RDF 스키마에서 UML 클래스 다이어그램으로의 변환

이 미 경† · 하 얀†† · 김 용 성†††

요 약

인터넷 상의 문서가 많아지고 이에 대한 정확한 접근이 요구됨에 따라, 인터넷 자원에 대한 메타데이터를 표준화시키고, 메타베이스를 구축·활용하는 것은 매우 중요하다. RDF(Resource Description Framework)는 구조화된 메타데이터를 표현하고, 교환하며 재사용하기 위한 기반 구조이며, 문법으로 XML을 이용하기 때문에 표준화된 메타데이터에 대한 일관된 표현 및 교환, 처리가 가능하다. RDF 스키마는 RDF 모델에서 사용을 위해 제안한 기본 타입 시스템으로 본 논문에서는 XML로 된 RDF 스키마를 UML 클래스 다이어그램에 사상시키는 규칙과 알고리즘을 제안하고 이를 통해 객체 모델링하므로써 객체지향 데이터베이스 스키마로의 변환을 용이하게 한다. 그리고, RDF 스키마에 대한 객체지향 스키마 형태인 형식 모델을 정의하여 객체지향 문서 처리와 검색을 위한 효율적인 환경을 제시할 것이다.

An Conversion a RDF Schema into an UML Class Diagram

Mi-Kyung Lee† · Yan Ha†† · Yong-Sung Kim†††

ABSTRACT

With increasing amounts of information on the web and needs to access accurately them, it is very important to standardize metadata and to store and manage metabase system. The RDF(Resource Description Framework) is a framework for representing, exchanging, and reusing metadata. And, it can be processing uniformly the standardized metadata, because it uses XML(eXtensible Markup Language) syntax. The RDF schema provides a basic type system for use in RDF models. In this paper, we propose rules and an algorithm to convert the RDF schema into an UML(Unified Modeling Language) class diagram and formal models to represent an object-oriented schema for the RDF schema. The proposed rules and algorithm are useful for natural mapping and the object modeling of RDF schema can be easily converted into the object-oriented schema, and the formal models supports an efficient environment for retrieving and processing object-oriented documents.

1. 서 론

인터넷 상의 방대한 문서들에서 사용자가 필요로 하는 문서를 정확하게 찾아내기 위해서는 해당 문서에 대한 정확한 정보를 알아야 한다. 이렇게 문서 내용을 기술하는 정보 뿐 만 아니라 문서 자체에 대한 정보를 메타데이터라 한다. 최근에는 메타데이터의 중요성이 강조됨에 따라 인터넷상의 자원을 대상으로 한 메타데이터의 표준화 작업과 효율적이고 체계적인 관리를 위한 메타베이스를 구축에 관한 연구가 활발히 추진되고 있다[1]. 메타 데이터의 표준화를 위해 XML(eXtensible Markup Language)를 기반으로 기술된 RDF(Resource Description Framework)는 웹에 대한 모든 메타데이터를 수용하며, 인터넷상의 분산된 다양한 자원들을 기술하기 위한 표준화된 프레임워크를 제공하게 된다.

한편, XML은 SGML(Standard Generalized Markup

* 이 논문은 1998년도 한국학술진흥재단의 학술연구조성비(국제협력공동과제)에 의하여 연구되었음.
† 정 회 원 : 서울정수기능대학 정보통신설비과 교수
†† 준 회 원 : 전북대학교 대학원 전산통계학과 박사과정
††† 종신회원 : 전북대학교 컴퓨터과학과 교수
논문접수 : 1999년 8월 20일, 심사완료 : 1999년 12월 9일

Language)의 복잡한 기능을 축약시키고, HTML(HyperText Markup Language)의 장점을 추가한 형태로 1996년에 W3C(World Wide Web Consortium)에서 제정하였다. 이러한 XML은 특정 응용 목적에 맞게 엘리먼트, 애트리뷰트, 엔티티를 정의할 수 있는 융통성(flexibility)을 갖는데, 그 예로 CDF(Channel Definition Format), OSD(Open Software Description), RDF, WIDL(Web Interface Definition Language) 등이 있다[2].

XML의 응용 중 메타데이터를 정의한 RDF 스키마를 데이터베이스에 저장, 체계적으로 관리하기 위해서는 객체 모델링이 필요하다. 따라서 본 논문에서는 객체지향 기법의 표준이 되고 있는 UML(Unified Modeling Language)을 이용하여 RDF 스키마를 먼저 객체 모델링을 하고, 그 다음 객체지향 스키마 코드를 생성하고자 한다.

그런데, RDF 스키마와 UML은 둘 다 스키마를 갖고 있는 공통점이 있으나 다른 응용 목적을 갖고 사용되기 때문에 이들 간에 사상이 필요하다. UML 같은 E-R 다이어그램의 경우는 엔티티(entity)와 관계(relation)의 모델을 기술하고, RDF 스키마 같은 노드 및 아크(node-and-arc) 다이어그램의 경우에는 노드와 아크의 모델을 기술하기 위해 사용되고 있다[3]. 따라서, 본 논문에서는 노드와 아크 관계를 갖는 RDF 스키마를 UML 클래스 다이어그램에 사상시키는 규칙과 알고리즘을 제시하여 RDF 스키마에 대한 객체 모델링을 생성하고자 한다. 또한, RDF 스키마에 대한 객체지향 데이터베이스 스키마 형태인 형식 모델을 제안하여 RDF 스키마에 대한 객체지향 데이터베이스 문서 관리의 기반을 제공하고자 한다.

논문 구성은 다음과 같다. 2장은 관련 연구를 설명하고, 3장에서는 RDF 스키마와 UML 클래스 다이어그램에 대해 설명하며, 4장에서는 변환 알고리즘으로 변환 규칙, 모델링 함수, 형식 모델, 알고리즘, 시스템 및 적용 결과, 그리고, 비교 분석을 제시하며, 끝으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

메타데이터의 모델링으로 노드와 아크 형태의 데이터 모델[4]이 있으나 이것은 단지 메타데이터의 객체를 추출하여 사용하였을 뿐, 애트리뷰트나 관계성의 다양한 표기법은 지원되지 못한다.

