



인 터 넷 진 화 의 열 쇠

# 온톨로지

웹 2.0에서 3.0으로

노상규·박진수 공저

god's toy business, 2007

SNU IDB laboratory

# Ontology Contents

## MODULE 1 온톨로지의 개념 및 응용

- Chapter 1 온톨로지 개요
  - 1. 온톨로지의 유래
  - 2. 분류와 개념화 과정
  - 3. 컴퓨터 온톨로지
- Chapter 2 온톨로지의 분류와 용도
  - 1. 온톨로지의 분류
  - 2. 온톨로지의 사용 목적과 중요성
  - 3. 온톨로지와 시맨틱 웹
- Chapter 3 온톨로지 구축 프로젝트
  - 1. 사이크(Cyc)
  - 2. 워드넷(WordNet)
  - 3. 전자거래문서
  - 4. 통합의학언어시스템
  - 5. 오픈 디렉터리 프로젝트
  - 6. 국제상품분류코드(UNSPSC)
- Chapter 4 온톨로지 적용 분야
  - 1. 전자상거래 분야
  - 2. 의료 분야
  - 3. 법률 분야
  - 4. 검색 서비스 분야
  - 5. 문화컨텐츠 분야

## MODULE 2 온톨로지 언어와 구축도구

- Chapter 5 온톨로지 언어
  - 1. 온톨로지 언어의 발전 과정
  - 2. 인공지능 기반의 온톨로지 언어
  - 3. 온톨로지 마크업 언어
- Chapter 6 RDF(S): RDF와 RDF Schema
  - 1. XML과 RDF
  - 2. RDF
  - 3. RDF Schema
  - 4. RDF(S)의 한계점
- Chapter 7 OWL(Web Ontology Language)
  - 1. OWL의 기본 요소: 클래스와 속성
  - 2. OWL의 새로운 기능
  - 3. 세 종류의 OWL
  - 4. OWL 예제
- Chapter 8 토픽맵(Topic Maps)과 XTM(XML Topic Maps)
  - 1. 토픽맵(Topic Maps) 개념
  - 2. 토픽맵 구성요소
  - 3. XTM 예제
- Chapter 9 온톨로지 틀
  - 1. 온톨로지 틀의 분류
  - 2. 온톨로지 개발 틀
  - 3. 주요 온톨로지 틀 요약 정보

# Chapter 6 RDF(S): RDF와 RDF Schema

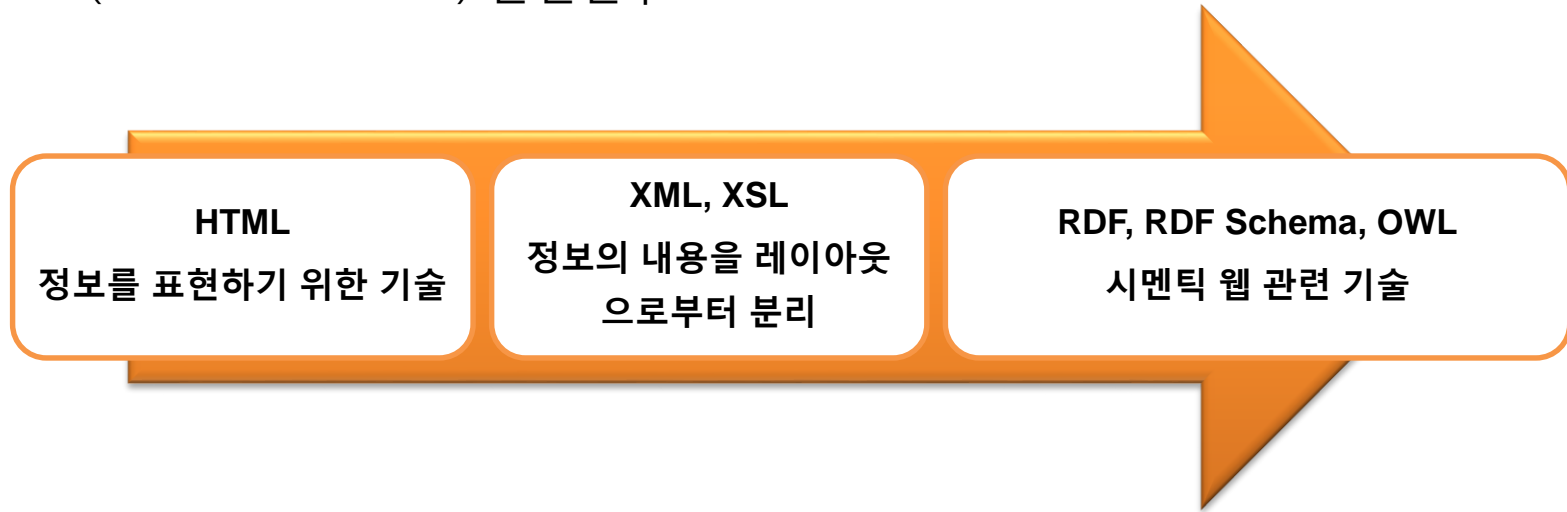
- RDF(Resource Description Framework)
  - 단순한 Triple 형태로 웹 자원을 기술
  - W3C에 의해 개발, 1999년 W3C 권고안 설정
  
- RDF Schema
  - RDF를 프레임 지식표현 패러다임으로 확장한 언어
  - 객체지향 모델링과 비슷한 도메인 구성에 대한 표현력 제공
  - W3C에 의해 개발, 2004년 W3C 권고안 설정
  
- $\text{RDF(S)} = \text{RDF} + \text{RDF Schema}$
- $\text{RDF(S)} + \text{기술논리 지식표현 패러다임} \rightarrow \text{OIL, DAML+OIL, OWL}$

# Chapter 6 RDF(S): RDF와 RDF Schema

- 1. XML과 RDF
- 2. RDF
  - 2.1 RDF 데이터 모델
  - 2.2 RDF 그래프와 코딩 예
- 3. RDF Schema
  - 3.1 RDF와 RDF Schema
  - 3.2 기본 요소: 클래스와 속성
  - 3.3 RDF Schema 코딩 예
- 4. RDF(S)의 한계점

# 1. XML과 RDF

## ■ WWW(World Wide Web) 관련연구



## ■ XML: W3C의 후원으로 결성된 [XML Working Group](#)에 의해 1996년 제안

- XML 태그의 역할
  - ▶ 정보검색 및 정렬을 위해 정보를 구분하는 이름을 사용자가 임의로 작성
- 텍스트 파일로 저장
- 여러 기종과 OS에 대해 이식성 높음
- 문서의 구조를 정의하는 스키마 사용, 우수한 호환성
- 내용과 디자인을 완전히 분리

