하다. 더 나아가 임베디드 환경에서의 저장 시스템도 중요시 되고 있다. 특히 로봇 시스템 환경을 예로 들 수 있는데 실버타운에서 홀로 거주하는 노인을 돌보는 로봇이 있다고 가정하자. 이 로봇은 노인을 돌보기 위해서 청소, 노인의 건강상태 검사, 노인이 지시하는 명령 등을 수행할 의무가 있다. 이러한 작업을 하기 위해서 로봇은 실버타운의 건물의 구조, 건물 안에 있는 가구, 전자제품, 노인의 위치 등을 모두 알고 있어야만 한다. 그래서 DAML+OIL를 이용하여 환경에 대한 온톨로지, 사물에 대한 온톨로지 등을 기술할 필요가 있다. 이렇게 기술된 문서는 로봇이 저장하고 있어야 하는데, 문제점은 로봇이라는 임베디드 시스템은 제한된 메모리 공간, 디스크 공간으로 인하여 효율적인 데이타 관리를 필요로 한다는 것이다. 그러나 기존의 시스템이 데이타 관리에 있어서 비효율적이기 때문에 임베디드 시스템 환경에 적용시키기에 어려움이 있다. 그래서 이 논문에서는 효율적인 데이타 관리에 초점을 맞추어서 임베디드 환경에서 DAML+OIL문서의 효율적인 저장을 위한 데이타베이스 스키마를 제안하고 문서 관리 시스템을 개발하였다.

논문의 구성은 다음과 같다. 시멘틱 웹 언어인 RDF, RDFS, DAML+OIL과 시멘틱 웹서비스를 위한 온톨로지 기술 언어인 DAML-S, 시멘틱 웹 언어와 시멘틱 웹 기술 언어에 대한 개념, 그리고 시멘틱 웹 언어의 저장 기법 및 저장시스템에 대한 관련 연구를 2장에서 소개한다. 3장에서는 Jena, Sesame+Bor의 데이타베이스 스키마를 분석하고 그것을 기반으로 임베디드 환경에서 효율적인 데이타 관리를 위한 새로운 데이타베이스 스키마를 4장에서 제안한다. 5장에서는 제안한 데이타베이스 스키마를 실제로 구현한 시스템에 대해서 알아보고 6장에서는 구현된 시스템에 대해서 Sesame+BOR와의 비교 분석을 한다. 마지막으로 7장에서는 제안된 데이타베이스 스키마에 대한 결론으로 마무리한다.

2. 관련 연구

2장에서는 시멘틱 웹 언어(RDF, RDFS, DAML+OIL)에 대해서 알아보고 DAML+OIL를 기반으로 하는 시멘틱 웹서비스 기술 언어인 DAMLS[5,18]를 알아본다. 또한 기존의 시멘틱 웹 언어의 저장기법 및 저장 시스템을 분석한다. 이것은 이 논문에서 다룰 DAML-S 서비스 지원을 위한 DAML+OIL 문서 저장 시스템에 대한 기본 지식이 된다.

2.1 RDF와 RDF Schema

RDF(Resource Description Framework)는 웹상에 존재하는 자원을 기술하기 위해 만들어졌다. 기본적으로 XML을 기반으로 하고 있으며 자원에 관한 내용을 주어(Subject), 술어(Predicate), 목적어(Object)의 조합 형태로 표현한다. 이를 통해 RDF는 기계가 처리할 수 있는 데이타의 의미를 기술하는 표준을 제공한다.

RDF는 단지 웹상에 존재하는 자원을 기술하기 위한 최소한의 틀만을 제공한다. 그러나 RDF 상에 존재하는 자원과 자원 사이의 관계 또한 RDF의 관점에서는 자원이다. 이에 RDF Schema에서는 RDF에 객체 지향적인 개념을 도입해서 웹상의 자원을 클래스와 인스턴스로 구분한다. 그리고 클래스의 일부를 클래스와 클래스 사

이의 관계를 표현하는 속성으로 지정할 수 있게 만들어졌다.

2.2 DAML+OIL

DAML(DARPA Agent Markup Language, [12])은 DARPA(Defense Advanced Research Project Agency)에서 에이전트간의 지식을 교환하고 공유하기 위해서 개발한 언어였다. 이 DAML에 온톨로지의 개념을 가지고 있는 OIL(ontology inference layer)를 통합시킨 것이 DAML+OIL이다. DAML+OIL는 RDF Schema를 기반으로 하고 있으며, RDF Schema에는 존재하지 않는 논리적 관계와 제약 사항 등을 기술할 수 있다. RDF Schema는 클래스와 인스턴스에 관련된 사항을 기술하기 위해 만들어졌다면 DAML+OIL은 그러한 클래스가 어떤 특징을 가지고 있는가를 좀더 세밀하게 기술할 수 있다. 즉, DAML+OIL은 RDF Schema가 논리적인 액시엄(axiom)을 기술하는 표준의 결핍을 보완하기 위해 개발된 시멘틱 웹 언어이다.

2.3 DAML-S

DAML-S[13,14]는 DAML+OIL을 바탕으로 하는 상위의 계층이며, 실제 세계에서 소프트웨어 에이전트들이 필요한 서비스에 대한 추론을 통해 미리 사용자의 의도를 파악해서 서비스를 발견, 선택하도록 서비스를 기술하는 것이 목적이다.

DAML-S에 의해 가능한 자동화는 다음과 같이 네 가지로 나눌 수 있다. 첫 번째, 웹서비스 탐색이다. 이것은 상품들을 서울로 보내줄 운송 서비스를 찾는 것이다. 두 번째는 웹서비스 실행이다. 사용자는 찾은 웹서비스에 접속하여 www.acmemoo.com로부터 우유 500lbs를 산다. 이를 통해 발견된 서비스로 거래를 시작한다. 세 번째는 에이전트에 의해 이루어지는 웹서비스 선택과 구성이다. 에이전트가 서울에서 2 주 동안 500명의 사람들에게 보낼 수 있는 음식을 준비하기 위해 여러 서비스를 선택하고 이 서비스는 여러 절차로 구성될 수 있는 데 이러한 절차를 자동으로 구성하게 된다. 네 번째는 웹서비스 실행과 모니터링이다. 우유가 주문되었고, 대금이 지불되었는지 여부를 체크하고 모니터링 한다. 이러한 자동화를 가능케 하기 위해 DAML-S는 그림 2

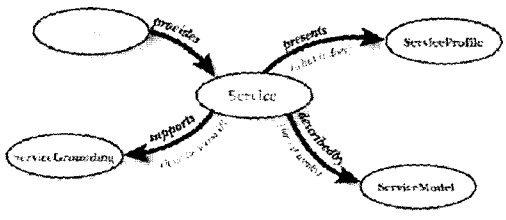

그림 2 Upper ontology of DAML Services

와 같은 Upper 온톨로지를 가지며, 서비스를 Profile, Process, Grounding의 세 가지 모델로 기술하게 된다.

2.4 RDF(S), DAML의 저장기법에 대한 연구

여기에서는 본 논문의 DAML+OIL 저장 시스템의 토대로서 RDF, DAML+OIL과 관련된 저장기법에 대한 연구에 대해서 알아본다. 기본적으로 DAML+OIL, DAML-S는 RDF를 기반으로 확장된 언어이기 때문에 RDF가 저장될 수 있는 데이타베이스 스키마는 RDF 기반의 다른 언어들도 저장할 수 있다. 대표적으로 Sesame[15], Jena[16] 같은 경우에는 RDF가 가지고 있는 특성을 그대로 데이타베이스 스키마화하고 있다. RDF는 리소스를 기술하기 위한 프레임워크로서 RDF 그래프로 나타낼 수 있으며 이 그래프는 트리플에 의해 표현될 수 있다. 즉, Sesame, Jena는 트리플, 리소스를 데이타베이스 스키마화 함으로써 RDF문서를 저장하고 있다. 이 외에 논문[17]에서도 Sesame, Jena와 비슷한 모델을 형성하고 있다. 저장기법에 대한 다른 접근으로 논문[18]에서는 RDF문서에 있는 클래스, 속성들을 전부 스키마화 하는 방식을 취하고 있다. 그러나 이 같은 디자인의 경우 작은 시스템에서는 효율적으로 문서를 관리할 수 있지만 대용량의 문서 관리 시스템에서는 효율성이 현격하게 떨어지는 단점이 있다.

