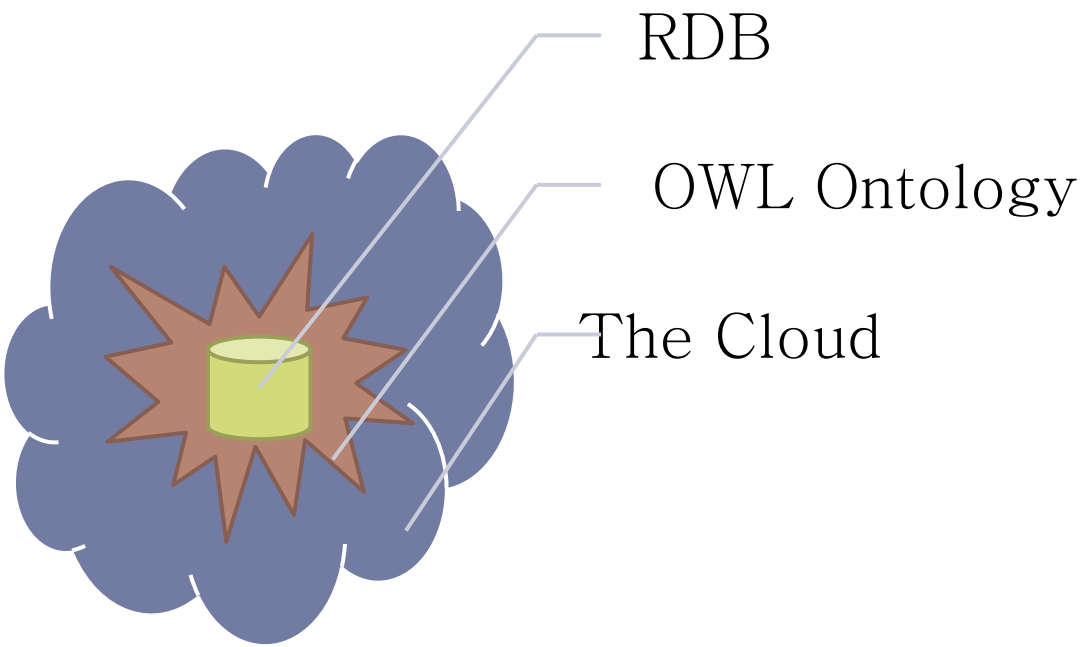
시맨틱 클라우드 컴퓨팅 환경 제공을 위한 ORM(Ontology Relational Mapping) 도구개발

