E04 Querying RDF Datasets with SPARQL

Quiz

Decide whether the following statements are true or false.

Q4.1 A basic graph pattern without variables is valid.

Q4.2 *SPARQL* queries are expressed in Turtle syntax.

Q4.3 Turtle abbreviations “,” and “;” are allowed inside the WHERE clause, as are unnamed blank nodes via “[]”.

Q4.4 *SELECT DISTINCT* filters out duplicate solution mappings.

Q4.5 *ASK* queries return an RDF graph.

Q4.6 *GRAPH* and *UNION* may lead to unbound solution mappings.

Q4.7 Blank nodes in the *CONSTRUCT* clause lead to new blank nodes (with generated blank node labels) in the solution sequence.

Q4.8 SPARQL queries are always translated to SQL queries and then evaluated in a relational database.

Q4.9 A SPARQL processor first performs the *LIMIT* operation before the *ORDER BY*.

Q4.10 In SPARQL, the default graph is always the union of all named graphs.

Exercises

E4.1 Given the following query:

```
SELECT ?x ?z
WHERE {
{ ?x :p ?z . }
UNION
{ ?x :q _:y . _:y :q ?z . }
}
```

Identify the triple patterns and the basic graph patterns in the query.

E4.2 Write a query that checks whether triple :s :p :o . occurs in RDF document **http://example.org/g.ttl**.

E4.3 Write a query that returns all triples (as graph) from the graphs **people.ttl** and **lib.ttl**.

E4.4 Write a query that returns the URIs of the property resources occurring in the default graph.

E4.5 Write a query that returns all triples of paths up to hop-2 from foo#bar, assuming the following graph at **foo.ttl**.

```
@prefix : <foo#> .

:bar  :p    :o .
:o    :q    :o2 .
:o2   :r    :o3 .
```

E4.6 Write a query that returns all triples of paths up to hop-2 from foo#bar, assuming the following graph at **foo.ttl**.

```
@prefix : <foo#> .

:bar :p [ :q [ :r :o3 ] ].
```

E4.7 Give the solutions to the following query, assuming the RDF document **foo.ttl** of E 4.6:

```
SELECT *
FROM <foo.ttl>
WHERE {
      ?s    ?p    ?o .
      ?s1   ?p1   ?o1 .
}
```

E4.8 Given the following RDF graph **doc.ttl**:

```
@prefix : <http://example.org/foo#> .

:s :p :o .
```

Write a query that checks whether **doc.ttl** is an instance of the following graph:

```
@prefix : <http://example.org/foo#> .

_:s :p _:o .
```

E4.9 Given the following RDF document at **labels.ttl**, write a SPARQL query that returns only the labels in the German language.

```
@prefix : <labels#> .

:HD   :label      "Heidelberg"@de , "ハイデルベルク"@ja , "
      하이델베르크"@ko ,"Гейдельберг"@ru .


:N    :label      "Niurnbergas"@it , "Norymberga"@pl , "Núremberg"@es ,
                  "Nürnberg"@de , "纽伦堡"@zh .
```

E4.10 Given the following RDF document at **titles.ttl**, write a SPARQL query that returns all titles and the year of publication, if available.

```
@prefix : <titles#> .

[     :name      "Meditationes de prima philosophia"@la ;
      :year      1641 ;
      :author    "René Descartes" ] .


[     :titel "Die beiden Grundprobleme der Erkenntnistheorie"@de ;
      :author "Karl Popper" ] .


[     :entitled  "Philosophische Untersuchungen"@de ;
      :author    "Ludwig Wittgenstein" ] .
```

E4.11 Given the following RDF document at **observations.ttl**, write a query that returns a table with the temperature, the location and the date.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <observations#> .

[ :temperature [   :value        14 ;
                   :unit         :Celsius ;
                   :date         "2016-04-20"^^xsd:date ;
                   :refArea      :Heidelberg ] ] .

[ :temperature [   :value        -18 ;
                   :unit         :Celsius ;
                   :date         "2016-01-01"^^xsd:date ;
                   :refArea      :Novosibirsk ] ] .

[ :temperature [   :value        34 ;
                   :unit         :Celsius ;
                   :date         "2016-12-11"^^xsd:date ;
                   :refArea      :Canberra ] ] .

[ :temperature [   :value        32 ;
                   :unit         :Celsius ;
                   :date         "2016-12-12"^^xsd:date ;
                   :refArea      :Canberra ] ] .
```

E4.12 Write a query over the RDF graph in E 4.11 that returns the latest temperature in Canberra

E4.13 Write a query on the RDF document from E 4.11 that, next to the location and the date, returns the temperature in Fahrenheit ($T_{\circ F} = T_{\circ C} \times 9/5 + 32$.)

E4.14 Write a query that returns the URIs and labels of the celestial bodies. Use FROM and FROM NAMED to construct your RDF dataset for the query, and GRAPH to only select planets.

The following graph at **planets.ttl** describes celestial bodies.

```
@prefix : <astro.ttl#> .

:Mercury   :orbits :Sun .
:Venus     :orbits :Sun .
:Earth     :orbits :Sun .
:Mars      :orbits :Sun .
```

The following graph at **objects.ttl** contains labels of celestial bodies.

```
@prefix : <astro.ttl#> .

:Mercury   :label "Mercury" .
:Venus     :label "Venus" .
:Earth     :label "Earth" .
:Mars      :label "Mars" .
:Pluto     :label "Pluto".
```

E4.15 Assume the following graph at **integer.ttl**:

```
@prefix : <integer#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:a :value "023"^^xsd:integer .
```

The following query does not return any solutions:

```
SELECT ?x
FROM <integer.ttl>
WHERE {
   ?x <integer#value> 23 .
}
```

The following query, on the other hand, returns :a as solution:

```
SELECT ?x
FROM <integer.ttl>
WHERE {
   ?x <integer#value> ?y .
   FILTER (?y = 23)
}
```

Why?

Practices

P3.1 Try out the SPARQL endpoint of nobelprize.org.

P3.2 Try out the SPARQL endpoint of DBpedia.org.

P3.3 Try out the SPARQL endpoint of Wikidata.org.

Learning Goals

G 4.1 Write BGP queries in SPARQL with query forms (SELECT, CONSTRUCT, ASK and DESCRIBE).

G 4.2 Use FROM, FROM NAMED and GRAPH in queries in conjunction with RDF datasets.

G 4.3 Correctly apply UNION and OPTIONAL in queries.

G 4.4 Use FILTER and BIND … AS in conjunction with expressions involving functions.

G 4.5 Describe the handling of typed literals in SPARQL graph pattern matching and filter expressions.

G 4.6 Apply ORDER BY, LIMIT and OFFSET in queries.