**SPARQL RDF Query Language Reference v1.6**

Copyright © 2005 Dave Beckett, ILRT, University of Bristol.

**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>  
- <abc.rdf>  
- Relative URI resolved against base URI
- ex:name  
- URI shorthand using XML-style prefix ex and local name.

**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>  
- Declared with PREFIX (SPARQL) or @prefix (Turtle)

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

- _:node

**2. Common RDF Namespaces and Prefixes**

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Common Prefix</th>
<th>Namespace 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 Datatypes</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>  
- _:bl "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 block that matches part of the queried RDF graph. see Section 4.

**Basic Graph Pattern:** A set of Triple Patterns which binds RDF Terms in the graph to variables.

- 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:name ?name } UNION  
- { ?node vcard:FN ?name }

**Graph:** 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 (required, pick 1)**

- SELECT (DISTINCT) sequence of ?variable  
- SELECT (DISTINCT)*  
- DESCRIBE sequence of ?variable or <uri>  
- DESCRIBE *  
- CONSTRUCT ( graph pattern )  
- GRAPH ?graph { ?person foaf:name ?name }

**Query Data Sources (optional)**

- Set the background graph:
  
- FROM <uri>

**Graph Pattern (optional, required for ASK)**

- WHERE ( graph pattern )

**Query Results Controls (optional)**

- LIMIT m, OFFSET m, ORDER BY ...
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:  
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


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

Comparison (A op B):  =, !=, <, >, <=, >=

Arithmetic operators:  
Unary: +A, -A

Binary (A op B):  +, - , *, /, *, ^

RDF operators:  
BOUND(A), ISURI(A), ISBLANK(A), ISLITERAL(A)

String:  STR(A), LANG(A), DATATYPE(A)

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

Extension Functions and Explicit Type Casting:  
QName (expression, expression, ...)  
from xsd:decimal to xsd:float  
from xsd:float to xsd:double

8. Turtle RDF Syntax Reference

Turtle (Terse 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.irlt.bris.ac.uk/discovery/2004/01/turtle/>

RDF Terms:  
URI <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:  

, operator:  Triples with the same subject and predicate may be abbreviated with ‘,’:  
My Book , Mein Buch

; operator:  Triples with the same subject may be abbreviated with ‘;’:  
My Workplace

[ ... ] operator:  A sequence of (predicate object) pairs may be put inside [...] and a blank node subject will be assigned to them:  
dc:creator [ foaf:name “Dave” ; foaf:homePage <http://...> ]

[ ] operator:  A blank node:  
[ ] a Foaf:Document  

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):  
<http://work.example.org/> dc:title “My Workplace” ;

dc:publisher “My Employer”.

( ... ) 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

BASE <http://example.org/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
# This is a relative URI to BASE above  
PREFIX ex: <properties/1.0#>

SELECT DISTINCT $person ?name $age  
FROM <http://rdf.example.org/people.rdf>  
WHERE { $person a foaf:Person ;  
  foaf:name ?name .
  OPTIONAL { $person ex:age $age } .
  FILTER ! REGEX(?name, “Bob”) }  
LIMIT 10 OFFSET 20 ORDER BY ASC(?name)