



AMPER Course

Introduction to SPARQL

Raúl García-Castro, Óscar Corcho, Óscar Muñoz-García

{rgarcia, ocorcho, omunoz}@fi.upm.es

<http://www.oeg-upm.net/>



Ontology Engineering Group
Laboratorio de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain

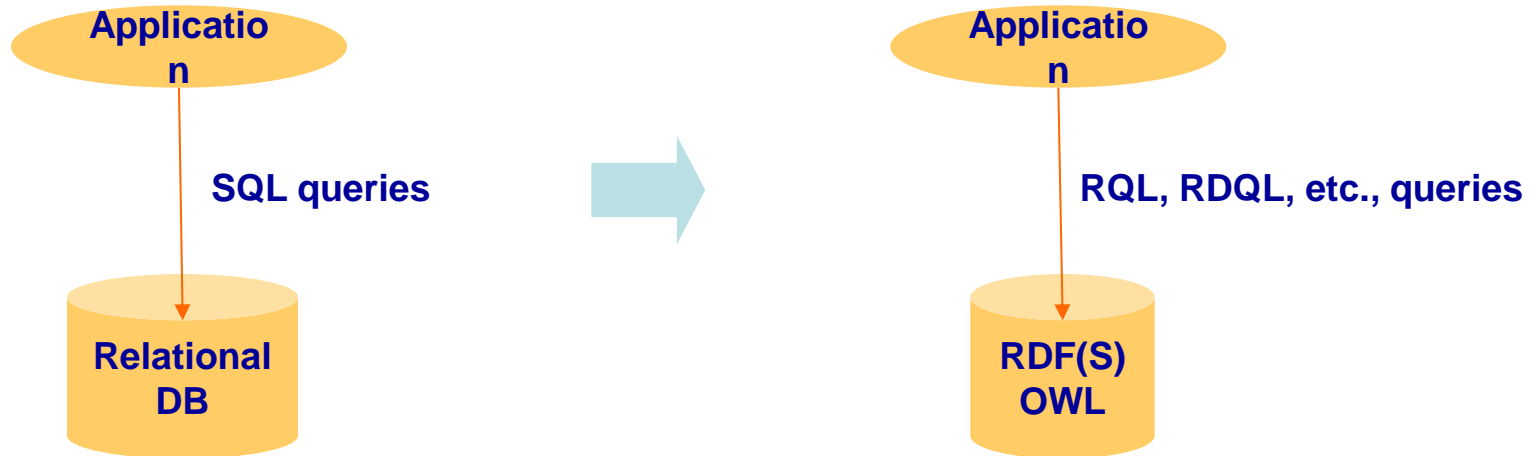


Index

- **RDF query languages**
- SPARQL
- Turtle RDF syntax
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- Hands-on

