Chapter 3
Querying RDF stores with SPARQL
We will want to query large RDF datasets, e.g. LOD.

SPARQL is the SQL of RDF.

SPARQL is a language to query and update triples in one or more triples stores.

It’s key to exploiting Linked Open Data.
Three RDF use cases

- **Markup web documents** with semi-structured data for better understanding by search engines (Microdata)
- Use as a *data interchange language* that’s more flexible and has a richer semantic schema than XML or SQL
- Assemble and link large datasets and publish as knowledge bases to support a domain (e.g., genomics) or in general (DBpedia)
Three RDF use cases

- Markup web documents with semi-structured data for better understanding by search engines (Microdata)
- Use as a data interchange language that's more flexible and has a richer semantic schema than XML or SQL
- Assemble and link large datasets and publish as knowledge bases to support a domain (e.g., genomics) or in general (DBpedia)
  - Such knowledge bases may be very large, e.g., Dbpedia has ~300M triples
  - Using such a large dataset requires a language to query and update it
Use Semantic Web Technology
to publish shared data &
knowledge

Semantic web technologies
allow machines to share
data and knowledge using
common web language and
protocols.

~ 1997
Use Semantic Web Technology to publish shared data & knowledge

Data is inter-linked to support integration and fusion of knowledge

Semantic Web => Linked Open Data

LOD beginning
Use Semantic Web Technology to publish shared data & knowledge.

Data is inter-linked to support integration and fusion of knowledge.

Semantic Web => Linked Open Data

2008

LOD growing
Semantic Web => Linked Open Data

Use Semantic Web Technology to publish shared data & knowledge

Data is inter-linked to support integration and fusion of knowledge

... and growing
Linked Open Data

Use Semantic Web Technology to publish shared data & knowledge

LOD is the new Cyc: a common source of background knowledge

Data is interlinked to support integration and fusion of knowledge
Linked Open Data

Use Semantic Web Technology to publish shared data & knowledge

LOD is the new Cyc: a common source of background knowledge

Data is inter-linked to support integration and fusion of knowledge

2011: 31B facts in 295 datasets interlinked by 504M assertions on ckan.net
Linked Open Data (LOD)

- **Linked data** is just RDF data, typically just the instances (ABOX), not schema (TBOX)
- RDF data is a graph of triples
  - URI URI string
    
    dbr:Barack_Obama dbo:spouse “Michelle Obama”
  - URI URI URI
    
    dbr:Barack_Obama dbo:spouse dbpedia:Michelle_Obama
- Best **linked** data practice prefers the 2\textsuperscript{nd} pattern, using nodes rather than strings for “entities”
- Liked **open** data is just linked data freely accessible on the Web along with any required ontologies
DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows you to make sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. We hope this will make it easier for the amazing amount of information in Wikipedia to be used in new and interesting ways, and that it might inspire new mechanisms for navigating, linking, and improving the encyclopedia itself.

**News**

**DBpedia 3.8 released, including enlarged Ontology and additional localized Versions**
Hi all, we are happy to announce the release of DBpedia 3.8. The most important improvements of the new release compared to DBpedia 3.7 are: 1. the new release is based on updated Wikipedia dumps dating from late May / early June 2012. 2. the DBpedia ontology is enlarged and the number of infobox to ontology mappings has [...] 

**DBpedia Spotlight has been selected for Google Summer of Code. Please apply now!**
The Google Summer of Code (GSoc) is a global program that offers student developers (BSc, MSc, PhD) stipends to write code for open source software projects. It has had thousands of participants since the first edition in 2005, connecting prospective students with mentors from open source communities such as Debian, KDE, Gnome, Apache Software Foundation, Mozilla, etc. For [...] 

**DBpedia 3.7 released, including 15 localized Editions**
Hi all, we are happy to announce the release of DBpedia 3.7. The new release is based on Wikipedia dumps dating from late July 2011. The new DBpedia data set describes more than 3.64 million things, of which 1.83 million are classified in a consistent ontology, including 416,000 persons, 526,000 places, 106,000 music albums, 60,000 films, 17,500 [...] 

---

**The DBpedia Knowledge Base**

Knowledge bases are playing an increasingly important role in enhancing the intelligence of Web and enterprise...
Available for download

3. Canonicalized Datasets

These datasets contain triples extracted from the respective Wikipedia whose subject and object resource have an equivalent English article.

All DBpedia IRIs/URIs in the canonicalized datasets use the generic namespace http://dbpedia.org/resource/. For backwards compatibility, the N-Triples files (.nt, .nq) use URIs, e.g. http://dbpedia.org/resource/Bras%C3%ADlia. The Turtle (.ttl) files use IRIs, e.g. http://dbpedia.org/resource/Brasilia.

NOTE: You can find DBpedia dumps in 111 languages at our DBpedia download server.

Click on the dataset names to obtain additional information. Click on the question mark next to a download link to preview file contents.

| Dataset                        | en | bg | ca | cs | de | el | es | fr | hu | it | ko | pl | pt | ru | sl | tr |
|-------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| **Ontology Infobox Types**    | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq |
| **Ontology Infobox Properties** | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq |
| **Ontology Infobox Properties (Specific)** | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq |
| **Titles**                    | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq |
| **Short Abstracts**           | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq |
| **Extended Abstracts**        | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq | nq |

- Broken up into files by information type
- Contains all text, links, infobox data, etc.
- Supported by several ontologies
- Updated ~ every 3 months
- About 300M triples!
Queryable

- You can query any of several RDF triple stores
- Or download the data, load into a store and query it locally
Browseable

There are also RDF browsers
These are driven by queries against a RDF triple store loaded with the DBpedia data
Why an RDF Query Language?

- Why not use an XML query language?
- XML at a lower level of abstraction than RDF
- There are various ways of syntactically representing an RDF statement in XML
- Thus we would require several XPath queries, e.g.
  - //uni:lecturer/uni:title if uni:title element
  - //uni:lecturer/@uni:title if uni:title attribute
- Both XML representations equivalent!
A key to exploiting such large RDF data sets is the SPARQL query language.

Sparql Protocol And Rdf Query Language

W3C began developing a spec for a query language in 2004.

There were/are other RDF query languages, and extensions, e.g., RQL and Jena’s ARQ.

SPARQL a W3C recommendation in 2008.

SPARQL 1.1 is a proposed recommendation with update, aggregation functions, federation & more.

Most triple stores support SPARQL 1.1.
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?age
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:age ?age
}
ORDER BY ?age DESC
LIMIT 10
SPARQL Protocol, Endpoints, APIs

- **SPARQL query language**
- **SPROT = SPARQL Protocol for RDF**
  - Among other things specifies how results can be encoded as RDF, XML or JSON
- **SPARQL endpoint**
  - A service that accepts queries and returns results via HTTP
  - Either generic (fetching data as needed) or specific (querying an associated triple store)
  - May be a service for federated queries
SPARQL Basic Queries

- SPARQL is based on matching graph patterns
- The simplest graph pattern is the triple pattern
  - "?person foaf:name ?name"
  - Like an RDF triple, but variables can be in any position
  - Variables begin with a question mark
- Combining triple patterns gives a graph pattern; an exact match to a graph is needed
- Like SQL, a set of results is returned with a result for each way the graph pattern can be instantiated
Turtle Like Syntax

As in Turtle and N3, we can omit a common subject in a graph pattern.

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?age
WHERE {
    ?person a foaf:Person;
    foaf:name ?name;
    foaf:age ?age
}
Optional Data

The query fails unless the entire pattern matches.

We often want to collect some information that might not always be available.

Note difference with relational model.

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?age
WHERE {
  ?person a foaf:Person;
    foaf:name ?name.
  OPTIONAL {?person foaf:age ?age}
}
Example of a Generic Endpoint

● Use the sparql endpoint at
  – http://demo.openlinksw.com/sparql

● To query graph at
  – http://ebiq.org/person/foaf/Tim/Finin/foaf.rdf

● For foaf knows relations
SELECT ?name ?p2
WHERE { ?person a foaf:Person;
  foaf:name ?name;
  foaf:knows ?p2. }
Example

Virtuoso SPARQL Query Editor

Default Data Set Name (Graph IRI)
http://ebsiquity.umbc.edu/person/foaf/Tim/Finin/foaf.rdf

Query Text

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?p2
WHERE {
  ?person a foaf:Person;
  foaf:name ?name;
}

Sponging:
Retrieve remote RDF data for all missing source graphs:

Results Format:
HTML

Execution timeout:
0 milliseconds (values less than 1000 are ignored)

Options:
Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see details)

Run Query  Reset
<table>
<thead>
<tr>
<th>name</th>
<th>p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Cynthia/Parr/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Cynthia/Parr/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/id/272/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/id/272/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Sheetal/Agarwal/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Sheetal/Agarwal/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Boanerges/Aleman-Meza/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Boanerges/Aleman-Meza/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Budak/Arpinar/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Budak/Arpinar/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Sasikanth/Avancha/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Sasikanth/Avancha/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Akram/Boughannam/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Akram/Boughannam/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Mark/Burstein/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Mark/Burstein/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Christoph/Bussler/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Christoph/Bussler/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Dipanjan/Chakraborty/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Dipanjan/Chakraborty/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Harry/Chen/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Harry/Chen/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Ye/Chen/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Ye/Chen/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Deepak/Chinavle/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Deepak/Chinavle/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Mohinder/Chopra/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Mohinder/Chopra/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Danielle/Chou/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Danielle/Chou/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Amit/Choudhri/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Amit/Choudhri/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Bill/Chu/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Bill/Chu/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Mark/Cornwell/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Mark/Cornwell/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/R/Scott/Cost/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/R/Scott/Cost/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Stephen/">http://ebiquity.umbc.edu/person/foaf/Stephen/</a> Cranefield/foaf.rdf#me</td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Grit/Denker/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Grit/Denker/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Marie/desJardins/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Marie/desJardins/foaf.rdf#me</a></td>
</tr>
<tr>
<td>Tim Finin</td>
<td><a href="http://ebiquity.umbc.edu/person/foaf/Redhika/Dhanukar/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Redhika/Dhanukar/foaf.rdf#me</a></td>
</tr>
</tbody>
</table>
Other result format options

- Auto
- HTML
- Spreadsheet
- XML
- JSON
- Javascript
- NTriples
- RDF/XML
- CSV
- CXML (Pivot Collection)
- CXML (Pivot Collection with QRcode)
Example of a dedicated Endpoint

- Use the sparql endpoint at
  - [http://dbpedia.org/sparql](http://dbpedia.org/sparql)
- To query DBpedia
- To discover places associated with President Obama

```sparql
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX dbpo: <http://dbpedia.org/ontology/>
SELECT distinct ?Property ?Place
  ?Place rdf:type dbpo:Place .}
```
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX dbpo: <http://dbpedia.org/ontology/>

SELECT distinct ?Property ?Place
   ?Place rdf:type dbpo:Place .}

<table>
<thead>
<tr>
<th>Property</th>
<th>Place</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://dbpedia.org/property/birthPlace">http://dbpedia.org/property/birthPlace</a></td>
<td><a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/property/nationality">http://dbpedia.org/property/nationality</a></td>
<td><a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/nationality">http://dbpedia.org/ontology/nationality</a></td>
<td><a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/birthplace">http://dbpedia.org/ontology/birthplace</a></td>
<td><a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a></td>
</tr>
</tbody>
</table>
SELECT FROM

- The FROM clause lets us specify the target graph in the query
- SELECT * returns all

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
FROM <http://ebiq.org/person/foaf/Tim/Finin/foaf.rdf>
WHERE {
  ?P1 foaf:knows ?p2
}
A generic web client

Try it: http://aers.data2semantics.org/yasgui/
Source: https://github.com/LaurensRietveld/yasgui
Find landlocked countries with a population >15 million

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
        rdfs:label ?country_name ;
        prop:populationEstimate ?population .
    FILTER (?population > 15000000) .
}
FILTER Functions

- Logical: !, &&, ||
- Math: +, -, *, /
- Comparison: =, !=, >, <, ...
- SPARQL tests: isURI, isBlank, isLiteral, bound
- SPARQL accessors: str, lang, datatype
- Other: sameTerm, langMatches, regex
- Conditionals (SPARQL 1.1): IF, COALESCE
- Constructors (SPARQL 1.1): URI, BNODE, STRDT, STRLANG
- Strings (SPARQL 1.1): STRLEN, SUBSTR, UCASE, ...
- More math (SPARQL 1.1): abs, round, ceil, floor, RAND
- Date/time (SPARQL 1.1): now, year, month, day, hours, ...
- Hashing (SPARQL 1.1): MD5, SHA1, SHA224, SHA256, ...
The UNION keyword forms a disjunction of two graph patterns

Both subquery results are included

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name
WHERE
{
  { [ ] foaf:name ?name } UNION { [ ] vCard:FN ?name }
}
Query forms

Each form takes a WHERE block to restrict the query

- **SELECT**: Extract raw values from a SPARQL endpoint, the results are returned in a table format
- **CONSTRUCT**: Extract information from the SPARQL endpoint and transform the results into valid RDF
- **ASK**: Returns a simple True/False result for a query on a SPARQL endpoint
- **DESCRIBE**: Extract an RDF graph from the SPARQL endpoint, the contents of which is left to the endpoint to decide based on what the maintainer deems as useful information
SPARQL 1.1 includes

- Updated 1.1 versions of SPARQL Query and SPARQL Protocol
- SPARQL 1.1 Update
- SPARQL 1.1 Graph Store HTTP Protocol
- SPARQL 1.1 Service Descriptions
- SPARQL 1.1 Entailments
- SPARQL 1.1 Basic Federated Query
An important use case for RDF is exploiting large collections of semi-structured data, e.g., the linked open data cloud.

We need a good query language for this.

SPARQL is the SQL of RDF.

SPARQL is a language to query and update triples in one or more triples stores.

It’s key to exploiting Linked Open Data.