SPARQL

Werner Nutt

Acknowledgment

These slides are based on the slide set

- Jena
  By Mariano Rodriguez (see http://www.slideshare.net/marianomx)
  (CC Attribution Share Alike 3.0 License)

which are based on the slide set

- An introduction to SPARQL and Queries over Linked Data.
  Chapter 2 SPARQL.
  By Olaf Hartig (CC Attribution Share Alike 3.0 License)

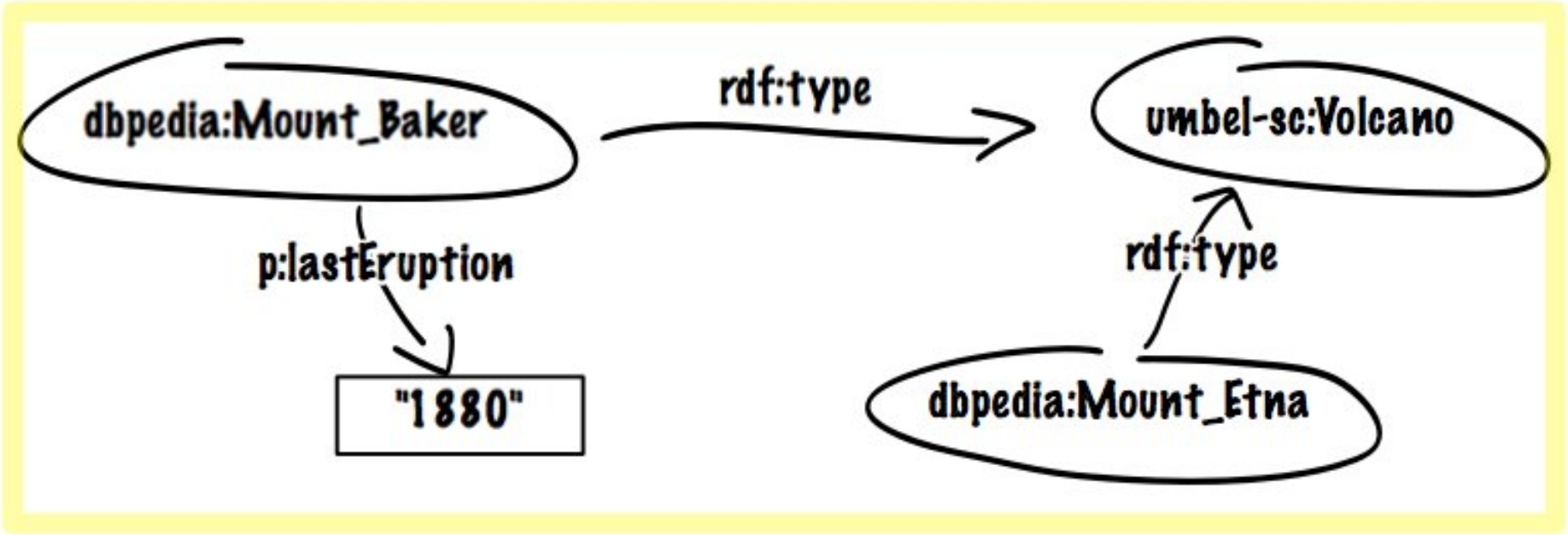
SPARQL in General

- A family of W3C recommendations
- SPARQL Query
  - Declarative query language for RDF data
- SPARQL Update
  - Declarative update language for RDF data
- SPARQL Protocol
  - Communication between SPARQL processing services (a.k.a. SPARQL endpoints) and clients
- SPARQL Query Results XML Format
  - XML Format for serializing query results

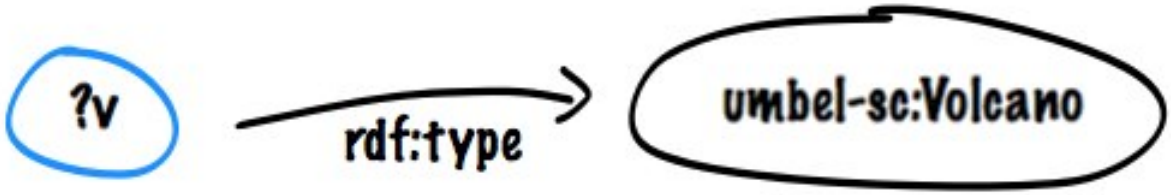
Idea of SPARQL queries

- Pattern matching
  - Describe subgraphs of the queried RDF graph
  - Subgraphs that match your description contribute an answer
  - Building blocks: graph patterns (i.e. RDF graphs with variables)

