Semantic Web

8. SPARQL

PD Dr. Matthias Thimm

thimm@uni-koblenz.de

Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau

WeST
People and Knowledge Networks

Overview SPARQL

- SPARQL Query
  - Declarative query language for RDF data
  - `http://www.w3.org/TR/rdf-sparql-query/`
- SPARQL Algebra
  - Standard for communication between SPARQL services and clients
  - `http://www.w3.org/2001/sw/DataAccess/rq23/rq24-algebra.html`
- SPARQL Update
  - Declarative manipulation language for RDF data
  - `http://www.w3.org/TR/sparql11-update/`
- SPARQL Protocol
  - Standard for communication between SPARQL services and clients
  - `http://www.w3.org/TR/sparql11-protocol/`

Outline

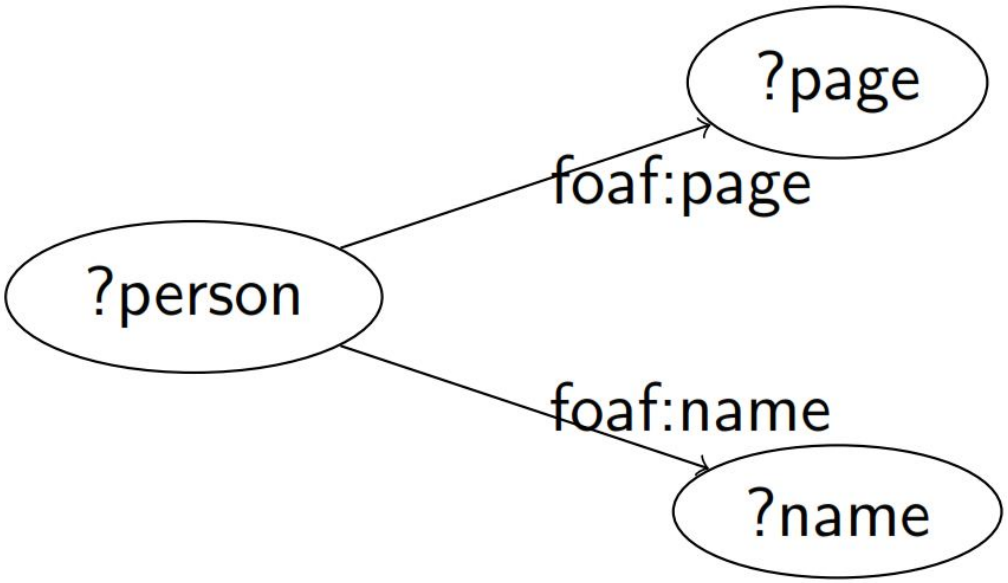
Outline

SPARQL Query Types

- **SELECT**
  - returns the set of variables bound in a query pattern match
- **CONSTRUCT**
  - returns an RDF graph constructed by substituting variables in a set of triple templates
- **DESCRIBE**
  - returns an RDF graph that describes the resources found
- **ASK**
  - returns whether a query pattern matches any triples or not (true / false query)

SPARQL SELECT Query - Example

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
    ?person foaf:page ?page .
    ?person foaf:name ?name .
}
```



SPARQL ASK Query - Example

Query: Is the river Amazon longer than the river Nile?

```
PREFIX prop: <http://dbpedia.org/property/>
ASK
{
<http://dbpedia.org/resource/Amazon_River> prop:length ?amazon .
<http://dbpedia.org/resource/Nile> prop:length ?nile .
  FILTER(?amazon > ?nile)
}
```

Answer: true

SPARQL CONSTRUCT Query - Example

CONSTRUCT query is used to build an RDF graph.

