Chapter 3
Querying RDF stores with SPARQL

TL;DR

- 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 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 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

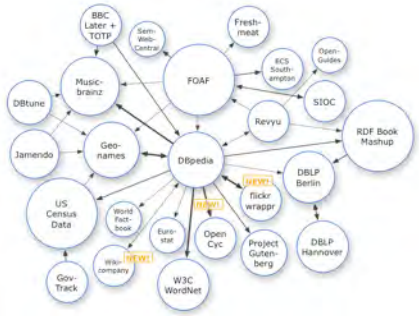
Semantic web technologies
allow machines to share data
and knowledge using common
web language and protocols.

~ 1997

Semantic Web beginning

Semantic Web => Linked Open Data

Use Semantic Web Technology to
publish shared data & knowledge



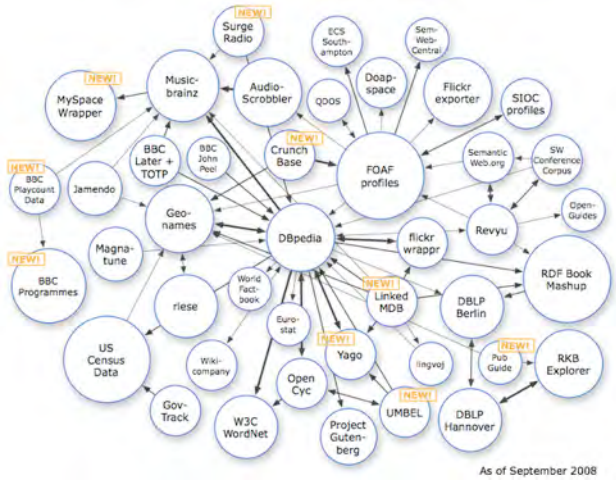
2007

Data is inter-
linked to support inte-
gration and fusion of knowledge

LOD beginning

Semantic Web => Linked Open Data

Use Semantic Web Technology to
publish shared data & knowledge



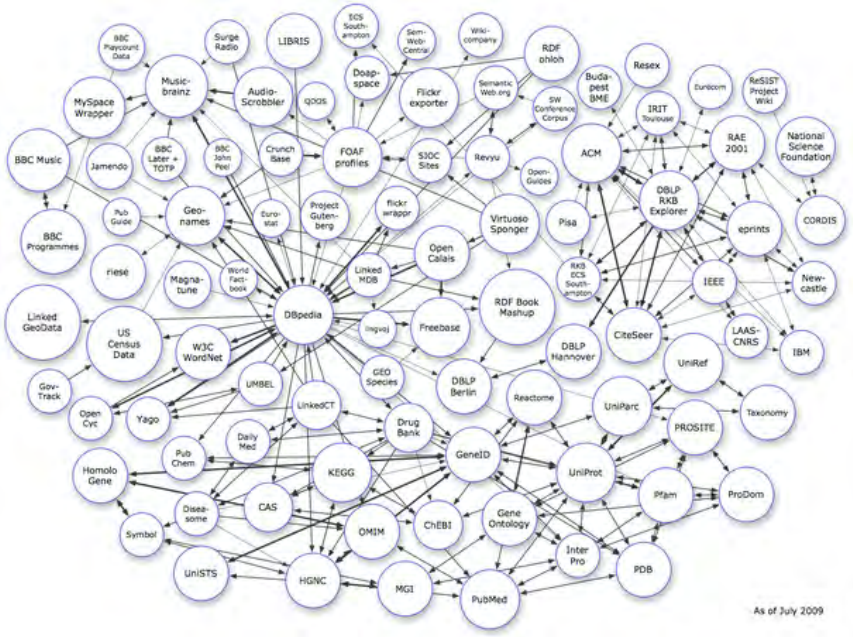
2008

Data is inter-
linked to support inte-
gration and fusion of knowledge

LOD growing

Semantic Web => Linked Open Data

Use Semantic Web Technology to
publish shared data & knowledge



2009

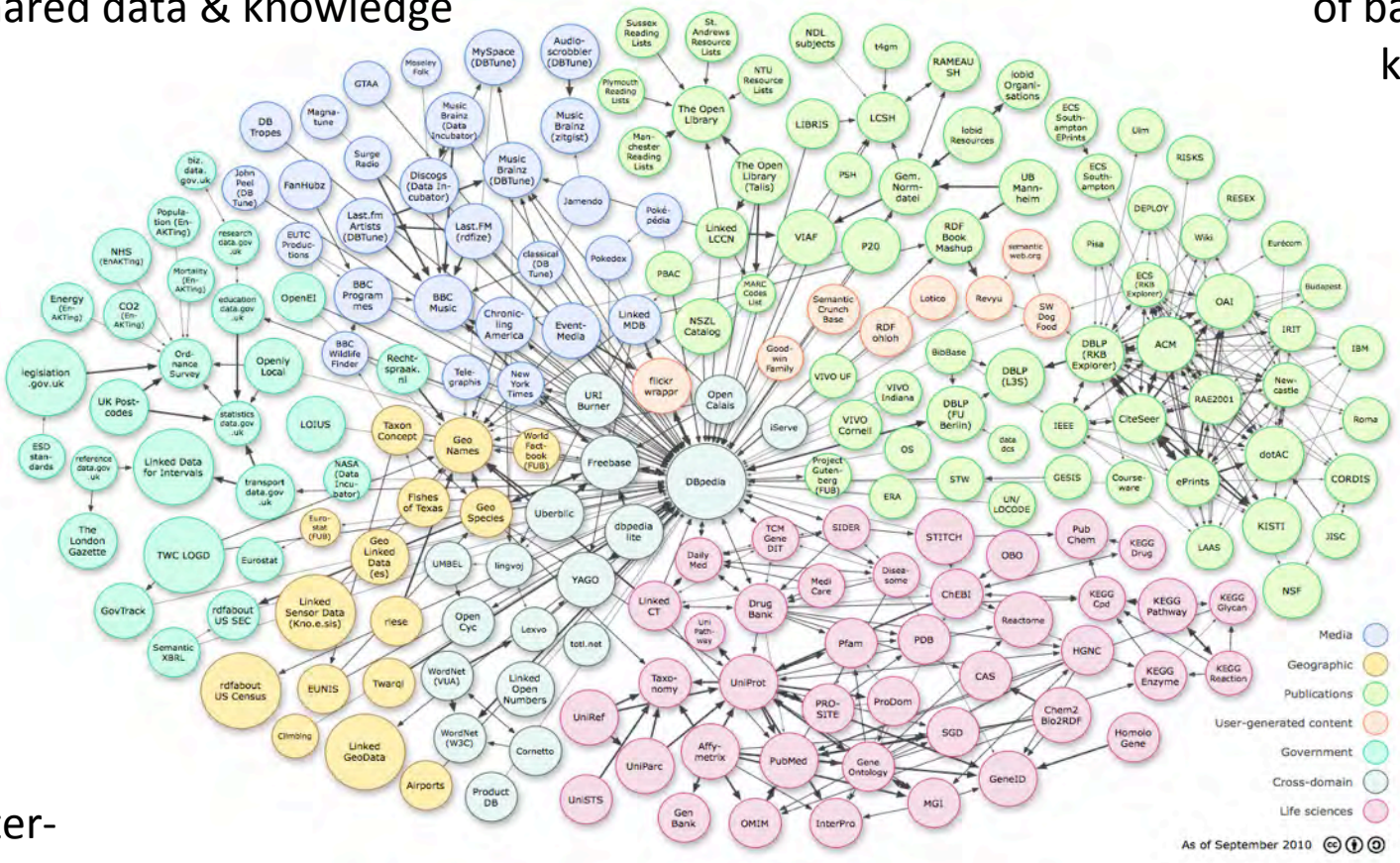
Data is inter-
linked to support inte-
gration and fusion of knowledge

