

# 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 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

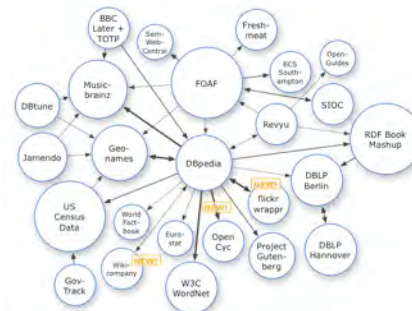
Semantic web technologies  
allow machines to share data  
and knowledge using common  
web language and protocols.

~ 1997

Semantic Web beginning

# Semantic Web => Linked Open Data

Use Semantic Web Technology to publish shared data & knowledge



2007

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

LOD growing

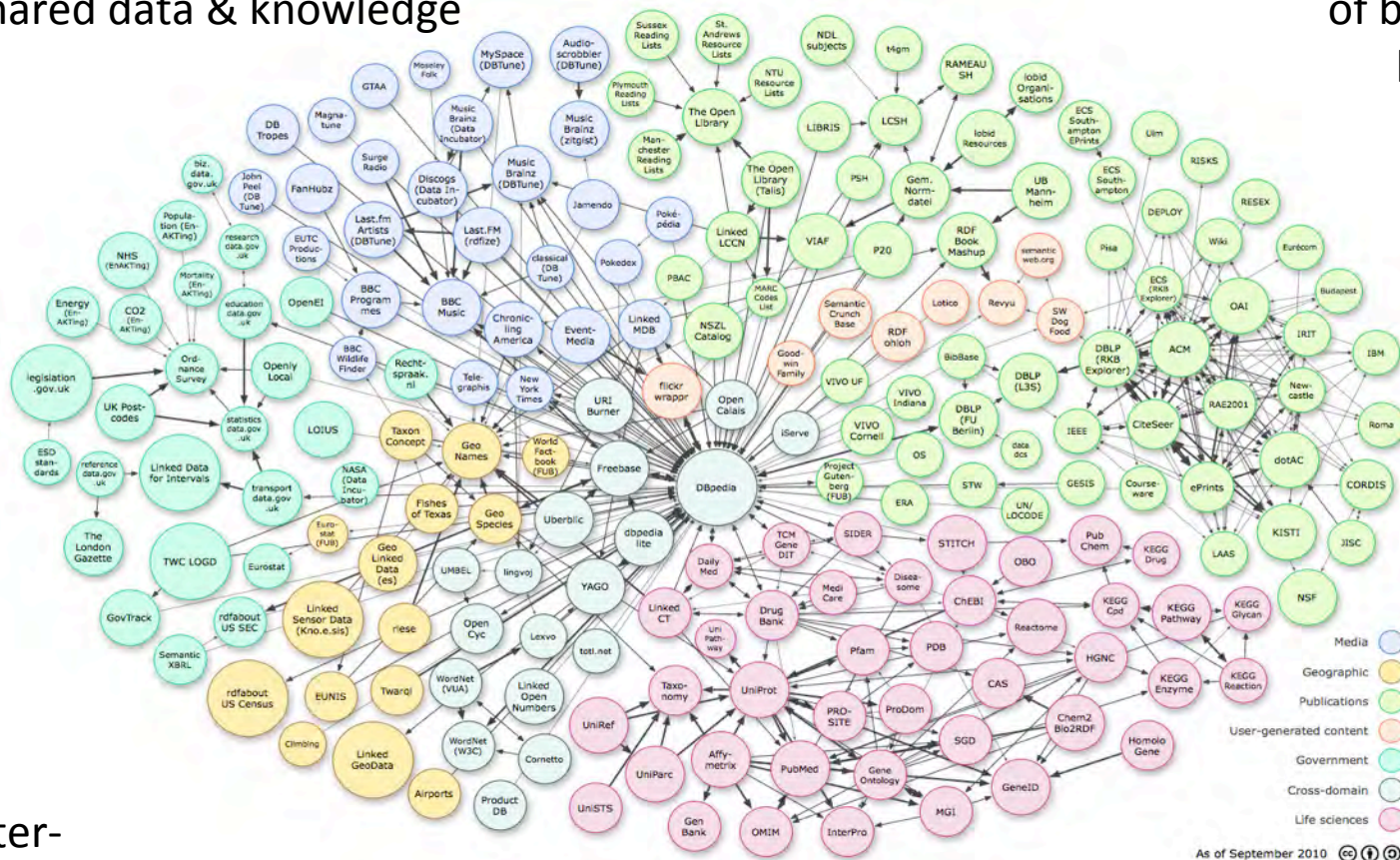




# Linked Open Data

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 (LOD)