```
  PREFIX ex: <http://example.org/schema/>
  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  CONSTRUCT
  {
   ?person rdf:type ex:Adult .
  }
  WHERE
{
   ?person ex:age ?age .
   FILTER ( ?age > 17)
}
```

Result is an RDF graph.

SPARQL DESCRIBE Query - Example

DESCRIBE query is used to provide further information about an RDF resource

```
DESCRIBE <http://dbpedia.org/resource/Amazon_River>
```

Result is an extensive description of the resource "Amazon River". Here is just an excerpt:

```
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpedia:<http://dbpedia.org/resource/> .
@prefix dbpedia owl :<http://dbpedia.org/ontology/> .

dbpedia : Amazon_River rdf:type dbpedia-owl:NaturalPlace ,
          dbpedia-owl:Place .
...
```

Outline

Triple pattern

RDF triple/statement: smallest unit of information in RDF

```
dbpedia:The_Beatles foaf:name "The Beatles"
```

RDF triple pattern: an RDF triple that may contain variables

```
dbpedia:The_Beatles foaf:made ?album
?album mo:track ?track
?album ?p ?o
```

Convention: Variables have the symbol "?" as prefix ("$" is also allowed)

Basic SPARQL Patterns

```
SELECT <vars> WHERE { <graph pattern> }
```

- Graph pattern:
  - template for matching in RDF graph constructed using RDF triple patterns
  - describe sub-graphs of the queried graph
  - are RDF graphs specified in Turtle syntax
- Matching patterns in the **WHERE** clause
  - Matching conjunction of triple patterns
  - Matching a triple pattern to a graph
    - Finding bindings between variables and RDF Terms
  - Underneath use of reasoners
    - Inferring triples originally not present in the knowledge base

Query Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
_:a foaf:name ‘‘Jandson Santos’’ .
_:a foaf:homepage <https://west.uni-koblenz.de/.../jandson-santos> .
_:b foaf:name ‘‘Matthias Thimm’’ .
_:b foaf:homepage <http://www.mthimm.de> .
```

Query:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
  ?person foaf:homepage ?page .
  ?person foaf:name ?name . }
```

Query Result:

name	page
"Jandson Santos"	<https://west.uni-koblenz.de/.../jandson-santos>
"Matthias Thimm"	<http://www.mthimm.de>

Outline

Filters

- Further constrain graph patterns
- Applied to the **whole group** of patterns
- FILTER clause:
  - Support for AND and OR logic operators
  - Extensive applications for testing literals
  - Support for numerical operations
  - Support for math equality operators for literals ($<; =; >$)
  - Use of regular expressions
  - support for data types defined in XSL (e.g., comparision of dates, time)
  - possible comparison of resources (equal, not equal)
  - even possible user extensions

FILTER - Value Constraints

Data:

```
@prefix dc:<http://purl.org/dc/elements/1.1/> .
@prefix ex:<http://example.org/book/> .
@prefix ns:<http://example.org/ns#> .
ex:book1 dc:title ‘‘SPARQL Tutorial’’ .
ex:book1 ns:price 42 .
ex:book2 dc:title ‘‘The Semantic Web’’ .
ex:book2 ns:price 23 .
```

Query:

```
PREFIX dc:<http://purl.org/dc/elements/1.1/ >
PREFIX ns:<http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
        FILTER ?price < 30
        ?x dc:title ?title .}
```

Query Result:

title	price
"The Semantic Web"	23

FILTER - Regular Expressions

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ‘‘Jandson Santos’’ .
_:a foaf:homepage <https://west.uni-koblenz.de/.../jandson-santos> .
_:b foaf:name ‘‘Matthias Thimm’’ .
_:b foaf:homepage <http://www.mthimm.de > .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:homepage ?page .
 ?person foaf:name ?name .
 FILTER regex(?name, ‘‘Jandson’’)
}
```

Query Result:

name	page
"Jandson Santos"	<https://west.uni-koblenz.de/.../jandson-santos>

FILTER - Regular Expressions

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ‘‘Jandson Santos’’ .
_:a foaf:homepage <https://west.uni-koblenz.de/.../jandson-santos> .
_:b foaf:name ‘‘Matthias Thimm’’ .
_:b foaf:homepage <http://www.mthimm.de > .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:homepage ?page .
 ?person foaf:name ?name .
 FILTER regex (?name, ‘‘thimm’’, ‘‘i’’)
}
```

Query Result: case insensitive

name	page
"Matthias Thimm"	<http://www.mthimm.de>

Scope of Filters

- Filters are applied to the whole group of patterns where they appear.

