SPARQL - Query Language for RDF

Fulvio Corno, Laura Farinetti
Politecnico di Torino
Dipartimento di Automatica e Informatica
e-Lite Research Group – http://elite.polito.it
The “new” Semantic Web vision

- To make data machine processable, we need:
  - Unambiguous names for resources (that may also bind data to real world objects): URIs
  - A common data model to access, connect, describe the resources: RDF
  - Access to that data: SPARQL
  - Define common vocabularies: RDFS, OWL, SKOS
  - Reasoning logics: OWL, Rules
- “SPARQL will make a huge difference” (Tim Berners-Lee, May 2006)
SPARQL basics
SPARQL

- **Queries** are very important for distributed RDF data
  - Complex queries into the RDF data are often necessary
  - E.g.: “give me the (a,b) pair of resources, for which there is an x such that (x parent a) and (b brother x) holds” (i.e., return the uncles)

- This is the **goal** of SPARQL (Query Language for RDF)
SPARQL

- **W3C Recommendation**: January 15th, 2008
- **SPARQL queries RDF graphs**
  - An RDF graph is a set of triples
- **SPARQL can be used to express queries across diverse data sources**, whether the data is stored natively as RDF or viewed as RDF via middleware
SPARQL and RDF

- It is the triples that matter, not the serialization
  - RDF/XML is the W3C recommendation but it is not a good choice because it allows multiple ways to encode the same graph
- SPARQL uses the Turtle syntax, an N-Triples extension
Turtle - Terse RDF Triple Language

- A serialization format for RDF
- A subset of Tim Berners-Lee and Dan Connolly’s Notation 3 (N3) language
  - Unlike full N3, doesn’t go beyond RDF’s graph model
- A superset of the minimal N-Triples format
- Turtle has no official status with any standards organization, but has become popular amongst Semantic Web developers as a human-friendly alternative to RDF/XML
“Triple” or “Turtle” notation

http://www.w3.org/2000/10/swap/pim/contact#Person

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://www.w3.org/2000/10/swap/pim/contact#me

http://www.w3.org/2000/10/swap/pim/contact#fullName

http://www.w3.org/2000/10/swap/pim/contact#mailbox

mailto:em@w3.org

http://www.w3.org/2000/10/swap/pim/contact#personalTitle

Dr.

Eric Miller
“Triple” or “Turtle” notation

```
<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/2000/10/swap/pim/contact#fullName>
"Eric Miller" .

<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/2000/10/swap/pim/contact#mailbox>
<mailto:em@w3.org> .

<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/2000/10/swap/pim/contact#personalTitle>
"Dr." .

<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2000/10/swap/pim/contact#Person> .
```
Triple” or “Turtle” notation (abbreviated)

```turtle
w3people:EM#me contact:fullName "Eric Miller" .
w3people:EM#me contact:mailbox <mailto:em@w3.org> .
w3people:EM#me contact:personalTitle "Dr." .
w3people:EM#me rdf:type contact:Person .
```
Turtle - Terse RDF Triple Language

- Plain text syntax for RDF
  - Based on Unicode
- Mechanisms for namespace abbreviation
- Allows grouping of triples according to subject
- Shortcuts for collections
Turtle - Terse RDF Triple Language

- Simple triple:
  subject predicate object .

- Grouping triples:
  subject predicate object ; predicate object ...

```turtle
:john rdf:label "John" .

:john
  rdf:label "John" ;
  rdf:type ex:Person ;
  ex:homePage http://example.org/johnspage/ .
```
Prefixes

- Mechanism for namespace abbreviation
  
  ```
  @prefix abbr: <URI>
  ```

- Example:
  
  ```
  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  ```

- Default:
  
  ```
  @prefix : <URI>
  ```

- Example:
  
  ```
  @prefix : <http://example.org/myOntology#>
  ```
Identifiers

- URIs: <URI>
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#

- Qnames (Qualified names)
  - namespace-abbr?:localname

- Literals
  - "string"(@lang)?(^type)?
    - "John"
    - "Hello"@en-GB
    - "1.4"^^xsd:decimal
Blank nodes

- Simple blank node:
  
  \[
  [] \text{ or } _:x
  \]

  \[
  :john \text{ ex:hasFather } [] .
  
  :john \text{ ex:hasFather } _:x .
  \]

- Blank node as subject:
  
  \[
  [ \text{ predicate object ; predicate object } \ldots ] .
  \]

  \[
  [ \text{ ex:hasName "John"} ] .
  
  [ \text{ ex:authorOf :lotr ;}
    
  \text{ ex:hasName "Tolkien"} ] .
  \]
Blank nodes
Collections

- ( object1 ... objectn )

\[ \text{:doc1 ex:hasAuthor (:john :mary)} \].

- Short for

\[ \text{:doc1 ex:hasAuthor [ rdf:first :john;}
\text{ rdf:rest [ rdf:first :mary;}
\text{ rdf:rest rdf:nil ]}
\text{]} . \]
Example

@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntaxns# .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/#> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  :editor [
    :fullName "Dave Beckett";
    :homePage <http://purl.org/net/dajobe/>
  ] .
SPARQL

- Uses SQL-like syntax

