Semantic Web Technologies

Topic: SPARQL

Olaf Hartig
olaf.hartig@liu.se
SPARQL Family of Standards

- Family of W3C recommendations related to querying RDF data
- **SPARQL Query Language**
  - Declarative query language for RDF data
  - Our focus now
- **SPARQL Update**
  - Declarative update language for RDF data
- **SPARQL Protocol**
  - Communication between SPARQL processing services (a.k.a. SPARQL endpoints) and clients
- and more
  - e.g., formats for serializing query results
RDF in a Nutshell

- Data comes as a set of **triples** (subject, predicate, object)
  - **Subject**: URI or blank node
  - **Predicate**: URI
  - **Object**: URI, blank node, or literal

```sparql
@prefix dbpedia : <http://dbpedia.org/resource/> .
@prefix p : <http://dbpedia.org/property/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbpedia:Mount_Baker p:lastEruption "1880"^^xsd:integer ;
```

- Example:
  - p:name "Mount Baker"@en
  - p:lastEruption "1880"^^xsd:integer
Main Idea of SPARQL Queries
Main Idea of SPARQL Queries

- **Pattern matching:**
  - Describe subgraphs of the queried RDF graph
  - Subgraphs that match the description yield a result
  - Mean: **graph patterns** (essentially, RDF graphs with variables)
Main Idea of SPARQL Queries

Queried RDF graph:

```
dbpedia:Mount_Baker rdfs:subClassOf umbel-sc:Volcano.
p:lastEruption <1880> dbpedia:Mount_Etna.
```

Result:
```
?v rdfs:subClassOf umbel-sc:Volcano.

```
Another Example

Data

http://dbpedia.org/resource/Washington

http://.../location

http://dbpedia.org/resource/Mount_Baker

http://.../lastEruption

1880

Query

http://dbpedia.org/resource/Washington

http://.../location

http://.../lastEruption

Query Result:

<table>
<thead>
<tr>
<th>?x</th>
<th>?e</th>
</tr>
</thead>
<tbody>
<tr>
<td>http://.../Mount_Baker</td>
<td>1880</td>
</tr>
</tbody>
</table>
Components of SPARQL Queries
Components of a SPARQL Query

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 ?v
Components of a SPARQL Query

```sparql
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 ?v
```

• Prologue:
  • **Prefix definitions** for using compact URIs (CURIEs)
Components of a SPARQL Query

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 ?v

• Result form specification:
  • SELECT for projection (similar to SELECT in SQL)
  • Other forms: DESCRIBE, CONSTRUCT, and ASK (more about these later)
Components of a SPARQL Query

```sparql
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 ?v
```

- Dataset specification:
  - Specify the RDF dataset to be queried (use URIs that identify particular RDF graphs in your RDF database)
Components of a SPARQL Query

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 ?v

• Query Pattern:
  • WHERE clause specifies the graph pattern to be matched
Components of a SPARQL Query

```
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 ?v
```

- Solution modifiers (for SELECT queries):
  - Modify the result set as a whole (not single solutions)
  - Keywords: DISTINCT, ORDER BY, LIMIT, and OFFSET
Simple Types of Graph Patterns
Basic Graph Pattern

- Set of triple patterns (i.e., RDF triples with variables)
- Variable names prefixed with “?” (or “$”)
- Turtle syntax
  - Including syntactic sugar (e.g., property and object lists)

```sparql
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 .
}
```
Basic Graph Pattern (Example)

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

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 "Бееренберг"@ru .

• Question: What are the names of all (known) volcanos?

Result:

?name
"Etna"
"Бееренберг"@ru
"Beerenberg"@en
Optional Graph Pattern

- **Problem:** Mount Baker missing (b/c no name in the data)

- **Question:** What are all (known) volcanos and their names?