그 외에 SGML과 XML 문서에 관한 모델링에 관한 연구를 살펴보면 다음과 같다. SGML DTD(Document Type Definition)를 객체 모델링하는 연구로 XOMT(eXtended Object Modeling Technique)[5] 다이어그램과 UML 클래스 다이어그램을 이용한 모델링[6]방법이 있다. XOMT는 OMT(Object Modeling Technique)를 기반으로 하였으나 엔티티, 순위번호, 예외처리 등에 대해 확장된 표기법을 제안하고 있으며, 일반화(generalization) 관계성(relationship)에 발생 횟수를 표시해야 하는 등 객체지향 개념에 어긋나는 부분이 있다. 이에 비해 [6]의 모델링 방법은 SGML DTD를 객체지향 개념에 부합되게 객체 모델링을 하였다. 그러나, 이것 역시 SGML DTD에 의존적으로 설계된 모델링이기 때문에 XML 형태로 된 RDF 스키마를 제대로 표현할 수 없다.

XML 문서를 위한 표준화된 API(Application Programming Interface)를 제공하여 웹 문서에 접근하고 조작하기 위한 방법으로 DOM(Document Object Model)[2, 7]이 있는데, 이것은 문서의 논리적 구조를 정의하는 객체 기반 구조이기는 하나 트리 계층 구조의 형태이므로 클래스의 세부적인 정보인 애트리뷰트나 일반화 및 집단화(aggregation) 관계성 등을 제대로 표현하지 못한다.

그 외에 XML 문서에 대한 모델링으로 Elm 트리 다이어그램[8]과 UML 클래스 다이어그램을 이용한 방법[9]이 있다. Elm 트리 다이어그램은 트리 구조에 기반하여 표기법을 제안, XML DTD를 모델링하였으나, 엘리먼트 선언에서 괄호에 대한 처리, 링크 등에 대한 처리가 미흡한 단점이 있다. 이에 비해 UML 클래스 다이어그램을 이용한 모델링은 XML DTD의 링크 등을 포함한 다양한 경우를 지원하고 있다.

그러나, [9]의 모델링 방법에서 XML의 엘리먼트는 UML의 클래스가 되고, XML의 엘리먼트 선언에서 이름 부분과 내용 부분에 오는 엘리먼트들간에 집단화 관계성을 갖도록 하나, RDF 스키마에서는 엘리먼트 태그에서 클래스와 클래스들 간의 관계성 등을 모두 나타내고 있다. 따라서, 일반적인 XML 문서의 모델링에서 지원되지 않는 RDF 스키마에 대한 새로운 모델링 방법이 필요하다. 그리하여, 본 연구에서는 RDF 스키마를 UML 클래스 다이어그램으로 변환시키는 규칙과 알고리즘을 제안하고자 한다.

3. RDF 스키마와 UML 클래스 다이어그램

본 장에서는 RDF 스키마와 UML 클래스 다이어그램의 중요한 구성 요소에 대해 살펴본다.

3.1 RDF 스키마

웹 관련 정보를 위한 프레임워크를 설정하기 위해 활동하는 메타데이터 워킹 그룹(Working Group)은 크게 PICS(Platform for Internet Content Selection) 분야가 있다. 그 중 RDF 워킹 그룹은 인터넷상의 자원에 대한 정보를 기술하는 표준을 정의하기 위한 목적으로 활동하는데, RDF 모델 및 구문(Model and Syntax) 워킹 그룹, RDF 스키마(Schema) 워킹 그룹으로 나뉘어 있다. W3C(WWW Consortium)는 RDF 모델 및 문법을 1999년 2월에 권고안(Recommendation)으로 제정하였고, RDF 스키마를 1999년 3월에 제안 권고안(Proposed Recommendation)으로 제정하였다[7].

다음은 RDF 모델 및 구문과 스키마에 대해 개략적으로 기술한다.

(1) RDF 모델 및 구문(Model and Syntax)

RDF는 메타데이터를 처리하기 위한 기반으로 웹에서 기계가 읽을 수 있는 정보를 교환하는 응용들(applications) 사이의 상호운영성(interoperability)을 제공한다[4]. RDF는 메타데이터를 처리하고 교환하는 공통의 문법으로 XML을 이용한다.

(2) RDF 스키마(Schema)

RDF 스키마는 RDF 데이터 모델에서 제공하는 문장들에 대한 인터프리터 정보를 제공한다. 다시 말해, RDF 스키마 메카니즘은 RDF 모델에서 사용을 위한 기본 타입 시스템을 제공한다[10].

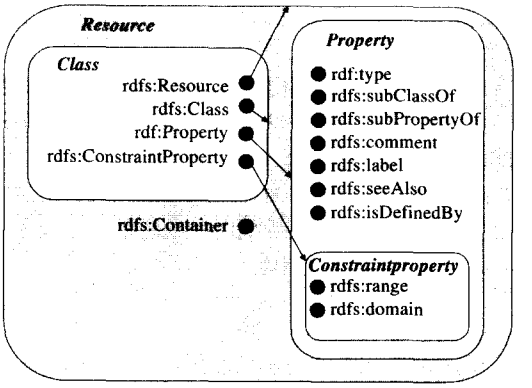
RDF 스키마에 대한 자원(resource)과 클래스들을 집합과 엘리먼트로써 표현하면 (그림 1)과 같다.

(그림 1)에서 둥근 직사각형은 클래스를 나타내고, 큰 점들은 각 자원을 나타내며, 화살표는 자원이 정의하는 클래스를 가리킨다. 그리고, 서브클래스(subclass)는 수퍼클래스(superclass)에 의해 둘러 쌓여 있다.

다음은 RDF 스키마의 자원에 대한 설명이다. 자원은 크게 클래스, 프로퍼티, 제한조건, 그리고, 설명 부분으로 구성된다.

① 클래스(Class)

RDF 표현에 의해 기술되는 모든 것은 자원이라 하며, RDF 스키마의 클래스가 된다. 이에 해당하는 클래스는 rdfs:Resource, rdfs:Class, rdf:Property 등이 있다.



(그림 1) 클래스와 자원들

② 프로퍼티(Property)

RDF 표현에서 프로퍼티는 애트리뷰트(attribute)나 관계성을 나타낸다. 즉, rdf:Property 클래스의 인스턴스로 클래스와 그들의 인스턴스 혹은 수퍼클래스 사이의 관계를 나타낸다.

㉠ rdf:type는 어떤 자원에 대해 한 클래스의 멤버가 되도록 하는 것으로 클래스의 멤버가 갖는 타입을 나타낸다.

㉡ rdfs:subClassOf는 클래스들 사이의 서브집합과 수퍼집합 관계를 나타낸다.

㉢ rdfs:subPropertyOf는 하나의 프로퍼티가 다른 프로퍼티의 특수화(specialization)가 되도록 규정한다.

