A brief introduction to RDF, RDF Schema and SPARQL Practical assignment

Oscar Corcho (ocorcho@fi.upm.es)
Universidad Politécnica de Madrid

Acknowledgements:
Asunción Gómez-Pérez, Raúl García-Castro

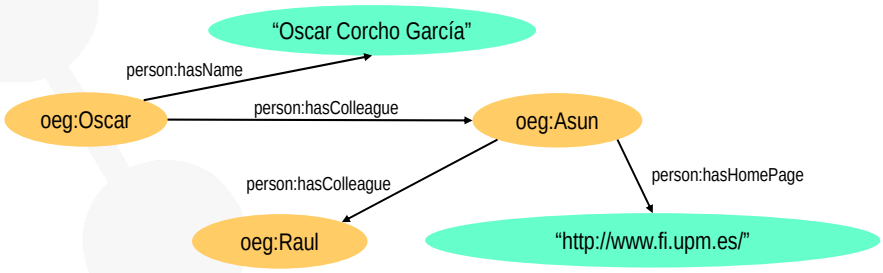
Index

- **Resource Description Framework (RDF)**
- RDF Schema
- RDF(S) Management APIs
- SPARQL

- Assignment details

RDF: Resource Description Framework

- W3C recommendation
- RDF is graphical formalism (+ XML syntax + semantics)
  - For representing metadata
  - For describing the semantics of information in a machine-accessible way
  - Resources are described in terms of properties and property values using RDF statements
  - Statements are represented as triples, consisting of a subject, predicate and object. [S, P, O]



URIs (Universal-Uniform Resource Identifer)

- Two types of identifiers can be used to identify Linked Data resources
  - Hash URIs or URIRefs (Unique Resource Identifiers References)
    - A URI and an optional Fragment Identifier separated from the URI by the hash symbol '#'
    - http://www.ontology.org/people#Person
    - people:Person
  - Slash URIs or plain URIs can also be used, as in FOAF:
    - http://xmlns.com/foaf/0.1/Person

RDF Serialisations

- Normative
  - RDF/XML (www.w3.org/TR/rdf-syntax-grammar/)
- Alternative (for human consumption)
  - N3 (http://www.w3.org/DesignIssues/Notation3.html)
  - Turtle (http://www.dajobe.org/2004/01/turtle/)
  - TriX (http://www.w3.org/2004/03/trix/)
  - …

Important: the RDF serializations allow different syntactic variants. E.g., the order of RDF statements has no meaning

RDF Serialisations. RDF/XML

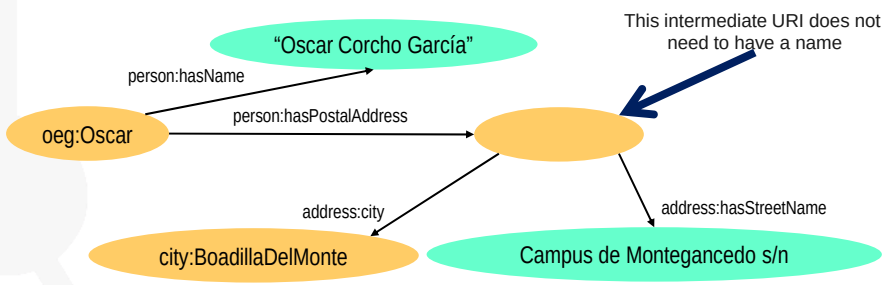
```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:person="http://www.ontologies.org/ontologies/people#"
   xmlns="http://www.oeg-upm.net/ontologies/people#"
   xml:base="http://www.oeg-upm.net/ontologies/people">

   <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
   <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague"/>
   <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasName"/>

   <rdf:Description rdf:about="#Raul"/>
   <rdf:Description rdf:about="#Asun">
      <person:hasColleague rdf:resource="#Raul"/>
      <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
   </rdf:Description>
   <rdf:Description rdf:about="#Oscar">
      <person:hasColleague rdf:resource="#Asun"/>
      <person:hasName>Oscar Corcho García</person:hasName>
   </rdf:Description>

</rdf:RDF>
```