… 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 inte-gration and fusion of knowledge

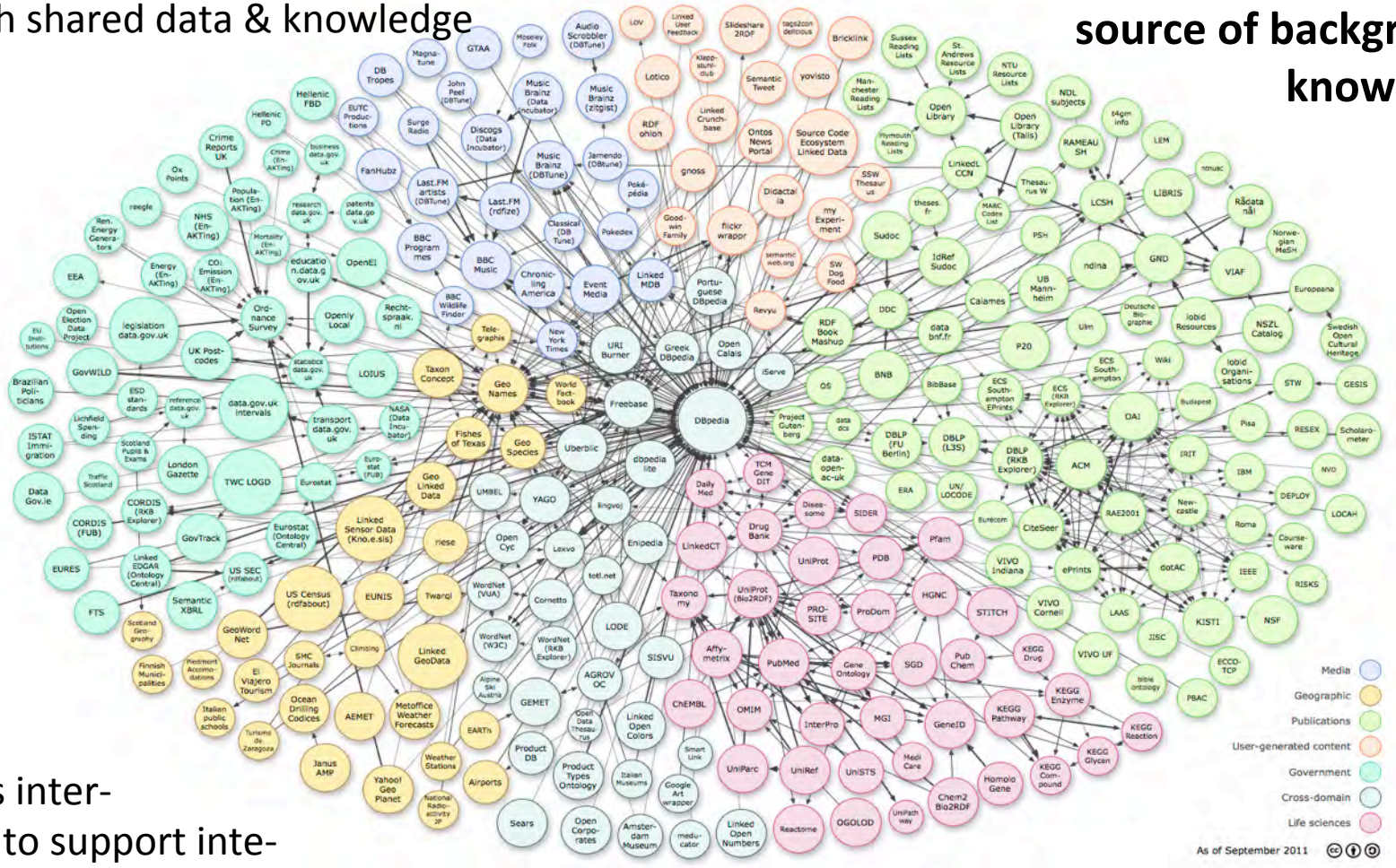
2010

...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 inte-gration and fusion of knowledge