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

- **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 ;
  rdfs:label "Beerenberg"@en .
```

- **Example Output**

<table>
<thead>
<tr>
<th>?v</th>
<th>?name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbpedia:Mount_Etna</td>
<td>&quot;Etna&quot;</td>
</tr>
<tr>
<td>dbpedia:Beerenberg</td>
<td>&quot;Beerenberg&quot;@en</td>
</tr>
</tbody>
</table>
Optional Graph Pattern

- Keyword OPTIONAL indicates optional patterns

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

<table>
<thead>
<tr>
<th>?v</th>
<th>?name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbpedia:Mount_Etna</td>
<td>&quot;Etna&quot;</td>
</tr>
<tr>
<td>dbpedia:Mount_Baker</td>
<td>&quot;Beerenberg&quot; @en</td>
</tr>
<tr>
<td>dbpedia:Beerenberg</td>
<td></td>
</tr>
</tbody>
</table>

- Optional patterns may result in unbound variables
UNION Clauses

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 .

Question: Which volcanos are located in the Italy or in Norway?

Query

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

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

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

• Syntax: Keyword FILTER followed by filter expression

```sparql
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
PREFIX p: <http://dbpedia.org/property/>

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

• Filter expressions contain operators and functions
## Unary Operators in Constraints

<table>
<thead>
<tr>
<th>Operator</th>
<th>Type(A)</th>
<th>Result type</th>
</tr>
</thead>
<tbody>
<tr>
<td>! A</td>
<td>xsd:boolean</td>
<td>xsd:boolean</td>
</tr>
<tr>
<td>+ A</td>
<td>numeric</td>
<td>numeric</td>
</tr>
<tr>
<td>- A</td>
<td>numeric</td>
<td>numeric</td>
</tr>
<tr>
<td>BOUND(A)</td>
<td>variable</td>
<td>xsd:boolean</td>
</tr>
<tr>
<td>isURI(A)</td>
<td>RDF term</td>
<td>xsd:boolean</td>
</tr>
<tr>
<td>isBLANK(A)</td>
<td>RDF term</td>
<td>xsd:boolean</td>
</tr>
<tr>
<td>isLITERAL(A)</td>
<td>RDF term</td>
<td>xsd:boolean</td>
</tr>
<tr>
<td>STR(A)</td>
<td>literal / URI</td>
<td>simple literal</td>
</tr>
<tr>
<td>LANG(A)</td>
<td>literal</td>
<td>simple literal</td>
</tr>
<tr>
<td>DATATYPE(A)</td>
<td>literal</td>
<td>simple literal</td>
</tr>
</tbody>
</table>
Binary and Other Operators

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

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

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

• **Question:** What volcanos have an “e” in their name?

Data

```
?v
```

```
dbpedia:Beerenberg
```

```
dbpedia:Mount_Etna
```

```
dbpedia:Beerenberg rdf:type umbel-sc:Volcano,
              umbel-sc:NaturalElevation ;
  rdfs:label "Beerenberg"@en ;
  rdfs:label "Бееренберг"@ru .
```
Constraints (Example, cont’d)

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

● **Question:** What volcanos have an “e” in their name?

```sparql
SELECT ?v WHERE {
    FILTER( REGEX(STR(?name),"e","i") )
}
```

?v
dbpedia:Mount_Etna
dbpedia:Beerenberg
Querying RDF Datasets

that contain Named Graphs
Idea

- SPARQL queries are evaluated over an RDF dataset
- An **RDF dataset** consists of:
  - one **default graph** and
  - zero or more **named graphs** (identified by an URI)
- Keyword **GRAPH** makes one of the named graphs the **active graph** used for pattern matching
Examples of GRAPH Clauses

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

- dbpedia:Mount_Etna http://example.org/d1  
  rdf:type umbel-sc:Volcano ;  
  rdfs:label "Etna" .

- dbpedia:Mount_Baker http://example.org/d2  
  rdf:type umbel-sc:Volcano .

- dbpedia:Beerenberg http://example.org/d3  
  rdf:type umbel-sc:Volcano ;  
  rdfs:label "Beerenberg"@en .
Examples of GRAPH Clauses

SELECT ?v WHERE {
  GRAPH <http://example.org/d1> {
    ?v rdf:type umbel-sc:Volcano .
  }
}

dbpedia:Mount_Etna rdfs:seeAlso <http://example.org/d1>.
dbpedia:Mount_Baker rdfs:seeAlso <http://example.org/d2>.
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 ; rdfs:label "Beerenberg"@en .
Examples of GRAPH Clauses

dbpedia:Mount_Etna rdfs:seeAlso <http://example.org/d1>.
dbpedia:Mount_Baker rdfs:seeAlso <http://example.org/d2>.
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 ; rdfs:label "Beerenberg"@en .

