Knowledge Engineering and Semantic Web

Exercise Sheet: 6

Will be discussed on: June 27,2023

TUTORS:
Yaser Jaradeh, Hassan Hussien, and some other ORKG members

QUESTIONS: Please don't hesitate to ask any questions. Questions help you and your peers.
PRINT: Please consider the environment before printing the exercise.

1 Review Questions

1. Which statements are true or false?

   (a) SPARQL stands for "SPARQL Protocol and RDF Query Language".
   ✓ **True, see slides**

   (b) SPARQL endpoints expose only one graph.
   ✗ **Multiple graphs can be exposed, see FROM tag**

   (c) SPARQL queries must have prefix definitions.
   ✗ **Prefixes are syntactic sugar to keep queries readable**

   (d) SPARQL queries must have the where clause.
   ✗ **No, but it is better to read**

   (e) All statements in a SPARQL must be closed by a '.'
   ✗ **No, the last one can be without, but again, best practice**
   Example: PREFIX dbr: <http://dbpedia.org/resource/>
   select ?p ?o where {
   dbr:Nikola_Tesla ?p ?o .
   ?o a <http://www.w3.org/2002/07/owl#Class>
   }

   (f) SPARQL queries can only retrieve variables.
   ✗ **No, what about count(?var), avg(?var), min(?var), etc.**

   (g) SPARQL responses are RDF triples.
   ✗ **Depends on the query, SELECT are given in XML,JSON,CSV/TSV format, CONSTRUCT: RDF/XML**

2 Learning by Doing

Open the DBpedia endpoint in your browser : http://dbpedia.org/sparql/

1. Run the example query :
   SELECT DISTINCT ?Concept WHERE [] a ?Concept LIMIT 100

   (a) Explain in your own the query. Particularly explain the individual commands. (SELECT,DISTINCT, WHERE, LIMIT)
   Solution: SELECT defines the variable we want to retrieve. Here Concept. DISTINCT will make the results unique (distinct entries, no duplicates). WHERE defines the graph triple patter, which searches for a blank node that has a property rdf:type (a) and ?Concept is a variable, thus it can be anything. LIMIT will only show the first 100 elements that has been found in the graph.

   (b) How could you extend / modify the query to get the next 10 entries.
   Solution: OFFSET 100 , LIMIT 10

2. Create a SPARQL query to find all triples about Nikola Tesla.

(a) Without using prefixes.
Solution: SELECT ?p ?o WHERE {
<http://dbpedia.org/resource/Nikola_Tesla> ?p ?o .
}

(b) Using prefixes
Solution:
PREFIX dbr: <http://dbpedia.org/resource/>
select ?p ?o where {
dbr:Nikola_Tesla ?p ?o .
}

(c) How can you modify the query so the result will be provided in a triple format.
Solution:
PREFIX dbr: <http://dbpedia.org/resource/> select ?tesla ?p ?o where {
dbr:Nikola_Tesla ?p ?o .
BIND(dbr:Nikola_Tesla AS ?tesla)
}

(d) Return the number of triples associated with Nikola Tesla.
Solution:
PREFIX dbr: <http://dbpedia.org/resource/>
select (COUNT(?p) AS ?pc) WHERE {
dbr:Nikola_Tesla ?p ?o .
}

(e) Create a SPARQL query that will return the individual properties and their counts (given the subject is Nikola Tesla. **Solution:**
PREFIX dbr: <http://dbpedia.org/resource/>
select ?p (COUNT(?p) AS ?pc) WHERE {
dbr:Nikola_Tesla ?p ?o .
}

(f) Create a SPARQL query that will return all different labels for Nikola Tesla **Solution:**
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?lab WHERE
dbr:Nikola_Tesla <http://www.w3.org/2000/01/rdf-schema#label> ?lab .
}

Consider the following knowledge base about people who work for an exemplary company and solve the tasks 2 to 4.

```
@prefix ex:<http://example.org#> .
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:<http://www.w3.org/2001/XMLSchema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.

ex:p1   ex:name             "John"@en;
        ex:salary           "23000"^^xsd:integer;
        ex:birthYear        "1989"^^xsd:integer;
        ex:friendWith       ex:p3;
        ex:knows            ex:p2,ex:p4;
        ex:workingStatus    "fullTime";
        ex:jobTitle         ex:Programmer;
        ex:nationality      ex:American;
        ex:email            "john@fake.com";
        ex:workingProject   ex:pr1.

ex:p2   ex:name             "Jens"@de;
        ex:salary           "43000"^^xsd:integer;
        ex:birthYear        "1977"^^xsd:integer;
        ex:knows            ex:p1, ex:p3, ex:p4;
        ex:workingStatus    "fullTime";
        ex:jobTitle         ex:Manager;
        ex:nationality      ex:German;
        ex:workingProject   ex:pr2.

ex:p3   ex:name             "Hamed"@de;
        ex:salary           "8000"^^xsd:integer;
        ex:birthYear        "1995"^^xsd:integer;
        ex:friendWith       ex:p1;
        ex:knows            ex:p2;
        ex:workingStatus    "partTime";
        ex:jobTitle         ex:Programmer;
        ex:nationality      ex:Iranian;
        ex:email            "hamed@fake.com";
        ex:workingProject   ex:pr2.

ex:p4   ex:name             "Dean"@en;
        ex:salary           "24000"^^xsd:integer;
        ex:birthYear        "1963"^^xsd:integer;
        ex:knows            ex:p1, ex:p2;
        ex:workingStatus    "Retired";
        ex:jobTitle         ex:Manager;
        ex:nationality      ex:American;
        ex:workingProject   ex:pr2.

ex:pr1   a                  ex:Project;
         ex:startYear       "2013"^^xsd:gYear;
         ex:supervisor      ex:p4;
         ex:headWorker      ex:p1.

ex:pr2   a                  ex:Project;
         ex:supervisor      ex:p2;
         ex:advisor         ex:p3.

ex:headWorker   rdfs:subClassOf        ex:Manager.
ex:friendWith   rdfs:subPropertyOf     ex:knows;
                a                      owl:symmetricProperty.
```