위의 저장 모델 같은 경우에는 기존의 관계 데이타베이스를 그대로 사용하는 기법들이다. 그러나 논문 [19]에서는 기존의 관계 데이타베이스를 사용하지 않고 RDF 저장 모델을 위한 데이타베이스를 설계하는 방식을 취하고 있다. 이 모델의 가장 큰 장점은 저장 시스템이 가볍고, 효율적이라는 것이다. 그러나 시스템을 확장하기 어렵다는 것이 단점이다. 또한 DAML+OIL를 저장하기 위해서는 새롭게 설계 해야 되기 때문에 확장성이 떨어진다.

앞부분에서는 RDF와 관련된 저장 모델에 대한 연구를 간략히 알아보았다. 이러한 모델을 기반으로 해서 여러 가지 저장 시스템이 구현되었다. 연구가 가장 활발히 진행되고 있는 시스템으로는 Jena(시멘틱 웹 온톨로지 저장을 위한 저장 시스템 개발을 위한 툴킷), Sesame, Parka[20], TAP[21] 등이 있다. Jena, Sesame에 대해서는 3장에서 자세히 알아보도록 한다. Parka는 전통적인 관계 데이타베이스를 기반으로 하여 웹 온톨로지 문서를 저장하고 관리하며 질의할 수 있는 시스템이다. 그러나 문서 저장에서 약 250만개의 트리플만 저장할 수 있는 제한을 가지고 있어서 방대한 문서관리에는 부적합하다. TAP은 Parka처럼 관계 데이타베이스 기반이며 아파치 모듈을 제공함으로써 웹으로 접근할 수 있게 하고 있다. 그러나 이 시스템의 단점은 RDQL 같은 질의 인터페이스를 통한 그래프 매칭 메커니즘을 제공하지

않는 다는 것이다. 단순히 GetData라는 함수를 통해서 트리플 매칭만을 제공하고 있다.

지금까지 RDF, DAML+OIL에 대한 저장 모델, 저장 시스템에 대해서 알아보았다. 그 중에서 Sesame, Jena가 가장 활발히 연구가 진행 중에 있으며, 저장 방식 및 저장 모델에 있어서 가장 높은 완성도를 보이고 있다. 그래서 이 논문에서는 이 두개 시스템의 분석을 통해서 임베디드 환경에서 효율적인 문서 관리를 위한 시스템을 설계하고자 한다.

3. 기존 시스템에 대한 분석(Jena, Sesame)

3장에서는 RDF, RDFS, DAML+OIL문서를 저장하기 위한 시스템으로 가장 활발히 연구가 진행되고 있는 Jena, Sesame에 대해서 알아본다. 이러한 시스템을 분석함으로써 본 논문에서 설계하고자 하는 DAML-S 서비스 지원을 위한 DAML+OIL 문서 저장 시스템의 토대를 이룰 것이다.

3.1 Jena

이 논문에서는 Jena 1.6.1 버전에 대해서 다룬다. Jena는 HP에서 만든 시멘틱 웹 애플리케이션 빌딩을 위한 자바 프레임워크이다. 즉, 개발자가 Jena API를 사용하여 시멘틱 웹과 관련된 상위의 애플리케이션을 개발할 수 있도록 패키지 식으로 모듈화가 되어 있다. 기본적으로 Jena는 시멘틱 웹 언어(RDF, RDFS, DAML+OIL)로 작성된 문서를 파싱하고 문장을 검사하는 API를 지원하며 시멘틱 웹 언어로 작성된 문서를 저장하기 위해 메모리 스토리지와 데이타베이스(버클리, MySQL 등) API를 지원하고 있다. 이러한 API를 통해서 저장된 문서에 대한 검색을 위해 RDQL[22]이라는 질의 언어를 기반으로 한 API를 제공한다. 즉, Jena는 이러한 기능이 패키지화되어 있어서 상위의 시멘틱 웹 애플리케이션을 개발하는 데 편리함을 제공해 주는 툴킷이다.

다음은 Jena 1.6.1(이하 Jena)의 아키텍처에 대해 알아보겠다. Jena의 아키텍처는 그림 3과 같다.

Jena는 여러 종류의 시멘틱 웹 언어를 처리할 수 있다. 그림 3의 아키텍처는 시멘틱 웹 문서를 데이타베이스에 저장하고, 저장된 문서를 어떻게 질의하는 지를 보여주는 간단한 아키텍처 그림이다. RDF, DAML+OIL 문서를 저장할 때 먼저 그 문서들을 모델화(메모리 상에 생성되는 방향성 그래프)시킨다. 그런 다음 모델화 시킨 것을 데이타베이스에 저장한다. 사용자가 질의 할 때는 데이타베이스로부터 저장된 특정 문서를 메모리로 모델화 시켜서 그 모델에 대해서 RDQL이라는 질의 언어를 통해 질의 값을 얻어온다. 더 자세한 사항은 참고 문헌[16]을 참조하기 바란다.

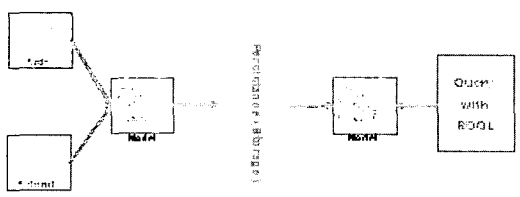

그림 3 Jena 1.6.1 아키텍처

다음은 이 논문에서 주된 논의가 되고 있는 데이타베이스 스키마에 대한 부분이다. 그림 4는 Jena의 데이타베이스 스키마를 보여주고 있다. Jena는 6개의 테이블로 구성되어 있으며 하나의 문서를 RDF 트리플로 나누어서 rdf_statements에 저장하게 된다. rdf_statements는 RDF의 트리플을 저장 하는 테이블이다. rdf_statements에서 SUBJECT, PREDICATE, OBJECT는 rdf_resources, rdf_literals의 ID를 참조하게 되며 MODEL은 rdf_models의 ID를 참조하게 된다. 가장 큰 특징은 질의 할 수 있는 대상이 한 문서에 한정되어 있다는 것이다. 즉, 전체 데이타베이스를 대상으로 질의하는 것이 아니라 저장되어 있는 한 문서를 메모리에 올려서 그것을 대상으로 질의하는 것이다. 그것은 저장 되는 문서의 이름과 ID를 저장하는 rdf_models 테이블이 있기 때문이다. 이것은 저장된 문서를 그대로 메모리 상에 모델화시킬 때 사용된다. 단점은 질의 시 메모리 상에 모델화시킨 다음에 추론엔진에 의해 질의하기 때문에 효율성이 떨어진다. 그러나 테이블 수가 적기 때문에 유지관리하기가 쉽다는 것이 장점이다. 또 하나의 장점은 의미의 손실이 없다는 것이다. 그것은 RDF의 특징 때문인데, RDF 기반의 웹 온톨로지 언어(RDFS, DAML+OIL)같은 경우에는 문서 자체가 동일한 의미를 가지는 RDF 트리플

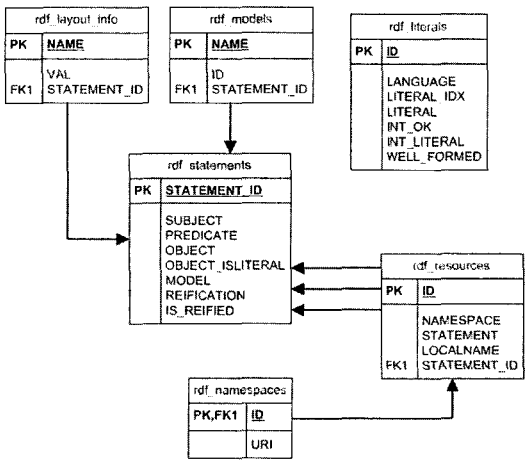

그림 4 Jean의 데이타베이스 스키마 다이어그램

형태의 그래프로 완전하게 표현되게 된다. 이것을 기반으로 하여 Jena는 이 RDF 그래프 모델을 데이타베이스 스키마화 하였기 때문에 의미의 손실이 없는 것이다.

