# SPARQL RDF Query Language Reference v1.8

Copyright © 2005, 2006 Dave Beckett.
Latest version: <http://www.dajobe.org/2005/04-sparql/>
Comments to: dave@dajobe.org

## 1. RDF Model and SPARQL RDF Terms Syntax

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

**URI:** An absolute IRI which may include a # fragment.

```
<http://www.w3.org/>
<http://example.org/#fragment>
```

**RDF Literal:** A Unicode string with an optional language tag.

```
“hello”  “bonjour”#fr
```

**RDF Typed Literal:** A Unicode string and datatype IRI for encoding datatypes.

```
“abc”^^<http://example.org/myDatatype>
```

**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>RDF</th>
<th>Dublin Core</th>
<th>FOAF</th>
<th>XML Schema Datatypes</th>
<th>RDFS</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix</td>
<td>rdf</td>
<td>dc</td>
<td>foaf</td>
<td>xsd</td>
<td>rdfs</td>
<td>owl</td>
</tr>
</tbody>
</table>

## 3. SPARQL Query Language Reference

Based on SPARQL Query Language 23 November 2005


### RDF Term:
A part of an RDF Triple. An IRI, 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
```

### Anonymous Query Variable:
Blank Nodes in a graph pattern act as variables that cannot be SELECTed.
```
_:abc
```

### Triple Pattern:
An RDF Triple with Query Variables or blank nodes allowed in each term:
```
```

### Graph Pattern:
A set of Triple Patterns binding RDF Terms in the graph to variables.
```
PREFIX ex: <http://example.org/abc>

GRAPH <http://example.org/myfoaf>

GROUP

{ ?person rdf:type foaf:Person }
{ ?person foaf:name “Dave” }

OPTIONAL { ?person foaf:nick ?nick }

UNION

GROUP

{ ?node ex:name ?name } UNION
{ ?node vcard:FN ?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 Language Structure

### Prologue (optional)
```
BASE <iri>
```
```
PREFIX prefix: <iri>  (repeatable)
```

### Query Result forms (required, pick 1)
```
SELECT (DISTINCT) sequence of variable
```
```
SELECT (DISTINCT) *
```
```
DESCRIBE sequence of variable or <iri>
```
```
DESCRIBE *
```
```
CONSTRUCT { graph pattern }
```
```
ASK
```

### Query Dataset Sources (optional)
Add triples to the background graph (repeatable):
```
FROM <iri>
```
```
Add a named graph (repeatable):
```
FROM NAMED <iri>
```

### Graph Pattern (optional, required for ASK)
```
WHERE { graph pattern [ FILTER expression ] }
```

### Query Results Ordering (optional)
```
ORDER BY ... 
```

### Query Results Selection (optional)
```
LIMIT n, OFFSET m
```
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 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 made by substituting variables into a triple template.

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 Ordering and Modifying

The optional modifications on query results are performed in the following order:

1. DISTINCT to ensure solutions in the sequence are unique
2. ORDER BY ordering solutions sequences by variable, expression or extension function call:
   ORDER BY DESC(?name) ?title ASC(?familyName) my:fn(?a)
   in descending order by date, by ascending title order, by familyName ascending, by extension function
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, 1 A, (A)

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

Arithmetic operators:

Unary: +A, -A

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

RDF operators:

Boolean: BOUND(A), isIRI(A) / isURI(A), isBlank(A), isLiteral(A)

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

String Match operator:

REGEX(string expression, pattern expression [flags expression])

declares an operator: the pattern syntax is from XQuery 1.0 / XPath 2.0, XML Schema, similar to Perl. flags are s, m, i, x

Extension Functions and Explicit Type Casting:

Automatic Type from xsd:decimal to xsd:float

Promotion: from xsd:float to xsd:double


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

RDF Terms:

IRI < IRI > (<> is the base IRI, often the document IRI)

Literal: “string” or “string”@language or ~< datatype IRI >

Blank Node: _ : name [ ] for an anonymous blank node

@prefix operator: IRIs can be written as XML-style QNames by defining a prefix / IRI binding:

@prefix dc: <http://purl.org/dc/elements/1.1/> .

Triples: 3 RDF terms with whitespace separating them as necessary, and ’,’ between triples:

\< dc:title ”SPARQL Reference” \>

\< dc:date “2006-02-06” ”xsd:dateTime” \>

\< dc:creator \[
foaf:name “Dave”; foaf:homePage <http:...> \]

\< foaf:Document \>

Decimal integers: Positive or negative decimal integers can be written as (type xsd:integer)

\< xs:inBytes 12345 \>

Decimal numbers: Positive or negative decimal numbers can be written as (type xsd:decimal)

\< xs:inBytes 8.5 \>

( ... ) collections: RDF collections can be written inside ( ... ) as space-separated lists of 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 IRI to BASE above

PREFIX ex: <properties/1.0#>

SELECT DISTINCT ?person ?name ?age

WHERE { 

 PERSON A foaf:Person ; foaf:name ?name.

 OPTIONAL { 

 PERSON ex:age ?age .

 FILTER (!REGEX(?name, “Bob”))

 }

 ORDER BY ASC(?name) LIMIT 10 OFFSET 20