숭실대학교

시스템 개요

- 클라우드 컴퓨팅 환경에서 RDB를 OWL 온톨로지로 포장하여 서비스



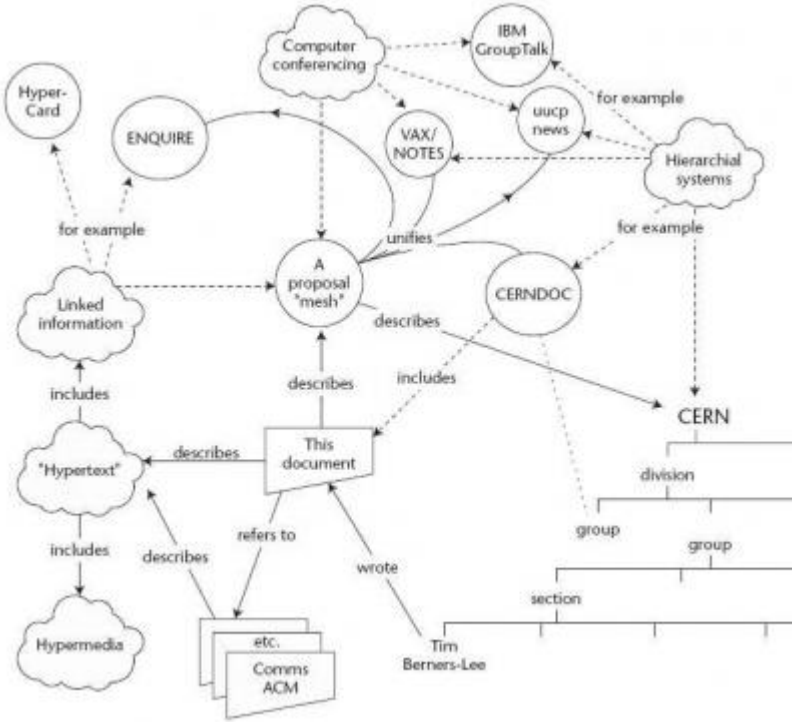
목차

- 과제 동기
- 과제 목적
- 과제 범위
- 시스템 설계 및 구현

과제 동기

시맨틱 웹: 데이터의 웹

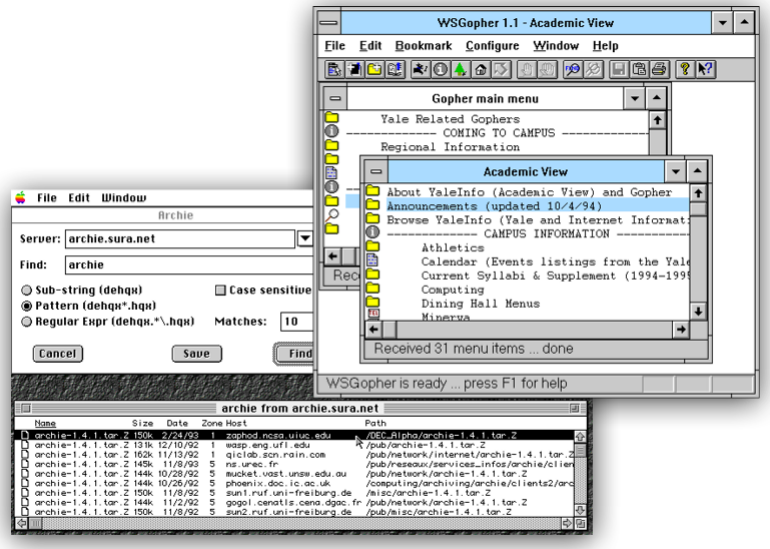
- 웹 스케일의 데이터 통합



By Tim Berners-Lee

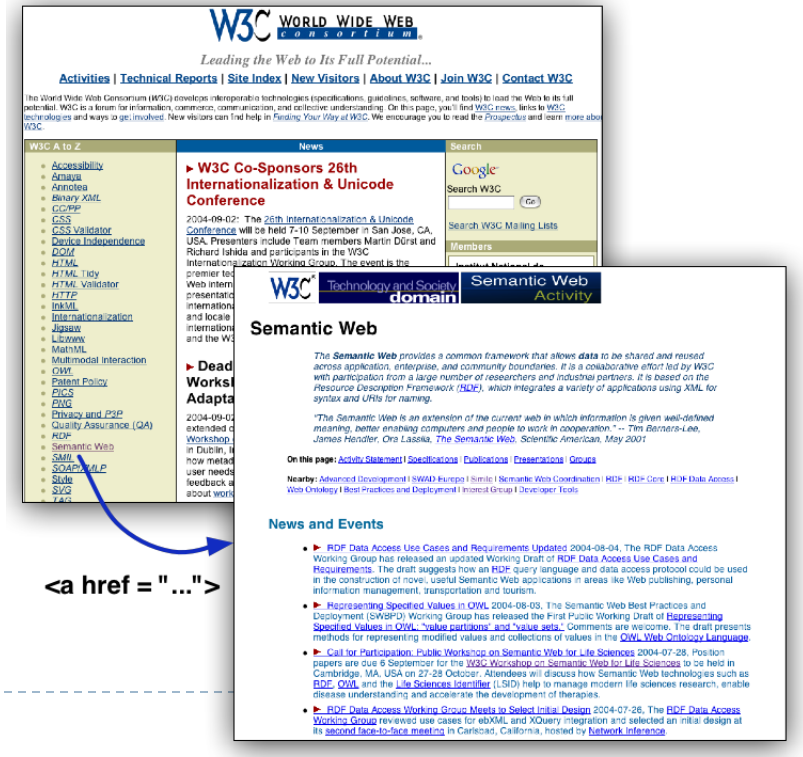
데이터 통합

- 파일 수준
  - 1993 경
  - FTP, Gopher and Archie



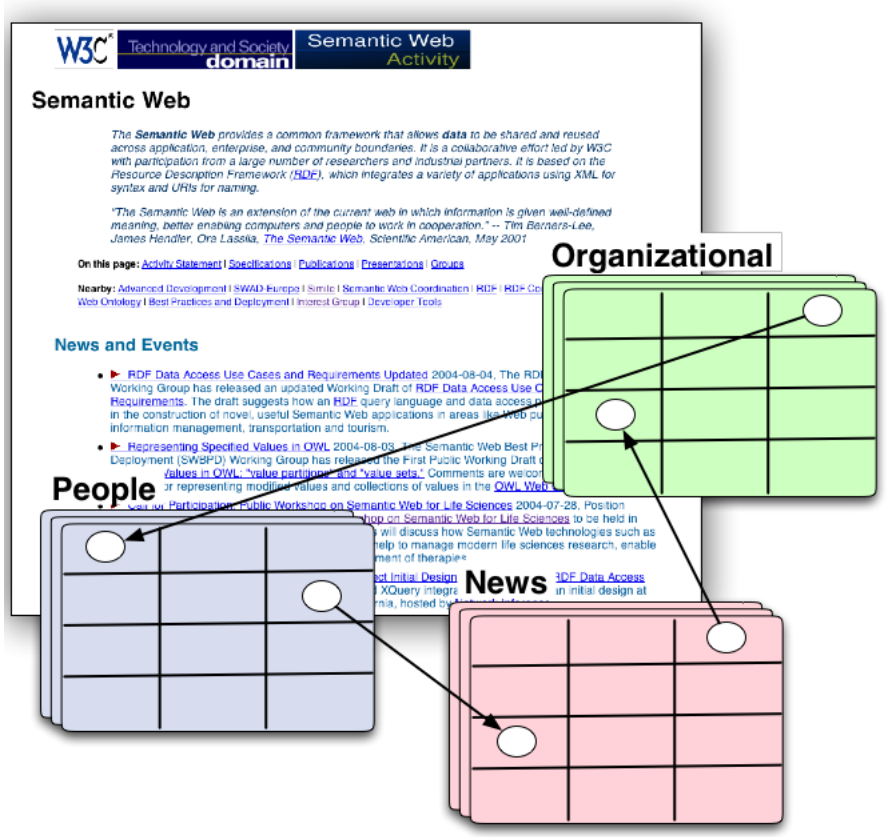
데이터 통합

- 텍스트 수준
- 1994 경
- HTML, URL



데이터 통합

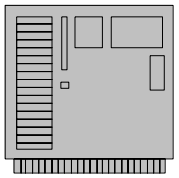
- 데이터 수준
- 현재
- XML, RDF, OWL, URI

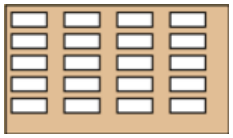


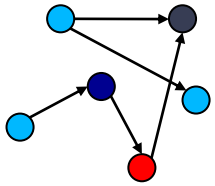
WWW: 다큐먼트의 웹

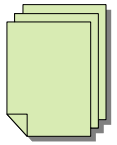
- 데이터의 고립

Applications





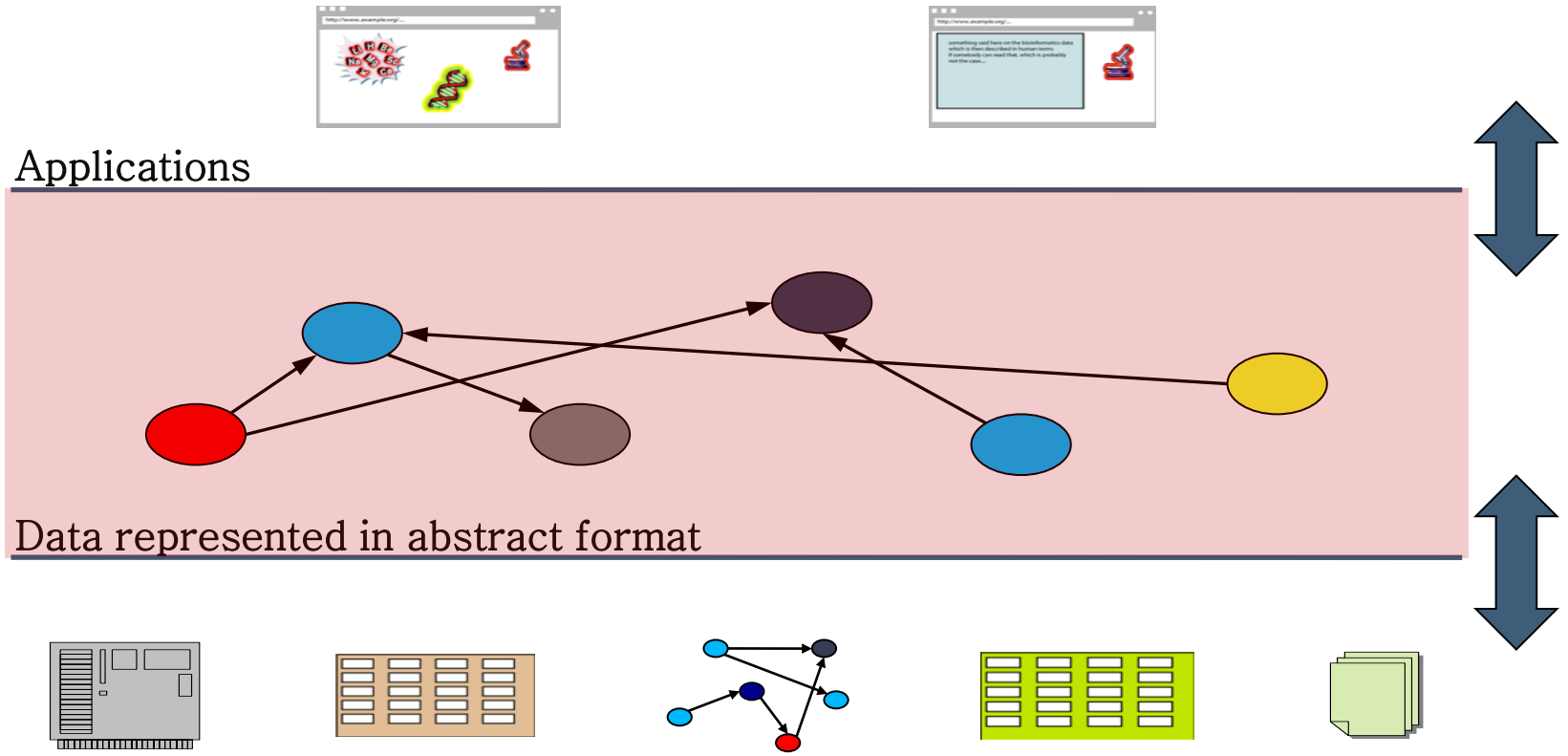




Data in various formats

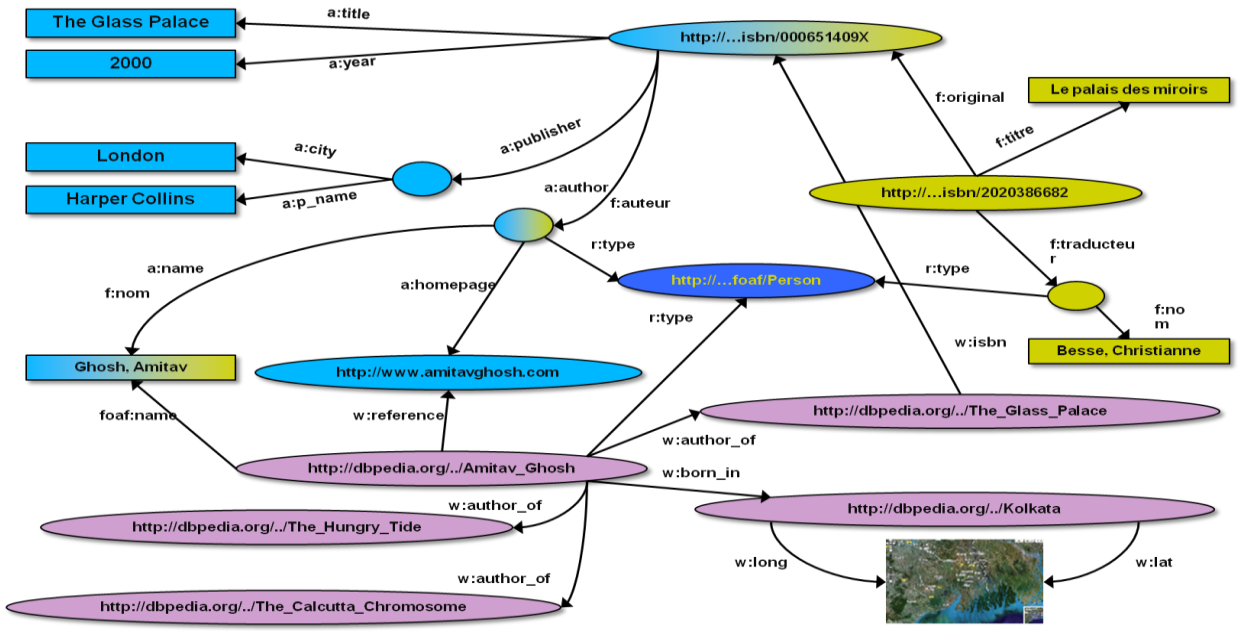
시맨틱 웹: 데이터의 웹

- 현재 웹의 기저에 데이터 통합을 위한 기반을 구축

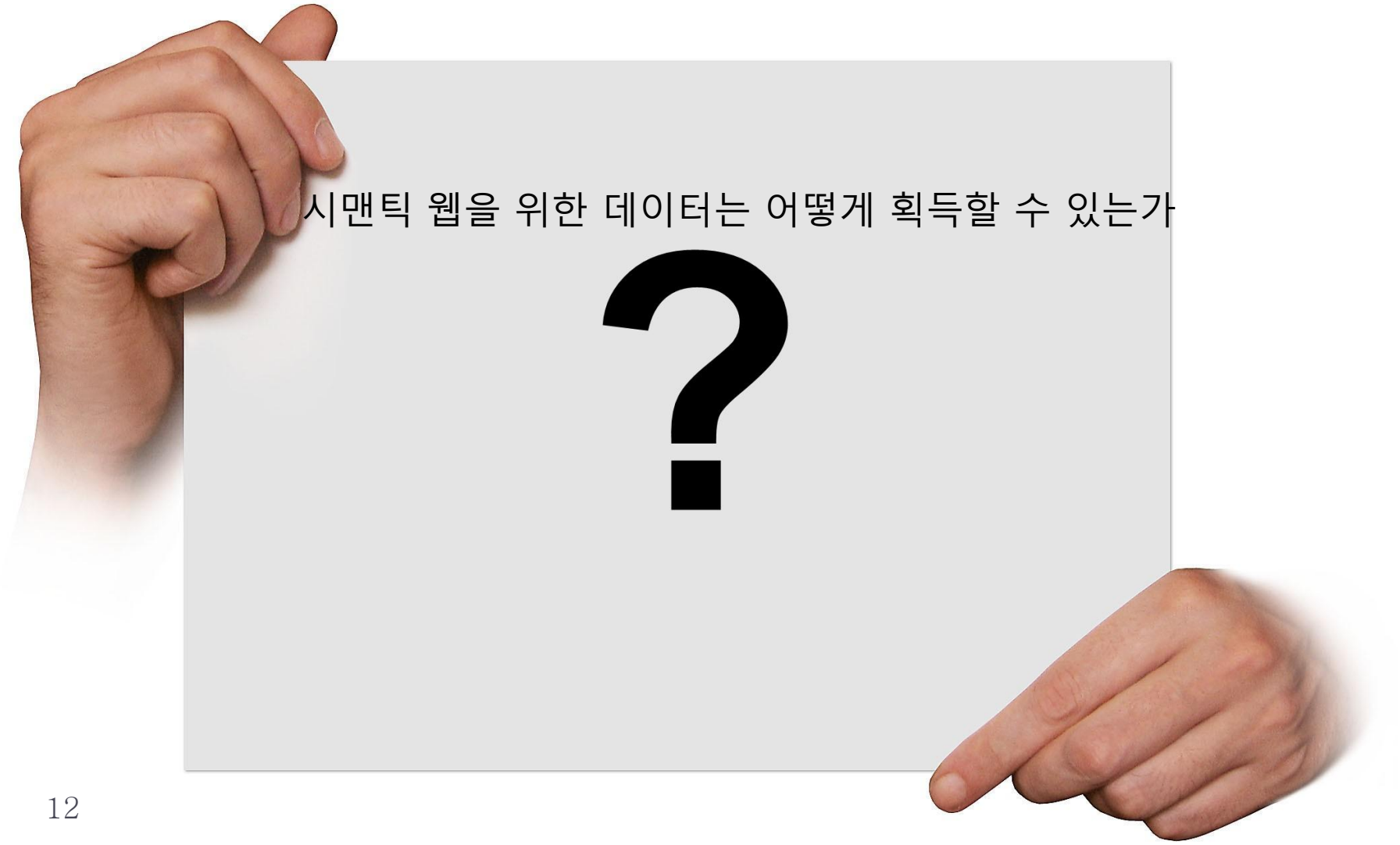


시맨틱 웹을 위한 표준

- URI
  - 데이터 식별 체계
- RDF와 OWL
  - URI 데이터 사이의 관계 기술 언어

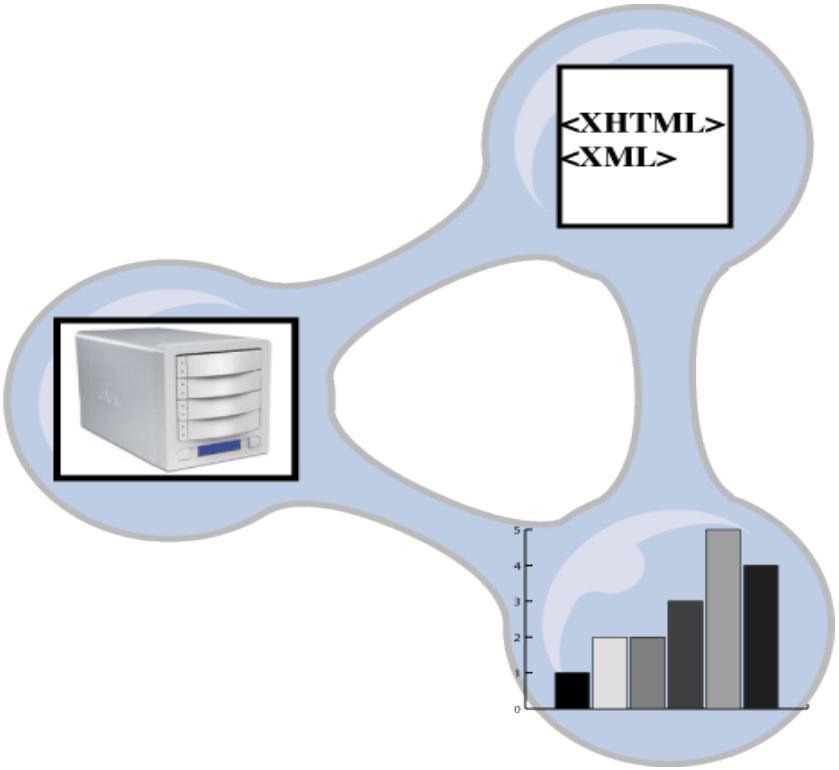


시맨틱 웹을 위한 데이터



시맨틱 웹을 위한 데이터

- 현재 웹을 위한 다양한 포맷의 데이터를 시맨틱 웹의 표준 포맷으로 변환 및 포장



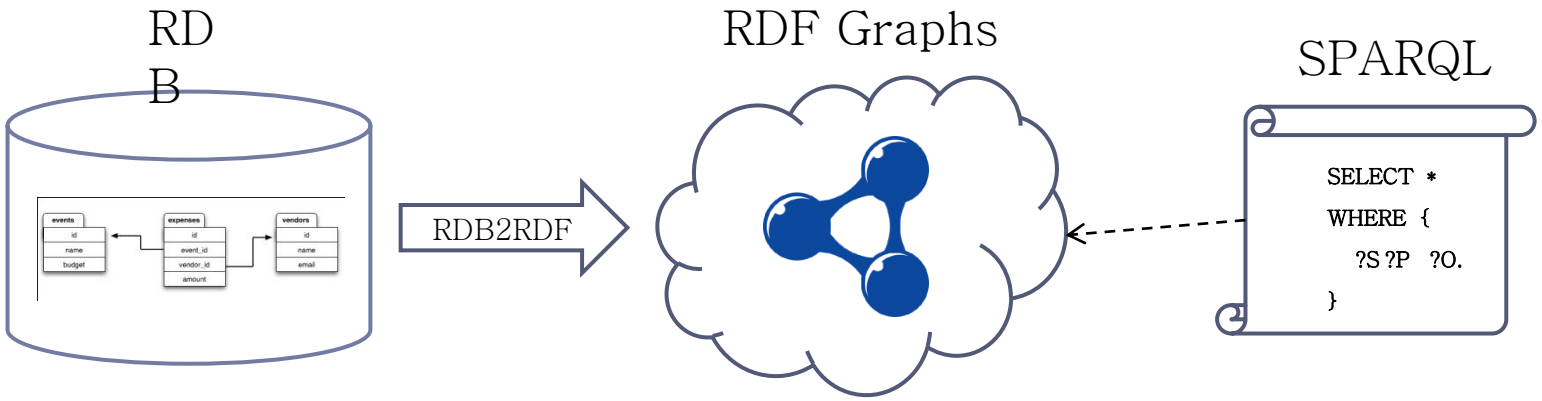
RDB로부터의 온톨로지 생성

- 현재, 동적 웹 컨텐트의 대부분은 RDB로부터 생성

- 시맨틱 웹의 주요 동기는 RDB 데이터를 시맨틱 웹에서 공유하고자 하는 것이다.
  - [Tim Berners-Lee]

