SPARQL By Example: The Cheat Sheet

Accompanies slides at:
http://www.cambridgesemantics.com/2008/09/sparql-by-example/

Comments & questions to:
Lee Feigenbaum <lee@cambridgesemantics.com>
VP Technology & Standards, Cambridge Semantics
Co-chair, W3C SPARQL Working Group
Conventions

Red text means:

“This is a core part of the SPARQL syntax or language.”

Blue text means:

“This is an example of query-specific text or values that might go into a SPARQL query.”
Nuts & Bolts

**URIs**

- **Write full URIs:**
  
  `<http://this.is.a/full/URI/written#out>`

- **Abbreviate URIs with prefixes:**
  
  ```
  PREFIX foo: <http://this.is.a/URI/prefix#>
  ...
  foo:bar ...
  ⇒ http://this.is.a/URI/prefix#bar
  ```

  **Shortcuts:**
  
  ```
  a ⇒ rdf:type
  ```

**Variables**

- **Variables:**
  
  ```
  ?var1, ?anotherVar, ?and_one_more
  ```

**Comments**

- **Comments:**
  
  ```
  # Comments start with a ‘#’
  # continue to the end of the line
  ```

** literals**

- **Plain literals:**
  
  ```
  “a plain literal”
  ```

- **Plain literal with language tag:**
  
  ```
  “bonjour”@fr
  ```

- **Typed literal:**
  
  ```
  “13”^^xsd:integer
  ```

- **Shortcuts:**
  
  ```
  true ⇒ “true”^^xsd:boolean
  3 ⇒ “3”^^xsd:integer
  4.2 ⇒ “4.2”^^xsd:decimal
  ```

**Triple Patterns**

- **Match an exact RDF triple:**
  
  ```
  ex:myWidget ex:partNumber “XY24Z1” .
  ```

- **Match one variable:**
  
  ```
  ?person foaf:name “Lee Feigenbaum” .
  ```

- **Match multiple variables:**
  
  ```
  ```
# Common Prefixes

<table>
<thead>
<tr>
<th>Prefix</th>
<th>...stands for</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:</td>
<td><a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a></td>
</tr>
<tr>
<td>rdfs:</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>owl:</td>
<td><a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a></td>
</tr>
<tr>
<td>xsd:</td>
<td><a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a></td>
</tr>
<tr>
<td>dc:</td>
<td><a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a></td>
</tr>
<tr>
<td>foaf:</td>
<td><a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a></td>
</tr>
</tbody>
</table>

More common prefixes at [http://prefix.cc](http://prefix.cc)
Anatomy of a Query

Declare prefix shortcuts (optional)

Define the dataset (optional)

Query modifiers (optional)

PREFIX foo: <...>
PREFIX bar: <...>

... SELECT ...
FROM <...>
FROM NAMED <...>
WHERE {
    ...
}
GROUP BY ...
HAVING ...
ORDER BY ...
LIMIT ...
OFFSET ...
BINDINGS ...

Query result clause

Query pattern
4 Types of SPARQL Queries

**SELECT queries**

*Project out specific variables and expressions:*

```
SELECT ?c ?cap (1000 * ?people AS ?pop)
```

*Project out all variables:*

```
SELECT *
```

*Project out distinct combinations only:*

```
SELECT DISTINCT ?country
```

Results in a table of values (in XML or JSON):

<table>
<thead>
<tr>
<th>?c</th>
<th>?cap</th>
<th>?pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex:France</td>
<td>ex:Paris</td>
<td>63,500,000</td>
</tr>
<tr>
<td>ex:Canada</td>
<td>ex:Ottawa</td>
<td>32,900,000</td>
</tr>
<tr>
<td>ex:Italy</td>
<td>ex:Rome</td>
<td>58,900,000</td>
</tr>
</tbody>
</table>

**CONSTRUCT queries**

*Construct RDF triples/graphs:*

```
CONSTRUCT {
  ?country a ex:HolidayDestination ;
  ex:arrive_at ?capital ;
  ex:population ?population .
}
```

Results in RDF triples (in any RDF serialization):

```
ex:France a ex:HolidayDestination ;
ex:arrive_at ex:Paris ;
ex:population 635000000 .
ex:Canada a ex:HolidayDestination ;
ex:arrive_at ex:Ottawa ;
ex:population 329000000 .
```