3.2 Sesame and BOR

Sesame는 NLNet Foundation's Sesame에서 만든 RDF(S)문서를 저장하기 위한 시스템이다. Sesame의 목적은 RDF(S)문서를 저장시스템에 저장하는데 있어서, 문서가 가지고 있는 의미들을 잃어버리지 않고 데이타베이스에 효율적으로 저장하는 것을 목적으로 하고 있다. 특징을 보면, 다양한 저장시스템(메모리 스토리지, MySQL, PostgreSQL, Oracle 9i)을 지원을 한다. 또한 RDF, RDFS로 작성된 문서를 쉽게 업로드, 삭제할 수 있는 웹 기반의 인터페이스를 지원하며 저장된 문서에 대한 검색을 위해 RQL, RDQL의 질의 언어를 지원하고 있다. Sesame는 Jena와 달리 RDF, RDFS를 저장하고 관리하기 위한 통합시스템이다. 물론 Jena처럼 각각의 기능들을 사용할 수 있도록 패키지화되어 있지만, Sesame는 톰캣(Tomcat)기반의 웹 인터페이스를 제공해 주는 완성된 시스템이다.

다음은 Sesame의 아키텍처에 대해서 알아보자. 그림 5는 Sesame의 아키텍처를 보여주고 있다. 아키텍처에 대한 설명은 다음과 같다. Sesame 안에는 HTTP Protocol Handler와 SOAP Protocol Handler가 client와 통신을 제어함으로써 HTTP, SOAP과 같은 통신 프로토콜을 지원하고 있다. 그 다음으로 Request Router는 client의 요청을 처리하기 위해 해당되는 작업들을 Admin Module, Query Module, Export Module에 전달하는 역할을 한다. 그리고 Repository Abstraction Layer가 있는데 이것은 Repository와의 통신을 하는 부분이다. 이 모듈은 SAIL(Storage And Inference Layer)이라고도 하는데 특정 데이타베이스 시스템을 관리할 수 있도록 하는 API를 제공하는 모듈이다. 이것의 장점은 특정 데이타베이스 시스템을 Sesame와 연결하고자 할 때 SAIL의 인터페이스 부분만을 작성함으로써 연동을 쉽게 한다. 자세한 내용은 참고문헌([15])을 참조하기 바란다.

BOR[23]는 Sesame가 RDF(S)문서만 저장하는 단점을 보완하기 위해 DAML+OIL, DAML-S문서를 저장할 수 있도록 모듈을 추가한 부분이다. 그림 6에서처럼 BOR는 기존의 Sesame의 아키텍처위에 모듈형식으로 들어갈 수 있는 것을 볼 수 있다. 이 것이 Sesame의 장점이라고 볼 수 있는데, 기능 확장이 매우 유연하다는 것이다.

다음은 Sesame+BOR의 데이타베이스 스키마에 대해서 알아보겠다. Sesame+BOR의 데이타베이스는 그림 7에서 보는 것처럼 20개의 테이블(나머지 6개의 테이블은 버전정보에 대한 테이블이기에 생략함)로 이루어져 있다. Jena의 6개에 비해 상당히 많음을 알 수 있다. Sesame+BOR에서도 Jena와 마찬가지로 DAML+OIL, DAML-S문서가 들어왔을 경우에 RDF 트리플로 나누어서 저장하는 형태이다. 그러나 Jena에서는 6개의 테이블을 통해서 RDF 트리플로만 단순히 저장하지만 Sesame+BOR에서는 DAML+OIL이 가지고 있는 의미적인 부분을 추론을 통해서 저장하는 방식을 취하고 있다.

Sesame+BOR는 DAML+OIL문서가 들어오면 문서로부터 리소스(Resource)를 추출한 다음에 추출한 리소스로부터 RDF 트리플 형태로 변환한다. 리소스를 추출할 때 리소스가 가지고 있는 특징을 분류하게 되는데 range, domain을 통해 리소스가 가지는 범주를 range, domain 테이블에 넣는다. 그리고 리소스를 크게 인스턴스(Instance), 클래스(Class), 속성(Property)으로 분류한다. 마지막으로 클래스(Class)에 대해서 subclassof, direct_subclassof의 테이블에 넣고, 마찬가지로 속성(Property)에 대해서 subpropertyof, direct_subpropertyof의 테이블에 넣게 된다(이 부분은 추론을 통해서 추출하게 됨). Sesame+BOR의 특징으로는 질의 시 추론엔진이 해야 될 일들을 테이블로 분류를 한 것이다. 즉, 추론엔진이 이러한 테이블을 이용하여 질의 시 추론엔

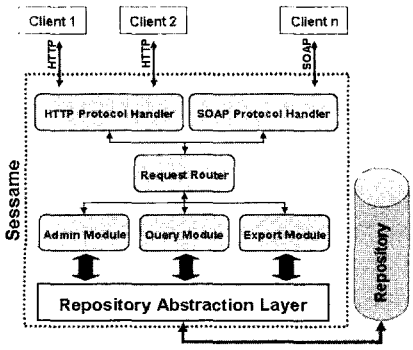

그림 5 Sesame 아키텍처

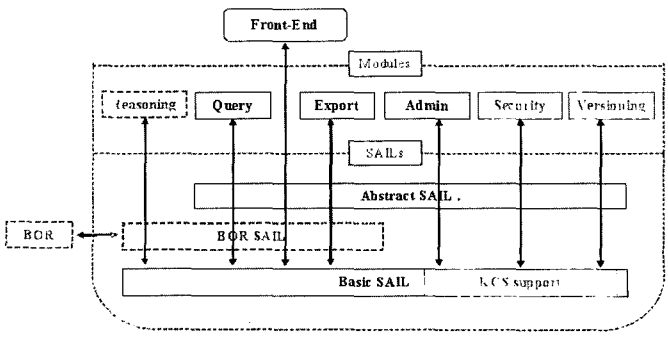

그림 6 BOR and Sesame Architecture

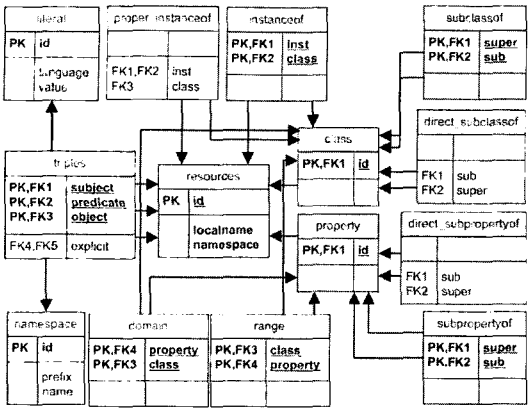

그림 7 Sesame+BOR의 데이타베이스 스키마 다이어그램

진이 해야 될 일을 많이 줄여주고 있다는 것이다.

4. DAML-S 문서 저장을 위한 데이타베이스 스키마 설계

여기에서는 위에서 소개한 Jena와 Sesame+BOR의 데이타 저장 측면에서 장단점을 분석하여 Jena와 Sesame를 절충하는 효율적인 데이타베이스 스키마를 제안하고자 한다. 표 1은 Sesame와 Jena의 성능 비교 분석표이다. 표에서는 Sesame 0.92 + BOR를 대상으로 했는데 Sesame가 RDF, RDFS만 저장할 수 있기 때문에 Jena와 비교를 위해 BOR를 사용하였다.