- 이 분야의 표준화를 위한 W3C 워킹 그룹이 활동중

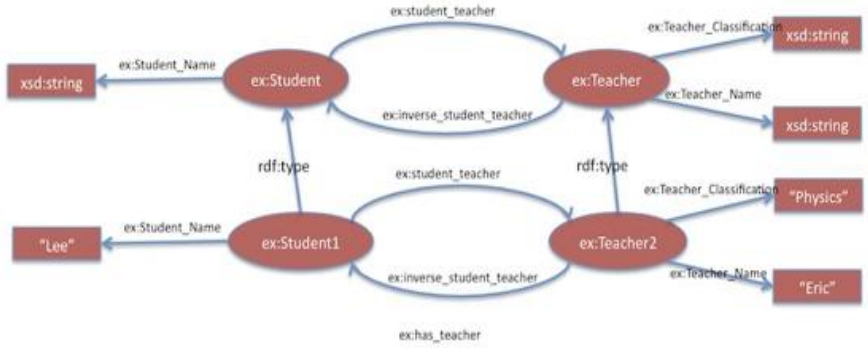
RDB to RDF



StudentID	Name
1	Lee
2	Juan

StudentID	TeacherID
1	2
2	1

TeacherID	Name	Classification
1	Michael	Math
2	Eric	Physics



Virtual RDF Graphs

- 질의 시점에 결과 데이터 셋을 RDF 포맷으로 제공
  - 시맨틱 웹을 위하여 모든 RDB 데이터를 물리적으로 RDF 로 변환하는 것은 비현실적

