

Tech 워크숍

# 링크드 오픈 데이터 (Linked Open Data)

- 소개부터 구축 및 활용까지 -

**Semantics**      **SPARQL**      **RDFS**  
**Interlinking**      **RDF**      **Ontology**      **SWRL**  
**Content Negotiation**  
**XML**      **OWL**      **URI**      **Triple**  
**Rule**      **Inference**

따라 할 수 있겠어?

A photograph of a man in a business suit meditating with his eyes closed and hands raised in a mudra, sitting cross-legged on a conference table. He is surrounded by several other business people who appear to be working or talking. A speech bubble above him contains the Korean text "쉽게 쉽게 갑시다." (Easy, easy, let's do it.).

쉽게 쉽게 갑시다.

**HTML**의 목적은  
사람이 읽고 해석할 수 있는  
연결된 문서를 만드는 것



## 인물 정보

유라 (김마영) 가수



출생 1992년 11월 6일

신체 168cm, 49kg

소속그룹 걸스데이

소속사 드림티엔터테인먼트

학력 동덕여자대학교 방송연예과

사이트 공식사이트, 트위터, 유튜브

내 프로필 수정

최근 정

## 인물

유라 (김마영) 가수



출생 1992년 11월 6일 (만 21세) | 원승이띠, 전갈자리

신체 168cm, 49kg | 0형

그룹 걸스데이

데뷔 2010년 걸스데이 싱글 앨범 'Girl's Day Party #2'

소속 드림티엔터테인먼트

학력 동덕여자대학교 방송연예학과 외 1건

사이트 트위터

관계자 정보 확인 | 2013.12.01 (?)

## 정보의 재사용!!!

?

다음 뮤직 ①

다른 뮤직 사이트 보기

[Girl's Day Party #2](#)

마이스트 걸스데이

앨범정보 싱글, 가요 &gt; 마이돌 | 2010.10.29

앨범소개 파격변신 5인조 걸그룹 걸스데이, 더 새롭게 더 높게 더 멀리 날다 음악과... [더보기](#)[앨범듣기](#)

번호	곡명	마이스트	듣기	가사	뮤비	구매
1	잘해줘봐야 [타이틀]	걸스데이				<a href="#">MP3</a> <a href="#">BGM</a>
2	잘해줘봐야 (Inst.)	걸스데이				<a href="#">MP3</a> <a href="#">BGM</a>

[http://search.naver.com/search.naver?sm=tab\\_hty.top&where=nexearch&ie=utf8&query=%EC%9C%A0%EB%9D%BC&x=0&y=0](http://search.naver.com/search.naver?sm=tab_hty.top&where=nexearch&ie=utf8&query=%EC%9C%A0%EB%9D%BC&x=0&y=0)[http://search.daum.net/search?w=tot&DA=YZRR&t\\_=nil\\_searchbox=btn&sug=&sq=&o=&q=%EC%9C%A0%EB%9D%BC](http://search.daum.net/search?w=tot&DA=YZRR&t_=nil_searchbox=btn&sug=&sq=&o=&q=%EC%9C%A0%EB%9D%BC)[http://search.daum.net/search?nil\\_suggest=btn&w=tot&DA=SBCO&q=Girl%27s+Day+Party+%2323](http://search.daum.net/search?nil_suggest=btn&w=tot&DA=SBCO&q=Girl%27s+Day+Party+%2323)

# 만약 ...

유라에 대한 정보를 줘.

**NAVER**

Dum



```
{  
    "name": "유라",  
    "realName": "김아영",  
    "job": "가수",  
    "birth": "1992.11.6",  
    "bodySize": {  
        "height": 168,  
        "weight": 49  
    },  
    "group": {  
        "groupName": "걸스데이",  
        "groupUrl": "http://www.naver.com/GirlsDay"  
    },  
    "pic": "http://www.naver.com/GirlsDay/Yura.jpg"  
}
```

# 이렇다면?

재사용성 ↑

기계에 의한 가독성 ↑

새로운 서비스 생산성 ↑



문서  
중심의  
웹

데이터  
중심의  
웹



# Linked Open Data

The Semantic Web isn't just about putting data on the web. It is about **making links**, so that a person or machine **can explore the web of data**. With linked data, when you have some of it, you **can find other, related, data**.



# Linked Open Data



웹을 통해  
사람과 기계가  
읽고 처리할 수 있는 형태로  
데이터에 대한 정보를  
기술(description)

# Linked Open Data



웹 페이지가 서로 연결된 것처럼  
데이터들끼리 다양한 관계에 의해  
연결되어 있는 형태

# Linked Open Data

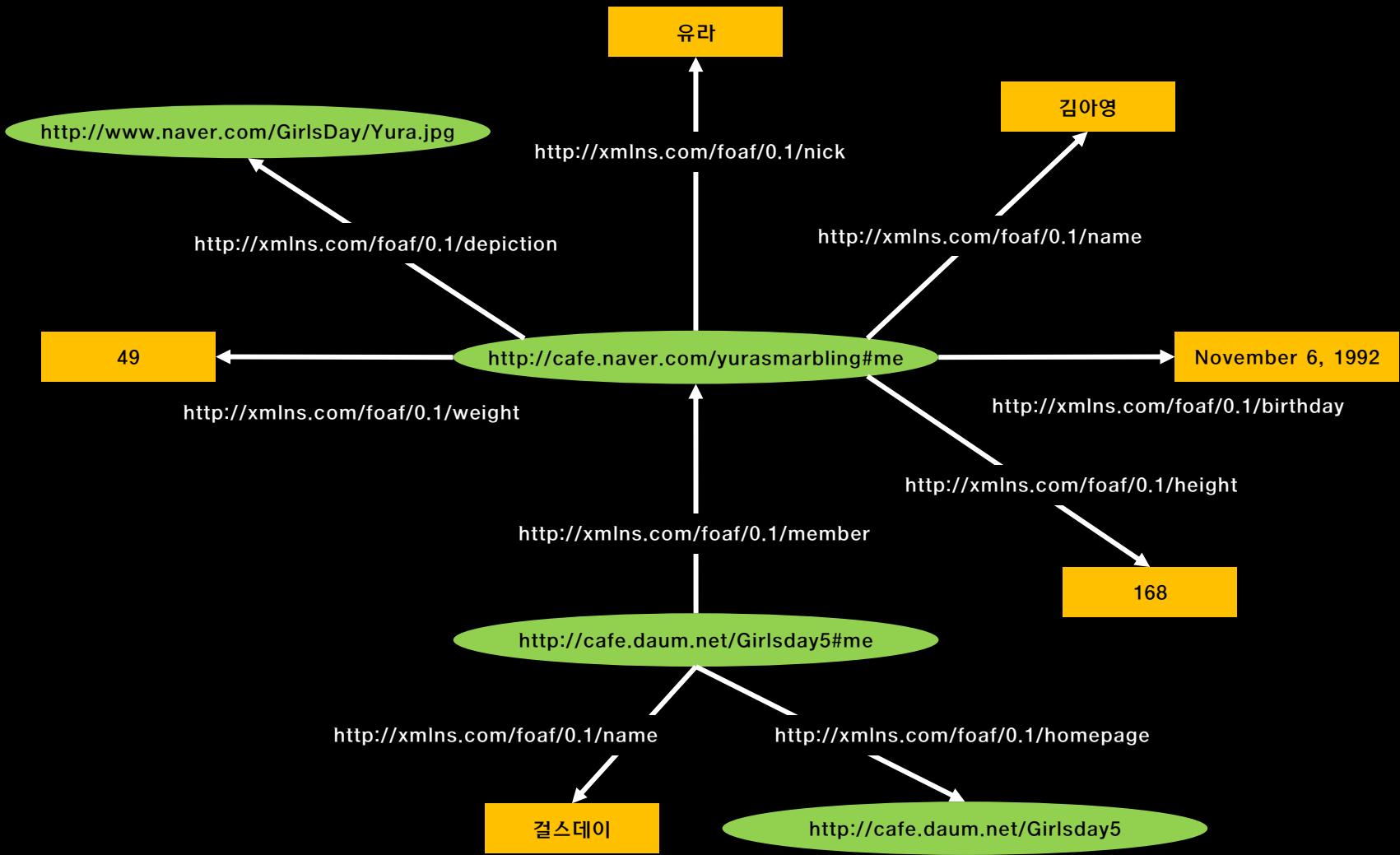


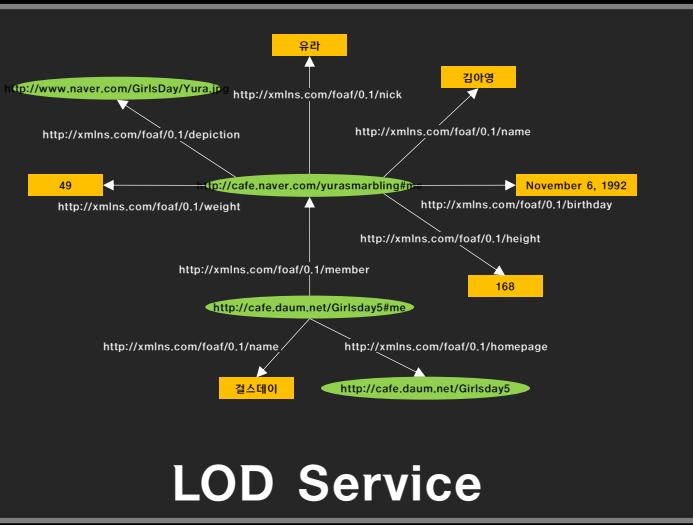
URI를 이용해서  
누구나 데이터에 접근할 수 있으며  
데이터에 대한 정보를 제공

# Linked Data의 네 가지 원칙

Linked Data를 위한 필수 체크 요소

1. 웹 상의 자원을 식별하기 위해 URI를 사용하라.
2. 웹 상의 자원들이 사람과 사용자 에이전트에 의해 참조(refer)되고 탐색(look up)될 수 있는 HTTP URI를 사용하라.
3. 자원의 URI가 역참조(dereference)될 때 자원에 대한 유용한 정보를 RDF/XML과 같은 표준 형식을 사용하여 제공하라.
4. 웹에서 다른 연관된 정보의 탐색을 개선하기 위해 다른 데이터와의 URI 연결(link)를 포함하라.





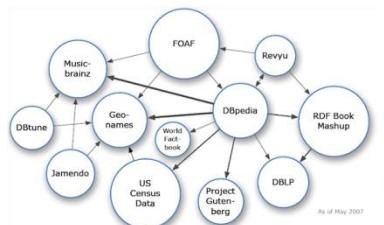
**http://cafe.naver.com/yurasmarbling#me**



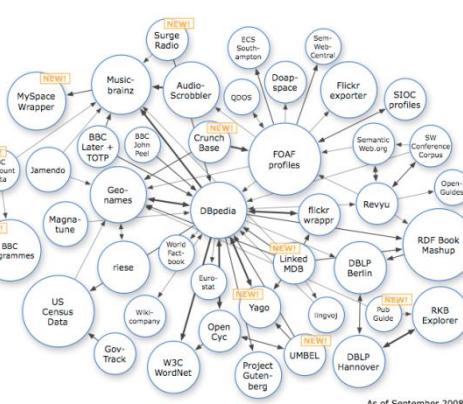
<foaf:Person  
     rdf:about="http://cafe.naver.com/yurasmarbling#me">  
     <foaf:nick>유라</foaf:nick>  
     <foaf:name>김아영</foaf:name>  
     <foaf:birthday>November 6, 1992</foaf:birthday>  
     <foaf:height>168</foaf:height>  
     <foaf:weight>49</foaf:weight>  
     <foaf:depiction  
         rdf:resource="http://www.naver.com/GirlsDay/Yura.jpg" />  
   </foaf:Person>

**RDF/XML**

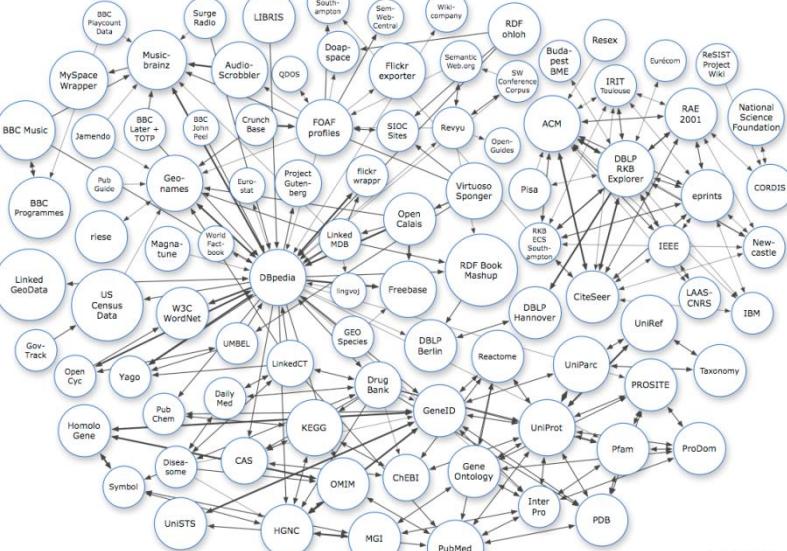
# The Linking Open Data cloud diagram



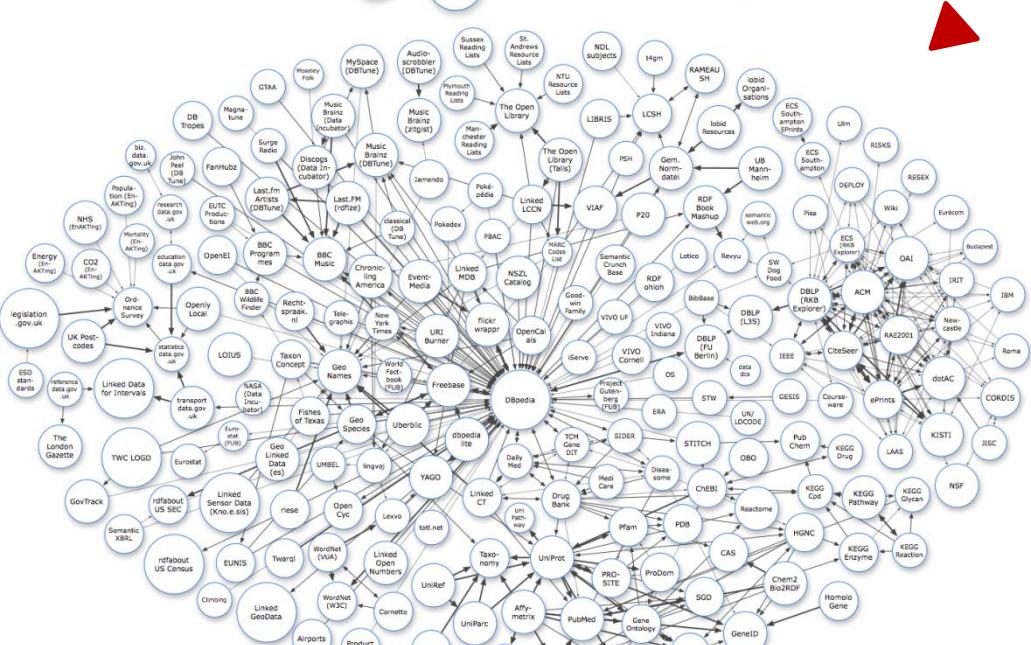
As of May 2007



As of September 2008

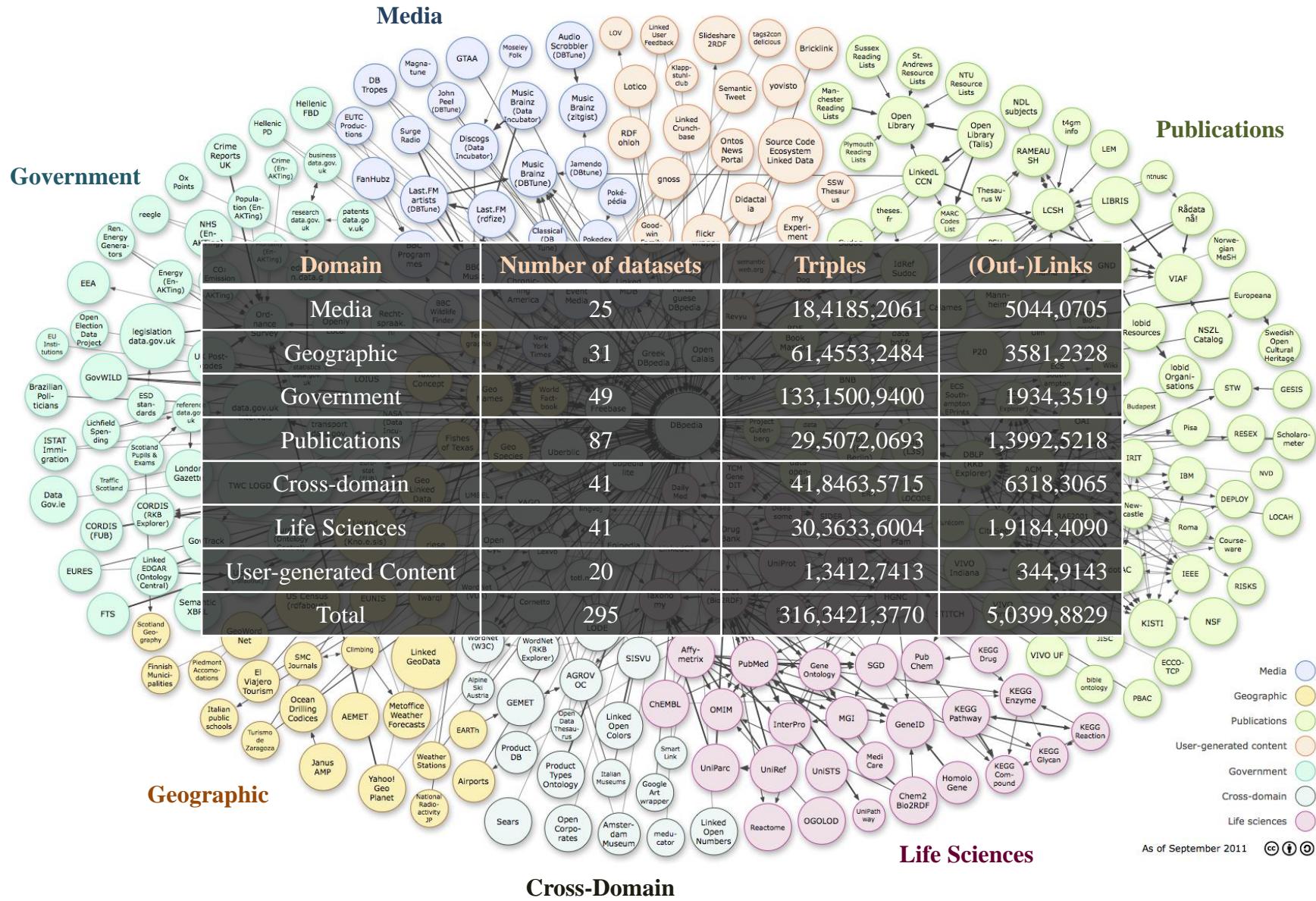


As of July 2009



As of September 2010

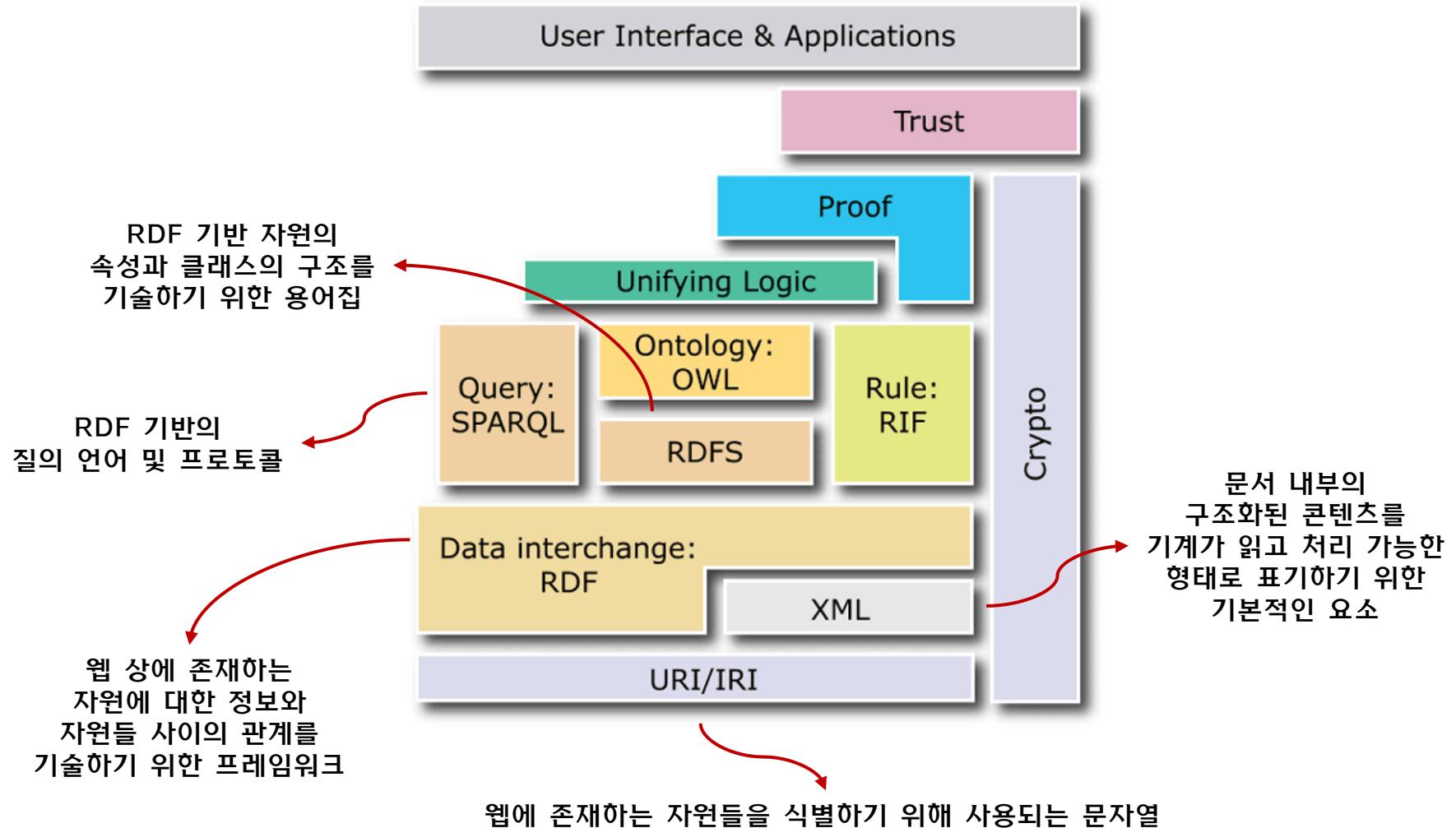
## User Generated Content



본격적으로  
시작해  
볼까요?

