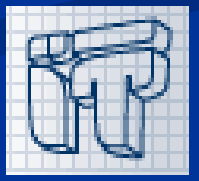
Semantic Web Information Management

Norberto Fernández
Telematics Engineering Department
berto@ it.uc3m.es



Motivation

- Module 1: An ontology models a domain of knowledge…
- Module 2: … using the ontology as vocabulary, we represent (meta)data in RDF…
- …, OK, so we have tons of triples representing all our information: What do we do with them?
  - RDF storage
  - RDF querying

  → Semantic Web Information management

Introduction

- Module 3: Semantic Web information management
  - RDF querying
    - We will see this in theoretical lessons
      - SPARQL: http://www.w3.org/TR/rdf-sparql-query/
  - RDF storage
    - We will see this mainly in lab
      - Jena: http://jena.sourceforge.net/

RDF Querying

Query Languages for RDF

- Several proposals out there…
  - RDQL, Triple, SeRQL, RQL, RDFQL, N3, etc.
    - http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/
- But nowadays there is also a W3C recommendation: SPARQL

SPARQL

- Pronounced "*sparkle*"
- Formerly an acronym:
  - *SPARQL Protocol And RDF Query Language*
- Defined by the RDF Data Access Working Group of the W3C
- Three related specifications:
  - SPARQL Query Language for RDF
    - Query language
  - SPARQL Protocol for RDF
    - Protocol to access SPARQL endpoints
  - SPARQL Query Results XML Format
    - Representing query results in XML
- W3C recommendations since January 2008

SPARQL Syntax: IRIs (I)

- Subset of URIs used in SPARQL queries as resources references
- RDF URI references containing "<", ">", double quote, space, "{", "}", "|", "\", "^", and "`" are not IRIs
- The behavior of a SPARQL query against RDF statements composed of such RDF URI references is not defined

SPARQL Syntax: IRIs (II)

- Three mechanisms to represents IRIs:

```
<http://example.org/book/book1>
```

Absolute IRI

```
BASE <http://example.org/book/>
<book1>
```

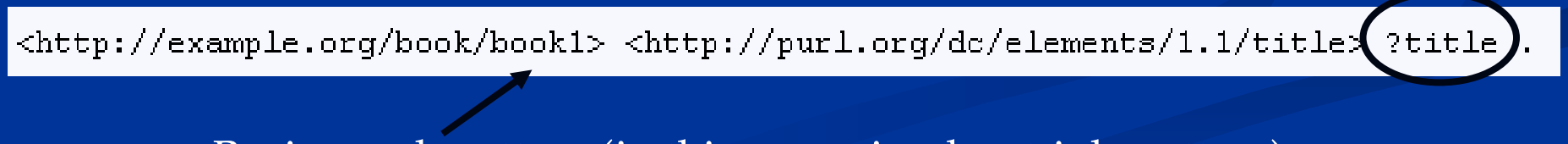
Relative IRI

```
PREFIX book: <http://example.org/book/>
book:book1
```

Prefixed name

SPARQL Syntax: Variables

- Most forms of SPARQL query contain a set of triple patterns called a *basic graph pattern*. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable
  - Use ? Or $ to denote variables
  - Example:



SPARQL Syntax: Literals

- Sometimes literals are included as part of SPARQL queries
- Literal syntax:
  - "chat" (a string)
  - 'chat'@ fr (a string with language tag "fr")
  - "xyz"^^http://example.org/ns/userDatatype (user datatype)
  - "abc"^^appNS:appDataType (another user datatype)
  - 1, which is the same as "1"^^xsd:integer
  - 1.300, which is the same as "1.300"^^xsd:decimal
  - 1.0e6, which is the same as "1.0e6"^^xsd:double
  - true, which is the same as "true"^^xsd:boolean

SPARQL Syntax: Hello World query

Data

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

SPARQL Query

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

Basic graph pattern

Results

title
"SPARQL Tutorial"

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Another example

RDF Data
(N3 format)

```
@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 }
```

SPARQL query

Results

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

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Queries with 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"@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>

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Abbreviations

- Predicate-object lists
  - For triples with the same subject

```
?x  foaf:name  ?name ;
    foaf:mbox  ?mbox .
```

- Object lists
  - For triples with the same subject and predicate

```
?x foaf:nick  "Alice" , "Alice_" .
```

- Abbreviation for rdf:type a

```
?x  a  :Class1 .
```

SPARQL: Restrictions (Filters)

