1. RDF Model and SPARQL RDF Terms Syntax

RDF Graph: An absolute URI which may include a # fragment.

Subject: URI or Blank Node

Predicate: URI

Object: URI or Blank Node or Literal

RDF Triple: A triple (3-tuple) of:

`<subject predicate object>`

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

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

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

namespace: `_: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 Query Language Reference


| RDF Term:                  | A part of an RDF Triple. A URI, Blank Node or a Literal. |
| Query Variable:           | Identifiers for binding to RDF Terms in matches. |
| Anonymous Blank Node:     | Blank Nodes in a graph pattern act as variables that cannot be SELECTed |
| Triple Pattern:           | An RDF Triple with Query Variables or blank nodes allowed in each term: |

Graph Pattern:

- **Basic**: A set of Triple Patterns binding RDF Terms in the graph to variables.
- **Group**: A graph pattern containing multiple graph patterns which must all match
- **Optional**: A graph pattern which may fail to match and provide bindings but not cause the entire query to fail. Written with OPTIONAL before a graph pattern.
- **Union**: A pair of graph patterns any of which may match and bind the same variables. Written with the UNION keyword between two graph patterns.
- **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.

Value Constraints:

- **ASK**: A boolean expression in a graph pattern over query variables that constrains matched graph patterns.
- **BASE**: BASE `<uri>`
- **PREFIX**: `prefix: <uri>` (repeatable)
- **SELECT**: `(DISTINCT) sequence of ?variable`
- **DESCRIBE**: `sequence of ?variable or <uri>`
- **CONSTRUCT**: `(graph pattern)`
- **FROM**: `uri`
- **WHERE**: `(FILTER expression)`
- **ORDER BY**: `...`
- **LIMIT**: `n, OFFSET m`
5. SPARQL Query Result Forms

<table>
<thead>
<tr>
<th>Variable Bindings:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A sequence of (set of variable bindings) for each query pattern match.</td>
</tr>
</tbody>
</table>

**SELECT**
WHERE { ?a rdf:type ?b }
to ask for bindings for all variables mentioned in the query and
SELECT * ?b
WHERE { ?a rdf:type ?b }
to list them explicitly.

| RDF Graph: |
| An RDF graph describing resources either given by URI |

**RDF graph**
Build an RDF graph made by substituting variables into a triple template.

**Describe**
DESCRIPT <http://example.org/thing>
or by binding variables using the same syntax as SELECT.
DESCRIPT ?person
WHERE { ?person foaf:name "Dave" }

**Resources:**
RDF Terms:
This description is based on Turtle 2005-07-01 from <http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/>

**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 2005-07-01 from <http://www.ilrt.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:

\[ dc:title "SPARQL Reference" .
\]
\[ date "2005-04-19"^^xsd:dateTime .
\]
\[ ask: "My Book", "Mein Buch" @de .
\]

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

\]

**& 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:

\[ [ a ex:creator foaf:name "Dave" ; foaf:homePage <http:...> ] .
\]

**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)
\[ <http://example.org/mybook> dc:title "My Book" , "Mein Buch" @de .
\]

**ex:sizeInBytes 12345 .
**

**collections:** RDF collections can be written as XML-style QNames by defining a prefix / URI binding:
\[ ex:sizeInBytes 12345 .
\]

**...**
... 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**

```
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/personB.rdf>  
WHERE { ?person a foaf:Person ;  
  foaf:name ?name ;  
  ex:sizeInBytes 12345 .  
  ?person ex:age $age } .

OPTIONAL { ?person ex:age $age } .
FILTER (!REGEX(?name, "Bob")) .
```

ORDER BY ASC(?name) LIMIT 10 OFFSET 20