```sparql
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
```

Prefix mechanism to abbreviate URIs

Variables to be returned

Query pattern (list of triple patterns)

FROM Name of the graph
SELECT

- Variables selection
- Variables: ?string

- Syntax: `SELECT var_1,...,var_n`

(SELECT ?name
SELECT ?x,?title)
WHERE

- **Graph patterns** to match
- **Set of triples**
  
  \[
  \{ (\text{subject} \ \text{predicate} \ \text{object} .) \}^* \}
  
- **Subject**: URI, QName, Blank node, Literal, Variable
- **Predicate**: URI, QName, Blank node, Variable
- **Object**: URI, QName, Blank node, Literal, Variable
Graph patterns

- The pattern contains unbound symbols
- By binding the symbols (if possible), subgraphs of the RDF graph are selected
- If there is such a selection, the query returns the bound resources
Graph patterns

- E.g.: (subject, ?p, ?o)
  - ?p and ?o are “unknowns”
Graph patterns

- The triplets in WHERE define the graph pattern, with ?p and ?o “unbound” symbols
- The query returns a list of matching p,o pairs

SELECT ?p ?o
WHERE {subject ?p ?o}
Example 1

```
SELECT ?cat ?val
  ?x category ?cat }
```

- Returns:

```
[['"Total Members",100],["Total Members",200],...,
  ["Full Members",10],...]
```
Example 2

```
SELECT ?cat ?val
WHERE {
  ?x category ?cat.
  FILTER(?val>=200).
}
```

- Returns:

```
[["Total Members",200],...]
```
Example 3

```
SELECT ?cat ?val ?uri
  ?x category ?cat.
  ?al contains ?x.
  ?al linkTo ?uri }
```

**Returns:**

