The SPARQL Query Language

Raúl García-Castro, Óscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

Speaker: Raúl García Castro
rgarcia@fi.upm.es

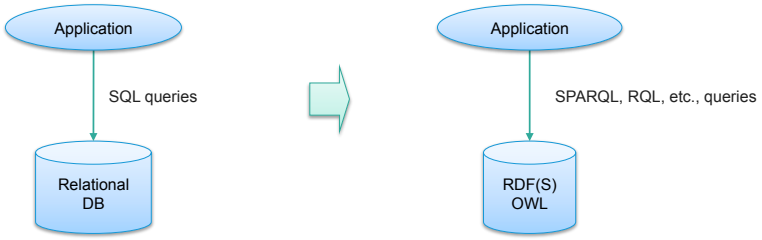
Curso Biblioteca Nacional
Madrid, Spain
21-25th November 2011

Index

- **Introduction**
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- Hands-on
- SPARQL 1.1

RDF(S) query languages

- Languages developed to allow accessing datasets expressed in RDF(S) (and in some cases OWL)



- Supported by the most important language APIs
  - Jena (HP labs)
  - Sesame (Aduna)
  - Boca (IBM)
  - ...
- There are some differences wrt. languages like SQL, such as
  - Combination of different sources
  - Trust management
  - Open World Assumption

Query types

- Selection and extraction
  - "Select all the essays, together with their authors and their authors' names"
  - "Select everything that is related to the book 'Bellum Civille'"
- Reduction: we specify what it should not be returned
  - "Select everything except for the ontological information and the book translators"
- Restructuring: the original structure is changed in the final result
  - "Invert the relationship 'author' by 'is author of'"
- Aggregation
  - "Return all the essays together with the mean number of authors per essay"
- Combination and inferences
  - "Combine the information of a book called 'La guerra civil' and whose author is Julius Caesar with the book whose identifier is 'Bellum Civille'"
  - "Select all the essays, together with its authors and author names", *including also the instances of the subclasses of Essay*
  - "Obtain the relationship 'coauthor' among persons who have written the same book"

RDF(S) query language families

- SquishQL Family
  - SquishQL
  - rdfDB Query Langua
  - RDQL
  - BRQL
  - TriQL
- XPath, XSLT, XQuery
  - XQuery for RDF
  - XsRQL
  - TreeHugger and RDFTwig
  - RDFT, Nexus Query Language
  - RDFPath, Rpath and RXPath
  - Versa
- RQL Family
  - RQL
  - SeRQL
  - eRQL
- Controlled natural language
  - Metalog
- Other
  - Algae
  - iTQL
  - N3QL
  - PerlRDF Query Language
  - RDEVICE Deductive Language
  - RDFQBE
  - RDFQL
  - TRIPLE
  - WQL

Triple database Query structure

Description graphs Query semantics

XML repository Query syntax

SPARQL

W3C Recommendation

15 January 2008

SPARQL

- SPARQL Protocol and RDF Query Language
- Supported by: Jena, Sesame, IBM Boca, etc.
- Features
  - It supports most of the aforementioned queries
  - It supports datatype reasoning (datatypes can be requested instead of actual values)
  - The domain vocabulary and the knowledge representation vocabulary are treated differently by the query interpreters
  - It allows making queries over properties with multiple values, over multiple properties of a resource and over reifications
  - Queries can contain optional statements
  - Some implementations support aggregation queries
- Limitations
  - Neither set operations nor existential or universal quantifiers can be included in the queries
  - It does not support recursive queries

SPARQL is also a protocol

- SPARQL is a Query Language …
  Find names and websites of contributors to PlanetRDF:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?website
FROM <http://planetrdf.com/bloggers.rdf>
WHERE {
     ?person foaf:weblog ?website .
     ?person foaf:name ?name .
     ?website a foaf:Document }
```

- … and a Protocol