2011: 31B facts in 295 datasets interlinked by 504M assertions on ckan.net

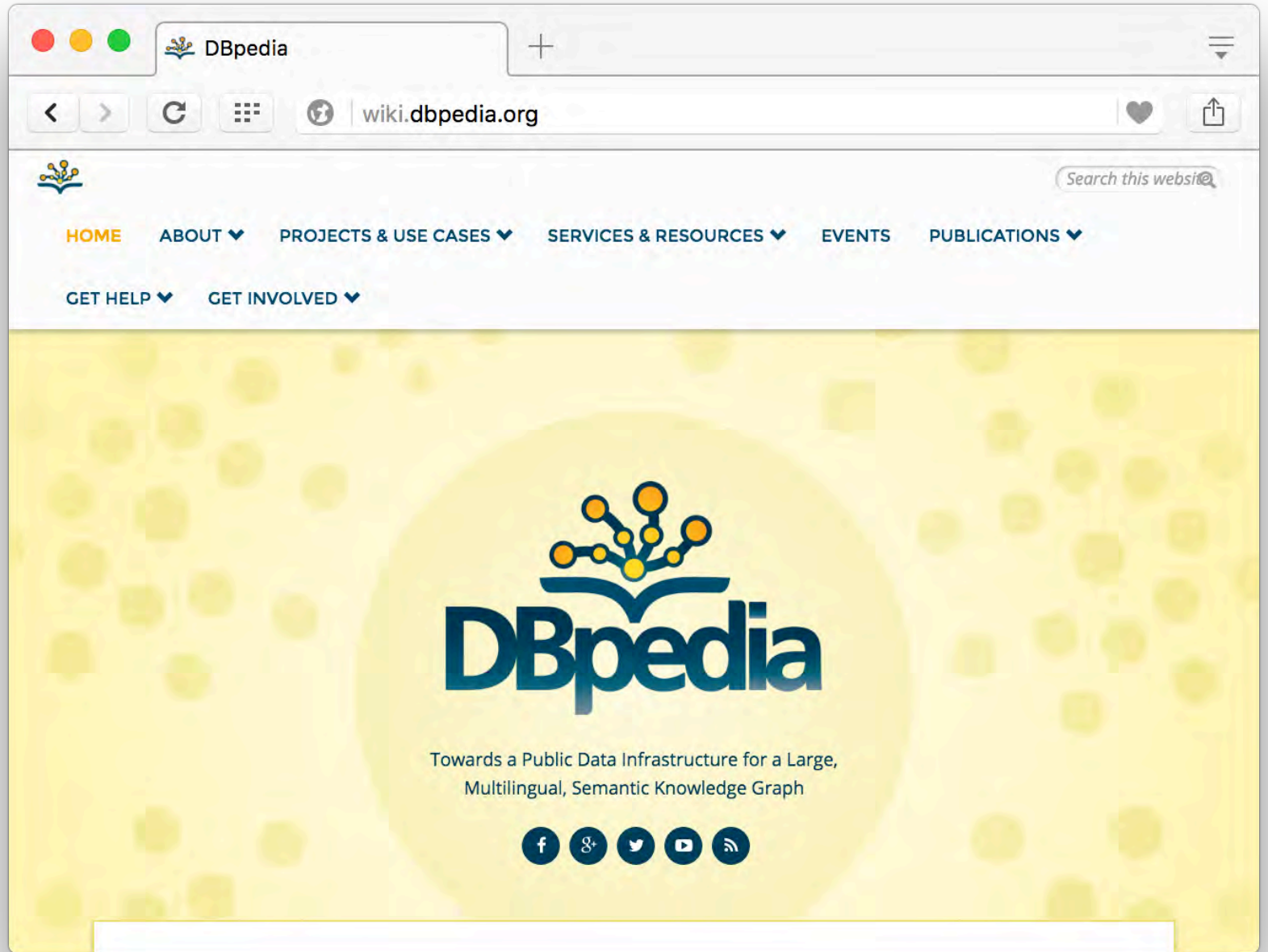
Linked Open Data (LOD)

- 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 2nd 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