표 1 Sesame와 Jena의 성능 비교

	Jena 1.6.1	Sesame 0.92 + BOR
테이블	6	20
확장성 (statement)	small	3,000,000 이상
성능	small	O(10~3) classes and O(10~5) triples 이상

표 1에서 볼 수 있는 것처럼 Jena는 6개의 테이블을 통해서 RDF, RDFS, DAML+OIL를 저장할 수 있다. 확장성, 성능을 보면 Jena는 대용량의 데이타를 저장하기에는 부적절하다는 것을 알 수 있다. 왜냐하면, Jena의 질의 시스템의 구조 때문이다. Jena는 Sesame+BOR처럼 데이타베이스의 모든 테이블을 대상으로 직접 질의 하는 것이 아니다. Jena는 A라는 문서를 저장할 때 문서 A가 저장되었다는 것을 그대로 데이타베이스에 유지한다. 그리고 질의 할 때는 반드시 하나의 문서만을 대상으로 질의하게 된다. 즉, 질의 시 A라는 문서를 지정하고 데이타베이스로부터 A의 문서를 메모리 상에 모델화 시킨 다음에 그 모델을 대상으로 질의하는 구조로 되어 있다. 결과적으로 많은 문서가 데이타베이스에 저장될 경우 질의 시 성능상의 문제가 발생하는 것은 당연한 결과이다.

이에 비해 Sesame+BOR는 데이타베이스를 대상으로 질의 한다. 그래서 많은 문서를 저장할 수 있다. 표 1에서처럼 Sesame+BOR는 데이타베이스 시스템의 성능에 좌우되는 경향이 있는 것을 볼 수 있다. Sesame+BOR가 목표로 하는 것은 RDF, RDFS, DAML+OIL를 저장하는 데 있어서, 관계형 데이타베이스에서의 의미 손실 없이 문서를 저장하고 효율적으로 문서를 관리하는 것이다.

2개의 데이타베이스 스키마(Jena, Sesame)의 공통점으로는 문서를 저장하는데 있어서 RDF 트리플을 기반으로 한다는 것을 알 수 있다. 기본적으로 DAML+OIL은 RDF를 기반으로 한 확장된 언어이기 때문에 RDF가 가지는 특징을 그대로 가지고 있다. 다시 말해서 RDF가 주어(Subject), 술어(Predicate), 목적어(Object)의 트리플에 의한 그래프와 XML 형태의 RDF 문서가 상호 변환 가능하기 때문에 의미적으로 동등한 2개의 서로 다른 모델을 가지고 있는 것이다. 그래서 Jena, Sesame는 스키마의 기본 골격을 RDF 트리플에 기반을 둠으로써 관계 데이타베이스에 저장 시에 의미 손실을 없게 만들었다.

이에 이 논문에서는 Jena와 Sesame의 장점을 모아 그림 8과 같은 데이타베이스 스키마를 제안하고 자한다. 제안한 데이타베이스 스키마는 기본적으로 3가지의 조건을 만족 시키도록 설계되었다.

첫째는 문서를 저장할 때 발생하는 의미 손실을 없게 하는 것이다. 본 데이타베이스 스키마는 RDF 트리플의 그래프 구조를 그대로 스키마로 변형한 것이기 때문에 (Jena, Sesame가 했던 것처럼) 의미 손실이 없다.

두 번째는 효율성을 중요시 하였다. 의미 손실이 없다면 그 다음으로 중요한 것이 데이타베이스의 효율성을 들 수 있다. 그래서 그림 8의 스키마는 데이타베이스의 성능과 추론 엔진이 부담해야 하는 비용을 절충하도록 설계되었다. 데이타베이스 성능을 높이고자 한다면 문서로부터 적은 레코드를 삽입해야 되며, 추론엔진 쪽 입장에서는 데이타베이스 쪽에서 많은 정보를 가지고 있으면 그 만큼 추론을 위해서 발생할 수 있는 비용을 줄일 수 있다. 즉, 그림 9의 관계를 가지게 된다. 그림 9의 관계로부터 y축의 DBMS의 처리비용은 데이타베이스의 스키마에 저장되는 레코드의 수, 삭제 및 업데이트 시 테이블 접근수로서 명시할 수 있으며 x축은 추론엔진의 추론비용에 해당된다.

세 번째는 유연성을 들 수 있다. DAML+OIL같은 경우 아직 연구 중에 있으며 앞으로 개념이나 언어의 요

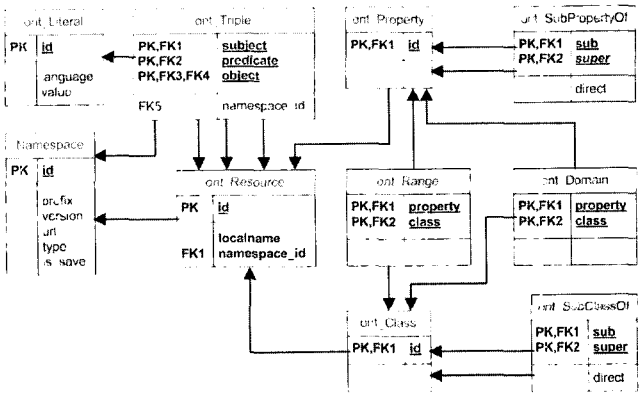

그림 8 제안한 데이타베이스 스키마

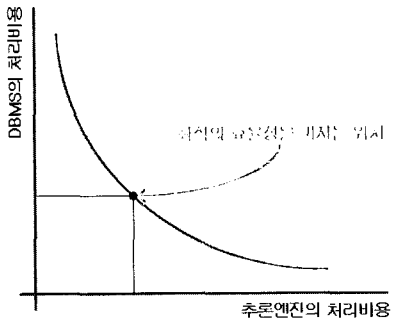

그림 9 비용 효율 곡선

소(Element)가 바뀔 수 있기 때문에 유연성을 갖추어야 한다. 이에 설계된 데이타베이스 스키마는 RDF 트리플 기반의 그래프 모델을 유지하고 있기 때문에 매우 유연하게 대처할 수 있도록 해준다.

위에서 다룬 설계의 기본 방침 3가지를 기반으로 하여, 그림 8에서처럼 데이타베이스 스키마는 총 10개의 테이블로 구성되어 있다. 본 테이블 중 실제 데이타가 저장되는 것은 ont_Triple, ont_Resource와, ont_Literal, 그리고 Namespace이다. 나머지 테이블들은 이 세 가지 테이블에 저장된 정보의 ID를 참조한다. 각각의 테이블에 대한 설명은 아래와 같다.

① Namespace : 이름 공간에 대한 정보를 저장하는 테이블이다. prefix 정보와, Namespace에 해당하는 문서에 대한 Version 정보, URL 등이 저장된다. 또한, 이 테이블은 현재 데이타베이스에 저장된 문서에 대한 정보를 가지고 있다.

② ont_Resource : 온톨로지에 포함된 모든 자원들을 저장하는 테이블이다. 각 자원의 Namespace의 ID와 Local name을 저장한다.

③ ont_Literal : 온톨로지에 포함된 모든 Literal들을 저장하는 테이블이다. 실제 Literal의 문자열 값과 어떤 언어인지 정보, 그리고 그 Literal의 Namespace의 정보를 저장한다.

④ ont_Triple : 온톨로지에 포함된 모든 Triple들을 저장하는 테이블이다. 각 Triple의 subject, predicate, object에 해당하는 자원의 ID 또는 Literal ID를 저장한다. 그리고 각 Triple이 포함된 문서의 위치 정보를 포함한다.

⑤ ont_Range : 자원들의 Range 정보를 저장한다. 실제 저장되는 값은 이에 해당하는 각 자원의 ID이다.

⑥ ont_Domain : 자원들의 Domain 정보를 저장한다. 실제 저장되는 값은 이에 해당하는 각 자원의 ID이다.

⑦ ont_Subclassof : 클래스들의 상호 상속 관계 정보를 저장한다. 실제 저장되는 값은 이에 해당하는 각 자원의 ID이다.

⑧ ont_Subproperty : 속성들의 상호 상속 관계 정보를 저장한다. 실제 저장되는 값은 이에 해당하는 각 자원의 ID이다.

⑨ ont_Class : 자원들 중 클래스로 정의된 것들을 추출하여 그 ID를 저장한다.

⑩ ont_Property : 자원들 중 속성으로 정의된 것들을 추출하여 그 ID를 저장한다.