```
[["Total Members",100,http://...],...]
```
Example 4

```
SELECT ?cat ?val ?uri
         ?x category ?cat.
OPTIONAL ?al contains ?x.
         ?al linkTo ?uri }
```

- **Returns:**
  
  ```
  [["Total Members",100,http://...], ..., 
   ["Full Members",20, ],...]
  ```
Other SPARQL Features

- Limit the number of returned results
- Remove duplicates, sort them,…
- Specify several data sources (via URI-s) within the query (essentially, a merge)
- Construct a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern
SPARQL use in practice

- **Locally**, i.e., bound to a programming environments like Jena
  
  - Jena is a Java framework for building Semantic Web applications; provides an environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine

- **Remotely**, e.g., over the network or into a database
Application

SPARQL Query \[\rightarrow\]

SPARQL “Engine”

\[\uparrow\text{Return in XML, JSON, …}\]

GRDDL, (e.g., microformats)

RDFa

SQL-SPARQL “Bridge”

RDF Data

Documents (XHTML, XML, …)

(Relational) Database
Tools for conversion

- **GRDDL** (Gleaning Resource Descriptions from Dialects of Languages)
  - W3C Recommendation (Sep 11th, 2007)
  - Enables users to obtain RDF triples out of XML documents

- **RDFa** (Resource Description Framework-in-attributes)
  - W3C Recommendation (October, 2008)
  - Set of extensions to XHTML that allows to annotate XHTML markup with semantics
  - Uses attributes from XHTML's meta and link elements, and generalizes them so that they are usable on all elements
  - A simple mapping is defined so that RDF triples may be extracted
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:foaf="http://xmlns.com/foaf/0.1/
     xmlns:dc="http://purl.org/dc/elements/1.1/
     version="XHTML+RDFa 1.0" xml:lang="en">
<head>
  <title>John's Home Page</title>
  <base href="http://example.org/john-d/" />
  <meta property="dc:creator" content="Jonathan Doe" />
</head>
<body>
<h1>John's Home Page</h1>
<p>My name is <span property="foaf:nick">John D</span> and I like <a href="http://www.neubauten.org/" rel="foaf:interest" xml:lang="de">Einstürzende Neubauten</a>. </p>
<p>My favorite book is the inspiring <cite property="dc:title">Weaving the Web</cite> by <span property="dc:creator">Tim Berners-Lee</span></p>
</body>
</html>
Automatic conversion to RDF/XML

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description rdf:about="http://example.org/john-d/">
     <dc:creator xml:lang="en">Jonathan Doe</dc:creator>
     <foaf:nick xml:lang="en">John D</foaf:nick>
     <foaf:interest rdf:resource="http://www.neubauten.org/"/>
     <foaf:interest>
       <rdf:Description rdf:about="urn:ISBN:0752820907">
         <dc:creator xml:lang="en">Tim Berners-Lee</dc:creator>
         <dc:title xml:lang="en">Weaving the Web</dc:title>
       </rdf:Description>
     </foaf:interest>
   </rdf:Description>
</rdf:RDF>
SPARQL use in practice

- Where to find meaningful RDF data to search?

- The Linked Data Project
The Linked Data Project
The Linked Data Project

- A fundamental prerequisite of the Semantic Web is the existence of **large amounts of meaningfully interlinked RDF data** on the Web.
- Linked Data is about using the Web to **connect related data** that wasn’t previously linked, or using the Web to lower the barriers to linking data currently linked using other methods.
- It is a **recommended best practice** for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF.
The Linked Data Project

- **Community effort** to make various open datasets available on the Web as RDF and to set RDF links between data items from different datasets.

- The datasets are published according to the [Linked Data principles](http://linkeddata.org/home) and can therefore be crawled by Semantic Web search engines and navigated using Semantic Web browsers.

- Supported by W3C.


  - [http://linkeddata.org/home](http://linkeddata.org/home)
The Web of Documents

- Analogy: a global filesystem
- Designed for human consumption
- Primary objects: documents
- Links between documents (or sub-parts)
- Degree of structure in objects: fairly low
- Semantics of content and links: implicit
The Web of Linked Data

- Analogy: a global database
- Designed for machines first, humans later
- Primary objects: things (or descriptions of things)
- Links between things
- Degree of structure in (descriptions of things): high
- Semantics of content and links: explicit
Linked Data example
Linked Data example
Why publish Linked Data?

- Ease of discovery
- Ease of consumption
  - Standards-based data sharing
- Reduced redundancy
- Added value
  - Build ecosystems around your data/content
Linked Open Data cloud

As of May 2007

F. Corno, L. Farinetti - Politecnico di Torino 44
DBpedia

- DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web.
- DBpedia allows to ask sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data.
GeoNames

- GeoNames is a geographical database that contains over eight million geographical names
- Available for download free of charge under a creative commons attribution license
Main contributors

- **DBLP** Computer science bibliography
  - Richard Cyganiak, Chris Bizer (FU Berlin)
- **DBpedia** Structured information from Wikipedia
  - Universität Leipzig, FU Berlin, OpenLink
- **DBtune, Jamendo** Creative Commons music repositories
  - Yves Raimond (University of London)
- **Geonames** World-wide geographical database
  - Bernard Vatant (Mondeca), Marc Wick (Geonames)
- **Musicbrainz** Music and artist database
  - Frederick Giasson, Kingsley Idehen (Zitgist)
- **Project Gutenberg** Literary works in the public domain
  - Piet Hensel, Hans Butschalowsky (FU Berlin)
- **Revyu** Community reviews about anything
  - Tom Heath, Enrico Motta (Open University)
- **RDF Book Mashup** Books from the Amazon API
  - Tobias Gauß, Chris Bizer (FU Berlin)
- **US Census Data** Statistical information about the U.S.
  - Josh Tauberer (University of Pennsylvania), OpenLink
- **World Factbook** Country statistics, compiled by CIA
  - Piet Hensel, Hans Butschalowsky (FU Berlin)
Statistics on datasets

http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics

<table>
<thead>
<tr>
<th>Data set</th>
<th>Size of the data set (number of triples)</th>
<th>Wrapper?</th>
<th>endpoint?</th>
<th>RDF dump?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM (RKB)</td>
<td>12,644,052</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>AudioScrobbler</td>
<td>600,000,000</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>BBC John Peel</td>
<td>277,000</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>BBC Later + TOTP (link not responding - 2009-04-01)</td>
<td>10,000</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>BBC Music</td>