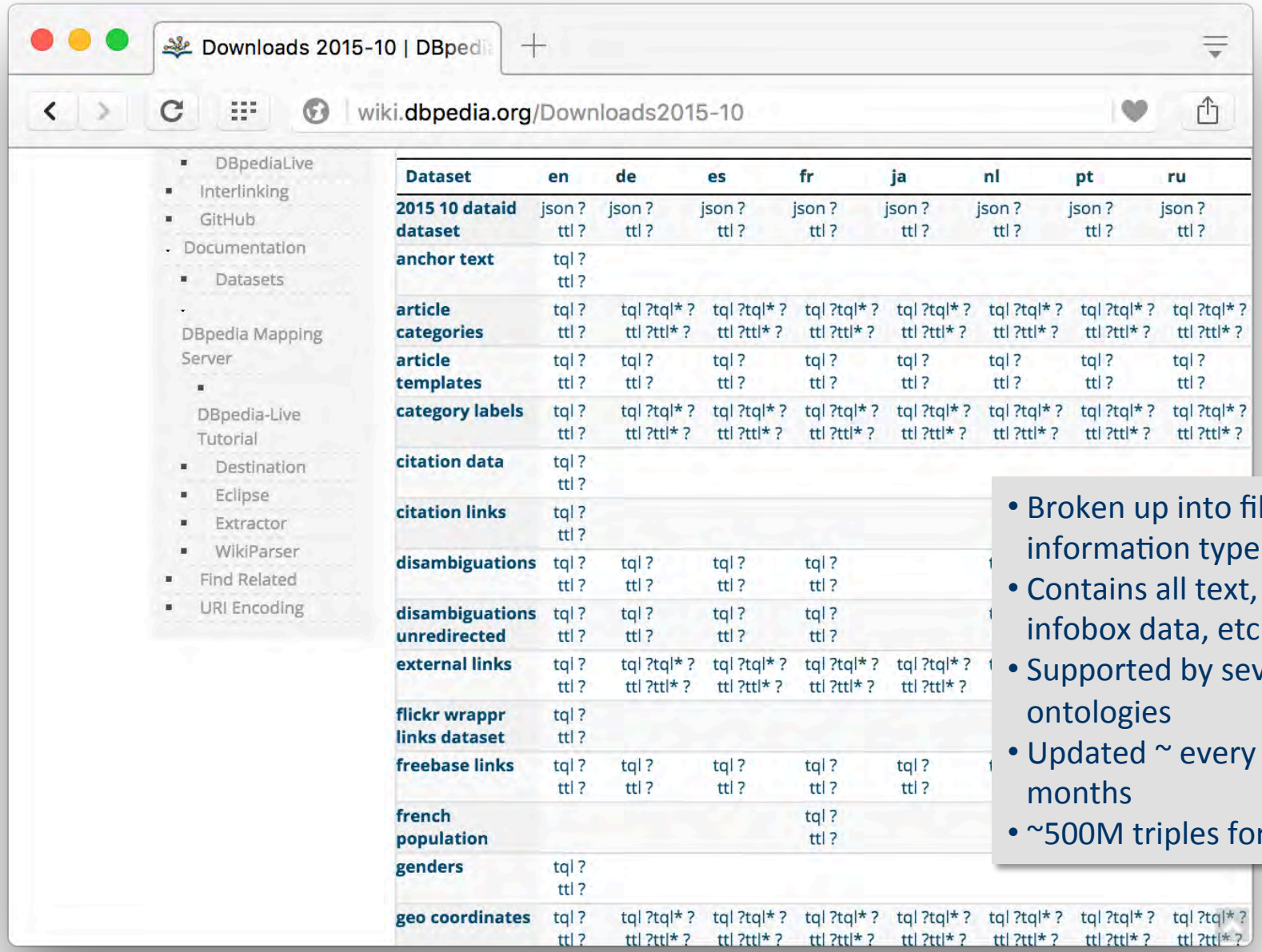
The Linked Data Mug

See Linked Data Rules, Tim Berners-Lee, circa 2006

Dbpedia: Wikipedia data in RDF



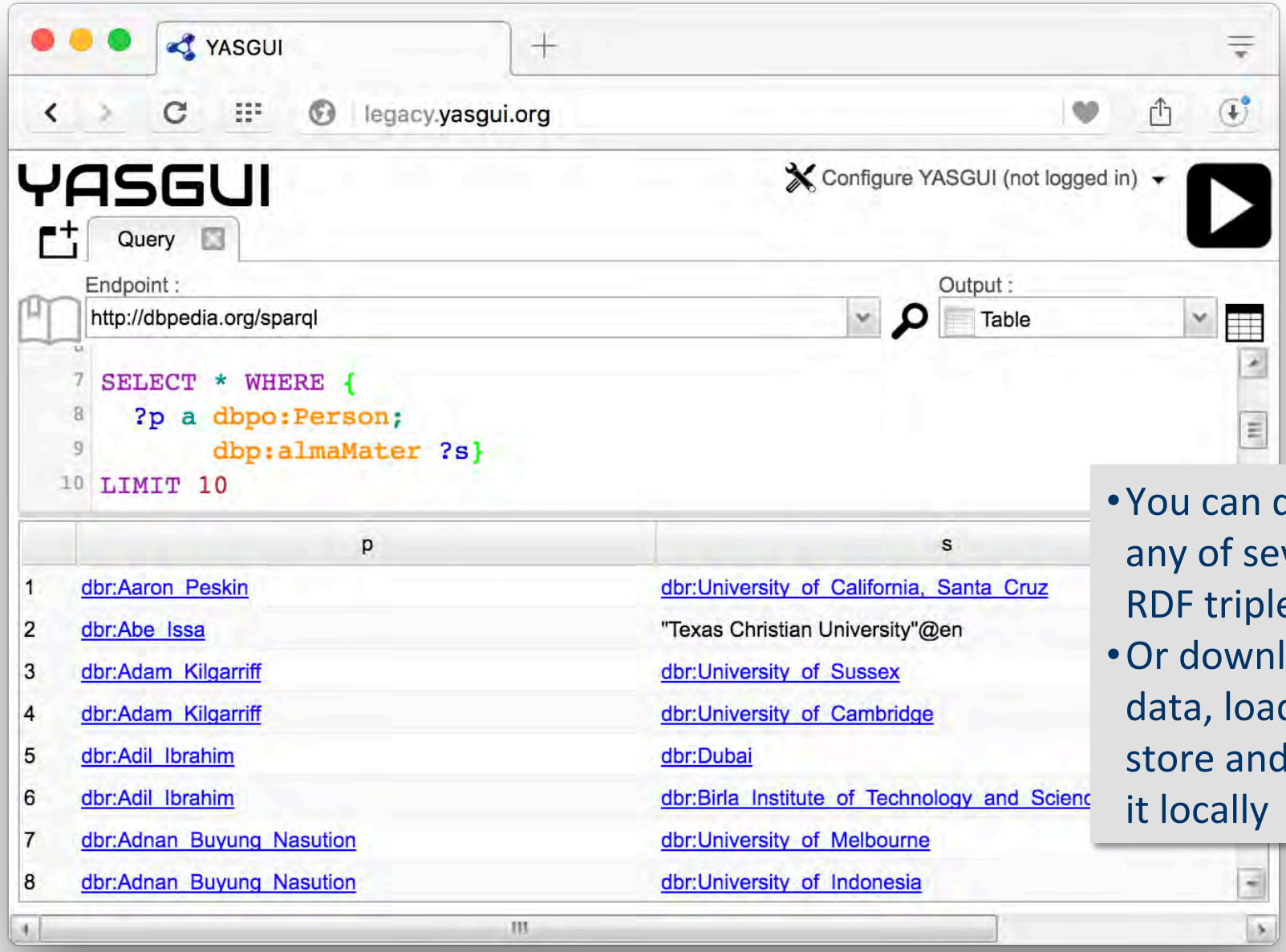
Available for download



Dataset	en	de	es	fr	ja	nl	pt	ru
2015 10 dataid dataset	json ? ttl ?	json ? ttl ?	json ? ttl ?	json ? ttl ?	json ? ttl ?	json ? ttl ?	json ? ttl ?	json ? ttl ?
anchor text	tql ? ttl ?							
article categories	tql ? ttl ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?
article templates	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?
category labels	tql ? ttl ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?
citation data	tql ? ttl ?							
citation links	tql ? ttl ?							
disambiguations	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?				
disambiguations unredirected	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?				
external links	tql ? ttl ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?			
flickr wrappr links dataset	tql ? ttl ?							
freebase links	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?	tql ? ttl ?			
french population				tql ? ttl ?				
genders	tql ? ttl ?							
geo coordinates	tql ? ttl ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?	tql ?tql* ? ttl ?ttl* ?