# 1. XML과 RDF: XML의 Limitation

- 정보의 의미를 명확히 전달해주는 매커니즘을 제공하지 않음
  - <전화번호> = <전화> = <Telephone> = <TelephoneNumber>..
- 태그 해석규칙이 없어 태그 간의 의미의 연관성을 컴퓨터가 추론하기 어려움
- HTML보다 유연하나 Semantic Web을 위한 표준 언어로는 부족함

예) “김동건이 ‘생각하는 컴퓨터’를 썼다.”

```
<책 이름="생각하는 컴퓨터">  
  <저자>김동건</저자>  
</책>
```

```
<저자 이름="김동건">  
  <썼다>생각하는 컴퓨터</썼다>  
</저자>
```

```
<책>  
  <저자>김동건</저자>  
  <제목>생각하는 컴퓨터</제목>  
</책>
```

# 1. XML과 RDF: RDF의 등장

## ■ XML의 문제 해결을 위해 제시된 기술

- '자원(주어부)-속성(서술부)-속성값(목적부)'을 하나의 기본 단위로 취급
- 예) "김동건이 '생각하는 컴퓨터'를 썼다." → Triple을 기본 단위로 연결해 표현 가능



## ■ RDF/XML

- 위와 같은 RDF 데이터 모델을 컴퓨터가 이해할 수 있는 기계적 언어로 표현
- 일반적으로 RDF 데이터 모델은 XML Syntax를 사용
  - ▶ XML에 대한 오랜 연구 축적, 분산환경에 적합한 특성 높은 호환성 등에 기인
- RDF데이터 모델의 표현을 위한 XML Syntax: RDF/XML

# 1. XML과 RDF: 데이터 모델 관점에서의 비교

XML	RDF
순서가 중요한 트리구조	트리플구조(Subject/Predicate/Object)
검색하기 복잡	검색이 용이(독립적 트리플 집합)
Tag 배치 순서 다르면 다른 문서로 인식	각각 트리플이 독립적 집합 - 순서 상관 없음
메타데이터 표현의 유연성 낮음 노드(Tag)가 문서에 포함되어 인덱싱	메타데이터 표현의 유연성 높음 노드가 URIref 를 갖는 자원
XML Schema: 문서의 구문적 해석이 주요 기능	RDF Schema: 주로 의미적 해석에 사용



# 1. XML과 RDF: URI(Uniform Resource Identifier)

- **URI:** 웹 상에 존재하는 자원을 지칭하는 스트링 표준형식
  - URL(Uniform Resource Location)
  - URN(Uniform Resource Name)
- **URL:** 웹 페이지 같은 자원에 접근할 때 사용되는 실제 네트워크 경로
  - '프로토콜://파일이 저장된 서버의 DNS이름/디렉터리/파일 이름'으로 구성
  - 예) `http://www.snu.ac.kr/index.html`
- **URN:** 임의의 자원을 가리키는 영속적이고 고유한 이름
  - 자원이 저장된 위치와 무관
  - 'urn: NID(Namespace ID) : NSS(Namespace Specific String)'으로 구성
  - NSS는 NID안에서 유일해야 함
    - ▶ 예) ISBN 번호 3960152782를 가진 책     `urn: isbn: 3960152782`
    - ▶ 예) 대한민국 국민, NSS로 주민등록번호 사용     `urn: korean:901010-1234567`

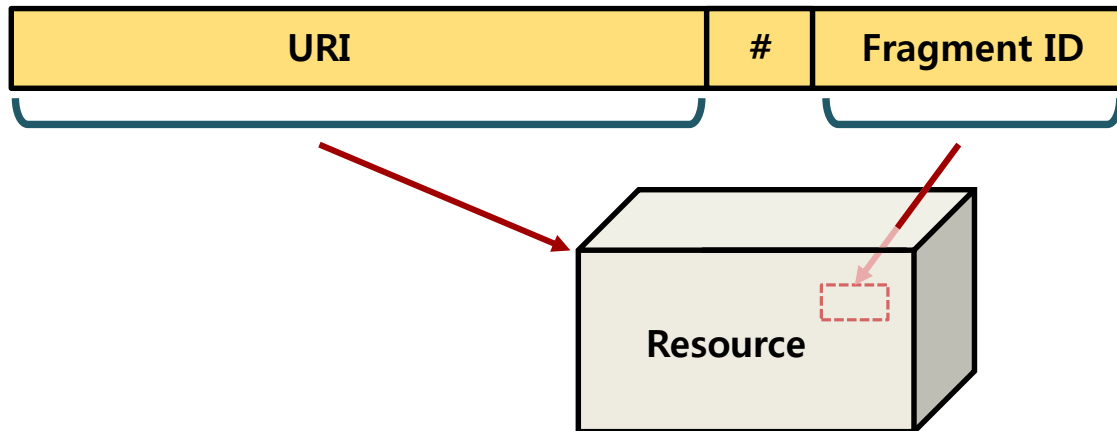
# 1. XML과 RDF: URIref(URI Reference)

- RDF는 자원에 대한 식별자로 URIref 사용

## 1. Absolute URIref: Context Independent

- URI + '#' + Fragment Identifier(단편식별자) 표시

- 단편식별자 : 네임 스페이스에 해당하는 앞의 URI 안에서 효력이 있는 자원의 식별자



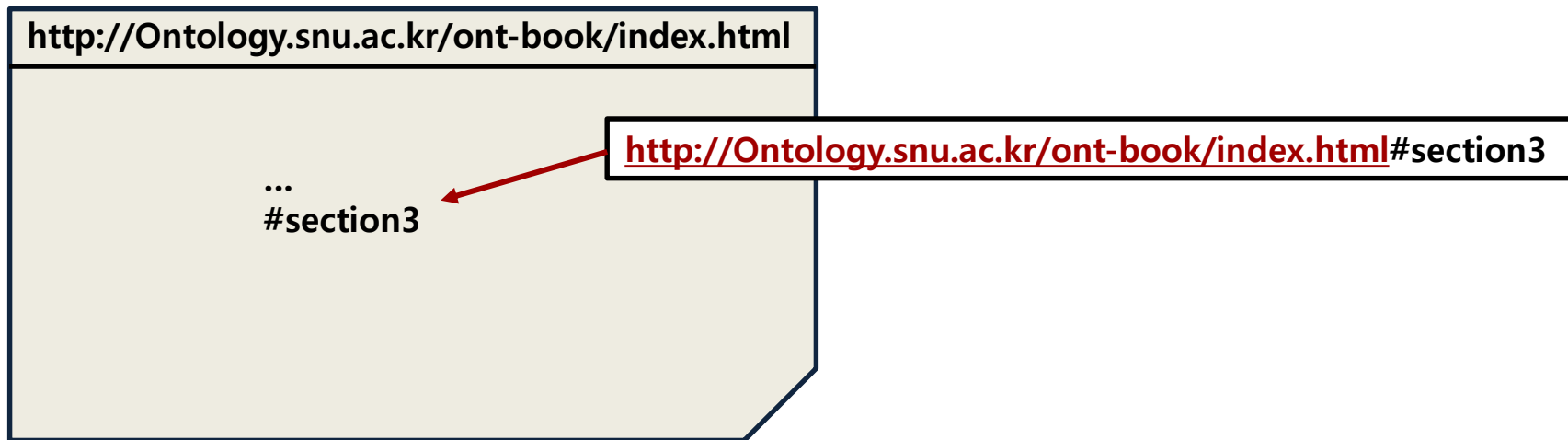
- 예) <http://Ontology.snu.ac.kr/ont-book/index.html#section3>