Idea of SPARQL queries



?v
dbpedia:Mount_Baker
dbpedia:Mount_Etna



Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Prologue**
  - Prefix definitions for compact URIs
  - Attention (difference w.r.t. Turtle):
    No period (“.”) character as separator

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Result form specification**
  - SELECT, DESCRIBE, CONSTRUCT or ASK (more later)

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Dataset specification**
  - Specify the RDF dataset to be queries (more on that later)

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Query pattern**
  - WHERE clause specifies the graph pattern to be matched

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Solution modifiers**
  - ORDER BY, LIMIT, OFFSET
    (more about that later)
  - Only available in some of the result forms

Graph patterns

- Different types of graph patterns for the query pattern (WHERE clause):
  - Basic graph pattern (BGP)
  - Group graph pattern
  - Optional graph pattern
  - Union graph pattern
  - Named Graph pattern
  - (Constraints)

Basic Graph Patterns

- Set of triple patterns in Turtle syntax (i.e., RDF triples with variables)
- Variable names prefixed with “?” or “$”

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?name
WHERE {
  ?v rdf:type umbel-sc:Volcano .
  ?v rdfs:label ?name
}
```

Basic Graph Patterns

- Set of triple patterns in Turtle syntax (i.e., RDF triples with variables)
- Variable names prefixed with “?” or “$”

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?name
WHERE {
  ?v rdf:type umbel-sc:Volcano ;
      rdfs:label ?name
}
```

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- What are the names of all (known) volcanos?

```
SELECT ?name WHERE {
  ?v rdf:type umbel-sc:Volcano ;
      rdfs:label ?name
}
```

?name
“Etna”
"Беренберг"@ru
"Beerenberg"@en

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- List all types of the volcano called “Beerenberg”

```
SELECT ?type WHERE {
  ?v rdf:type ?type ;
      rdfs:label “Beerenberg”
}
```

?type

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- List all types of the volcano called “Beerenberg”

```
SELECT ?type WHERE {
  ?v rdf:type ?type ;
     rdfs:label “Beerenberg”@en
}
```

?type
umbel-sc:Volcano
umbel-sc:NaturalElevation

Example

Data

dbpedia:Mount_Baker	rdf:type	umbel-sc:Volcano;
	p:location	dbpedia:United_States.
dbpedia:United_States	rdfs:label	"United States"

- Where are all (known) volcanos located? (List the names of these locations)

- Blank nodes in SPARQL queries
  - As subject or object of triple patterns
  - Non-selectable variables

```
SELECT ?name WHERE {
_:x     rdf:type      umbel-sc:Volcano ;
        p:location [ rdfs:label ?name ] .
}
```

?name
"United States"

Example

Data

```
dbpedia:Mount_ Baker    rdf:type umbel-sc:Volcano ;
                        p:location [ rdfs:label      "United States"@en ,
                                                     "États-Unis"@fr ] .
dbpedia:Mount_ Etna     rdf:type umbel-sc:Volcano ;
                        p:location [ rdfs:label "Italy" ] .
```

- Blank nodes in the queried data
  - Blank node identifiers may appear in the results

```
SELECT ?l ?name WHERE {
  ?v rdf:type    umbel-sc:Volcano ;
       p:location ?l .
   ?l  rdfs:label  ?name .
}
```

?l	?name
_:n0	“United States”@en
_:n0	"États-Unis"@fr
_:n1	"Italy"

Optional Graph Patterns

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en .
```

- What are **all** volcanos and their names?