RDF(S) query languages

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

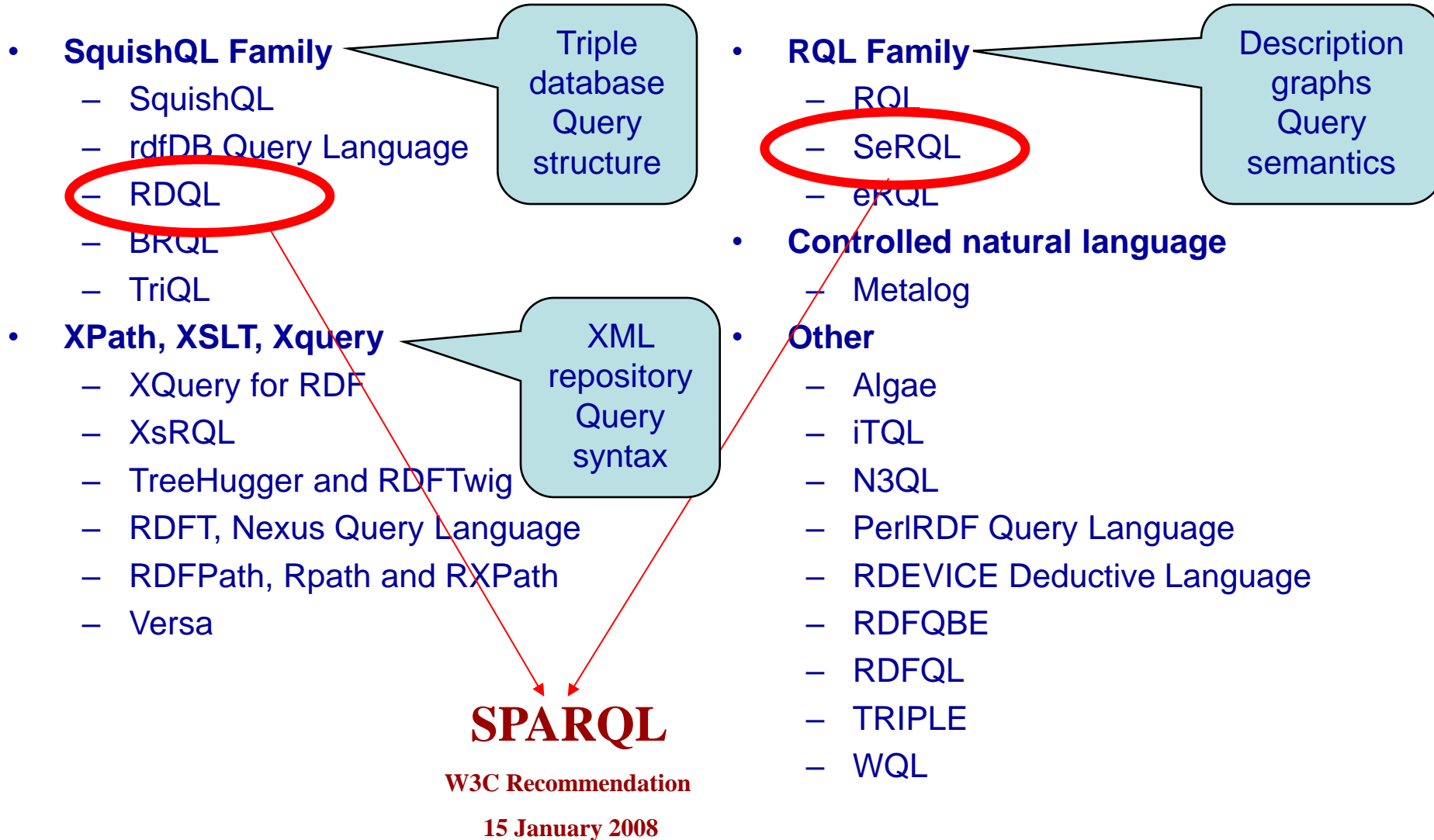


- Supported by the most important language APIs
 - Jena (HP labs)
 - Sesame (Aduna)
 - Boca (IBM)
 - ...
- There are some differences wrt languages like SQL, such as
 - Combination of different sources
 - Trust management
 - Open World Assumption

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 Civile’”
- **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 Civile’”
 - “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



Index

- RDF query languages
- **SPARQL**
- Turtle RDF syntax
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- Hands-on

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
- Based on ideas from earlier protocol work such as Joseki
- Describing the service with Web Service technologies

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

- RDF query languages
- SPARQL
- **Turtle RDF syntax**
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- Hands-on

Turtle: URIs, blank nodes, literals

- URIs

Enclosed in <>

<URI>

or

@prefix prefix <http://....>

prefix:name

- Blank Nodes

:name

or

[] for a Blank Node used once

- Literals

"Literal"

"Literal"@language

"""Long literal with
newlines"""

- Datatyped Literals

"lexical form"^^datatype URI

"10"^^xsd:integer

"2006-09-04"^^xsd:date

Turtle: Triples and abbreviations

- Triples separated by .

:a :b :c . :d :e :f .

- Common triple predicate and subject:

:a :b :c, :d .

which is the same as :a :b :c . :a :b :d .

- Common triple subject:

:a :b :c; :d :e .

which is the same as: :a :b :c . :a :d :e .