- RDF에서 기술하는 자원이 URL로 표현되는 전자문서일 때
- 이 문서의 URL(즉, 단순한 URI)가 즉 URIref가 되는 것

# 1. XML과 RDF: URIref(URI Reference)

## 2. Relative URIref: Context Dependent

- Absolute URIref의 축약형, URIref의 URI 부분 생략
- 예) 'http://Ontology.snu.ac.kr/ont-book/index.html' 이라는 문서 안에서
  - '#section3'라는 Relative URIref가 있으면,
  - Absolute URIref 'http://Ontology.snu.ac.kr/ont-book/index.html#section3'로 해석



# 1. XML과 RDF: URIref(URI Reference)

- Context를 고려 → 해당 문서의 Base URI가 무엇인지 찾는 것
- Base URI를 지정 안하면 – 해당문서의 URI가 Base URI
- Base URI를 지정 하면 – Attribute 'xml:base' 사용하여 지정
  - 예) `<rdf:RDF xml:base="http://www.Ontologytech.com/2007/01/products">`  
이 문서에 Relative URIref '#item101'가 있으면, 문서의 URI와 상관없이  
'http://www.Ontologytech.com/2007/01/products#item101'로 해석



# Chapter 6 RDF(S): RDF와 RDF Schema

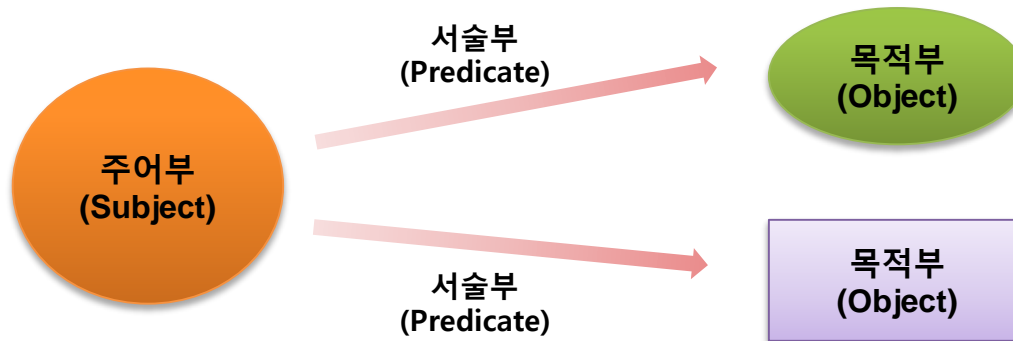
- 1. XML과 RDF
- 2. RDF
  - 2.1 RDF 데이터 모델
  - 2.2 RDF 그래프와 코딩 예
- 3. RDF Schema
  - 3.1 RDF와 RDF Schema
  - 3.2 기본 요소: 클래스와 속성
  - 3.3 RDF Schema 코딩 예
- 4. RDF(S)의 한계점

# 2.1 RDF Data Model

## ■ RDF

- 모든 사물/개념을 Resource로 본다
- ID로 URIref를 사용한다
  - ▶ "http://Ontology.snu.ac.kr/ontbook/plant/fruits#apple"
  - ▶ "http://dictionary.snu.ac.kr/language/words#apology"
- Resource의 속성, Resource간의 관계를 기술하는 데이터 모델

## ■ 기본단위: Triple 구조 (자원(주어부)-속성(서술부)-속성값(목적부)로 이루어진 서술문)



# 2.1 RDF Data Model: Triple

- Subject (주어부)
  - 문장의 주어 역할
  - Predicate과 Object로 기술되는 Resource를 URIref로 나타냄
- Predicate or Property (서술부)
  - Subject의 Resource를 설명하는 속성이나 Resource간의 관계 표현
  - Predicate도 특수한 형태의 Resource이므로 URIref로 명시
- Object (목적부)
  - 문장의 목적어 역할
  - Resource로 표현되어 URIref를 명시하거나
  - Resource외에 Literal Data 가능(strings, integer..)

## 2.1 RDF Data Model: Triple

- '동건의 나이는 34세이다.'

자원 (Subject)
동건

속성 (Predicate)
나이

속성값 (Object)
34

- '동건은 경영학을 전공했다.'