START

# Linked Data를 위한 요소 기술들



# URI (Uniform Resource Identifier)

Linked Data에서 **URI**는  
웹 상에서 사물(thing)을 식별하기 위한  
식별자로 활용됩니다.

현실에서는 ...

주민등록번호

861002-1052747

웹에서는 ...

URI

<http://semantics.kr/MyungjinLee>



식별하기 위해서

# URI도 신중하게 만들어야 해요.

Cool URIs와 미국 및 영국 정부의 사례를 참고해 보세요.

- **단순성**
  - 짧고 기억하기 쉬운 형태로
- **안정성**
  - 가능한 오랫동안 유지될 수 있게
- **관리의 용이성**
  - 관리할 수 있는 형태로 만들도록

UK 공공 사례

URI Type	URI structure	Examples
Identifier	<code>http://{domain}/id/{concept}/{reference}</code>	<code>http://education.data.gov.uk/id/school/78</code>

'`http://` BASE '/` id` `/` ORG` `/` CATEGORY ( `/` TOKEN )+`

미국정부 공공 사례

# XML (Extensible Markup Language)

웹에서 사람과 기계 가독성이 있는 형태로  
데이터의 구조를 표현하기 위한 텍스트 형태의 포맷

“이명진은 유라의 남편이다”를 XML로 표현하면?

```
<conjugalrelation>
    <husband>Myungjin Lee</husband>
    <wife>Yura</wife>
</conjugalrelation>
```

```
<conjugalrelation husband="Myungjin Lee">
    <wife>Yura</wife>
</conjugalrelation>
```

```
<conjugalrelation husband="Myungjin Lee" wife="Yura" />
```

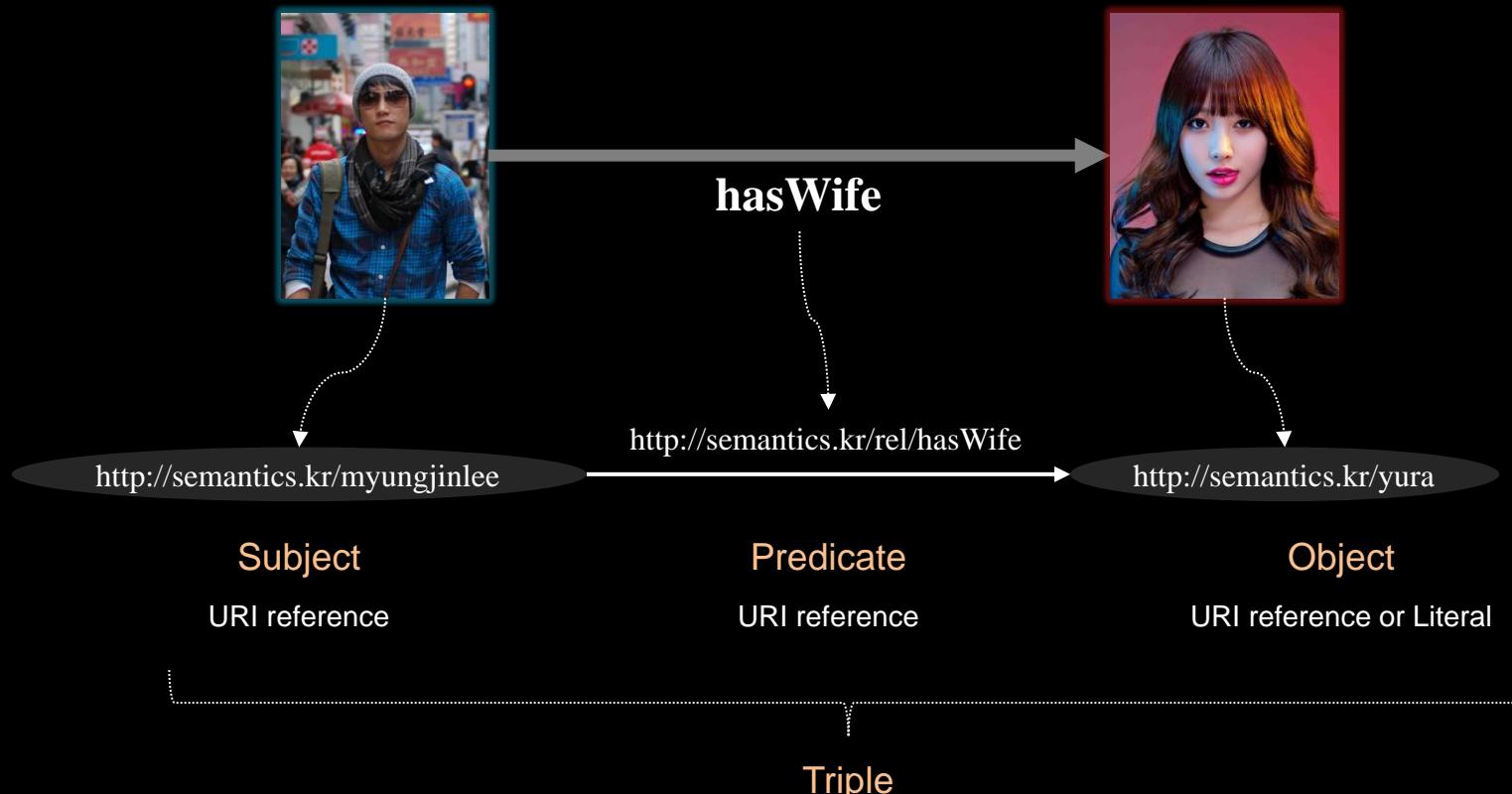
말하고자 하는 내용은 동일하지만,  
구조에 초점이 맞추어져 있기 때문에 다르게 해석된다.

추상 레벨에서 데이터에 대해 기술하기 위한 방법이 필요하다.

# RDF (Resource Description Framework)

웹에 존재하는 자원에 대한 정보를  
개념적으로 기술하거나 모델링하기 위한 방법을 제공

“이명진은 유라의 남편이다”를 RDF로 기술하면?



# 앞서의 기억을 떠올리며…



# 그렇다면 RDF를 어떻게 기계가 읽고 처리하지?

RDF는 데이터에 대한 정보와 관계를 기술하기 위한 그래프 기반의 모델일 뿐 기계가 읽고 처리할 수 있는 노테이션(notation) 방법은 다양한다.

- **RDF/XML**
  - W3C Recommendation, 10 February 2004
  - RDF 그래프 모형을 쓰고 교환하기 위해 사용되는 XML 기반의 문법
- **N-Triples**
  - RDF Test Cases, W3C Recommendation, 10 February 2004
  - RDF 데이터를 저장하고 전송하기 위한 평문(plain text) 형태의 포맷
- **Notation 3 (N3)**
  - 사용자 가독성이 보다 확장된 형태로 RDF 모델을 직렬화(serialization)하기 위한 단축 표현
  - RDF/XML보다 더 간결하고 가독성이 있는 형태
- **Turtle (Terse RDF Triple Language)**
  - W3C Recommendation 25 February 2014
  - RDF에서 데이터를 표현하기 위한 포맷

## N-Triple

```
<http://en.wikipedia.org/wiki/Tony_Benn> <http://purl.org/dc/elements/1.1/title> "Tony Benn" .  
<http://en.wikipedia.org/wiki/Tony_Benn> <http://purl.org/dc/elements/1.1/publisher> "Wikipedia" .
```

## N3

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.  
  
<http://en.wikipedia.org/wiki/Tony_Benn> dc:title "Tony Benn";  
dc:publisher "Wikipedia".
```

## RDF/XML

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
         xmlns:dc="http://purl.org/dc/elements/1.1/">  
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">  
    <dc:title>Tony Benn</dc:title>  
    <dc:publisher>Wikipedia</dc:publisher>  
  </rdf:Description>  
</rdf:RDF>
```

## Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix ex: <http://example.org/stuff/1.0/> .  
  
<http://www.w3.org/TR/rdf-syntax-grammar>  
  dc:title "RDF/XML Syntax Specification (Revised)" ;  
  ex:editor [ ex:fullname "Dave Beckett";  
             ex:homePage <http://purl.org/net/dajobe/>  
           ] .
```

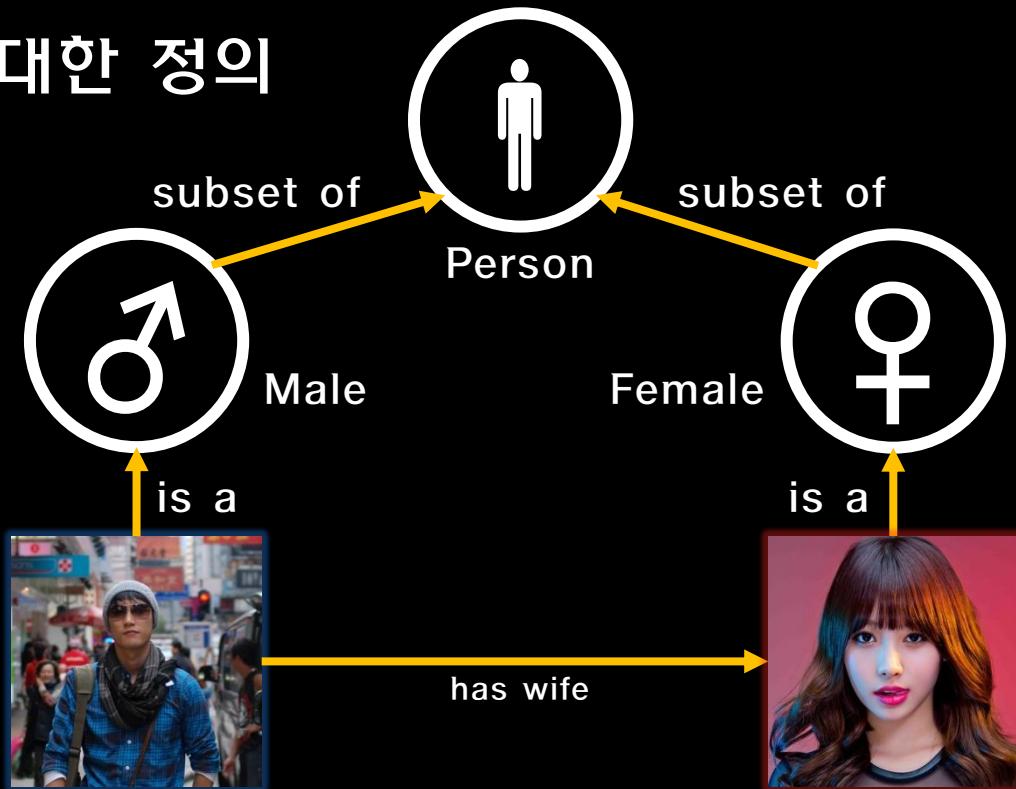
# RDFS(RDF Schema)

아무렇게나 RDF 형태로 데이터를 만들면 되나?

물론 그래도 되지만 먼저 RDF 데이터를 담기 위한 그릇을 만드는 것이 필요해요.

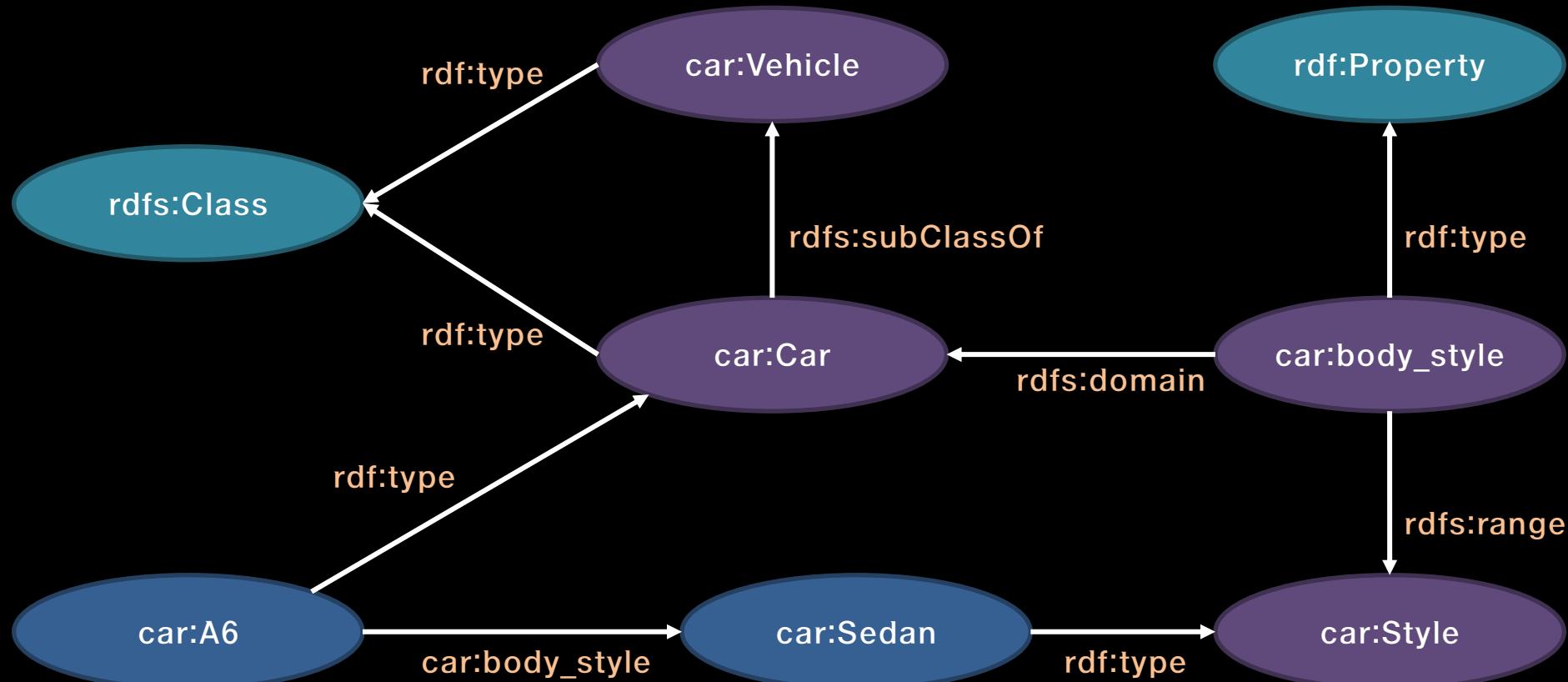
RDF 형태의 데이터를 담기 위한 구조를 만드는 방법을 제공

- 클래스 정의
- 속성과 관련 제약조건에 대한 정의
- 계층 구조에 대한 정의



# RDF Schema Example

RDF 데이터 구조에 대한 정의 부분



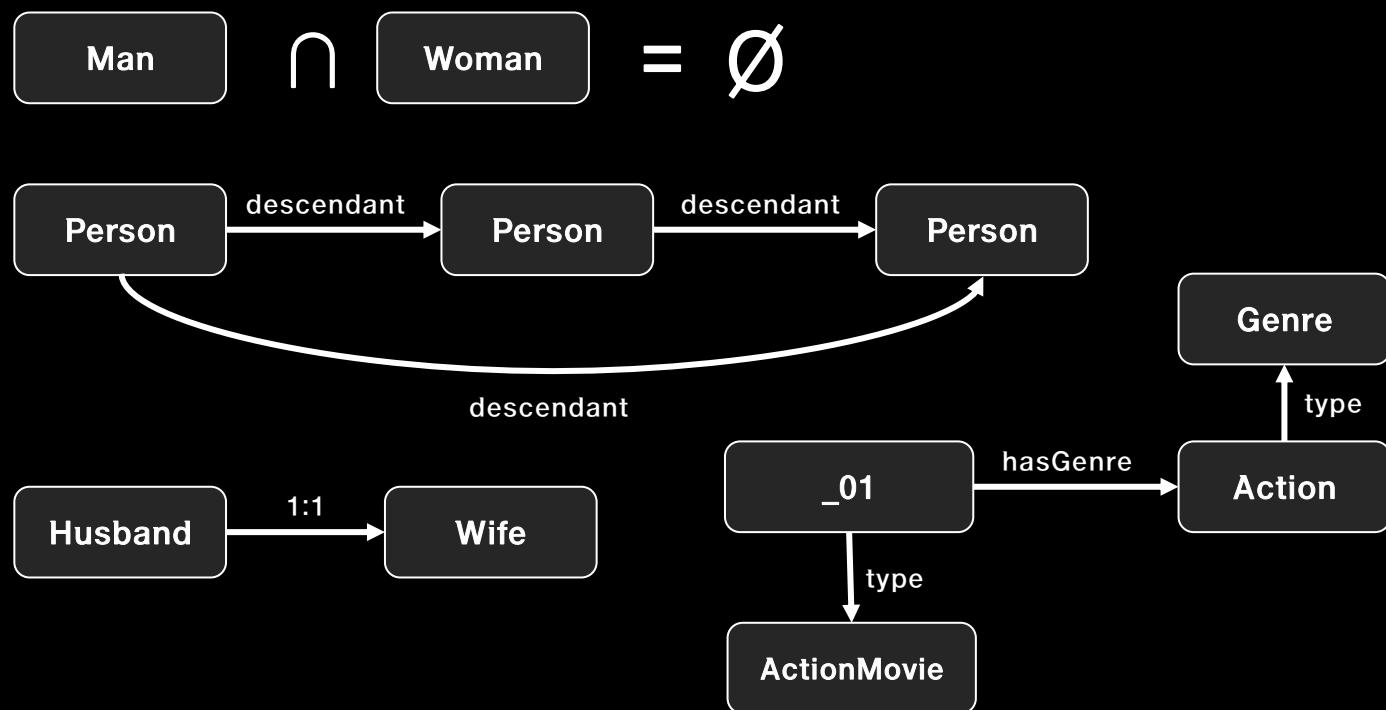
구조에 따른 RDF 데이터 부분

# OWL(Web Ontology Language)

더 복잡한 지식에 대한 구조 정의가 필요할 경우는 어떻하지?

단순한 데이터의 구조를 벗어나  
온톨로지를 만들기 위한 지식 표현 언어

더 많은 표현력이 필요할 경우,



# SPARQL (SPARQL Protocol and RDF Query Language)

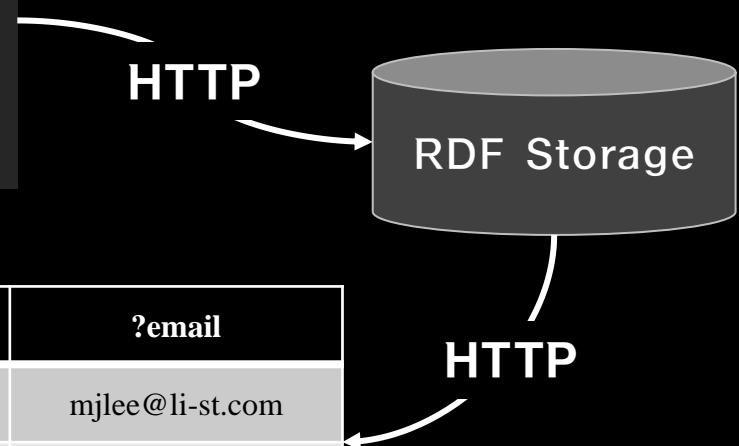
RDF는 그 구조 자체가 RDB와 다르기 때문에  
다른 형태의 질의 언어가 필요하지 않을까?

RDF 형식으로 저장된 데이터를 검색하고 조작하기 위한 언어  
HTTP를 통한 SPARQL의 사용

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
    ?person a foaf:Person.
    ?person foaf:name ?name.
    ?person foaf:mbox ?email.
}
```