㉣ rdfs:seeAlso는 주제(subject) 자원에 관한 정보를 포함하는 자원을 정의한다.

㉤ rdfs:isDefinedBy는 rdfs:seeAlso의 서브프로퍼티로써 주제 자원을 정의하는 자원을 가리킨다.

③ 제한조건(Constraints)

rdfs:ConstraintResource와 rdfs:ConstraintProperty는 각각 rdf:Resource와 rdf:Property의 서브클래스를 정의한 것으로 그것의 인스턴스가 제한조건을 포함한 RDF 스키마 구조나 프로퍼티이다.

rdfs:range와 rdfs:domain는 ConstraintProperty의 인스턴스로 각각 프로퍼티 값을 제한하거나 프로퍼티가 사용되는 클래스를 규정한다.

④ 설명(Documentation)

rdf:label는 자원 이름을 인간이 읽을 수 있는 형태

(자연어)로 제공하고 rdf:comment는 설명을 기술한다.

⑤ 그 외 자원

rdfs:Container는 rdfs:Class와 rdfs:subClassOf의 인스턴스이며, 종류로는 rdf:Bag, rdf:Seq, rdf:Alt가 있다. rdf:Bag는 자원이나 원소 값의 순서 없는 리스트를 나타내며, rdf:Seq는 순서 있는 리스트를 나타낸다. rdf:Alt는 프로퍼티의 값을 선택할 수 있도록 하는 자원이나 값의 리스트를 나타낸다.

3.2 UML 클래스 다이어그램

UML 클래스 다이어그램은 클래스, 관계성, 제한조건, 주석 등으로 구성된다[11].

(1) 클래스(Class)

클래스는 클래스 이름, 애트리뷰트 리스트(attribute list)와 오퍼레이션 리스트(operation list) 부분으로 구성된다. 각 클래스는 클래스에 대한 타입을 갖는다.

(2) 관계성(Relationships)

관계성은 크게 연관 관계성과 일반화 관계성으로 구별된다.

① 연관(Association) 관계성

연관성 관계는 클래스들간의 상호 관계를 나타내는 것으로 대표적인 예로 집단화 관계성이 있다. 집단화 관계성은 수퍼클래스와 서브클래스가 'part-of'관계이며 '◇'로 표기한다.

② 일반화(Generalization) 관계성

일반화 관계는 수퍼클래스와 서브클래스가 'is-a'관계이며, '△'로 표기한다. 즉, 수퍼클래스의 애트리뷰트와 오퍼레이션은 하위클래스에 상속된다.

(3) 제한조건

2개 이상의 집단화 관계에서 선택적으로 서브클래스가 발생하는 경우 'or' 관계를 갖는다. 여러 개의 집단화 관계를 점선으로 연결하고 '{or}'라는 제한조건을 준다. 그리고, '{ordered}'라는 제한조건은 원래 클래스 안의 요소들이 순서를 갖는 경우를 나타내는데, 본 논문에서는 다형성(polymorphism)에 의해 클래스를 구성하는 하위클래스들간에 순서 관계를 갖도록 한다[6].

(4) 주석

주석은 설명이나 다른 텍스트적 정보를 표시하기 위한 기호로 UML에 의해서 번역되지 않는다.

4. 변환 알고리즘

본 논문에서 RDF 스키마를 UML 클래스 다이어그램으로 변환시키기 위한 규칙, 모델링 함수, 형식 모델, 알고리즘, 시스템 및 적용 결과, 그리고, 비교분석을 기술한다.

4.1 변환 규칙

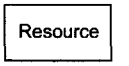
RDF 스키마의 집합을 UML 클래스 다이어그램 집합으로 변환시키기 위해 다음과 같이 규칙을 정의한다. 먼저 클래스와 애트리뷰트에 관련된 정의이다.

4.1.1 클래스와 애트리뷰트

RDF 스키마의 클래스는 UML 클래스 다이어그램의 클래스에 해당된다. 그리고, RDF의 프로퍼티와 제한조건은 각각 Private, Public 애트리뷰트를 나타낸다. 그러나, 일부 프로퍼티들은 애트리뷰트 대신 클래스들 간의 관계를 나타낸다. 이것은 [규칙 4], [규칙 5]에서 기술한다.

[규칙 1] RDF 스키마의 rdf:Description는 UML 클래스 다이어그램에서 클래스가 되고, rdf:type은 클래스의 타입이 된다.

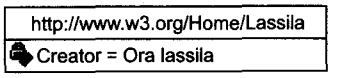
예 1)
```
<rdf:Description ID="Resource">
    <rdf:type resource="#Class"/>
</rdf:Description>
```



(그림 2) 예 1)의 UML 클래스 다이어그램

[규칙 2] RDF 스키마의 애트리뷰트를 나타내는 프로퍼티와 값은 Private 애트리뷰트와 이에 대한 초기 값(initial value)이 된다.

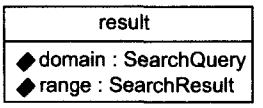
예 2)
```
<rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</S:Creator>
</rdf:Description>
```



(그림 3) 예 2)의 UML 클래스 다이어그램

[규칙 3] RDF 스키마의 제한조건(constraints)는 UML 클래스 다이어그램에서 클래스의 Public 애트리뷰트가 된다.

예 3)
```
<rdf:Property ID-"result">
    <rdf:domain rdf:resource="#SearchQuery"/>
    <rdfs:range rdf:resource="#SearchResult"/>
</rdf:Property>
```



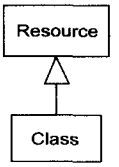
(그림 4) 예 3)의 UML 클래스 다이어그램

4.1.2 관계성

RDF 스키마에서 수퍼클래스와 서브클래스들 간의 관계를 나타내는 프로퍼티는 rdfs:subClassOf와 rdfs:Container가 있다. 이에 대한 규칙은 다음과 같다.

[규칙 4] RDF 스키마의 rdfs:subClassOf는 UML 클래스다이어그램의 일반화 관계성이 된다.