이처럼 제안한 데이타베이스 스키마는 10개의 최적화된 테이블을 가진다. 이 10개의 테이블을 통해서 RDF(S), DAML+OIL, DAML-S 문서를 관계 데이타베이스에 저장할 때 의미의 손실 없이 문서를 관리할 수 있도록 한다. 데이타베이스 스키마에서 중심을 이루는 테이블은 ont_Triple, ont_Resource, ont_Literal, Namespace부분이다. RDF(S), DAML+OIL, DAML-S 문서들은 RDF의 트리플 형식을 가지고 있기 때문에, 이것을 반영한 설계이다. 가장 핵심을 이루는 테이블은 ont_Resource이다. 시멘틱 웹 문서 자체가 웹에 있는 자원의 기술이기 때문이며, ont_Resource는 그러한 모든 자원을 저장하는 가장 중요한 테이블이다. 그리고 이 리소스 테이블을 기반으로 해서 트리플 형태로 ont_Triple에 저장하고, 리소스와 관련된 네임스페이스 부분을 Namespace 테이블에 저장한다. 또한 데이타베이스에 저장할 때 추론을 통해서 데이타를 저장하는데 추후에 추론엔진에 의해 추론이 되어질 때 추론 엔진의 효율성을 높이기 위함이다. 이것에 해당 되는 것이 ont_Class, ont_Property, ont_Range, ont_Domain, ont_Subclassof, ont_Subpropertyof 이다. 물론, 추론엔진의 효율성을 높이기 위해서는 DAML+OIL의 요소(Element)들을

전부 테이블로 만들어서 추론엔진이 이용할 수 있게 할 수 있지만 그림 9의 비용 효율 곡선에서 보는 것처럼 효율성에는 한계가 있는 것을 볼 수 있기 때문에, 질의 시 가장 많이 발생할 수 있는 요소들에 해당되는 테이블 만을 고려하여 설계한 것이다.

이 논문에서 설계한 시스템은 시멘틱 웹서비스를 위한 시멘틱 웹서비스 문서 저장 시스템이다. 시멘틱 웹서비스 문서는 DAML-S를 통하여 기술된다. 그리고 이 DAML-S 문서는 DAML+OIL를 통해서 기술된다. 즉, 이 2종류의 문서는 이 논문에서 제안한 데이타베이스 스키마를 통해서 저장될 수 있다는 것을 의미하게 된다.

5. DAML-S 서비스 지원을 위한 DAML+OIL 저장 시스템 설계 및 개발

여기에서는 4장에서 설계한 데이타베이스 스키마를 적용한 시스템 구현에 대해서 설명한다. 그림 10은 데이타베이스 생성기에 대한 아키텍처이다.

데이타베이스 생성기는 DAML-S 문서를 분석하여 다른 모듈(질의 모듈, 사용자 인터페이스 등)에서 사용할 수 있는 적절한 데이타를 생성하여 데이타베이스에 저장하는 작업을 수행하는 모듈이다. 전체적으로 3개의 모듈이 데이타베이스 생성 작업을 수행한다. 기본적으로 문서 Validator와 문서 파서는 Jena 1.6.1을 사용한다. 왜냐하면, 개발자 입장에서 Jena 1.6.1은 효율적인 문서 Validator와 문서 파서를 제공하기 때문이다. 그리고 Jena를 사용해서 RDF 트리플을 추출하게 된다. 그림 10에 있는 각각의 컴포넌트에 대한 설명은 아래와 같다.

① Client Proxy : 관리자 인터페이스로부터 데이타베이스에 추가 하고자 하는 문서 정보를 입력받아 문서 객체를 생성하고 전체 시스템을 가동시킨다. 한 문서에 대한 모든 처리가 종료되면 관리자 인터페이스로 처리 결과를 출력하여 관리자로 하여금 결과를 확인할 수 있도록 한다. 문서의 주소는 인터넷에 존재하는 DAML-S 문서의 URL을 입력 받는다.

② DAML-S Validator : Client Proxy에서 받아들인 문서의 유효성을 검사한다. 기존의 DAML Validator를 기반으로 DAML-S를 처리 할 수 있도록 한다. 실제로는 Jena 내부의 Validator를 사용하였다. Jena의 Validator는 DAML 문서에 대한 간단한 수준의 유효성 검사를 수행한다.

③ Data Generator : DAML-S Validator로부터 정상적인 문서임을 검증 받으면, 이 문서로부터 적절한 정보를 추출하고, 그 정보를 데이타베이스에 저장한다. 실제 데이타베이스와의 통신은 이 모듈 내부의 DB Connector가 수행한다.

위와 같은 컴포넌트의 작동되는 시나리오를 보면 그림 11과 같다. ①에서 DAML-S 문서로부터 온톨로지들에 대한 정보를 추출하여 각각의 문서에 포함된 정보를 1차적으로 생성하여 DAML-S 데이타 생성기(DDG)에 보낸다. 이 과정에서는 Jena에 의해 추출된 정보와 네임스페이스와 관련된 정보를 생성한다. URL을 이용하여 외부에서 온톨로지 문서를 가져와서 ②와 같이 정보를 추출하여 DDG에 보낸다. ②와 ③의 과정에서 생성된 1차 정보를 통해서 실제 데이타베이스에 저장할 데이타를 생성한다. 마지막으로 ④는 추출된 데이타들을 데이타베이스에 저장하는 역할을 한다.

DAML-S 데이타베이스 생성기는 문서를 데이타베이스에 추가하는 모듈이다. 위에서는 데이타베이스 생성기의 아키텍처를 보았는데 그림 12는 시스템에 대한 전체 유스케이스 다이어그램을 보여주고 있다.

Administrator가 데이타베이스에 저장하고자 하는 DAML-S 문서를 선택하여 삽입 명령을 내리면 위에서 설명한 모듈들이 각자의 역할을 수행하여, 필요한 정보를 데이타베이스에 저장한다. Administrator가 영향을

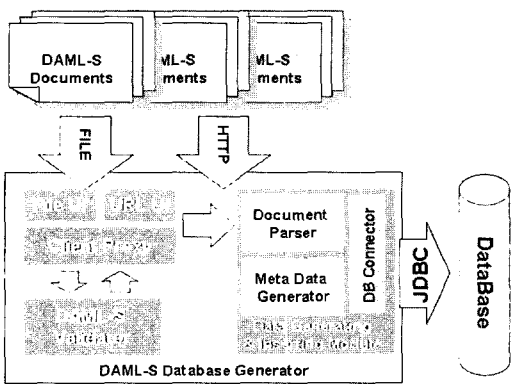

그림 10 데이타베이스 생성기 아키텍처

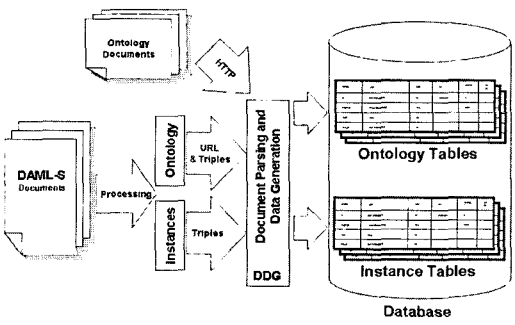

그림 11 데이타베이스 생성 과정

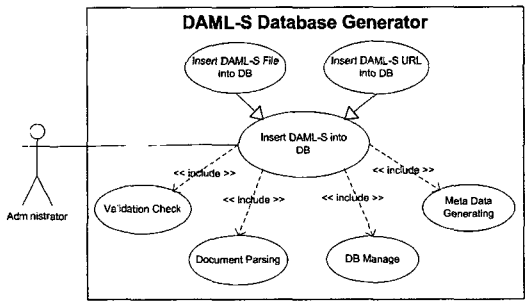

그림 12 데이타베이스 생성기 유스케이스 다이어그램

미치는 Use Case는 "Insert DAML-S into DB"이고, 이 과정에서 "Insert DAML-S URL into DB"를 이용하게 된다. "Insert DAML-S URL into DB"는 "Insert DAML-S into DB"를 상속하여 기능을 확장한 유스케이스가 된다. 그리고 "Insert DAML S into DB"는 "Check Validation", "Parse Document", "Manage DB", "Generate Meta Data"와 같은 유스케이스를 Include한다.

