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.

### RDF Literal:
An Unicode string with an optional language tag.

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

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

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. An IRI, Blank Node or a Literal.

### Query Variable:
Identifiers for binding to RDF Terms in matches.

### Anonymous
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:

```
<?xml-stylesheet href="http://www.w3.org/2001/XMLSchema#" type="text/xsl" ?>
```

### RDF Term Syntax

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

- **Subject:** IRI or Blank Node
- **Predicate:** IRI
- **Object:** IRI or Blank Node or Literal

#### RDF Graph
A set of RDF Triples

#### RDF Literal
An Unicode string with an optional language tag.

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

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

**Hort forms for several common datatypes:**

- 1.2345
- "true"
- "10"^^xsd:integer
- "-10"^^xsd:integer
- "true"^^xsd:boolean
- "1.2345"^^xsd:decimal

**URIs:**

- Relative IRI resolved against base IRI.
- Base IRI, usually the query document IRI
- Declared with PREFIX (SPARQL) or @prefix (Turtle)

**Turtle abbreviations can be used for Triple Patterns, see Section 4.**

4. SPARQL Query Language Structure

### Prologue (optional)
```
BASE <iri>
```

### Query Result forms (required, pick 1)
```
SELECT (DISTINCT) sequence of ?variable
SELECT (DISTINCT)*
DESCRIBE sequence of ?variable or <iri>
```
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 or by binding variables using the same syntax as SELECT.

```
DESCRIBE <http://example.org/thing>
```
or

```
DESCRIBE ?person
WHERE { ?person foaf:name "Dave" }
```

Build an RDF graph 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(?date) ?title ASC(?familyName) my:fn(?a)
```
in descending order by date, by ascending title order, by familyName ascending, by extension function

3. OFFSET n to restrict the number of solutions to n

4. LIMIT m to start the results in the solution from item m

7. Values – datatypes, expressions and operators


Logical operators: Logical: $A || B, A & B, 1, 0, (A), (A)

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

Arithmetic operators: Unary: +, -, *, /

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])
```

pattern syntax is from XQuery 1.0 / XPath 2.0, XML Schema and similar to Perl. Flags are s, m, i, x

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 2006-01-21 from <http://www.dajobe.org/2004/01/turtle/>

RDF Terms:

```
IRI: <IRI>
Literal: "string" or "string"@language or "^^<datatype IRI>
Blank Node: __:name or {} 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: Written as 3 RDF terms with whitespace separating them as necessary, and ',' between triples:

```
<dc:title "SPARQL Reference"
<dc:date "2006-02-06"
<xs:dateTime
```

, 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:creator "My Employer".

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

```
<dc:creator (maker: "Dave"; foaf:homePage <http://...>
```

[ ] operator: A blank node:

```
[ ] a ex:Book (dc:title "Thing"; dc:description "On the shelf"
```

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

```
<.dc:Document .
```

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

```
<ex:sizeInBytes 12345
```

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

```
<ex:shoeSize 8.5
```

( ... ) 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 IRI to BASE above
PREFIX ex: <properties/1.0#>

SELECT DISTINCT $person ?name $age
FROM <http://rdf.example.org/person.rdf>
FROM <http://rdf.example.org/personB.rdf>
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
```