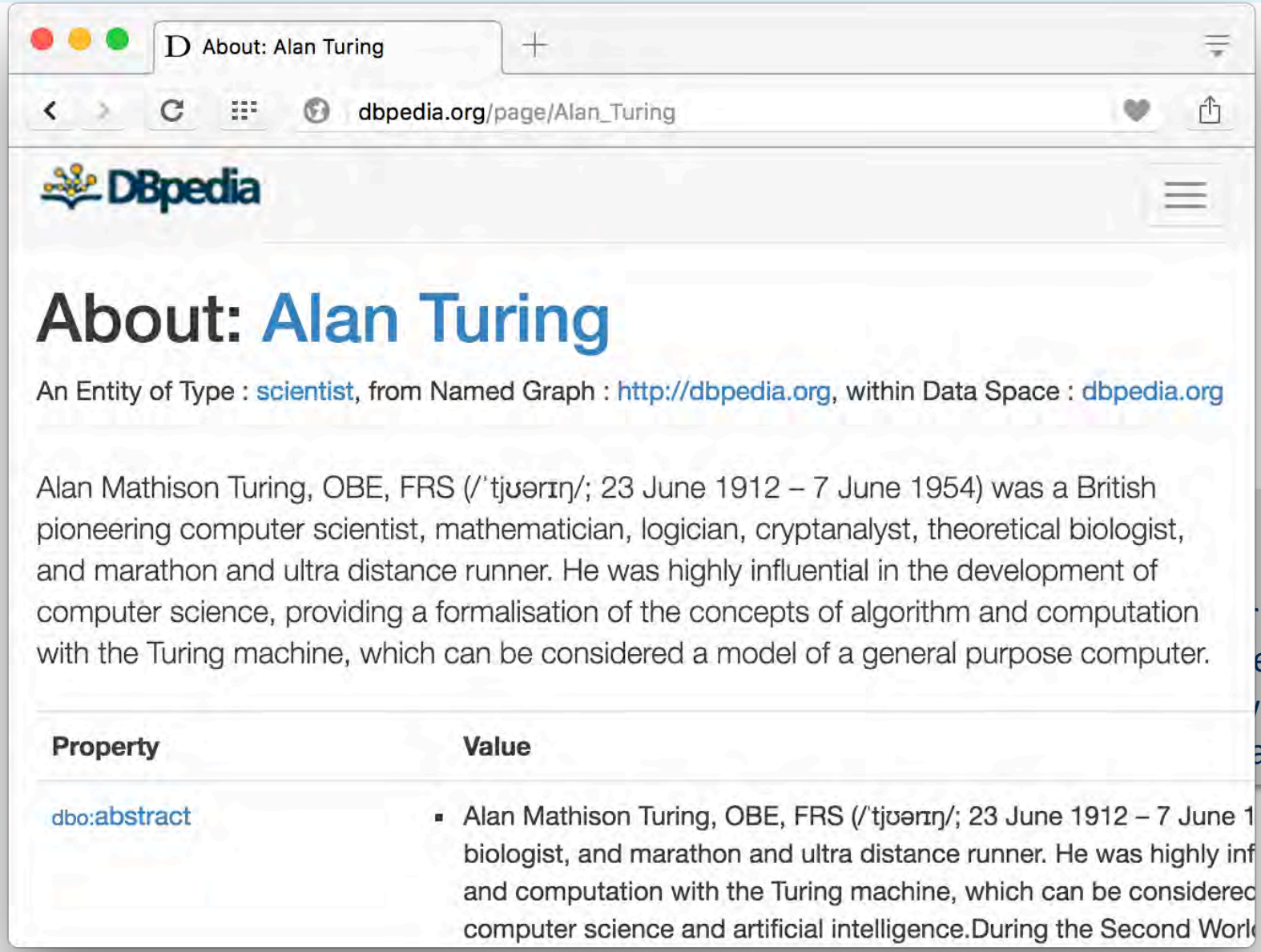
- Broken up into files by information type
- Contains all text, links, infobox data, etc.
- Supported by several ontologies
- Updated ~ every 3 months
- ~500M triples for en

Queryable



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

Browseable



RDF

ies

e

ith

ata

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!

SPARQL

- A key to exploiting such large RDF data sets is the SPARQL query language
- **S**parql **P**rotocol **A**nd **R**df **Q**uery **L**anguage
- 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

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 uses a Turtle like syntax

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
  - 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 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