RDF Serialisations. N3

```
@base  <http://www.oeg-upm.net/ontologies/people >
@prefix person: <http://www.ontologies.org/ontologies/people#>
:Asun    person:hasColleague :Raul ;
         person:hasHomePage “http://www.fi.upm.es/”.
:Oscar   person:hasColleague :Asun ;
         person:hasName “Óscar Corcho García”.
```


Blank nodes: structured property values

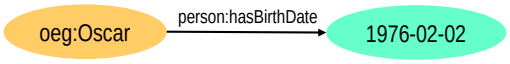
- Most real-world data involves structures that are more complicated than sets of RDF triple statements



- In RDF/XML, it is an <rdf:Description> node with no rdf:about
- In N3, it is a resource identifier that starts with ‘_’
  - E.g., “_:nodeX”

Typed literals

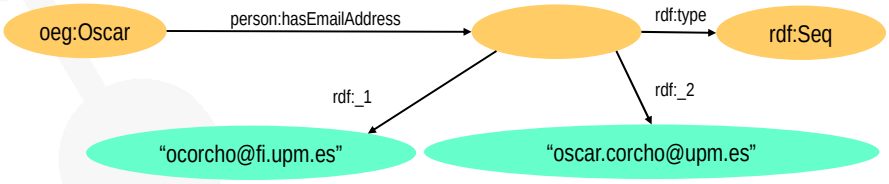
- So far, all values have been presented as strings
- XML Schema datatypes can be used to specify values (objects in some RDF triple statements)



- In RDF/XML, this is expressed as:
  - <rdf:Description rdf:about="#Oscar">
    <person:hasBirthDate
    rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1976-02-02
    </person:hasBirthDate>
    </rdf:Description>
- In N3, this is expressed as:
  - oeg:Oscar person:hasBirthDate "1976-02-02"^^xsd:date .

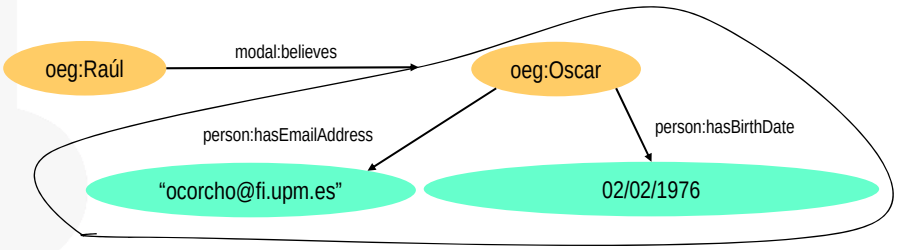
RDF Containers

- There is often the need to describe groups of things
  - A book was created by several authors
  - A lesson is taught by several persons
  - etc.
- RDF provides a container vocabulary
  - rdf:Bag → A group of resources or literals, possibly including duplicate members, where the order of members is not significant
  - rdf:Seq → A group of resources or literals, possibly including duplicate members, where the order of members is significant
  - rdf:Alt → A group of resources or literals that are alternatives (typically for a single value of a property)



RDF Reification

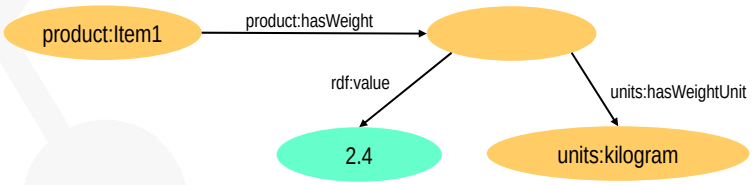
- RDF statements about other RDF statements
  - "Raúl believes that Oscar's birthdate is on Feb 2nd, 1976 and that his e-mail address is ocorcho@fi.upm.es"