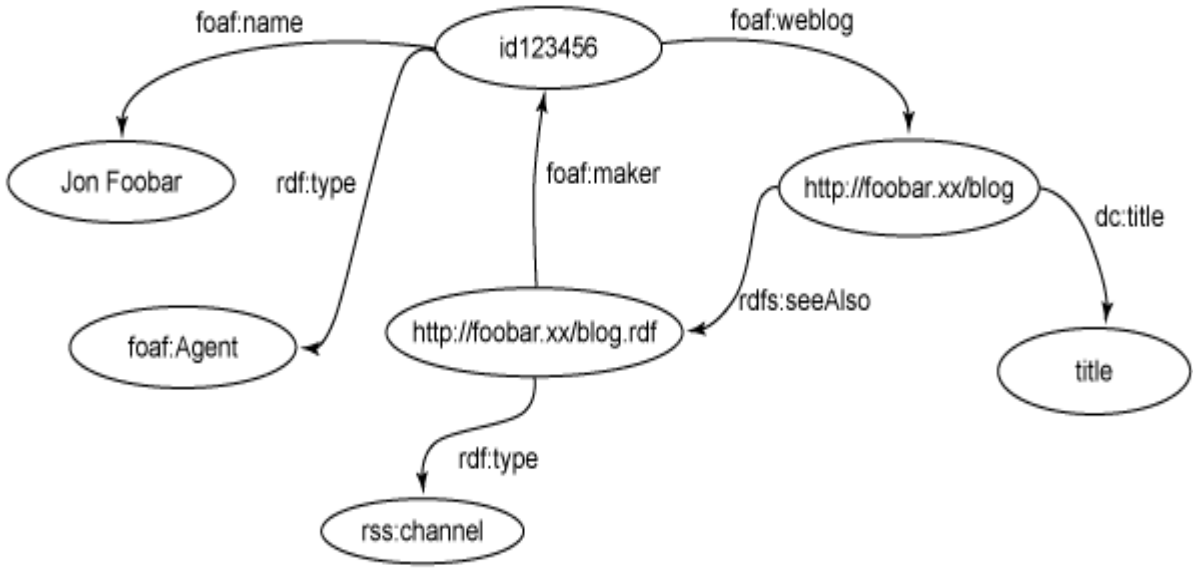
- “RDF-Based Ontology View”라고도 함

RDF의 한계

- RDF는 그래프다!
  - directed, labeled graph

- SPARQL
  - RDF를 위한 표준 질의 언어
  - 그래프 패턴 매칭에 기반

SPARQL 예제

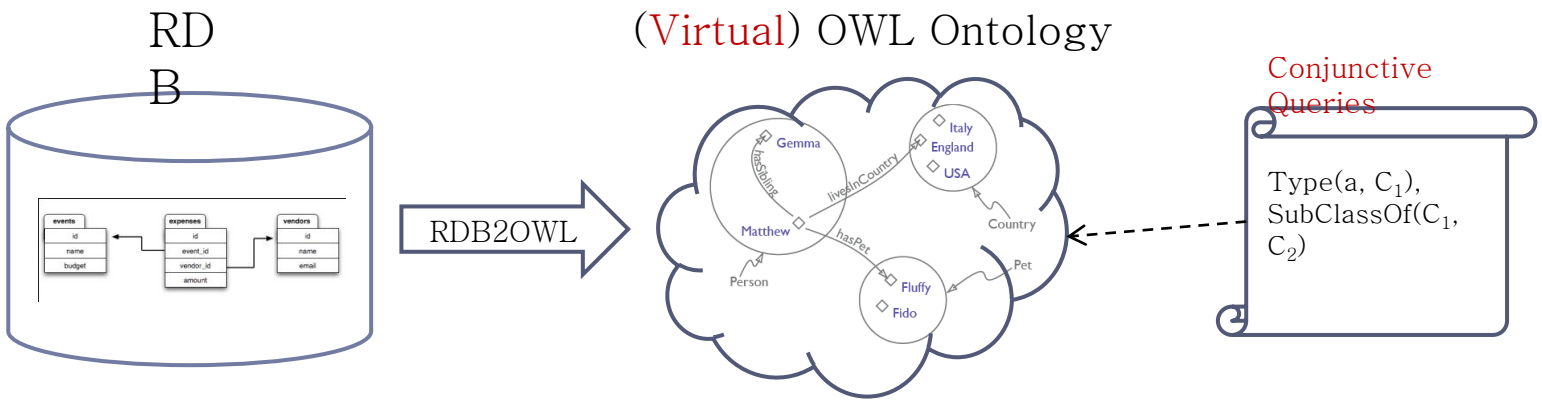



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT          ?url
FROM   <bloggers.rdf>
WHERE  {
   ?contributor   foaf:name        "Jon Foobar" .
   ?contributor   foaf:weblog      ?url .
}
```

시맨틱 웹 애플리케이션의 요구 사항

- 프로퍼티의 특성
  - transitive, inverse or symmetric
- 서로 다른 URI의 오브젝트의 식별
  - sameAs or differentFrom
- 클래스간의 관계
  - disjointness or equivalence
- 익명 클래스 정의
  - intersection, union and complement
- 정교한 클래스 계층구조
  - inferred class hierarchy
- 추론 연산
  - “if «Person» resources «A» and «B» have the same «foaf:email» property, then «A» and «B» are identical”

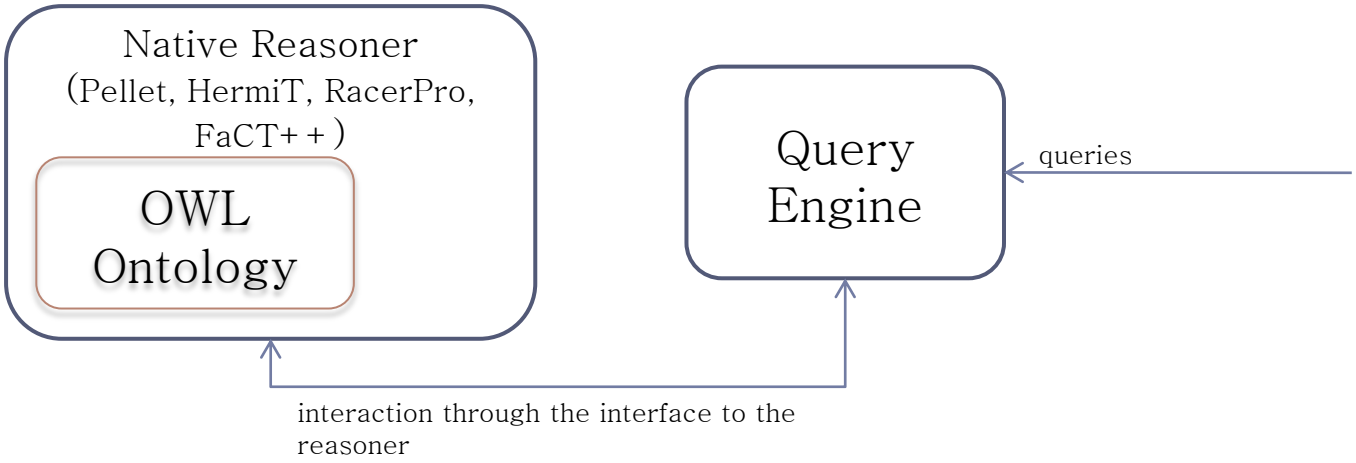
RDB to OWL



- 존재하지 않음

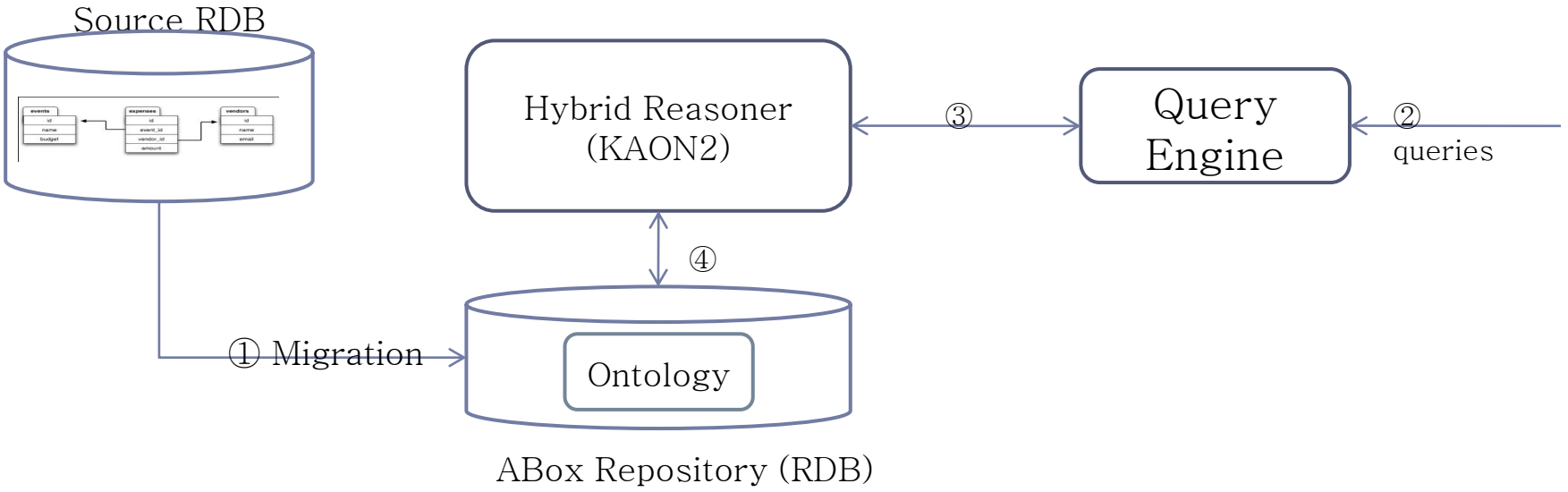
추론기의 문제점 #1

- 대용량 Abox를 지원하는 네이티브 추론기가 존재하지 않음.
  - 표준 tableau 알고리즘에 기반한 in-memory reasoning



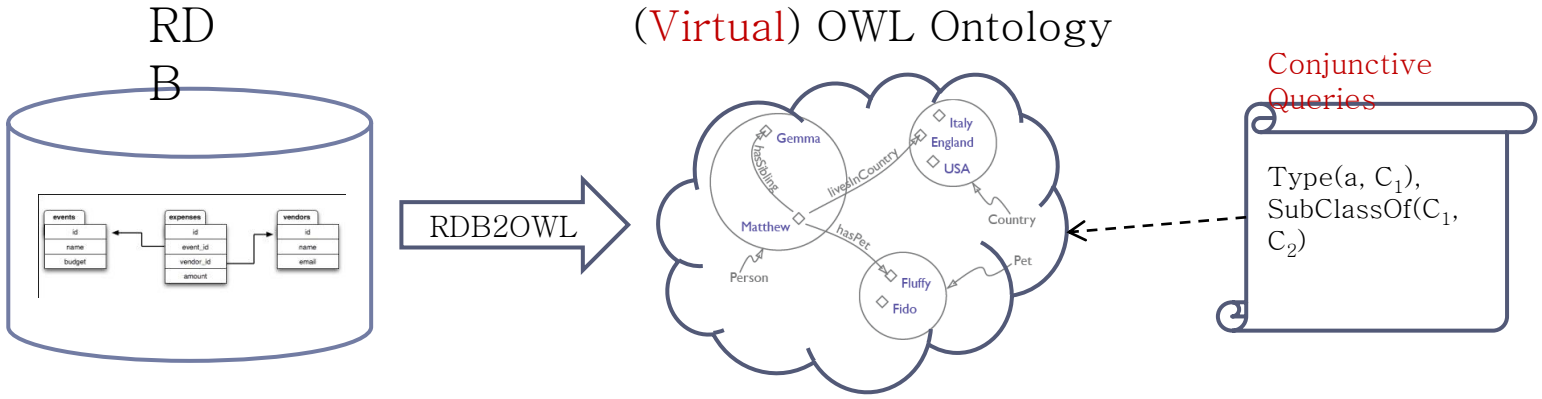
추론기의 문제점 #2

- 하이브리드 추론기는 대용량 Abox를 처리할 수 있음
  - on-disk reasoning
- 그러나, Abox 저장소는 고유의 스키마를 가지고 있음
- 따라서, 데이터 마이그레이션이 요구됨



과제 목적

과제 목적



- RDB에 변경을 가하지 않고 질의 시점에 추론된 OWL 온톨로지로부터 얻을 수 있는 데이터 셋을 제공할 수 있는 시스템의 개발

과제 범위

과제 범위 #1

- RDB의 데이터와 스키마가 내포하는 의미에 대응하는 OWL 온톨로지 생성을 위한 매핑 규칙
  - RDB 스키마
    - 엔티티: OWL 클래스와 프로퍼티로 매핑
    - 무결성 제약조건: OWL 공리로 매핑
  - RDB 데이터
    - OWL 개체 및 리터럴로 매핑
  - 추론
    - SQL 조인 연산에 의해서 얻어질 수 있는 데이터간의 모든 관계를 추론을 통해 획득할 수 있도록 설계

온톨로지의 구조

TBox (terminological box)	ABox (assertional box)
Mother = Women ∩ ∃hasChild.Person Parent = Mother ∪Father Grandmother = Mother ∩ ∃hasChild.Mother	Father(Peter) Grandmother(Mary) hasChild(Mary, Peter) hasChild(Mary, Paul) hasChild(Peter, Harry)
class axioms	instance assertions
classification	instance checking

- RDB 스키마는 TBox로 매핑
- RDB 데이터는 ABox로 매핑

과제 범위 #2

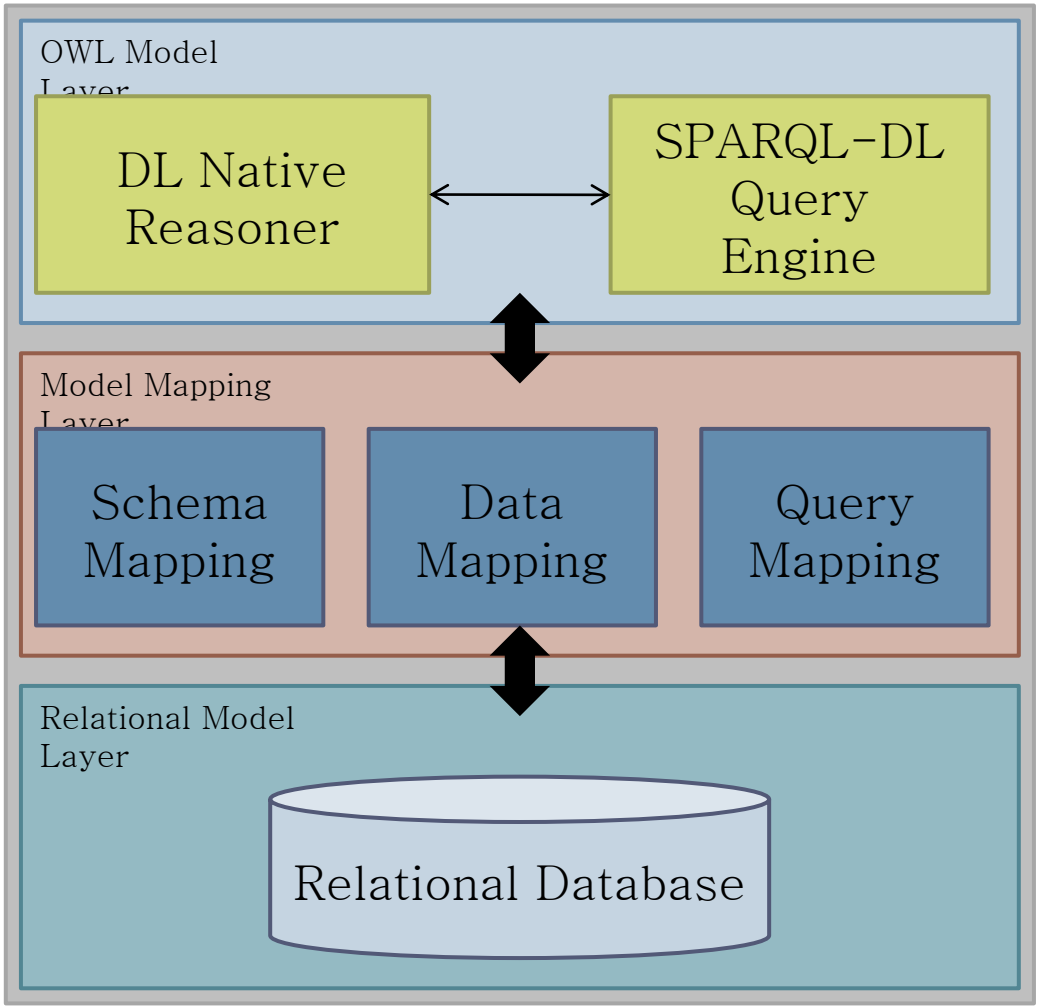
- OWL 온톨로지를 위한 질의 언어(SPARQL-DL)의 처리
  - SPARQL-DL 질의 엔진