SPARQL

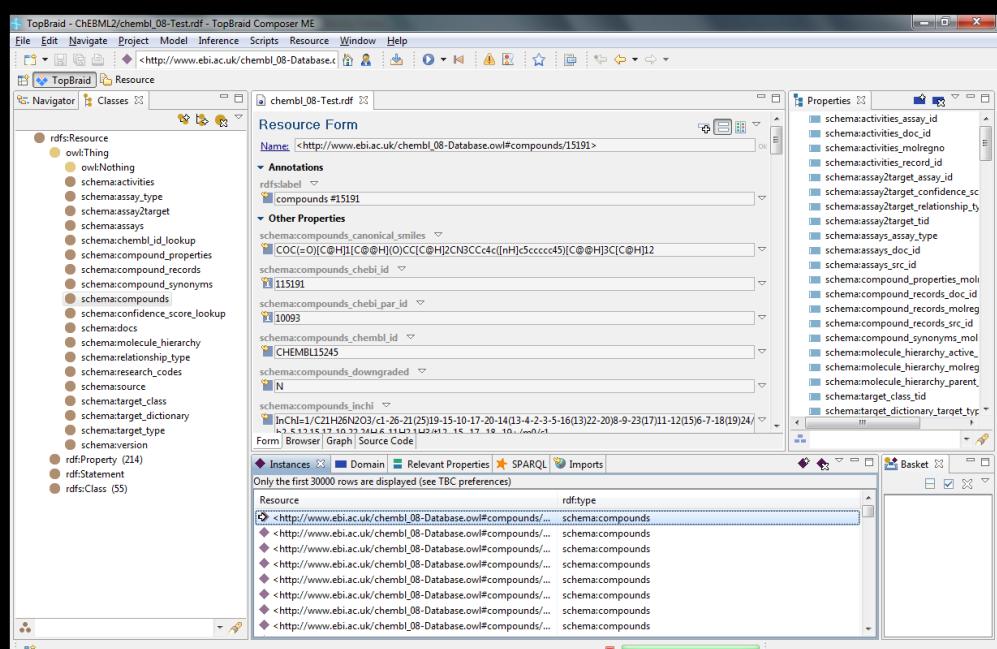
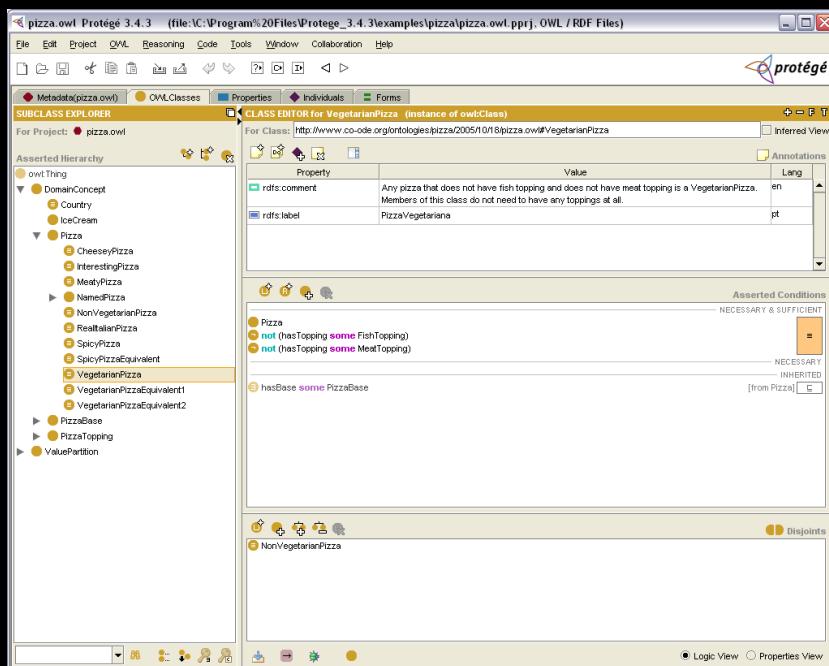
?name	?email
Myungjin Lee	mjlee@li-st.com
Gildong Hong	gildong@daum.net
Grace Byun	grace@naver.com



# LOD를 위해 필요한 도구들

# LOD 구축자에게 필요한 도구

- 온톨로지 편집기
  - 온톨로지 모델링 및 인스턴스 입력을 위한 도구
  - 관련 도구
    - Stanford Center for Biomedical Informatics Research의 Protégé
    - TopQuadrant의 TopBraid Composer

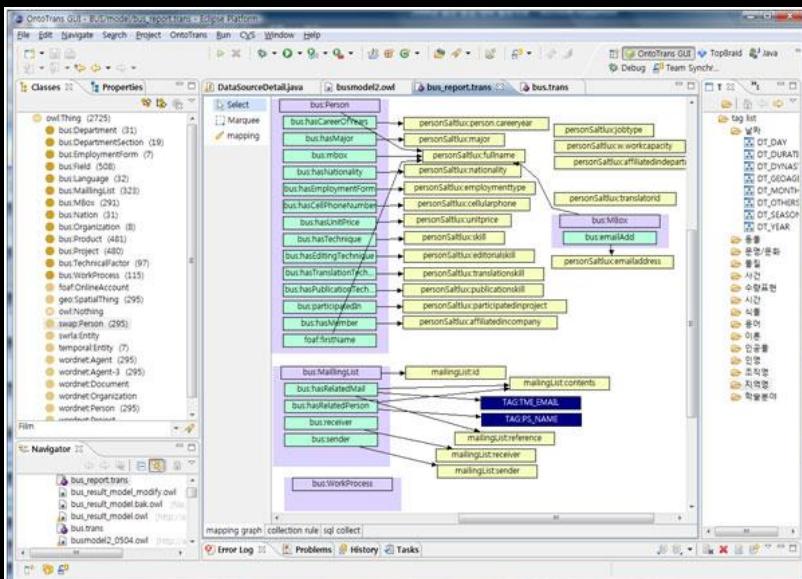


Protégé

TopBraid Composer

# LOD 구축자에게 필요한 도구

- **변환기**
  - 데이터베이스의 관계형 데이터를 RDF 형태로 변환하기 위한 도구
  - R2RML(RDB to RDF Mapping Language)를 기반으로 변환 수행
  - 관련 도구
    - LiST의 OntoTrans
    - OpenLink Software의 Virtuoso RDF Views



OntoTrans

# LOD 운영자에게 필요한 도구

- **트리플 저장소**
  - RDF 형태의 데이터를 저장하고 운용 및 추론과 질의 처리를 수행
  - 관련 도구
    - LiST의 OntoBase
      - 적용사례: KDATA, 주소데이터 LOD, 수목원 LOD, 국립중앙도서관 LOD 등
    - OpenLink Software의 Virtuoso
    - Franz의 AllegroGraph
    - OntoText의 OWLIM
- **LOD 발행 도구**
  - 구축된 RDF 데이터를 웹을 통해 발행하고 접근할 수 있도록 지원
  - 관련 도구
    - Pubby
    - OpenLink Software의 Virtuoso

# LOD 활용자에게 필요한 도구

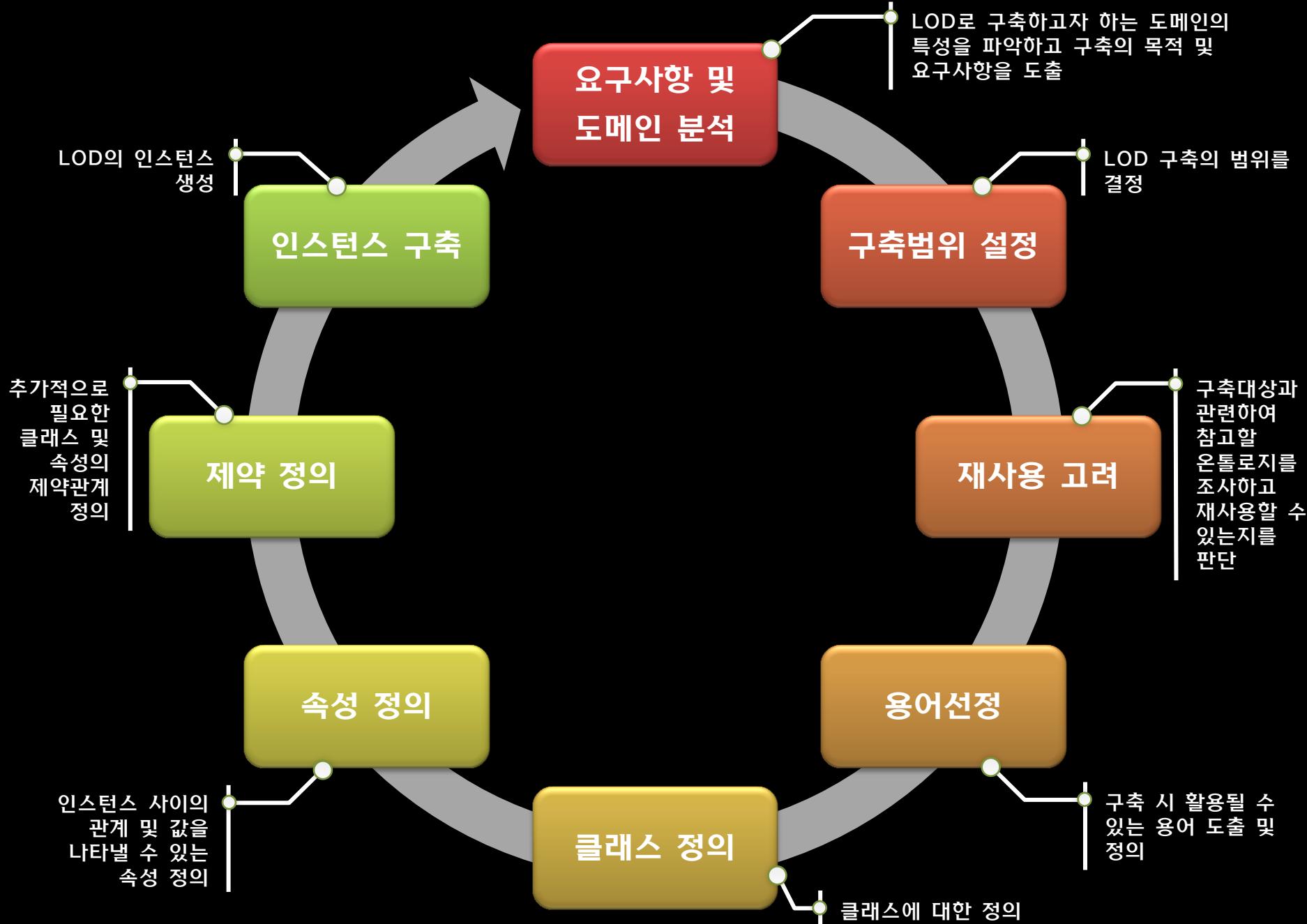
- **프로그래밍 라이브러리**
  - RDF 형태의 데이터를 처리하기 위한 라이브러리
  - 관련 도구
    - Apache의 Jena
    - University of Manchester의 OWL API

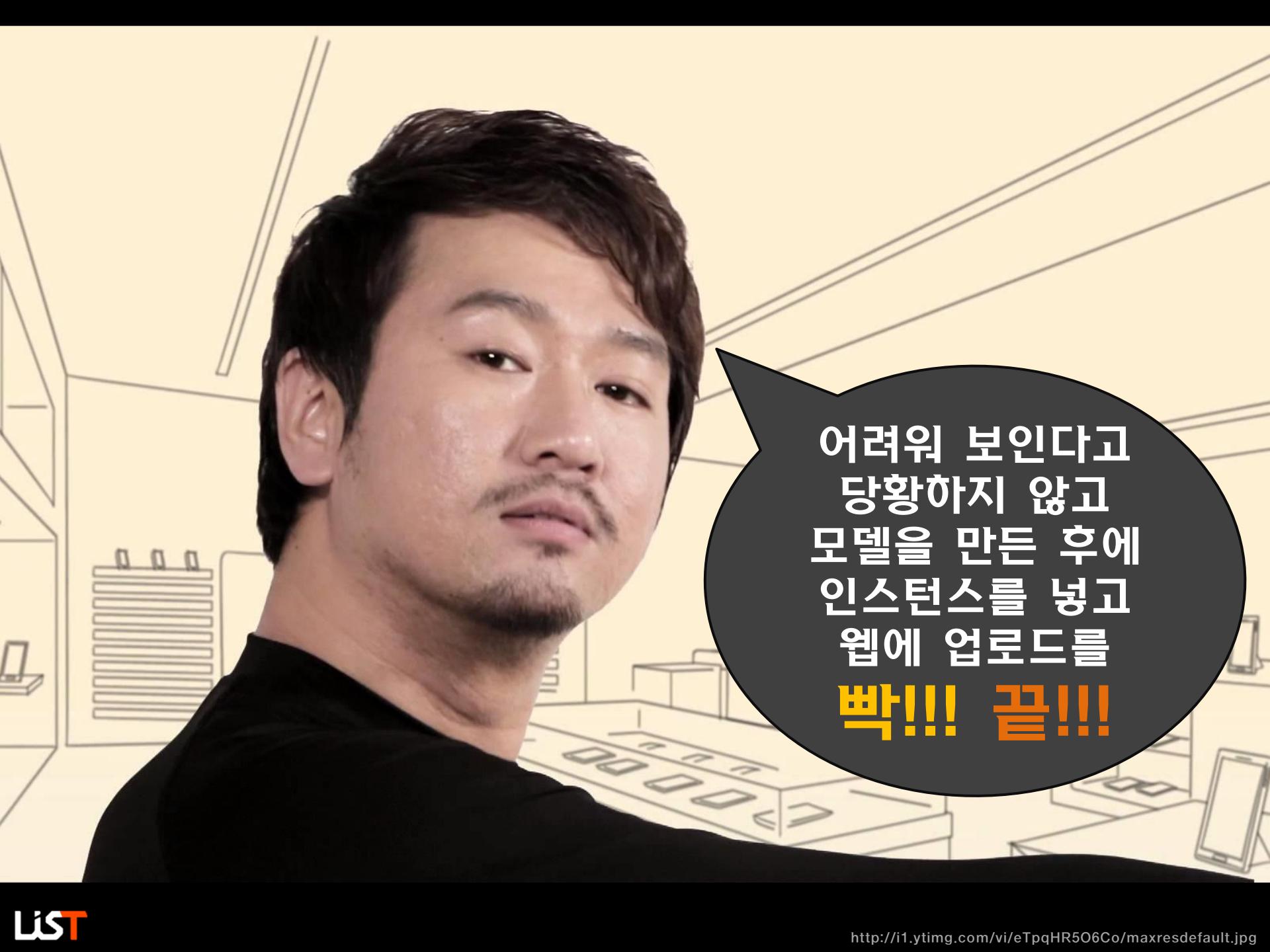
# 그 외 활용할 수 있는 유용한 도구들

- [http://semanticweb.org/wiki/Category:Semantic\\_Web\\_tool](http://semanticweb.org/wiki/Category:Semantic_Web_tool)
- <http://semanticweb.org/wiki/Tools>
- <http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/SemWebClients>

# 온톨로지를 만들어 봅시다.







어려워 보인다고  
당황하지 않고  
모델을 만든 후에  
인스턴스를 넣고  
웹에 업로드를  
**빡!!! 끝!!!**

# 데이터베이스 설계

# 온톨로지 설계

7

# 객체지향 설계

# 그러나 오늘만은,

# 데이터베이스 설계

# 온톨로지 설계

A yellow circular marker is positioned at the bottom right corner of the slide.

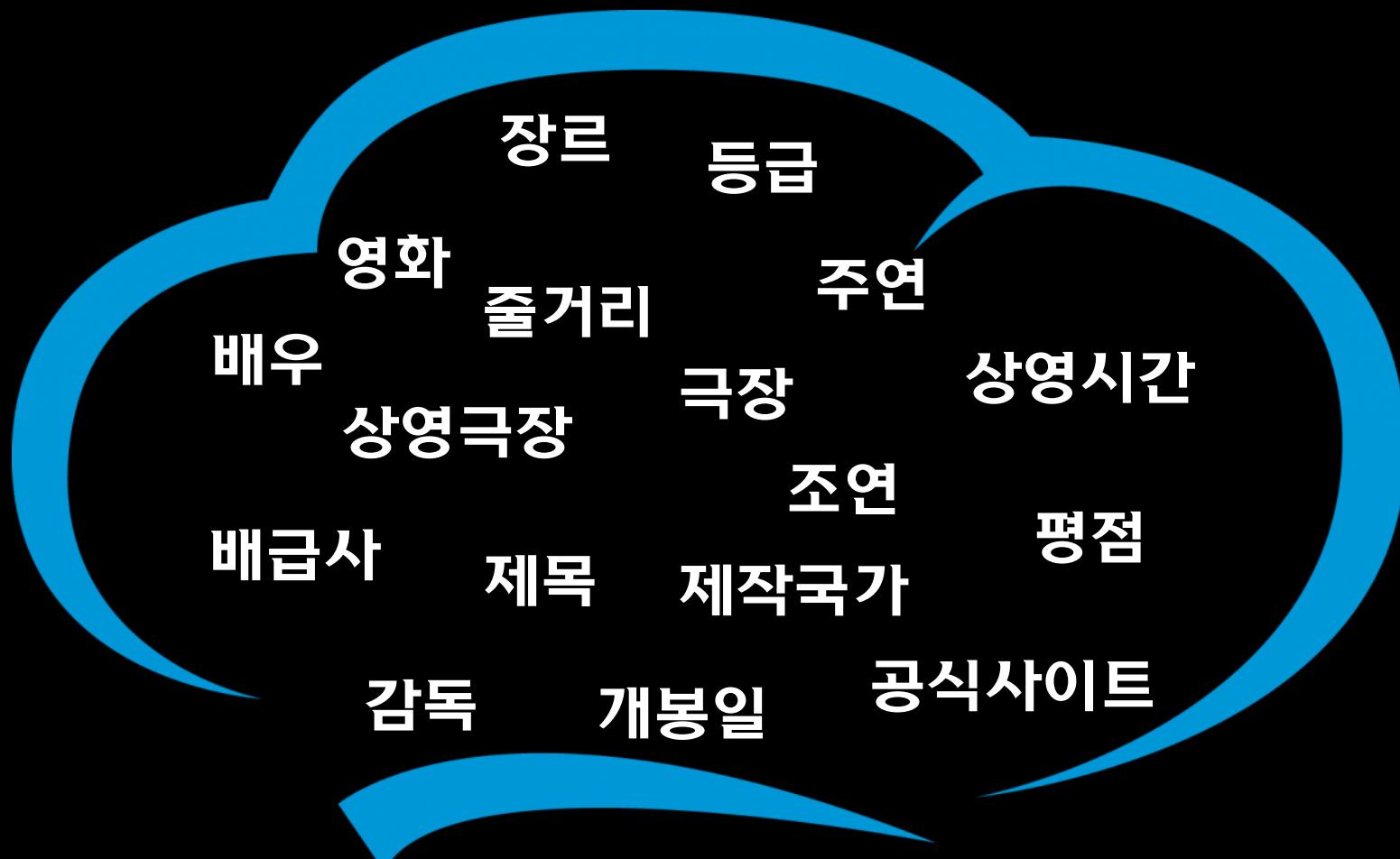
# 객체지향 설계



**우리는 오늘  
영화 LOD를  
만들어 볼꺼에요.**

# 1. 개념 도출 및 용어 정의

- 영화와 관련된 키워드를 모두 뽑아봅시다.