이 유스케이스 다이어그램은 시스템에 대한 계략적인 그림이다. 따라서 이것을 기반으로 상세한 유스케이스를 기술 할 필요가 있다. 여기에서는 지면 관계상 생략한다. 상세한 유스케이스로부터 명사가 되는 단어로부터 클래스가 될 수 있는 요소들을 판단하여 클래스를 추출한다. 그런 다음에 추출된 클래스들 사이의 관계를 기술하고 그러한 관계들이 올바른지를 검증한다. 이 논문에서 구현한 시스템은 최종적으로 그림 13과 같은 클래스 다이어그램을 형성하게 된다.

각 클래스에 대한 자세한 설명은 생략하겠다. 위와 같이 설계된 데이타베이스 생성기 모듈은 질의 프로세서나 사용자 인터페이스와 쉽게 결합하여 시멘틱 비즈니스 정보 검색 엔진구현에 이용할 수 있다. 특히, 이 논문에서 구현된 데이타베이스 생성기는 임베디드 시스템 환경에서 DAML-S 문서를 관리하는 시스템의 한 모듈이다. 이 시스템의 환경은 실시간 운영체제(RTOS)기반의 임베디드 SQL Anywhere를 사용하였으며 원격에서 저장시스템을 모니터링 및 관리하기 위해 Jakarta "Tomcat" HTTP Server를 이용하였다. 그림 14와 같은 사용자 인터페이스를 가지면서 문서에 대한 모니터링 및 관리를 할 수 있다.

6. 효율성 분석

6장에서는 이 논문에서 설계한 데이타베이스 스키마와 Sesame+BOR와의 효율성 비교를 설명한다. 본 논문

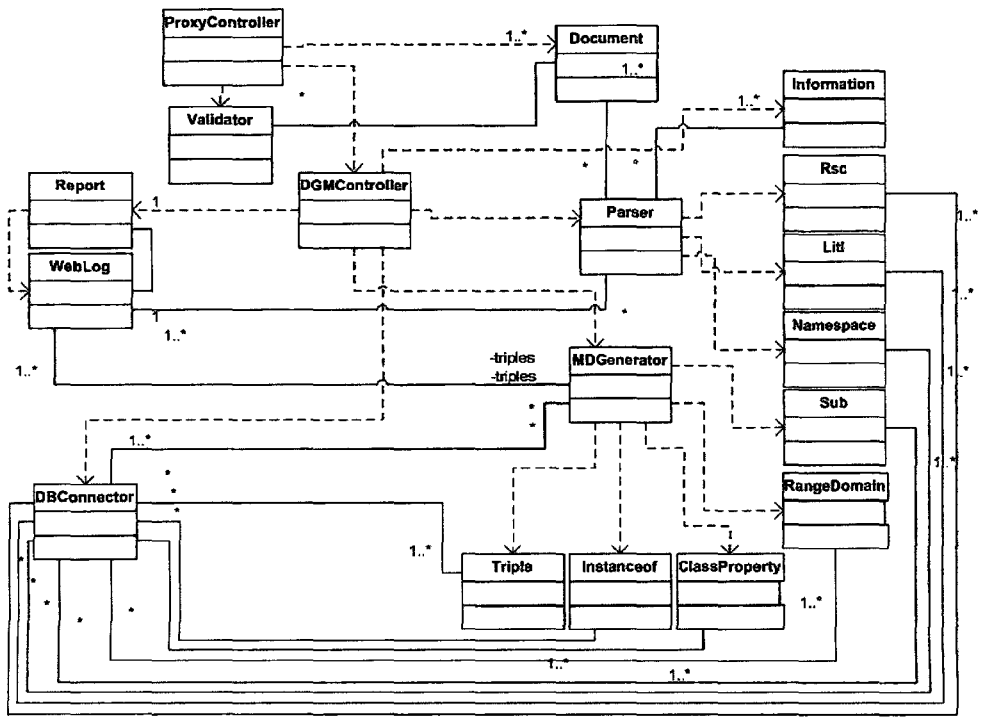

그림 13 데이타베이스 생성기 클래스 다이어그램

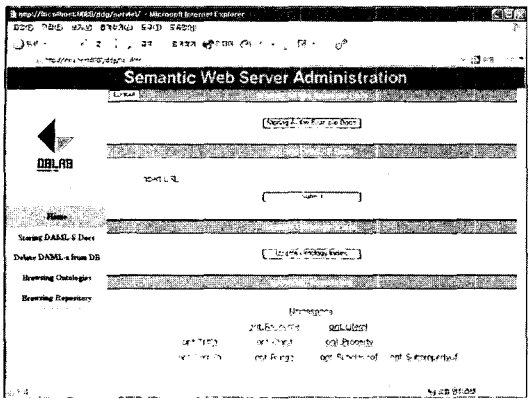

그림 14 데이타베이스 생성기와 관리 모듈 구현

에서의 데이타베이스 스키마 및 문서 관리 시스템은 임베디드 환경에서 데이타의 효율적인 관리를 위해 최적화되어 있다. 따라서 데이타의 저장 측면을 기준으로 설명한다. 먼저, 비교기준을 다음과 같이 분류하였다. 데이타베이스 스키마 다이어그램의 형태, 스키마의 단순함(중복성, 유연함), 문서 저장 시 데이타의 저장량, 질의에 따른 효율성 분석, 이렇게 4가지 기준에 의해 평가를 하였다.

첫 번째로, 데이타베이스 스키마 다이어그램의 형태에 대한 평가이다. 데이타베이스 스키마 다이어그램은 어떻게 데이타 모델링을 했는지 보여주는 청사진과 같다. 그래서 잘못 설계된 데이타 모델은 사용자의 요구변경이나 업무의 변경에 따라 함부로 바꿀 경우에 매우 위험하고 많은 유지, 보수비용이 필요하게 되는 원인이 된다. 즉, 데이타 모델을 분석함으로써 얼마나 잘 설계되었는지를 평가할 수 있다. Sesame+BOR와 이 논문의 데이타베이스 스키마의 전체 구조는 비슷하다. 그것은 RDF의 특징을 그대로 반영하고 있기 때문이다. RDF는 리소스를 기술하는 언어이며, 리소스를 기술하는데 있어서 주제(Subject), 술어(Predicate), 목적어(Object)의 3-트리플(Triple)로 표현하고 있다. 그림 7(Sesame+BOR)과 9(본 논문)에서 보는 것처럼 문서의 리소스를 저장하기 위해 리소스 테이블, 리소스를 기술하기 위해 트리플 테이블로 나눈 것을 볼 수 있다. 이 2개의 테이블을 기반으로 해서 데이타 모델링이 이루어져 있다. 보통 데이타 모델링의 형태를 형제형, 네트워크형, 족보형으로 3가지로 분류하고 있는데 Sesame+BOR처럼 제안된 데이타베이스 스키마는 족보형 형태를 가지고 있다. 이것은 트리형태를 말하는 것으로 자식의 테이블이 부모의 기본 키를 외부 키로 가지고 있는 형태이다.