- SPARQL FILTERs restrict solutions to those for which the filter expression evaluates to TRUE
- Filters can use functions (like regex or lang) and operators:
  - =, !=, <, <=, >=, >, &&, ||
- Restriction on solutions over the whole group of graph patterns in which the filter appears
  - The order of the filter in the group does not matter

SPARQL: Filters (I)

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

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

_:b  foaf:name       "Ms A.".
_:b  foaf:mbox       <mailto:alice@work.example> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name1 ?name2
 WHERE { ?x foaf:name  ?name1 ;
            foaf:mbox  ?mbox1 .
         ?y foaf:name  ?name2 ;
            foaf:mbox  ?mbox2 .
         FILTER (?mbox1 = ?mbox2 && ?name1 != ?name2)
       }
```

name1	name2
"Alice"	"Ms A."
"Ms A."	"Alice"

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Filters (II)

- Filtering using regular expresions (regex)
  - Works for string literals without language tag. In order to be used with other literals function str

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

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

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

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

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

SPARQL: Filters (III)

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

_:a  foaf:name       "Robert"@EN.
_:a  foaf:name       "Roberto"@ES.
_:a  foaf:mbox       <mailto:bob@work.example> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
 WHERE { ?x foaf:name  ?name ;
            foaf:mbox  ?mbox .
         FILTER ( lang(?name) = "ES" ) }
```

name	mbox
"Roberto"@ES	<mailto:bob@work.example>

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Optional (I)

- Regular, complete structures cannot be assumed in all RDF graphs
- It is useful to be able to have queries that allow information to be added to the solution where the information is available, but do not reject the solution because some part of the query pattern does not match Optional matching
- A graph pattern may have zero or more optional graph patterns

SPARQL: Optional (II)

```
@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"	

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Union (I)

- SPARQL provides a means of combining graph patterns so that one of several alternative graph patterns may match
- If more than one of the alternatives matches, all the possible pattern solutions are found
- Alternative graph patterns are represented with the keyword union

SPARQL: Union (II)

```
@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"

Source:
http://www.w3.org/TR/rdf-sparql-query/

Specifying the RDF Dataset (I)

- Many RDF data stores hold multiple RDF graphs and record information about each graph, allowing an application to make queries that involve information from more than one graph
- A SPARQL query may specify the dataset to be used for matching by using the FROM clause and the FROM NAMED clause

Specifying the RDF Dataset (II)

- FROM
  - An optional clause that provides the URI of the dataset to use, pointing to a local file or a URL of a graph somewhere on the Web
- FROM NAMED
  - Loads a graph and gives it a name (an IRI)
  - This name can be accessed from queries using GRAPH

Specifying the RDF Dataset (III)

```
# Default graph (stored at http://example.org/dft.ttl)
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://example.org/bob>    dc:publisher  "Bob Hacker" .
<http://example.org/alice>  dc:publisher  "Alice Hacker" .
```

```
# Named graph: http://example.org/bob
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .
```

```
# Named graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

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

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?who ?g ?mbox
FROM <http://example.org/dft.ttl>
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE
{
   ?g dc:publisher ?who .
   GRAPH ?g { ?x foaf:mbox ?mbox }
}
```

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Order By

- Establishes the order of a solution sequence
  - ASC() or DESC()

```
PREFIX     :     <http://example.org/ns#>
PREFIX foaf:     <http://xmlns.com/foaf/0.1/>
PREFIX xsd:      <http://www.w3.org/2001/XMLSchema#>

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

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Distinct

- Eliminates duplicate solutions

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

_:x    foaf:name   "Alice" .
_:x    foaf:mbox   <mailto:alice@example.com> .

_:y    foaf:name   "Alice" .
_:y    foaf:mbox   <mailto:asmith@example.com> .

_:z    foaf:name   "Alice" .
_:z    foaf:mbox   <mailto:alice.smith@example.com> .
```

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

name
"Alice"

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Limit

- Puts an upper bound on the number of solutions returned
- A LIMIT of 0 would cause no results to be returned. A limit may not be negative

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

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

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Offset

- Causes the solutions generated to start after the specified number of solutions
- An OFFSET of zero has no effect
- Using LIMIT and OFFSET requires the order to be made predictable by using ORDER BY

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

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

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Construct (I)

- CONSTRUCT
  - The CONSTRUCT query form returns a single RDF graph specified by a graph template. The result is an RDF graph formed by taking each query solution in the solution sequence, substituting for the variables in the graph template, and combining the triples into a single RDF graph by set union

SPARQL: Construct (II)

```
@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 }
```

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

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

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL: Ask (I)