## 2. 클래스 정의

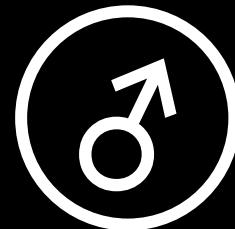
- **클래스**란?
  - 아주 간단히 무엇들의 **집합**

≒ 데이터베이스의 **테이블**

≒ 객체지향의 **클래스**



Myungjin Lee



Male



Class

Person Table

rrn	name	affiliation
841002-1045617	Myungjin Lee	10
410203-3983612	Gildong Hong	20
841105-2056143	Grace Byun	10

School Table

sid	sname
10	Yonsei Univ.
20	Seoul Univ.

Person 클래스

```
public class Person {  
    String rrn;  
    String name;  
    School affiliation;  
}
```

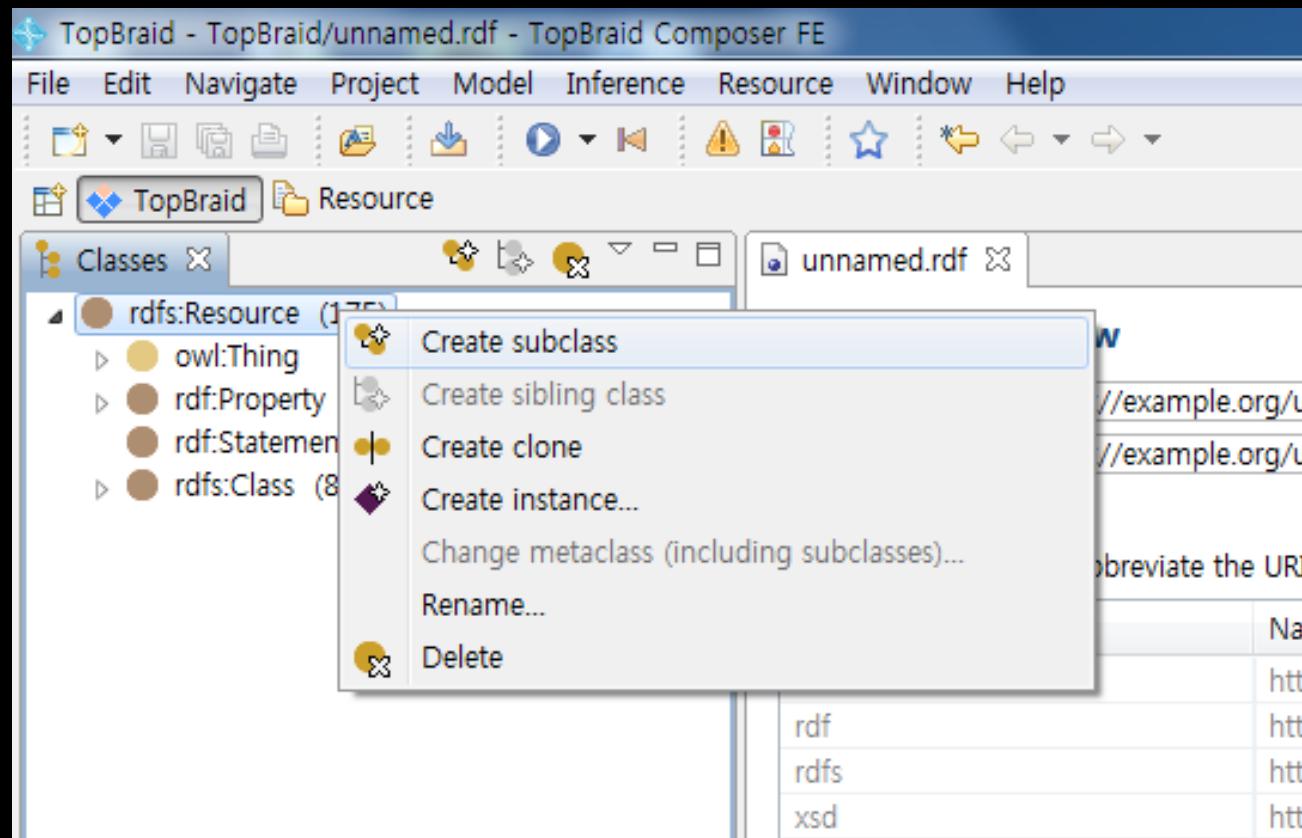
Person Class

School 클래스

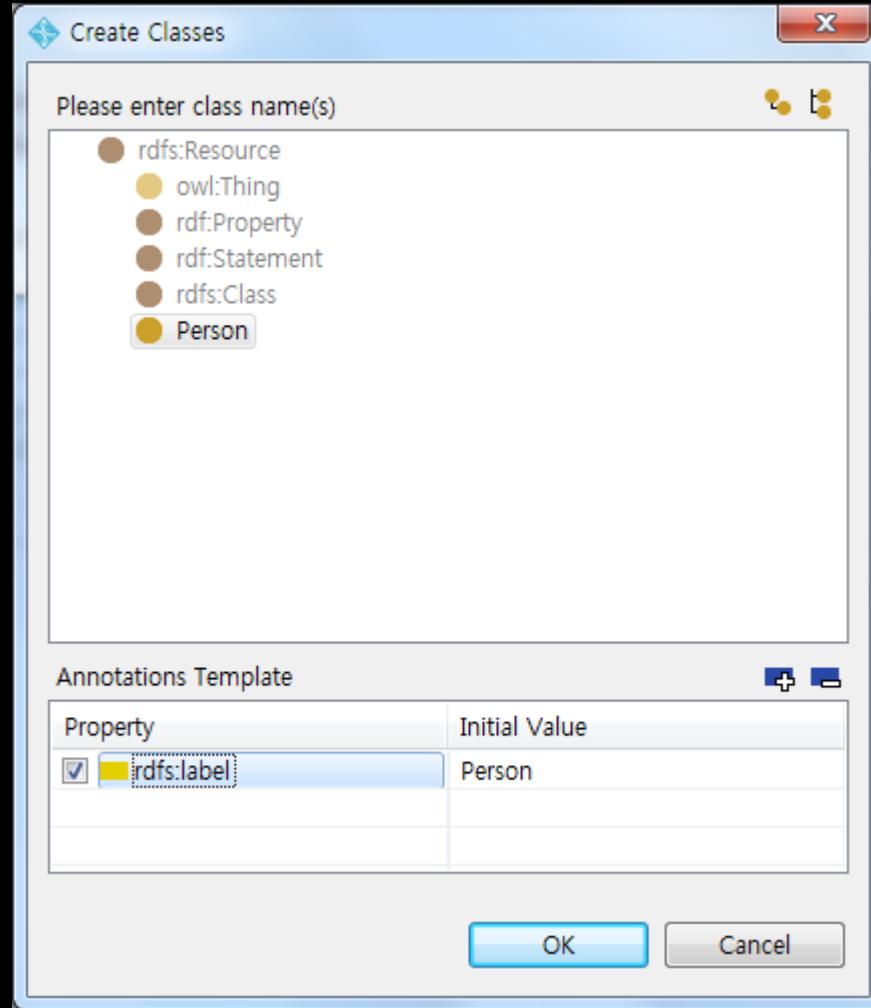
```
public class School {  
    String sname;  
}
```

School Class

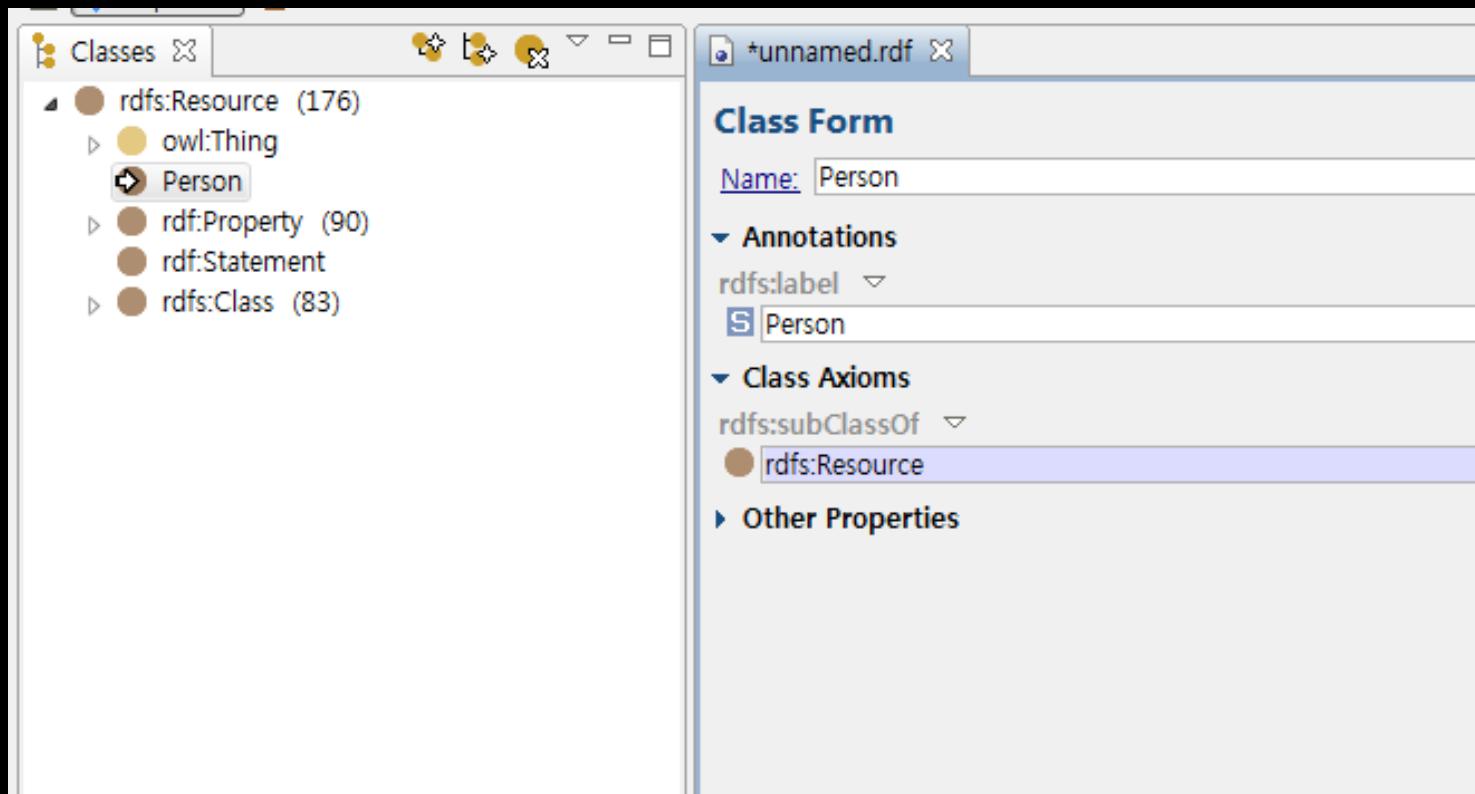
# 따라 해 볼시다.



# 따라 해 봅시다.



# 따라 해 봅시다.



# 실제로는 이렇게 만들어져요.

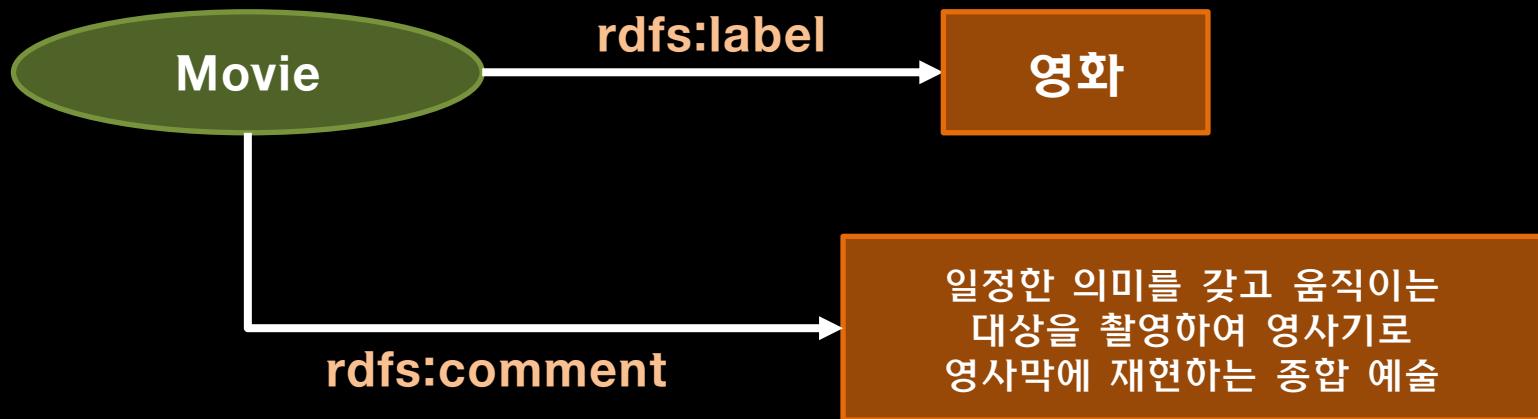


**rdf:type**은 **is-a** 관계를 나타내요.

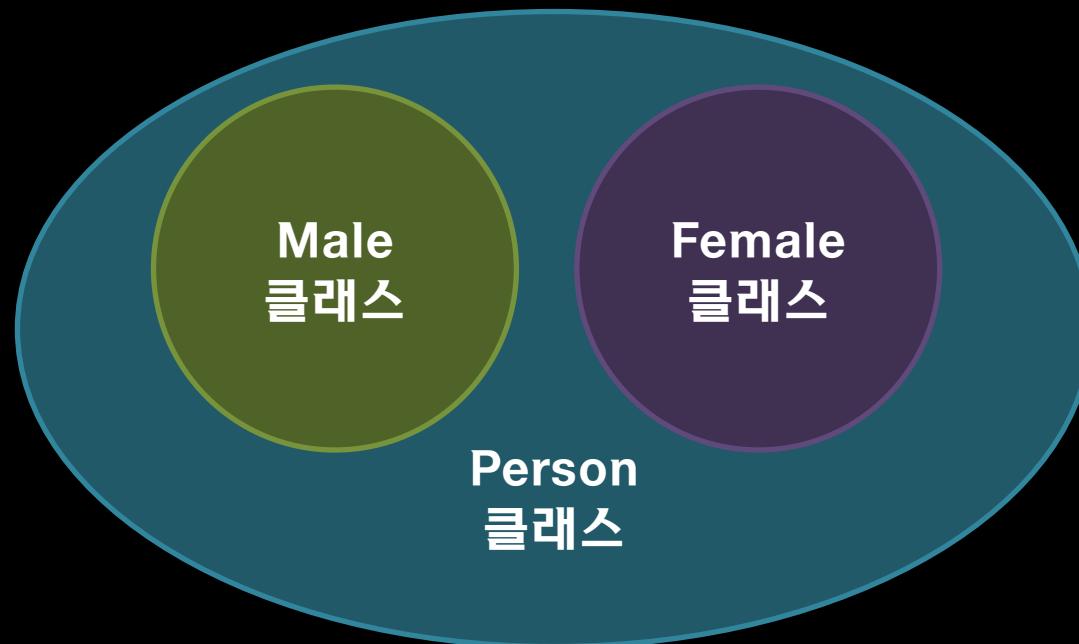
즉, 해석하면 “**Movie는 Class이다**” 가 되네요.

# 좋은 습관 – 이름과 설명 달기

- **rdfs:label**
  - 자원에 대한 **이름**을 정의
- **rdfs:comment**
  - 자원에 대한 **설명**을 작성

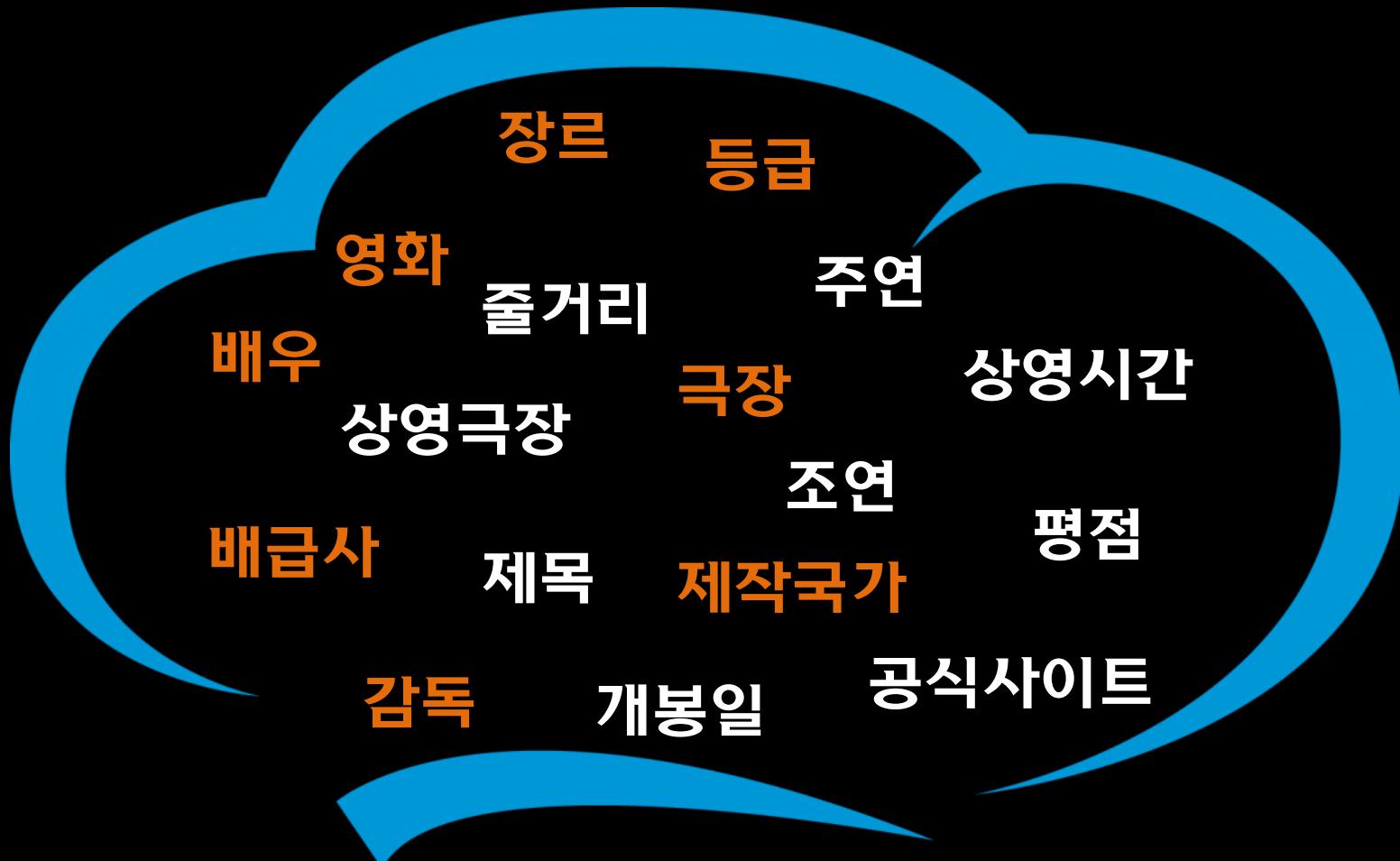


# 클래스는 상하위 관계를 가질 수 있어요.



Male 클래스 → rdfs:subClassOf → Person 클래스  
Female 클래스 → rdfs:subClassOf → Person 클래스

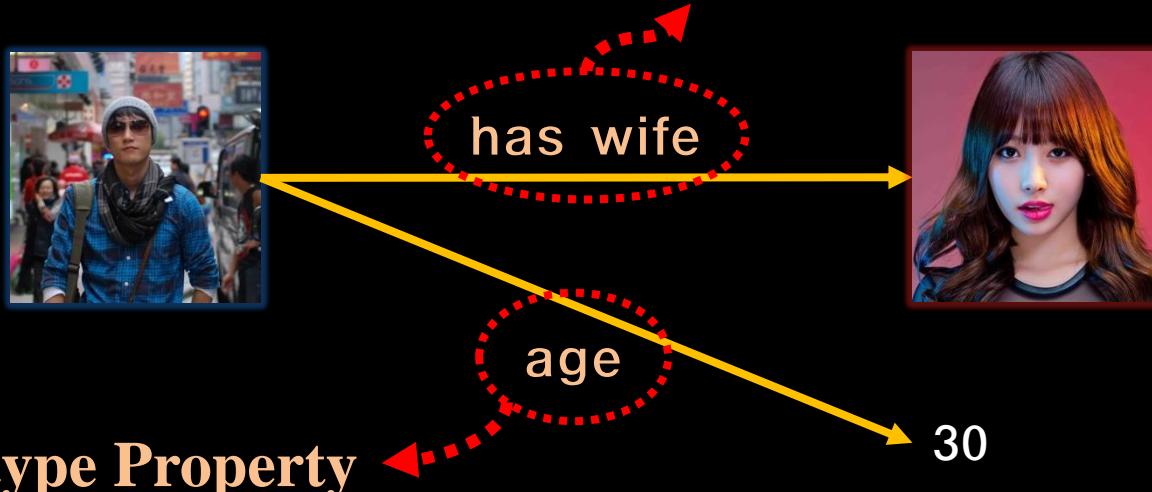
# 클래스가 될만한 것들은 어떤 것일까요?