자원 (Subject)
동건

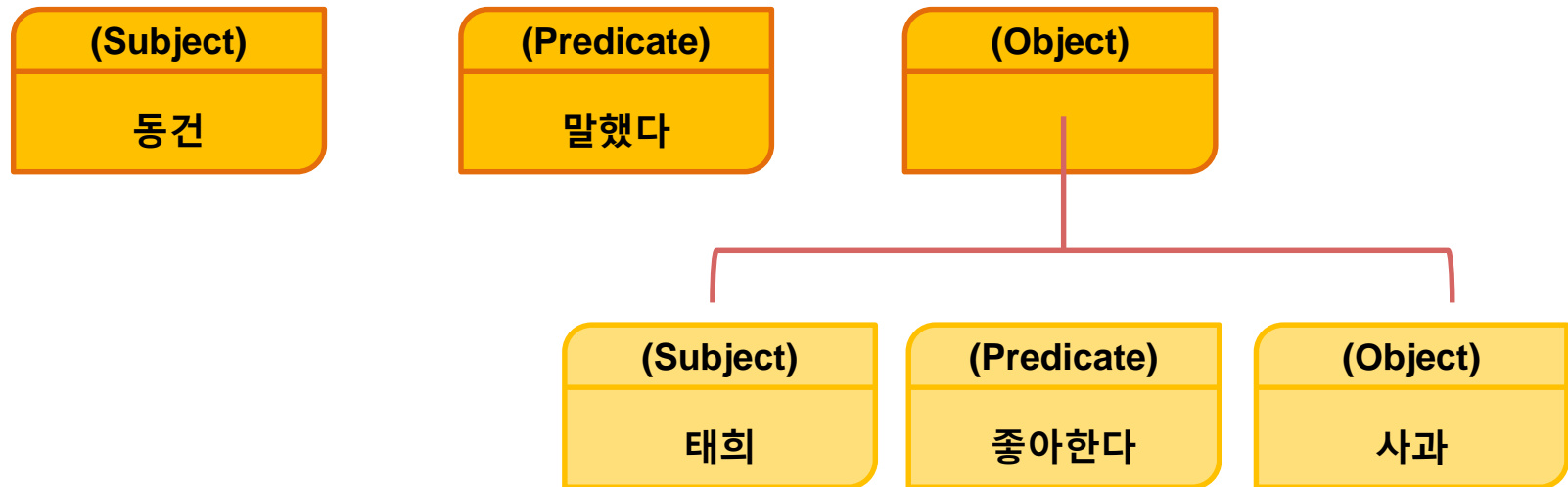
관계 (Predicate)
전공했다

자원 (Object)
경영학



## 2.1 RDF Data Model: Triple

- **Reification (구체화)**: 하나의 서술문에 대한 서술문을 추가하여 모델링
- 동건은 태희가 사과를 좋아한다고 말했다.
  - Subject: 동건, Predicate: 말했다, Object: 태희가 사과를 좋아한다



## 2.1 RDF Data Model: Limitation

- RDF 속성은 양쪽에 인수가 두 개인 이진 속성
  - 인수가 두개보다 많은 상황에는 한 번에 표현하기 어려움
- Reification 매커니즘: powerful expression, but complicated
- RDF의 XML 기반 Syntax: computer-friendly, not human-friendly
  
- But ! RDF는 시맨틱 웹의 기본 계층으로서의 충분한 표현력을 제공
- 다양한 온톨로지 툴들을 이용하면 RDF Syntax를 정확히 몰라도 RDF 편집 가능

## 2.2 RDF 그래프 코딩 예

### ■ Encoding RDF Graphs

- RDF/XML
  - ▶ XML syntax for RDF
  - ▶ Namespaces in XML, the XML Information Set and XML Base.
- *Notation 3* (also known as *N3*)
  - ▶ an assertion and logic language which is a superset of RDF
  - ▶ adding formulae (literals which are graphs themselves), variables, logical implication, and functional predicates
- N-Triple
  - ▶ line-based, plain text format
  - ▶ Subset of N3

## 2.2 RDF 그래프 코딩 예: RDF/XML

### ENTITY


- Entity 정의는 DOCTYPE(Document Type Declaration) 안에 포함
- 긴문장을 지정된 짧은 문장으로 대체
- '&지정된 문자열;'

```
<?xml version="1.1">
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"> ]>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ...
  xmlns:processor="http://www.intel.com/core2duo#">

  <rdf:Description rdf:about="MacBookAir">
    <computer:processor>IntelCore2Duo</computer:processor>
    <computer:weight rdf:datatype="&xsd;decimal">3.0</computer:weight>
  </rdf:Description>

</rdf:RDF>
```



```
<computer:weight http://www.w3.org/2001/XMLSchema#decimal>3.0</computer:weight>
```

## 2.2 RDF 그래프 코딩 예: RDF/XML

- Attribute내에서 긴문장을 짧은문장으로 대체는 Entity 방식

```
<rdf:Description rdf:about="http://Ontology.snu.ac.kr/ont-book#Kimdk">
```



```
<!DOCTYPE rdf : RDF [  
  <!ENTITY ontbook "http://Ontology.snu.ac.kr/ont-book#"> ]>  
  
<rdf:Description rdf:about="&ontbook;Kimdk">
```

‘rdf:about’라는 Attribute에 대한 값을 ‘ontbook’이란 Entity를 사용해 표현

- XML Tag Name안에서의 축약은 QName(Qualified Name) 방식

- Syntax: [Prefix]+ [':'] + [Local Name]
- <?xml version='1.0'?>  
 <doc xmlns: ontbook = "http://Ontology.snu.ac.kr/ont-book#">  
 < ontbook:owns />  
 </doc>

"ontbook:owns" → <http://Ontology.snu.ac.kr/ont-book#owns>

## 2.2 RDF 그래프 코딩 예: RDF/XML

### ■ 예문

- "김동건은 온톨로지텍이란 회사를 소유하고 있다."
- "온톨로지텍의 홈페이지 주소는 <http://www.ontologytech.com/~ont> 이다"
- "김동건은 경영학을 전공했다."
- "김동건의 나이는 34세이다."

### ■ RDF Graph



## 2.2 RDF 그래프 코딩 예: RDF/XML

```
<!DOCTYPE rdf : RDF [  
  <!ENTITY ontbook "http://Ontology.snu.ac.kr/ont-book#" > ]>
```

Attribute표현의  
축약을 위해

1. <rdf:RDF

2.     xmlns : rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

3.     xmlns : ontbook = "http://Ontology.snu.ac.kr/ont-book#">

4.     <rdf:Description rdf:about = "&ontbook; Kimdk">

5.         <ontbook:majorsIn rdf : resource = "&ontbook ; Mangement" />

6.         <ontbook:owns>

7.             <rdf:Description rdf:about = "&ontbook ; OntologyTech">

8.                 <ontbook : hasHomepage rdf : resource=http://www.Ontologytech.com/~ont/>

9.             </rdf:Description>

10.         </ outbook:owns>

11.         <ontbook:age rdf : datatype="&xsd; integer" > 34 </ ontbook:age>

12.     </rdf: Description>

13. </rdf: RDF>

Tag표현의  
축약을 위해

## 2.2 RDF 그래프 코딩 예: RDF/XML

- 대부분의 RDF문서: 하나의 rdf:RDF와 하나 이상의 rdf:Description들로 구성
  - <rdf:RDF> 안에 필요한 Name Space(Resource를 정의한 문서)를 선언

```
<rdf:RDF
  xmlns : rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns : ontbook="http://ontology.sun.ac.kr/ont-book#">
```

- <rdf:Description> 안에는 하나이상의 서술문이 있고, 중첩가능

```
<rdf:Description rdf:about="&ontbook; Kimdk">
  <ontbook : majorsIn rdf : resource="&ontbook ; Mangement"/>
  ...
  <ontbook : age rdf : datatype="&xsd; integer" >34</ontbook : age>
</rdf: Description>
```