- Blank node as a subject

:a :b [:c :d]

which is the same as: :a :b _:x . _:x :c :d .

for blank node _:x

- RDF Collections

– :a :b (:c :d :e :f)

which is short for many triples

Index

- RDF query languages
- SPARQL
- Turtle RDF syntax
- **Graph patterns**
- Restricting values and solutions
- SPARQL query forms
- Hands-on

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

Basic graph patterns: 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>

Basic graph patterns: 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>

Basic graph patterns: 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

- RDF query languages
- SPARQL
- Turtle RDF syntax
- Graph patterns
- **Restricting values and solutions**
- SPARQL query forms
- Hands-on

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
- **Projection modifier:** choose certain variables
- **Distinct modifier:** ensure solutions in the sequence are unique
- **Reduced modifier:** permit elimination of some non-unique solutions
- **Offset modifier:** control where the solutions start from in the overall sequence of solutions
- **Limit modifier:** restrict the number of solutions

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

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

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

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

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

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

Index

- RDF query languages
- SPARQL
- Turtle RDF syntax
- Graph patterns
- Restricting values and solutions
- **SPARQL query forms**
- Hands-on

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

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

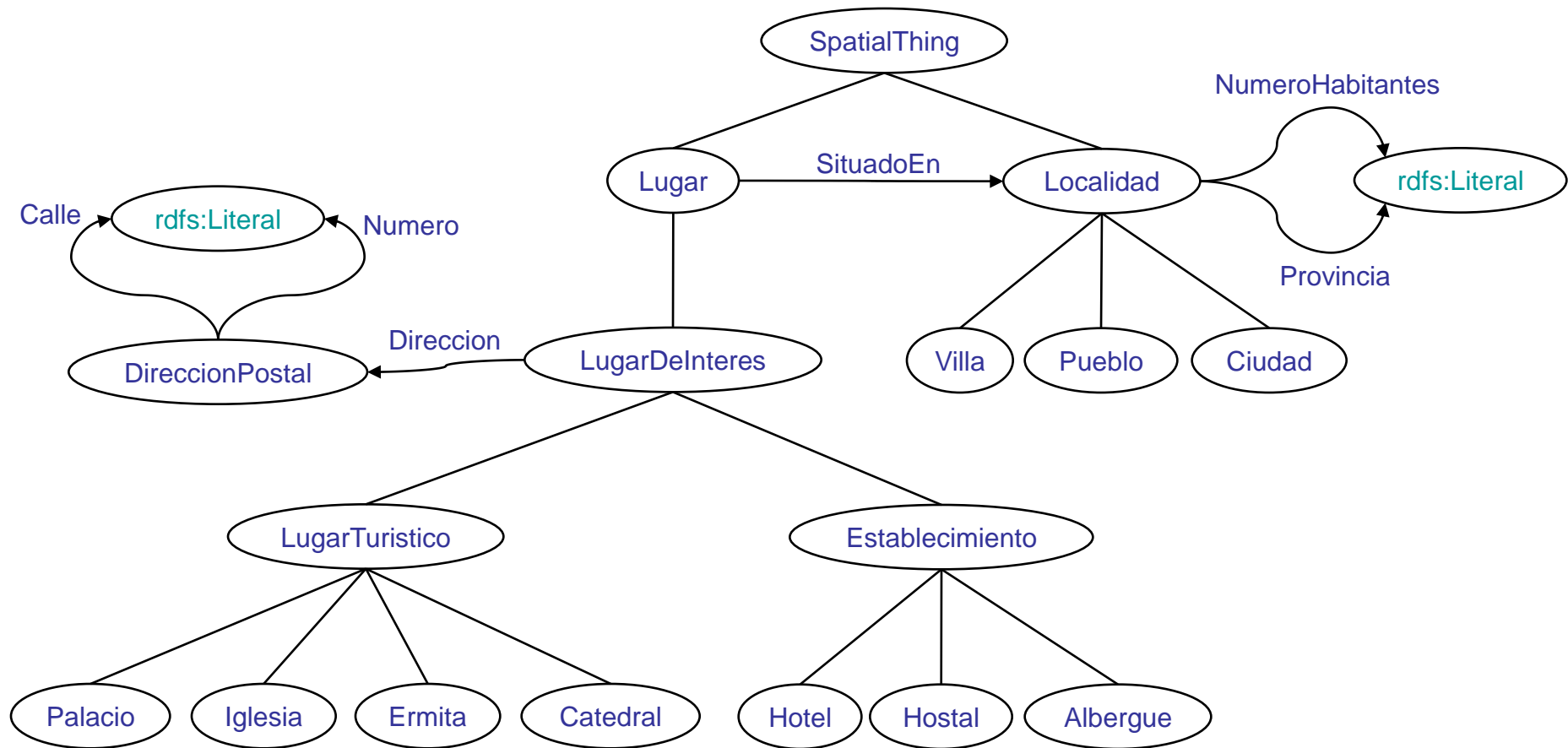
Index

- RDF query languages
- SPARQL
- Turtle RDF syntax
- Graph patterns
- Restricting values and solutions
- SPARQL query forms
- **Hands-on**

Hands-on

- Perform a set of queries over the sample ontology
- Browse to:
 - <http://my.computer.ip:8080/openrdf-workbench>
- Select repository *GP-native-rdfs*
- Select the *Query* option from the left menu

Sample ontology



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 *Establecimiento*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>  
SELECT ?x WHERE { ?x rdfs:subClassOf pr:Establecimiento. }
```

3) Get the instances of the class *Ciudad*

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>  
SELECT ?x WHERE { ?x a pr:Ciudad. }
```

Queries on the instances

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

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
SELECT ?x WHERE { pr:Santiago_de_Compostela pr:NumeroHabitantes ?x. }
```

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

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
SELECT ?x WHERE {
    {pr:Santiago_de_Compostela pr:NumeroHabitantes ?x.}