### 3. 속성 정의

- 속성
  - 둘 사이의 관계를 나타내기 위한 것
- 속성의 두 가지 유형
  - 자원과 자원 사이의 관계: Object Property
  - 자원과 값 사이의 관계: Datatype Property

#### Object Property



# Datatype Property

- 자원과 값 사이의 관계

≒ 데이터베이스의 일반 필드

≒ 객체지향의 문자열과 기본 데이터타입 변수

Person Table

rrn	name	affiliation
841002-1045617	Myungjin Lee	10
410203-3983612	Gildong Hong	20
841105-2056143	Grace Byun	10

School Table

sid	sname
10	Yonsei Univ.
20	Seoul Univ.

## Datatype Property

```
public class Person {  
    String rrn;  
    String name;  
    School affiliation;  
}
```

```
public class School {  
    String sname;  
}
```

# Object Property

- 자원과 자원 사이의 관계

≒ 데이터베이스의 기본키와 외래키의 관계  
≒ 객체지향의 객체타입 변수

Person Table

rrn	name	affiliation
841002-1045617	Myungjin Lee	10
410203-3983612	Gildong Hong	20
841105-2056143	Grace Byun	10

School Table

sid	sname
10	Yonsei Univ.
20	Seoul Univ.

Object Property

```
public class Person {  
    String rrn;  
    String name;  
    School affiliation;  
}
```

```
public class School {  
    String sname;  
}
```

# 두 가지 유형의 속성

```
CREATE TABLE Person {  
    rrn      varchar(14) primary key  
    name     varchar(10)  
    FOREIGN KEY (affiliation) REFERENCES school(sid)  
}
```

Datatype Property

ObjectProperty

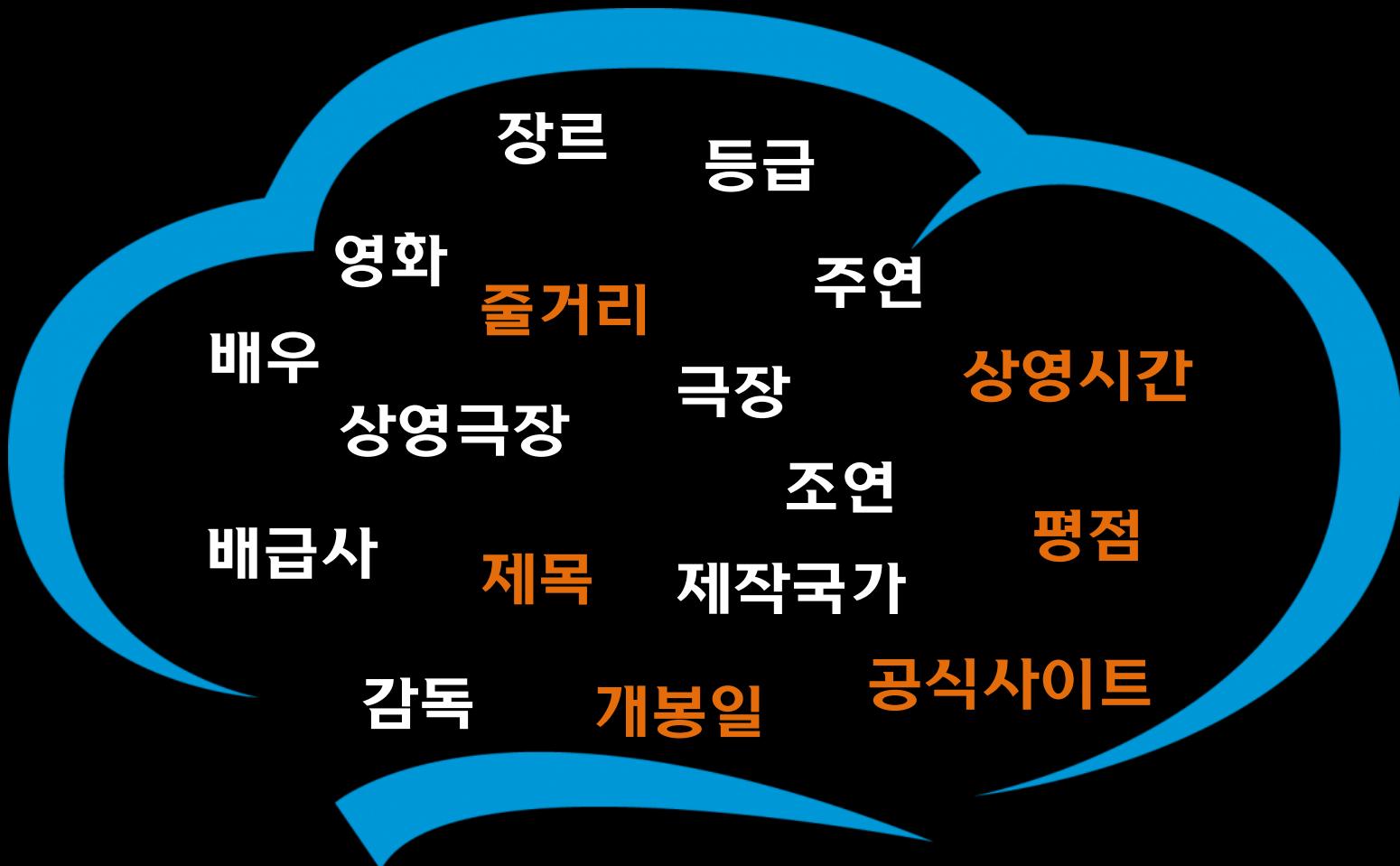
Person Table

rrn	name	affiliation
841002-1045617	Myungjin Lee	10
410203-3983612	Gildong Hong	20
841105-2056143	Grace Byun	10

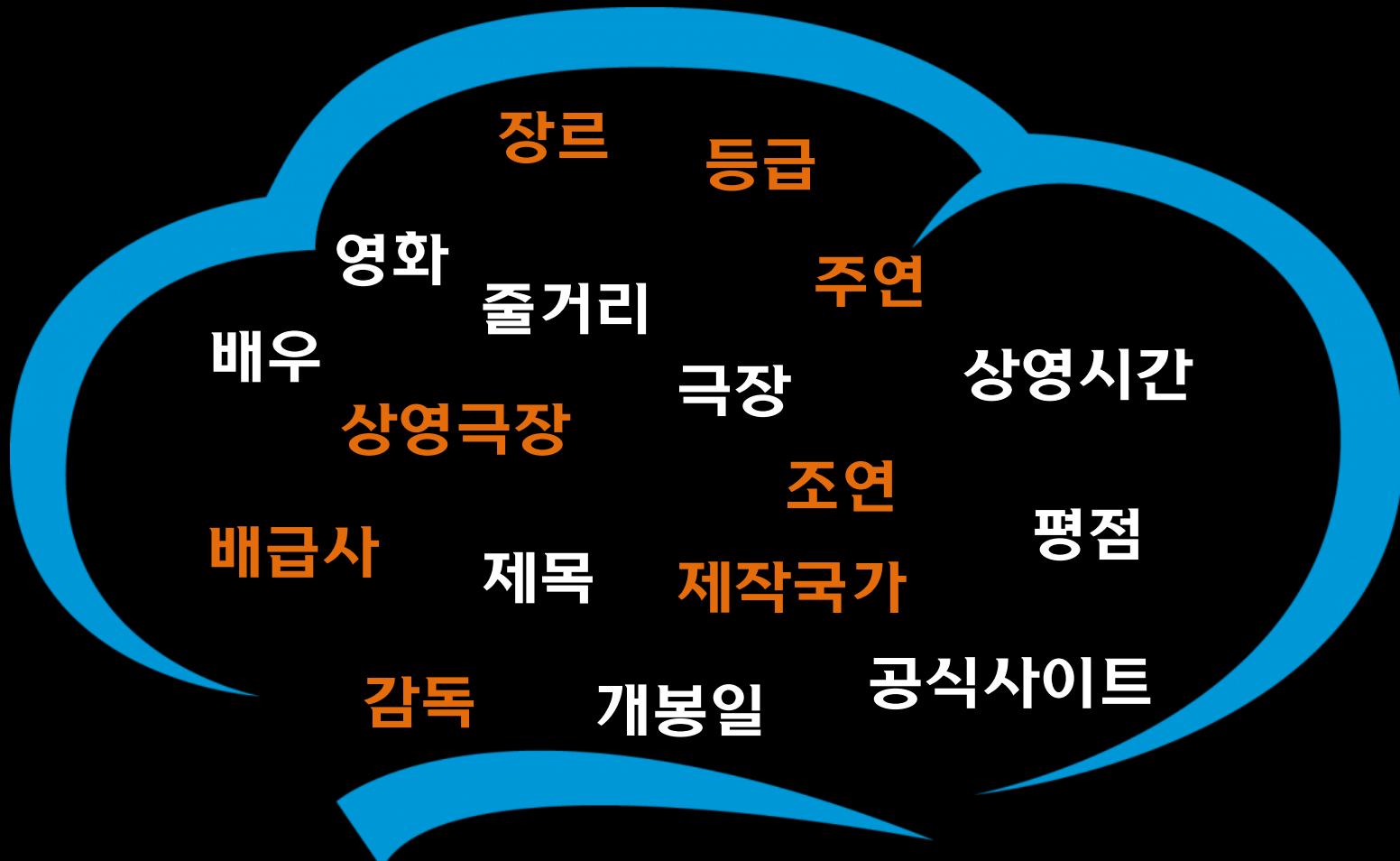
School Table

sid	sname
10	Yonsei Univ.
20	Seoul Univ.

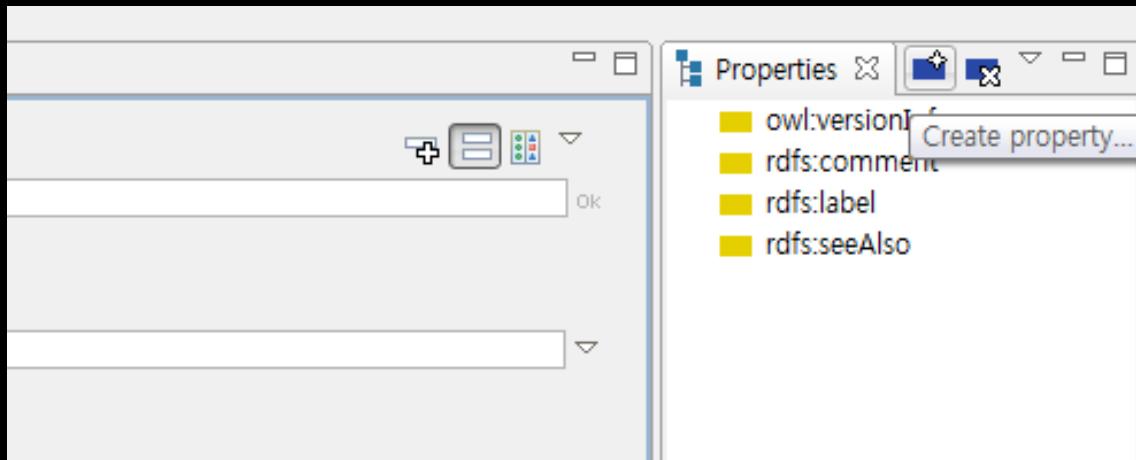
# Datatype Property가 될만한 것들은 어떤 것일까요?



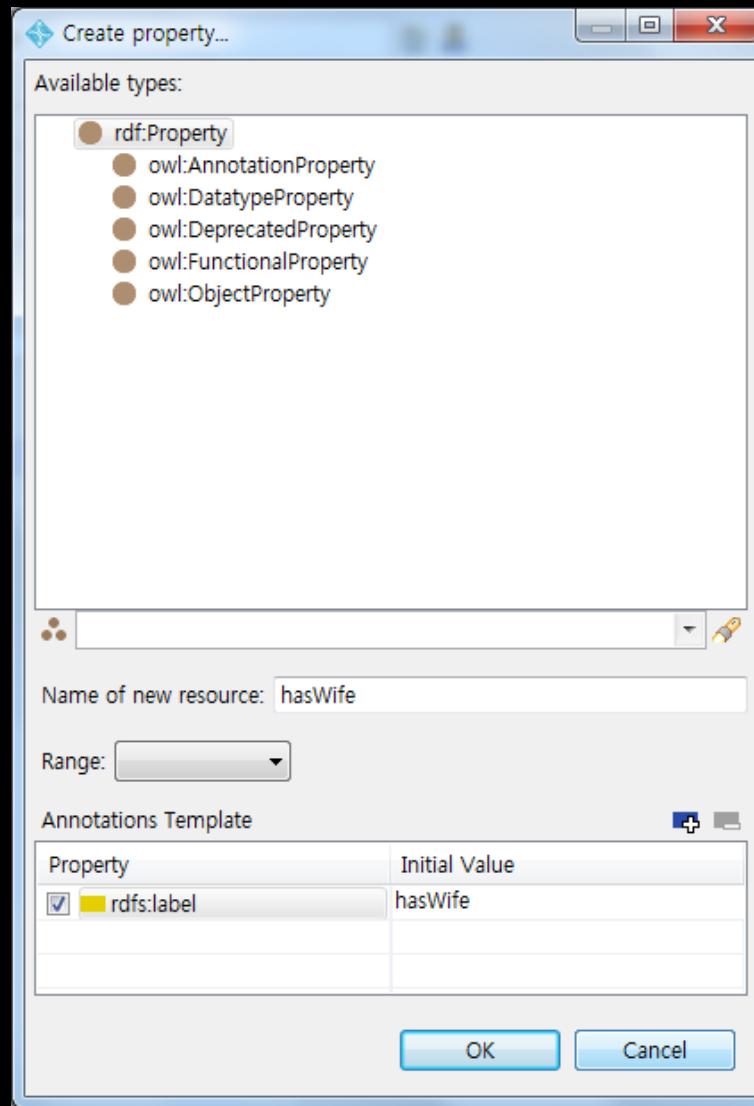
# Object Property가 될만한 것들은 어떤 것일까요?



# 따라 해 봅시다.

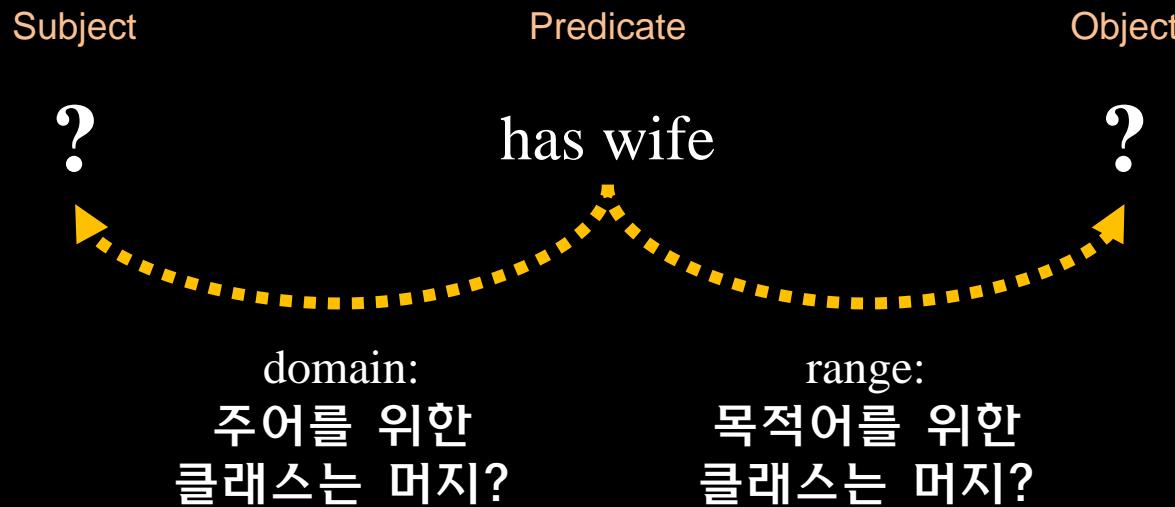


# 따라 해 봅시다.



# 속성의 간단한 제약 만들기

- rdfs:domain
  - 속성을 가질 수 있는 클래스가 무엇인가?
- rdfs:range
  - 속성 값의 타입은 무엇인가?



```
CREATE TABLE Person {  
    rrn    varchar(14) primary key  
    name   varchar(10)  
    FOREIGN KEY (affiliation) REFERENCES school (sid)  
}
```

name 속성의 range

name 속성의 domain

name Datatype Property

```
public class Person {  
    String rrn;  
    String name;  
    School affiliation;  
}
```

```
CREATE TABLE Person {  
    rrn    varchar(14) primary key  
    name   varchar(10)  
    FOREIGN KEY (affiliation) REFERENCES school (sid)  
}
```