## 2.2 RDF 그래프 코딩 예: RDF/XML

3. <rdf:Description> 요소는 자원을 지칭하는 attribute인 <rdf:about> 포함

- <rdf:ID>와 동등한 의미지만 그 자원은 이미 다른 곳에서 정의되었음을 암시
- RDF 서술문 : 동일한 자원의 정의를 여러 곳에서 할 수 없음
- 한 곳에서 자원을 정의하고 다른 곳에서 추가적인 속성 기술 가능

■ Property Element: <rdf:Description> 의 Child Element

- 해당 자원이 가지고 있는 속성을 나타냄
- 이러한 속성에 대한 값은 요소의 내용으로 기록

```
<rdf:Description rdf:about="&ontbook; Kimdk">  
  <ontbook : majorsIn rdf : resource="&ontbook ; Mangement"/>  
  ...  
  <ontbook : age rdf : datatype="&xsd; integer" >34</ontbook : age>  
</rdf: Description>
```

## 2.2 RDF 그래프 코딩 예: RDF/XML

### 4. <rdf:resource>

- <rdf:about> 나 <rdf:ID>처럼 Resource를 분명하게 지칭하기 위해 사용
- 자원에 대한 다른 정보나 지식을 추가하지 않을 때도 사용할 수 있어 주로 Object에 해당하는 자원을 가리킬 때 많이 사용

```
<rdf : Description rdf : about="#000001">
  <ontbook : name> Kimdk </ontbook : name>
  <ontbook : age rdf : datatype="&xsd;integer"> 34 </ontbook : age>
</rdf : Description>

<rdf : Description rdf : about="#COM345">
  <ontbook : name> OnrologyTech </ontbook : name>
  <ontbook : isOwnedBy> Kimdk </ontbook : isOwnedBy>
</rdf : Description>
```

'#000001'의 '김동건'과 '#COM345'의 '김동건'이 동명이인일 가능성이 있다

## 2.2 RDF 그래프 코딩 예: RDF/XML

```
<rdf : Description rdf : about = "#000001">
    <ontbook : name>Kimdk</ontbook : name>
    <ontbook : age rdf : datatype="&xsd;integer">34</ontbook : age>
</rdf : Description>

<rdf : Description rdf : about = "#COM345">
    <ontbook : name>OnrologyTech</ontbook : name>
    <ontbook : isOwnedBy rdf : resource="#000001" />
</rdf : Description>
```

<rdf:resource>를 사용해 '김동건'이 같은 사람이라는 것을 명시적으로 표현가능

<rdf:resource>도 URIref를 값으로 취하며 Entity를 활용해 나타낼 수 있다

```
<rdf:Description rdf:about="&ontbook; Kimdk">
    <ontbook : majorsIn rdf : resource = "&ontbook ; Mangement" />
    ...
</rdf: Description>
```

## 2.2 RDF 그래프 코딩 예: RDF/XML

5. <rdf : datatype = "&xsd;integer">

- 데이터 타입의 속성 값 범위를 정해주기 위해 사용된 attribute

```
<rdf:Description rdf:about="&ontbook; Kimdk">  
  <ontbook : majorsIn rdf : resource="&ontbook ; Mangement"/>  
  ...  
  <ontbook : age rdf : datatype="&xsd; integer" >34</ontbook : age>  
</rdf: Description>
```

데이터 타입 age가 integer임을 나타냄

# Chapter 6 RDF(S): RDF와 RDF Schema

- 1. XML과 RDF
- 2. RDF
  - 2.1 RDF 데이터 모델
  - 2.2 RDF 그래프와 코딩 예
- 3. RDF Schema
  - 3.1 RDF와 RDF Schema
  - 3.2 기본 요소: 클래스와 속성
  - 3.3 RDF Schema 코딩 예
- 4. RDF(S)의 한계점

# 3.1 RDF와 RDF Schema

## ■ RDF

- 속성의 도메인을 제한하지 못함
- 비슷한 자원을 한 클래스로 묶어 표현하는 기능 없음
- 트리플 구조의 실질적 의미를 컴퓨터가 정확하게 이해 할 수 없다
  - ▶ “김동건은 학생이다” , “학생은 사람이다”
  - ▶ 컴퓨터는 김동건이 사람이라는 사실을 Reasoning(추론)할 수 없다.

## ■ RDF Schema

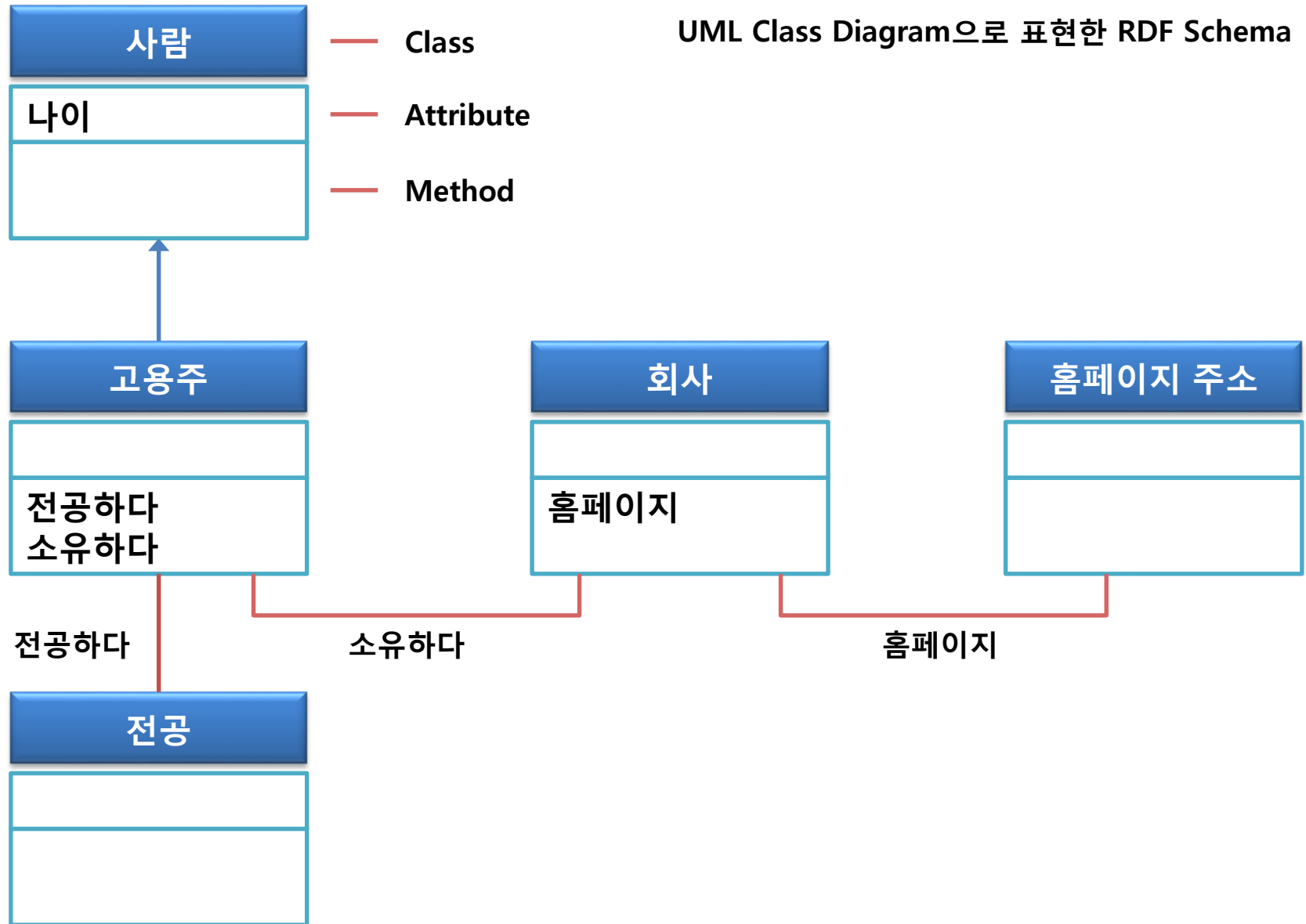
- RDF를 프레임기반으로 확장, 2004년 2월 W3C 권고안으로 발표
- 도메인에 필요한 어휘와 기본 가정들 정의 가능 (Brickly and Guha, 2004)
- 객체지향 프로그래밍(자바) 언어의 데이터 모델과 비슷
  - ▶ 클래스 상속 개념 지원