**ASK queries**

*Ask whether or not there are any matches:*

```
ASK
```

Result is either “true” or “false” (in XML or JSON):

true, false

**DESCRIBE queries**

*Describe the resources matched by the given variables:*

```
DESCRIBE ?country
```

Result is RDF triples (in any RDF serialization):

```
ex:France a geo:Country ;
ex:continent geo:Europe ;
ex:flag <http://.../flag-france.png> ;
...
Combining SPARQL Graph Patterns

Consider A and B as graph patterns.

A Basic Graph Pattern – one or more triple patterns

A . B

⇒ Conjunction. Join together the results of solving A and B by matching the values of any variables in common.

Optional Graph Patterns

A OPTIONAL { B }

⇒ Left join. Join together the results of solving A and B by matching the values of any variables in common, if possible. Keep all solutions from A whether or not there’s a matching solution in B
Combining SPARQL Graph Patterns

Consider A and B as graph patterns.

Either-or Graph Patterns

\{ A \} UNION \{ B \}

⇒ Disjunction. Include both the results of solving A and the results of solving B.

“Subtracted” Graph Patterns (SPARQL 1.1)

A MINUS \{ B \}

⇒ Negation. Solve A. Solve B. Include only those results from solving A that are not compatible with any of the results from B.
Consider A and B as graph patterns.

```
A .
{
    SELECT ...
    WHERE {
        B
    }
}

C .
```

⇒ Join the results of the subquery with the results of solving A and C.
SPARQL Filters

- **SPARQL FILTERs** eliminate solutions that do not cause an expression to evaluate to true.
- Place **FILTERs** in a query inline within a basic graph pattern

```
A . B . FILTER ( ...expr... )
```

<table>
<thead>
<tr>
<th>Category</th>
<th>Functions / Operators</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>!, &amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td>Math</td>
<td>+, −, *, /</td>
<td>?decimal * 10 &gt; ?minPercent</td>
</tr>
<tr>
<td>Existence (SPARQL 1.1)</td>
<td>EXISTS, NOT EXISTS</td>
<td>NOT EXISTS { ?p foaf:mbox ?email }</td>
</tr>
<tr>
<td>SPARQL tests</td>
<td>isURI, isBlank, isLiteral, bound</td>
<td>isURI(?person)</td>
</tr>
<tr>
<td>Accessors</td>
<td>str, lang, datatype</td>
<td>lang(?title) = “en”</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>sameTerm, langMatches, regex</td>
<td>regex(?ssn, &quot;\d{3}-\d{2}-\d{4}&quot;&quot;)</td>
</tr>
</tbody>
</table>
Aggregates (*SPARQL 1.1*)

1. Partition results into groups based on the expression(s) in the `GROUP BY` clause
2. Evaluate projections and aggregate functions in `SELECT` clause to get one result per group
3. Filter aggregated results via the `HAVING` clause

<table>
<thead>
<tr>
<th>?key</th>
<th>?val</th>
<th>?other1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>?key</th>
<th>?sum_of_val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>?key</th>
<th>?sum_of_val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

*SPARQL 1.1 includes: COUNT, SUM, AVG, MIN, MAX, SAMPLE, GROUP_CONCAT*
Property Paths (*SPARQL 1.1*)

- Property paths allow triple patterns to match arbitrary-length paths through a graph
- Predicates are combined with regular-expression-like operators:

<table>
<thead>
<tr>
<th>Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>path1/path2</td>
<td>Forwards path (path1 followed by path2)</td>
</tr>
<tr>
<td>^path1</td>
<td>Backwards path (object to subject)</td>
</tr>
<tr>
<td>path1</td>
<td>Either path1 or path2</td>
</tr>
<tr>
<td>path1*</td>
<td>path1, repeated zero or more times</td>
</tr>
<tr>
<td>path1+</td>
<td>path1, repeated one or more times</td>
</tr>
<tr>
<td>path1?</td>
<td>path1, optionally</td>
</tr>
<tr>
<td>path1{m,n}</td>
<td>At least m and no more than n occurrences of path1</td>
</tr>
<tr>
<td>path1{n}</td>
<td>Exactly n occurrences of path1</td>
</tr>
<tr>
<td>path1{m,}</td>
<td>At least m occurrences of path1</td>
</tr>
<tr>
<td>path1{,n}</td>
<td>At most n occurrences of path1</td>
</tr>
</tbody>
</table>
RDF Datasets

A SPARQL queries a *default graph* (normally) and zero or more *named graphs* (when inside a `GRAPH` clause).

**Default graph**

*the merge of zero or more graphs*