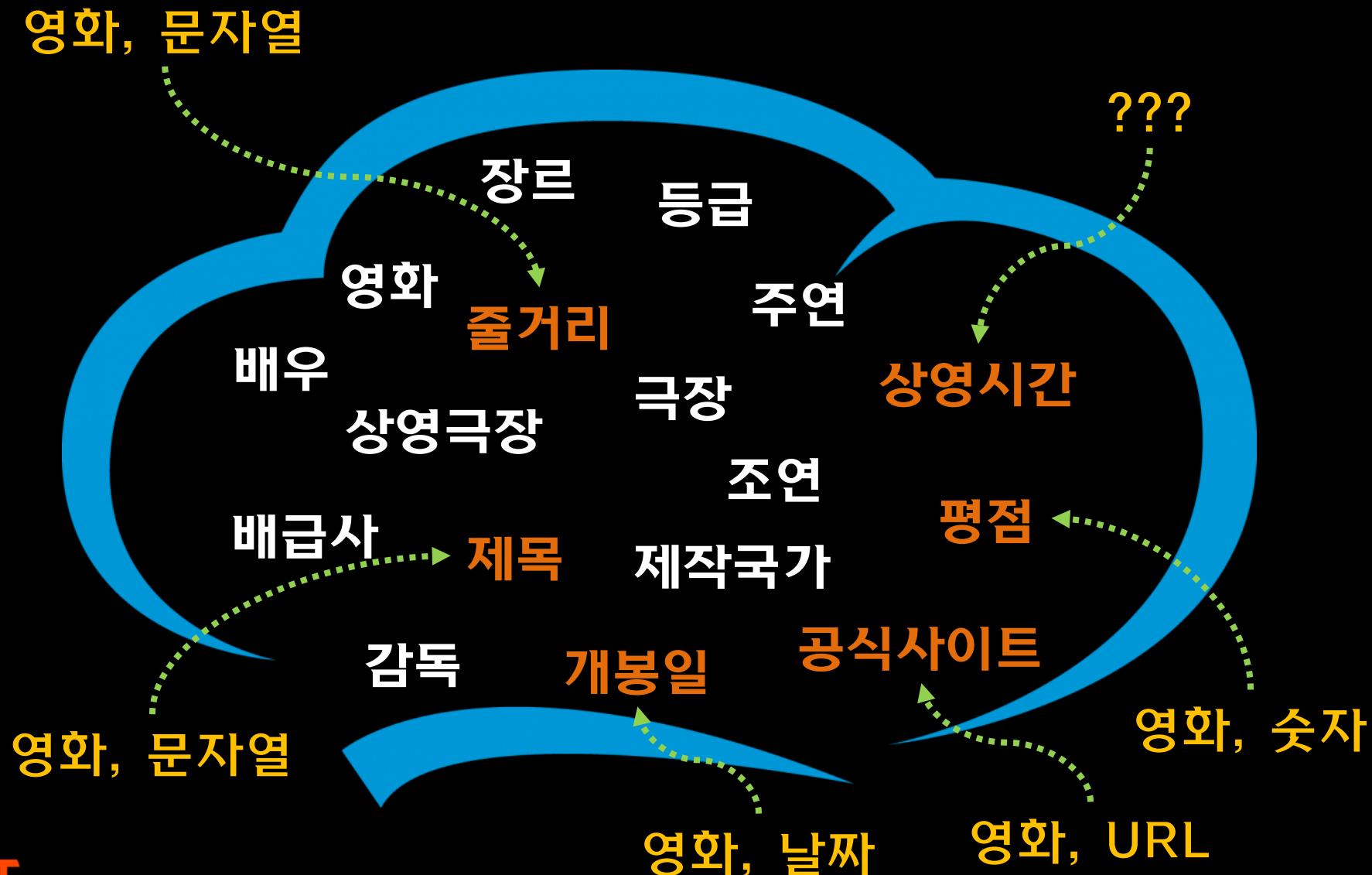
affiliation 속성의 domain

affiliation Object Property

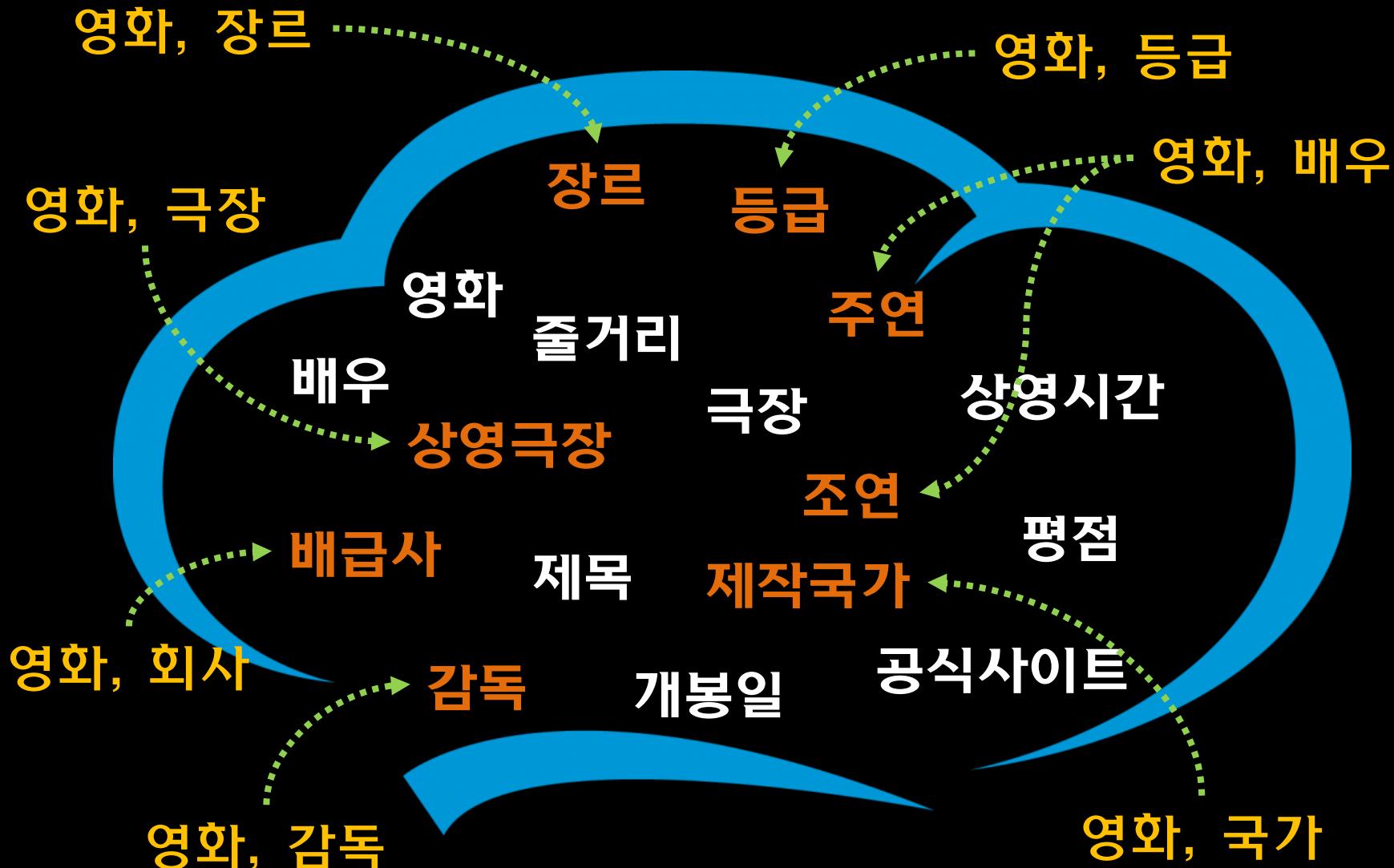
affiliation 속성의 range

```
public class Person {  
    String rrn;  
    String name;  
    School affiliation;  
}
```

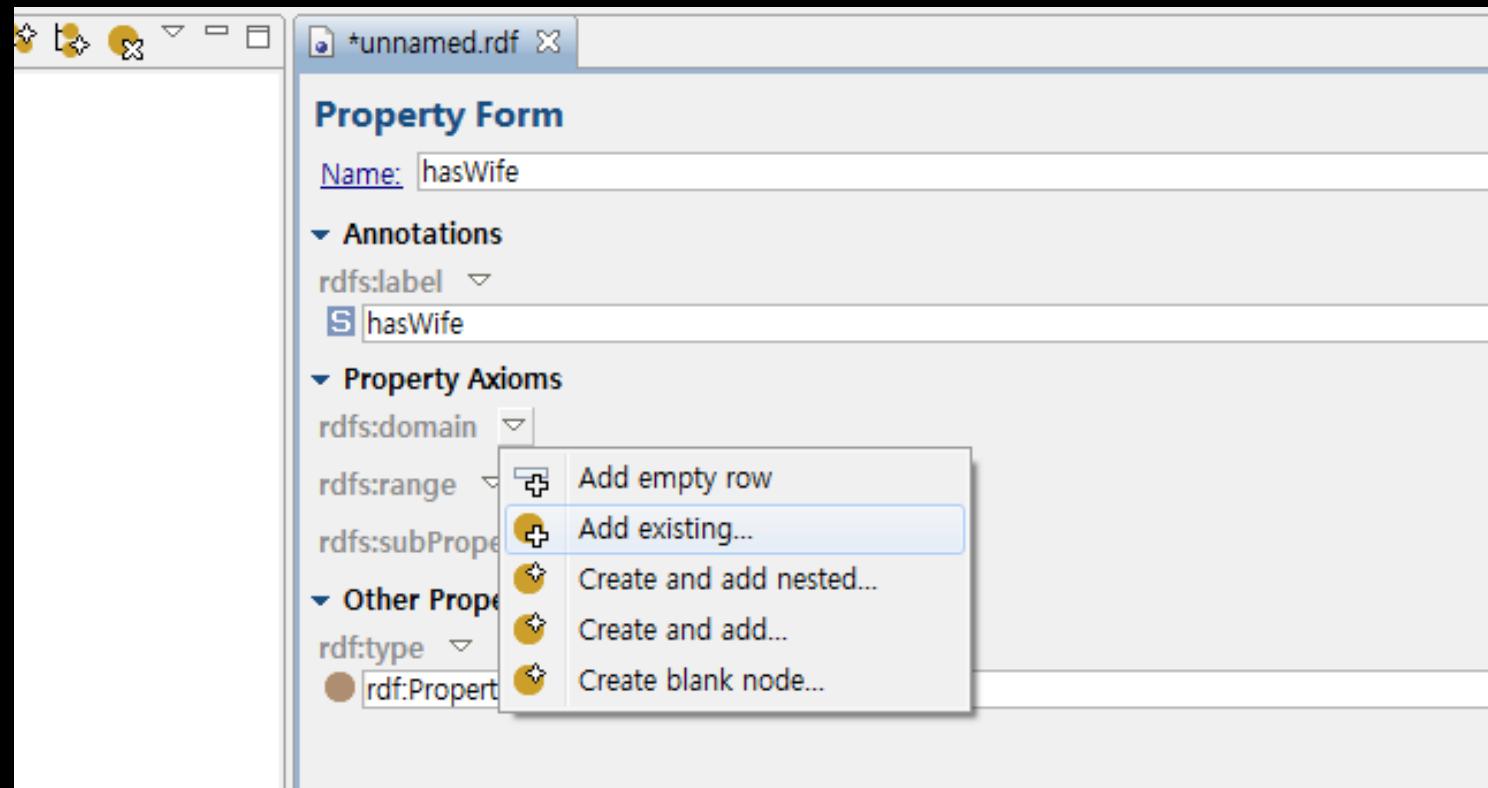
# Datatype Property의 domain과 range 맞추기



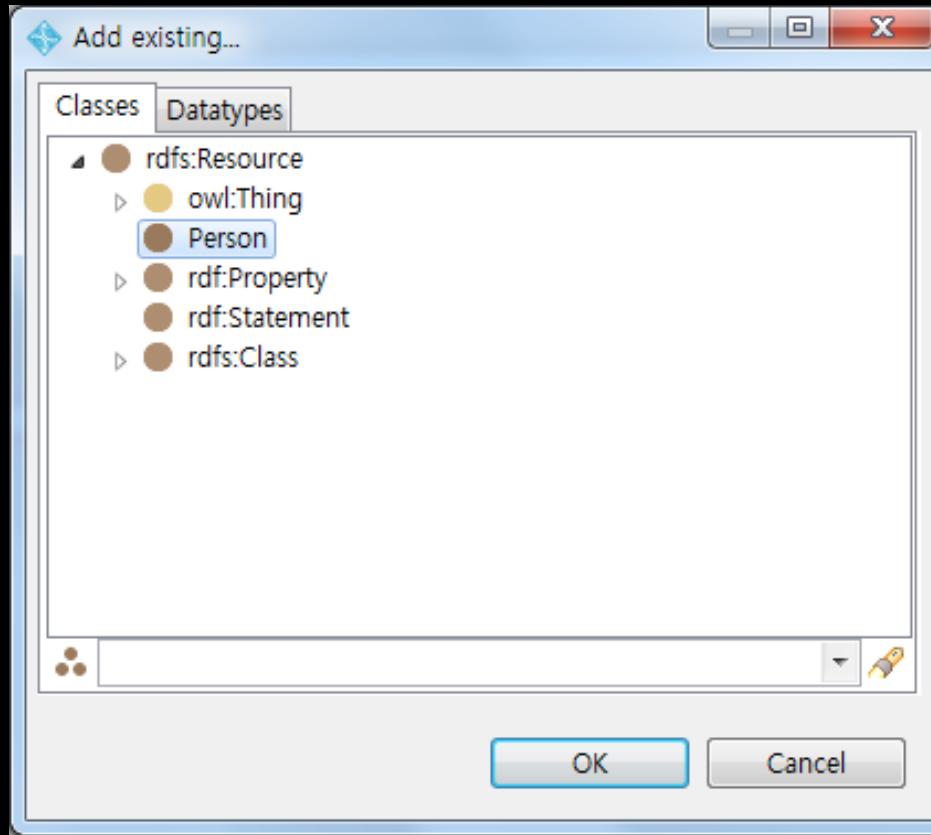
# Object Property의 domain과 range 맞추기



# 따라 해 볼시다.

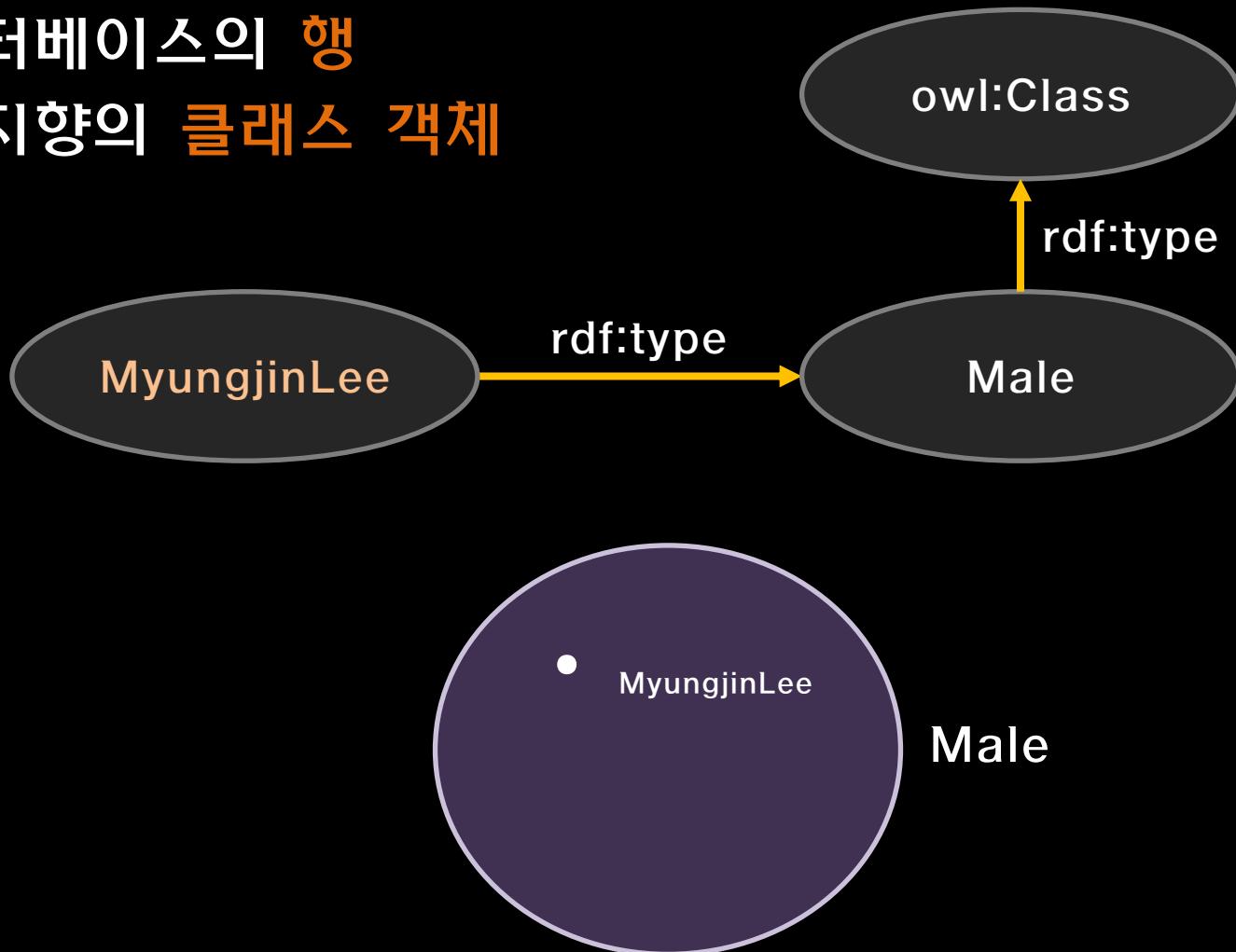


# 따라 해 봅시다.



## 4. 인스턴스 구축

- 클래스를 구성하는 멤버
  - ≒데이터베이스의 행
  - ≒객체지향의 클래스 객체



Person Table

rrn	name	affiliation
841002-1045617	Myungjin Lee	10
410203-3983612	Gildong Hong	20
841105-2056143	Grace Byun	10

School Table

sid	sname
10	Yonsei Univ.
20	Seoul Univ.

Person 클래스의 인스턴스

School 클래스의 인스턴스

```
public static void main(String[] args) {  
    Person p = new Person();  
    School s = new School();  
}
```

# 따라 해 봅시다.

The screenshot shows the JBoss Seamstress interface for defining an RDF class. On the left, a tree view of available namespaces and classes is displayed:

- rdfs:Resource (177)
- owl:Thing
- Person
- rdf:Property (91)
- rdf:Statement
- rdfs:Class (83)

The main panel shows the "Class Form" for the "Person" class. The "Name" field is set to "Person". Under "Annotations", the "rdfs:label" field is also set to "Person". Under "Class Axioms", the "rdfs:subClassOf" field is set to "rdfs:Resource". There is also a "Other Properties" section.

At the bottom, there are tabs for "Form" (selected) and "Source Code". Below the tabs is a toolbar with icons for "Imports", "Instances", "Domain", "Relevant Properties", "Error Log", and "SPARQL". A table at the bottom lists the properties of the "Person" class:

[Resource]	rdf:type	rdfs:label

# 따라 해 봅시다.

**Class Form**

Name: Person

▼ Annotations

rdfs:label ▾  
S Person

▼ Class Axioms

rdfs:subClassOf ▾  
● rdfs:Resource

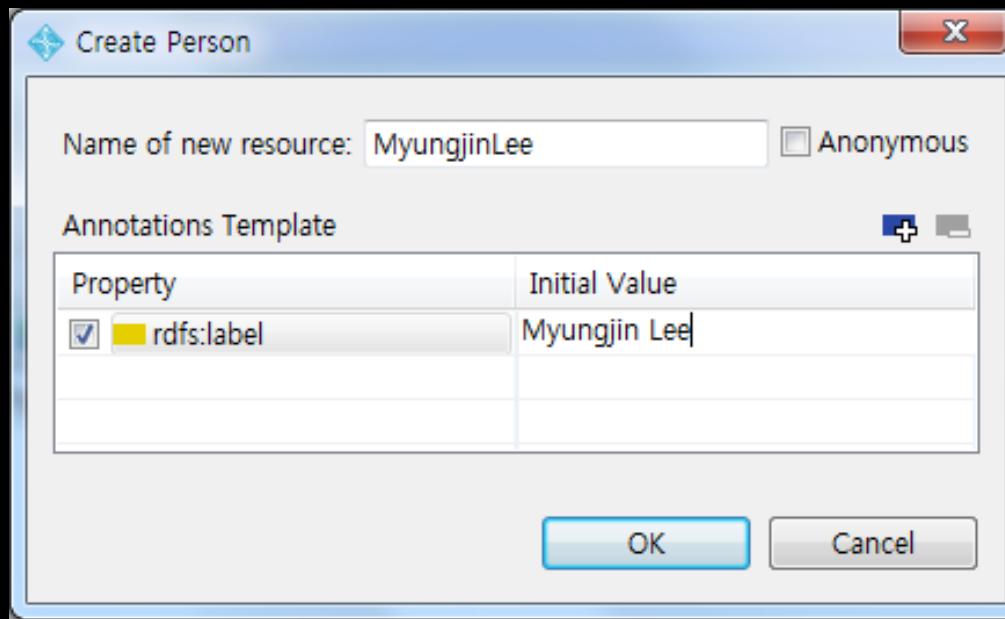
▶ Other Properties

Form Source Code

Imports Instances X Domain Relevant Properties Error Log SPARQL

[Resource]	rdf:type	rdfs:label	rdfs:comment	Create instance

# 따라 해 봅시다.



# 모든 인스턴스를 손으로 넣어야 하나?

실제로는 수작업으로 인스턴스를 만들지 않아요.

주로 변환기라는 것을 이용해서 데이터베이스의 데이터를 LOD로 변환하지요.

- R2RML
  - RDB to RDF Mapping Language
  - 관계형 데이터베이스에 저장된 데이터를 RDF의 구조에 맞추어 변환을 수행하기 위한 매팅 언어

RDB

EMP			
EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APP SERVER	NEW YORK

R2RML

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.  
@prefix ex: <http://example.com/ns#>.  
  
<#TriplesMap1>  
    rr:logicalTable [ rr:tableName "EMP" ];  
    rr:subjectMap [  
        rr:template "http://data.example.com/employee/{EMPNO}";  
        rr:class ex:Employee;  
    ];  
    rr:predicateObjectMap [  
        rr:predicate ex:name;  
        rr:objectMap [ rr:column "ENAME" ];  
    ].
```

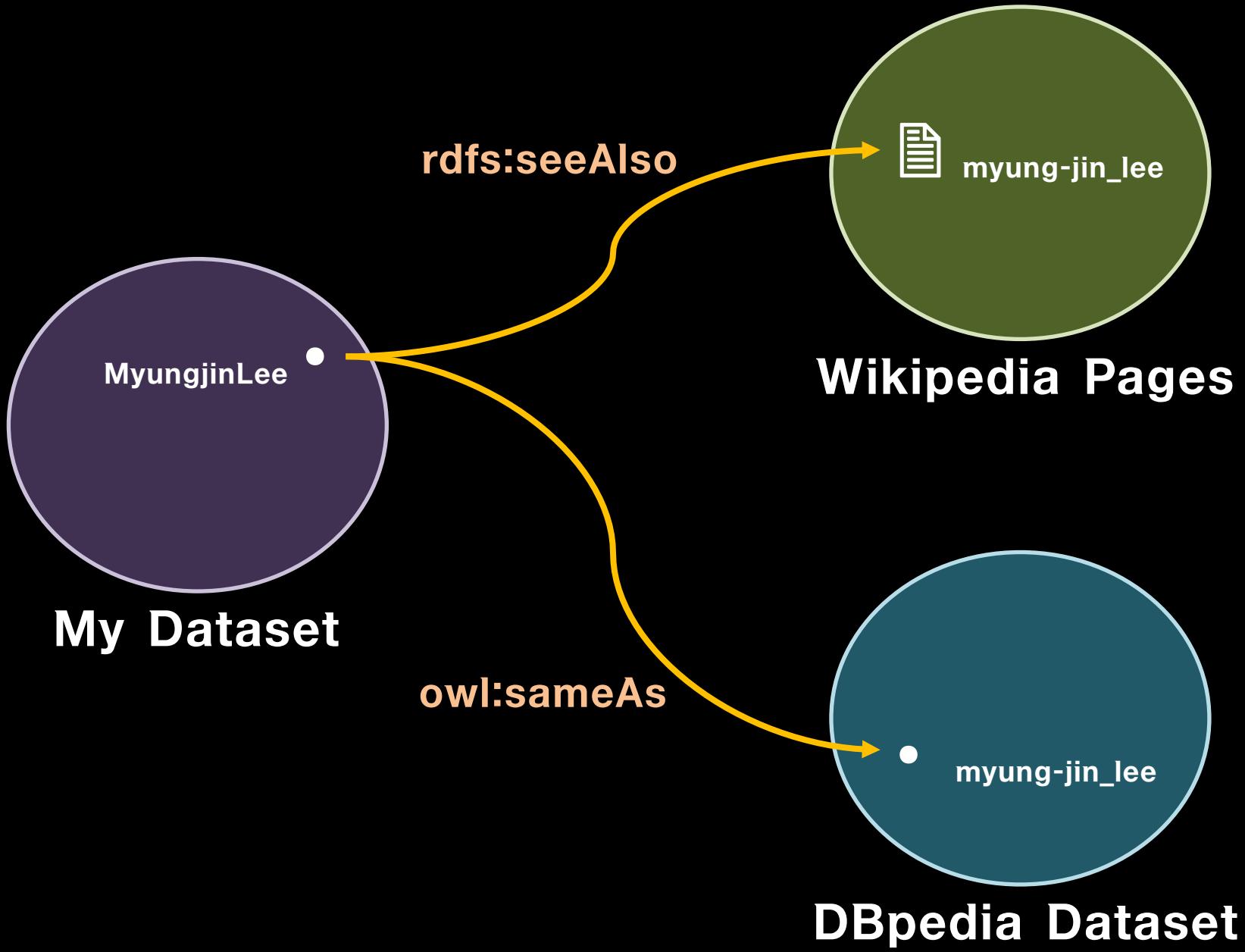
Result

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.  
<http://data.example.com/employee/7369> ex:name "SMITH".
```

# 5. 외부 자원과의 Interlinking

놓치기 쉬운 부분 중 하나지만 너무나도 중요해요.

- Interlinking이란?
  - 외부에 존재하는 자원과의 연결
- 인스턴스 Interlinking의 두 가지
  - rdfs:seeAlso
    - 인스턴스에 대한 추가적인 정보 제공을 목적으로 해당 자원과 연결하는 것
  - owl:sameAs
    - 의미적으로 동일한 외부의 인스턴스와 연결하는 것



# 따라 해 볼시다.

\*unnamed.rdf

### Resource Form

Name: MyungjinLee

Annotations

rdfs:label ▾  
S Myungjin Lee

Other Properties

hasWife ▾  
rdf:type ▾  
● Person

Incoming References

Add Annotations ▾  
Add Other Properties ▾

Form Source Code

Imports Instances Domain

[Resource] rdf  
↳ MyungjinLee Pe

- owl:equivalentClass
- owl:equivalentProperty
- owl:inverseOf
- owl:sameAs
- owl:topObjectProperty
- owl:bottomDataProperty
- owl:topDataProperty
- rdf:first
- rdf:object
- rdf:predicate
- rdf:rest
- rdf:subject
- rdf:value
- rdfs:domain
- rdfs:member
- rdfs:range
- rdfs:subClassOf
- rdfs:subPropertyOf
- xsd:fractionDigits
- xsd:length
- xsd:maxExclusive
- xsd:maxInclusive

SPARQL

label

ngjin Lee

The screenshot shows the JBoss Seam Resource Form editor interface. On the left, there's a tree view of properties: Annotations (rdfs:label), Other Properties (hasWife, rdf:type), and Incoming References. Under Annotations, 'rdfs:label' is expanded to show 'Myungjin Lee'. Under Other Properties, 'rdf:type' is expanded to show 'Person'. At the bottom, there are tabs for 'Form' and 'Source Code', and buttons for 'Imports', 'Instances', and 'Domain'. A context menu is open at the bottom right of the main area, with options 'Add Annotations' and 'Add Other Properties'. To the right of the main form, a large list of RDF properties is displayed, many of which are highlighted in blue, indicating they are currently selected or being used. A vertical scroll bar is visible on the right side of the list.

# 데이터를 발행해 봅시다.



OntoBase도 있지만

오늘은 Virtuoso를

쓰도록 할께요.

# Virtuoso 설치

- OpenLink Software의 Virtuoso
  - LOD 운용을 위한 플랫폼
  - 현재 ver. 7까지 있으며, Open Source로도 배포를 하고 있어요.
- Virtuoso 설치 및 실행
  - 기본 Document
    - <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/>
  - Windows 환경
    - <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSUsageWindows>
  - Linux 계열 환경
    - <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSMake>

# Virtuoso를 이용한 발행 절차

1. 작성한 온톨로지 업로드
2. 가상 디렉토리 생성
3. URL Rewriting 규칙 작성
4. 발행 테스트

# Virtuoso Conductor

- <http://localhost:8890/conductor/>

The screenshot shows the Virtuoso Conductor web interface. On the left, there is a sidebar with a login form for 'Account' (username and password fields, 'Remember me' checkbox, and a 'Login' button). Below the login are links for 'Interactive SQL (ISQL)', 'Virtuoso Start Menu', 'Documentation (web)', 'Tutorials (web)', 'Virtuoso Web Site', and 'OpenLink Software'. At the bottom of the sidebar, it says 'Version: 06.01.3127' and 'Build: Jul 22 2013'. The main content area features a navigation bar with tabs: Home, System Admin, Database, Replication, Web Application Server, XML, Web Services, Linked Data, and NNTP. The 'Home' tab is selected. To the right of the navigation bar is a status message 'not logged in | Home'. The central part of the page displays a circular diagram representing 'BUSINESS PROCESS INTEGRATION'. The diagram is divided into segments labeled 'Enterprise Data Management', 'Enterprise Information Integration', 'Web Services Platform', 'Enterprise Collaboration', and 'Business Process Integration'. A central figure of a person is shown interacting with the diagram. At the bottom right of the main content area, there is a copyright notice: 'Copyright © 1998-2014 OpenLink Software'.

# 1. 작성한 온톨로지 업로드

VIRTUOSO CONDUCTOR

logged in as dba | Log out | Home

Home System Admin Database Replication Web Application Server XML Web Services Linked Data NNTP

SPARQL Sponger Statistics Graphs Schemas Namespaces Views Quad Store Upload

Quad Store Upload

File\* 파일 선택 test.rdf

Resource URL\*

Create graph explicitly

Named Graph IRI http://localhost:8890/test

Cancel Upload

Copyright © 1998-2014 OpenLink Software

# 업로드 확인

The screenshot shows the Virtuoso Conductor interface. The top navigation bar includes links for Home, System Admin, Database, Replication, Web Application Server, XML, Web Services, Views, Quad Store Upload, Linked Data, and NNTP. The Views link is highlighted with a red dashed circle. Below the navigation is a sub-menu for SPARQL, Sponger, Statistics, Graphs, Schemas, Namespaces, Views, Quad Store Upload, and Linked Data. The Views link in this sub-menu is also circled in red.

The main area is titled "SPARQL Execution". It has tabs for "Query" and "Saved Queries", with "Query" selected. A "Default Graph IRI" input field is present. The "Query" text area contains the following SPARQL code:

```
SELECT *  
FROM <http://localhost:8890/test>  
WHERE {  
    ?x ?y ?z .  
}
```

Below the query is a button bar with "Execute", "Save", "Load", and "Clear" buttons. The "Execute" button is highlighted with a red dashed circle. The results are displayed in a table:

x	y	z
<a href="http://localhost:8890/test/">http://localhost:8890/test/</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/2002/">http://www.w3.org/2002/</a>
<a href="http://localhost:8890/test/">http://localhost:8890/test/</a>	<a href="http://www.w3.org/2002/07/owl#versionInfo">http://www.w3.org/2002/07/owl#versionInfo</a>	"Created with TopBraid Composer™<a href="http://www.w3.org/2002/07/owl#versionInfo">"Created with TopBraid Composer™</a>"
<a href="http://localhost:8890/test/Movie">http://localhost:8890/test/Movie</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/2002/">http://www.w3.org/2002/</a>

# 업로드 확인 - SPARQL

```
SELECT *
FROM <http://localhost:8890/test>
WHERE {
  ?x ?y ?z .
}
```

## 2. 가상 디렉토리 생성

The screenshot shows the Virtuoso Conductor web interface. A red dashed circle highlights the top navigation bar, which includes links for Home, System Admin, Database, Replication, Web Application Server, XML, Web Services, Linked Data, and NNTP. Below this, another red dashed box highlights the 'Virtual Domains & Directories' link under the 'Content Management' section. The main content area is titled 'Hosted Domains and Virtual Directories'. It displays a table with two entries:

Interface	Port	HTTP Host *	Action
0.0.0.0	8890	{Default Web Site}	<a href="#">New Directory</a>
0.0.0.0	0.0.0.0	{Default SSL Web Site}	<a href="#">New Directory</a>

A red dashed arrow points from the 'Action' column of the first row to the 'Add' button at the bottom right of the table. The bottom right corner of the interface contains the copyright notice: 'Copyright © 1998-2014 OpenLink Software'.



logged in as dba | [Log out](#) | [Home](#)

[Interactive SQL \(ISQL\)](#)  
[WebDAV Browser](#)  
[Virtuoso Start Menu](#)

[Documentation \(web\)](#)  
[Tutorials \(web\)](#)  
[Virtuoso Web Site](#)  
[OpenLink Software](#)

Version: 06.01.3127  
Build: Jul 22 2013

Home System Admin Database Replication Web Application Server XML Web Services Linked Data NNTP

Content Management

Virtual Domains & Directories

## HTTP Virtual Directory

Please choose virtual directory type or an existing virtual directory to use as template.

None

Existing path

Type

[Cancel](#)

[Next>>](#)

Copyright © 1998-2014 OpenLink Software

 Interactive SQL (ISQL)  
 WebDAV Browser  
 Virtuoso Start Menu

 Documentation (web)  
 Tutorials (web)

 Virtuoso Web Site  
 OpenLink Software

Version: 06.01.3127  
Build: Jul 22 2013

Home System Admin Database Replication Web Application Server XML Web Services Linked Data NNTP

Content Management Virtual Domains & Directories

## HTTP Virtual Directory

### Virtual Directory Information

Host \*ini\*  
Interface \*ini\*  
Path    
 Default directory  
 Physical path is a WebDAV repository

Physical path /

Default page

### Permissions

Style Sheet for browsing   
 Allow Directory Browsing  
 Allow XML template execution  
 Override exec permission flag in WebDAV  
 Allow persistent session variables

VSP User

### Authentication options

Method   
Realm   
Authentication Function

### Advanced

Cross-Origin Resource Sharing

Reject Unintended CORS

Post-processing Function

# 3. URL Rewriting 규칙 작성

VIRTUOSO CONDUCTOR

logged in as dba | Log out | Home

Home System Admin Database Replication Web Application Server XML Web Services Linked Data NNTP

Content Management Virtual Domains & Directories

Help

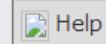
Hosted Domains and Virtual Directories

Interface	Port	HTTP Host *	Action
0.0.0.0	8890	{Default Web Site}	+ New Directory
/rdf_net	FS	*disabled*	Edit Delete URL-rewrite Export
/sparql	SPARQL	dba	Edit Delete URL-rewrite Export
/sparql-auth	SPARQL	dba	Edit Delete URL-rewrite Export
/sparql-graph-crud	FS	dba	Edit Delete URL-rewrite Export
/sparql-graph-crud-auth	FS	dba	Edit Delete URL-rewrite Export
/sparql-sd	FS	dba	Edit Delete URL-rewrite Export
/test	FS	dba	Edit Delete URL-rewrite * Export
/uriqa	FS	dba	Edit Delete URL-rewrite Export
/vsmx	FS	dba	Edit Delete URL-rewrite Export
0.0.0.0		{Default SSL Web Site}	+ New Directory

Version: 06.01.3127  
Build: Jul 22 2013

Copyright © 1998-2014 OpenLink Software

## URL rewriting rules for /test



URL rewrite rules		Content negotiation	Rulelist Container	Action
Source pattern	Destination			
(/[^#]*)	/sparql?query=CONSTRUCT%20%7B%20%3Chttp%3A%2F%2Flo ...			Delete  Up  Down
(/[^#]*)	/sparql?query=SELECT%20%3Fs%20%3Fp%20%3Fo%20WHERE% ...			Delete  Up  Down

Rulelist container: Main\*

Pattern Type: REGEX

Rule matching: Last matching

Request Path pattern: /test/(.\*)\$

Accept header Request pattern:

Destination Path format: /test/\$1

SPARQL

Destination format Conversion function:

HTTP Response Code: Internal redirect

HTTP Response Headers:

Add

Back

### 3.1. HTML 요청을 위한 URL Rewriting 규칙

- Rule matching
  - Normal
- Request Path pattern
  - ( $/[^#]^*$ )
- Accept Header Request pattern
  - (text/html)
- Destination Path format
  - /sparql?query=SELECT%20%3Fs%20%3Fp%20%3Fo%20WHERE%20%7B%20%3Fs%20%3Fp%20%3Fo%20%20.%20FILTER%20(%3Fs%20%3D%20%3Chttp%3A%2F%2Flocalhost%3A8890\$U1%3E)%20%7D&format=\$accept

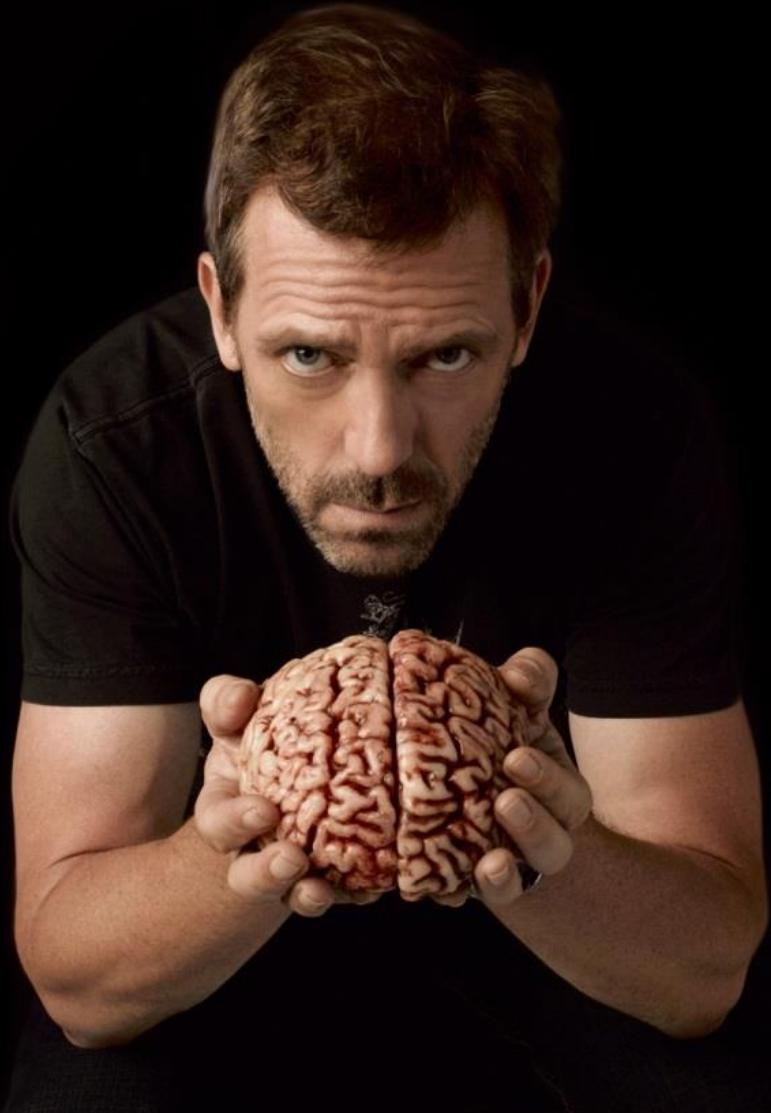
## 3.2. RDF 요청을 위한 URL Rewriting 규칙

- Rule matching
  - Normal
- Request Path pattern
  - $(/[^#]^*)$
- Accept Header Request pattern
  - (text/rdf.n3) | (application/rdf.xml)
- Destination Path format
  - /sparql?query=CONSTRUCT%20%7B%20%3Chttp%3A%2F%2Flocalhost%3A8890\$U1%3E%20%3Fp%20%3Fo%20%7D%20WHERE%20%7B%20%7B%20%3Chttp%3A%2F%2Flocalhost%3A8890\$U1%3E%20%3Fp%20%3Fo%20%7D%20.%20%7D&format=\$accept

## 4. 발행 테스트

- <http://localhost:8890/test/Movie>

s	p	o
<a href="http://localhost:8890/test/Movie">http://localhost:8890/test/Movie</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/2002/07/owl#Class">http://www.w3.org/2002/07/owl#Class</a>
<a href="http://localhost:8890/test/Movie">http://localhost:8890/test/Movie</a>	<a href="http://www.w3.org/2000/01/rdf-schema#label">http://www.w3.org/2000/01/rdf-schema#label</a>	"Movie"^^< <a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a> >
<a href="http://localhost:8890/test/Movie">http://localhost:8890/test/Movie</a>	<a href="http://www.w3.org/2000/01/rdf-schema#subClassOf">http://www.w3.org/2000/01/rdf-schema#subClassOf</a>	<a href="http://www.w3.org/2002/07/owl#Thing">http://www.w3.org/2002/07/owl#Thing</a>



LOD를  
활용해 봅시다.