<td>&gt;10,000,000</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>BBC Playcount Data</td>
<td>10,000</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>BBC Programmes</td>
<td>10,000,000</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Budapest BME (RKB)</td>
<td>42,064</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:Affymetrix</td>
<td>45,560,115</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:BioCYC</td>
<td>18,699,622</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:EBI:ChEBI</td>
<td>7,376,253</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:DBpedia</td>
<td>190,780</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:GO</td>
<td>8,188,649</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:HGNC</td>
<td>1,208,802</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:KEGG:Compound</td>
<td>177,199</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bio2RDF:KEGG:Drug</td>
<td>116,822</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Statistics on links between datasets

- [http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/LinkStatistics](http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/LinkStatistics)

<table>
<thead>
<tr>
<th>Source data set</th>
<th>Type</th>
<th>Target data set</th>
<th>Link count (range)</th>
<th>Link count (actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>DBLP (RKB)</td>
<td>&gt; 100,000</td>
<td>834,923</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>LAAS CNRS (RKB)</td>
<td>&gt; 100</td>
<td>173</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>Newcastle (RKB)</td>
<td>&gt; 100</td>
<td>684</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>eprints (RKB)</td>
<td>&gt; 10,000</td>
<td>53,081</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>IRIT Toulouse (RKB)</td>
<td>&gt; 100</td>
<td>729</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>CiteSeer (RKB)</td>
<td>&gt; 1,000,000</td>
<td>1,760,94</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>Pisa (RKB)</td>
<td>&gt; 100</td>
<td></td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>Reseaux (RKB)</td>
<td>&gt; 100</td>
<td></td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>IBM (RKB)</td>
<td>&gt; 100</td>
<td>116</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>IEEE (RKB)</td>
<td>&gt; 1,000</td>
<td>2,267</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>RAE 2001 (RKB)</td>
<td>&gt; 1,000</td>
<td>4,534</td>
</tr>
<tr>
<td>ACM (RKB)</td>
<td>Data Set</td>
<td>ECS Southampton (RKB)</td>
<td>&gt; 1,000</td>
<td>1,358</td>
</tr>
<tr>
<td>AudioScrobbler</td>
<td>Wrapper</td>
<td>Musicbrainz</td>
<td>&gt; 100</td>
<td></td>
</tr>
<tr>
<td>AudioScrobbler</td>
<td>Wrapper</td>
<td>FOAF profiles</td>
<td>&gt; 100,000</td>
<td></td>
</tr>
<tr>
<td>BBC John Peel</td>
<td>Data Set</td>
<td>DBpedia</td>
<td>&gt; 1,000</td>
<td></td>
</tr>
<tr>
<td>BBC Later + TOTP</td>
<td>Data Set</td>
<td>DBpedia</td>
<td>&gt; 1,000</td>
<td></td>
</tr>
</tbody>
</table>
Linked Data shopping list

- List of sites/datasets that the “community” would like to see published as Linked Data
  - This list may form the basis for some campaign/action to encourage these data publishers to embrace Linked Data

The Linked Data principles ("expectations of behavior")

- The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With linked data, when you have some of it, you can find other, related, data

- It is the unexpected re-use of information which is the value added by the web

  (Tim Berners-Lee)
The Linked Data principles ("expectations of behavior")

- Use **URIs** as names for things
  - Anything, not just documents
  - You are not your homepage
  - Information resources and non-information resources

- Use **HTTP URIs**
  - Globally unique names, distributed ownership
  - Allows people to look up those names
The Linked Data principles (“expectations of behavior”)

- Provide **useful information** in RDF
  - When someone looks up a URI
- Include RDF links to other URIs
  - To enable discovery of related information
Link to other datasets

- Popular predicates for linking
  - owl:sameAs
  - foaf:homepage
  - foaf:topic
  - foaf:based_near
  - foaf:maker/foaf:made
  - foaf:page
  - foaf:primaryTopic
  - rdfs:seeAlso
Link to other Data Sets

DBpedia

Wikicomp\(\text{\textup{\textsc{\textit{anyone\ can\ edit}}}\text{\textsc{\textup{\textit{free\ content\ license\ worldwide}}}}\text{\textsc{\textup{business\ directory}}}}\text{\textsc{\textup{\textit{anyone\ can\ edit}}}})\text{\textsc{\textit{free\ content\ license\ worldwide}}\text{\textsc{\textit{business\ directory}}}}\text{\textsc{\textit{anyone\ can\ edit}}})

brands \leftrightarrow \text{distilleries}

regions

Homepages

Geonames

FlickrWrappr

(flickr wrappr extends DBpedia with RDF links to photos posted on flickr)
Linked Data tools

- Tools for Publishing Linked Data
  - **D2R Server**: a tool for publishing relational databases as Linked Data
  - **Talis Platform**: the Talis Platform provides Linked Data-compliant hosting for content and RDF data
  - **Pubby**: a Linked Data frontend for SPARQL endpoints

- Linked Data/RDF Editors and Validators
  - **Hyena**: RDF Editor
  - **Vapour**: Linked Data Validator

- Tools for consuming Linked Data
  - **Semantic Web Browsers and Client Libraries**
  - **Semantic Web Search Engines**

- Linked Data applications for end users
  - **Semantic Web Browsers and Client Libraries**
  - **Other Linked Data Applications**
Pubby

- Many triple stores and other SPARQL endpoints can be accessed only by SPARQL client applications that use the SPARQL protocol
  - It cannot be accessed by the growing variety of Linked Data clients
- Pubby is designed to provide a Linked Data interface to those RDF data sources
- http://www4.wiwiss.fu-berlin.de/pubby/
Pubby
The Tabulator Project

- Generic **data browser** and editor
  - Provides a way to browse RDF data on the web
- Open source under the W3C software license
  - [http://www.w3.org/2005/ajar/tab](http://www.w3.org/2005/ajar/tab)
The Tabulator Project
Marbles

- Linked Data browser
  - Colored dots are used to correlate the origin of displayed data with a list of data sources (hence the name)

- source code is available in the SourceForge project
SPARQL syntax
SPARQL query structure

- A SPARQL query includes, in order
  - Prefix declarations, for abbreviating URIs
  - A result clause, identifying what information to return from the query
  - The query pattern, specifying what to query for in the underlying dataset
  - Query modifiers: slicing, ordering, and otherwise rearranging query results
SPARQL query structure

- A SPARQL query includes, in order