```
http://.../qps?query-lang=http://www.w3.org/TR/rdf-sparql-query/
&graph-id=http://planetrdf.com/bloggers.rdf&query=PREFIXfoaf:
<http://xmlns.com/foaf/0.1/...
```

- Services running SPARQL queries over a set of graphs
- A transport protocol for invoking the service
- Describing the service with Web Service technologies

SPARQL Endpoints

- SPARQL protocol services
  - Enables users (human or other) to query a knowledge base using SPARQL
  - Results are typically returned in one or more machine-processable formats
- List of SPARQL Endpoints
  - http://esw.w3.org/topic/SparqlEndpoints
- Programmatic access using libraries:
  - ARC, RAP, Jena, Sesame, Javascript SPARQL, PySPARQL, etc.
- Examples:

Project	Endpoint
DBpedia	http://dbpedia.org/sparql
BBC Programmes and Music	http://bbc.openlinksw.com/sparql/
data.gov	http://semantic.data.gov/sparql
data.gov.uk	http://data.gov.uk/sparql
Musicbrainz	http://dbtune.org/musicbrainz/sparql

Example: Querying dbpedia

- People who were born in Berlin before 1900

SPARQL:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name ?birth ?death ?person WHERE {
     ?person dbpedia2:birthPlace <http://dbpedia.org/resource/Berlin> .
     ?person dbo:birthDate ?birth .
     ?person foaf:name ?name .
     ?person dbo:deathDate ?death
     FILTER (?birth < "1900-01-01"^^xsd:date) .
}
ORDER BY ?name
```

Results: Browse Go! Reset

Example: Querying dbpedia

name	birth	death	person
"Adolf Otto Reinhold Windaus"@en	"1876-12-25"^^xsd:date	"1959-06-09"^^xsd:date	:Adolf_Otto_Reinhold_Windaus
"Adolf von Baeyer"@en	"1835-10-31"^^xsd:date	"1917-08-20"^^xsd:date	:Adolf_von_Baeyer
"Alexander von Humboldt"@en	"1769-09-14"^^xsd:date	"1859-05-06"^^xsd:date	:Alexander_von_Humboldt
"Carl Borchardt"@en	"1817-02-22"^^xsd:date	"1880-06-27"^^xsd:date	:Carl_Wilhelm_Borchardt
"Carl Ludvig Engel"@en	"1778-07-03"^^xsd:date	"1840-05-04"^^xsd:date	:Carl_Ludvig_Engel
"Carl Ludwig Siegel"@en	"1896-12-31"^^xsd:date	"1981-04-04"^^xsd:date	:Carl_Ludwig_Siegel
"Carl Wilhelm Borchardt"@en	"1817-02-22"^^xsd:date	"1880-06-27"^^xsd:date	:Carl_Wilhelm_Borchardt
"Constantin Carathéodory"@en	"1873-09-13"^^xsd:date	"1950-02-02"^^xsd:date	:Constantin_Carath%C3%A9odory
"Eduard von Hartmann"@en	"1842-02-23"^^xsd:date	"1906-06-05"^^xsd:date	:Karl_Robert_Eduard_von_Hartmann
"Ernst Zermelo"@en	"1871-07-27"^^xsd:date	"1953-05-21"^^xsd:date	:Ernst_Zermelo
"Eugen Rosenstock-Huessy"@en	"1888-07-06"^^xsd:date	"1973-02-24"^^xsd:date	:Eugen_Rosenstock-Huessy
"Franz Karl Achard"@en	"1753-04-28"^^xsd:date	"2020-04-21"^^xsd:date	:Franz_Karl_Achard
"Fritz Perls"@en	"1893-07-08"^^xsd:date	"1970-03-14"^^xsd:date	:Fritz_Perls
"Georg Simmel"@en	"1858-03-01"^^xsd:date	"1918-09-28"^^xsd:date	:Georg_Simmel
"Gustav Magnus"@en	"1802-05-02"^^xsd:date	"1870-04-04"^^xsd:date	:Heinrich_Gustav_Magnus
"Hans Luther"@en	"1879-03-10"^^xsd:date	"1962-05-11"^^xsd:date	:Hans_Luther
"Heinrich Gustav Magnus"@en	"1802-05-02"^^xsd:date	"1870-04-04"^^xsd:date	:Heinrich_Gustav_Magnus
"Heinrich Louis d'Arrest"@en	"1822-07-13"^^xsd:date	"1875-06-14"^^xsd:date	:Heinrich_Louis_d%27Arrest
"Herbert Marcuse"@en	"1898-07-19"^^xsd:date	"1979-07-29"^^xsd:date	:Herbert_Marcuse
"Karl Dönitz"@en	"1891-09-16"^^xsd:date	"1980-12-24"^^xsd:date	:Karl_D%C3%B6nitz
"Kurt Eisner"@en	"1867-05-14"^^xsd:date	"1919-02-21"^^xsd:date	:Kurt_Eisner
"Leo Frobenius"@en	"1873-06-29"^^xsd:date	"1938-08-09"^^xsd:date	:Leo_Frobenius
"Moritz Schlick"@en	"1882-04-14"^^xsd:date	"1936-06-22"^^xsd:date	:Moritz_Schlick
"Oskar Messter"@en	"1866-11-21"^^xsd:date	"1943-12-06"^^xsd:date	:Oskar_Messter
"Otto Wilhelm Hermann von Abich"@en	"1806-12-11"^^xsd:date	"1886-07-01"^^xsd:date	:Otto_Wilhelm_Hermann_von_Abich
"Paul Richard Heinrich Blasius"@en	"1883-08-09"^^xsd:date	"1970-04-24"^^xsd:date	:Paul_Richard_Heinrich_Blasius
"Philipp Veit"@en	"1793-02-13"^^xsd:date	"1877-12-18"^^xsd:date	:Philipp_Veit
"Rudolph Schoenheimer"@en	"1898-05-10"^^xsd:date	"1941-09-11"^^xsd:date	:Rudolph_Schoenheimer
"Walter Benjamin"@en	"1892-07-15"^^xsd:date	"1940-09-27"^^xsd:date	:Walter_Benjamin
"Walther Rathenau"@en	"1867-09-29"^^xsd:date	"1924-06-22"^^xsd:date	:Walther_Rathenau

