Semantic Web

Assignment 8

Dr. Jandson S Ribeiro
jandson@uni-koblenz.de

Isabelle Kuhlmann
iskuhlmann@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission by: June 27, 2021
Tutorial on: July 1, 2021

Please submit your solutions to your group's OLAT folder.
Always list all group members contributing to the solution!
Do not plagiarize from others!

For all the assignment questions that require you to code, make sure to include the code in the answer sheet, along with a separate Python file.

Team Name: gamma

1. Aman Bisht (amanvista@uni-koblenz.de)
2. Muralikrishna Naripeddi (mnaripeddi@uni-koblenz.de)
3. Ndang Hesley Fonane (fonaneh@uni-koblenz.de)

1 SPARQL 10 points

Consider the following excerpt of an RDF dataset containing patient information from a single hospital.

```
@prefix x:      <http://example.org/resource/> .
@prefix xo:     <http://example.org/ontology#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .

# patient information

x:pat127     rdf:type             xo:Patient.
x:pat127     rdf:type             xo:Female.
x:pat127     xo:name              "Janis Kranowitz".
x:pat127     xo:address           "Whalers Rd 127, Nantucket, MA".
x:pat127     xo:dateOfBirth       "1978-07-24"^^xsd:date.

# admission information

x:pat127     xo:admission         x:adm1272.
x:adm1272    rdf:type             xo:Admission.
x:adm1272    xo:careUnit          x:OBGyn.
x:adm1272    xo:physician         x:DrCarolineSmith.
x:adm1272    xo:reason            "Child Delivery".
x:adm1272    xo:admissionDate     "2017-03-20"^^xsd:date.
x:adm1272    xo:releaseDate       "2017-03-24"^^xsd:date.

# medical reports during patients time of admission

x:adm1272    xo:report        x:rep12721.
x:rep12721   xo:text      "C-section indicated for patient J. Kranowitz...".
x:rep12721   xo:date          "2017-03-20T08:28:30"^^xsd:dateTime.

x:adm1272    xo:report        x:rep12722.
x:rep12722   xo:text          "Child delivered. It is a healthy boy.".
x:rep12722   xo:date          "2017-03-20T09:01:00"^^xsd:dateTime.

# medical costs for patients admission

x:adm1272    xo:medicalBill   x:bill362.
x:bill362    xo:description   "Caesarean section".
x:bill362    xo:cost          "9350.00"^^xsd:float.

x:adm1272    xo:medicalBill   x:bill363.
x:bill363    xo:description   "Postnatal child care".
x:bill363    xo:cost          "1980.00"^^xsd:float.
...
```

Study the data structure apparent from the above snippet. Based on that structure, formulate SPARQL queries for each of the following tasks.

- Specify your queries as precisely as possible.
- You can assume the used prefixes to be known. You don't need to declare them.

1.1 2.5 points

List all female patients who were admitted before June 2018.

```
SELECT DISTINCT ?patient
WHERE {
        ?patientId xo:name ?patient .
        ?patientId rdf:type xo:Patient .
        ?patientId rdf:type xo:Female .
        ?patientId xo:admission ?admissionId .
        ?admissionId xo:admissionDate ?date
    filter(
    ?date < "2017-06-01"^^xsd:date
    ) .
}
```

1.2 2.5 points

Select all known admissions to the intensive care unit (`xo:ICU`) or the physical rehabilitation unit (`xo:PRU`). The resulting list should state the reasons for the admission, the admission dates and, if applicable, the release dates.

```
SELECT DISTINCT ?patientId ?admissionDate ?releaseDate
WHERE {
        ?patientId xo:name ?patient .
        ?patientId rdf:type xo:Patient .
        ?patientId xo:admission ?admissionId .
        ?admissionId xo:careUnit ?careUnit .
        ?admissionId xo:reason ?reason .
        ?admissionId xo:admissionDate ?admissionDate .
        ?admissionId xo:releaseDate ?releaseDate .
    filter(
    ?careUnit = xo:ICU || ?careUnit = xo:PRU
    ) .
}
```