Default Graph

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

http://example.org/d1

http://example.org/d2

http://example.org/d3
Examples of GRAPH Clauses

```
SELECT ?g ?v WHERE {
  GRAPH ?g {
    ?v rdf:type umbel-sc:Volcano ;
    rdfs:label "Etna" .
  }
}
```

```
dbpedia:Mount_Etna rdfs:seeAlso <http://example.org/d1>.  
dbpedia:Mount_Baker rdfs:seeAlso <http://example.org/d2>.  
dbpedia:Beerenberg rdf:type umbel-sc:Volcano ;
  rdfs:label "Beerenberg"@en .
```
Examples of GRAPH Clauses

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

Default Graph

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

dbpedia:Mount_Etna http://example.org/d1
  rdf:type umbel-sc:Volcano ;
  rdfs:label "Etna" .

dbpedia:Mount_Baker http://example.org/d2
  rdf:type umbel-sc:Volcano .

dbpedia:Beerenberg http://example.org/d3
  rdf:type umbel-sc:Volcano ;
  rdfs:label "Beerenberg"@en .
```
Solution Modifiers
DISTINCT

Removes duplicates from the result set

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

```
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 "Бееренберг"@ru .
```
DISTINCT

Removes duplicates from the result set

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

umbel-sc:Volcano
umbel-sc:NaturalElevation
ORDER BY

Specify an order for the returned solution mappings

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

Order for different kinds of elements:

- unbound variable < 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
```
LIMIT and OFFSET

**LIMIT** – limits the number of solution mappings returned

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

**OFFSET** – position/index of the first returned mapping
(useful only in combination with ORDER BY)

```
SELECT ?name WHERE { ?v rdf:type umbel-sc:Volcano ;
                      rdfs:label ?name }
ORDER BY ?name
LIMIT 5 OFFSET 10
```
Result Forms
Components of a SPARQL Query

```sparql
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 ?v
```

- Result form specification:
  - SELECT for projection (similar to SELECT in SQL)
  - Other forms: DESCRIBE, CONSTRUCT, and ASK
    
    (more about these later) now ...

Semantic Web Technologies – Topic: SPARQL
Olaf Hartig
SELECT and ASK

SELECT

Sequence of solutions mappings
Selected variables separated by space (not by comma!)
Asterisk (“*”) character selects all variables in the pattern

ASK

Check if there is at least one solution mapping
Example: Do we have any volcanos?

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

Returns an RDF graph with data about resources

Not deterministic (i.e. query processor determines the actual structure of the returned RDF graph)

Name the resource:

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

Specify the resource with a query pattern:

DESCRIBE ?v WHERE {
  FILTER ( STR(?name) = "Beerenberg" )
}

Multiple variables possible or asterisk ("*") for all
CONSTRUCT

Returns an RDF graph created from a template
Template: graph pattern with variables from the query pattern

```sparql
CONSTRUCT { ?v rdfs:label ?name ;
    rdf:type myTypes:VolcanoInTheUS }
WHERE {
    ?v rdf:type umbel-sc:Volcano ;
    rdfs:label ?name ;
    p:location ?l .
    FILTER ( ?l = dbpedia:United_States )
}
```
SPARQL Version 1.1
SPARQL 1.1

- New features of SPARQL 1.1 Query:
  - Aggregate functions (e.g., COUNT, SUM, AVG)
  - Sub-queries
  - Negation (EXISTS, NOT EXISTS, MINUS)
  - Assignments (e.g., BIND, SELECT expressions)
  - Property paths (navigation à la XPath)
  - Basic query federation (SERVICE, VALUES)

- SPARQL 1.1 Update:
  - Graph update (INSERT DATA, DELETE DATA, INSERT, DELETE, DELETE WHERE, LOAD, CLEAR)
  - Graph management (CREATE, DROP, COPY, MOVE, ADD)