- RDF Reification
  - Allows expressing beliefs (and other modalities)
  - Allows expressing trust models, digital signatures, etc.
  - Allows expressing metadata about metadata

Main value of a structured value

- Sometimes one of the values of a structured value is the main one
  - The weight of an item is 2.4 kilograms
  - The most important value is 2.4, which is expressed with rdf:value
- Scarcely used



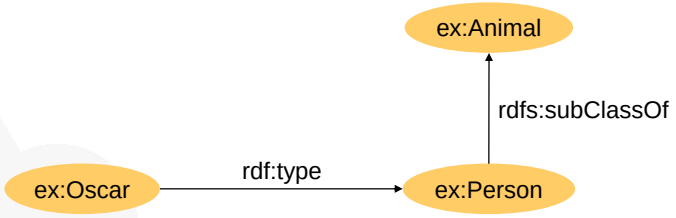
Index

- Resource Description Framework (RDF)
- **RDF Schema**
- RDF(S) Management APIs
- SPARQL

- Assignment details

RDFS: RDF Schema

- W3C Recommendation
- RDF Schema extends RDF to enable talking about classes of resources, and the properties to be used with them
  - Class definition: rdfs:Class, rdfs:subClassOf
  - Property definition: rdfs:subPropertyOf, rdfs:range, rdfs:domain
  - Other primitives: rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
- RDFS vocabulary adds constraints on models, e.g.:
  - $\forall$x,y,z type(x,y) and subClassOf(y,z) $\rightarrow$ type(x,z)



RDF(S) Serialisations. RDF/XML syntax

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:person="http://www.ontologies.org/ontologies/people#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns="http://www.oeg-upm.net/ontologies/people#"
   xml:base="http://www.oeg-upm.net/ontologies/people">

  <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#Professor">
    <rdfs:subClassOf>
      <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#Person"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#Lecturer">
    <rdfs:subClassOf rdf:resource="http://www.ontologies.org/ontologies/people#Person"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.ontologies.org/ontologies/people#PhD">
     <rdfs:subClassOf rdf:resource="http://www.ontologies.org/ontologies/people#Person"/>
  </rdfs:Class>
 …
```


RDF(S) Serialisations. RDF/XML syntax

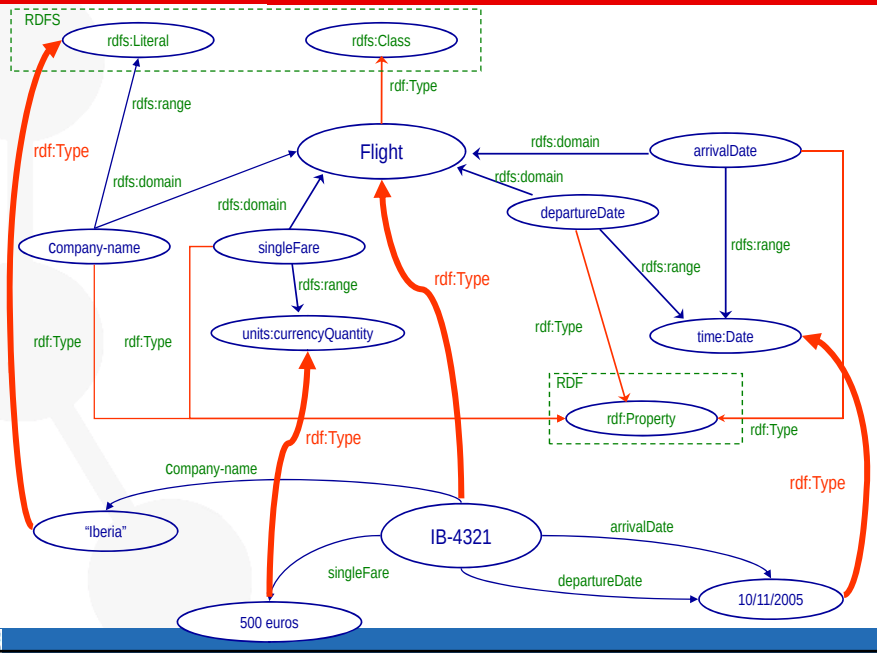
```
…
 <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasHomePage"/>
 <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasColleague">
   <rdfs:domain rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
   <rdfs:range rdf:resource=" http://www.ontologies.org/ontologies/people#Person"/>
 </rdf:Property>
 <rdf:Property rdf:about="http://www.ontologies.org/ontologies/people#hasName">
   <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
 </rdf:Property>