```sparql
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# result clause
SELECT ...
# query pattern
WHERE {
   ...
}
# query modifiers
ORDER BY ...
```
Dataset: Friend of a Friend (FOAF)

- **FOAF** is a standard RDF vocabulary for describing people and relationships
- Tim Berners-Lee's FOAF information available at [http://www.w3.org/People/Berners-Lee/card](http://www.w3.org/People/Berners-Lee/card)

```html
@prefix card: <http://www.w3.org/People/Berners-Lee/card#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
card:i foaf:name "Timothy Berners-Lee" .
<http://bblfish.net/people/henry/card#me> foaf:name "Henry Story" .
<http://www.cambridgesemantics.com/people/about/lee> foaf:name "Lee Feigenbaum" .
card:amy foaf:name "Amy van der Hiel" .
...
Example 1 – simple triple pattern

- In the graph http://www.w3.org/People/Berners-Lee/card, find all subjects (?person) and objects (?name) linked with the foaf:name predicate.
- Then return all the values of ?name.
- In other words, find all names mentioned in Tim Berners-Lee’s FOAF file

```sql
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?person foaf:name ?name .
}
```
SPARQL endpoints

- **Accept queries** and returns results via HTTP
  - Generic endpoints queries any Web-accessible RDF data
  - Specific endpoints are hardwired to query against particular datasets
- The **results of SPARQL queries** can be returned in a variety of formats:
  - XML, JSON, RDF, HTML
  - JSON (JavaScript Object Notation): lightweight computer data interchange format; text-based, human-readable format for representing simple data structures and associative arrays
SPARQL endpoints

- This query is for an arbitrary bit of RDF data (Tim Berners-Lee's FOAF file)
- => generic endpoint to run it

Possible choices
- SPARQLer - General purpose processor - [sparql.org](http://sparql.org/sparql.html)
  - http://sparql.org/sparql.html
- OpenLink's Virtuoso (Make sure to choose "Retrieve remote RDF data for all missing source graphs")
- Redland’s Rasqal
  - http://librdf.org/rasqal/
SPARQLer

General SPARQL query: input query, set any options and press "Get Results"

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

Target graph URI (or use FROM in the query): http://www.w3.org/People/Ber
Output XML: with XSLT style sheet (leave blank for none): /xml to html.xsl
or JSON output:
or text output:
Force the accept header to text/plain regardless

Get Results

F. Corno, L. Farinetti - Politecnico di Torino
OpenLink’s Virtuoso

OpenLink Virtuoso SPARQL Query

This query page is designed to help you test OpenLink Virtuoso SPARQL protocol endpoint. Consult the Virtuoso Wiki page describing the service or the Online Virtuoso Documentation section RDF Database and SPARQL.

There is also a rich Web based user interface with sample queries.

Dataset

SPARQL query

F. Corno, L. Farinetti - Politecnico di Torino
Example 1 - simple triple pattern

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?person foaf:name ?name .
}
```
Example 2 – multiple triple pattern

- Find all people in Tim Berners-Lee’s FOAF file that have names and email addresses
- Return each person’s URI, name, and email address

- Multiple triple patterns retrieve multiple properties about a particular resource
- SELECT * selects all variables mentioned in the query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE {
  ?person foaf:name ?name .
}
```
Example 2 - multiple triple pattern

<table>
<thead>
<tr>
<th>person</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/People/Connolly/#me">http://www.w3.org/People/Connolly/#me</a></td>
<td>Dan Connolly</td>
<td><a href="mailto:connolly@w3.org">mailto:connolly@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#i">http://www.w3.org/People/Berners-Lee/card#i</a></td>
<td>Timothy Berners-Lee</td>
<td><a href="mailto:timbl@w3.org">mailto:timbl@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.aaronsw.com/about.xrdf#aaronsw">http://www.aaronsw.com/about.xrdf#aaronsw</a></td>
<td>Aaron Swartz</td>
<td><a href="mailto:me@aaronsw.com">mailto:me@aaronsw.com</a></td>
</tr>
<tr>
<td><a href="http://www.dajobe.org/foaf.rdf#i">http://www.dajobe.org/foaf.rdf#i</a></td>
<td>Dave Beckett</td>
<td><a href="mailto:dave@dajobe.org">mailto:dave@dajobe.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#amy">http://www.w3.org/People/Berners-Lee/card#amy</a></td>
<td>Amy van der Hiel</td>
<td><a href="mailto:amy@w3.org">mailto:amy@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/EM/contact#me">http://www.w3.org/People/EM/contact#me</a></td>
<td>Eric Miller</td>
<td><a href="mailto:em@w3.org">mailto:em@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/karl/karl-foaf.xrdf#me">http://www.w3.org/People/karl/karl-foaf.xrdf#me</a></td>
<td>Karl Dubost</td>
<td><a href="mailto:karl@w3.org">mailto:karl@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#cm">http://www.w3.org/People/Berners-Lee/card#cm</a></td>
<td>Coralie Mercier</td>
<td><a href="mailto:coralie@w3.org">mailto:coralie@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#dj">http://www.w3.org/People/Berners-Lee/card#dj</a></td>
<td>Dean Jackson</td>
<td><a href="mailto:dean@w3.org">mailto:dean@w3.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#dj">http://www.w3.org/People/Berners-Lee/card#dj</a></td>
<td>Dean Jackson</td>
<td><a href="mailto:dino@grcrg.org">mailto:dino@grcrg.org</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#edd">http://www.w3.org/People/Berners-Lee/card#edd</a></td>
<td>Edd Dumbill</td>
<td><a href="mailto:edd@usefulinc.com">mailto:edd@usefulinc.com</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#edd">http://www.w3.org/People/Berners-Lee/card#edd</a></td>
<td>Edd Dumbill</td>
<td><a href="mailto:edd@xml.com">mailto:edd@xml.com</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#edd">http://www.w3.org/People/Berners-Lee/card#edd</a></td>
<td>Edd Dumbill</td>
<td><a href="mailto:edd@xmlhack.com">mailto:edd@xmlhack.com</a></td>
</tr>
<tr>
<td><a href="http://swordfish.rdfweb.org/people/libby/rdfweb/webwho.xrdf#me">http://swordfish.rdfweb.org/people/libby/rdfweb/webwho.xrdf#me</a></td>
<td>Libby Miller</td>
<td><a href="mailto:libby.miller@bristol.ac.uk">mailto:libby.miller@bristol.ac.uk</a></td>
</tr>
<tr>
<td><a href="http://people.csail.mit.edu/kagal/foaf.xrdf#me">http://people.csail.mit.edu/kagal/foaf.xrdf#me</a></td>
<td>Lalana Kagal</td>
<td><a href="mailto:lalana@csail.mit.edu">mailto:lalana@csail.mit.edu</a></td>
</tr>
<tr>
<td>nodeID://1003907694</td>
<td>Susie Stephens</td>
<td><a href="mailto:susie.stephens@gmail.com">mailto:susie.stephens@gmail.com</a></td>
</tr>
</tbody>
</table>
Example 3 – traversing a graph