U S E I T

# SPARQL

데이터베이스에 저장된 데이터를 사용하려면 SQL을 알아야 하는 것처럼  
LOD 형태의 데이터를 사용하기 위해서는 LOD를 위한 질의언어를 알아야 해요.

- SPARQL Protocol and RDF Query Language
  - SPARQL 위한 프로토콜에 대한 정의
  - RDF 질의를 위한 질의 문법에 대한 정의
  - W3C Recommendation 21 March 2013
  - <http://www.w3.org/TR/sparql11-query/>

# SPARQL의 구성

## SQL

```
SELECT name FROM Person WHERE rrn = “841002-1045617”
```

원하는 결과

질의 대상

질의 조건

```
SELECT ?y  
FROM <http://localhost:8890/test>  
WHERE {  
    ?x <http://localhost:8890/rrn> “841002-1045617”  
    ?x <http://localhost:8890/name> ?y .  
}
```

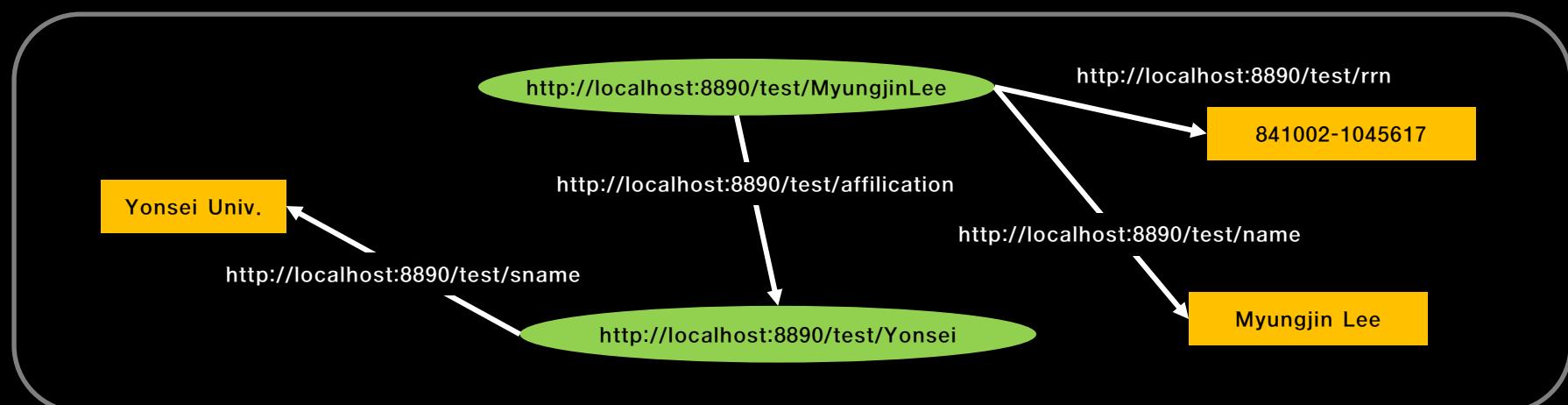
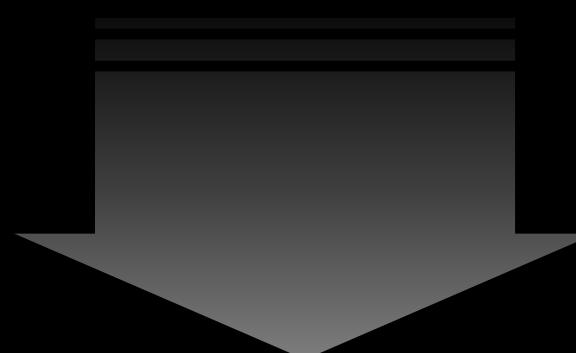
## SPARQL

## Person Table

rrn	name	affiliation
841002-1045617	Myungjin Lee	10
410203-3983612	Gildong Hong	20
841105-2056143	Grace Byun	10

## School Table

sid	sname
10	Yonsei Univ.
20	Seoul Univ.



<http://localhost:8890/test> 그래프

# RDF가 그래프 구조를 갖는다는 것만 기억하면 SPARQL 그리 어렵지 않아요.

SQL의 SELECT와 같이  
특정 정보를 꺼내오기  
위한 구문

그래프 구조이기 때문에 SQL과 달리  
필드를 지정하는 것이 아니라 변수를  
지정하여 해당 변수에 바인딩 된  
결과를 획득하기 위해 변수를 지정

SELECT

?y

지정된 그래프에서 결과를 추출

FROM

WHERE {

<<http://localhost:8890/test>>

조건을 지정하기 위한 영역 선언

?x <<http://localhost:8890/rrn>> “841002-1045617”.

?x <<http://localhost:8890/name>> ?y .

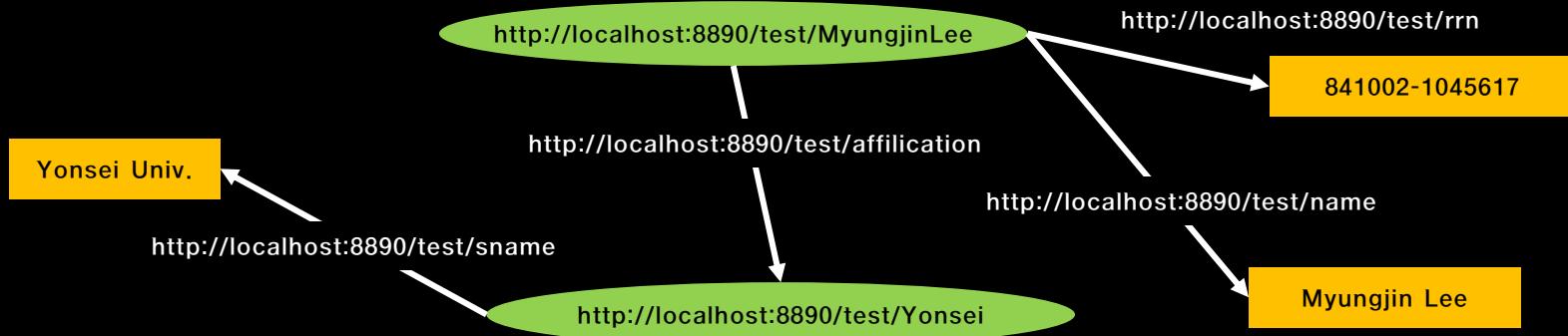
}

질의의 대상이 되는  
그래프를 지정  
(생략 가능. 생략할 경우  
전체를 대상으로 조회)

질의의 조건을 그래프의  
트리플 구조에 따른  
AND 형태로 기술

# 같이 해 볼까요?

- 모든 데이터를 가져오기 위한 질의

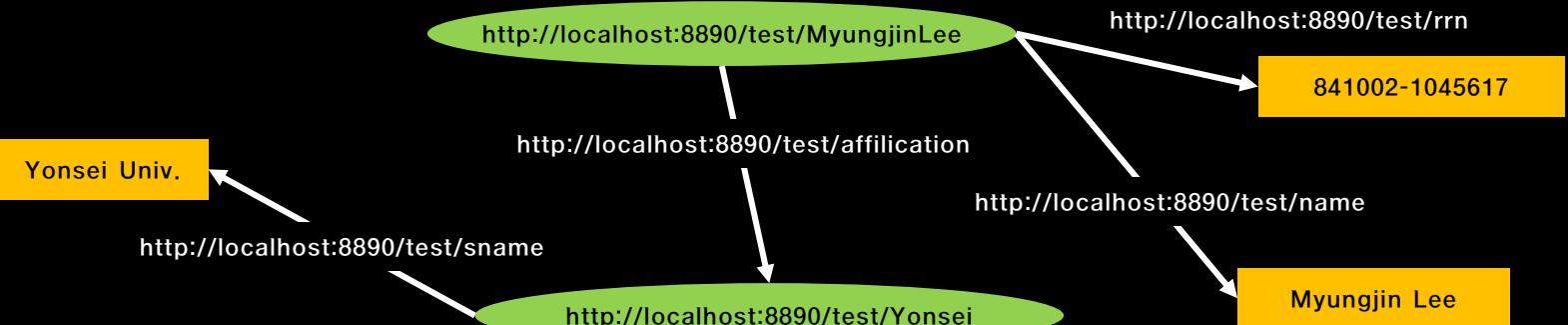


http://localhost:8890/test 그레프

```
SELECT *
WHERE {
    ?x      ?y      ?z .
}
```

# 같이 해 볼까요?

- 모든 학생의 데이터를 가져오기 위한 질의

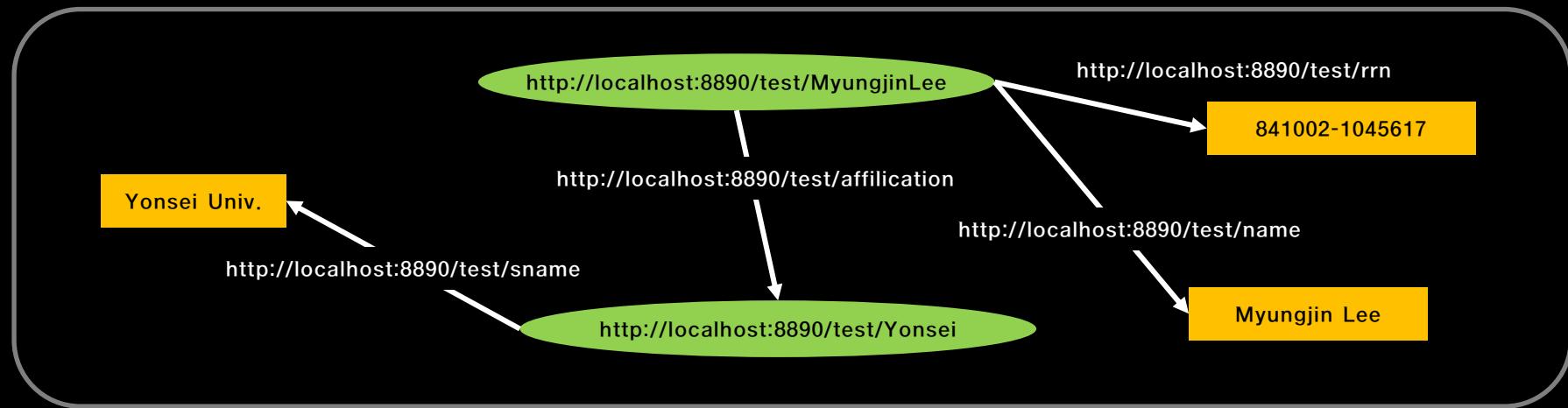


http://localhost:8890/test 그래프

```
SELECT *  
WHERE {  
    ?x  rdf:type  <http://localhost:8890/test/Student> .  
    ?x  ?y  ?z .  
}
```

# 같이 해 볼까요?

- <<http://localhost:8890/test/MyungjinLee>>의 이름을 가져오기 위한 질의

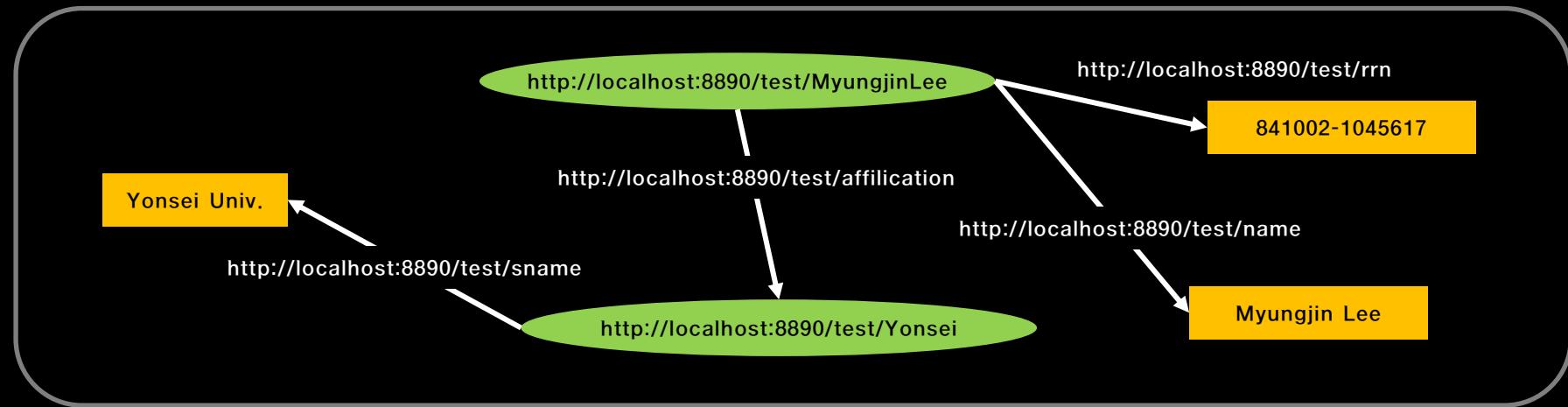


<http://localhost:8890/test> 그래프

```
SELECT *
WHERE {
    <http://localhost:8890/test/MyungjinLee>
    <http://localhost:8890/test/name> ?z .
}
```

# 같이 해 볼까요?

- <<http://localhost:8890/test/MyungjinLee>>가 소속된 학교의 이름을 가져오기 위한 질의



<http://localhost:8890/test> 그래프

```
SELECT ?b
WHERE {
    <http://localhost:8890/test/MyungjinLee>
        <http://localhost:8890/test/affilication> ?a .
    ?a <http://localhost:8890/test/sname> ?b . }
```

# 그 외의 SPARQL 질의 형태

다양한 질의 유형이 있으며 필요에 따라 이것 저것 사용하지만,  
굳이 몰라도 상관없어요.

- SELECT
  - 지정된 변수에 바인딩 된 값을 추출하기 위한 질의
- CONSTRUCT
  - SELECT와 유사하지만 결과를 그래프 형태로 구성하고자 할 경우 사용하는 질의
- ASK
  - 지정된 조건을 만족하는지에 따라 true와 false를 반환하는 질의
- DESCRIBE
  - 지정된 URI에 대한 RDF 데이터를 포함하는 그래프를 얻고자 사용하는 질의

# Virtuoso의 SPARQL Endpoint

- <http://localhost:8890/sparql>

**Virtuoso SPARQL Query Editor**

Default Data Set Name (Graph IRI)

About | Namespace Prefixes | Inference rules

Query Text

```
SELECT * WHERE { ?s rdf:type <http://localhost:8890/test/Movie> . ?s ?p ?o . }
```

질의 대상이 되는  
그래프 이름 입력 부분

SPARQL 입력 부분

(Security restrictions of this server do not allow you to retrieve remote RDF data, see [details](#).)

Results Format: HTML

Execution timeout: 0 milliseconds

Options:  Strict checking of void variables

결과의 형식 지정

(The result can only be sent back to browser, not saved on the server, see [details](#))

Run Query Reset

Copyright © 2014 [OpenLink Software](#)  
Virtuoso version 06.01.3127 on Win64 (x86\_64-generic-win-64), Single Server Edition

# 프로그래밍을 이용해 활용하기

- LOD를 프로그래밍적으로 활용하기 위해 알아야 할 Tip
  - LOD로 구축된 데이터는 HTTP를 이용해서 접근 가능해요.
  - SPARQL 또한 HTTP를 이용해서 질의를 수행할 수 있어요.
  - 그러니까 프로그래밍 언어에서 HTTP Connection을 이용하면 데이터를 얻을 수 있어요.
  - 잘 구축된 LOD는 HTTP 헤더에 지정된 MIME 타입에 따라 적절한 결과를 반환해 줘요.
  - SPARQL에 대한 지식은 필수는 아니지만 알아두어야 LOD를 잘 쓸 수 있어요.
  - 잘 구축된 SPARQL Endpoint는 질의 결과로 다양한 형태를 지원해요. (JSON, Triple, RDF/XML 등)
  - 그러니까 RDF 프로그래밍 방법을 몰라도 어느 정도 프로그래밍이 가능하지만, 알아두는 것이 좋아요.

# 간단한 예제 프로그램

## 자원에 대한 정보를 요청하는 자바 프로그램

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class SPARQLEndpointTest {

    public static void main(String[] args) {
        try {
            URL url = new URL("http://localhost:8890/test/Movie");
            HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
            urlConn.setRequestProperty("Accept", "application/rdf+xml");
            BufferedReader br = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
            String inputLine;
            while ((inputLine = br.readLine()) != null) {
                System.out.println(inputLine);
            }
            br.close();
        } catch (Exception e) {
            System.out.println("Exception : " + e.toString());
        }
    }
}
```

수용할 수 있는 형식을 application/rdf+xml로 지정했기 때문에 실행 결과는 RDF 형식의 XML 구문으로 반환 됩니다. 따라서 이를 프로그래밍으로 처리하는 과정이 필요하답니다.



# 결과는요

## 자원에 대한 정보를 요청하는 자바 프로그램의 결과

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:about="http://localhost:8890/test/Movie">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class" />
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Movie</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  </rdf:Description>
</rdf:RDF>
```

# 간단한 예제 프로그램

## SPARQL을 이용해서 결과를 추출하는 자바 프로그램

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

public class SPARQLEndpointTest {
    public static void main(String[] args) {
        String sparqlEndpoint = "http://localhost:8890/sparql";
        String sparql = "SELECT * WHERE { ?s rdf:type <http://localhost:8890/test/Movie> . ?s ?p ?o . }";
        try {
            URL naver = new URL(sparqlEndpoint + "?query=" + URLEncoder.encode(sparql) + "&format=JSON");
            HttpURLConnection urlConn = (HttpURLConnection) naver.openConnection();

            BufferedReader br = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
            String inputLine;
            while ((inputLine = br.readLine()) != null) {
                System.out.println(inputLine);
            }
            br.close();
        } catch (Exception e) {
            System.out.println("Exception : " + e.toString());
        }
    }
}
```

# 결과는요

## SPARQL을 이용해서 결과를 추출하는 자바 프로그램의 결과

```
{ "head": { "link": [], "vars": ["s", "p", "o"] } ,  
"results": { "distinct": false, "ordered": true, "bindings": [  
  { "s": { "type": "uri", "value": "http://localhost:8890/test/Movie_1" } ,  
   "p": { "type": "uri", "value": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" } ,  
   "o": { "type": "uri", "value": "http://localhost:8890/test/Movie" } } ,  
  { "s": { "type": "uri", "value": "http://localhost:8890/test/Movie_1" } ,  
   "p": { "type": "uri", "value": "http://www.w3.org/2000/01/rdf-schema#label" } ,  
   "o": { "type": "typed-literal", "datatype": "http://www.w3.org/2001/XMLSchema#string",  
"value": "Diehard" } } ] } }
```

나머지  
활용은  
여러분의  
몫이에요.



# 숙제

1. 못다한 모델링 완성하기
2. 인스턴스 채워 넣기
3. 발행 실습하기
4. SPARQL 더 알아보기
5. 활용방안 찾아보기





**Thanks for  
your attention.**

**Dr. Myungjin Lee**

**e-Mail : [mjlee@li-st.com](mailto:mjlee@li-st.com)**

**Twitter : <http://twitter.com/MyungjinLee>**

**Facebook : <http://www.facebook.com/mjinlee>**

**SlideShare : <http://www.slideshare.net/onlyjiny/>**