A simple SPARQL query

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
:book1 dc:title "SPARQL Tutorial" .
```

Query:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Query result:

title
"SPARQL Tutorial"

- A pattern is *matched* against the RDF data
- Each way a pattern can be matched yields a solution
- The sequence of solutions is filtered by: Project, distinct, order, limit/offset
- One of the result forms is applied: SELECT, CONSTRUCT, DESCRIBE, ASK

Index

- Introduction
- **Graph patterns**
- Restricting values and solutions
- SPARQL query forms
- Hands-on
- SPARQL 1.1

Graph patterns

- Basic Graph Patterns, where a set of triple patterns must match
- Group Graph Pattern, where a set of graph patterns must all match
- Optional Graph patterns, where additional patterns may extend the solution
- Alternative Graph Pattern, where two or more possible patterns are tried
- Patterns on Named Graphs, where patterns are matched against named graphs

Multiple matches

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

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

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Matching RDF literals

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ns:   <http://example.org/ns#> .
@prefix :     <http://example.org/ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

:x   ns:p     "cat"@en .
:y   ns:p     "42"^^xsd:integer .
:z   ns:p     "abc"^^dt:specialDatatype .
```

```
SELECT ?v WHERE { ?v ?p "cat" }
```

v

```
SELECT ?v WHERE { ?v ?p "cat"@en }
```

v
<http://example.org/ns#x>

```
SELECT ?v WHERE { ?v ?p 42 }
```

v
<http://example.org/ns#y>

```
SELECT ?v WHERE { ?v ?p "abc"^^<http://example.org/datatype#specialDatatype> }
```

v
<http://example.org/ns#z>

Blank node labels in query results

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Alice" .
_:b  foaf:name   "Bob" .
```

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

x	name
_:c	"Alice"
_:d	"Bob"

=

x	name
_:r	"Alice"
_:s	"Bob"

Group graph pattern

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