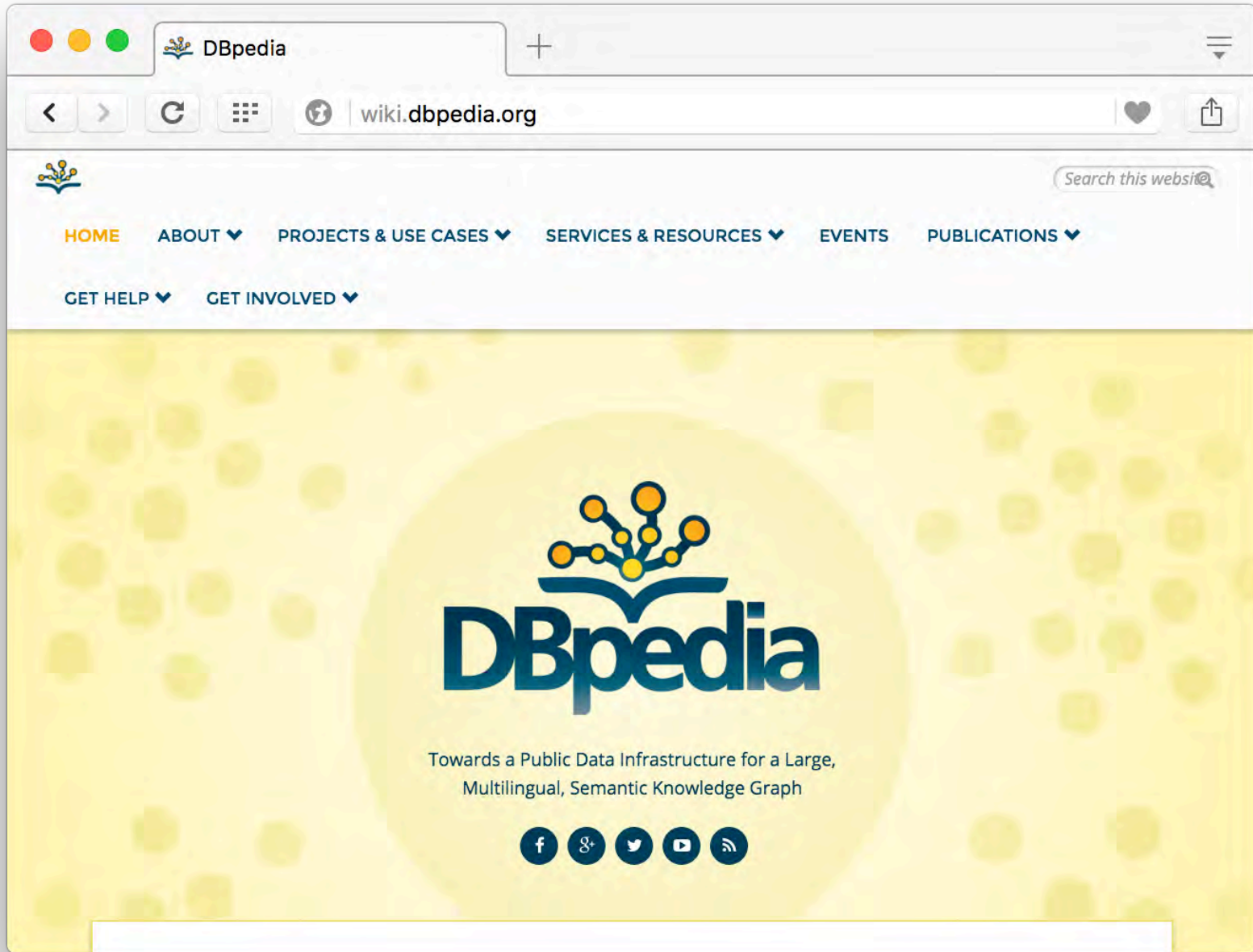
- Linked **data** is just RDF data, typically just the instances ([ABOX](#)), not schema ([TBOX](#))
- RDF data is a graph of triples
  - URI URI string  
dbr:Barack\_Obama dbo:spouse “Michelle Obama”
  - URI URI URI  
dbr:Barack\_Obama dbo:spouse dbpedia:Michelle\_Obama
- Best **linked** data practice prefers the 2<sup>nd</sup> 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