- Find the homepage of anyone known by Tim Berners-Lee

```
<table>
<thead>
<tr>
<th>homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://purl.org/net/eric/">http://purl.org/net/eric/</a></td>
</tr>
<tr>
<td><a href="http://www.johnseelybrown.com/">http://www.johnseelybrown.com/</a></td>
</tr>
<tr>
<td><a href="http://www.grorg.org/dean/">http://www.grorg.org/dean/</a></td>
</tr>
<tr>
<td><a href="http://heddley.com/edd/">http://heddley.com/edd/</a></td>
</tr>
<tr>
<td><a href="http://www.mellon.org/about_foundation/staff/program-area-staff/irafuchs">http://www.mellon.org/about_foundation/staff/program-area-staff/irafuchs</a></td>
</tr>
</tbody>
</table>
```
Example 3 – traversing a graph

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX card: <http://www.w3.org/People/Berners-Lee/card#>
SELECT ?homepage
FROM <http://www.w3.org/People/Berners-Lee/card>
WHERE {
  card:i foaf:knows ?known .
}
```

- The FROM keyword specifies the target graph in the query
- By using ?known as an object of one triple and the subject of another, it is possible to traverse multiple links in the graph
Dataset: DBPedia

- **DBPedia** is an RDF version of information from Wikipedia
- Contains data derived from Wikipedia’s infoboxes, category hierarchy, article abstracts, and various external links
- Contains over 100 million triples
- Dataset: [http://dbpedia.org/](http://dbpedia.org/)
Example 4 – exploring DBPedia

Find 15 example concepts in the DBPedia dataset

```
SELECT DISTINCT ?concept
WHERE {
    ?s a ?concept .
} LIMIT 15
```

<table>
<thead>
<tr>
<th>concept</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/2004/02/skos/core#Concept">http://www.w3.org/2004/02/skos/core#Concept</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/MusicalWork">http://dbpedia.org/ontology/MusicalWork</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/Resource">http://dbpedia.org/ontology/Resource</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/Work">http://dbpedia.org/ontology/Work</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/Album">http://dbpedia.org/ontology/Album</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/Musical">http://dbpedia.org/ontology/Musical</a></td>
</tr>
<tr>
<td><a href="http://umbel.org/umbel/sc/Product">http://umbel.org/umbel/sc/Product</a></td>
</tr>
<tr>
<td><a href="http://umbel.org/umbel/ac/Artifact">http://umbel.org/umbel/ac/Artifact</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/class/yago/1982Novels">http://dbpedia.org/class/yago/1982Novels</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/Book">http://dbpedia.org/ontology/Book</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/class/yago/EnglishAstronomers">http://dbpedia.org/class/yago/EnglishAstronomers</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/class/yago/EnglishPoliticians">http://dbpedia.org/class/yago/EnglishPoliticians</a></td>
</tr>
<tr>
<td><a href="http://umbel.org/umbel/sc/Astronomer">http://umbel.org/umbel/sc/Astronomer</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/class/yago/LivingPeople">http://dbpedia.org/class/yago/LivingPeople</a></td>
</tr>
<tr>
<td><a href="http://dbpedia.org/class/yago/AmericanCompetitiveEaters">http://dbpedia.org/class/yago/AmericanCompetitiveEaters</a></td>
</tr>
</tbody>
</table>
Example 4 – exploring DBPedia

- LIMIT is a solution modifier that limits the number of rows returned from a query.
- SPARQL has two other solution modifiers:
  - ORDER BY for sorting query solutions on the value of one or more variables.
  - OFFSET, used in conjunction with LIMIT and ORDER BY to take a slice of a sorted solution set (e.g. for paging).
- The SPARQL keyword a is a shortcut for the common predicate rdf:type (class of a resource).
- The DISTINCT modifier eliminates duplicate rows from the query results.
Example 5 – basic SPARQL filters

- Find all landlocked countries with a population greater than 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 constraints use boolean conditions to filter out unwanted query results
- A semicolon (;) can be used to separate two triple patterns that share the same subject
SPARQL filters

- Conditions on literal values
- Syntax