1.3 2.5 points

List the ten most expensive total medical bills. For that, calculate the sum of costs of all medical bills per person. The resulting list should contain the ten patients with the highest costs and the amount of these costs.

```
SELECT DISTINCT ?patientId ?patient (sum(?cost) as Total_Cost)
WHERE {
        ?patientId xo:name ?patient .
        ?patientId rdf:type xo:Patient .
        ?patientId xo:admission ?admissionId .
        ?admissionId xo:medicalBill ?billId.
        ?billId xo:cost ?cost .
}
group by ?patientId
ORDER BY DESC(?Total_Cost)
limit 10
```

1.4 2.5 points

Count the number of medical bills for each patient admitted to `xo:OBGyn` after June 2017. The resulting list should contain each patient and their number of medical bills.

```
SELECT DISTINCT ?patientId ?patient count(?Bill)
WHERE {
        ?patientId xo:name ?patient .
        ?patientId rdf:type xo:Patient .
        ?patientId xo:admission ?admissionId .
        ?admissionId xo:careUnit xo:OBGyn.
        ?admissionId xo:admissionDate ?admissionDate .
        ?admissionId xo:medicalBill ?billId.
   filter(
    ?admissionDate >= "2017-06-01"^^xsd:date
   )
}
group by ?patientId
```

2 Querying DBPedia 10 points

In this exercise you will write SPARQL queries to retrieve information from the DBPedia SPARQL endpoint (http://dbpedia.org/sparql).

Please take into account that DBPedia may be under maintenance at times, so we recommend you to run your queries early.

Hints:

- dbo : http://dbpedia.org/ontology/, dbp : http://dbpedia.org/property/
- dbr : http://dbpedia.org/resource/, foaf : http://xmlns.com/foaf/0.1/
- You should use SELECT DISTINCT in order to avoid duplicate, if sensible for the respective task.
- You will need sensible FILTERs that modify your query results such that they fit the requirement. You can find possible expression here: https://www.w3.org/TR/rdf-sparql-query/#OperatorMapping.
- You will need to familiarize yourself with properties usually found with certain classes (e.g., birth date of a person). For that, explore example resources in DBPedia.

For the following tasks, specify an appropriate query and give the result you are getting (max. 20, if multiple lines are returned):

2.1 3 points

List those persons who lived through the entire 20th century, i.e., who were born before 1900 and died after 1999. List their names, birth dates and death dates.

```
SELECT ?name ?birth ?death ?person WHERE {
?person dbo:birthDate ?birth .
?person foaf:name ?name .
?person dbo:deathDate ?death .
FILTER (
?birth < "1900-01-01"^^xsd:date &&
?death > "1999-12-31"^^xsd:date) .
} ORDER BY ?name limit 20
```

2.2 4 points

List authors who wrote only books with least 1000 pages, i.e., they did not write any books with less than 1000 pages.

Hint: You can make use of the HAVING modifier[1][2].

[1] https://www.w3.org/TR/sparql11-query/#having
[2] https://en.wikibooks.org/wiki/SPARQL/Modifiers

```
SELECT ?author WHERE {
?work dbo:author ?author .
?work dbo:numberOfPages ?pages .
}
group by ?author
having (?pages > 1000)
```

2.3 3 points

List 10 pairs of (distinct) persons that share the dates for both birth and death (according to DBPedia). They should all have been born in 1927 or after.

```
select ?p1 ?p2 ?birthDate1 ?deathDate1 ?birthDate2 ?deathDate2
where  {
?p1 dbo:birthDate ?birthDate1 .
?p1 dbo:deathDate ?deathDate1 .
?p2 dbo:birthDate ?birthDate2 .
?p2 dbo:deathDate ?deathDate2 .
filter (
?birthDate1 = ?birthDate2 && ?deathDate1 = ?deathDate2 &&
?p1 != ?p2 && ?birthDate1 >= "1927"^^xsd:date
) .
} order by ?p1
LIMIT 10
```

Important Notes

Submission

- Solutions have to be submitted to your group's OLAT folder.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the OLAT folder. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This pdfLaTeX template was adapted by Isabelle Kuhlmann based on the LuaLaTeX version by Lukas Schmelzeisen.

LaTeX

Use `pdflatex assignment_X.tex` to build your PDF.