SPARQL RDF Query Language Reference v1.3

Copyright © 2005 Dave Beckett, ILRT, University of Bristol.
Latest version: <http://www.ilrt.bris.ac.uk/people/cmdjb/2005/04-sparql/>
Comments to: dave.beckett@bristol.ac.uk

1. RDF Model and SPARQL RDF Terms Syntax

RDF Graph: A set of RDF Triples
RDF Triple: A triple (3-tuple) of:
   Subject: URI or Blank Node
   Predicate: URI
   Object: URI or Blank Node or Literal

URI: An absolute URI which may include a # fragment.
   <http://www.w3.org/>
   <http://example.org/#fragment> Relative URI resolved against base URI.
   <> Base URI, usually the query document URI.

RDF Literal: A Unicode string with an optional language tag.
   "hello" "bonjour"@fr

RDF Typed Literal: A Unicode string and datatype URI for encoding datatypes.
   "abc"^^<http://example.org/myDatatype>
   abbreviated with an XML QName style as:
   "10"^^xsd:integer
   Short forms for several common datatypes:
   10 "10"^^xsd:integer
   1.2e3 "1.2e3"^^xsd:double
   true "true"^^xsd:boolean

Blank Node: A node in a graph with a local name. The scope of the name is the RDF graph.
   _:node

2. Popular RDF Namespaces

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Common Prefix</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDF</td>
<td>rdf:</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a></td>
</tr>
<tr>
<td>Dublin Core</td>
<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>foaf:</td>
<td><a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a></td>
</tr>
<tr>
<td>XML Schema</td>
<td>xsd:</td>
<td><a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a></td>
</tr>
<tr>
<td>RDFS</td>
<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>owl:</td>
<td><a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a></td>
</tr>
</tbody>
</table>

3. SPARQL Language Reference


RDF Term: A part of an RDF Triple. A URI, Blank Node or a Literal.
   <uri> _:b1 "Literal"@en "abc123"^^my:datatype
Query Variable: Identifiers for binding to RDF Terms in matches.
   ?a / ?b or in lists: $name $title $place

Triple Pattern: An RDF Triple with Query Variables allowed in each term:
   <http://example.org/abc> ?x "Hello"
   ?subject ?predicate ?object

Graph Pattern: A set of Triple Patterns which binds RDF Terms in the graph to variables.
   Turtle abbreviations can be used for Triple Patterns, see Section 4.

Basic Graph Pattern: Written as a (..) block with ',', separating the triple patterns:
   { <http://example.org/abc> ?y "Hello" .
     ?subject $predicate "Literal" }

Group Graph Pattern: A graph pattern that contains multiple graph patterns which must all match
   to provide a result.
   { { ?person rdf:type foaf:Person }.
     { ?person foaf:name "Dave" . } }

Optional Graph Pattern: A graph pattern which may fail to match and provide bindings but not
   cause the entire query to fail. Written with the OPTIONAL keyword before a graph pattern.
   OPTIONAL { ?person foaf:nick ?nick }

Union Graph Pattern: A pair of graph patterns any of which may match and bind the same
   variables. Written with the UNION keyword between two graph patterns.
   { ?node ex:firstName ?name } UNION
   { ?node vcard:FN ?name }

Graph Graph Pattern: A keyword for specifying a graph name to use or to return a graph name
   as a binding. Written with the GRAPH keyword before a graph pattern.
   GRAPH <http://example.org/myfoaf> { ?person foaf:name ?name }

Value Constraints: A boolean expression in a graph pattern over query variables that
   constrains matched graph patterns.
   { ?item ex:size $size . FILTER $size < 10 }

4. SPARQL Query Structure

Prologue (optional)

BASE <uri>

PREFIX prefix: <uri> (repeatable)

Query Result forms (pick 1)

SELECT (DISTINCT) sequence of ?variable
SELECT (DISTINCT) *
DESCRIBE sequence of ?variable or <uri>
DESCRIBE *
CONSTRUCT (triple template)

ASK

Query Results Controls (optional)

LIMIT n, OFFSET m, SORT ...