# 3.1 RDF와 RDF Schema

- RDF Schema와 객체지향 모델링의 차이점 : 속성을 다루는 방법

RDF Schema	Object Oriented Modeling
Property를 Class와 독립적으로 정의 Property가 온톨로지 전 범위에 걸쳐 유효	클래스 정의 안에 속성 포함
Class 정의 수정없이 New Property 적용가능	New Property 추가하려면 Class 정의까지 수정 해야 함
Property에 대해, Subject가 될 수 있는 클래스(domain)와 Object로 올 수 있는 클래스(range)를 명시	

# 3.1 RDF와 RDF Schema: UML Class Diagram



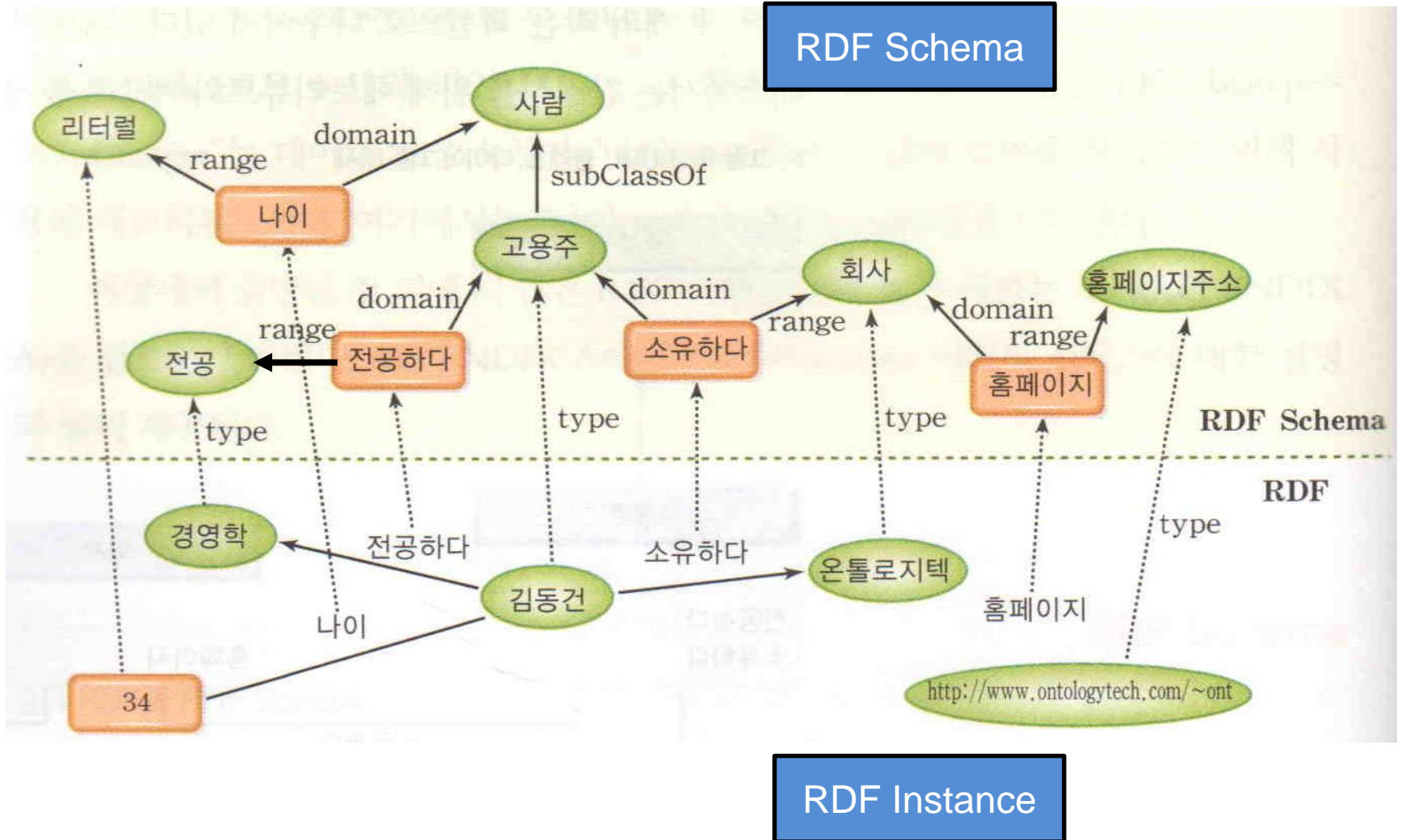


# 3.1 RDF와 RDF Schema: UML Class Diagram

\*\* 아래와 같은 RDF Graph는 RDF Schema에 대한 Instance 정보라고 할 수 있다

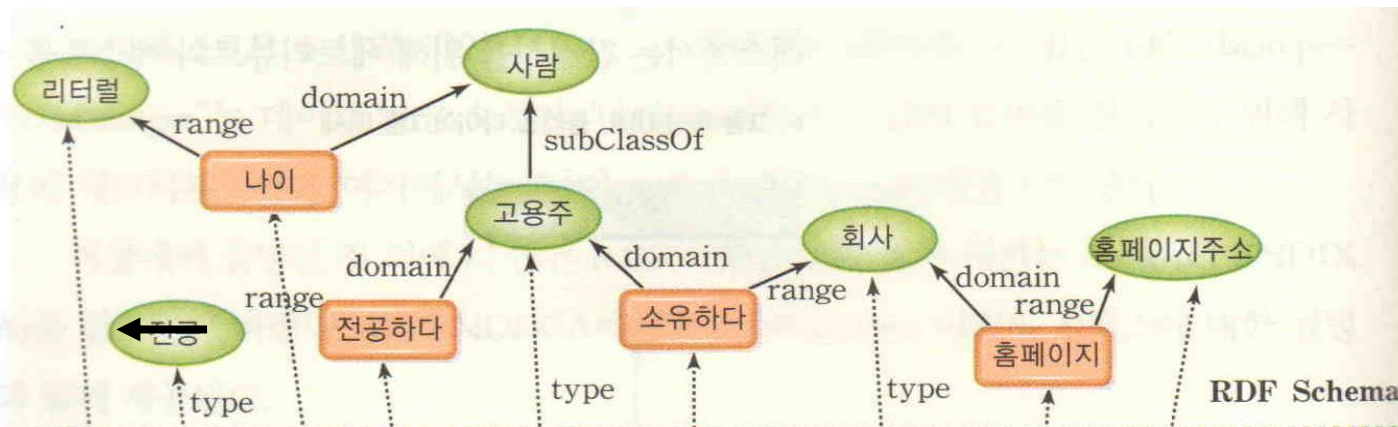


# 3.1 RDF와 RDF Schema



# 3.1 RDF와 RDF Schema

- RDF Schema의 역할
  - 해당 도메인을 기술하기 위해 필요한 어휘를 정의
  - Property의 Domain과 Range를 정의
  - Resource 및 Property 간의 계층구조를 포함하는 다양한 관계를 정의
- 예: 만약 “서태희는 의류디자인학을 소유하고 있다”라는 RDF/XML이 있다면
  - RDF Schema의 “소유하다”에 대한 Constraint로 이 문장의 잘못 감지
    - ▶ Property “소유하다”: Domain은 ‘고용주’, Range는 ‘회사’로 규정
    - ▶ 의류디자인학’은 회사가 아니므로 ‘소유하다’의 Range가 될 수 없음

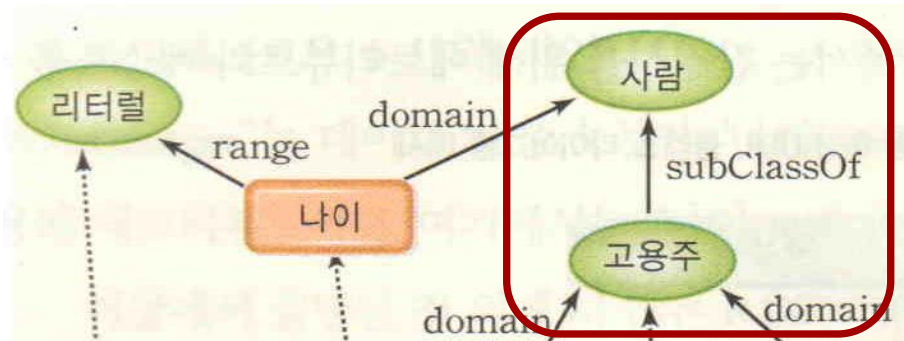


# 3.1 RDF와 RDF Schema

- XML에서 아래 표현에 대해서 모든 '사람'을 검색하면 두 사람만 검색

```
<사람> 서태희 </사람>  
<사람> 오혜수 </사람>  
<고용주 이름="김동건">  
    <소유하다> 온톨로지텍</소유하다>  
</고용주>
```

- RDF Schema가 있으면 세사람이 검색



- '고용주'는 '사람'의 subClassOf의 Semantics에 의해
  - "모든 고용주 Instance는 사람의 Instance"

## 3.2 기본 요소: Class and Property

- 3. RDF Schema
  - 3.1 RDF와 RDF Schema