    - ABox 질의 처리를 위해 인스턴스 데이터를 추론기에 로드하지 않는 처리 방식
      - 결과 질의 셋을 얻기 위한 SQL 질의문(들) 생성

과제 범위 #3

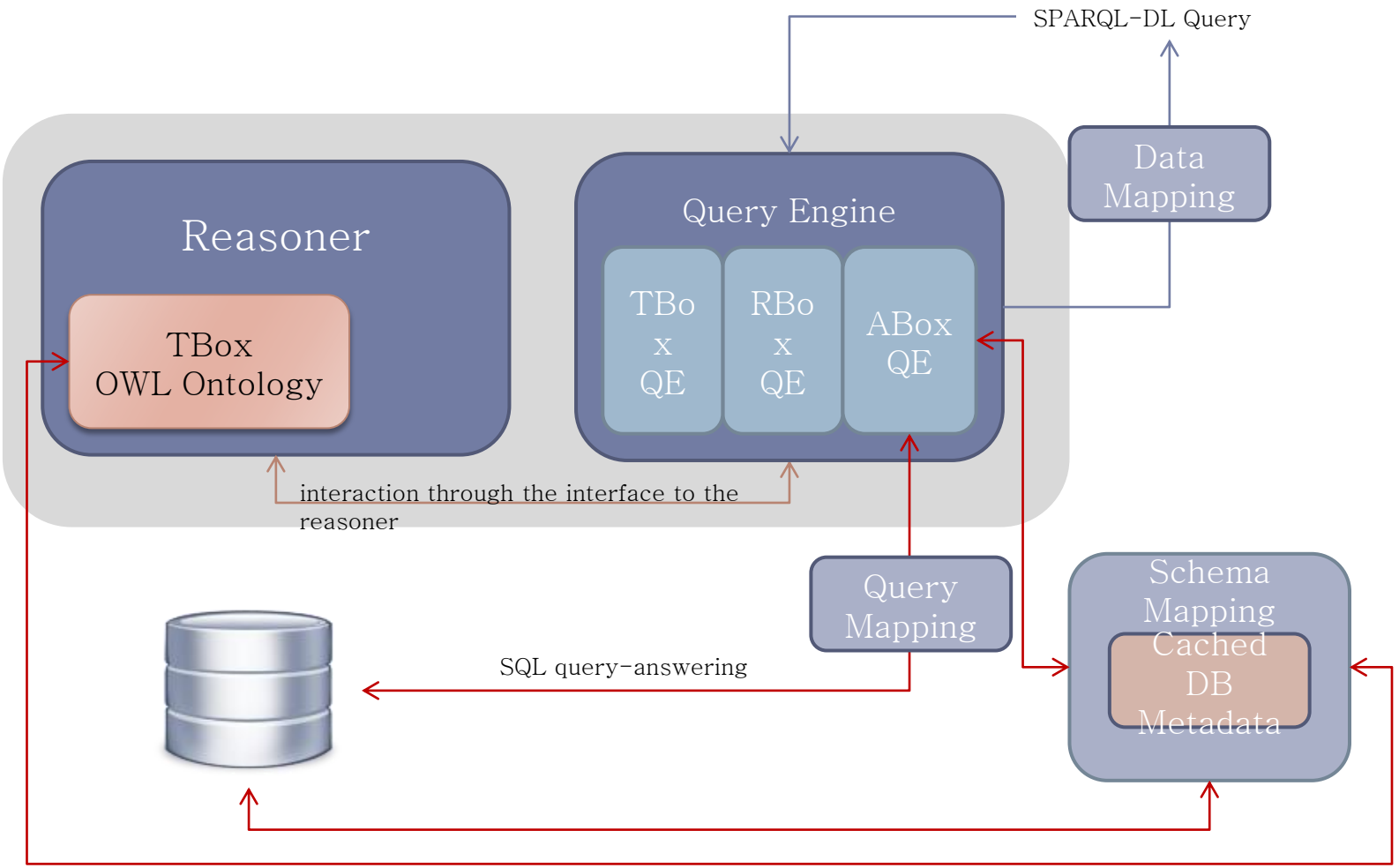
- SPARQL-DL 질의 도구
  - 클라이언트(인간)이 GUI 환경에서 직관적으로 SPARQL-DL 질의-응답 과정을 수행할 수 있는 환경을 제공
- REST 인터페이스 API
  - 클라이언트(애플리케이션)이 REST 인터페이스를 통해 클라우드 환경 내의 RDB를 OWL 온톨로지로서 간주하고 서비스 받을 수 있도록 하는 API 제공

시스템 설계 및 구현

시스템 구조도


OWL Model
Layer
DL Native
Reasoner
SPARQL-DL
Query
Engine
Model Mapping
Layer
Schema
Mapping
Data
Mapping
Query
Mapping
Relational Model
Layer
Relational Database

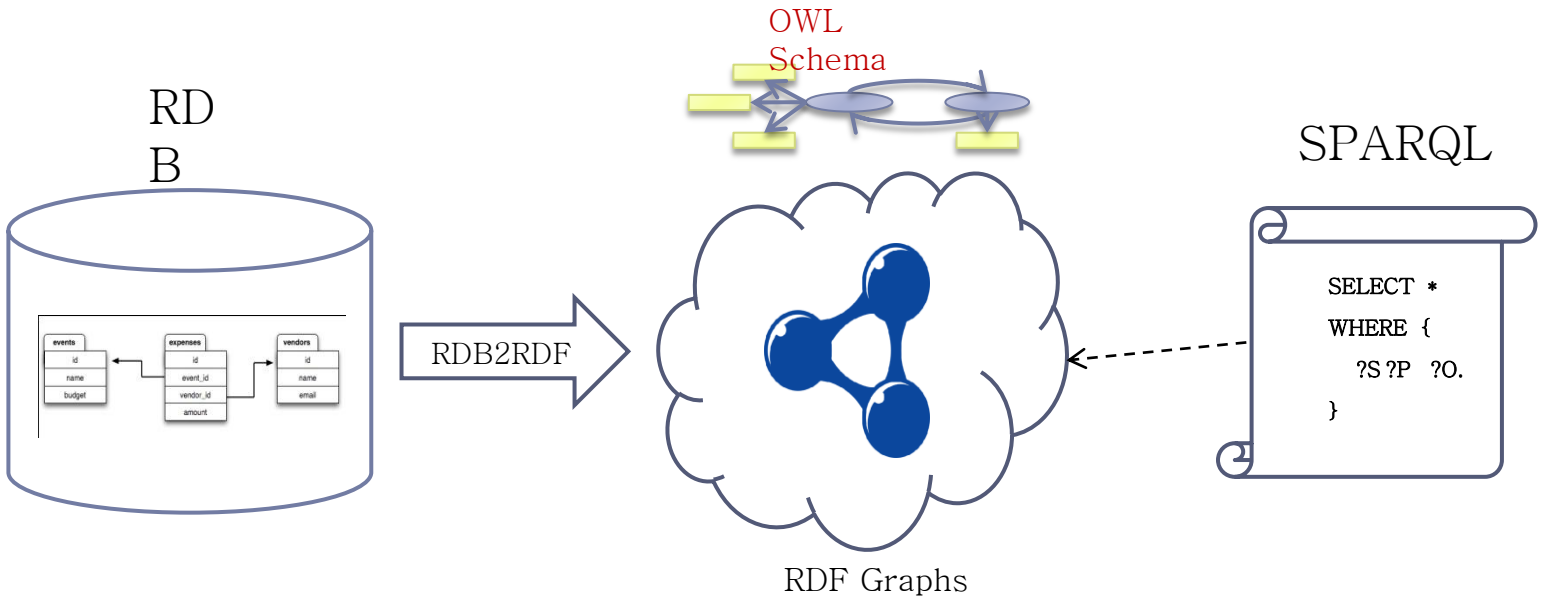
시스템 동작 시나리오


SPARQL-DL Query
Data Mapping
Reasoner
TBox
OWL Ontology
Query Engine
TBo x QE
RBo x QE
ABox QE
interaction through the interface to the reasoner
Query Mapping
Schema Mapping
Cached DB Metadata
SQL query-answering

매핑 알고리즘

기존 RDB2OWL 매핑 규칙

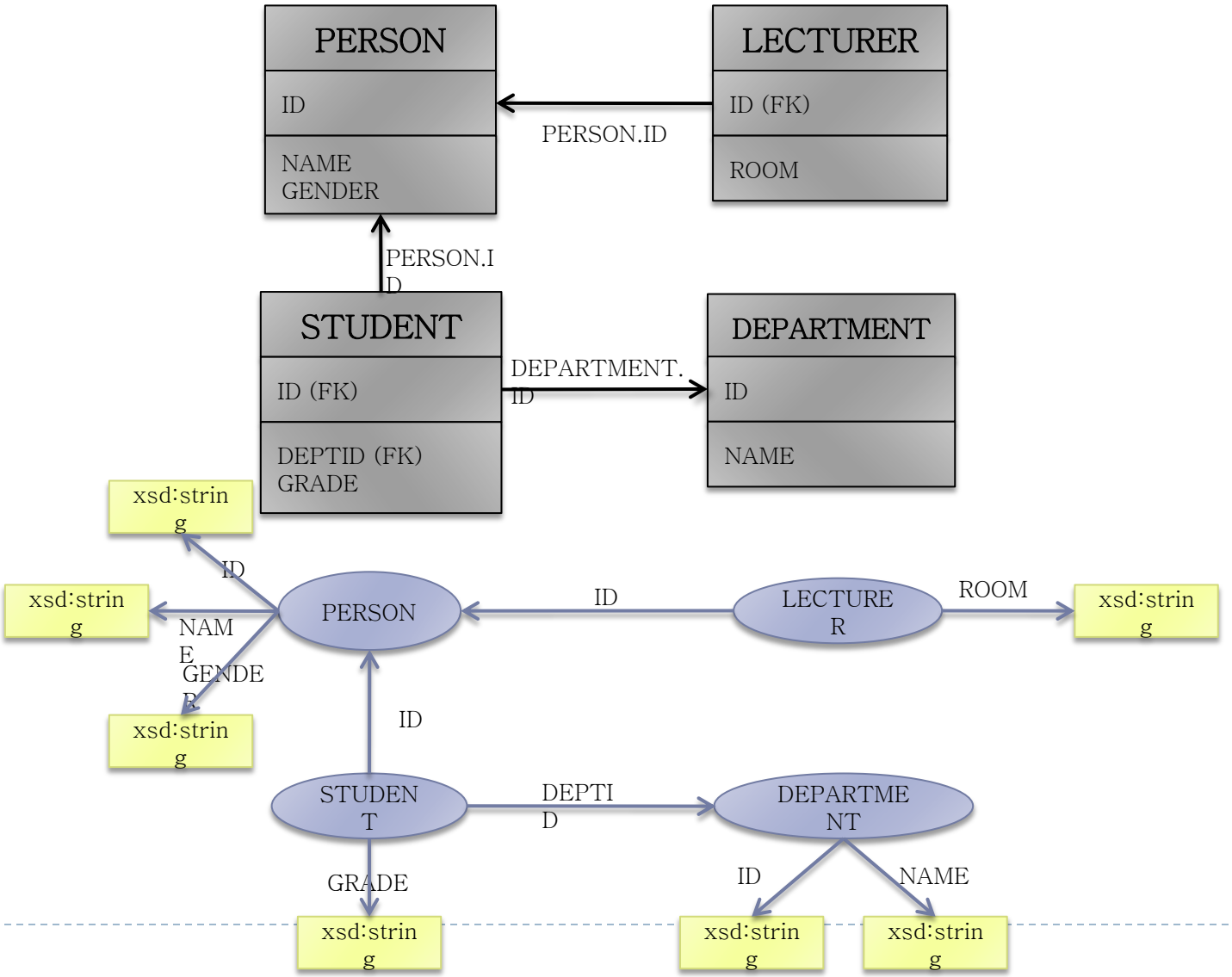
- RDF 그래프에 대한 OWL 형식의 스키마 제공에 적합
  - RDF 그래프를 위한 메타데이터 역할
  - RDF 모델 지향적인 OWL 변환



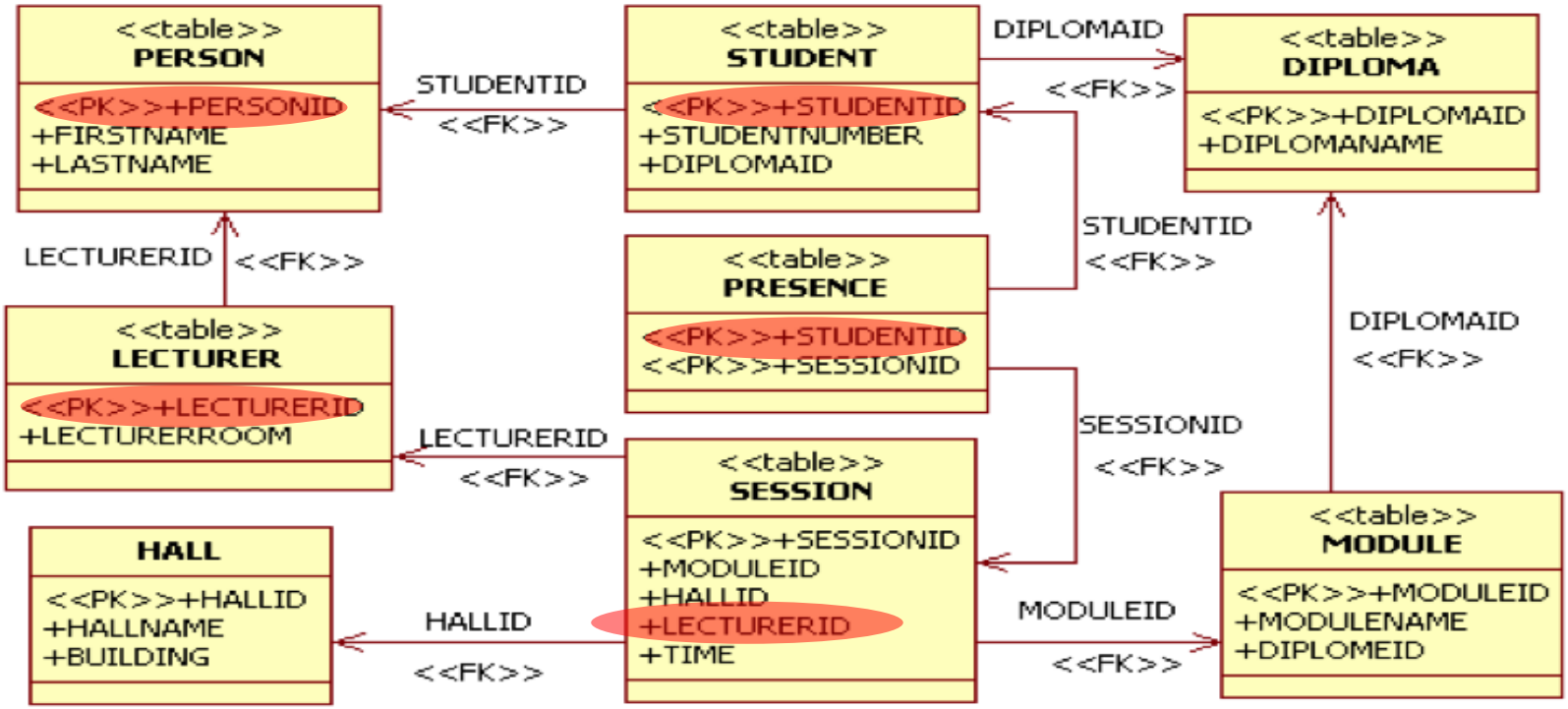
기존 RDB2OWL 매핑의 문제점

- 변환 불가능한 스키마 존재
- OWL 익명 개체(blank node)의 사용
- 서로 다른 URI 오브젝트의 식별을 위한 공리 부재

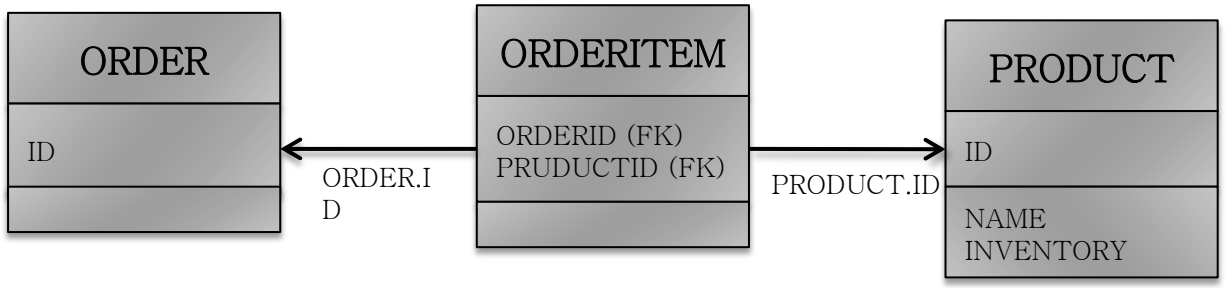
기존 RDB2OWL 매핑

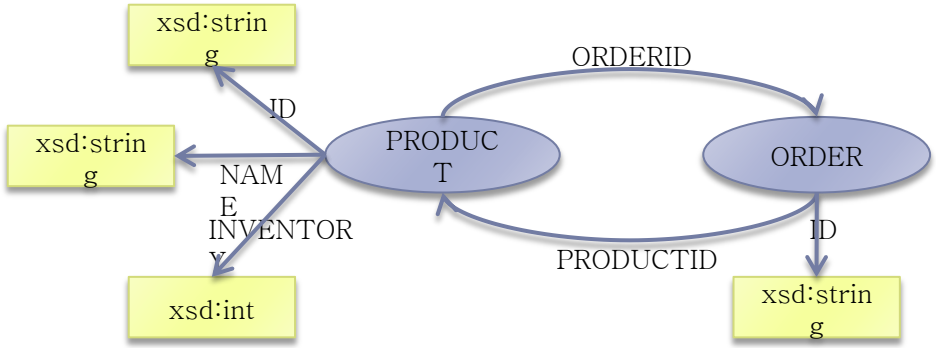


실제 RDB


<<table>>
PERSON
<<PK>>+PERSONID
+FIRSTNAME
+LASTNAME
STUDENTID
<<FK>>
<<table>>
STUDENT
<<PK>>+STUDENTID
+STUDENTNUMBER
+DIPLOMAID
DIPLOMAID
<<FK>>
<<table>>
DIPLOMA
<<PK>>+DIPLOMAID
+DIPLOMANAME
LECTURERID
<<FK>>
<<table>>
LECTURER
<<PK>>+LECTURERID
+LECTURERROOM
<<table>>
PRESENCE
<<PK>>+STUDENTID
<<PK>>+SESSIONID
STUDENTID
<<FK>>
DIPLOMAID
<<FK>>
LECTURERID
<<FK>>
<<table>>
SESSION
<<PK>>+SESSIONID
+MODULEID
+HALLID
+LECTURERID
+TIME
SESSIONID
<<FK>>
<<table>>
MODULE
<<PK>>+MODULEID
+MODULENAME
+DIPLOMEID
HALL
<<PK>>+HALLID
+HALLNAME
+BUILDING
HALLID
<<FK>>
MODULEID
<<FK>>

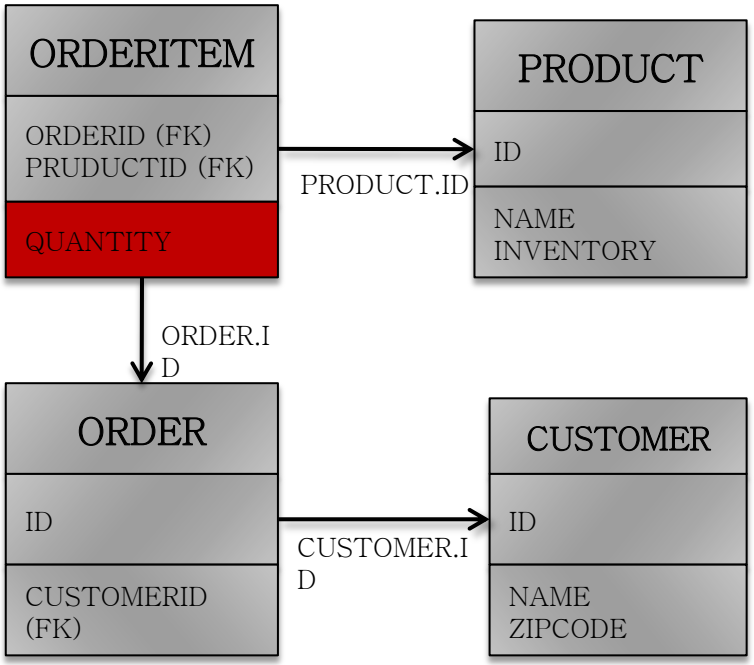
기존 매핑 방법에서의 링크 테이블 매핑





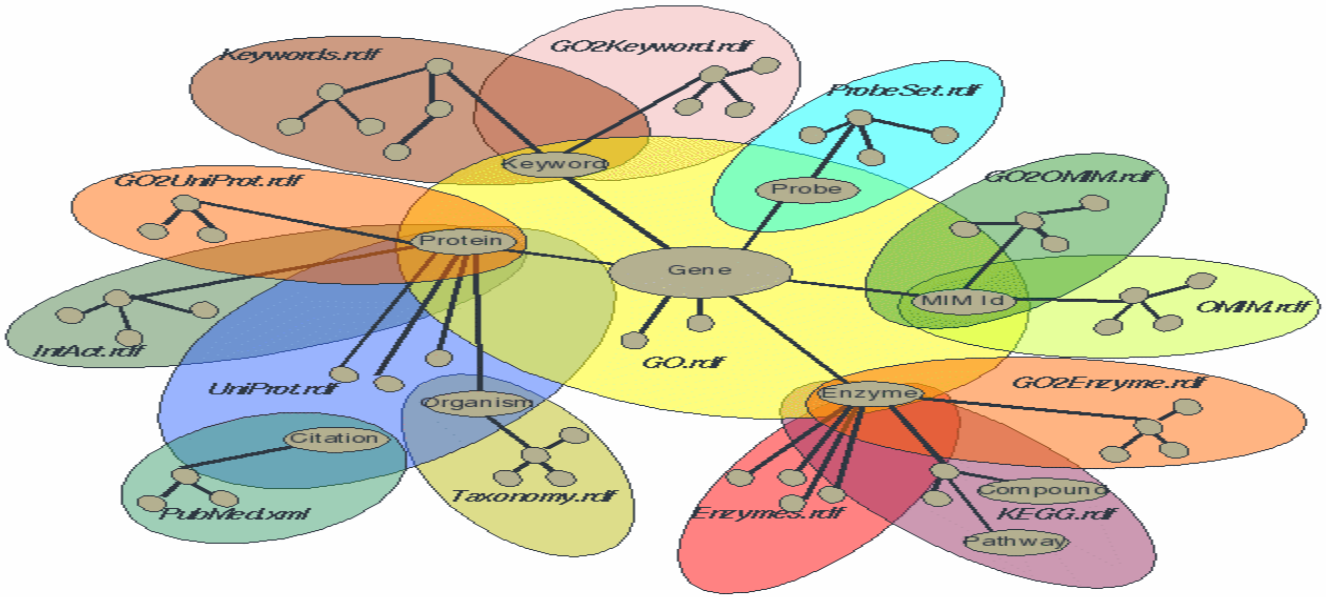
링크 테이블 매핑의 제약

- 링크 테이블의 추가적인 컬럼은 매핑에서 누락



테이블 레코드의 OWL 익명 개체로의 매핑

- OWL 익명 개체는 내부적으로 로컬 온톨로지 안에서 유일한 노드 ID를 부여받음.
- 서로 다른 온톨로지에서 동일한 노드 ID의 노드는 동일하게 식별됨.
  - 온톨로지 merging시 잘못된 고리 역할

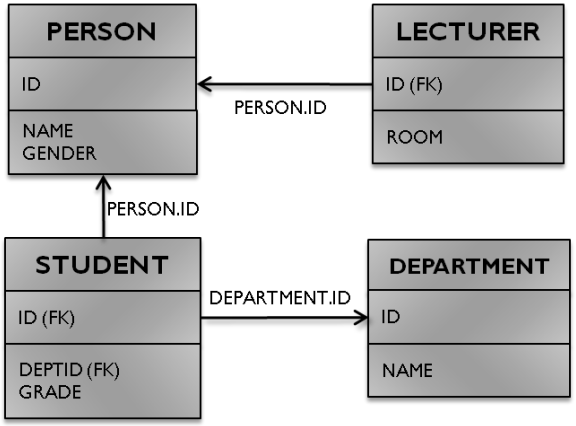


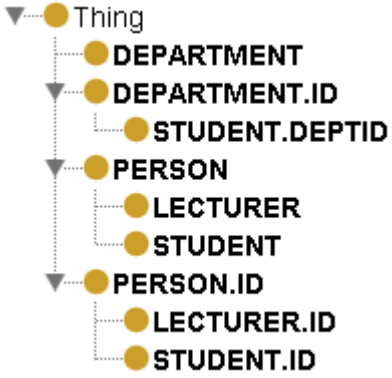
서로 다른 URI 오브젝트의 식별

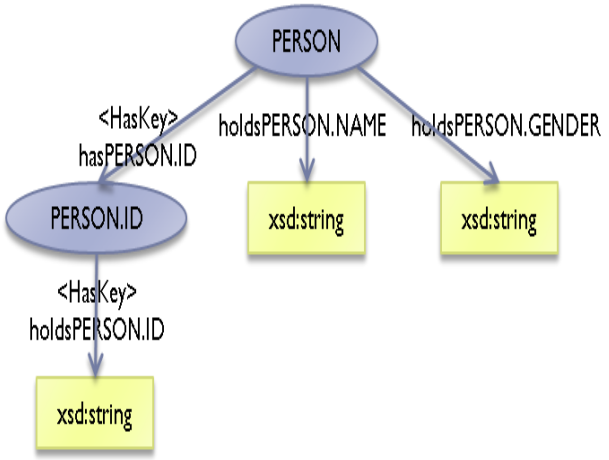
- 서로 다른 URI의 오브젝트가 동일한 트리플 셋을 가지고 있다고 하더라도 OWL은 두 오브젝트를 동일하다라고 하지 않는다.
- 서로 다른 Root의 클래스 계층 구조에 속한 두 클래스를 서로소(disjoint)로 판단하지 않는다.
  - OWL은 Open-World Assumption 기반이기 때문

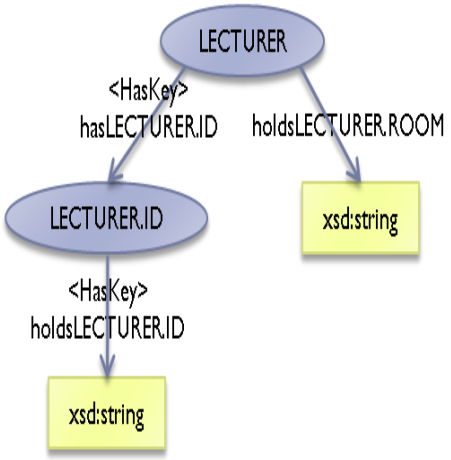
- 오브젝트 식별을 위한 공리
  - Disjoint 공리
  - HasKey 공리

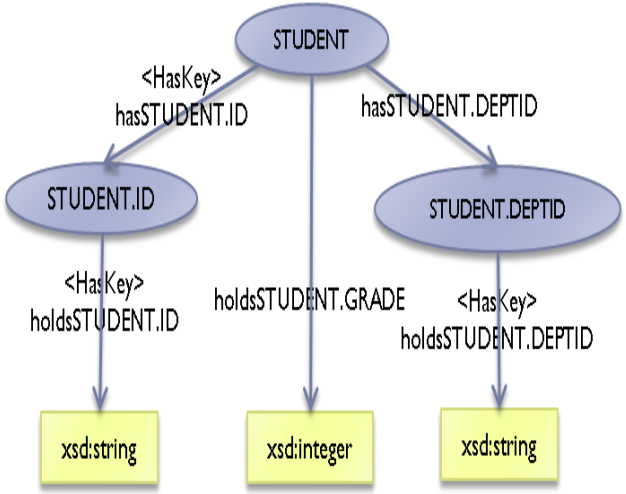
본 과제에서의 매핑

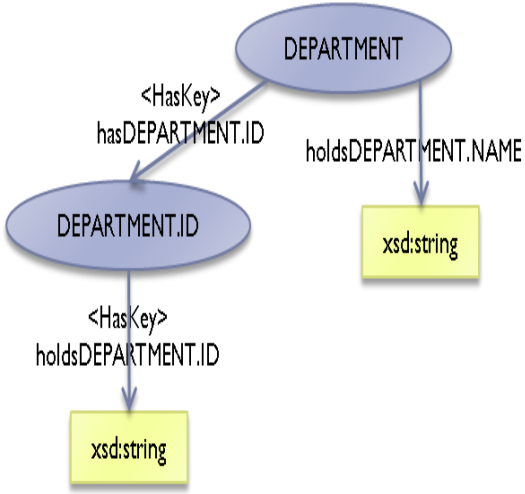

PERSON
ID
NAME
GENDER
LECTURER
ID (FK)
ROOM
PERSON.ID
PERSON.ID
STUDENT
ID (FK)
DEPTID (FK)
GRADE
DEPARTMENT.ID
DEPARTMENT
ID
NAME


Thing
DEPARTMENT
DEPARTMENT.ID
STUDENT.DEPTID
PERSON
LECTURER
STUDENT
PERSON.ID
LECTURER.ID
STUDENT.ID

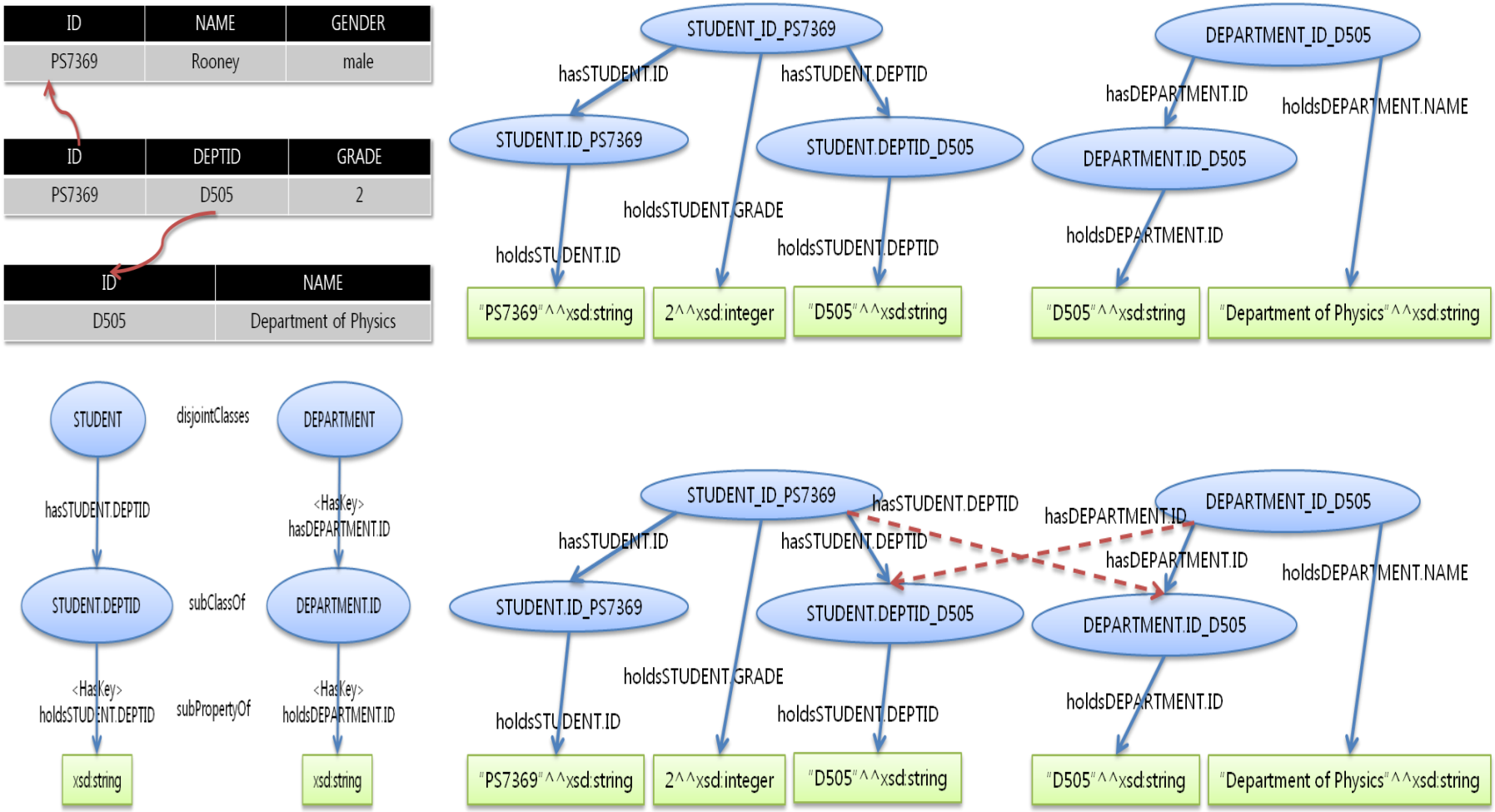

PERSON
<HasKey>
hasPERSON.ID
holdsPERSON.NAME
holdsPERSON.GENDER
PERSON.ID
xsd:string
xsd:string
<HasKey>
holdsPERSON.ID
xsd:string


LECTURER
<HasKey>
hasLECTURER.ID
holdsLECTURER.ROOM
LECTURER.ID
xsd:string
<HasKey>
holdsLECTURER.ID
xsd:string


STUDENT
<HasKey>
hasSTUDENT.ID
hasSTUDENT.DEPTID
STUDENT.ID
STUDENT.DEPTID
<HasKey>
holdsSTUDENT.ID
holdsSTUDENT.GRADE
<HasKey>
holdsSTUDENT.DEPTID
xsd:string
xsd:integer
xsd:string


DEPARTMENT
<HasKey>
hasDEPARTMENT.ID
holdsDEPARTMENT.NAME
DEPARTMENT.ID
xsd:string
<HasKey>
holdsDEPARTMENT.ID
xsd:string

추론된 온톨로지는 그래프를 동적으로 연결

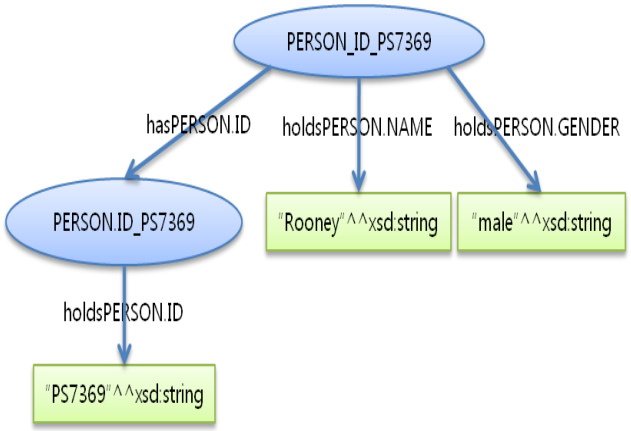


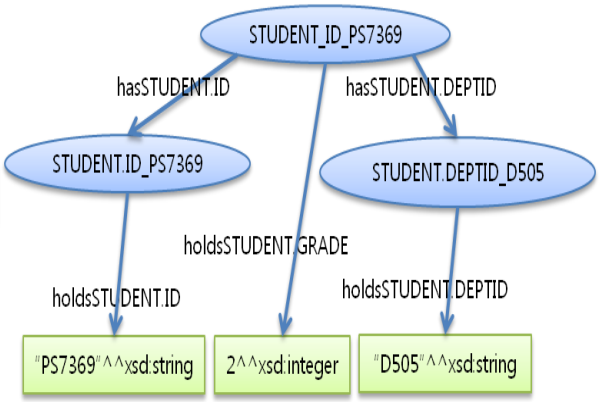
추론된 온톨로지는 그래프를 동적으로 연결

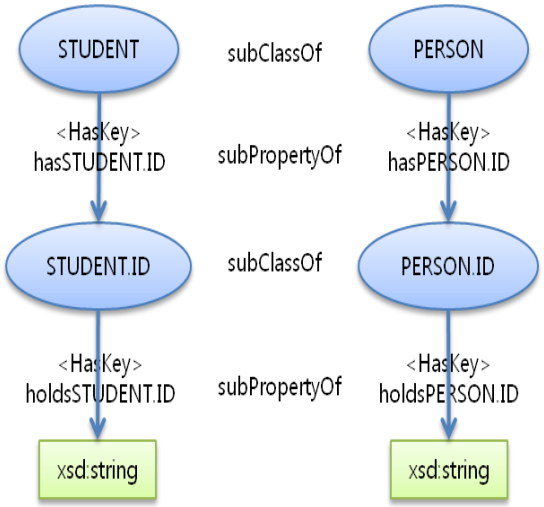
ID	NAME	GENDER
PS7369	Rooney	male

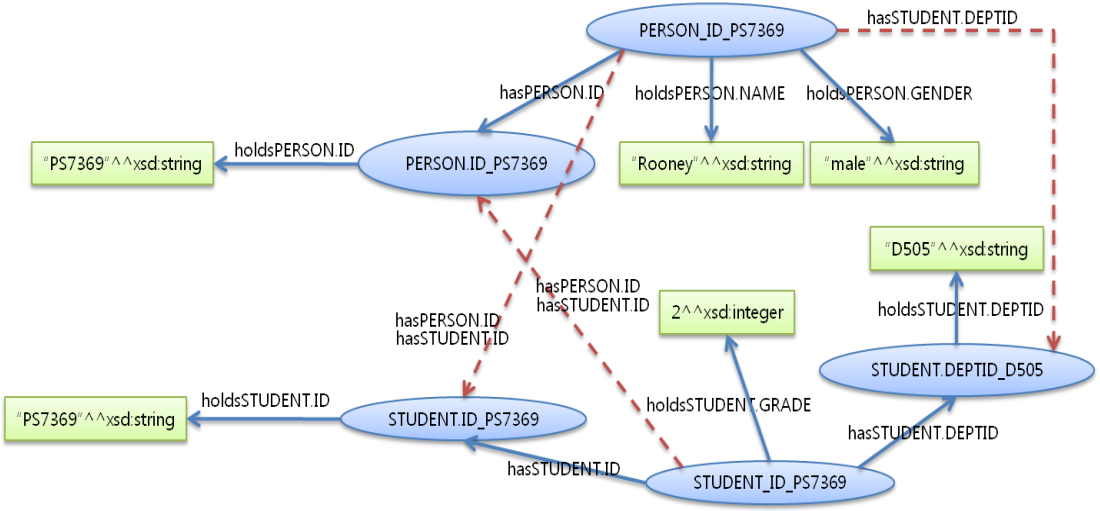
ID	DEPTID	GRADE
PS7369	D505	2

ID	NAME
D505	Department of Physics

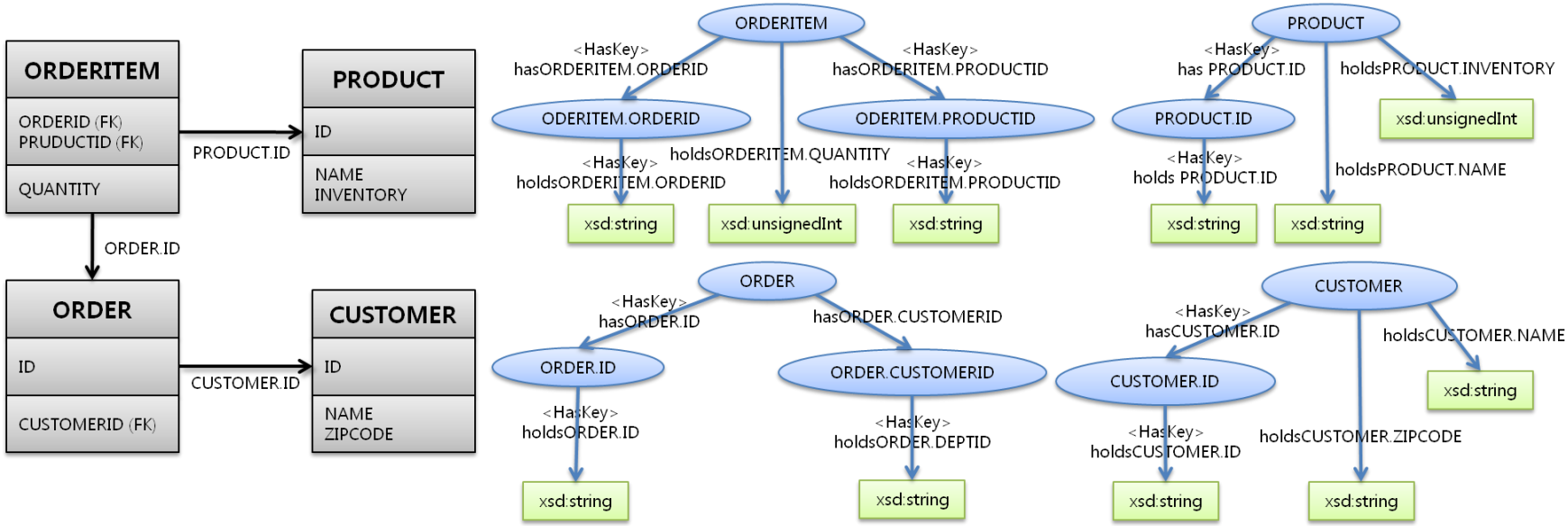






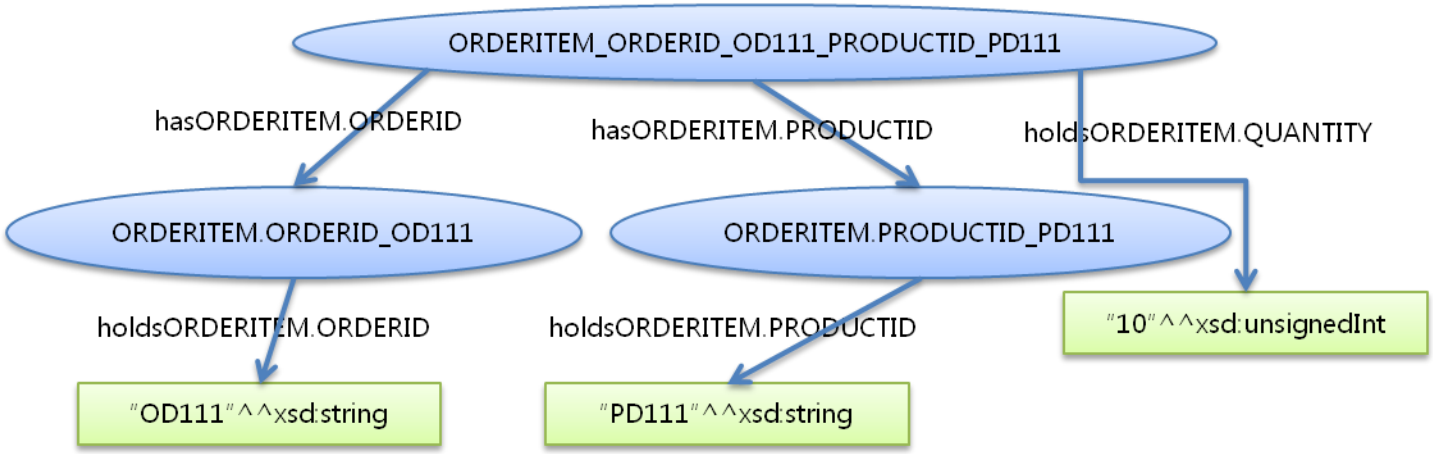


추가적인 컬럼을 갖는 링크 테이블의 매핑



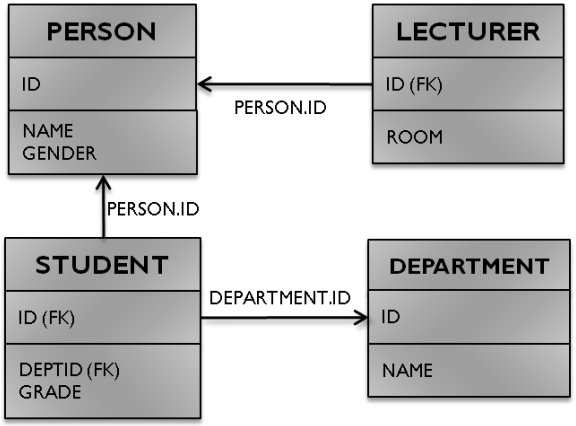
테이블의 행을 명명된 OWL 개체로 매핑

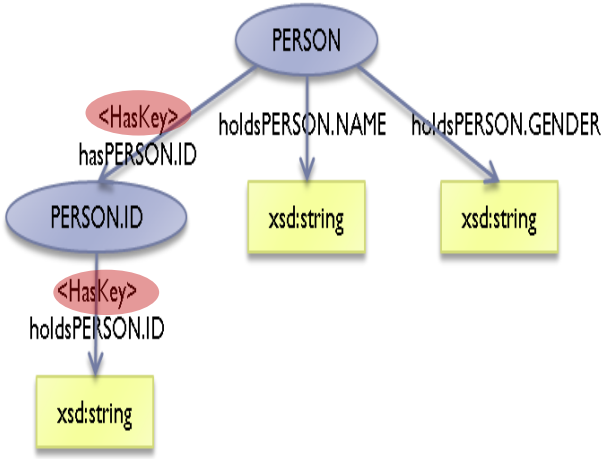