```
PREFIX ex: <...>
SELECT ... FROM ex:g1 FROM ex:g4 FROM NAMED ex:g1 FROM NAMED ex:g2 FROM NAMED ex:g3
WHERE {
  ... A ...
  GRAPH ex:g3 {
    ... B ...
  }
  OR
  GRAPH ?g {
    ... C ...
  }
}
```
SPARQL Over HTTP (the SPARQL Protocol)

http://host.domain.com/sparql/endpoint?\<parameters>\n
where \<parameters> can include:

query=\<encoded query string>  
   e.g.  SELECT+*%0DWHERE+{…

default-graph-uri=\<encoded graph URI>  
   e.g.  http%3A%2F%2Fexample.com%2Ffoo…
   n.b. zero of more occurrences of default-graph-uri

named-graph-uri=\<encoded graph URI>  
   e.g.  http%3A%2F%2Fexample.com%2Fbar…
   n.b. zero of more occurrences of named-graph-uri

HTTP GET or POST. Graphs given in the protocol override graphs given in the query.
Federated Query (SPARQL 1.1)

```
PREFIX ex: <...>
SELECT ...
FROM ex:g1
WHERE {
    ...
    A ...
    SERVICE ex:s1 {
        ...
        B ...
    }
    SERVICE ex:s2 {
        ...
        C ...
    }
}
```
### SPARQL Update Language Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INSERT DATA</strong> { triples }</td>
<td></td>
</tr>
<tr>
<td><strong>DELETE DATA</strong> {triples}</td>
<td></td>
</tr>
<tr>
<td>[ DELETE { template } ] [ INSERT { template } ] WHERE { pattern }</td>
<td></td>
</tr>
<tr>
<td>LOAD &lt;uri&gt; [ INTO GRAPH &lt;uri&gt; ]</td>
<td></td>
</tr>
<tr>
<td>CLEAR GRAPH &lt;uri&gt;</td>
<td></td>
</tr>
<tr>
<td>CREATE GRAPH &lt;uri&gt;</td>
<td></td>
</tr>
<tr>
<td>DROP GRAPH &lt;uri&gt;</td>
<td></td>
</tr>
</tbody>
</table>

[*denotes optional parts of SPARQL 1.1 Update syntax*]
## Some Public SPARQL Endpoints

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
<th>What’s there</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARQLer</td>
<td><a href="http://sparql.org/sparql.html">http://sparql.org/sparql.html</a></td>
<td>General-purpose query endpoint for Web-accessible data</td>
</tr>
<tr>
<td>DBPedia</td>
<td><a href="http://dbpedia.org/sparql">http://dbpedia.org/sparql</a></td>
<td>Extensive RDF data from Wikipedia</td>
</tr>
<tr>
<td>DBLP</td>
<td><a href="http://www4.wiwiss.fu-berlin.de/dblp/snorql/">http://www4.wiwiss.fu-berlin.de/dblp/snorql/</a></td>
<td>Bibliographic data from computer science journals and conferences</td>
</tr>
<tr>
<td>LinkedMDB</td>
<td><a href="http://data.linkedmdb.org/sparql">http://data.linkedmdb.org/sparql</a></td>
<td>Films, actors, directors, writers, producers, etc.</td>
</tr>
<tr>
<td>bio2rdf</td>
<td><a href="http://bio2rdf.org/sparql">http://bio2rdf.org/sparql</a></td>
<td>Bioinformatics data from around 40 public databases</td>
</tr>
</tbody>
</table>
SPARQL Resources

• The SPARQL Specification
  – http://www.w3.org/TR/rdf-sparql-query/

• SPARQL implementations
  – http://esw.w3.org/topic/SparqlImplementations

• SPARQL endpoints
  – http://esw.w3.org/topic/SparqlEndpoints

• SPARQL Frequently Asked Questions
  – http://www.thefigtrees.net/lee/sw/sparql-faq

• SPARQL Working Group
  – http://www.w3.org/2009/sparql/wiki/

• Common SPARQL extensions
  – http://esw.w3.org/topic/SPARQL/Extensions