Semantic Web - Erdoğan Doğdu
Fall 2014 – Midterm - TOBB University of Economics and Tech
Oct 21, 2014 – 120 min **Name**:

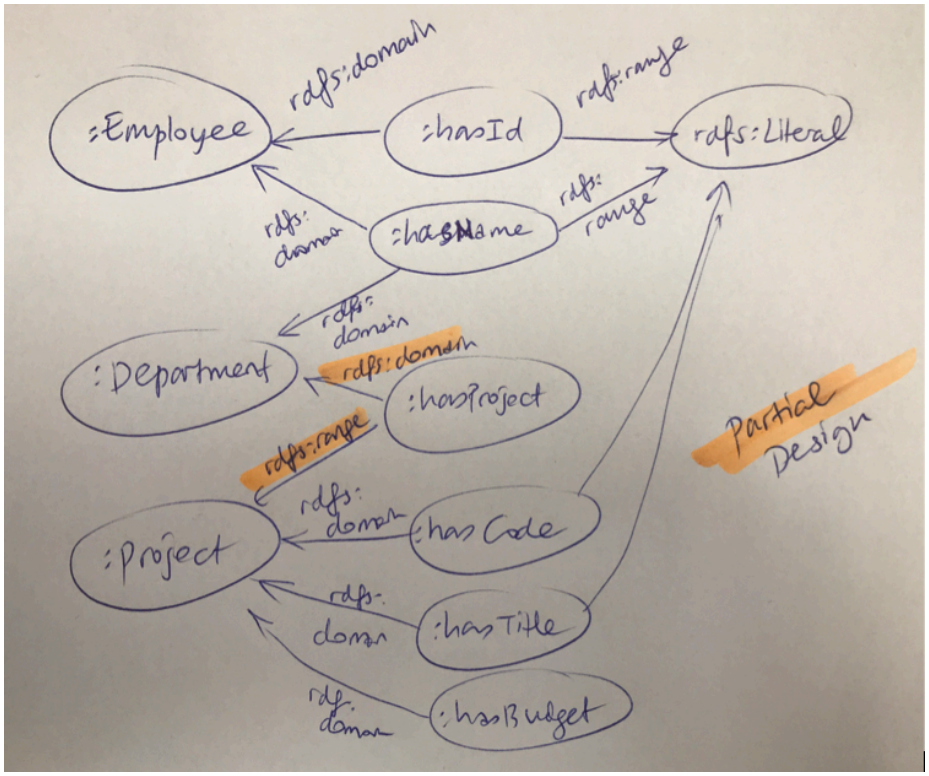
Question	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*Total*
Value	*5*	*30*	*10*	*15*	*10*	*10*	*6*	*9*	*5*	*100*
Points										

1. (5p) Give full name of the following acronyms:
   a. RDF Resource Description Framework
   b. RDFS Resource Description Framework Schema
   c. OWL Web Ontology Language
   d. SPARQL SPARQL Protocol and RDF Query Language
   e. AAA Anyone can say Anything about Any topic

2. (RDF data model and SPARQL, 30) Following relational data model is given. Underlined columns are primary keys and the same columns are used as foreign keys in other tables.

Employee(<u>id</u>, ename)	*Employee (id)*
Department(<u>deptID</u>, dname, id)	*Deparment managed by employee (id)*
Project(<u>pcode</u>, title, deptID, budget)	*Project (pcode) that belongs to department (deptID)*
ProjectEmployee(<u>pcode, id</u>, weekHours)	*Employee (id) working in project (pcode) x hrs a week*

   Answer the following questions:
   a. Convert the above relational data model to an RDF data model. You should capture all classes and properties in the model. Use the graph notation we used in the class (not the one in the book).