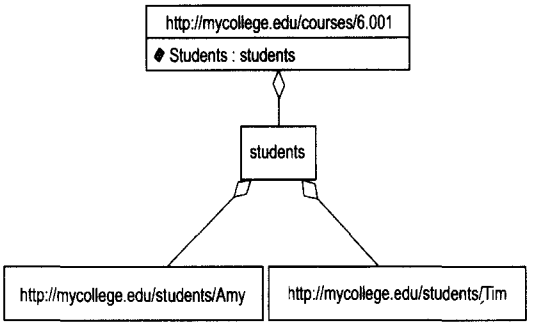
예 4)
```
<rdf:Description ID="Class">
    <rdf:type resource="#Class"/>
    <rdfs:subClassOf rdf:resource="#Resource"/>
</rdf:Description>
```



(그림 5) 예 4)의 UML 클래스 다이어그램

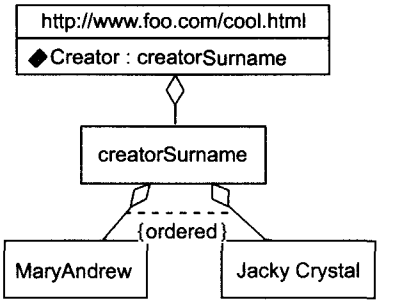
[규칙 5] RDF 스키마의 프로퍼티의 rdfs:Container는 UML 클래스 다이어그램의 집단화 관계성이 된다. rdfs:Container에서 rdf:Seq는 '{ordered}' 제한조건을 갖고, rdf:Alt는 '{or}' 제한조건을 갖는다. 이 때 클래스의 애트리뷰트 타입이 되는 하위클래스가 없는 경우는 자동 생성한다.

예 5)
```
<rdf:Description about="http://mycollege.edu/courses/6.001">
    <s:Students>
        <rdf:Bag>
            <rdf:li resource="http://mycollege.
            edu/students/Amy"/>
            <rdf:li resource="http://mycollege.
            edu/students/Tim"/>
        </rdf:Bag>
    </s:students>
</rdf:Description>
```



(그림 6) 예 5)의 UML 클래스 다이어그램

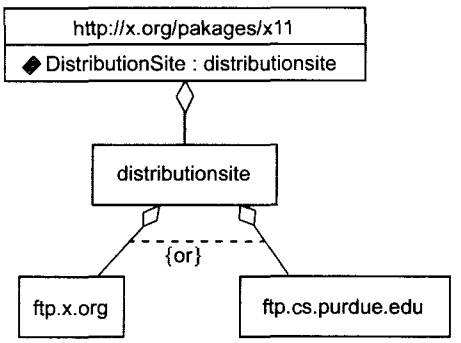
예 6)
```
<rdf:Description about="http://www.foo.com/cool.html">
    <dc:Creator>
        <rdf:Seq ID="creatorSurname">
            <rdf:li>Mary Andrew</rdf:li>
            <rdf:li>Jacky Crystal</rdf:li>
        </rdf:Seq>
    </dc:Creator>
</rdf:Description>
```



(그림 7) 예 6)의 UML 클래스 다이어그램

예 7)
```
<rdf:Description about="http://x.org/pakages/x11">
    <s:DistributionSite>
        <rdf:Alt>
```

```
            <rdf:li>ftp.x.org</rdf:li>
            <tdf:li>ftp.cs.purdue.edu</rdf:li>
        </rdf:Alt>
    </s:DistributionSite>
</rdf:Description>
```



(그림 8) 예 7)의 UML 클래스 다이어그램

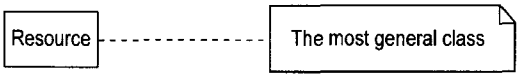
예 5)와 예 7)의 애트리뷰트 Students와 DistributionSite는 애트리뷰트 타입이 명시되어 있지 않으므로 이에 해당하는 클래스를 자동 생성한다. 이 클래스와 서브클래스들간에는 집단화 관계성을 갖는다.

4.1.3 주석

RDF 스키마에서 실행은 안되지만 클래스에 대한 설명을 덧붙일 수 있다. 일반적으로 rdfs:label은 클래스의 ID와 동일하므로 이에 대한 처리는 하지 않는다. 다음은 rdfs:comment에 대한 규칙이다.

[규칙 6] RDF 스키마의 rdfs:comment는 UML 클래스 다이어그램에서 주석(Note)이 된다.

```
예 8) <rdfs:Class ID="Resource"
          rdfs:label="Resource"
          rdfs:comment="The most general class/">
```



(그림 9) 예 8)의 UML 클래스 다이어그램

4.2 모델링 함수

UML 클래스 다이어그램에서 각 클래스의 오퍼레이션리스트 부분에 삽입(insert)되는 것으로, 클래스 다이어그램의 구조 파악을 용이하게 해준다. 이를 통해 RDF 스키마의 구조 변경 및 통합이 가능하다.

모델링 함수를 정의하기 위한 가정은 다음과 같다.

[가정 1] 연결자의 집합은 Con = {Inherit, Seq, Bag, Alt} 이다.

[가정 2] 클래스 다이어그램은 (S, p)의 쌍으로 이루어진다. 단, $\forall t \in S$에 대해 S는 클래스 집합이고, 함수 p 는 S에서 S로의 응용이다.

위의 [가정 2]에서 S의 루트(root) 클래스 t는 <표 1>과 같은 특성을 만족한다[12].

〈표 1〉 루트 클래스의 특성

함수	의미
$p(t) = t$	t의 부모(parent)는 t이다.
$\forall x \in S,\ \exists k \in N,\ p^k(x) = t,$ 단, k는 트리의 레벨, N은 정수 집합	t는 모든 x의 조상(ascendants)이다.

위의 가정들을 기반으로 모델링 함수들은 다음과 같이 정의된다.

[정의 1] $\forall t \in S$에 대해 함수 $p, c : S \rightarrow p(S)$는 각각 수퍼클래스와 서브클래스를 나타낸다. 단, $t_2 = c(t_1)$이면 $t_1 \neq t_2$이다.

즉, RDF 스키마의 rdf:subClassOf와 rdf:Container에서 수퍼클래스와 서브클래스 관계가 발생한다.

[정의 2] $\forall t \in S$에 대해 함수 $cons : S \rightarrow Con$는 연결자의 종류를 표시한다.

즉, RDF 스키마의 rdf:subClassOf는 cons(t) = inherit 이며, rdf:Container의 rdf:Seq, rdf:Bag, rdf:Alt는 cons(t) 값이 각각 Seq, Bag, Alt이다.

[정의 3] $\forall t \in S$에 대해 함수 $r : S \rightarrow Integer$는 cons(t) = Seq일 경우 서브클래스들의 순서를 표시한다.

즉, RDF 스키마의 rdf:Container가 rdf:Seq일 때 서브클래스는 r(t)의 값으로 1, 2, …를 순차적으로 갖는다.

[정의 4] $\forall t \in S$에 대해 함수 $m : S \rightarrow A$는 클래스에 속한 모든 애트리뷰트를 표시한다.