```
SELECT ?x
WHERE {}
```

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { { ?x foaf:name ?name . }
         { ?x foaf:mbox ?mbox . FILTER regex(?name, "Smith")}
       }
```


Optional graph patterns

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .
@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a  rdf:type        foaf:Person .
_:a  foaf:name       "Alice" .
_:a  foaf:mbox       <mailto:alice@example.com> .
_:a  foaf:mbox       <mailto:alice@work.example> .

_:b  rdf:type        foaf:Person .
_:b  foaf:name       "Bob" .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x  foaf:mbox  ?mbox }
       }
```

name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example>
“Bob"	

Multiple optional graph patterns

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name       "Alice" .
_:a  foaf:homepage   <http://work.example.org/alice/> .

_:b  foaf:name       "Bob" .
_:b  foaf:mbox       <mailto:bob@work.example> .
```

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

name	mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@work.example>	

Alternative graph patterns

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .

_:a  dc10:title     "SPARQL Query Language Tutorial" .
_:a  dc10:creator   "Alice" .
_:b  dc11:title     "SPARQL Protocol Tutorial" .
_:b  dc11:creator   "Bob" .
_:c  dc10:title     "SPARQL" .
_:c  dc11:title     "SPARQL (updated)" .
```

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

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

```
SELECT ?x ?y
WHERE  { { ?book dc10:title ?x } UNION
         { ?book dc11:title ?y } }
```

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

```
SELECT ?title ?author
WHERE
  { { ?book dc10:title ?title . ?book dc10:creator ?author }
    UNION
    { ?book dc11:title ?title . ?book dc11:creator ?author }}
```

author	title
"Alice"	"SPARQL Protocol Tutorial"
"Bob"	"SPARQL Query Language Tutorial"

Patterns on named graphs

```
# Named graph: http://example.org/foaf/aliceFoaf
@prefix foaf:<http://.../foaf/0.1/> .
@prefix rdf:<http://.../1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://.../2000/01/rdf-schema#> .

_:a  foaf:name     "Alice" .
_:a  foaf:mbox     <mailto:alice@work.example> .
_:a  foaf:knows    _:b .

_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
_:b  foaf:nick     "Bobby" .
_:b  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .

<http://example.org/foaf/bobFoaf>
     rdf:type      foaf:PersonalProfileDocument .
```

```
# Named graph: http://example.org/foaf/bobFoaf
@prefix foaf:<http://.../foaf/0.1/> .
@prefix rdf:<http://.../1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://.../2000/01/rdf-schema#> .

_:z  foaf:mbox     <mailto:bob@work.example> .
_:z  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .
_:z  foaf:nick     "Robert" .

<http://example.org/foaf/bobFoaf>
     rdf:type      foaf:PersonalProfileDocument .
```


Patterns on named graphs II

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
  {
    GRAPH ?src
    { ?x foaf:mbox <mailto:bob@work.example> .
      ?x foaf:nick ?bobNick
    }
  }
```

src	bobNick
<http://example.org/foaf/aliceFoaf>	"Bobby"
<http://example.org/foaf/bobFoaf>	"Robert"

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX data: <http://example.org/foaf/>

SELECT ?nick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
  {
     GRAPH data:bobFoaf {
         ?x foaf:mbox <mailto:bob@work.example> .
         ?x foaf:nick ?nick }
  }
```

nick
"Robert"

Index

- Introduction
- Graph patterns
- **Restricting values and solutions**
- SPARQL query forms
- Hands-on
- SPARQL 1.1

Restricting values

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "^SPARQL")
        }
```

title
"SPARQL Tutorial"