The image shows a browser window displaying the DBpedia website. The browser's address bar shows the URL `wiki.dbpedia.org`. The website's header includes a search bar with the placeholder text "Search this website" and a navigation menu with the following items: HOME, ABOUT, PROJECTS & USE CASES, SERVICES & RESOURCES, EVENTS, PUBLICATIONS, GET HELP, and GET INVOLVED. The main content area features a large yellow background with a circular logo in the center. The logo consists of a stylized tree-like structure with orange nodes and blue lines, positioned above the text "DBpedia" in a bold, blue, sans-serif font. Below the logo, the text "Towards a Public Data Infrastructure for a Large, Multilingual, Semantic Knowledge Graph" is displayed. At the bottom of the logo area, there are five social media icons: Facebook, Google+, Twitter, YouTube, and RSS.

# Available for download

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

The screenshot shows the YASGUI web interface. The browser address bar displays 'legacy.yasgui.org'. The main header includes the 'YASGUI' logo and a 'Configure YASGUI (not logged in)' button. Below the header, there is a 'Query' tab and an 'Endpoint' field containing 'http://dbpedia.org/sparql'. The 'Output' field is set to 'Table'. The query text is as follows:

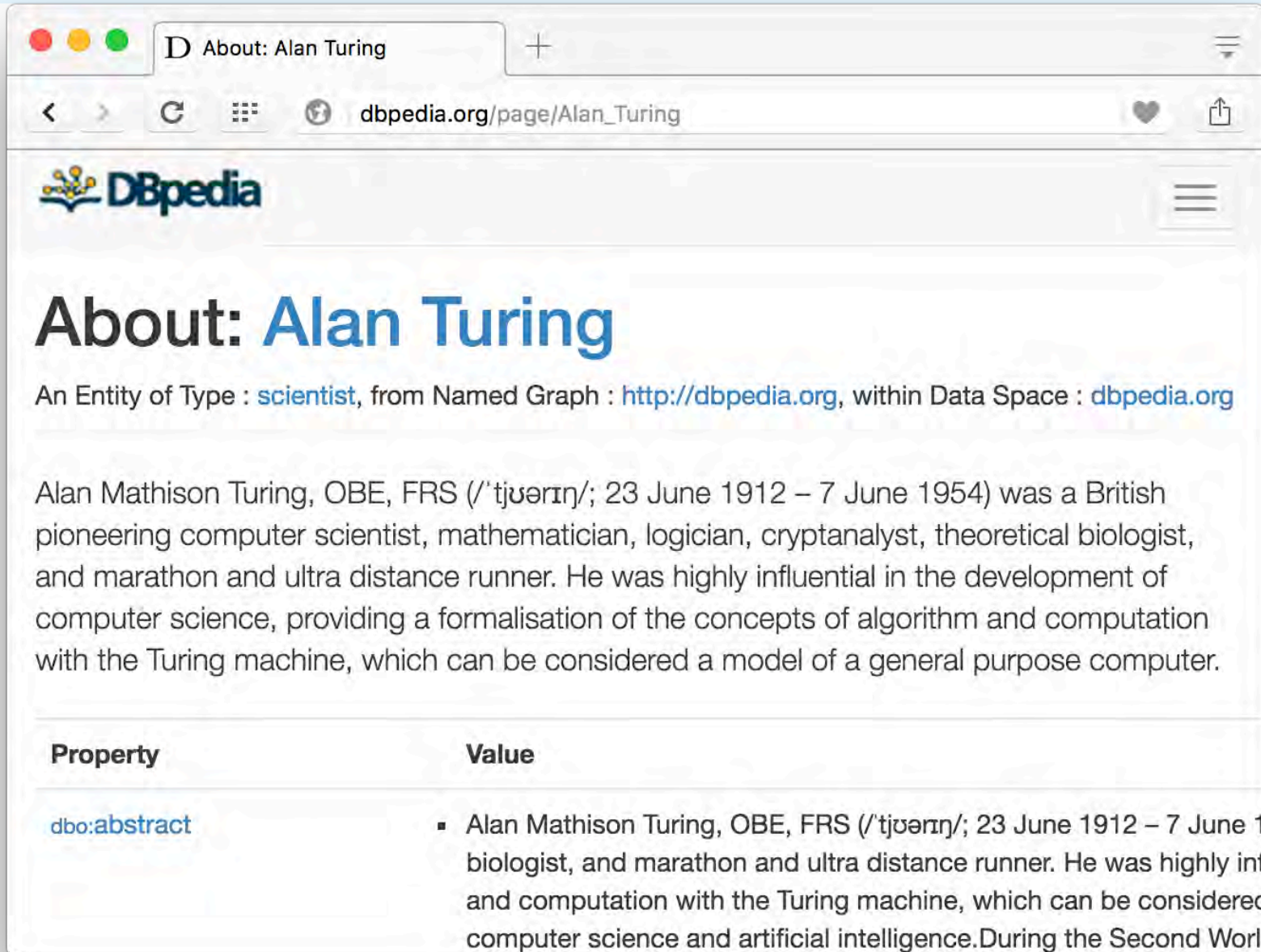
```
7 SELECT * WHERE {  
8   ?p a dbpo:Person;  
9     dbp:almaMater ?s}  
10 LIMIT 10
```