3 Explain the queries below in your own words and find their results.

1.
   ```
   PREFIX ex:<http://example.org#>
   SELECT ?name
   {
   ?p   ex:name   ?name;
        ex:salary   ?salary.
   FILTER(?salary>15000)}
   ```

2.
   ```
   PREFIX ex:<http://example.org#>
   ASK {
   ?person ex:name   ?name;
           ex:salary   ?salary;
           ex:nationality   ex:German .
   FILTER(?salary >= 40000)}
   ```

3.
   ```
   PREFIX ex:<http://example.org#>
   SELECT (COUNT(?name) as ?count)
   {
   ?p   ex:name   ?name;
        ex:workingStatus   ?stat.
   MINUS{?p   ex:workingStatus   "Retired"}
   OPTIONAL{?p   ex:email   ?email.}
   FILTER(!bound(?email))
   }
   ```

4.
```
PREFIX ex:<http://example.org#>
SELECT (SUM(?salary) as ?sum)
{
?p   ex:salary   ?salary.
{?p   ex:workingStatus   ?status.
FILTER(?status="partTime")} UNION
{?p   ex:workingStatus   ?status.
FILTER(?status="fullTime")}
}
```

5.
```
PREFIX ex:<http://example.org#>
SELECT DISTINCT ?p ?job ?name2
{
?p   ex:name   ?name;
     ex:jobTitle   ?job;
     ex:knows   ?p2.
?p2   ex:name   ?name2.
FILTER(lang(?name2)="en")
}
```

Solution:

1. Return the name of people with salary more than 15000.
   Result:

```
----------------
| name         |
================
| "John"@en    |
| "Jens"@de    |
| "Dean"@en    |
----------------
```

2. Ask if there is a German person with name who has salary more than or equal to 40000.
   Result: yes (for entry Jens)

3. Return the number of people with name who are working people (not retired) who don't have email.
   Result:

```
---------
| count |
=========
| 1     |
---------
```

4. Return sum of salaries of fullTime and partTime working people.
   Result:

```
---------
| sum   |
=========
| 74000 |
---------
```

5. Return people (without duplication) with their job titles and the English name of people whom they know.
   Result:

```
---------------------------------------
| p     | job           | name2       |
=======================================
| ex:p1 | ex:Programmer | "Dean"@en    |
| ex:p4 | ex:Manager    | "John"@en    |
| ex:p2 | ex:Manager    | "John"@en    |
| ex:p2 | ex:Manager    | "Dean"@en    |
---------------------------------------
```

4 Write SPARQL queries to answer the following requests.

1. The average age of all Working Employees in the year 2016.
2. The salary and email (if it's given) of American employees.
3. Names of people with a salary of less than 20,000 who are not American.
4. Names of supervisors of projects which American people work in.
5. Does any American worker aged over 30 works for the company who is payed more than 30000 annually?

Solution:

1.
```
PREFIX ex:<http://example.org#>
SELECT (AVG(2016 - ?byear) AS ?average)
{
?p   ex:birthYear   ?byear;
     ex:workingStatus   ?status.
FILTER(?status != "Retired")
}
```

2.
```
PREFIX ex:<http://example.org#>
SELECT ?p ?email ?salary
{
?p   ex:salary   ?salary;
     ex:nationality   ?nationality.
FILTER (?nationality = ex:American)
OPTIONAL {?p   ex:email   ?email}
}
```

3.
```
PREFIX ex:<http://example.org#>
SELECT ?name
{
?p   ex:name   ?name;
     ex:nationality   ?nationality;
     ex:salary   ?salary.
FILTER(?salary < 20000 && ?nationality != ex:American)
}
```

4.
```
PREFIX ex:<http://example.org#>
SELECT ?name
{
?prj   ex:supervisor   ?p1.
?p1   ex:name   ?name.
?p2   ex:nationality   ?nationality;
      ex:workingProject   ?prj.
FILTER(?nationality = ex:American)
}
```

5.
```
PREFIX ex:<http://example.org#>
ASK {
?P   ex:nationality   ex:American;
     ex:birthYear   ?year;
     ex:salary   ?salary.
FILTER(?salary >= 30000 && (2018 - ?year > 30 ))
}
```