Graph Pattern (optional)

{ ... }

Set the background graph:

FROM <uri>

Query Data Sources (optional)

Add a named graph (repeatable):

FROM NAMED <uri>
5. **SPARQL Query Result Forms**

**Variable Bindings:** A sequence of (set of variable bindings) for each query pattern match.  
SELECT *  
WHERE {?a rdf:type ?b}  
to ask for bindings for all variables mentioned in the query and  
SELECT ?a ?b  
WHERE {?a rdf:type ?b}  
to list them explicitly.

**RDF Graph:**  
**Describe Resources:** An RDF graph describing resources either given by URI  
DESCRIBE <http://example.org/thing>  
or by binding variables using the same syntax as SELECT.  
DESCRIBE ?person  
WHERE { ?person foaf:name "Dave" }  
**Build an RDF graph**  
CONSTRUCT { ?a foaf:knows ?b }  
WHERE { ?a ex:KnowsQuiteWell ?b }  
**Boolean:** True if the query pattern could be answered.  
ASK  
WHERE { ?a rdf:type foaf:Person }

6. **Query Results Controls and Sorting**

The optional controls on query results are optionally performed in the following order:  
1. DISTINCT to ensure solutions in the sequence are unique  
2. ORDER BY ordering solutions sequences by variable or function call:  
ORDER BY DESC(?date) ?title ASC(?familyName)  
3. LIMIT n to restrict the number of solutions to n  
4. OFFSET m to start the results in the solution from item m

7. **Values – datatypes, expressions and operators**

Supported datatypes: RDF Terms, xsd:string, xsd:double, xsd:float, xsd:decimal, xsd:integer and xsd:dateTime

**Logical operators:**  
A | B, A & B, !A  

**Arithmetic operators:**  
Unary: +A, -A  
Binary (A op B):  
=, !=, <, >, <=, >=

**RDF operators:**  
BOUND(A), ISURI(A), ISBLANK(A), ISLITERAL(A)  
STRING(A), LANG(A), DATATYPE(A)

**String Match operator:**  
REGEX(string, pattern [flags])

**Extension Functions and Explicit Type Casting:**  
Automatic Type:  
from xsd:decimal to xsd:float  
Promotion:  
from xsd:float to xsd:double

8. **Turtle RDF Syntax Reference**

Turtle (Turtle RDF Triple Language) describes triples in an RDF graph and allows abbreviations. Triple Patterns in SPARQL can use the same abbreviations.

This description is based on Turtle 2004-12-23 from <http://www.ilrt.bris.ac.uk/discovery/2004/04/turtle/>

**RDF Terms:**  
URI  
<URI>  
(=) is the base URI, often the document URI  
LITERAL: "string" or "string"@language or ^^<datatype URI>  
Blank Node: _:name or [] for an anonymous blank node

@prefix operator: URIs can be written as XML-style QNames by defining a prefix / URI binding:  
@prefix dc: <http://purl.org/dc/elements/1.1/> .

**Triples:** Written as 3 RDF terms with whitespace separating them as necessary, and ',' between triples:  
<dc:title "SPARQL Reference" .  

, operator: Triples with the same subject and predicate may be abbreviated with ';':  

; operator: Triples with the same subject may be abbreviated with ';':  
<http://work.example.org/> dc:title "My Workplace",   
dc:publisher "My Employer" .

[ ... ] operator: A sequence of (predicate object) pairs may be put inside [...] and a blank node subject will be assigned to them:  

[] operator: A blank node:  

a predicate: The often-used rdf:type QName may be abbreviated by the keyword a as a predicate:  
< a Foaf:Document .

Integers: Decimal integers 0 or larger can be written directly as literals (type xsd:integer)  
< ex:filesizeBytes 12345 .

( ... ) collections: RDF collections can be written inside (...) as a space-separated list of the contents:  
< ex:contents ( ex:apple ex:banana ex:pear ) .

9. **Example SPARQL Query**

```sparql
BASE <http://example.org/>  
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
PREFIX ex: <http://example.org/properties/1.0#>  
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
# This is a relative URI to BASE above  
FROM <http://example.org/people.rdf>  
SELECT DISTINCT Sperson ?name $age  
WHERE { ?person foaf:name ?name  
  foaf:age ?age.  
  OPTIONAL { ?person ex:age $age } .  
  FILTER ! REGEX(?name, "Bob") }  
LIMIT 3
```