Chapter 3 Querying RDF stores with SPARQL

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!

Enter SPARQL

- 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, Jena's ARQ,
- SPARQL a W3C recommendation in 2008
  - Query language + protocol + xml result format
- SPARQL 1.1 currently a last-call working draft
  - Includes updates, aggregation functions, federation, …
- Most triple stores support SPARQL

SPARQL Example

```
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, not just one

Turtle Like Syntax

As in N# and Turtle, 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 patter matches
- We often want to collect some information that might not always be available
- Note difference with relational model

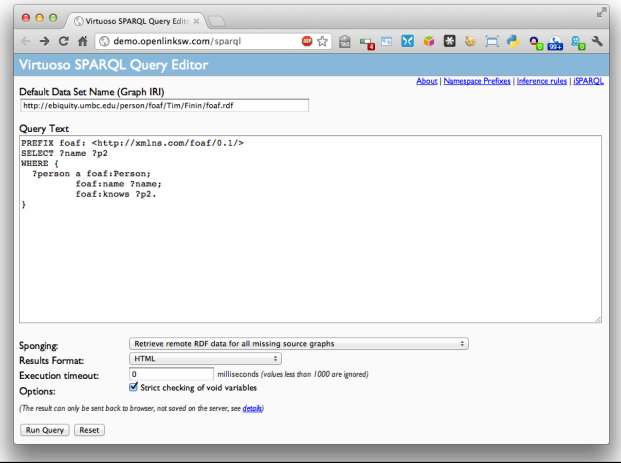
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?age
WHERE {
  ?person a foaf:Person;
            foaf:name ?name.
OPTIONAL {?person foaf:age ?age}
}
```

Example of a Generic Endpoint

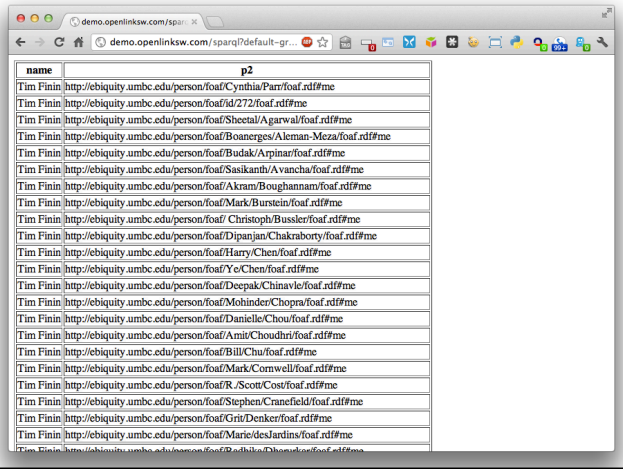
- Use the sparql endpoint at
  - http://demo.openlinksw.com/sparql
- To query graph at
  - http://ebiq.org/person/foaf/Tim/Finin/foaf.rdf
- For foaf knows relations

```
SELECT ?name ?p2
WHERE { ?person a foaf:Person;
                foaf:name ?name;
                foaf:knows ?p2.  }
```

Example

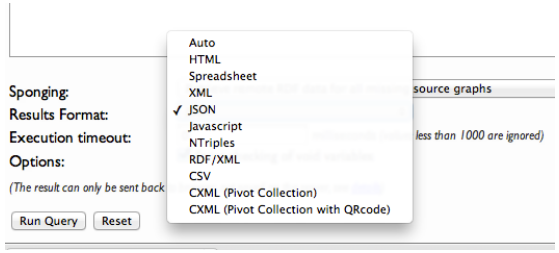


Query results as HTML



name	p2
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Cynthia/Parr/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/id/272/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Sheetal/Agarwal/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Boanerges/Aleman-Meza/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Budak/Arpinar/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Sasikanth/Avancha/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Akram/Boughannam/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Mark/Burstein/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/ Christoph/Bussler/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Dipanjan/Chakraborty/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Harry/Chen/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Ye/Chen/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Deepak/Chinavle/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Mohinder/Chopra/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Danielle/Chou/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Amit/Choudhri/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Bill/Chu/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Mark/Cornwell/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/R./Scott/Cost/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Stephen/Cranefield/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Grit/Denker/foaf.rdf#me
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Marie/desJardins/foaf.rdf#me

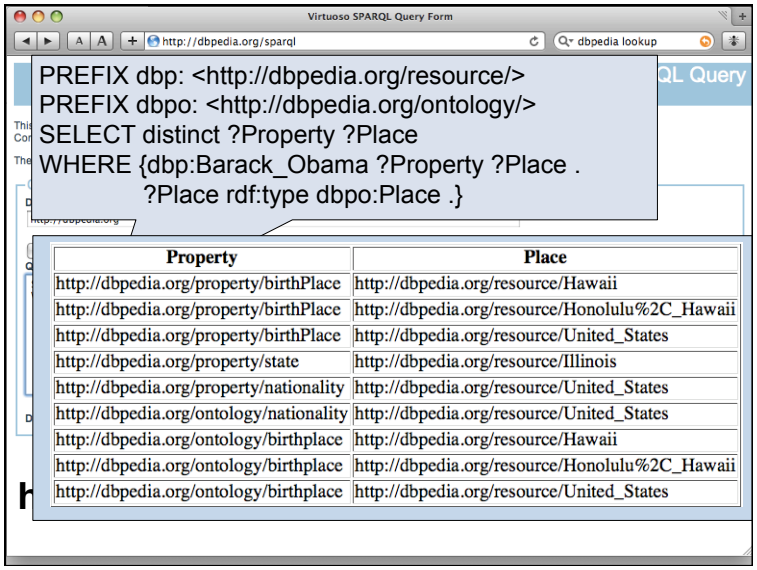
Other result format options



Example of a dedicated Endpoint

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

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



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
}
```

FILTER

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

- 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

SPARQL 1.1 is in last draft status & 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