 <person:PhD rdf:ID="Raul"/>
 <person:Professor rdf:ID="Asun">
     <person:hasColleague rdf:resource="#Raul"/>
     <person:hasHomePage>http://www.fi.upm.es</person:hasHomePage>
 </person:Professor>
 <person:Lecturer rdf:ID="Oscar">
     <person:hasColleague rdf:resource="#Asun"/>
     <person:hasName>Óscar Corcho García</person:hasName>
 </person:Lecturer>
</rdf:RDF>
```

RDF(S) Serialisations. N3

```
@base  <http://www.oeg-upm.net/ontologies/people >
@prefix person: <http://www.ontologies.org/ontologies/people#>
person:hasColleague        a rdf:Property;
                           rdfs:domain person:Person;
                           rdfs:range person:Person.
person:Professor rdfs:subClassOf person:Person.
person:Lecturer rdfs:subClassOf person:Person.
person:PhD rdfs:subClassOf person:Person.
:Asun     a person:Professor;
          person:hasColleague :Raul ;
          person:hasHomePage "http://www.fi.upm.es/".
:Oscar    a person:Lecturer;
          person:hasColleague :Asun ;
          person:hasName "Óscar Corcho García".
:Raul     a person:PhD.
```

a is equivalent to rdf:type

RDF(S) Example



Index

- Resource Description Framework (RDF)
- RDF Schema
- **RDF(S) Management APIs**
- SPARQL

- Assignment details

Sample RDF APIs

- RDF libraries for different languages:
  - Java, Python, C, C++, C#, .Net, Javascript, Tcl/Tk, PHP, Lisp, Obj-C, Prolog, Perl, Ruby, Haskell
  - List in
- Usually related to a RDF repository
- Multilanguage:
  - Redland RDF Application Framework (C, Perl, PHP, Python and Ruby): http://www.redland.opensource.ac.uk/
- Java:
  - Jena: http://jena.sourceforge.net/
  - Sesame: http://www.openrdf.org/
- PHP:
  - RAP - RDF API for PHP: http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/
- Python:
  - RDFLib: http://rdflib.net/
  - Pyrple: http://infomesh.net/pyrple/

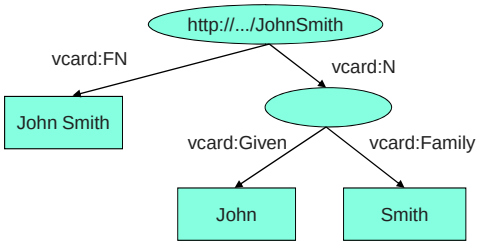
Jena

- Java framework for building Semantic Web applications
- Open source software from HP Labs
- The Jena framework includes:
  - A RDF API
  - An OWL API
  - Reading and writing RDF in RDF/XML, N3 and N-Triples
  - In-memory and persistent storage
  - A rule based inference engine
  - SPARQL query engine

Sesame

- A framework for storage, querying and inferencing of RDF and RDF Schema
- A Java Library for handling RDF
- A Database Server for (remote) access to repositories of RDF data
- Highly expressive query and transformation languages
  - SeRQL, SPARQL
- Various backends
  - Native Store
  - RDBMS (MySQL, Oracle 10, DB2, PostgreSQL)
  - main memory
- Reasoning support
  - RDF Schema reasoner
  - OWL DLP (OWLIM)
  - domain reasoning (custom rule engine)

Jena example. Graph creation