Answer the following queries using SPARQL for the RDF data model you developed in (a):

b. Find the projects that belong to "Information Technology" department; list the project codes, titles and the budgets of those projects.

```
SELECT ?c ?t ?b
WHERE { ?p :hasCode ?c.
        ?p :hasTitle ?t.
        ?p :hasBudget ?b.
        ?d :hasProject ?p.
        ?d :hasName ?n.
        FILTER (?n=="Information Technology").
      }
```

c. List the departments (id and name), the number of projects and the total budget of those projects. If a department does not have any project, that department should also be in the list with 0 projects and 0 budget (*output:deptID, name,#projects,totalBudget*)

```
SELECT ?id ?n COUNT(?p) SUM(?b)
WHERE { OPTIONAL { ?d :hasProject ?p. ?p :hasBudget ?b.}
        ?d :hasName ?n.
        ?d :hasID ?id.
      }
GROUP BY ?d
```

d. Find the names of employees who do not work in any project.

```
SELECT ?n
WHERE { ?e :hasName ?n.
        UNSAID { ?e :worksAt ?pe. }
      }
```

e. Find the names of employees who work in projects (pcode=10 **and** pcode=15) **both** (*names should be listed once in the output*).

```
SELECT ?n
WHERE { ?e :hasName ?n.
        ?e :worksAt ?pe1.
        ?pe1 :hasProject ?p1.
        ?p1 :hasCode "10".
        ?e :worksAt ?pe2.
        ?pe2 :hasProject ?p2.
        ?p2 :hasCode "15".
      }
```

f. Find the names of employees who work in **either** project (pcode=10) **or** project (pcode=15) (or may be both) (*names should be listed once in the output*).

```
SELECT DISTINCT ?n
WHERE { ?e :hasName ?n.
        ?e :worksAt ?pe.
        ?pe :hasProject ?p.
        ?p :hasCode ?c.
        FILTER (c=="10" || c=="15")
      }
```

3. (SPARQL, 10p) Following class and porperties are given.

   Classes: :**Student** **:Teacher** **:Course**

   Properties: **(:student1 :takes :course1) (:teacher1 :advises :student1)**

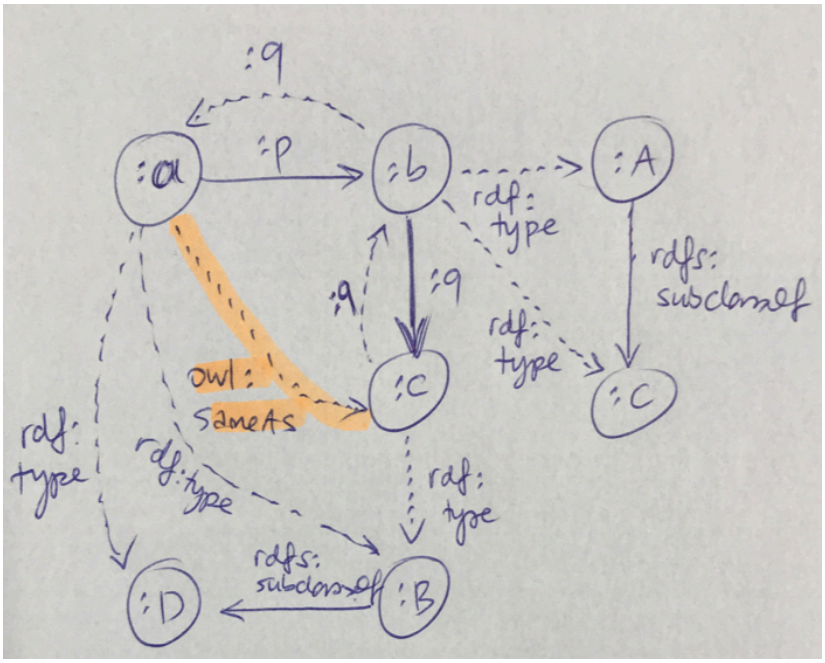
   Using only the above classes and properties, write a SPARQL CONSTRUCT query that creates a graph of those students who are :UnregisteredStudents; meaning students who are not registered to a course yet. Your output should give, for example ":ahmet is an :UnregisteredStudent" and similar triples in RDF format.

```
CONSTRUCT { ?s rdf:type :UnregisteredStudent }
WHERE { UNSAID {?s :takes :c } }
```

4. (Reasoning, 15p) Following triples are given:

```
:p owl:inverseOf :q                    (1)
:q rdf:type owl:FunctionalProperty     (2)
:q rdfs:domain :A                      (3)
:q rdfs:range :B                       (4)
:A rdfs:subClassOf :C                  (5)
:B rdfs:subClassOf :D                  (6)
:a :p :b                               (7)
:b :q :c                               (8)
```

Make all inferences and show them on the graph representation using dashed lines.