```
SELECT ?v ?name WHERE {
  ?v rdf:type umbel-sc:Volcano ;
     rdfs:label ?name .
}
```

?v	?name
dbpedia:Mount_Etna	“Etna”
dbpedia:Beerenberg	"Beerenberg"@en

- Problem: Missing Mount Baker (it has no name)

Optional Graph Patterns

```
SELECT ?v ?name WHERE {
  ?v rdf:type umbel-sc:Volcano .
  OPTIONAL { ?v rdfs:label ?name }
}
```

?v	?name
dbpedia:Mount_Etna	“Etna”
dbpedia:Mount_Baker	
dbpedia:Beerenber	"Beerenberg"@en

- Optional patterns may result in unbound variables

Union Graph Patterns

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" ;
                        p:location      dbpedia:Italy .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano ;
                        p:location      dbpedia:United_States.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                        rdfs:label      "Beerenberg"@en ;
                        p:location      dbpedia:Norway .
```

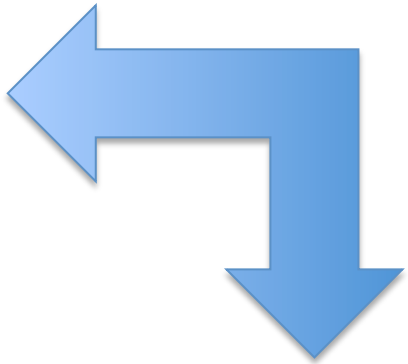
- List all volcanos located in Italy or Norway

```
SELECT ?v WHERE {
  ?v rdf:type umbel-sc:Volcano;
     p:location  ???????
}
```

Union Graph Patterns

```
SELECT ?v WHERE {
  ?v rdf:type umbel-sc:Volcano.
      { ?v p:location dbpedia:Italy }
      UNION
      {  ?v p:location dbpedia:Norway }
}
```

semantically equivalent



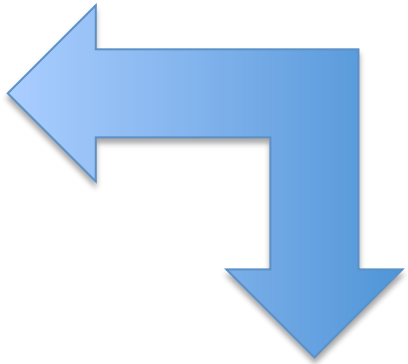
```
SELECT ?v WHERE {
  { ?v rdf:type umbel-sc:Volcano;  p:location dbpedia:Italy }
  UNION
  {?v rdf:type umbel-sc:Volcano;  p:location dbpedia:Norway.  }
}
```

Union Graph Patterns

```
SELECT ?v WHERE {
  ?v rdf:type umbel-sc:Volcano.
      { ?v p:location dbpedia:Italy }
      UNION
      {  ?v p:location dbpedia:Norway }

}
```

semantically equivalent



```
SELECT ?v WHERE {
  { ?v rdf:type umbel-sc:Volcano.  } {
      { ?v p:location dbpedia:Italy }
      UNION
      {  ?v p:location dbpedia:Norway }
    }
}
```

Constraints on Solutions

- Syntax: Keyword `FILTER` followed by a filter expression

```
SELECT ?v WHERE {
  ?v        rdf:type umbel-sc:Volcano ;
            p:lastEruption ?le
  FILTER ( ?le > 1900 )
}
```

- Filter expressions are formed by operators and functions

- Operators and functions operate on RDF terms

Constraints (Truth Table)

- Filter expressions evaluate to true, false or error
- Truth table:

A	B	A || B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

Unary Operators in Constraints

Operator	Type(A)	Result type
!A	xsd:boolean	xsd:boolean
+A	numeric	numeric
-A	numeric	numeric
BOUND(A)	variable	xsd:boolean
isURI(A)	RDF term	xsd:boolean
isBLANK(A)	RDF term	xsd:boolean
isLITERAL(A)	RDF term	xsd:boolean
STR(A)	literal / URI	simple literal
LANG(A)	literal	simple literal
DATATYPE(A)	literal	simple literal

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- List all types of the volcano called “Beerenberg”

```
SELECT ?type WHERE {
  ?v rdf:type ?type ;
      rdfs:label “Beerenberg”
}
```

?type

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- List all types of the volcano called “Beerenberg”

```
SELECT ?type WHERE {
  ?v rdf:type ?type ;
      rdfs:label ?name .
      FILTER( STR(?name) = “Beerenberg)
}
```

?type
umbel-sc:Volcano
umbel-sc:NaturalElevation

Constraints (Further operators)