The results are displayed in a table with two columns: 'p' and 's'. The first eight rows of results are shown below:

	p	s
1	<a href="#">dbr:Aaron_Peskin</a>	<a href="#">dbr:University_of_California,_Santa_Cruz</a>
2	<a href="#">dbr:Abe_Issa</a>	"Texas Christian University"@en
3	<a href="#">dbr:Adam_Kilgarriff</a>	<a href="#">dbr:University_of_Sussex</a>
4	<a href="#">dbr:Adam_Kilgarriff</a>	<a href="#">dbr:University_of_Cambridge</a>
5	<a href="#">dbr:Adil_Ibrahim</a>	<a href="#">dbr:Dubai</a>
6	<a href="#">dbr:Adil_Ibrahim</a>	<a href="#">dbr:Birla_Institute_of_Technology_and_Science</a>
7	<a href="#">dbr:Adnan_Buyung_Nasution</a>	<a href="#">dbr:University_of_Melbourne</a>
8	<a href="#">dbr:Adnan_Buyung_Nasution</a>	<a href="#">dbr:University_of_Indonesia</a>

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

# Browseable



The image shows a browser window with the address bar containing "dbpedia.org/page/Alan\_Turing". The page title is "About: Alan Turing". The DBpedia logo is visible in the top left. The main heading is "About: Alan Turing". Below the heading, there is a line of text: "An Entity of Type : [scientist](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)".

Alan Mathison Turing, OBE, FRS (/ˈtʃʊərɪŋ/; 23 June 1912 – 7 June 1954) was a British pioneering computer scientist, mathematician, logician, cryptanalyst, theoretical biologist, and marathon and ultra distance runner. He was highly influential in the development of computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general purpose computer.

Property	Value
<a href="#">dbo:abstract</a>	<ul style="list-style-type: none"><li>Alan Mathison Turing, OBE, FRS (/ˈtʃʊərɪŋ/; 23 June 1912 – 7 June 1954) was a British pioneering computer scientist, mathematician, logician, cryptanalyst, theoretical biologist, and marathon and ultra distance runner. He was highly influential in the development of computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general purpose computer.</li></ul>

RDF  
ies  
e  
with  
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
- **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

# 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

The screenshot shows the Virtuoso SPARQL Query Editor interface. The browser address bar displays `demo.openlinksw.com/sparql`. The page title is "Virtuoso SPARQL Query Editor".

**Default Data Set Name (Graph IRI)**  
A text input field contains the URL: `http://ebiquity.umbc.edu/person/foaf/Tim/Finin/foaf.rdf`