즉, RDF 스키마의 제한조건인 rdf:resource, rdf: range 등은 애트리뷰트가 된다.

4.3 형식 모델

RDF 스키마를 객체지향 데이터베이스 스키마로 변환시키려면 현재 상용되고 있는 데이터베이스에서 지원되지 않는 스키마 형태가 필요하다. 본 논문에서는 O2 데이터베이스를 기반으로 하여 스키마를 확장, rdf: Container에 대한 형식 모델을 정의한다. 이를 위해 다음과 같은 몇 가지 가정을 한다.

[가정 3] 기본 타입의 집합은 dom이고, 애트리뷰트의 집합은 $A = \{ a_1, a_2, ..., a_n \}$, 객체의 집합은 $O = \{ o_1, o_2, ..., o_n \}$, 클래스의 집합은 $C = \{ c_1, c_2, ..., c_n \}$이다.

[가정 4] 객체 집합 O의 값들은 $V = \{ v_1, ..., v_i, ..., v_n \}$로 표기한다. 단, V의 원소 v_i는 n_{il}이거나 dom 혹은 O의 각 원소들이 되고, 튜플(tuple) $[a_1: v_1, ..., a_n : v_k]$, 집합(set) $\{ v_1, ..., v_k \}$ 그리고, 리스트(list) $[v_1, ..., v_k]$도 V의 원소들이 된다[13].

[가정 5] 클래스의 집합 C의 각 원소들의 타입 집합은 $\tau = \{ \tau_1, ..., \tau_i, ..., \tau_n \}$이고, 이에 대한 의미(interpretation)는 $dom(\tau)$이다.

위의 가정들을 기반으로 rdf:Container의 rdf:Seq, rdf: Bag, rdf:Alt에 대해 각각 다음과 같이 정의한다.

[정의 7] 튜플 $[a_1 : \tau_1, ..., a_n : \tau_n]$은 Seq 타입이다. 즉, $dom([a_1 : \tau_1, ..., a_n : \tau_n]) = \{ [a_1 : v_1, ..., a_n : v_n, ..., a_{n+l} : v_{n+l}] \mid v_i \in dom(\tau_i), i = 1,...,n; l \geq 0 \}$이다[8].

[정의 8] 튜플 $[a_1 : \tau_1 \,\&\cdots\&\, a_n : \tau_n]$은 Bag 타입이다. 즉, $dom([a_1 : \tau_1 \,\&\cdots\&\, a_n : \tau_n]) = \{ [a_{i1} : v_{i1}, ..., a_{in} : v_{in}] \mid v_{im} \in dom(\tau_i), i = 1,...,n$; 단, 각 $i_1,...,i_m, ...,i_n$은 1,...,n의 순열(permutation)$\}$이다.

[정의 9] 집합 $\{ a_1 : \tau_1 +\cdots+ a_n : \tau_n \}$은 Alt 타입이다. 즉, $dom(\{ a_1 : \tau_1 +\cdots+ a_n : \tau_n \}) = \cup \{ dom([a_i : v_i]) \mid 1 \leq i \leq n \}$이다.

4.4 알고리즘

4.1절의 사상 규칙에 의해 RDF 스키마의 데이터 모델을 UML 클래스 다이어그램으로 변환시키는 알고리즘은 다음과 같다.

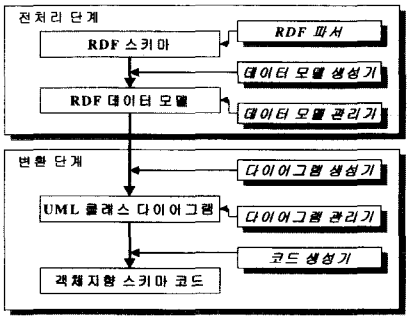
```
입력: RDF 데이터 모델
출력: UML 클래스 다이어그램

begin
{
 while ( RDF 데이터 모델 요소 )
 {
  if ( rdf:Description )
      make_class() // 클래스 생성
  else if ( rdf:comment )
      make_note() // 주석 첨가
 }

 while ( RDF 데이터 모델 요소 )
 {
  if ( 프로퍼티나 제한조건 )
  {
   if ( rdf:type )
       class_type() // 클래스 타입 정의
   else if ( rdf:subClassOf )
           make_inheritance() // 상속 관계
   else if ( rdf:Container )
     {
       if ( 애트리뷰트 타입이 없으면)
            make_class()
       make_aggregation() //집단화 관계
       switch ( rdf:Container )
        {
          case : rdf:Seq
                 order_constraint()
                 // 순서 관계 표시
          case : rdf:Alt
                 or_constraint()
                 // 선택적인 관계 표시
        }
     }
   else insert_attribute()
        // 애트리뷰트 삽입
  }
 }
}
end;
```

4.5 시스템 및 적용 결과

본 절에서는 시스템 구성과 적용 결과를 나타낸다.

4.5.1 시스템 구성 및 기능

전체 시스템 구성도는 (그림 10)과 같다.



(그림 10) 시스템 구성도

(1) 전처리 단계

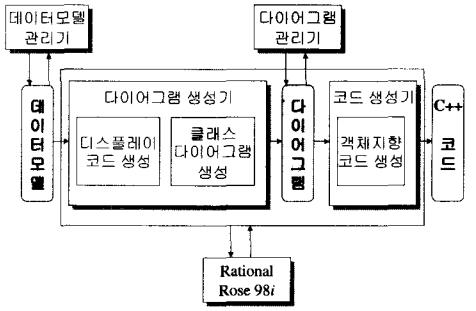
① RDF 파서 : XML을 이용한 RDF 스키마를 파싱하여 문법상의 오류가 없는지 조사한다.

② 데이터 모델 생성기 : 파싱된 RDF 스키마를 RDF 데이터 모델로 변환시킨다.

③ 데이터 모델 관리기 : RDF 데이터 모델에서 노드와 아크의 정보를 저장, 관리한다.

(2) 변환 단계

변환 단계에 대한 구조도는 (그림 11)과 같다.



(그림 11) 변환 단계의 구조도

위의 (그림 11)에서 각 구성 요소의 기능과 역할은 다음과 같다.

① 다이어그램 생성기 : 데이터 모델을 사상 알고리즘에 의해 클래스 다이어그램으로 변환시킨다.

② 다이어그램 관리기 : 클래스 다이어그램의 위치정보 등을 저장, 관리한다.

③ 코드 생성기 : 클래스 다이어그램에 대한 객체지향 코드를 생성한다.