```
{ ?x foaf:name ?name .
  ?x foaf:homepage ?page .
  FILTER regex(?name, ``Jandson'') }

{ ?x foaf:name ?name .
  FILTER regex(?name, ``Jandson'')
  ?x foaf:homepage ?page .}

{ FILTER regex(?name, ``Jandson'')
  ?x foaf:name ?name .
  ?x foaf:homepage ?page .}
```

- These patters are equivalent (have the same solution)

Outline

Optional Patterns

- include optional triple patterns to match
- optional is a pattern itself — can include further constraints

```
SELECT <vars> WHERE {
  <graph pattern>
   OPTIONAL {...}
}
```

Query Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ‘‘Jandson Santos’’ .
_:a foaf:homepage <https://west.uni-koblenz.de/.../jandson-santos> .
_:b foaf:name ‘‘Matthias Thimm’’ .
_:b foaf:mbox <thimm@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
 ?person foaf:homepage ?page .
}
```

Query Result:

name	page
"Jandson Santos"	<https://west.uni-koblenz.de/.../jandson-santos>

Query Example with OPTIONAL

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ‘‘Jandson Santos’’ .
_:a foaf:homepage <https://west.uni-koblenz.de/.../jandson-santos> .
_:b foaf:name ‘‘Matthias Thimm’’ .
_:b foaf:mbox <thimm@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
 OPTIONAL ?person foaf:homepage ?page .
}
```

Query Result:

name	page
"Jandson Santos"	<https://west.uni-koblenz.de/.../jandson-santos>
"Matthias Thimm"	

Outline

Union of Graph Pattern

- combining alternative graph patterns
- if more than one of the alternatives matches, all the possible pattern solutions are included in result

```
SELECT <vars>
WHERE{
  { <graph pattern> }
  UNION
  { <graph pattern> }
}
```

Query Example — UNION

Data:

```
@prefix dc10: <http:/purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
book1 dc10:title ‘‘SPARQL Tutorial’’ .
book1 dc10:creator ‘‘Alice’’ .
book2 dc11:title ‘‘The Semantic Web’’ .
book2 dc11:creator ‘‘Robert’’ .
```

Query:

```
PREFIX dc10:<http://purl.org/dc/elements/1.0/>
PREFIX dc11:<http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { { ?x dc10:title ?title . }
          UNION
          { ?x dc11:title ?title . } }
```

Query Result: title

"SPARQL Tutorial"

"The Semantic Web"

Outline

Modification of the Result Set

Result of SPARQL query can be further modified

- ORDER BY
  - sort results alphabetically/numerically by specific variable
- LIMIT
  - Limit number of returned results (only top n results)
- OFFSET
  - Skip n top results and return the rest

These expressions can be combined in a query.

Sequencing and Limiting Results

Query:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
    ?person foaf:homepage ?page .
    ?person foaf:name ?name .
}
ORDER BY ?name
LIMIT 20
OFFSET 10
```

Tricky Negation

Find people who do **not** know Jandson.
First attempt:

```
SELECT ?person
WHERE {
    ?person ex:knows ?person2 .
    FILTER ( ?person2 != ‘‘Jandson’’) }
```

Data:

```
ex:paul ex:knows ‘‘Jandson’’
ex:paul ex:knows ‘‘Matthias’’
```

... so ex:paul is still a valid answer (do you know why?)
... and we do not want this!

Negation with bound

Find people who do **not** know Jandson.

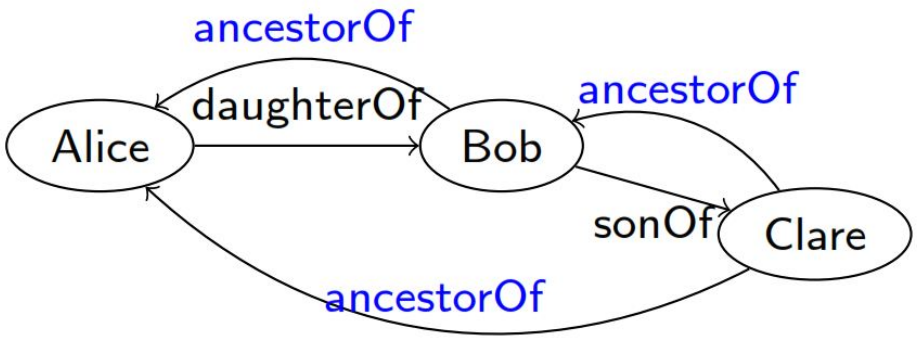
- now the correct way using bound expression and optional graph pattern

```
SELECT ?person
WHERE {
    OPTIONAL { ?person ex:knows ?person2 .
               FILTER (?person2 = ``Jandson'') }
    FILTER ( !bound(?person2) )
}
```

