Lecture 4
Querying RDF with SPARQL

Grigoris Antoniou-
Sotiris Batsakis

SPARQL Basic Queries

- SPARQL is based on matching graph patterns
- The simplest graph pattern is the triple pattern :
  - like an RDF triple, but with the possibility of a variable instead of an RDF term in the subject, predicate, or object positions
- Combining triple patterns gives a basic graph pattern, where an exact match to a graph is needed to fulfill a pattern

Examples

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?c
WHERE
{
    ?c rdf:type rdfs:Class .
}
```

- Retrieves all triple patterns, where:

-the property is rdf:type

-the object is rdfs:Class

- Which means that it retrieves all classes

Examples (2)

- Get all instances of a particular class (e.g. course) :

(declaration of rdf, rdfs prefixes omitted for brevity)

```
PREFIX uni: <http://www.mydomain.org/uni-ns#>
SELECT ?i
WHERE
{
    ?i rdf:type uni:course .
}
```

Using select-from-where

- As in SQL, SPARQL queries have a SELECT-FROM-WHERE structure:
  - **SELECT** specifies the projection: the number and order of retrieved data
  - **FROM** is used to specify the source being queried (optional)
  - **WHERE** imposes constraints on possible solutions in the form of graph pattern templates and boolean constraints
- Retrieve all phone numbers of staff members:

```
SELECT ?x ?y
WHERE
{ ?x uni:phone ?y .}
```

- Here **?x** and **?y** are variables, and **?x uni:phone ?y** represents a resource-property-value triple pattern

Implicit Join

- Retrieve all lecturers and their phone numbers:

```
SELECT ?x ?y
WHERE
{ ?x rdf:type uni:Lecturer ;
        uni:phone ?y . }
```

- Implicit join: We restrict the second pattern only to those triples, the resource of which is in the variable **?x**
  - Here we use a syntax shorcut as well: a semicolon indicates that the following triple shares its subject with the previous one

Implicit join (2)

- The previous query is equivalent to writing:

```
SELECT ?x ?y
WHERE
{
   ?x rdf:type uni:Lecturer .
   ?x uni:phone ?y .
}
```

Explicit Join

- Retrieve the name of all courses taught by the lecturer with ID 949352

```
SELECT ?n
WHERE
{
   ?x rdf:type uni:Course ;
         uni:isTaughtBy :949352 .
   ?c uni:name ?n .
   FILTER (?c = ?x) .
}
```

Optional Patterns

```
<uni:lecturer rdf:about="949352">
    <uni:name>Grigoris Antoniou</uni:name>
</uni:lecturer>
<uni:lecturer rdf:about="94318">
    <uni:name>David Billington</uni:name>
    <uni:email>david@work.example.org</uni:email>
</uni:lecturer>
```

- For one lecturer it only lists the name
- For the other it also lists the email address

Optional Patterns (2)

- All lecturers and their email addresses:

```
SELECT ?name ?email
WHERE
{  ?x rdf:type uni:Lecturer ;
        uni:name ?name ;
        uni:email ?email .
}
```

Optional Patterns (3)

- **The result of the previous query would be:**

?name	?email
David Billington	david@work.example.org

- **Grigoris Antoniou is listed as a lecturer, but he has no e-mail address**

Optional Patterns (4)

- As a solution we can adapt the query to use an optional pattern:

```
SELECT ?name ?email
WHERE
{  ?x rdf:type uni:Lecturer ;
        uni:name ?name .
        OPTIONAL { ?x uni:email ?email }
}
```

Optional Patterns (5)

- The meaning is roughly “give us the names of lecturers, and if known also their e-mail address”
- The result looks like this:

?name	?email
Grigoris Antoniou	
David Billington	david@work.example.org

Web links

- http://www.cambridgesemantics.com/semantic-university/sparql-by-example
- http://www.slideshare.net/ldodds/sparql-tutorial
- http://logd.tw.rpi.edu/tutorial/a_crash_course_on_sparql