Semantic Web – Tutorial #9

Isabelle Kuhlmann

Institute for Web Science & Technologies
University of Koblenz-Landau

July 1, 2021 | Assignment 8

1: SPARQL

1: SPARQL

Task: Consider the given excerpt of an RDF dataset containing patient information from a single hospital. Based on the apparent data 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.

1: SPARQL

Task: List all female patients who were admitted before June 2018.

```
SELECT ?patient
WHERE {
    ?patient rdf:type xo:Patient .
    ?patient rdf:type xo:Female .
    ?patient xo:admission ?adm .
    ?adm xo:admissionDate ?admDate .
    FILTER ( ?admDate < "2018-06-01" )
}
```

1: SPARQL

Task: 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 ?adm ?admReason ?admDate ?releaseDate
WHERE {
    ?adm xo:reason ?admReason.
    ?adm xo:admissionDate ?admDate.
    OPTIONAL ?adm xo:releaseDate ?releaseDate.
    ?adm xo:careUnit ?careType.

    FILTER (?careType = xo:ICU | ?careType = xo:PRU)
}
```

1: SPARQL

Task: 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 ?patient ( SUM(?cost) AS ?sumCosts )
WHERE {
    ?patient rdf:type xo:Patient .
    ?patient xo:admission ?adm .
    ?adm xo:medicalBill ?medBill .
    ?medBill xo:cost ?cost .
}
GROUP BY ?patient
ORDER BY DESC(?sumCosts)
LIMIT 10
```

1: SPARQL

Task: 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 ?patient ( COUNT(?medBill) AS ?numberOfBills )
WHERE {
    ?patient rdf:type xo:Patient .
    ?patient xo:admission ?adm .
    ?adm xo:careUnit xo:OBGyn .
    ?adm xo:medicalBill ?medBill .
    ?adm xo:admissionDate ?admDate .
    FILTER ( ?admDate > "2017-05-31")
}
GROUP BY ?patient
```

2: Querying DBPedia

1: Querying DBPedia

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

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

1: Querying DBPedia

Task: 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 DISTINCT ?person ?name ?birth ?death
WHERE {
    ?person rdf:type dbo:Person.
    ?person foaf:name ?name.
    ?person dbo:birthDate ?birth.
    ?person dbo:deathDate ?death.
    FILTER (xsd:date(?birth) < "1900-01-01"^^xsd:date
        && xsd:date(?death) > "1999-12-31"^^xsd:date)
}
LIMIT 20
```

1: Querying DBPedia

Task: 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.

```
SELECT DISTINCT ?author
    WHERE {
    ?author rdf:type dbo:Person.
    ?book dbo:author ?author.
    ?book rdf:type dbo:Book.
    ?book dbo:numberOfPages ?numPages.
}

GROUP BY ?author
HAVING ( MIN(?numPages) > 999 )
LIMIT 20
```

1: Querying DBPedia

Task: 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 DISTINCT ?person ?name ?birth ?death ?person2 ?name2 ?birth2 ?death2
WHERE {
    ?person rdf:type dbo:Person.
    ?person foaf:name ?name.
    ?person dbo:birthDate ?birth.
    ?person dbo:deathDate ?death.

    ?person2 rdf:type dbo:Person.
    ?person2 foaf:name ?name2.
    ?person2 dbo:birthDate ?birth2.
    ?person2 dbo:deathDate ?death2.

    FILTER ("1927-01-01^^xsd:date > xsd:date(?birth)
        && ?birth = ?birth2 && ?death = ?death2
        && ?person != ?person2 && ?name != ?name2) }
LIMIT 10
```