The Semantic Web — RDF, RDF Schema, and OWL (Part 1)

Mitchell W. Smith
Array BioPharma, Inc.
msmith@arraybiopharma.com
Agenda

- RDF
- RDF/XML Syntax
- RDF Schema
- SPARQL
- Part Two:
  - OWL
  - Ontologies
  - Reasoners
  - RDF, RDF Schema, and OWL *versus* UML
  - Tools
RDF Introduction

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

Semantic Web Layering

From: Berners-Lee XML 2000
RDF

- RDF is an assertional language intended to be used to express propositions.
  - RDF triples are propositions – a labeled connection between two resources.
  - An RDF triple contains three components:
    - The subject, which is an RDF URI reference or a blank node.
    - The predicate, which is an RDF URI reference.
    - The object, which is an RDF URI reference, a literal or a blank node.
  - RDF is monotonic and is defined to have an open-world assumption.
RDF – Open-World Assumption

- An open-world assumption means that, with nothing asserted, every interpretation is true.
  - An RDF triple puts a constraint in the world – there are fewer ways the world could be.
  - Large numbers of assertions constrain the world even more as they eliminate other “possible” interpretations.
  - Making an assertion amounts to claiming that the world is an interpretation which assigns the value true to the assertion.
RDF – URI Reference

- RDF attaches no semantic meaning to a URI reference (URI with an optional fragment identifier at the end).
  - Makes no assumptions about the relationship between a URI reference and the document or resource that can be retrieved from the URI reference.
  - URI references are simple names.
    - Ignores any aspect of meaning encoded in a URI reference.
  - Assumes that there is global consistency – a URI reference means the same thing whenever it occurs.
    - RDF does not handle changes over time.
RDF – Graph

- An RDF graph is a set of RDF triples.
RDF – Literals

- Can be the object of a RDF triplet.
  - This restricts literals so that the subject and predicate can be uniquely identified with a URI Reference.
- Character string literals can have a language tag.
  - Syntax: “aaa”@en
- Literals can be typed.
  - Uses a subset of the XML Schema simple types.
  - Example: “1”^^xs:integer
  - However, any type can be used:
    - Example: “b,1”^^http://www.example.org/types#appDataType
RDF – Blank Nodes

- A RDF triplet can have a blank subject or object.
- Blank nodes have no name and have no global meaning (unlike URIs and literals).
- Do have a node identifier, but the identifier is surface syntax only.
- They indicate the existence of a thing, without specifying what the thing is.
- Can only be used as the subject or object in a RDF triplet.
RDF – Blank Nodes Example

```
exstaff:85740   exterms:address   _:johnaddress
_:johnaddress   exterms:street   "1501 Grant Avenue"
_:johnaddress   exterms:city     "Bedford"
_:johnaddress   exterms:state    "Massachusetts"
_:johnaddress   exterms:postalCode "01730"
```
RDF – Entailment

- Definition: To impose, involve, or imply as a necessary accompaniment or result <the project will entail considerable expense>.
- In RDF, this means that any interpretation that makes a set of RDF statements true also makes another set true as well.
- Allows for deriving implications.
RDF – Vocabulary

- rdf:XMLLiteral
- Reification:
  - rdf:Statement, rdf:subject, rdf:predicate, rdf:object
- Containers:
  - rdf:Bag, rdf:Alt, rdf:Seq, rdf:_1, rdf:_2, ...
- Collections:
  - rdf:List, rdf:Nil, rdf:first, rdf:rest
- rdf:value
- rdf:type
- rdf:Property
RDF – Vocabulary – rdf:XMLLiteral

- XML content is a literal value.
- rdf:XMLLiteral is just well-typed XML.
- URI:
  http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral
RDF – Reification

- Definition: To regard (something abstract) as a material or concrete thing.
- Example:
  - RDF: `<ex:foo> <ex:bar> <ex:blee>`
  - Reified RDF:
    - `_:xxx rdf:type      rdf:Statement`
    - `_:xxx rdf:subject   <ex:foo>`
    - `_:xxx rdf:predicate <ex:bar>`
    - `_:xxx rdf:object    <ex:blee>`
- In RDF, reification describes other RDF.
- A reified statement does not entail the RDF statement. It only says that the RDF statement exists, not that it is true.
RDF – Containers

- Containers have a type and membership.
  - rdf:Bag – unordered and allows duplicates
  - rdf:Seq – ordered
  - rdf:Alt – collection of alternatives
    
    ```
    _:xxx rdf:type rdf:Bag
    _:xxx rdf:_1 <ex:foo>
    _:xxx rdf:_2 <ex:bar>
    ```

- However, these are informal definitions. RDF does not enforce these entailments.
- Describes containers rather than constructing containers.
- Containers are “open”, that is new members can be added indefinitely.
RDF – Collections

- **Rdf:List** – a list structure in terms of head/tail links.
  