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "web", "i" )
        }
```

title
"The Semantic Web"

```
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 < 30.5)
          ?x dc:title ?title . }
```

title	price
"The Semantic Web"	23

Value tests

- Based on XQuery 1.0 and XPath 2.0 Function and Operators
- XSD boolean, string, integer, decimal, float, double, dateTime
- Notation <, >, =, <=, >= and != for value comparison Apply to any type
- BOUND, isURI, isBLANK, isLITERAL
- REGEX, LANG, DATATYPE, STR (lexical form)
- Function call for casting and extensions functions

Solution sequences and modifiers

- Order modifier: put the solutions in order

```
SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY ?name DESC(?emp)
```

- Projection modifier: choose certain variables

```
SELECT ?name
WHERE
 { ?x foaf:name ?name }
```

- Distinct modifier: ensure solutions in the sequence are unique

```
SELECT DISTINCT ?name
WHERE { ?x foaf:name ?name }
```

- Reduced modifier: permit elimination of some non-unique solutions

```
SELECT REDUCED ?name
WHERE { ?x foaf:name ?name }
```

- Limit modifier: restrict the number of solutions

```
SELECT ?name
WHERE { ?x foaf:name ?name }
LIMIT 20
```

- Offset modifier: control where the solutions start from in the overall sequence of solutions

```
SELECT  ?name WHERE { ?x foaf:name ?name }
ORDER BY ?name
LIMIT   5
OFFSET  10
```

Index

- Introduction
- Graph patterns
- Restricting values and solutions
- **SPARQL query forms**
- Hands-on
- SPARQL 1.1

SPARQL query forms

- SELECT
  - Returns all, or a subset of, the variables bound in a query pattern match
- CONSTRUCT
  - Returns an RDF graph constructed by substituting variables in a set of triple templates
- ASK
  - Returns a boolean indicating whether a query pattern matches or not
- DESCRIBE
  - Returns an RDF graph that describes the resources found

SPARQL query forms: SELECT

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .

_:a    foaf:name   "Alice" .
_:a    foaf:knows  _:b .
_:a    foaf:knows  _:c .

_:b    foaf:name   "Bob" .

_:c    foaf:name   "Clare" .
_:c    foaf:nick   "CT" .
```

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
  { ?x foaf:knows ?y ;
       foaf:name ?nameX .
    ?y foaf:name ?nameY .
    OPTIONAL { ?y foaf:nick ?nickY }
  }
```

nameX	nameY	nickY
"Alice"	"Bob"	
"Alice"	"Clare"	"CT"

SPARQL query forms: CONSTRUCT

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .

_:a    foaf:name   "Alice" .
_:a    foaf:mbox   <mailto:alice@example.org> .
```

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>

CONSTRUCT   { <http://example.org/person#Alice> vcard:FN ?name }

WHERE       { ?x foaf:name ?name }
```

Query result:

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .

<http://example.org/person#Alice> vcard:FN "Alice" .
```

SPARQL query forms: ASK

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name       "Alice" .
_:a  foaf:homepage   <http://work.example.org/alice/> .

_:b  foaf:name       "Bob" .
_:b  foaf:mbox       <mailto:bob@work.example> .
```

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" }
```

Query result:

```
yes
```


SPARQL query forms: DESCRIBE

```
PREFIX ent:  <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
```

Query result:

```
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix vcard:  <http://www.w3.org/2001/vcard-rdf/3.0> .
@prefix exOrg:  <http://org.example.com/employees#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#>

_:a     exOrg:employeeId    "1234" ;

        foaf:mbox_sha1sum   "ABCD1234" ;
        vcard:N
         [ vcard:Family       "Smith" ;
           vcard:Given        "John"  ] .

foaf:mbox_sha1sum  rdf:type  owl:InverseFunctionalProperty .
```