두 번째로, 스키마의 단순함, 중복성, 유연함에 의한 평가이다. 단순함의 측면에서 볼 때, 두개의 스키마 각각의 테이블은 특징이 명확하며 쉽게 이해할 수 있도록 되어 있으며, 하나의 테이블은 하나의 특정한 정보만을 담도록 설계되어 있다. 유연성 측면에서 봤을 때, DAML+OIL의 문서가 버전 업이 되어 온톨로지를 기술하는 요소의 이름이 바뀌더라도 데이타베이스 스키마의 변경은 거의 없다. 단순히 버전업 된 DAML+OIL를 지원하는 파서만 교체하면 된다. 이것은 DAML+OIL의 문서에서 가장 기본이 되는 특징들만을 가지고서 설계했기 때문이다. 그 다음 중복성면에서 봤을 때, Sesame+BOR는 direct_subclassof, direct_subpropertyof의 중복된 테이블을 가지고 있다. Sesame+BOR에서 direct_subclassof, direct_subpropertyof의 역할은 특정한 문서에서 정의된 특정한 Class, Property의 추론하지 않은 실제 부모-자식 관계를 명시하기 위한 목적이다. 이것은 subclassof, subpropertyof 테이블과 중복이 발생된다. 이 논문의 데이트베이스 스키마에서는 subClassOf, subPropertyOf의 테이블 속성으로 "direct"를 둠으로써 중복되는 데이타를 제거하였다. 나머지 중복되는 테이블인 Sesame+BOR의 proper_instanceof와 instanceof는 DAML+OIL의 특정한 클래스에 의해 생성된 인스턴스를 나타내기 위한 목적으로 설계된 스키마이다. 질의 중에 "사람이라는 클래스의 모든 인스턴스는?" 이라는 질의가 있을 경우에 유용한 테이블이다. 그러나 이 스키마를 이용하지 않고도 질의를 할 수 있다. 그것은 트리플 테이블을 통해서 이다. 예를 들면, 트리플 테이블의 "Predicate"의 값이 "type"이고 "Object"로 "사람"이라는 질의를 통해서 같은 결과를 얻을 수 있다. 결과적으로 proper_instanceof, instanceof는 질의를 위한 효율성보다는 데이타의 중복으로 인한 데이타베이스의 효율성을 감소시키고 있다. 여기에 대한 분석은 뒤에서 질의의 효율성 분석에서 다시 언급한다.

세 번째로, 문서 저장 시 데이타의 저장량에 의한 분석이다. 먼저, 이 논문에서의 데이타의 추출 방식을 Sesame+BOR와 비교하여 설명한다. 그림 15는 데이타 추출을 설명할 예제를 보여주고 있다. 그림에서 보는 것처럼 "USState"라는 클래스를 정의하기 위한 예제이며 이 XML 형태의 DAML+OIL은 RDF 그래프와 동일한 의미를 가진다. 그림 16은 그림 15의 예제를 통해서 생성되는 트리플에 대한 Sesame+BOR와 본 논문의 트리플 추출 비교 그림이다. Sesame+BOR에서는 트리플 추출 시에 숨겨져 있는 의미를 모두 저장한다. 그림 16에서처럼 "USState"의 type은 "resource", "USState"의 subClassOf는 "resource"등에서 볼 수 있다. 이러한 숨겨져 있는 트리플을 저장할 경우에 생성되는 레코드는 거의 2배정도가 늘어나게 된다. 그래서 이 논문의 시스템에서

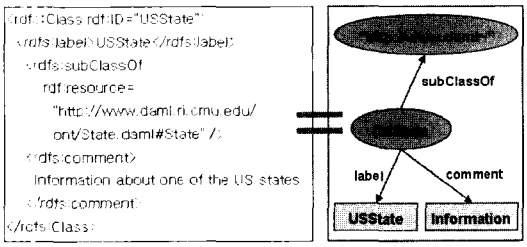

그림 15 DAML+OIL의 예제 및 RDF 그래프

subject	predicate	object
USState	type	resource
USState	type	class
USState	subClassOf	resource
USState	subClassOf	USState
USState	subClassOf	State
USState	label	USState
USState	comment	Info~
State	type	resource
State	type	class
State	subClassOf	resource
State	subClassOf	State

Sesame+BOR

subject	predicate	object
USState	type	class
USState	subClassOf	State
USState	label	USState
USState	comment	Info~

본 논문

그림 16 Sesame+BOR와 본 논문에서의 트리플 추출 비교

는 숨겨져 있는 트리플을 저장하지 않고 추론엔진에서 처리하도록 하였다. 즉, 추론엔진에서 이러한 트리플의 규칙들을 가지고 있도록 하여서 데이타베이스의 효율성을 증대시키게 하였다. 또한, 숨겨져 있는 의미들을 추론엔진이 가지고 있으므로 데이타베이스에 직접 접근하지 않고서 숨겨져 있는 의미들을 추론할 수 있기 때문에 추론엔진의 효율성도 높일 수도 있다. 즉, 결과적으로 그림 16의 본 논문에서의 트리플 추출에서와 같은 결과를 얻게 된다. 이러한 비슷한 방법을 통해서 다른 테이블의 레코드도 생성되게 된다. 그림 17은 그림 15의 예제를 삽입하였을 경우 생성되는 레코드를 비교한 것이다. 트리플의 효율적인 추출로 인해서, 그리고 중복 테이블의 제거를 통해서 효율성 및 성능의 향상을 볼 수 있다. 그 다음 삭제는 삽입의 반대 과정이기 때문에 당연히 제안한 데이타베이스 스키마가 효율적이다. 물론 업데이트도 마찬가지이다. 그림 18는 10개의 문서를 저장했을 때 저장되는 레코드의 수를 비교한 그래프이다. 그림에서 보는 것처럼 전체적으로 기존의 시스템 보다 약 2배정도 적게 데이타가 생성되는 것을 볼 수 있다. 이것을 통해서 본 논문에서 제안한 데이타베이스 스키마 및 관리 시스템이 임베디드 환경에서 기존 시스템보다 문서 관리 시스템을 구축하는데 있어 더 효율적이라고 볼 수 있다.

네 번째로, 질의에 따른 효율성 분석이다. 객관적으로 평가하기 위해, 추론엔진의 추론 능력을 배제하고 데이

Tables	Record #
namespaces	2
literals	4
resources	2
triples	14
subclassof	5
proper_instanceof	2
Instanceof	4
class	2
direct_subclassof	2
Total	37

<Sesame+BOR>

Tables	Record #
Namespaces	2
ont_Literals	2
ont_Resources	2
ont_Triples	4
ont_Subclassof	1
ont_Class	1
Total	12

<제안된 DB 스키마>

그림 17 테이블 접근과 삽입되는 레코드 비교 테이블

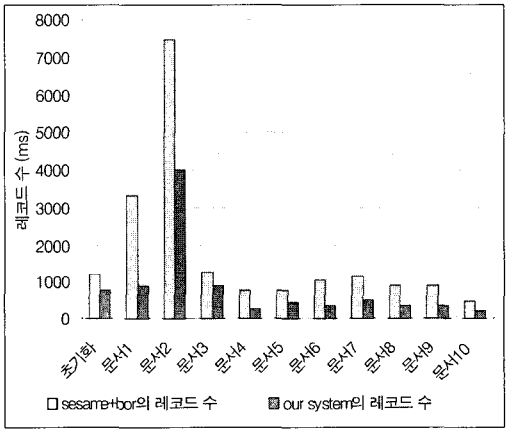

그림 18 문서에 따른 레코드 수 비교

타베이스 스키마의 효율성만을 분석하기로 한다. 즉, SQL 문을 이용하여 추론엔진이 추론 시 필요한 데이타를 가져오는데 걸리는 시간을 기반으로 하여 비교 분석한다. 질의의 특성에 따라 질의를 다음과 같이 분류한다. 각각의 질의는 추론엔진이 추론 시에 필요로 하는 데이타를 추출할 수 있도록 해준다.

(1) 트리플에 대한 질의

⇒ 질의1 : 술어(Predicate)가 "type"이고 목적어(Object)가 "Class"인 모든 리소스의 리스트

(2) SubClassOf 관계에 대한 질의

(SuperClassOf, SubPropertyOf, SuperPropertyOf 관계도 비슷한 질의에 해당됨)

⇒ 질의2-1 : 클래스 "Beer"의 모든 자식 클래스의 리스트 (direct, indirect 모두 포함)

⇒ 질의2-2 : 클래스 "Beer"의 모든 자식 클래스의 리스트 (direct만 해당)

(3) Range, Domain을 만족하는 속성(Property)에 대한 리스트

⇒ 질의3 : Domain이 "Race"인 모든 속성(Property)의 리스트

(4) 특정 클래스의 모든 인스턴스에 대한 질의

⇒ 질의4 : 클래스 "USCity"의 모든 인스턴스

질의1부터 질의4까지의 질의 처리 모듈을 만들고 1000번의 실행결과의 평균 질의 처리 시간의 결과는 그림 19와 같이 측정되었다. 전체적으로 봤을 때 본 논문의 데이타베이스 스키마가 질의 처리 시간이 더 효율적임을 알 수 있다.