4.5.2 적용 결과

다음은 본 시스템에 RDF 스키마를 입력시켜 데이터 모델과 클래스 다이어그램을 생성하고 이에 대한 C++ 코드를 생성한 결과이다.

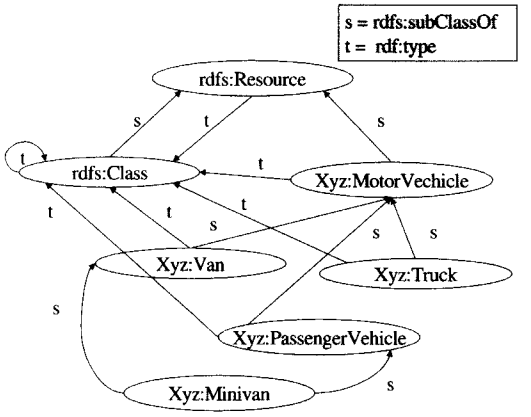
(1) RDF 스키마

본 시스템의 입력인 RDF 스키마는 XML 형태이다.

```
<rdf:RDF xml:lang="en"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf
                -schema-19990303#">

        <rdf:Description ID="MortorVechicle">
            <rdf:type  resource="http://www.w3.org/TR/
                1999/PR-rdf-schema-19990303#Class"/>
            <rdfs:subClassof  rdf:resource=http://www.
              w3.org/TR/1999/PR-rdf-schema-19990303
              #Resource"/>
        </rdf:Description>

        <rdf:Description ID="PassengerVechicle">
            <rdf:type  resource="http://www.w3.org/TR/
                1999/PR-rdf-schema-19990303#Class"/>
            <rdfs:subClassOf rdf:resource="#MotorVechicle"/>
        </rdf:Description>

        <rdf:Description ID="Truck">
            <rdf:type  resource="http://www.w3.org/TR/
                1999/PR-rdf-schema-19990303#Class"/>
            <rdfs:subClassOf rdf:resource="#MotorVechicle"/>
        </rdf:Description>

        <rdf:Description ID="Van">
            <rdf:type  resource="http://www.w3.org/TR/
                1999/PR-rdf-schema-19990303#Class"/>
            <rdfs:subClassOf rdf:resource="#MotorVechicle"/>
        </rdf:Description>

        <rdf:Description ID="MiniVan">
            <rdf:type  resource="http://www.w3.org/TR/
                1999/PR-rdf-schema-19990303#Class"/>
            <rdfs:subClassOf rdf:resource="#Van"/>
            <rdfs:subClassOf rdf:resource="#PassengerVechicle"/>
        </rdf:Description>
</rdf:RDF>
```

(2) 데이터 모델

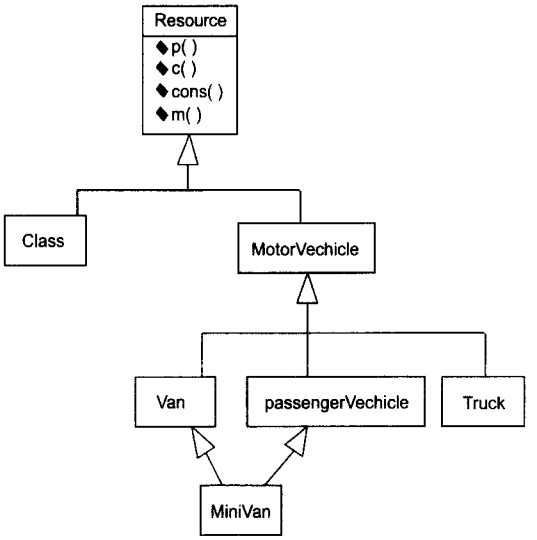
클래스 다이어그램으로 변환을 용이하게 하는 RDF 스키마의 데이터 모델은 (그림 12)이다.



(그림 12) RDF 데이터 모델

(3) UML 클래스 다이어그램

RDF 스키마의 데이터 모델을 입력으로 받아 UML 클래스 다이어그램을 생성한다. (그림 12)에 대해 생성된 UML 클래스 다이어그램은 (그림 13)과 같다.



(그림 13) UML 클래스 다이어그램

(4) C++ 코드

(그림 13)의 UML 클래스 다이어그램에 대해 C++ 코드가 생성되는데, 본 논문에서는 Resource 클래스에 대해 C++ 코드(Resource.cpp)와 헤더파일(Resource.h)을 제시한다.

① Resource.cpp

```
//## begin module%37AA3992010E.cm preserve=no
//            %X% %Q% %Z% %W%
//## end module%37AA3992010E.cm

//## begin module%37AA3992010E.cp preserve=no
//## end module%37AA3992010E.cp

//## Module: Resource%37AA3992010E; Pseudo Package body
//## Source file: C:\Program Files\Rational\Rose 98i\Resource.cpp

//## begin module%37AA3992010E.additionalIncludes preserve=no
//## end module%37AA3992010E.additionalIncludes

//## begin module%37AA3992010E.includes preserve=yes
//## end module%37AA3992010E.includes

// Resource
#include "Resource.h"
//## begin module%37AA3992010E.additionalDeclarations preserve=yes
//## end module%37AA3992010E.additionalDeclarations


// Class Resource

Resource::Resource()
  //## begin Resource::Resource%.hasinit preserve=no
  //## end Resource::Resource%.hasinit
  //## begin Resource::Resource%.initialization preserve=yes
  //## end Resource::Resource%.initialization
{
  //## begin Resource::Resource%.body preserve=yes
  //## end Resource::Resource%.body
}

Resource::Resource(const Resource &right)
  //## begin Resource::Resource%copy.hasinit preserve=no
  //## end Resource::Resource%copy.hasinit
  //## begin Resource::Resource%copy.initialization preserve=yes
  //## end Resource::Resource%copy.initialization
{
  //## begin Resource::Resource%copy.body preserve=yes
  //## end Resource::Resource%copy.body
}

Resource::~Resource()
{
  //## begin Resource::~Resource%.body preserve=yes
  //## end Resource::~Resource%.body
}


Resource & Resource::operator=(const Resource &right)
{
  //## begin Resource::operator=%.body preserve=yes
  //## end Resource::operator=%.body
}
```

```
int Resource::operator==(const Resource &right) const
{
  //## begin Resource::operator==%.body preserve=yes
  //## end Resource::operator==%.body
}

int Resource::operator!=(const Resource &right) const
{
  //## begin Resource::operator!=%.body preserve=yes
  //## end Resource::operator!=%.body
}

//## Other Operations (implementation)
void Resource::p ()
{
  //## begin Resource::p%37AFBE0003DE.body preserve=yes
  //## end Resource::p%37AFBE0003DE.body
}

void Resource::c ()
{
  //## begin Resource::c%37AFBE0B0172.body preserve=yes
  //## end Resource::c%37AFBE0B0172.body
}

void Resource::cons ()
{
  //## begin Resource::cons%37AFBE120348.body preserve=yes
  //## end Resource::cons%37AFBE120348.body
}

void Resource::m ()
{
  //## begin Resource::m%37AFBE1A0136.body preserve=yes
  //## end Resource::m%37AFBE1A0136.body
}

// Additional Declarations
  //## begin Resource%37AA3992010E.declarations preserve=yes
  //## end Resource%37AA3992010E.declarations

//## begin module%37AA3992010E.epilog preserve=yes
//## end module%37AA3992010E.epilog
```