ORDERID	PRODUCTID	QUANTITY
OD111	PD111	10

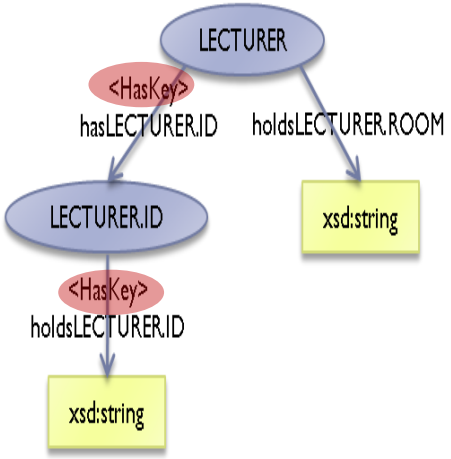


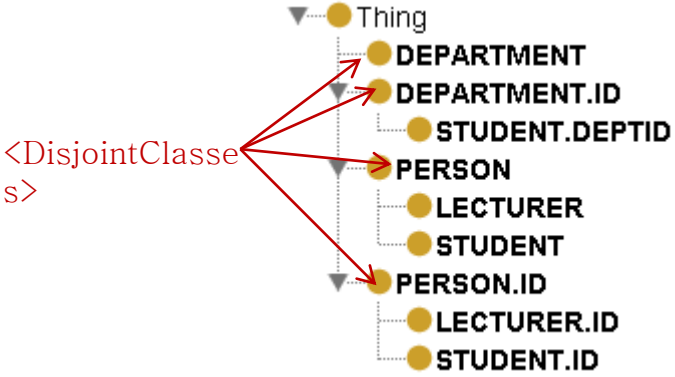
ex)
http://cosmos.ssu.ac.kr/ontologies/rdb.owl#ORDERITEM.ORDERID_OD111

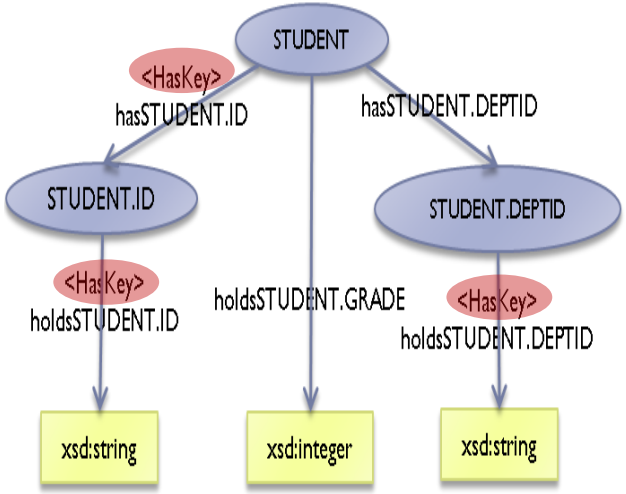
오브젝트 식별을 위한 공리

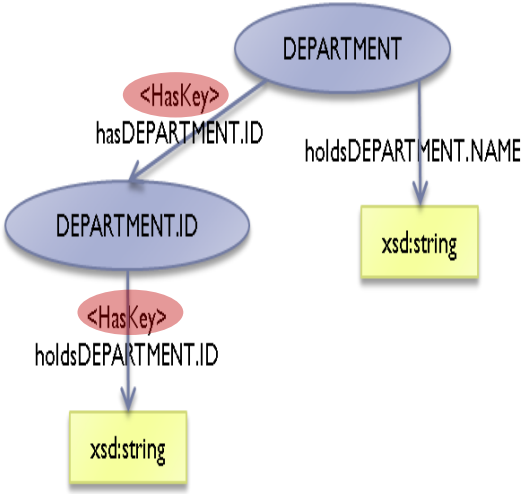












질의 엔진

OWL 위한 질의 언어

- OWL을 위한 질의 언어의 표준은 아직 없는 상태

- SPARQL은 RDF를 위한 질의 언어
  - OWL은 RDF로 직렬화 되기 때문에 SPARQL 또한 OWL을 위한 질의 언어로 사용할 수 있음
  - 그러나 SPARQL이 OWL의 의미를 이해하는 것은 아님

- W3C의 논의 방향
  - SPARQL의 기능 확장을 통해 OWL을 지원하고자 함
    - Conjunctive Query의 특성의 추가를 의미
      - OWL은 FOL 기반이므로 Conjunctive Query(first-order query)를 수행할 수 있기 때문
  - SPARQL-DL
    - [Sirin et al., 2007] "SPARQL Query for OWL-DL"

SPARQL-DL 예제

- Type(?x, Student), Type(?x, ?C), SubClassOf(?C, Employee)
  - ?x
    - 학생임과 동시에 직원인 모든 학생
  - ?C
    - 그 학생들은 어떤 종류의 직원인가? (예. ResearchAssistant)

SPARQL-DL

TBox Atoms	RBox Atoms	ABox Atoms
SubClassOf (C_1, C_2) EquivalentClass(C_1, C_2) DisjointWith(C_1, C_2) ComplementOf(C_1, C_2)	SubPropertyOf(p_1, p_2) EquivalentProperty(p_1, p_2) InverseOf(p_1, p_2) ObjectProperty(p) DataProperty(p) Functional(p) InverseFunctional(p) Symmetric(p) Transitive(p)	Type(a, C) PropertyValue(a, p, d) SameAs(a_1, a_2) DifferentFrom(a_1, a_2)

- $a_{(i)} \in V_{uri} \cup V_{var} \cup V_{bnode}$
- $d \in V_{uri} \cup V_{var} \cup V_{bnode} \cup V_{lit}$
- $C_{(i)} \in V_{var} \cup S_c$
- $p_{(i)} \in V_{uri} \cup V_{var}$

SPARQL-DL의 처리

- 전처리 단계
- 실행 단계
- 출력 단계

전처리 단계

- 변수 객체 생성
  - 변수 초기화: 타입 식별
    - 클래스, [이D] 프로퍼티, 개체, 리터럴
    - 오류 검출
  - 질의 간소화
    - SubClassOf (C_1, C_2), EquivalentClass(C_1, C_2) → EquivalentClass(C_1, C_2)
    - SameAs(a, ?x), Type(a, C) → Type(?x , C)
  - 변수 finalizing
    - SameAs(?x, a_2), DifferentFrom(?x, a_2)
    - ObjectProperty(?p), DataProperty(?p)
  - 변수 개수가 적은 순으로 atom을 reordering
    - Type(?x,GraduateStudent),PropertyValue(?x,?y,?z),Type(?z,Course) →Type(?x,GraduateStudent),Type(?z,Course) ,PropertyValue(?x,?y,?z)

변수 클래스