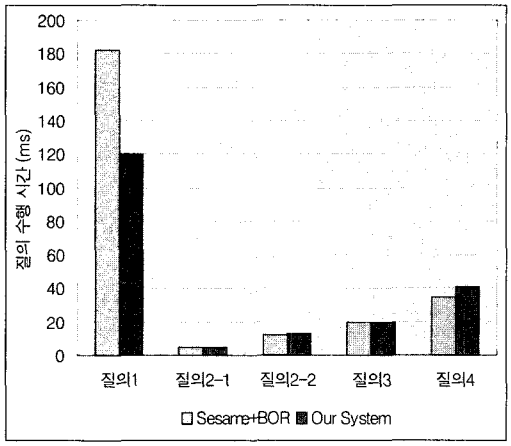

그림 19 질의에 따른 질의 처리 시간 비교

그림 19처럼 질의1은 저장된 레코드의 수가 적기 때문에 트리플 테이블의 질의 처리 속도가 본 논문의 데이타베이스 스키마가 빠름을 알 수 있다. 질의2-1, 질의2-2, 질의3은 거의 비슷함을 알 수 있다. 마지막으로 질의4의 경우 Sesame+BOR 같은 경우 클래스의 인스턴스에 대한 테이블을 따로 만들어서 더 효율적임을 알 수 있다. 하지만, 이 테이블을 유지하기 위해 중복되는 데이타가 너무 많다. 예를 들어, 미국의 주를 DAML+OIL문서로 만든다고 가정하면 클래스로서 "USState"를 정의하고 "USState"의 인스턴스로서 미국의 51개 주를 정의하게 된다. 그러면 이 문서의 전체 리소스는 52개가 되며 인스턴스에 대한 테이블의 레코드만 51개가 저장된다. 즉, 실제로 생성되는 레코드의 거의 2배가 증가하게 되는 것이다.

지금까지 본 논문의 데이타베이스 스키마와 Sesame+BOR의 데이타베이스 스키마의 효율성을 분석하였다. 본 논문의 가장 큰 특징은 Sesame+BOR와 거의 비슷한 질의 처리 능력을 가지면서 저장되는 레코드가 절반 수준에 있다는 것이다. 또한 Sesame+BOR와 같이 문서가 가지고 있는 의미의 손실 없이 저장한다는 것도 가장 큰 특징이다.

7. 결 론

시멘틱 웹에 대한 연구가 진행되면서 시멘틱 웹의 활용 분야가 점차 확대되고 있다. 이에 따라 효율적으로 시멘틱 웹 언어의 문서를 저장하고 관리하는 시스템의 중요성도 커지고 있는데, 이러한 문서는 기존의 텍스트 문서처럼 단순히 저장되는 것이 아니라 문서가 가지고 있는 의미적인 부분들 (SubjectOf, Range, Domain 등의)을 손실 없이 저장하는 것이 중요하다. 만약 문서가 가지고 있는 의미를 잃어버린다면 시멘틱 웹 기술언어로 작성된 문서로서의 의미가 사라지기 때문이다. 이 논문의 데이타베이스 스키마는 효율적이고 대용량의 문서를 저장하고 관리할 수 있는 골격을 제공해주며 문서의 의미를 손실 없이 저장할 수 있게 한다.

위에서 본 것처럼 본 논문의 시스템은 디스크 공간 및 메모리 공간이 제한된 임베디드 환경에서 DAML-S 서비스 지원을 위한 DAML+OIL 문서를 효율적으로 저장하도록 설계되었을 뿐만 아니라, 좀더 나아가 시멘틱 웹 검색엔진, 시멘틱 웹서비스, 데이타 통합, 네비게이션 등의 응용 애플리케이션에서도 사용될 수 있다. 그 중에서 대표적으로 로봇환경에서 로봇의 인공지능 시스템을 지원하는 모듈로 응용될 수 있다.

참 고 문 헌

[1] James Hendler, Sheila A. McIlraith, David L. Martin, Bringing Semantics to Web Services, 2003.
[2] Semantic Web, http://www.w3.org/2001/sw
[3] Sheila, A. McIraith et al, Semantic Web Services, IEEE Intelligent Systems, 2001.
[4] T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American, vol 284, no.5, pp.34-43, 2001.
[5] Alexander Maedche, Steffen Staab, Ontology Learning for the Semantic Web, 2001.
[6] Tom Gruber, What is an Ontology?
[7] Resource Description Framework, http://www.w3.org/RDF
[8] Extensible Markup Language, http://www.w3.org/XML
[9] RDF Vocabulary Description Language 2.0 : RDF Schema, http://www.w3.org/TR/rdf-sche-ma
[10] Dieter Fensel, Ontologies : A Silver Bullet for Knowledge Management and Electronic Com-merce, 2004.
[11] DAML+OIL, http://www.daml.org/2001/03/DAML+OIL-index.html
[12] DAML, http://www.daml.org
[13] DAML-S 0.9 Draft Release, http://www.daml.org/services/daml-s/0.9
[14] The DAML Services Coalition, DAML-S : Semantic Markup for Web Services.

[15] Sesame, http://sesame/aidministrator.nl
[16] Jena, http://www.hpl.hp.com/semweb/jena.htm
[17] Harris, S. and Gibbins, N, "3store : Efficient Bulk RDF Storage," In Proceedings of 1st International Workshop on Practical and Scalable Semantic Systems(PSSS'03), pages pp. 1-15, Sanibel Island, Florida.
[18] Zhengxiang Pan, Jeff Heflin, "DLDB : Extending Relational Databases to Support Semantic Web Queries," Workshop on Practical and Scaleable Semantic Web Systms, ISWC 2003, pp. 109-113.
[19] Alberto Reggiori, Dirk-Willem van Gulik, Zavisa Bjelogrlic, "Indexing and retrieving Semantic Web resources : the RDFStore model," SWAD-Europe Workshop on Semantic Web Storage and Retrieval, 2003.
[20] Kilian Stoffel, Merwyn Taylor, and James Hendler, "Efficient management of very large ontologies," Procedings of American Association for Artificial Intelligence Conference, 1997.
[21] TAP Project, GetData : Querying the Data Web, http://tap.stanford.edu/tap/getdata.html, 2003.
[22] RDQL-RDF Data Query Language : http://www.w3.org/TR/rdf-schema.
[23] BOR, http://www.ontotext.com/bor/index.html
[24] Jeen Broekstra, Arjohn Kampman, Sesame : A Generic Architecture for storing and Queryi-ng RDF and RDF Schema, 2001.
[25] WSDL, http://www.w3.org/2002/ws/

김 학 수

2004년 한양대학교 전기전자제어컴퓨터공학부 학사. 2004년~현재 한양대학교 컴퓨터공학과 석사과정. 관심분야는 그리드 컴퓨팅, XQuery, 시멘틱 웹, XML 데이타베이스

정 문 영

2003년 강남대학교 무역학과 학사. 2003년~현재 한양대학교 컴퓨터공학과 석사과정. 관심분야는 비즈니스 프로세스 관리 시스템, 시멘틱 웹, XML, 데이타베이스 시스템

차 현 석

2003년 계명대학교 컴퓨터공학과 학사 2004년~현재 한양대학교 컴퓨터공학과 석사과정. 관심분야는 시멘틱 웹, 센서 데이타베이스, XML

손 진 현

1996년 서강대학교 전산학과 학사. 1998년 한국과학기술원 전산학과 석사. 2001년 한국과학기술원 전자전산학과 박사 2001년 9월~2002년 8월 한국과학기술원 전자전산학과 박사후 연구원. 2002년 9월~현재 한양대학교 컴퓨터공학과 조교수. 관심분야는 데이타베이스, e-비즈니스, 유비쿼터스 컴퓨팅, 임베디드 시스템