```
FILTER expression
```

- Examples

```
FILTER (?age > 30)
FILTER isIRI(?x)
FILTER !BOUND(?y)
```
SPARQL filters

- **BOUND(var)**
  - true if var is bound in query answer
  - false, otherwise
  - !BOUND(var) enables negation-as-failure

- **Testing types**
  - isIRI(A): A is an “Internationalized Resource Identifier”
  - isBLANK(A): A is a blank node
  - isLITERAL(A): A is a literal
SPARQL filters

- Comparison between RDF terms
  - \( A = B \)
  - \( A \neq B \)

- Comparison between Numeric and Date types
  - \( A \geq B \)
  - \( A < B \)
  - \( A > B \)
  - \( A \leq B \)

- Boolean AND/OR
  - \( A \land B \)
  - \( A \lor B \)

- Basic arithmetic
  - \( A + B \)
  - \( A - B \)
  - \( A \times B \)
  - \( A / B \)
Example 5 – basic SPARQL filters

- Note all the translated duplicates in the results
- How can we deal with that?
Example 6 – SPARQL filters

Find me all landlocked countries with a population greater than 15 million (revisited), with the highest population country first

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 &&
    langMatches(lang(?country_name), "EN")) .
} ORDER BY DESC(?population)
Example 6 – SPARQL filters

- `lang` extracts a literal’s language tag, if any
- `langMatches` matches a language tag against a language range
Dataset: Jamendo

- **Jamendo** is a community collection of music all freely licensed under **Creative Commons** licenses
  - [http://www.jamendo.com/it/](http://www.jamendo.com/it/)
- **DBTune.org** hosts a queryable RDF version of information about Jamendo's music collection
  - Data on thousands of artists, tens of thousands of albums, and nearly 100,000 tracks
  - [http://dbtune.org/](http://dbtune.org/)
Example 7 – the wrong way

- Find all Jamendo artists along with their image, home page, and the location they’re near

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?img ?hp ?loc
WHERE {
  ?a a mo:MusicArtist ;
  foaf:name ?name ;
  foaf:img ?img ;
  foaf:homepage ?hp ;
  foaf:based_near ?loc .
}
```
Example 7 – DBTune SPARQL endpoint

- Jamendo has information on about 3,500 artists
- Trying the query we only get 2,667 results. What's wrong?

[F. Corno, L. Farinetti - Politecnico di Torino]
Example 7 – the right way

- Not every artist has an image, homepage, or location!
- OPTIONAL tries to match a graph pattern, but doesn't fail the whole query if the optional match fails
- If an OPTIONAL pattern fails to match for a particular solution, any variables in that pattern remain unbound (no value) for that solution

```sparql
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?img ?hp ?loc
WHERE {
  ?a a mo:MusicArtist ;
  foaf:name ?name .
  OPTIONAL { ?a foaf:img ?img }
  OPTIONAL { ?a foaf:homepage ?hp }
  OPTIONAL { ?a foaf:based_near ?loc }
}
```
Dataset: GovTrack

- **GovTrack** provides **SPARQL access** to data on the U.S. Congress
- Contains over 13,000,000 triples about legislators, bills, and votes
- [http://www.govtrack.us/](http://www.govtrack.us/)
Example 8 – querying alternatives

- Find Senate bills that either John McCain or Barack Obama sponsored and the other cosponsored

```sql
PREFIX bill: <http://www.rdfabout.com/rdf/schema/usbill/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?title ?sponsor ?status
WHERE {
  { ?bill bill:sponsor ?mccain ; bill:cosponsor ?obama . }
  UNION
  { ?bill bill:sponsor ?obama ; bill:cosponsor ?mccain . }
  ?bill a bill:SenateBill ;
  bill:status ?status ;
  bill:sponsor ?sponsor ;
  ?obama foaf:name "Barack Obama" .
  ?mccain foaf:name "John McCain" .
}
```
Example 8 – GovTrack specific endpoint

```
PREFIX bill: <http://www.rdfabout.com/rdf/schema/usbill/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?title ?sponsor ?status
WHERE {
  { ?bill bill:sponsor ?mccain ; bill:cosponsor ?obama . } 
  UNION 
  { ?bill bill:sponsor ?obama ; bill:cosponsor ?mccain . }
}
```

- The UNION keyword forms a disjunction of two graph patterns: solutions to both sides of the UNION are included in the results

http://www.govtrack.us/developers/rdf.xpd

F. Corno, L. Farinetti - Politecnico di Torino
RDF datasets

- All queries so far have been against a single graph
- In SPARQL this is known as the default graph
- RDF datasets are composed of a single default graph and zero or more named graphs, identified by a URI
- Named graphs can be specified with one or more FROM NAMED clauses, or they can be hardwired into a particular SPARQL endpoint
- The SPARQL GRAPH keyword allows portions of a query to match against the named graphs in the RDF dataset
- Anything outside a GRAPH clause matches against the default graph
Dataset: semanticweb.org

- **data.semanticweb.org** hosts RDF data regarding workshops, schedules, and presenters for the International Semantic Web (ISWC) and European Semantic Web Conference (ESWC) series of events.
- Presents data via FOAF, SWRC, and iCal ontologies.
- The data for each individual ISWC or ESWC event is stored in its own named graph, i.e., there is one named graph per conference event contained in this dataset.
- [http://data.semanticweb.org/](http://data.semanticweb.org/)
Example 9 – querying named graphs

Find people who have been involved with at least three ISWC or ESWC conference events

```sql
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?person
WHERE {
    GRAPH ?g1 { ?person a foaf:Person }
    GRAPH ?g2 { ?person a foaf:Person }
    GRAPH ?g3 { ?person a foaf:Person }
    FILTER(?g1 != ?g2 && ?g1 != ?g3 && ?g2 != ?g3) .
}
```
Example 9 – querying named graphs