- ASK
  - Applications can use the ASK query form to test whether or not a query pattern has a solution. No information is returned about the possible query solutions, just whether or not a solution exists
  - The result is thus “yes” /“no”

SPARQL: Ask (II)

```
@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" }
```

Source: http://www.w3.org/TR/rdf-sparql-query/

SPARQL Protocol for RDF

- Defines the protocol for issuing SPARQL queries over remote repositories and receiving the results
  - Defined using WSDL (as a Web Service)
  - The service contains one interface, SparqlQuery, which contains one operation, query
  - Bindings to HTTP and to SOAP are also defined

SPARQL Protocol for RDF

- The query operation…
  - … receives a query-request message that contains the query in SPARQL (a string) and information about the RDF datasets where the query is performed (default-graph-uri, named-graph-uri)
  - … returns a query-result message that contains..
    - SELECT and ASK queries a result set represented as an SPARQL results document in XML format (see next)
    - CONSTRUCT queries an RDF document
  - Faults are also defined
    - malformed-query, query-request-refused

SPARQL Query Results XML Format

- Defines an XML document format for representing the results of SPARQL SELECT and ASK queries
  - Three parts:
    - Sparql element
    - Header
    - Results

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#">

  <head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="age"/>
    <variable name="mbox"/>
    <variable name="friend"/>
  </head>

  <results>

    <result>
      <binding name="x">
       <bnode>r2</bnode>
      </binding>
      <binding name="hpage">
       <uri>http://work.example.org/bob/</uri>
      </binding>
      <binding name="name">
       <literal xml:lang="en">Bob</literal>
      </binding>
      <binding name="age">
       <literal datatype="http://www.w3.org/2001/XMLSchema#integer">30</literal>
      </binding>
      <binding name="mbox">
       <uri>mailto:bob@work.example.org</uri>
      </binding>
    </result>

    ...
  </results>

</sparql>
```

Source: http://www.w3.org/TR/rdf-sparql-XMLres/

Implementations

- Implementations
  - A list of related tools (clients, servers, parsers, …)
    - http://esw.w3.org/topic/SparqlImplementations
  - In the lab session we will use Jena with ARQ
    - http://jena.sourceforge.net/ARQ/

Applications

- SPARQL Endpoints:
  - DBLP: http://dblp.l3s.de/d2r/
  - DBPedia: http://dbpedia.org/sparql
- Garlik.com
  - DataPatrol: reports on personal information online
  - Uses SPARQL to build the reports
  - Thousands of users
  - Billion triples in several repositories
  - Reports generated in few seconds

RDF Storage

Approaches to store RDF (I)

- How do we store the RDF triples in a relational database?
- Triple store
  - Each RDF statement is stored as a single row in a three column 'statement' table
  - Typically, a fourth column is added to indicate if the object is a literal or a URI

Approaches to store RDF (II)

- Normalized triple store
  - Triple store plus two additional tables:
    - Resources table
    - Literals table
  - The statement table stores the subject, predicate and object as references to the values in the resources and literals tables
  - Less space (each literal or URI stored only once)
  - But complex queries (3-way join among the tables)
  - Jena 1 uses this approach

Approaches to store RDF (III)

- Denormalized triple store
  - Hybrid of the standard triple store and the normalized triple store
  - Three tables (statements, literals, resources)
  - Short URIs/literals are directly stored in the statements table, long ones in the respective tables
    - A threshold defines “long” vs “short” (Jena 2: 256)
  - More space than normalized
  - But in several cases the queries can be directly solved using only the information in the statements table
  - Used by Jena 2

References (I)

- SPARQL Query Language for RDF
  - W3C Recommendation 15 January 2008
    - http://www.w3.org/TR/rdf-sparql-query/
- SPARQL Protocol for RDF
  - W3C Recommendation 15 January 2008
    - http://www.w3.org/TR/rdf-sparql-protocol/
- SPARQL Query Results XML Format
  - W3C Recommendation 15 January 2008
    - http://www.w3.org/TR/rdf-sparql-XMLres/
- Jena2 Database Interface - Database Layout
    - http://jena.sourceforge.net/DB/layout.html

References (II)

- Semantics and Complexity of SPARQL
  - ISWC 2006: J. Perez, M. Arenas, C. Gutierrez
    - http://arxiv.org/abs/cs.DB/0605124
- SPARQL - Where are we? Current state, theory and practice
  - ESWC 2007 Tutorial
    - http://axel.deri.ie/~axepol/sparqltutorial/
- SPARQL Tutorial
  - Jena @ sourceforge
    - http://jena.sourceforge.net/ARQ/Tutorial/