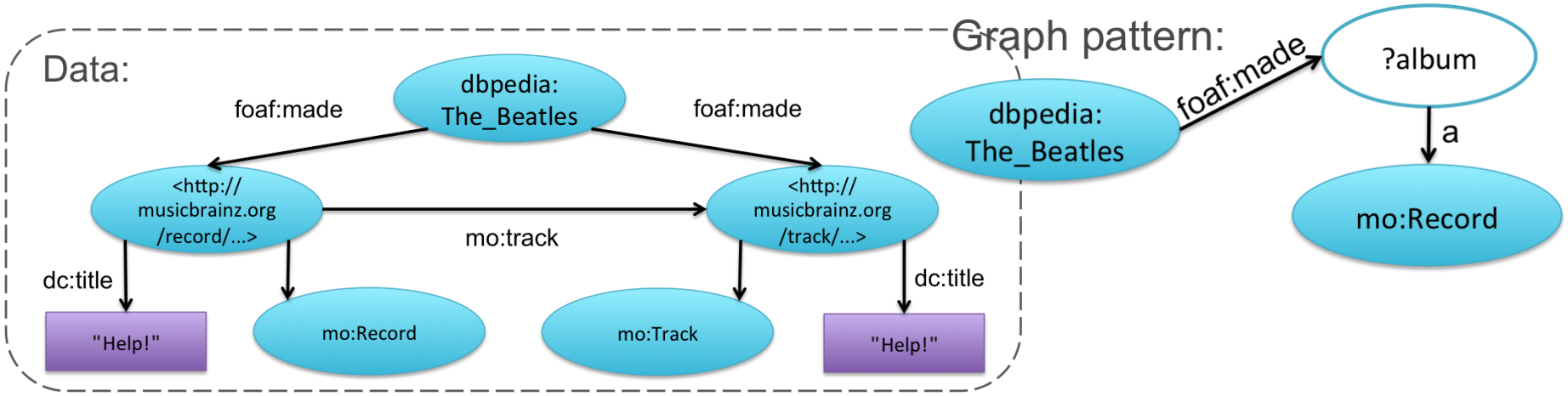
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF BUSINESS
AND ECONOMICS

E04 Querying RDF Datasets with SPARQL

Slides: Kian Schmalenbach

CHAIR OF TECHNICAL INFORMATION SYSTEMS



CC - Creative Commons Licensing

- This set of slides is part of the lecture „Foundations of Linked Data“ held at Friedrich-Alexander-Universität Erlangen-Nürnberg
- The content of the lecture was prepared based on the book „Introduction to Linked Data“ by Prof. Andreas Harth
- The slides were prepared by Maribel Acosta and Kian Schmalenbach
-
- Please cite as “Foundations of Linked Data (Exercises): Querying RDF Datasets with SPARQL” © Kian Schmalenbach | CC BY 4.0

Quiz (1)

Decide whether the following statements are true or false:

1) A basic graph pattern without variables is valid.

   TRUE

2) SPARQL queries are expressed in Turtle syntax.

   FALSE: SPARQL syntax contains additional elements (e.g. subgraphs “{ }”).

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

   TRUE

4) `SELECT DISTINCT` filters out duplicate solution mappings.

   TRUE

Quiz (2)

Decide whether the following statements are true or false:

5) `ASK` queries return an RDF graph.

FALSE: They return a boolean value (i.e. `true` or `false`).

6) `GRAPH` and `UNION` may lead to unbound solution mappings.

TRUE: Using `GRAPH` or `UNION` might lead to variables without assigned value.

7) Blank nodes in the `CONSTRUCT` clause lead to new blank nodes (with generated blank node labels) in the solution sequences.

TRUE

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

FALSE: SPARQL queries are evaluated by SPARQL processors.

Quiz (3)

Decide whether the following statements are true or false:

9) A SPARQL processor first performs the `LIMIT` operation before the `ORDER BY`.

   FALSE: The processing order is vice versa.

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

    FALSE: Named graphs are not part of the default graph.

Exercise 4.1 (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.

Exercise 4.1 (2)

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

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

Solution:

Triple Patterns:

- `?x :p ?z .`
- `?x :q _:y .`
- `_:y :q ?z .`

Basic Graph Patterns:

- `BGP(?x :p ?z .)`
- `BGP(?x :q _:y . _:y :q ?z .)`

Exercise 4.2

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

Solution:

```
PREFIX : <#>
ASK
FROM <http://example.org/g.ttl>
WHERE {
  :s :p :o .
}
```

Exercise 4.3

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

Solution:

```
CONSTRUCT {
  ?subject ?predicate ?object .
}
FROM <people.ttl>
FROM <lib.ttl>
WHERE {
  ?subject ?predicate ?object .
}
```

Exercise 4.4

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

Solution:

```
SELECT ?predicate
WHERE {
  _:s ?predicate _:o .
}
```

Exercise 4.5 (1)

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 .
```

Exercise 4.5 (2)

Write a query that returns all triples of paths up to hop-2 from `foo#bar`.

Solution:

```
PREFIX : <foo#>
CONSTRUCT {
  :bar ?p1 ?o1 .
   ?o1 ?p2 ?o2 .
}
FROM <foo.ttl>
WHERE {
  :bar ?p1 ?o1 .
  OPTIONAL { ?o1 ?p2 ?o2 . }
}
```

foo.ttl

```
@prefix : <foo#> .
:bar :p :o .
:o :q :o2 .
:o2 :r :o3 .
```

Exercise 4.6

Write a query that returns all triples of paths up to hop-2 from `foo#bar`, assuming the following document at foo.ttl:

Solution:

```
PREFIX : <foo#>
CONSTRUCT {
  :bar ?p1 ?o1 .
   ?o1 ?p2 ?o2 .
}
FROM <foo.ttl>
WHERE {
  :bar ?p1 ?o1 .
  OPTIONAL { ?o1 ?p2 ?o2 . }
}
```

foo.ttl

```
@prefix : <foo#> .
:bar :p [ :q [ :r :o3 ] ].
```

Exercise 4.7

Give the solutions to the following query, given the RDF document foo.ttl of E 4.6:

foo.ttl

```
@prefix : <foo#> .
:bar :p [
  :q [ :r :o3 ]
].
```

query

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

Solution:

?s	?p	?o	?s1	?p1	?o1
:bar	:p	_:b0	:bar	:p	_:b0
:bar	:p	_:b0	_:b0	:q	_:b1
:bar	:p	_:b0	_:b1	:r	:o3
_:b0	:q	_:b1	:bar	:p	_:b0
_:b0	:q	_:b1	_:b0	:q	_:b1
_:b0	:q	_:b1	_:b1	:r	:o3
_:b1	:r	:o3	:bar	:p	_:b0
_:b1	:r	:o3	_:b0	:q	_:b1
_:b1	:r	:o3	_:b1	:r	:o3

Exercise 4.8 (1)

Given the following RDF graph at 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 .
```

Solution:

```
ASK
FROM <doc.ttl>
WHERE {
  _:s <http://example.org/foo#p> _:o .
}
```

Bonus question: Why is this approach not working with arbitrary “instance candidates”?

Exercise 4.8 (2)

Answer to the bonus question:

Now consider the following RDF graph at doc2.ttl:

```
@prefix : <http://example.org/foo#> .
:s :p :o .
:foo :bar :foobar .
```

Note that the query on the previous slide still returns `true`, even though doc2.ttl is clearly not an instance of the second graph!

Exercise 4.9

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 "Norimberga"@it , "Norymberga"@pl , "Núremberg"@es , "Nürnberg"@de , "堡"@zh .
```

Solution:

```
SELECT ?label
FROM <labels.ttl>
WHERE {
  [ <labels#label> ?label ]
  FILTER(lang(?label)="de")
}
```

Exercise 4.10 (1)

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" ] .
```

Exercise 4.10 (2)

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

Solution:

```
PREFIX : <titles#>
SELECT ?title ?year
FROM <titles.ttl>
WHERE {
  { ?book :name ?title . }
  UNION
  { ?book :titel ?title . }
  UNION
  { ?book :entitled ?title . }
  OPTIONAL { ?book :year ?year . }
}
```

titles.ttl

```
@prefix : <titles#> .
[ :name "..."@la ; :year 1641 ;
   :author "René Descartes" ] .
[ :titel "..."@de ;
   :author "Karl Popper" ] .
[ :entitled "..."@de ;
   :author "Ludwig Wittgenstein" ] .
```

Exercise 4.11 (1)

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 ] ] .
```

Exercise 4.11 (2)

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

Solution:

```
PREFIX : <observations#>

SELECT ?value ?unit ?location ?date
FROM <observations.ttl>
WHERE {
  _:bn :temperature
       [ :value ?value ;
         :unit ?unit ;
         :date ?date ;
         :refArea ?location ] .
}
```

Bonus question: How can we return the temperature value and its unit as a concatenated string?

Exercise 4.11 (3)

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

Answer to the Bonus Question:

```
PREFIX : <observations#>

SELECT ?temp ?location ?date
FROM <observations.ttl>
WHERE {
  _:bn :temperature
       [ :value ?value ;
         :unit ?unit ;
         :date ?date ;
         :refArea ?location ] .
  BIND(concat(str(?value), "° ", str(?unit)) as ?temp)
}
```

Exercise 4.12

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

Solution:

```
PREFIX : <observations#>
SELECT ?temperature
FROM <observations.ttl>
WHERE {
  _:bn :temperature
       [ :refArea :Canberra;
         :value ?temperature;
         :date ?date ] .
}
ORDER BY DESC(?date)
LIMIT 1
```

Exercise 4.13

Write a query over the RDF graph in E 4.11 that, next to the location and the date, returns the temperature in Fahrenheit.
($T_{\circ F} = T_{\circ C} \cdot {}^{9}/_{5} + 32$)

Solution:

```
PREFIX : <observations#>
SELECT ?location ?date ?temp_fahrenheit
FROM <observations.ttl>
WHERE {
  _:bn :temperature
       [ :value ?value;
         :date ?date;
         :refArea ?location ] .
  BIND((?value * (9/5) + 32) as ?temp_fahrenheit)
}
```

Exercise 4.14 (1)

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".
```

Exercise 4.14 (2)

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.

planets.ttl

```
@prefix : <astro.ttl#> .
:Mercury :orbits :Sun .
:Venus :orbits :Sun .
:Earth :orbits :Sun .
:Mars :orbits :Sun .
```

objects.ttl

```
@prefix : <astro.ttl#> .
:Mercury :label "Mercury" .
:Venus :label "Venus" .
:Earth :label "Earth" .
:Mars :label "Mars" .
:Pluto :label "Pluto".
```

Solution:

```
PREFIX : <astro.ttl#>
SELECT ?planet ?label
FROM <objects.ttl>
FROM NAMED <planets.ttl>
WHERE {
  ?planet :label ?label .
  GRAPH <planets.ttl> { ?planet ?p [] . }
}
```

Exercise 4.15

Assume the following graph at integer.ttl:

```
@prefix : <integer#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
:a :value "023"^^xsd:integer .
```

The left query returns no results, the right one returns `:a`. Why?

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

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

Solution:

Filter expressions in SPARQL take into account different lexical forms of literals, BGP matching does not.