Querying for Inferred Knowledge

- SPARQL does not have specific constructs for accessing inferred knowledge
  - Underlying knowledge base is responsible for supporting inference, e.g.,
    - Class hierarchy
    - Property hierarchy
    - Transitive or symmetric properties
    - OWL restrictions
    - Defining classes by unions and/or intersections
- Different knowledge bases can offer different level of support
  - Same knowledge in different knowledge bases may return different results for the same query, depending on supported entailment

Query Example



ancestorOf = owl:transitiveProperty $\wedge$ union(inverse(daughterOf), inverse(sonOf))

Find ancestors of Alice:

Query:
SELECT ?x
WHERE { ?x ancestorOf Alice }

Outline

SPARQL 1.1

... some extensions in SPARQL 1.1:

- Projected expressions
- Aggregationen
- Subqueries
- Negation
- Property path
- Service descriptions
- Update language
- Update protocol
- HTTP RDF update (RESTful)
- Basic federated query

Projected expressions

- SELECT queries are no longer restricted to variables:

```
SELECT ( ?price * ?amount AS ?totalPrice)
WHERE { ... }
```

Aggregation

- Like in SQL: COUNT, SUM, MIN, MAX, GROUP BY

```
SELECT (MIN( ?price) AS ?minPrice) ...
WHERE { ... }
GROUP BY ?article
```

Subqueries

- nested queries
  - multiple queries can be combined in one query

```
SELECT ?article ?author
WHERE
   ?article ex:author ?author.
     { SELECT ?article
       WHERE { ... ?article ... }
       ORDER BY ...
       LIMIT ...
}
```

Result of a query is nested in another query.

Negation

- The trick with OPTIONAL and BOUND is no longer needed.

```
SELECT ...
WHERE { ?person a foaf:Person .
        MINUS { ?person foaf:mbox ?email }
}
```

Both graph patterns are evaluated and then the difference is returned.

Negation (2)

- Alternative construct NOT EXISTS

```
SELECT ...
WHERE { ?person a foaf:Person .
    { FILTER NOT EXISTS {
      ?person foaf:mbox ?email
      }
}
```

Both graph patterns are evaluated and then the difference is returned.

Property path

- Excerpt of constructors (let p be an IRI for a predicate, e.g., foaf:knows):
  - ^p inverse path
  - p$*$ multiple times (including 0)
  - p$+$ multiple times at least 1
  - p? zero or one times
  - p1/p2 sequence
  - p1|p2 alternative

Property path (2)

- "regular expression"

```
SELECT ...
WHERE {
     ?person foaf:knows+ ?person2 .
}
```

Property path (3)

```
SELECT ...
WHERE {
   ?book dc:title | rdfs:label ?displayString .
}
```

- "regular expression"
- e.g., alternative

Property path (3)

- "regular expression"
- e.g., Sequence:
  - "Find the names of people 2 foaf:knows links away."

```
SELECT ...
WHERE {
      ?x foaf:mbox <mailto:alice@example> .
      ?x foaf:knows/foaf:knows/foaf:name ?name .
}
```

$\rightarrow$ without property path?

Property path (4)

$\rightarrow$ equivalent query (without property path):

```
SELECT ...
WHERE {
      ?x foaf:mbox <mailto:alice@example> .
      ?x foaf:knows ?a1 .
      ?a1 foaf:knows ?a2 .
      ?a2 foaf:name ?name .
}
```

Outline

Summary

- SPARQL is the query language for RDF databases
- Query types:
  - SELECT
  - DESCRIBE
  - CONSTRUCT
  - ASK
- Query patterns are based on basic triple patterns
- FILTER, OPTIONAL, ...
- SPARQL 1.1
  - Projected expressions, aggregation, ...
  - Property paths

Pointers to further reading

- Segaran, Toby; Evans, Colin; Taylor, Jamie (2009). Programming the Semantic Web. O'Reilly Media. ISBN 978-0-596-15381-6.
- SPARQL 1.0 Specification: `http://www.w3.org/TR/rdf-sparql-query/`
- SPARQL 1.1 Specification: `http://www.w3.org/TR/sparql11-overview/`