**Query Text**  
A text area contains the following SPARQL query:

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

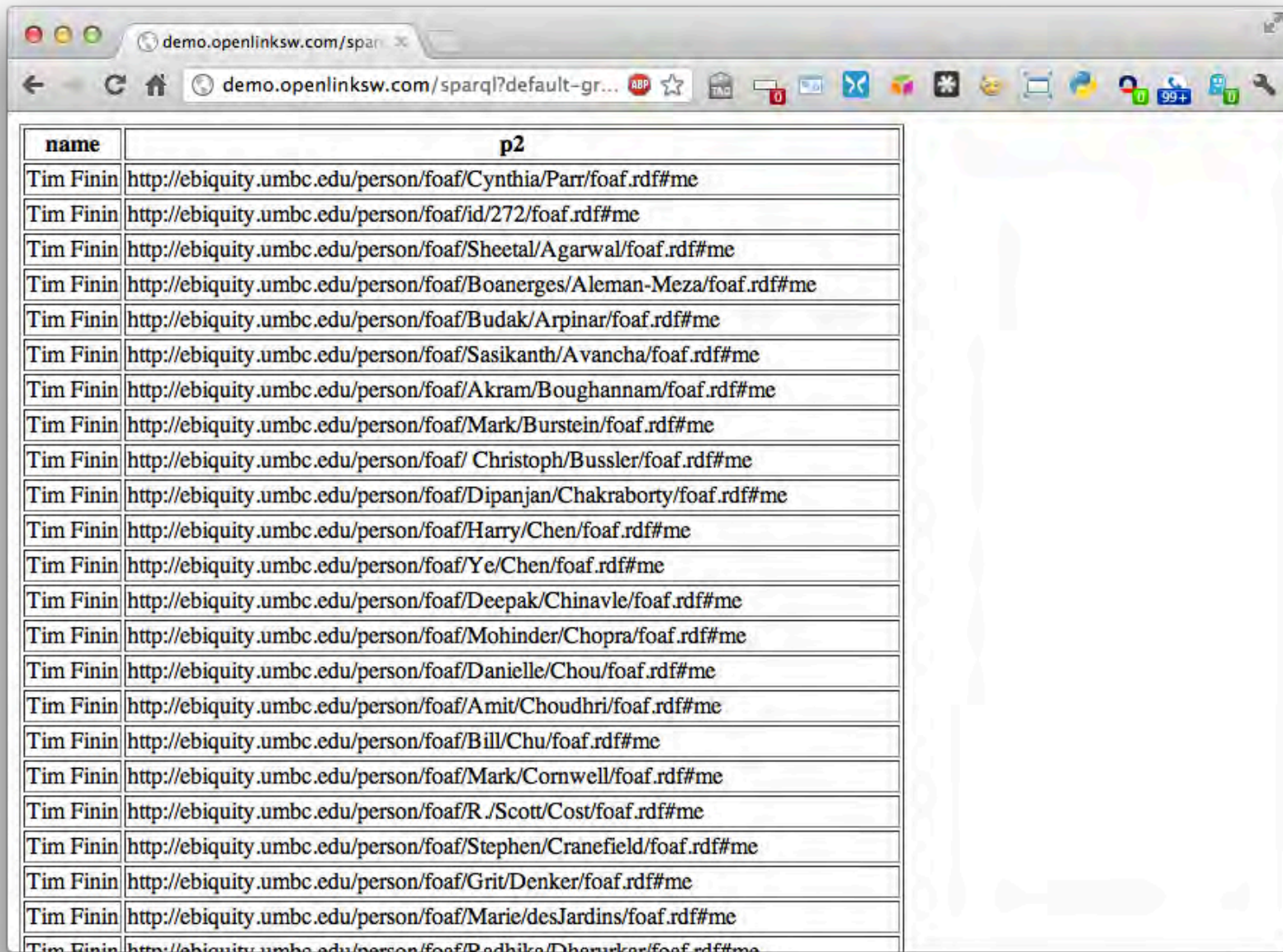
**Execution Options:**

- 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](#))*

Buttons: Run Query, Reset

# Query results as HTML



name	p2
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<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>
Tim Finin	<a href="http://ebiquity.umbc.edu/person/foaf/Redhika/Dhanuvar/foaf.rdf#me">http://ebiquity.umbc.edu/person/foaf/Redhika/Dhanuvar/foaf.rdf#me</a>

# Other result format options

The image shows a user interface for a query tool. On the left, there are several settings: 'Sponging:', 'Results Format:', 'Execution timeout:', and 'Options:'. Below 'Options:' is the text '(The result can only be sent back)'. At the bottom left are two buttons: 'Run Query' and 'Reset'. A dropdown menu is open over the 'Results Format:' label, listing the following options: 'Auto', 'HTML', 'Spreadsheet', 'XML', '✓ JSON', 'Javascript', 'NTriples', 'RDF/XML', 'CSV', 'CXML (Pivot Collection)', and 'CXML (Pivot Collection with QRcode)'. The 'JSON' option is selected, indicated by a checkmark. To the right of the interface, there is a table with a header 'source graphs' and a row containing the text '(less than 1000 are ignored)'.