```
class Var {
  Var_Type type;

  boolean isFinished;

  Set<URI> uri_s;
  Set<IndividualSet> indi_s;
  Set<LiteralSet> lit_s;

  boolean intersect(Set<URI> o) {}
  boolean intersect(Set<IndividualSet> o) {}
  boolean intersect(Set<LiteralSet> o) {}

  void printResultSet(Writer w) {}
}
```

실행 단계

- TBox/RBox atom의 처리
  - 추론기가 처리
    - 결과 셋은 URI 셋
- ABox atom의 처리
  - 질의 엔진에서 처리
    - 추론기, 매핑 계층의 모듈, RDB 스키마의 정보를 이용
    - 결과 셋은 SQL 문장(들)

변수 동기화를 위한 Atom 재실행

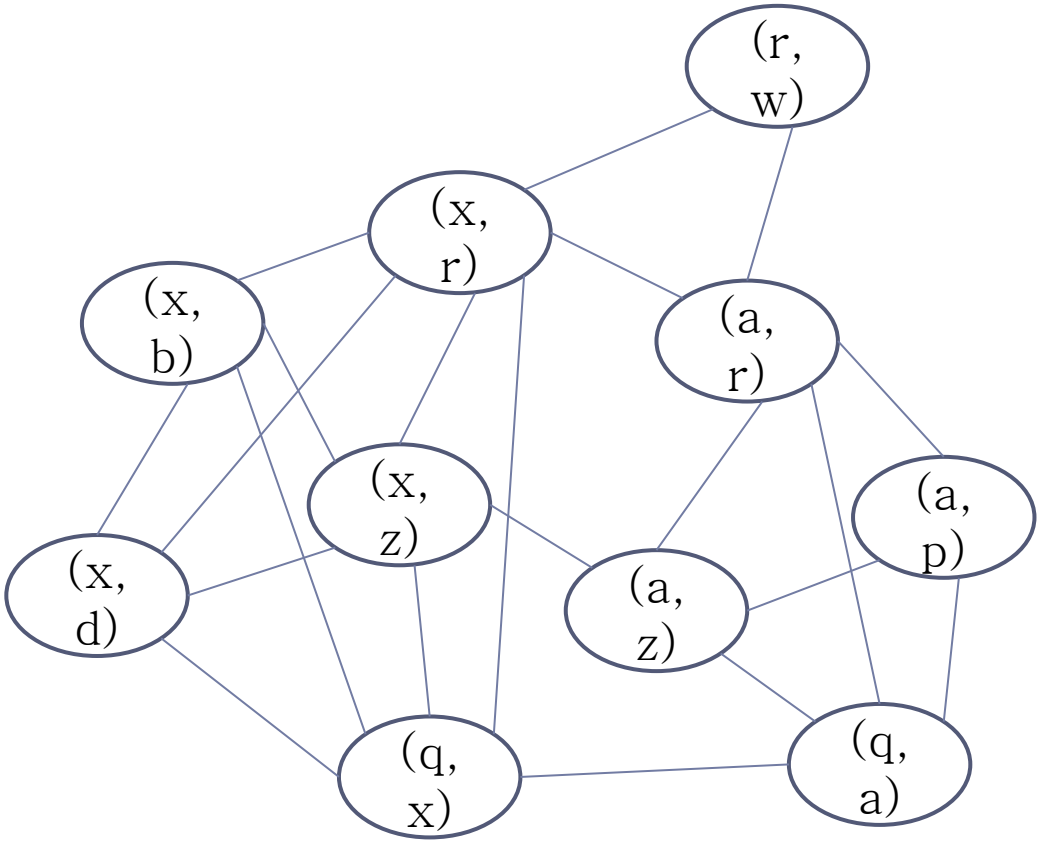
- Type(?w, University), PropertyValue(?r, subOrganizationOf, ?w), PropertyValue(?x, memberOf, ?r), PropertyValue(?a, memberOf, ?r), PropertyValue(?x, takesCourse, ?z), PropertyValue(?a, teacherOf, ?z), PropertyValue(?a, teacherOf, ?p), PropertyValue(?x, advisor, ?b), PropertyValue(?x, teachingAssistentOf, ?d),

  PropertyValue(?q, publicationAuthor,?x),

  PropertyValue(?q, publicationAuthor, ?a)

변수 동기화를 위한 Atom 재실행

- (r, w), (x, r), (a, r), (x, z), (a, z), (a, p), (x, b), (x, d), (q, x), (q, a)



출력 단계

- 변수의 타입이 Individual, Literal일 경우 SQL 결과 셋의 데이터를 URI 포맷으로 가공한 후 출력 스트림으로 출력


Q&A