② Resource.h

```
//## begin module%37AA3992010E.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module%37AA3992010E.cm

//## begin module%37AA3992010E.cp preserve=no
//## end module%37AA3992010E.cp

//## Module: Resource%37AA3992010E; Pseudo Package specification
//## Source file: C:\Program Files\Rational\Rose 98\Resource.h

#ifndef Resource_h
#define Resource_h 1

//## begin module%37AA3992010E.additionalIncludes preserve=no
//## end module%37AA3992010E.additionalIncludes

//## begin module%37AA3992010E.includes preserve=yes
//## end module%37AA3992010E.includes

//## begin module%37AA3992010E.additionalDeclarations preserve=yes
//## end module%37AA3992010E.additionalDeclarations


//## begin Resource%37AA3992010E.preface preserve=yes
//## end Resource%37AA3992010E.preface

//## Class: Resource%37AA3992010E
//## Category: <Top Level>
//## Persistence: Transient
//## Cardinality/Multiplicity: n

class Resource
{
  //## begin Resource%37AA3992010E.initialDeclarations preserve=yes
  //## end Resource%37AA3992010E.initialDeclarations

  public:
    //## Constructors (generated)
      Resource();

      Resource(const Resource &right);

    //## Destructor (generated)
      ~Resource();

    //## Assignment Operation (generated)
      Resource & operator=(const Resource &right);

    //## Equality Operations (generated)
      int operator==(const Resource &right) const;

      int operator!=(const Resource &right) const;

    //## Other Operations (specified)
      //## Operation: p%37AFBE0003DE
      void p ();

      //## Operation: c%37AFBE0B0172
      void c ();

      //## Operation: cons%37AFBE120348
      void cons ();

      //## Operation: m%37AFBE1A0136
      void m ();

    // Additional Public Declarations
      //## begin Resource%37AA3992010E.public preserve=yes
      //## end Resource%37AA3992010E.public

  protected:
    // Additional Protected Declarations
      //## begin Resource%37AA3992010E.protected preserve=yes
      //## end Resource%37AA3992010E.protected

  private:
    // Additional Private Declarations
      //## begin Resource%37AA3992010E.private preserve=yes
      //## end Resource%37AA3992010E.private

  private: //## implementation
```

```
    // Additional Implementation Declarations
      //## begin Resource%37AA3992010E.implementation preserve=yes
      //## end Resource%37AA3992010E.implementation

};

//## begin Resource%37AA3992010E.postscript preserve=yes
//## end Resource%37AA3992010E.postscript

// Class Resource

//## begin module%37AA3992010E.epilog preserve=yes
//## end module%37AA3992010E.epilog


#endif
```

4.6 비교 분석

본 연구의 모델링 방법의 평가를 위해 RDF 데이터 모델과 XML 문서를 모델링하는 [8, 9] 연구들과 비교, 분석해 본다. <표 2>는 각 RDF 자원에 대한 모델링 방법들에 대한 비교이다.

〈표 2〉 RDF 모델링 방법 비교

<table>
<tr><th colspan="2">RDF 자원</th><th>Elm트리다이어그램과 [9]의 모델링</th><th>RDF 데이터 모델링</th><th colspan="2">본 연구의 모델링</th></tr>
<tr><td colspan="2">rdf:Description</td><td rowspan="10">모두 DTD에서 선언된 엘리먼트이므로 클래스나 직사각형으로 표시</td><td>노드</td><td colspan="2">클래스</td></tr>
<tr><td colspan="2">rdf:type</td><td>아크</td><td colspan="2">클래스 타입</td></tr>
<tr><td colspan="2">Propery types: Description (사용자 정의, 값이 스트링일 경우)</td><td>아크 (직사각형)</td><td colspan="2">Private 애트리뷰트</td></tr>
<tr><td colspan="2">rdf:domain</td><td>아크</td><td colspan="2">Public 애트리뷰트</td></tr>
<tr><td colspan="2">rdfs:range</td><td>아크</td><td colspan="2">Public 애트리뷰트</td></tr>
<tr><td colspan="2">rdf:subClassOf</td><td>아크</td><td colspan="2">일반화 관계성</td></tr>
<tr><td rowspan="2">rdfs:Container (rdf:Bag)</td><td>rdf:Seq</td><td rowspan="2">아크</td><td rowspan="2">집단화 관계성</td><td>'{order}' 제한조건</td></tr>
<tr><td>rdf:Alt</td><td>아크</td><td>'{or}' 제한조건</td></tr>
<tr><td colspan="2">rdf:comment</td><td>아크</td><td colspan="2">주석</td></tr>
</table>

<표 2>에서 보면, RDF 자원에 대해 XML DTD 모델링 방법인 Elm 트리 다이어그램과 [9]의 모델링 방법은 RDF 자원의 종류와 의미와는 무관하게 엘리먼트나 클래스에 해당하는 직사각형으로 표시하게 된다. 따라서, 이 경우들은 각각에 대해 클래스들이 생성되나, 그들 간의 관계를 제대로 표현해 낼 수 없는 단점이 있다. 그리고, RDF 모델링의 경우에는 노드의 경우를 제외하고 모든 관계를 아크에 라벨을 붙여 나타내고 있다. 따라서, 노드들 간의 계층 구조를 파악할 수 없을 뿐만 아니라 아크의 종류가 많을수록 노드들 간의 관계를 제대로 파악하기가 무척 어려운 실정이다.

이에 비해 본 연구의 모델링은 RDF 자원의 시맨틱에 따라 UML을 클래스 다이어그램을 사상시켜 클래스들간의 계층 구조를 명확히 나타내며, 객체 지향 스키마로 쉽게 변환이 가능하도록 한 장점을 갖는다.