  - 3.2 기본 요소: 클래스와 속성
    - ▶ 3.2.1 클래스 (Class)
    - ▶ 3.2.2 속성 (Property)
  - 3.3 RDF Schema 코딩 예

## 3.2.1 Class

- 동일한 Property들 → 하나의 부류로 모아지는 개체들의 그룹
  - 예) A,B,C가 서울대 경영학과에서 MIS전공 대학원생이라는 공통의 특징 → '서울대 경영학과 MIS 전공 대학원생' Class 생성
- <rdfs : subClassOf>: 하위 클래스를 상위클래스와 연결
  - 하위 클래스에 속한 Instance는 자동으로 상위 클래스에 속함
  - 하나의 하위 클래스는 여러 개의 상위 클래스를 가질 수 있음
  - <rdfs : Class>와 </rdfs : Class> 사이에서만 쓰임
  - 예) 'LaserPrinter'가 'Printer'의 하위 클래스일때,

```
<rdfs : Class rdf : ID = "LaserPrinter">  
  <rdfs : subClassOf rdf : resource="#Printer"/>  
</rdfs : Class>
```

## 3.2.2 Property

### ■ RDF Schema의 Property

- Class 와 Class 간의 관계 ( $\leftrightarrow$ RDF의 Property는 Resource와 Resource 간의 관계)
- Class 개념을 통해 모든 개체를 효율적으로 묶어 표현 가능
- Class는 다른 Class와 관계를 형성, 풍부한 지식표현 가능

### ■ 예) 교수 Class – ‘김동건’, ‘서태희’, ‘오혜수’

강좌 Class – ‘경영학개론’, ‘온톨로지개론’, ‘시맨틱웹’

‘가르친다’ Property – Domain ‘교수’, Range ‘강좌’로 정의하면

➔ ‘오혜수 교수가 시맨틱웹 강좌를 가르친다’ 등의 새로운 의미  
형성 가능

## 3.2.2 Property: 정의역(Domain), 공역(Range)

- Property의 Subject와 Object에 올 수 있는 Class의 범위를 지정
  - Subject : 어떤 Property를 취할 수 있는 클래스 <rdfs : domain>
  - Object : 그 Property가 속성값을 취할 수 있는 범위 <rdfs : range>

```
<rdf : Property rdf : ID = "제조되다">  
  <rdfs : domain rdf : resource = "#프린터"/>  
  <rdfs : range rdf : resource = "#제조회사"/>  
</rdf : Property>
```