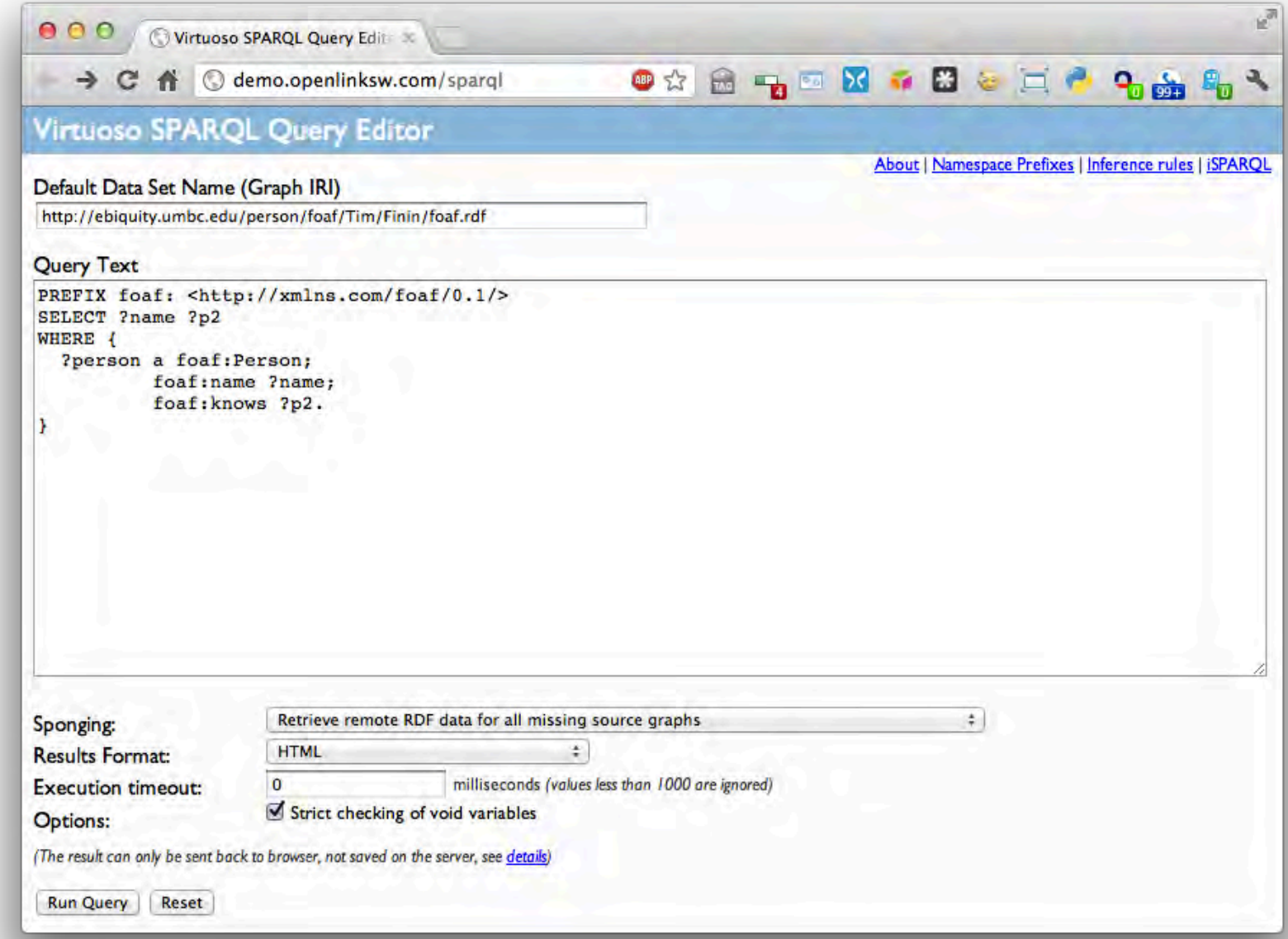
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?age
WHERE {
  ?person a foaf:Person;
            foaf:name ?name.
OPTIONAL {?person foaf:age ?age}
}
```

Example of a Generic Endpoint

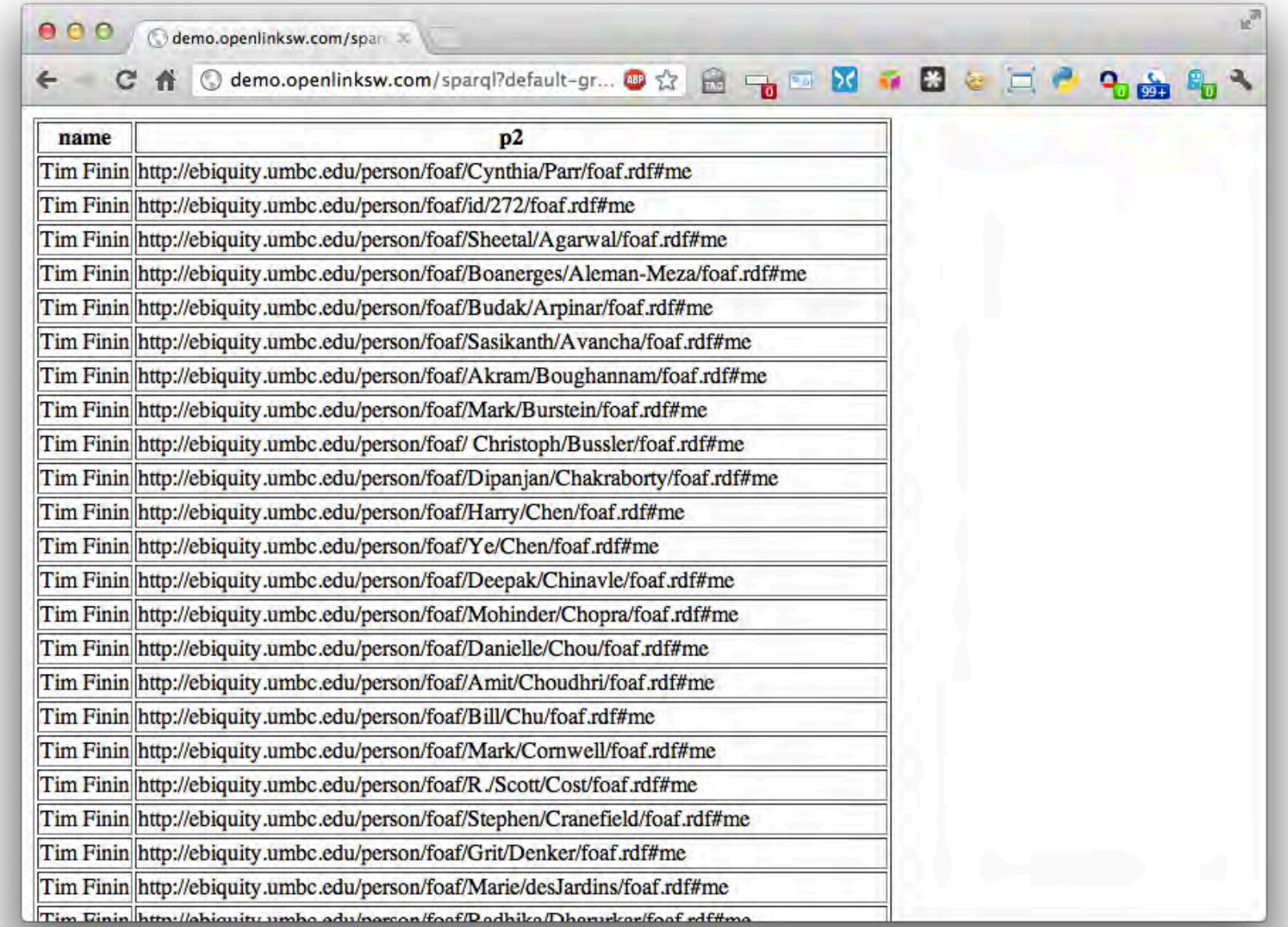
- Use the sparql endpoint at
  - http://demo.openlinksw.com/sparql
- To query graph at
  - http://ebiq.org/person/foaf/Tim/Finin/foaf.rdf
- For foaf knows relations

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

Example

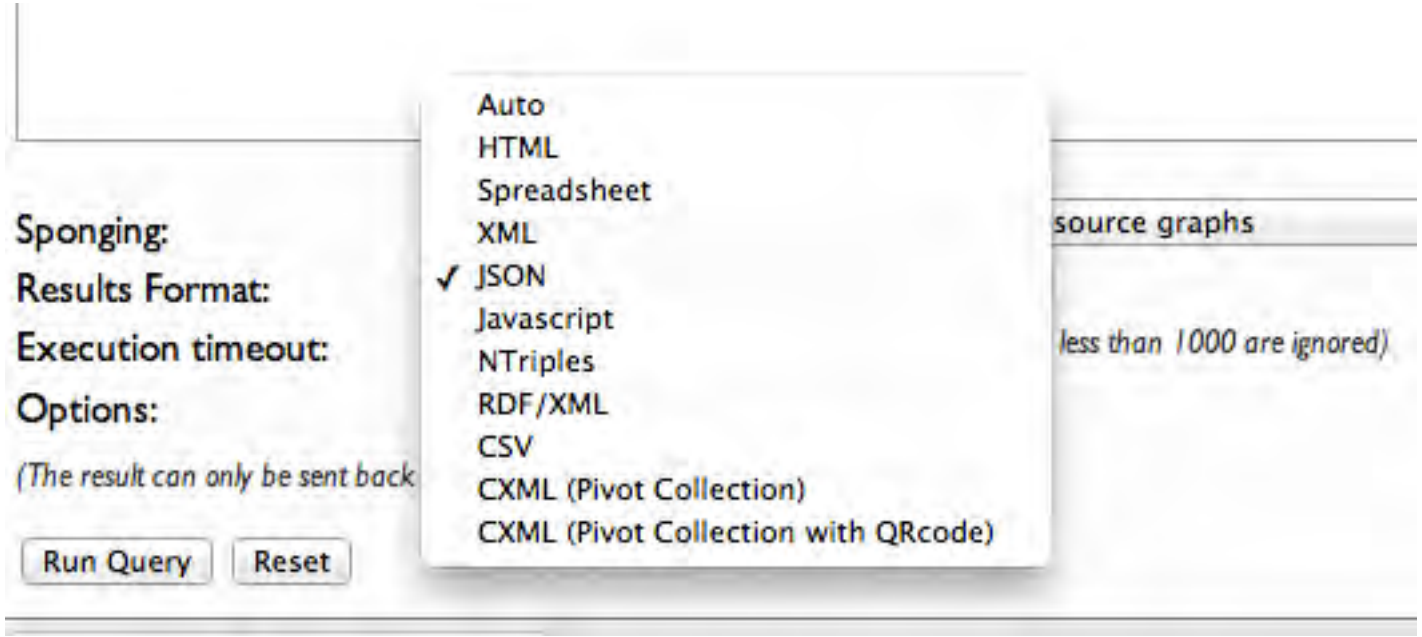

Virtuoso SPARQL Query Edit
demo.openlinksw.com/sparql
Virtuoso SPARQL Query Editor
About | Namespace Prefixes | Inference rules | iSPARQL
