SPARQL By Example: The Cheat Sheet

Accompanies slides at: http://www.cambridgesemantics.com/semantic-university/sparql-by-example

Comments & questions to:
Lee Feigenbaum <lee@cambridgesemantics.com>
VP Marketing & Technology, 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:

```html
<http://this.is.a/full/URI/written#out>
```

Abbreviate URIs with prefixes:

```html
PREFIX foo: <http://this.is.a/URI/prefix#>
... foo:bar ...
⇒ http://this.is.a/URI/prefix#bar
```

Shortcuts:

```html
a ⇒ rdf:type
```

Literals

Plain literals:

```
“a plain literal”
```

Plain literal with language tag:

```
“bonjour”@fr
```

Typed literal:

```
“13”^^xsd:integer
```

Shortcuts:

```
ture ⇒ “true”^^xsd:boolean
3 ⇒ “3”^^xsd:integer
4.2 ⇒ “4.2”^^xsd:decimal
```

Variables

```
?var1, ?anotherVar, ?and_one_more
```

Comments

```
# Comments start with a ‘#’ and # continue to the end of the line
```

Triple Patterns

```
ex:myWidget ex:partNumber “XY24Z1” .

?person foaf:name “Lee Feigenbaum” .

```
# 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)

Prefix foo:
Prefix bar:
...

Define the dataset (optional)

Select...
From <...>
From named <...>
Where { ...
... }
Group by ...
Having ...
Order by ...
Limit ...
Offset ...
Values ...

Query result clause

Query pattern

Query modifiers (optional)
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**

\[ A . B \]

\(\Rightarrow\) Conjunction. Join together the results of solving \(A\) and \(B\) by matching the values of any variables in common.

**Optional Graph Patterns**

\[ A \text{ OPTIONAL } \{ B \} \]

\(\Rightarrow\) 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

$$\{A\} \text{ UNION } \{B\}$$

$\Rightarrow$ Disjunction. Include both the results of solving $A$ and the results of solving $B$.

“Subtracted” Graph Patterns (SPARQL 1.1)

$A \text{ MINUS } \{B\}$

$\Rightarrow$ Negation. Solve $A$. Solve $B$. Include only those results from solving $A$ that are not compatible with any of the results from $B$. 
SPARQL Subqueries \textit{(SPARQL 1.1)}

Consider \textit{A} and \textit{B} as graph patterns.

\begin{verbatim}
A .
{
    SELECT ...
    WHERE {
        B
    }
}

C .
\end{verbatim}

$\Rightarrow$ Join the results of the subquery with the results of solving \textit{A} and \textit{C}.
SPARQL Filters

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

  ```sparql
  A . B . FILTER ( ...expr... )
  ```
<table>
<thead>
<tr>
<th>Category</th>
<th>Functions / Operators</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical &amp; Comparisons</td>
<td>!, &amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td>Conditionals (SPARQL 1.1)</td>
<td>EXISTS, NOT EXISTS, IF, COALESCE</td>
<td>NOT EXISTS { ?p foaf:mbox ?email }</td>
</tr>
<tr>
<td>Math</td>
<td>+, -, *, /, abs, round, ceil, floor, RAND</td>
<td>?decimal * 10 &gt; ?minPercent</td>
</tr>
<tr>
<td>Strings (SPARQL 1.1)</td>
<td>STRLEN, SUBSTR, UCASE, LCASE, STRSTARTS, CONCAT, STRENDS, CONTAINS, STRBEFORE, STRAFTER</td>
<td>STRLEN(?description) &lt; 255</td>
</tr>
<tr>
<td>Date/time (SPARQL 1.1)</td>
<td>now, year, month, day, hours, minutes, seconds, timezone, tz</td>
<td>month(now()) &lt; 4</td>
</tr>
<tr>
<td>SPARQL tests</td>
<td>isURI, isBlank, isLiteral, isNumeric, bound</td>
<td>isURI(?person)</td>
</tr>
<tr>
<td>Constructors (SPARQL 1.1)</td>
<td>URI, BNODE, STRDT, STRLANG, UUID, STRUUID</td>
<td>STRLANG(?text, &quot;en&quot;) = &quot;hello&quot;@en</td>
</tr>
<tr>
<td>Accessors</td>
<td>str, lang, datatype</td>
<td>lang(?title) = &quot;en&quot;</td>
</tr>
<tr>
<td>Hashing (1.1)</td>
<td>MD5, SHA1, SHA256, SHA512</td>
<td>BIND(SHA256(?email) AS ?hash)</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>sameTerm, langMatches, regex, REPLACE</td>
<td>regex(?ssn, &quot;\d{3}-\d{2}-\d{4}&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

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 (<em>path1</em> followed by <em>path2</em>)</td>
</tr>
<tr>
<td>^path1</td>
<td>Backwards path (object to subject)</td>
</tr>
<tr>
<td>path1</td>
<td>path2</td>
</tr>
<tr>
<td>path1*</td>
<td><em>path1</em>, repeated zero or more times</td>
</tr>
<tr>
<td>path1+</td>
<td><em>path1</em>, repeated one or more times</td>
</tr>
<tr>
<td>path1?</td>
<td><em>path1</em>, optionally</td>
</tr>
<tr>
<td>!uri</td>
<td>Any predicate except <em>uri</em></td>
</tr>
<tr>
<td>!^uri</td>
<td>Any backwards (object to subject) predicate except <em>uri</em></td>
</tr>
</tbody>
</table>
RDF Datasets

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

**Default graph**  
(the merge of zero or more graphs)

**Named 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 ?graph {
  ...
  C ...
}
```

```
OR
```
SPARQL Over HTTP (the SPARQL Protocol)

http://host.domain.com/sparql/endpoint?<parameters>

where <parameters> can include:

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

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

named-graph-uri=<encoded graph URI>
  e.g. http%3A%2F%2Fexmaple.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 ...
  }
}
```

Local Graph Store

Web

SPARQL Endpoint

ex:s1

SPARQL Endpoint

ex:s2
SPARQL Update

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

• SPARQL Specifications Overview
  – http://www.w3.org/TR/sparql11-overview/

• 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

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