- 특정 Class를 Domain이나 Range로 지정하지 않으면 모든 Class를 범위로 인식

```
<rdf : Property rdf : ID = "제조되다" />
```

- 하나의 Property에 대해,
  - Domain과 Range는 하나씩만 정의 가능
  - 한 번 지정된 Domain/Range를 반복안됨
  - 다른 Domain/Range 추가할 수 없음



## 3.2.2 Property: Hierarchy

- `<rdfs : subPropertyOf>`: Property간 계층관계 표현
- Subproperty는 상위 Property의 Domain/Range를 자신의 Domain/Range로 인식

```
<rdf : Property rdf : ID = "isAdaughterOf">  
  <rdfs : subPropertyOf rdf : resource = "#isAChildOf">  
</rdf : Property>
```

‘~의 딸이다’(isAdaughterOf)라는 속성은  
‘~의 자식이다’(isAChildOf)라는 속성의 SubProperty

# 3.3 RDF Schema 코딩 예

- 3. RDF Schema
  - 3.1 RDF와 RDF Schema
  - 3.2 기본 요소: 클래스와 속성
    - ▶ 3.2.1 클래스 (Class)
    - ▶ 3.2.2 속성 (Property)
  - 3.3 RDF Schema 코딩 예

# 3.3 RDF Schema 코딩 예

## ■ RDF Schema 주요 어휘

어휘	설명
<b>rdfs : Class</b>	클래스를 정의하는 Element Property들을 클래스에 할당하기 위해 <b>rdf : Property</b> , <b>rdfs : range</b> , <b>rdfs : domain</b> 을 함께 사용 클래스 Identifier로서 attribute <b>&lt;rdf:about&gt;</b> 에 <b>URIref</b> 를 써 줌
<b>rdfs : label</b>	클래스에 사람이 이해할 수 있는 라벨을 붙여주는 attribute
<b>rdfs :subclassOf</b>	한 클래스가 다른 기존 클래스의 하위 클래스임을 명시하는 요소
<b>rdf : Property</b>	클래스의 Property를 정의하는 Element <b>rdfs : range</b> , <b>rdfs : domain</b> 과 함께 사용됨
<b>rdfs : domain</b>	미리 정의된 클래스를 값으로 가짐, 한 속성이 클래스에 속하는지를 정의하는 attribute(주어부, 정의역)
<b>rdfs : range</b>	미리 정의된 클래스를 값으로 가짐, 한 속성이 취할 수 있는 값의 범위를 정의하는 attribute(목적부, 공역)
<b>rdfs : Literal</b>	문자열이나 정수 같은 상수 값들로 이루어진 클래스

## 3.3 RDF Schema 코딩 예: Class 부분

```
<?xml version="1.1" encoding="euc-kr" ?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
```

```
<rdf : RDF xmlns : rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns : rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xml : base = "http://Ontology.snu.ac.kr/ont-book" >
```

```
<rdfs : Class rdf : ID = "Person">
  <rdfs : label="사람" />
  <rdfs : subclassOf rdf : resource="&rdfs ; Resource" />
</rdfs : Class>
```

Base URI인 "http://Ontology.snu.ac.kr/ont-book" 과 "#"을 붙인 Absolute URIref "http://Ontology.snu.ac.kr/ont-book#Person"을 의미

```
<rdfs : Class rdf : ID = "Employer">
  <rdfs : label>고용주</rdfs : label>
  <rdfs : subclassOf rdf : resource="#Person" />
</rdfs : Class>
```

```
<rdfs : Class rdf : ID = "Company">
  <rdfs : label="회사" />
  <rdfs : subclassOf rdf : resource="&rdfs ; Resource" />
</rdfs : Class>
```

...

## 3.3 RDF Schema 코딩 예: Property 부분

```
<?xml version="1.1" encoding = "euc-kr" ?>
```

```
...
```

```
<rdfs : Class rdf : ID = "Person">
```

```
...
```

```
</rdfs : Class>
```

```
...
```

```
<rdf : Property rdf : ID = "owns">
```

```
<rdfs : label="소유하다" />
```

```
<rdfs : domain rdf : resource="#Employer" />
```

```
<rdfs : range rdf : resource="#Company" />
```

```
</rdf:Property>
```

```
<rdf : Property rdf : ID = "age">
```

```
<rdfs : label="나이" />
```

```
<rdfs : domain rdf : resource="#Person" />
```

```
<rdfs : range rdf : resource="&rdfs ; Literal" />
```

```
</rdf:Property>
```

```
...
```

```
</rdf : RDF>
```

UML에서 Attribute 였던 "Age" / Method 였던 "소유하다"가  
RDF Schema에서는 모두 Property로 표현

## 4. RDF(S): RDF & RDF Schema

- RDF와 RDFS는 현실 세계의 여러 개념을 표현할 수 있는 방법 제공
- RDF
  - Triple 구조의 간단한 형태로 무한한 지식을 표현
  - Resource – Predicate or Property – Resource 형태로 나타내기만 하면 됨
  - Resource를 연결한것에 불과해 Resource 간의 의미적관계의 표현에는 **한계**
    - ▶ (“김동건은 학생이다”, “학생은 사람이다” → 김동건이 사람이란 추론 불가)
- RDF Schema (**RDF의 한계를 극복!**)
  - 유사한 개체들을 클래스로 정의하고 Property를 기술하고 계층구조도 설정
  - 각 속성이 Subject와 Object에 어떤 값을 취하는지 Domain/Range 개념 사용
  - 메타데이터의 속성과 클래스 간의 관계 표현이 가능
- **그러나 아직도 정보의 의미를 표현하는 데에 부족함이 많음!**

## 4. RDF(S): Limitation [Antonious and Harmelen, 2004] [1/2]

- Property에 대한 속성값의 범위를 <rdfs : range>로 표현하면 온톨로지 전체에 대해 적용 → Subject에 따라 다르게 지정할 수 없음
  - Property가 여러가지 있을수 있으나 RDF(S)에서는 모든 속성을 동일하게 취급
  - Class간의 AND, OR, NOT 혹은 Union, Intersection, Disjointness 개념 부재
  - Class들과 Property들에서 Equality과 Inequality의 표현부재
    - 각기 다른 온톨로지를 병합하거나 재사용 할 때,
      - ▶ 같은 의미를 지닌 다른 이름의 클래스들과 속성들의 Equality 표현 못함
      - ▶ 다른 의미를 지닌 같은 이름의 클래스들과 속성들의 Inequality을 표현 못함
  - Property에 대한 cardinality 표현 부재
- ➔ 이와 같은 표현상의 한계 때문에 **모델링 요소들을 확장하고 언어의 표현력을 강화한 OWL과 같은 온톨로지 언어가 등장**