- Binary operators:
  - Logical connectives **&&** and **||** **for** xsd:boolean
  - Comparison operators **=, !=, <, >, <=,** and **>=** for numeric datatypes, xsd:dateTime, xsd:string, and xsd:boolean
  - Comparison operators **=** and **!=** for other datatypes
  - Arithmetic operators **+, -,** *, and **/** for numeric datatypes
- Furthermore:
  - **REGEX**(String,Pattern) or REGEX(String,Pattern,Flags)
  - **sameTERM**(A,B)
  - **langMATCHES**(A,B)

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- List all types of the volcano with “e” in their name

```
SELECT ?v WHERE {
  ?v rdf:type umbel-sc:Volcano ;
     rdfs:label ?name .
     FILTER( REGEX(STR(?name), "e") )
}
```

?v
dbpedia:Beerenberg
dbpedia:Beerenberg

Example

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- List all types of the volcano with “e” in their name (case insensitive)

```
SELECT ?v WHERE {
  ?v rdf:type umbel-sc:Volcano ;
     rdfs:label ?name .
     FILTER( REGEX(STR(?name), "e", "i") )
}
```

?v
dbpedia:Mount_Etna
dbpedia:Beerenberg
dbpedia:Beerenberg

Named Graph Patterns

Default graph

```
dbpedia:Mount_Etna      rdfs:seeAlso <http://example.org/d1>
dbpedia:Mount_Baker     rdfs:seeAlso <http://example.org/d2>
```

http://example.org/d1

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
```

http://example.org/d2

```
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
```

http://example.org/d3

```
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Beerenberg"@en ;
```

- SPARQL queries are executed over an **RDF dataset:**
  - One **default graph** and
  - Zero or more **named graphs** (identified by an URI)

Graphs in RDF

- Currently no official specification of named graphs in RDF
- However, most RDF triple stores support them
  - Loading data into graphs in Jena, use “ng4j”

Named Graphs in Jena (Example)

```
// Create a new graphset
NamedGraphSet graphset = new NamedGraphSetImpl();
NamedGraph graph = graphset.createGraph("http://example.org/persons/123");

// Add information to the NamedGraph
graph.add(new Triple(Node.createURI("http://fariz.darari.it/
                                                    foaf.rdf#FarizDarari"),
      Node.createURI("http://xmlns.com/foaf/0.1/name") ,
      Node.createLiteral("FarizDarari", null, null)));

// Create a quad
Quad quad = new Quad(Node.createURI("http://www.werner.de/
            InformationAboutFariz"),
      Node.createURI("http://fariz.darari.it/foaf.rdf#FarizDarari"),
      Node.createURI("http://xmlns.com/foaf/0.1/mbox") ,
      Node.createURI("mailto:fariz@darari.it"));

// Add the quad to the graphset. This will create a new NamedGraph in the
// graphset.
graphset.addQuad(quad);
```

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Dataset specification**
  - Specify the RDF dataset to be queried (more on that later)
- Specification using FROM and FROM NAMED

Named Graph Patterns

Default graph

```
dbpedia:Mount_Etna    rdfs:seeAlso <http://example.org/d1>
dbpedia:Mount_Baker   rdfs:seeAlso <http://example.org/d2>
```

http://example.org/d1

```
dbpedia:Mount_Etna    rdf:type     umbel-sc:Volcano ;
                      rdfs:label   "Etna" .
```

http://example.org/d2

```
dbpedia:Mount_Baker   rdf:type     umbel-sc:Volcano.
```

http://example.org/d3

```
dbpedia:Beerenberg    rdf:type     umbel-sc:Volcano ;
                      rdfs:label   "Beerenberg"@en ;
```

- Evaluation of patterns w.r.t. **the active graph**
- GRAPH clause for making a named graph the active graph

Named Graph Patterns

Default graph

```
dbpedia:Mount_Etna      rdfs:seeAlso <http://example.org/d1>
dbpedia:Mount_Baker     rdfs:seeAlso <http://example.org/d2>
```

http://example.org/d1

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
```

http://example.org/d2

```
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
```

http://example.org/d3

```
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Beerenberg"@en ;
```

```
SELECT ?v WHERE {
  GRAPH ?g {
       ?v rdf:type umbel-sc:Volcano }
}
```