```
// some definitions
String personURI = "http://somewhere/JohnSmith";
String givenName = "John";
String familyName = "Smith";
String fullName = givenName + " " + familyName;
// create an empty
Model Model model = ModelFactory.createDefaultModel();
// create the resource
// and add the properties cascading style
Resource johnSmith = model.createResource(personURI)
    .addProperty(VCARD.FN, fullName)
    .addProperty(VCARD.N, model.createResource()
    .addProperty(VCARD.Given, givenName)
    .addProperty(VCARD.Family, familyName));
```


Jena example. Read and write

```
// create an empty model
Model model = ModelFactory.createDefaultModel();

// use the FileManager to find the input file
InputStream in = FileManager.get().open( inputFileName );
if (in == null) {
    throw new IllegalArgumentException("File not found");
}

// read the RDF/XML file
model.read(in, "");

// write it to standard out
model.write(System.out);
```

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
 >
  <rdf:Description rdf:nodeID="A0">
    <vcard:Family>Smith</vcard:Family>
    <vcard:Given>John</vcard:Given>
  </rdf:Description>
  <rdf:Description rdf:about='http://somewhere/JohnSmith/'>
    <vcard:FN>John Smith</vcard:FN>
    <vcard:N rdf:nodeID="A0"/>
  </rdf:Description>
...
</rdf:RDF>
```

Some RDF editors

- IsaViz
  - http://www.w3.org/2001/11/IsaViz/
- Morla
  - http://www.morlardf.net/
- RDFAuthor
  - http://rdfweb.org/people/damian/RDFAuthor/
- RdfGravity
  - http://semweb.salzburgresearch.at/apps/rdf-gravity/
- Rhodonite
  - http://rhodonite.angelite.nl/
- Altova SemanticWorks

Main References

- Brickley D, Guha RV (2004) RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation
  http://www.w3.org/TR/PR-rdf-schema/
- Lassila O, Swick R (1999) Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation
  http://www.w3.org/TR/REC-rdf-syntax/

- RDF validator:
  http://www.w3.org/RDF/Validator/
- RDF resources:
  http://planetrdf.com/guide/

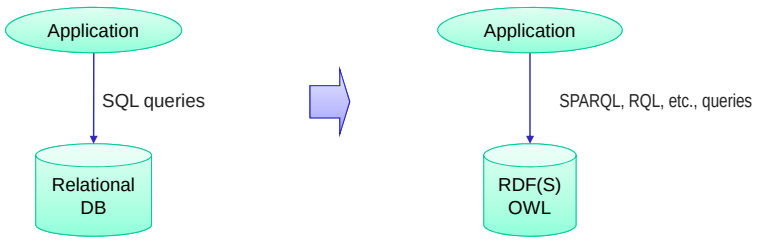
Index

- Resource Description Framework (RDF)
- RDF Schema
- RDF(S) Management APIs
- **SPARQL**

- Assignment details

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

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/...
```


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

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

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"

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
- Limit modifier: restrict the number of solutions
- Offset modifier: control where the solutions start from in the overall sequence 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 }
LIMIT 20
```

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

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 .
```


Main References

- 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/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

Index

- Resource Description Framework (RDF)
- RDF Schema
- RDF(S) Management APIs
- SPARQL

- **Assignment details**
  - http://delicias.dia.fi.upm.es/wiki/index.php/InteligenciaArtificial-grado-10-11

A brief introduction to RDF, RDF Schema and SPARQL Practical assignment

Oscar Corcho (ocorcho@fi.upm.es)
Universidad Politécnica de Madrid

Acknowledgements:
Asunción Gómez-Pérez, Raúl García-Castro