5. 결론 및 향후 연구과제

본 논문은 웹 문서에 대한 메타데이터 형태인 RDF 스키마를 객체지향 모델링 방법인 UML 클래스 다이어그램으로 변환시키는 규칙과 알고리즘을 제안하였다. 즉, RDF 스키마를 시맨틱적으로 분석하여 UML 클래스 다이어그램의 클래스, 애트리뷰트, 일반화 및 집단화 관계성, 그리고, 제한조건에 사상시켰다.

그리고, RDF 스키마에 대한 UML 클래스 다이어그램의 구조 파악, 변경 그리고 통합 등을 하기 위해 각 클래스의 오퍼레이션 리스트 부분에 오퍼레이션을 삽입하였으며, RDF 스키마에 대한 객체지향 데이터베이스 스키마 형태인 형식 모델을 제안하였다. 따라서, 본 논문은 RDF 스키마를 객체지향 데이터베이스에 저장하고 관리하기 위한 기반이 될 것이다.

향후 연구 과제로는 RDF 이외에 메타데이터인 MCF (Meta Content Format), WF(Warick Framework)에 대한 객체 모델링과 객체 지향 스키마의 자동 생성을 위한 알고리즘을 제안하는 것이다.

참 고 문 헌

[1] 정효택, 양영종, 김순용, 이상덕, 최윤철, "Web상의 전자문서를 위한 메타데이터 모델의 제안 및 관리 시스템의 개발", 정보처리학회 논문지, 제5권, 제4호, pp.924-940, 1998. 4.

[2] Natanya Pitts-Moultis, Cheryl Kirk, "XML Black Book," The Coriolis Group Inc., 1999.

[3] W3C, "A Discussion of the Relationship Between RDF-Schema and UML," August 1998, http://www.w3.org/TR/1998/NOTE-rdf-uml-19980804.

[4] W3C, "Resource Description Framework(RDF) Model and Syntax Specification", August 1998,

http://www.w3.org/TR/1998/WD-rdf-syntax-19980819/.

[5] 박인호, 한에노, 정은주, 김은정, 배종민, 강현석, 김완석, "XOMT:SGML DTD 설계를 위한 객체 다이어그래밍 기법", 정보과학회 논문지(C), 제3권, 제3호, pp.228-237, 1997. 6.

[6] 하얀, 황용주, 김용성, "SGML DTD로부터 UML 클래스 다이어그램으로의 사상 알고리즘", 정보과학회논문지(B), 제26권, 제4호, pp.508-520, 1999. 4.

[7] 신명기, 김용진, "W3C에서의 차세대 웹 표준 활동 동향", 정보처리학회지, 제6권, 제3호, pp.7-17, 1999. 5.

[8] ArborText Inc. "Data modeling Report prepared for:W3C XML Specification DTD("XML spec")," September 1998, http://www.oasis-open.org/cover/xml-report-19980910.html.

[9] 채원석, 하얀, 김용성, "UML 클래스 다이어그램을 이용한 XML 문서 구조 다이어그래밍", 정보처리논문지, 제6권, 제10호, pp.2670-2679, 1999. 10.

[10] W3C, Resource Description Framework(RDF) Schema Specification, March 1999, http://www.w3.org/TR/Tr-rdf-schema/.

[11] James rumbaugh, Ivar Jacobson, Grady Booch, "The unified modeling language reference manual," Addison Wesley Longman Inc., 1999.

[12] E. Akpotsui, V. Quint, C. Roisin. "Type Modelling for Document Transformation in Structured Edition Systems," Mathematical and Computer Modelling, Vol.25, No.4, pp.1-19, 1997, http://www.oasis-open.org/cover/.

[13] V. Christophides, S. Abiteboul, S. Cluet, M. Scholl, "From Structured Documents to Novel Query Facilities," In Poc. ACM SIGMOD Intl. Conf. Management of Data, pp.313-324, 1994. 5.

[14] Bruce Power Douglass, "Real-Time UML Developing Efficient Objects for Embedded Systems," Addison-Wesley Longman Inc., 1998.

[15] Craig Larman, "Applying UML and PATTERNS: An Introduction to Object-Oriented Analysis and Design," Prentice-Hall, 1998.

[16] Renato Iannella, "An Idiot's Guide to the Resource Description Framework," The New Review of Information Networking, Vol.4, March 1998, http://aechive.dstc.edu.au/RDU/reports/RDF-Idiot/.

[17] 김태석, 박철제, 임환섭, "차세대 웹 서비스 기술 개발 동향", 정보처리학회지, 제6권, 제3호, pp.18-24, 1999. 5.

[18] 나홍석, 채진석, 김창화, 백두권, "차세대 웹 상에서의 문서 교환 및 검색을 위한 프레임워크", 정보처리학회지, 제6권, 제3호, pp.52-61, 1999. 5.

이 미 경

e-mail : mklee@sjpc.ac.kr

1988년 전북대학교 전산통계학과(이학사)

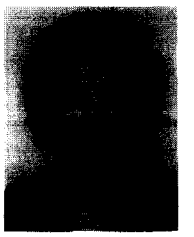
1991년 전북대학교 전산통계학과(이학석사)

1999년 전북대학교 전산통계학과 박사수료

1988년~1995년 전주영생여상고 교사

1995년~현재 서울정수기능대학 정보통신설비과 조교수

관심분야 : 정보 검색, 지식 공학, 전자 상거래, 컴퓨터 교육



하 얀

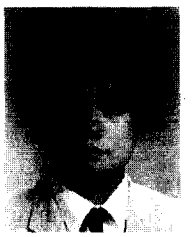
e-mail : yanha@cs.chonbuk.ac.kr

1992년 덕성여자대학교 전산학과 졸업(이학사)

1994년 이화여자대학교 전자계산 교육전공(교육학석사)

2000년 2월 전북대학교 전산통계학과 졸업예정(이학박사)

관심분야 : SGML/XML, 객체지향 모델링, 전자 도서관, 정보 검색, 지능형 교육 시스템 등



김 용 성

e-mail : yskim@moak.chonbuk.ac.kr

1978년 고려대학교 수학과 졸업(이학사)

1984년 광운대학교 전산학과(이학석사)

1992년 광운대학교 전산학과(이학박사)

1985년~현재 전북대학교 컴퓨터과학과 교수

1996년~1998년 한국학술진흥재단 전문위원

관심분야 : 디지털 도서관, 정보검색, 인터넷 기반 정보검색, 멀티미디어 시스템, 인공지능 등