?v
dbpedia:Mount_Etna
dbpedia:Mount_Baker
dbpedia:Beerenberg

Named Graph Patterns

Default graph

```
dbpedia:Mount_Etna      rdfs:seeAlso <http://example.org/d1>
dbpedia:Mount_Baker     rdfs:seeAlso <http://example.org/d2>
```

http://example.org/d1

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
```

http://example.org/d2

```
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
```

http://example.org/d3

```
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Beerenberg"@en ;
```

```
SELECT ?v ?g WHERE {
 GRAPH ?g {
      ?v rdf:type umbel-sc:Vo
}
```

?v	?g
dbpedia:Mount_Etna	<http://example.org/d1>
dbpedia:Mount_Baker	<http://example.org/d2>
dbpedia:Beerenberg	<http://example.org/d3>

Named Graph Patterns

Default graph

```
dbpedia:Mount_Etna      rdfs:seeAlso <http://example.org/d1>
dbpedia:Mount_Baker     rdfs:seeAlso <http://example.org/d2>
```

http://example.org/d1

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
```

http://example.org/d2

```
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
```

http://example.org/d3

```
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Beerenberg"@en ;
```

```
SELECT ?v  WHERE {
  _:x rdfs:seeAlso ?g GRAPH ?g {
        ?v rdf:type umbel-sc:Volcano }
}
```

?v
dbpedia:Mount_Etna
dbpedia:Mount_Baker

Negation

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- What volcanos do **not** have a name in our data?

```
SELECT ?v WHERE {
  ?v  rdf:type   umbel-sc:Volcano .
  OPTIONAL { ?v rdfs:label ?name }
  FILTER ( ! BOUND(?name) )
}
```

?v
dbpedia:Mount_Baker

Negation as failure

Negation

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- What volcanos are **not called** “Beerenberg”?

```
SELECT ?v WHERE {
  ?v   rdf:type      umbel-sc:Volcano ;
       rdfs:label    ?name .
  FILTER ( STR(?name) != “Beerenberg” ) }
}
```

?name
dbpedia:M[illegible]Etna
dbpedia:[illegible]g

Negation

Data

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- What volcanos are **not called** “Beerenberg”?

```
SELECT ?v WHERE {
 ?v  rdf:type    umbel-sc:Volcano .
 OPTIONAL {
          ?v  rdfs:label ?name .
          FILTER ( STR(?name) = “Beerenberg” )
          }
 FILTER ( ! BOUND (?name) )
}
```

?name
dbpedia:Mount_Etna
dbpedia:Mount_Baker

Negation as failure

Named Graph Patterns and Negation

Default graph

```
dbpedia:Mount_Etna    rdfs:seeAlso <http://example.org/d1>
dbpedia:Mount_Baker   rdfs:seeAlso <http://example.org/d2>
```

http://example.org/d1

```
dbpedia:Mount_Etna    rdf:type      umbel-sc:Volcano ;
                      rdfs:label    "Etna" .
```

http://example.org/d2

```
dbpedia:Mount_Baker   rdf:type      umbel-sc:Volcano.
```

http://example.org/d3

```
nbel-sc:Volcano ;
eerenberg"@en ;
```

```
SELECT ?g WHERE {
  GRAPH ?g {
     ?v    rdf:type umble-sc:Volcano ;
           rdfs:label ?name .
  }
  OPTIONAL { ?v rdfs:seeAlso ?r }
  FILTER ( ! BOUND(?r) )
}
```

name of a volcano

ult graph

Summary – Graph Patterns

- **Different types of graph patterns for the query pattern (WHERE clause):**
  - Basic graph pattern (BGP)
  - Group graph pattern
  - Optional graph pattern – keyword OPTIONAL
  - Union graph pattern – keyword UNION
  - Named Graph pattern – keyword GRAPH
  - Constraints – keyword FILTER

Components of SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Result form specification**
  - SELECT, DESCRIBE, CONSTRUCT or ASK (more later)

Result Forms

- SELECT
  - Result: sequence of solutions (i.e. sets of variable bindings)
  - Selected variables separated by space (not by comma!)
  - Asterisk character (“*”) selects all variables in the pattern

- ASK
  - Check whether there is at least one result
  - Result: true or false
  - Example: Do we hav

```
ASK WHERE {
    ?v rdf:type umble-sc:Volcano
}
```

Result Forms

- DESCRIBE
  - Result: an RDF graph with data about resources
  - Non-deterministic (i.e. query processor defines the actual structure of the resulting RDF graph)
  - Example: just name the resource

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

  - Possible to specify the resource by query pattern

```
DESCRIBE ?v WHERE {
        ?v rdf:type umbel-sc:Volcano .
}
```

Result Forms

- CONSTRUCT
  - Result: an RDF graph constructed from a template
  - Template: graph pattern with variables from the query pattern

```
CONSTRUCT {
   ?v     rdfs:label ?name ;
          rdf:type myTypes:VolcanosOutsideTheUS
} WHERE {
          ?v rdf:type umbel-sc:Volcano ;
          rdfs:label ?name .
 OPTIONAL { ?v p:location ?l
          FILTER ( ?l = dbpedia:United_States ) }
 FILTER ( ! BOUND(?l) )
}
```

Result Forms

Data

```
dbpedia:Mount_Etna      rdf:type          umbel-sc:Volcano ;
                        rdfs:label        "Etna" ;
                        p:location        dbpedia:Italy .
dbpedia:Mount_Baker     rdf:type          umbel-sc:Volcano ;
                        rdfs:label        "Mount Baker" ;
                        p:location        dbpedia:United_States.
dbpedia:Beerenberg      rdf:type          umbel-sc:Volcano,
                        rdfs:label        "Beerenberg"@en ;
                        p:location        dbpedia:Norway .
```

```
dbpedia:Mount_Etna      rdfs:label        "Etna" ;
                        rdf:type myTypes:VolcanosOutsideTheUS
dbpedia:Beerenberg      rdfs:label "Beerenberg"@en ;
                        rdf:type myTypes:VolcanosOutsideTheUS
```

Result

Solution Modifiers

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?name
```

- **Solution modifiers**
  - Only for SELECT queries
  - Modify the result set as a whole (not single solutions)
  - Keywords: DISTINCT, ORDER BY, LIMIT, OFFSET

Solution Modifiers

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- DISTINCT removes duplicates from the result set

```
SELECT ?type
WHERE {
  _:x rdf:type ?type
}
```

?name
umbel-sc:Volcano
umbel-sc:Volcano
umbel-sc:NaturalElevation
umbel-sc:Volcano

Solution Modifiers

```
dbpedia:Mount_Etna      rdf:type        umbel-sc:Volcano ;
                        rdfs:label      "Etna" .
dbpedia:Mount_Baker     rdf:type        umbel-sc:Volcano.
dbpedia:Beerenberg      rdf:type        umbel-sc:Volcano,
                                        umbel-sc:NaturalElevation ;
                        rdfs:label      "Beerenberg"@en ;
                        rdfs:label      "Беренберг"@ru  .
```

- DISTINCT removes duplicates from the result set

```
SELECT DISTINCT ?type
WHERE {
  _:x rdf:type ?type
}
```

?name
umbel-sc:Volcano
umbel-sc:NaturalElevation

Solution Modifiers

- ORDER BY orders the results

```
SELECT ?v WHERE { ?v rdf:type umbel-sc:Volcano ;
                  rdfs:label ?name }
ORDER BY ?name
```

- How do you order different kinds of elements?
  - unbound < blank node < URI < literal
- ASC for ascending (default) and DESC for descending
- Hierarchical order criteria

```
SELECT ?name WHERE { ?v rdf:type umbel-sc:Volcano ;
                        p:lastEruption ?le ;
                        rdfs:label ?name }
ORDER BY DESC(?le), ?name
```

Solution Modifiers

- LIMIT – limits the number of result

```
SELECT ?v WHERE { ?v rdf:type umbel-sc:Volcano ;
                  rdfs:label ?name }
ORDER BY ?name LIMIT 5
```

- OFFSET – position/index of the first reported results