    UNION
    {pr:Arzua pr:NumeroHabitantes ?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/Practica2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?y WHERE { $sitio pr:NumeroHabitantes ?y;
                      rdfs:label ?x.}

ORDER BY ASC(?x)
```

Queries on the instances II

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

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?y WHERE { $sitio a pr:Localidad;
                        rdfs:label ?x.
                        OPTIONAL {$sitio pr:NumeroHabitantes ?y.} }
```

8) Get all the places with more than 200.000 inhabitants

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?y WHERE { $sitio pr:NumeroHabitantes ?y;
                      rdfs:label ?x.
                      FILTER(?y > 200000) }
```

9) Get postal data of *Pazo de Breogan* (calle, número, localidad, provincia)

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?calle ?numero ?poblacion ?provincia
WHERE { pr:Pazo_Breogan pr:SituadoEn $pob;
        pr:Direccion $dir.
        $pob rdfs:label ?poblacion;
        pr:Provincia ?provincia.
        $dir pr:Calle ?calle;
        pr:Numero ?numero.}
```

Queries with inference

10) Get the subclasses of class *Lugar*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>  
SELECT ?x WHERE { ?x rdfs:subClassOf pr:Lugar. }
```

11) Get the instances of class *Localidad*

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>  
SELECT ?x WHERE { ?x a pr:Localidad. }
```

Special query (SELECT *)

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

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>  
SELECT * WHERE { ?x pr:NumeroHabitantes ?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 "*estaEn*".

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
CONSTRUCT {?x pr:estaEn ?y}
WHERE {
    ?x a pr:LugarTuristico;
        pr:SituadoEn $pob.
    $pob pr:Provincia ?y. }
```

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

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
ASK WHERE {?a a pr:Pueblo}
```

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

```
PREFIX pr: <http://GP-onto.fi.upm.es/Practica2#>
ASK WHERE {?a a pr:Ermita}
```

References

- SPARQL specification
 - <http://www.w3.org/TR/rdf-sparql-query/>
- SPARQL validator
 - <http://www.sparql.org/validator.html>
- SPARQL implementations
 - <http://esw.w3.org/topic/SparqlImplementations>
- SPARQL tutorials
 - <http://jena.sourceforge.net/ARQ/Tutorial/>
 - <http://www.w3.org/2004/Talks/17Dec-sparql/intro/all.html>
 - <http://www.cs.man.ac.uk/~bparsia/2006/row-tutorial/>



AMPER Course

Introduction to SPARQL

Raúl García-Castro, Óscar Corcho, Óscar Muñoz-García

{rgarcia, ocorcho, omunoz}@fi.upm.es

<http://www.oeg-upm.net/>



Ontology Engineering Group
Laboratorio de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain

