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
- 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 ~500M triples, Freebase has ~3B, Google’s Knowledge Graph has 70B
  - Using such large datasets requires a language to query and update it
Semantic Web

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
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
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

LOD growing
Semantic Web => Linked Open Data

Use Semantic Web Technology to publish shared data & knowledge

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

2010...growing faster
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
The Linked Data Mug

See [Linked Data Rules](#), Tim Berners-Lee, circa 2006
Dbpedia: Wikipedia data in RDF
Available for download

- Broken up into files by information type
- Contains all text, links, infobox data, etc.
- Supported by several ontologies
- Updated ~ every 3 months
- ~500M triples for en
Queryable

• You can query any of several RDF triple stores
• Or download data, load into a store and query it locally
There are also RDF browsers driven by queries to an 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’d 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 and SPARQL 1.1 in 2013.

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 query language

● SPROT = SPARQL Protocol for RDF
  – Among other things specifies how results can be encoded as RDF, XML or JSON

● SPARQL endpoint
  – Service accepts queries, 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
- Simplest graph pattern is the triple pattern
  - ?person foaf:name ?name
  - Like an RDF triple, but with variables
  - Variables begin with a question mark
- Combining triple patterns gives a graph pattern; an exact match to a graph is needed
- Like SQL, returns a set of results, one 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

- Query fails unless the entire pattern matches
- We often want to collect 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}
}

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://ebiquity.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/Cranefield/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Stephen/Cranefield/foaf.rdf#me</a></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/Dhavarkar/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Redhika/Dhavarkar/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

● Discover places associated with Pres. Obama

```
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 .}
To use this you must know

- Know: RDF data model and SPARQL
- Know: Relevant ontology terms and CURIEs for individuals
- More difficult than for a typical database because the schema is so large
- Possible solutions:
  - Browse the KB to learn terms and individual CURIEs
  - Query using rdf:label and strings
  - Use Lushan Han’s intuitive KB (Han, 2013)
About: **Barack Obama**

An Entity of Type: [agent](http://dbpedia.org), from Named Graph: [http://dbpedia.org](http://dbpedia.org), within Data Space: [dbpedia.org](http://dbpedia.org)

Barack Hussein Obama II is the 44th and current President of the United States, in office since 2009. He is the first African American to hold the office. Born in Honolulu, Hawaii, Obama is a graduate of Columbia University and Harvard Law School, where he was president of the Harvard Law Review. He was a community organizer in Chicago before earning his law degree.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
</table>
- Barack Hussein Obama II is the 44th and current President of the United States. He is the first African American to hold the office. Born in Honolulu, Hawaii, Obama is a graduate of Columbia University and Harvard Law School, where he was president of the Harvard Law Review. He was a community organizer in Chicago before earning his law degree. He worked as a civil rights attorney in Chicago and taught constitutional law at the University of Chicago Law School from 1995 to 2004. He served three terms representing the 13th District in the Illinois House of Representatives from 2000 to 2004. In 2004, Obama represented Illinois in the United States Senate with his victory in the Illinois Democratic Primary. In July 2004, his election to the Senate was announced, and he was re-elected in 2008, after a close primary campaign against Hillary Clinton. |
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX dbpo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT distinct ?Property ?Place

WHERE {?P a dbpo:Person;
        rdfs:label "Barack Obama"@en;
        ?Property ?Place .
        ?Place rdf:type dbpo:Place .}
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX dbpo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT distinct ?P ?Property ?Place
WHERE {?P a dbpo:Person;
    rdfs:label ?Name.
    FILTER regex(?Name, 'obama', 'i')
    ?Place rdf:type dbpo:Place .
}
Structured Keyword Queries

- Nodes are entities and links binary relations
- Entities described by two unrestricted terms: *name* or value and *type* or concept
- Outputs marked with ?
- Compromise between a natural language Q&A system and formal query
  - Users provide compositional structure of the question
  - Free to use their own terms to annotate structure
**Translation result**

*Concepts:* Place $\Rightarrow$ Place, Author $\Rightarrow$ Writer, Book $\Rightarrow$ Book

*Properties:* born in $\Rightarrow$ birthPlace, wrote $\Rightarrow$ author (inverse direction)
The translation of a semantic graph query to SPARQL is straightforward given the mappings

**Concepts**
- Place => Place
- Author => Writer
- Book => Book

**Relations**
- born in => birthPlace
- wrote => author

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

SELECT DISTINCT ?x, ?y WHERE {
  ?0 a dbo:Book .
  ?0 rdfs:label ?label0 .
  ?label0 bif:contains "The adventures of Tom Sawyer" .
  ?x a dbo:Writer .
  ?y a dbo:Place .
  {?0 dbo:author ?x} .
  {?x dbo:birthPlace ?y} .
}
```
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
}

睿�
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, ...
UNION keyword forms 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 RDF graph from endpoint, the contents of which is left to the endpoint to decide based on what 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.