```
SELECT ?v WHERE { ?v rdf:type umbel-sc:Volcano ;
                  rdfs:label ?name }
ORDER BY ?name LIMIT 5 OFFSET 10
```

- Order of the result should be predictable (combine with ORDER BY)

Exercise

Consider the example data set:

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

ex:Hamlet            ex:author    ex:Shakespeare ;
                     ex:price     "10.50"^^xsd:decimal .
ex:Macbeth           ex:author    ex:Shakespeare .
ex:Tamburlaine       ex:author    ex:Marlowe ;
                     ex:price     "17"^^xsd:integer .
ex:DoctorFaustus     ex:author    ex:Marlowe ;
                     ex:price     "12"^^xsd:integer ; ;
       ex:title  "The Tragical History of Doctor Faustus" .
ex:RomeusJuliet      ex:author    ex:Brooke ;
                     ex:price     ”12"^^xsd:integer .
```

Exercise

Formulate SPARQL queries for the vocabulary in the data set:

- Who are the authors of some book?
- Who are the authors of a book that costs less than 15?
- Who are the authors of a book that costs less than 15, and what are the titles, if available?
- Which is the most expensive book?
- Which is the most expensive book by Shakespeare?

SPARQL in Jena

- What components would you expect in a framework like Jena?

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  http://jena.apache.org/documentation/javadoc/arq/

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query
  - `QueryFactory` for creating queries in various ways
  - `QueryExecution` for the execution state of a query
  - `QueryExecutionFactory` for creating query executions
  - `ResultSet` for results of a SELECT
- `CONSTRUCT` and `DESCRIBE` return Models, `ASK` a Java boolean.
- SPARQL with ARQ is generally slow because queries are broken down into triple patterns.

Constructing a Query and a QueryExecution

Query objects are usually constructed by parsing:

```
String qStr =
        "PREFIX foaf:  <" + foafNS + ">"
      + "SELECT ?a ?b WHERE {"
      + "  ?a foaf:knows ?b ."
      + "} ORDER BY ?a ?b";
Query q = QueryFactory.create(qStr);
```

- Programming interface deprecated and badly documented
- A Query can be used several times, on multiple models
- For each execution, a new QueryExecution is needed
- To produce a QueryExecution for a given Query and Model:

```
QueryExecution qe =
        QueryExecutionFactory.create(q, model);
```

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)
- E.g. for a SELECT query:

```
ResultSet res = qe.execSelect();
```

- `ResultSet` is a sub-interface of

```
Iterator<QuerySolution>
```

- Also has methods to get list of variables
- `QuerySolution` has methods to get list of variables, value of single variables, etc.
- Important to call `close()` on query executions when no longer needed *(Why?)*

Example: SPARQL in Jena

```
String qStr = "SELECT ?a ?b ...";
Query q = QueryFactory.create(qStr);
QueryExecution qe =
        QueryExecutionFactory.create(q,model);
try {
        res = qe.execSelect();
        while( res.hasNext()) {
                QuerySolution soln = res.next();
                RDFNode a = soln.get("?a");
                RDFNode b = soln.get("?b");
                System.out.println(""+a+" knows "+b);
}
} finally {
        qe.close();
}
```

SPARQL on the Web

- Many sites (DBLP, dbpedia, dbtunes,. . .) publish SPARQL endpoints
- I.e. SPARQL queries can be submitted to a database server that sends back the results
- Uses HTTP to submit URL-encoded queries to server

  `GET /sparql/?query=... HTTP/1.1`
- Actually defined via W3C Web Services, see

  http://www.w3.org/TR/rdf-sparql-protocol/
- For SELECT queries you can get a XML or JSON result set, see

  `http://www.w3.org/TR/rdf-sparql-XMLres/`

  `http://www.w3.org/TR/rdf-sparql-json-res/`
- Nothing you would want to do manually!

Remote SPARQL with Jena

- Jena can send SPARQL queries to a remote endpoint!
- Use one of the sparqlService methodes in QueryExecutionFactory
- E.g.

```
String endpoint = "http://dblp.l3s.de/d2r/sparql";
String qStr = "SELECT ?a ?b ...";
Query q = QueryFactory.create(qStr);
QueryExecution qe =
   QueryExecutionFactory.sparqlService(endpoint,q);
try {
      res = qe.execSelect();
      ...
} finally {
qe.close();
}
```