Default Data Set Name (Graph IRI)
http://ebiquity.umbc.edu/person/foaf/Tim/Finin/foaf.rdf
Query Text
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?p2
WHERE {
?person a foaf:Person;
foaf:name ?name;
foaf:knows ?p2.
}
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

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
Tim Finin	http://ebiquity.umbc.edu/person/foaf/Radhika/Dhamankar/foaf.rdf#me

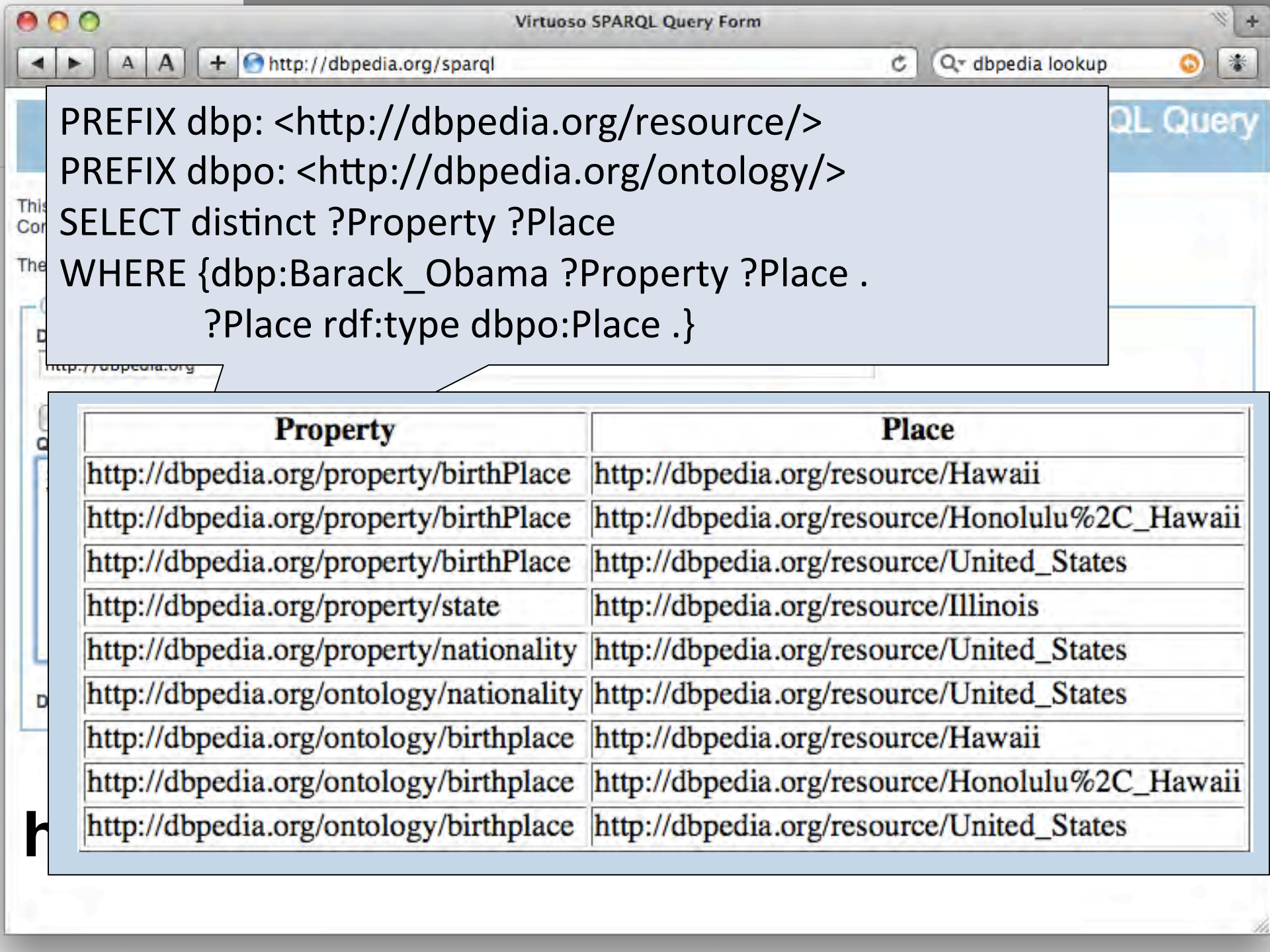
Other result format options



Example of a dedicated Endpoint

- Use the sparql endpoint at
  - 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
WHERE {dbp:Barack_Obama ?Property ?Place .
        ?Place rdf:type dbpo:Place .}
```