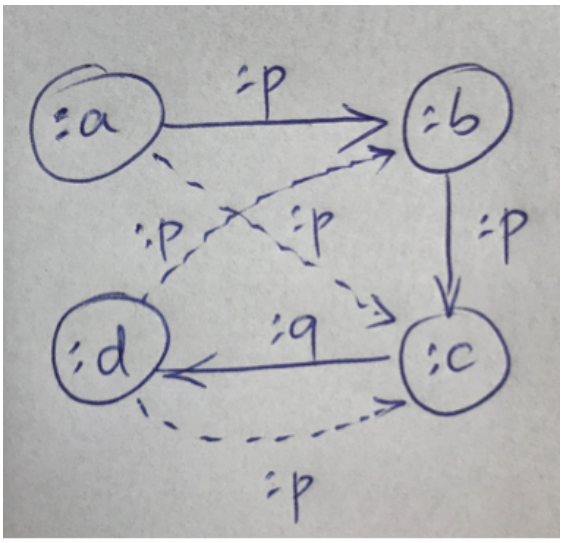


5. (OWL, 10 pts) Explain **`owl:ObjectProperty`** and **`owl:DataProperty`** properties briefly and give an example for each.

6. (OWL reasoning, 10p) Following triples are given:

```
:q rdf:type owl:SymmetricProperty
:p rdf:type owl:TransitiveProperty
:a :p :b
:b :p :c
:c :q :d
```

Make all inferences that are possible with the above assertions.



7. (OWL, 6p) Answer the following:

   a. **ex:Author** and **ex:Writer** classes are given. How do you define in OWL that these two classes are equivelant?

   ```
   ex:Author owl:equivalentClass ex:Writer
   ```

   b. **ex:authorOf** and **ex:writerOf** properties are given. How do you define in OWL that these two properties are equivelant?

   ```
   ex:authorOf owl:equivalentProperty ex:writerOf
   ```

   c. **ex:EDogdu** and **ex:ErdoganDogdu** resources are given. How do you define in OWL that these two resources are equivelant?

   ```
   ex:EDogdu owl:sameAs ex:ErdoganDogdu
   ```

8. (Restriction class, 9p) Consider the following data. In this model, classes are shown as *Class(prop, prop, ...),* object properties in class definitions are depicted as *propertyName(Class),* and other properties in class definitions are data properties (with string values). For example, offeredCourse(Course) object property links an offered course instance to its course (Course), offeredSemester (data property) is the semester (string value) that offered course is offered in.

   Teacher(id, name)

   Course(code, title, credit)

   OfferedCourses(offeredCourse(Course), offeredSemester, taughtBy(Teacher)*)

   Student(studentNum, name, has(OfferedCourseGrade)*)

   OfferedCourseGrade(hasOfferedCourse(OfferedCourse), hasGrade)

   * : *shows one to many relationship (e.g., more than one course being taken by a student or a course being taught by more than one teacher, etc.)*

   (a) Define a restriction class in OWL called `:OrtakEğitimQualifiedStudent` for which if a student took (taken and supposedly passed) `:oeg1` course (Ortak Eğitime Giriş 1), then the student belongs to this class.

   (b) Define a restriction class in OWL called `:HardWorkingTeacher` (teachers who did teach in 2014spring, 2014summer, 2014fall semesters).

   (c) Define a restriction class in OWL called :2014SummerCourses (courses that are offered in 2014summer semester).

9. (OWL Prop., 5 pts) Explain **`owl:InverseFunctionalProperty`** with an example.