Sponging:

Results Format:

Execution timeout:

Options:

(The result can only be sent back

Run Query Reset

source graphs

(less than 1000 are ignored)

# 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 .}
```

http://dbpedia.org/sparql

dbpedia lookup

```

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
<a href="http://dbpedia.org/property/birthPlace">http://dbpedia.org/property/birthPlace</a>	<a href="http://dbpedia.org/resource/Hawaii">http://dbpedia.org/resource/Hawaii</a>
<a href="http://dbpedia.org/property/birthPlace">http://dbpedia.org/property/birthPlace</a>	<a href="http://dbpedia.org/resource/Honolulu%2C_Hawaii">http://dbpedia.org/resource/Honolulu%2C_Hawaii</a>
<a href="http://dbpedia.org/property/birthPlace">http://dbpedia.org/property/birthPlace</a>	<a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a>
<a href="http://dbpedia.org/property/state">http://dbpedia.org/property/state</a>	<a href="http://dbpedia.org/resource/Illinois">http://dbpedia.org/resource/Illinois</a>
<a href="http://dbpedia.org/property/nationality">http://dbpedia.org/property/nationality</a>	<a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a>
<a href="http://dbpedia.org/ontology/nationality">http://dbpedia.org/ontology/nationality</a>	<a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a>
<a href="http://dbpedia.org/ontology/birthplace">http://dbpedia.org/ontology/birthplace</a>	<a href="http://dbpedia.org/resource/Hawaii">http://dbpedia.org/resource/Hawaii</a>
<a href="http://dbpedia.org/ontology/birthplace">http://dbpedia.org/ontology/birthplace</a>	<a href="http://dbpedia.org/resource/Honolulu%2C_Hawaii">http://dbpedia.org/resource/Honolulu%2C_Hawaii</a>
<a href="http://dbpedia.org/ontology/birthplace">http://dbpedia.org/ontology/birthplace</a>	<a href="http://dbpedia.org/resource/United_States">http://dbpedia.org/resource/United_States</a>

# 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



**About: [Barack Obama](#)**

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

Property	Value
<a href="#">dbpedia-owl:abstract</a>	<ul style="list-style-type: none"><li>Barack Hussein Obama II [bəˈɹɑːk hʊˈseɪn oʊˈbɑːmə] ist ein US-ameri Präsident der Vereinigten Staaten. Er wurde bei der Präsidentschafts 2012 für eine zweite Amtsperiode als US-Präsident bestätigt. Obama Kenianers, ist der erste Afroamerikaner in diesem Amt. Obama ist au 1992 Politiker der Demokratischen Partei. Von 2005 bis 2008 gehört Senat der Vereinigten Staaten an. Am 10. Dezember 2009 wurde ihr sich als erster US-Präsident im Amt öffentlich für die Legalisierung v</li><li>Barack Hussein Obama II is the 44th and current President of the Ur American to hold the office. Born in Honolulu, Hawaii, Obama is a gra where he was president of the Harvard Law Review. He was a comm He worked as a civil rights attorney in Chicago and taught constitutio to 2004. He served three terms representing the 13th District in the Il the United States House of Representatives in 2000. In 2004, Obam represent Illinois in the United States Senate with his victory in the M Democratic National Convention in July, and his election to the Sena 2007, and in 2008, after a close primary campaign against Hillary Ro</li></ul>

# 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.
```

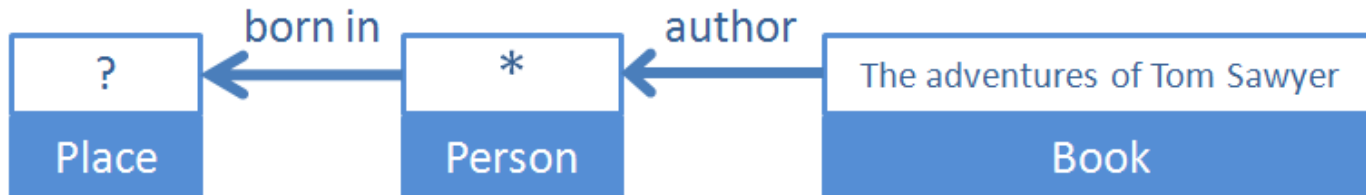
```
FILTER regex(?Name, 'obama', 'i')
```