Property	Place
http://dbpedia.org/property/birthPlace	http://dbpedia.org/resource/Hawaii
http://dbpedia.org/property/birthPlace	http://dbpedia.org/resource/Honolulu%2C_Hawaii
http://dbpedia.org/property/birthPlace	http://dbpedia.org/resource/United_States
http://dbpedia.org/property/state	http://dbpedia.org/resource/Illinois
http://dbpedia.org/property/nationality	http://dbpedia.org/resource/United_States
http://dbpedia.org/ontology/nationality	http://dbpedia.org/resource/United_States
http://dbpedia.org/ontology/birthplace	http://dbpedia.org/resource/Hawaii
http://dbpedia.org/ontology/birthplace	http://dbpedia.org/resource/Honolulu%2C_Hawaii
http://dbpedia.org/ontology/birthplace	http://dbpedia.org/resource/United_States

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)

Search for: dbpedia barack obama

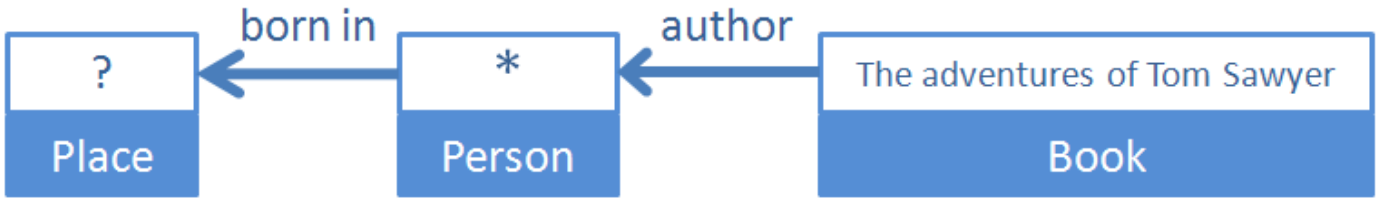
Query using labels

```
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 .}
```

Query using labels and FILTER

```
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')
      ?P ?Property ?Place .
      ?Place rdf:type dbpo:Place .
}
```

Structured Keyword Queries

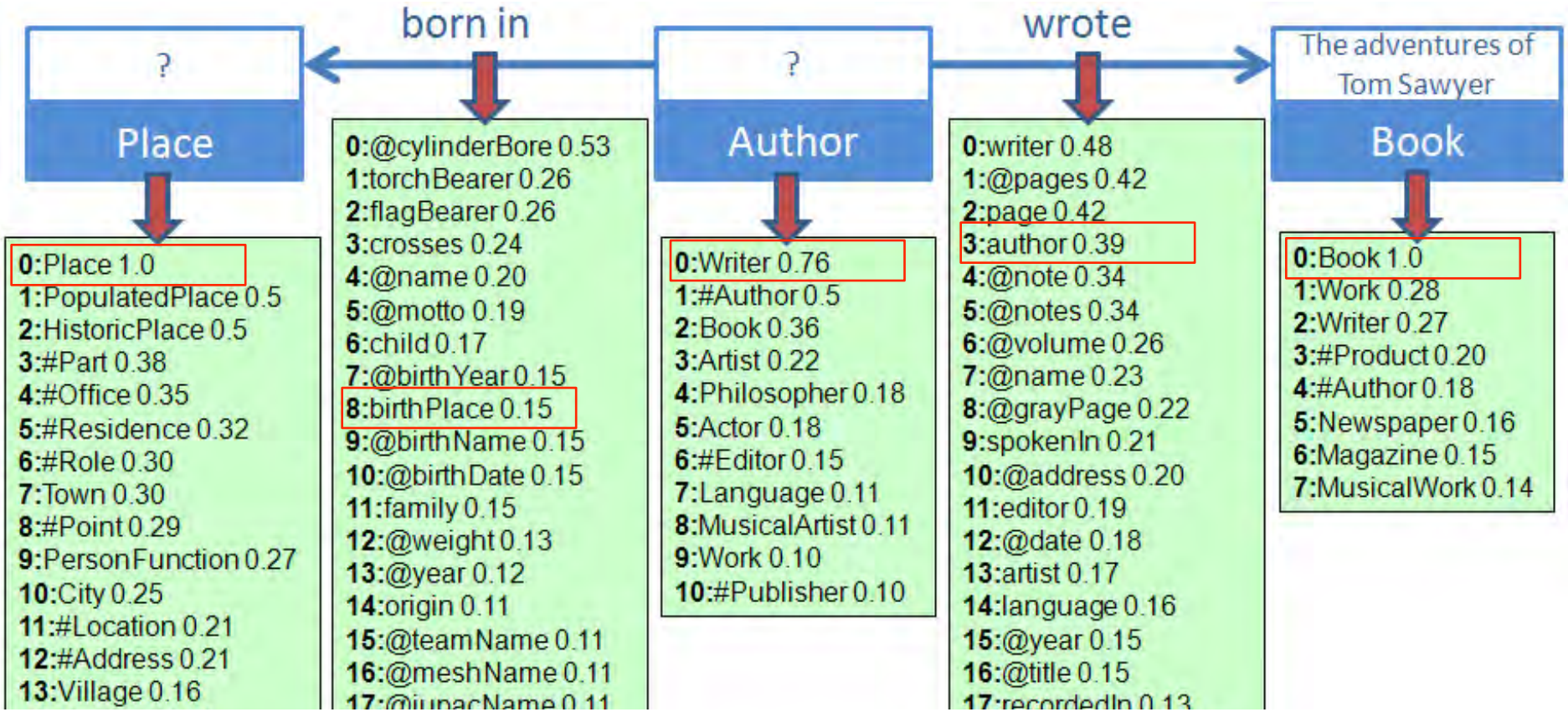


- Nodes are entities and links binary relations
- Entities described by two unrestricted terms: *name* or value and *type* or concept
- Outputs marked with *?*
- Compromise between a natural language Q&A system and formal query
  - Users provide compositional structure of the question
  - Free to use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

Translation result

Concepts: Place => Place, Author => Writer, Book => Book

Properties: born in => birthPlace, wrote => author (inverse direction)



SPARQL Generation

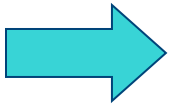
The translation of a semantic graph query to SPARQL is straightforward given the mappings

Concepts

- Place => Place
- Author => Writer
- Book => Book

Relations

- born in => birthPlace
- wrote => author



```
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
}
```

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

- 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

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

Summary

- An important usecase 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