- The GRAPH ?g construct allows a pattern to match against one of the named graphs in the RDF dataset.
- The URI of the matching graph is bound to ?g (or whatever variable was actually used).
- The FILTER assures that we’re finding a person who occurs in three distinct graphs.
- The Web interface used for this SPARQL query defines the foaf: prefix, which is why it is omitted here.
Data.semanticweb.org specific SPARQL endpoint

http://data.semanticweb.org/snorql/

Snorql: Exploring http://data.semanticweb.org/sparql

GRAPH: Default graph. List named graphs

GRAPH: Named graph goes here. Switch back to default graph

SPARQL:

```
SELECT DISTINCT ?person
WHERE {
  GRAPH ?g1 { ?person a foaf:Person }
  GRAPH ?g2 { ?person a foaf:Person }
  GRAPH ?g3 { ?person a foaf:Person }
  FILTER(?g1 != ?g2 && ?g1 != ?g3 && ?g2 != ?g3).
}
```

Results: Browse XSLT stylesheet URL: snorql/xml-to-html.xsl Go! Reset

Browse:
- Classes
- Properties
- Named Graphs

Powered by Sesame
SPARQL exercise
Exercises - RDF

```reasoning
@prefix : <http://example.org/data#> .
@prefix ont: <http://example.org/myOntology#> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .

:john
  vcard:FN "John Smith" ;
  vcard:N [ vcard:Given "John" ; vcard:Family "Smith" ] ;
  ont:hasAge 32 ;
  ont:marriedTo :mary .

:mary
  vcard:FN "Mary Smith" ;
  vcard:N [ vcard:Given "Mary" ; vcard:Family "Smith" ] ;
  ont:hasAge 29 .
```
SPARQL query – exercise 1

- Return the full names of all people in the graph

```
PREFIX vCard: <http://www.w3.org/2001/vcardrdfschema/3.0#>
SELECT ?fullName
WHERE {?x vCard:FN ?fullName}
```

- Result

```
fullName
=============
“John Smith”
“Mary Smith”
```
SPARQL query – exercise 2

- Return the relation between John and Mary

```sparql
PREFIX : <http://example.org/data#>
SELECT ?p
WHERE { :john ?p :mary }
```

- Result

```
p
=============
<http://example.org/myOntology#marriedTo>
```
SPARQL query – exercise 3

- Return the spouse of a person whose name is John Smith

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX ont: <http://example.org/myOntology#>
SELECT ?y
WHERE {?x vCard:FN "John Smith".
  ?x ont:marriedTo ?y}
```

- Result

```
Y =============
<http://example.org/data#mary>
```
SPARQL query – exercise 4

- Return the name and the first name of all people in the knowledge base

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name, ?firstName
WHERE {?x vCard:N ?name .
    ?name vCard:Given ?firstName}
```

- Result

<table>
<thead>
<tr>
<th>name</th>
<th>firstName</th>
</tr>
</thead>
<tbody>
<tr>
<td>“John Smith”</td>
<td>&quot;John&quot;</td>
</tr>
<tr>
<td>“Mary Smith”</td>
<td>&quot;Mary&quot;</td>
</tr>
</tbody>
</table>
SPARQL query – exercise 5

- Return all people over 30 in the knowledge base

```sparql
PREFIX ont: <http://example.org/myOntology#>
SELECT ?x
WHERE {?x ont:hasAge ?age .
FILTER(?age > 30)}
```

- Result

```
x
=================<http://example.org/data#john>
```
FROM

- Select RDF graph (= dataset) to be queried
- In case of multiple FROM clauses, graphs are merged
- Example

```prefix foaf: <http://xmlns.com/foaf/0.1/>
select ?name
from <http://example.org/foaf/aliceFoaf>
where { ?x foaf:name ?name }
```
SPARQL query – exercise 6

- **Graph** http://example.org/bob
  
  ```sparql
  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
  _:a foaf:name "Bob" .
  _:a foaf:mbox <mailto:bob@oldcorp.example.org> .
  ```

- **Graph** http://example.org/alice
  
  ```sparql
  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
  _:a foaf:name "Alice" .
  _:a foaf:mbox <mailto:alice@work.example> .
  ```
SPARQL query – exercise 6

- Return the names of people in both graphs

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?src ?name
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE
{ GRAPH ?src { ?x foaf:name ?name } }
```

- Result

<table>
<thead>
<tr>
<th>src</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://example.org/bob">http://example.org/bob</a></td>
<td>&quot;Bob&quot;</td>
</tr>
<tr>
<td><a href="http://example.org/alice">http://example.org/alice</a></td>
<td>&quot;Alice&quot;</td>
</tr>
</tbody>
</table>
References

- W3C, “Introduction to the Semantic Web”
- Lee Feigenbaum, “SPARQL By Example”
- Valentina Tamma, “Chapter 4: SPARQL”
  - http://www.csc.liv.ac.uk/~valli/Comp318/PDF/SPARQL.pdf
- Tom Heath, “An Introduction to Linked Data”
License

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License.

To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.