```
?P ?Property ?Place .
```

```
?Place rdf:type dbpo:Place .
```

```
}
```

# Structured Keyword Queries

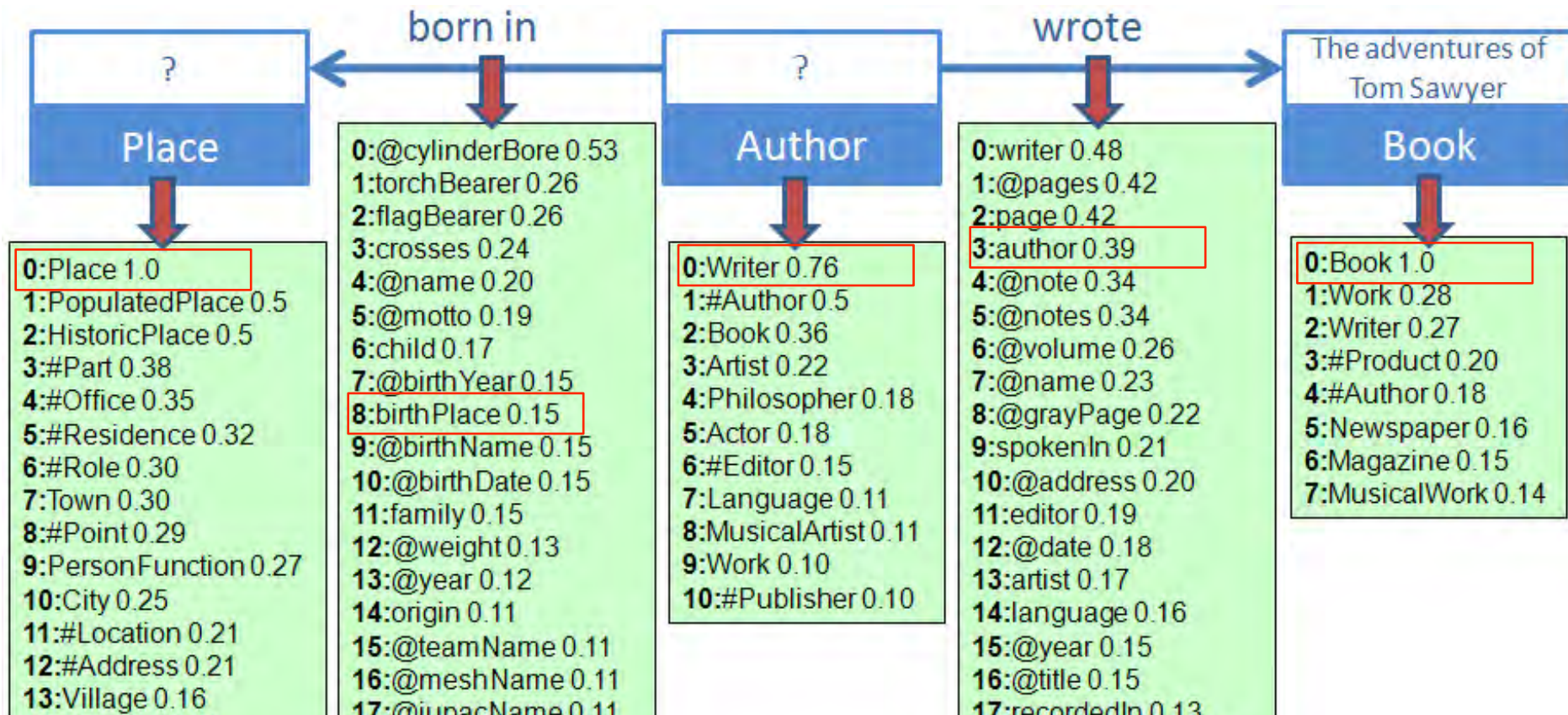


- Nodes are entities and links binary relations
- Entities described by two unrestricted terms: *name* or value and *type* or concept
- Outputs marked with ?
- Compromise between a natural language Q&A system and formal query
  - Users provide compositional structure of the question
  - Free to use their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

# Translation result

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

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



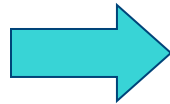
# SPARQL Generation



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