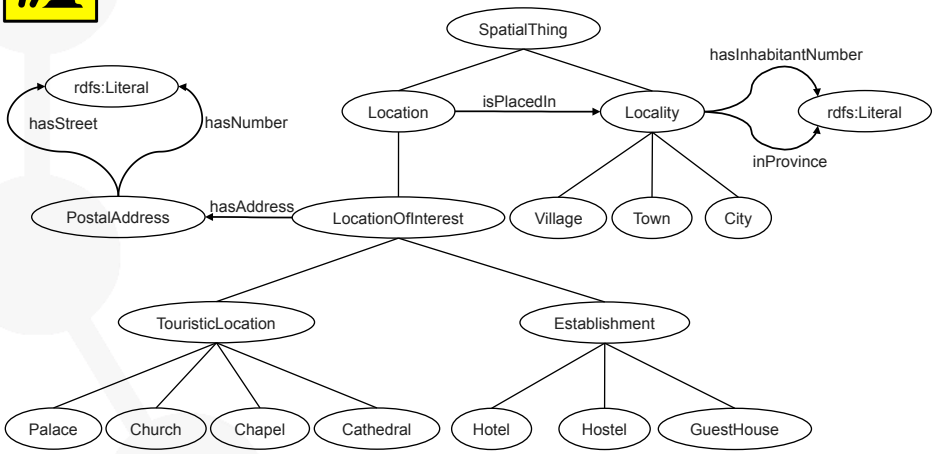
Index

- Introduction
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- **Hands-on**
- SPARQL 1.1

Exercise

- **Objective**
  - Understand how to perform SPARQL queries
- **Tasks**
  - Perform a set of SPARQL queries over a sample ontology
    - Browse to:
      - http://my.computer.ip:8080/openrdf-workbench
    - Select repository GP_InMemoryRDFS
    - Select the “Query” option from the left menu

Sample ontology



Queries on this model

1. Get all the classes
2. Get the subclasses of the class *Establishment*
3. Get the instances of the class *City*
4. Get the number of inhabitants of *Santiago de Compostela*
5. Get the number of inhabitants of *Santiago de Compostela* and of *Arzua*
6. Get different places with the inhabitants number, ordering the results by the name of the place (ascending)
7. Get all the instances of *Locality* with their inhabitant number (if it exists)
8. Get all the places with more than 200.000 inhabitants
9. Get postal data of *Pazo_Breogan (street, number, locality, province)*
10. Get the subclasses of class *Location*
11. Get the instances of class *Locality*
12. Describe the resource with *rdfs:label* "*Madrid*"
13. Construct the RDF(S) graph that directly relates all the touristic places with their respective provinces, using a new property called "*isIn*"
14. Ask if there is some instance of *Town*

Queries on the model

1) Get all the classes

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x WHERE { ?x a rdfs:Class. }
```

2) Get the subclasses of the class *Establishment*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT ?x WHERE { ?x rdfs:subClassOf pr:Establishment. }
```

3) Get the instances of the class *City*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT ?x WHERE { ?x a pr:City. }
```


Queries on the instances

4) Get the number of inhabitants of *Santiago de Compostela*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT ?x WHERE { pr:Santiago_de_Compostela pr:hasInhabitantNumber ?x. }
```

5) Get the number of inhabitants of *Santiago de Compostela* and of *Arzua*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT ?x WHERE {
                {pr:Santiago_de_Compostela pr:hasInhabitantNumber ?x.}
                  UNION
                {pr:Arzua pr:hasInhabitantNumber ?x.}
               }
```

6) Get different places with the inhabitants number, ordering the results by the name of the place (ascending)

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?y  WHERE { $loc pr:hasInhabitantNumber ?y;
                      rdfs:label ?x.}
ORDER BY ASC(?x)
```

Queries on the instances II

7) Get all the instances of *Locality* with their inhabitant number (if it exists)

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?y  WHERE { $loc a pr:Locality;
                            rdfs:label ?x.
                      OPTIONAL {$loc pr:hasInhabitantNumber ?y.} }
```

8) Get all the places with more than 200.000 inhabitants

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?y  WHERE { $loc pr:hasInhabitantNumber ?y;
                            rdfs:label ?x.
                      FILTER(?y > 200000) }
```

9) Get postal data of *Pazo_Breogan (street, number, locality, province)*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?street ?number ?locality ?province
WHERE { pr:Pazo_Breogan pr:isPlacedIn $pob;
                        pr:hasAddress $dir.
        $pob rdfs:label ?locality;
             pr:inProvince ?province.
        $dir pr:hasStreet ?street;
             pr:hasNumber ?number.}
```


Queries with inference

10) Get the subclasses of class *Location*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT ?x WHERE { ?x rdfs:subClassOf pr:Location. }
```

11) Get the instances of class *Locality*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT ?x WHERE { ?x a pr:Locality. }
```

Special query (SELECT *)

12) Get the values of all the variables in the query

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
SELECT * WHERE { ?x pr:hasInhabitantNumber ?y. }
```

Different query forms

13) Describe the resource with *rdfs:label* "*Madrid*"

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
DESCRIBE ?x WHERE { ?x rdfs:label "Madrid". }
```

14) Construct the RDF(S) graph that directly relates all the touristic places with their respective provinces, using a new property called "*isIn*"

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
CONSTRUCT {?x pr:isIn ?y}
WHERE {
        ?x a pr:TouristicLocation;
             pr:isPlacedIn $pob.
        $pob pr:inProvince ?y. }
```

15) Ask if there is some instance of *Town*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
ASK WHERE {?a a pr:Town}
```

16) Ask if there is some instance of *Chapel*

```
PREFIX pr: <http://GP-onto.fi.upm.es/exercise2#>
ASK WHERE {?a a pr:Chapel}
```


Index

- Introduction
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- Hands-on
- **SPARQL 1.1**

SPARQL 1.1

- W3C **Working Draft!!!**
- New features in **Query** language:
  - ***Aggregate functions***: COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT, and SAMPLE.
  - ***Subqueries***: Nest the results of a query within another query
  - ***Negation***: Check the absence of triples in a graph
  - ***Expressions in SELECT***: Introduce new variables in the SELECT clause
  - ***Property paths***: Search graphs through structures that involve arbitrary-length paths
  - ***Assignment***: BIND keyword and expressions in SELECT and GROUP_BY
  - ***Short form for CONSTRUCT***
  - ***Expanded functions and operators***: EXISTS, NOT EXISTS, SUBSTR, etc.

Other SPARQL 1.1 specifications

- **Update**
  - Language extension to express updates to an RDF graph/store
- **Protocol**
  - Changes related to the update operation
- **Service description**
  - Discover a SPARQL endpoint's capabilities and summary information of its data
- **Graph Store HTTP Protocol**
  - Update and fetch RDF graph content from a Graph Store over HTTP in the REST style
- **Entailment Regimes**
  - Define the semantics of SPARQL queries for some entailment frameworks: OWL flavors, RDFS, RIF
- **Federation Extensions**
  - Take a query and provide solutions based on information from many different sources
- **Query Result JSON, CSV, TSV**

Main References

- W3C SPARQL Working Group
  http://www.w3.org/2001/sw/DataAccess/
- Prud'hommeaux E, Seaborne A (2008) SPARQL Query Language for RDF. W3C Recommendation
  http://www.w3.org/TR/rdf-sparql-query/
- SPARQL validator:
  http://www.sparql.org/query-validator.html
- SPARQL implementations:
  http://esw.w3.org/topic/SparqlImplementations
- SPARQL Endpoints
  http://esw.w3.org/topic/SparqlEndpoints
- SPARQL in Dbpedia
  http://dbpedia.org/sparql

Thank you for your attention!

Speaker: Raúl García Castro
rgarcia@fi.upm.es

Curso Biblioteca Nacional
Madrid, Spain
21-25th November 2011