  `_:c1 rdf:first <ex:foo>`
  `_:c1 rdf:rest _:c2`
  `_:c2 rdf:first <ex:bar>`
  `_:c2 rdf:rest rdf:nil`

- Note that rdf:nil is of type list. rdf:first and rdf:rest have no assertions.

- **Legal:**
  
  `_:666 rdf:first <ex:foo>`
  `_:666 rdf:first <ex:bar>`
  `_:666 rdf:rest <ex:blee>`
  `_:666 rdf:rest rdf:nil`

- Collections are “closed”, that is they have a fixed membership.
RDF – rdf:value

- Used to identify a 'primary' or 'main' value of a property which has several values, or has as its value a complex entity with several facets or properties of its own.

- Example:

  `exproduct:scblur  exterms:weight  "5.1"^^xsd:decimal`  

  *What are the units?* Instead write:

  `exproduct:items:scblur  exterms:weight  _:weights:scblur  
  _:weights:scblur  rdf:value  "5.1"^^xsd:decimal  
  _:weights:scblur  exterms:units  exunits:pounds`
RDF – Axiomatic Triples

- `rdf:type` rdf:type rdf:Property
- `rdf:subject` rdf:type rdf:Property
- `rdf:predicate` rdf:type rdf:Property
- `rdf:object` rdf:type rdf:Property
- `rdf:first` rdf:type rdf:Property
- `rdf:rest` rdf:type rdf:Property
- `rdf:value` rdf:type rdf:Property
- `rdf:_1` rdf:type rdf:Property
- `rdf:_2` rdf:type rdf:Property
- ...
- `rdf:nil` rdf:type rdf:List
RDF/XML Syntax

XML Syntax to describe:
- Containers and Collections
- Reification
- rdf:value
- rdf:XMLLiteral
RDF/XML Syntax

- RDF Triplet Syntax:
  
  ```
  ex:index.html exterms:creation-date "October 21, 2006"
  ex:index.html dc:language "en"
  ex:index.html dc:creator exstaff:P96070
  ```

- RDF/XML Syntax:
  
  ```
  1.  <?xml version="1.0"?>
  2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  3.             xmlns:dc="http://purl.org/dc/elements/1.1/"
  4.             xmlns:exterms="http://www.example.org/terms/">
  5.    <rdf:Description rdf:about="http://www.example.org/index.html">
  6.      <exterms:creation-date>October 21, 2006</exterms:creation-date>
  7.      <dc:language>en</dc:language>
  8.      <dc:creator rdf:resource="http://www.example.org/staffid/P96070"/>
  9.    </rdf:Description>
  10.  </rdf:RDF>
  ```
RDF/XML Syntax – Blank Node

1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:dc="http://purl.org/dc/elements/1.1/"
4.             xmlns:exterms="http://example.org/stuff/1.0/">
5.     <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
6.         <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
7.         <exterms:editor rdf:nodeID="abc"/>
8.     </rdf:Description>
9.     <rdf:Description rdf:nodeID="abc">
10.        <exterms:fullName>Dave Beckett</exterms:fullName>
12.     </rdf:Description>
13. </rdf:RDF>
RDF/XML Syntax – Typed Literal with XML Entity

1. <?xml version="1.0"?>
2. <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>

3. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:exterms="http://www.example.org/terms/">

4.   <rdf:Description rdf:about="http://www.example.org/index.html">
      <exterms:creation-date rdf:datatype="&xsd;date">2006-10-22</exterms:creation-date>
   </rdf:Description>

7.  </rdf:RDF>
RDF/XML Syntax – ID and Base

1. <?xml version="1.0"?>
2. <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"]>}
3. <rdf:RDF xmlns:exterms="http://www.example.com/terms/"
4.       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6.   <exterms:Blur rdf:ID="scblur">
8.     <exterms:model rdf:datatype="&xsd:string">Blur</exterms:model>
9.     <exterms:colour rdf:datatype="&xsd:string">Blue</exterms:colour>
10.    <exterms:weight rdf:datatype="&xsd;decimal">5.1</exterms:weight>
11.   </exterms:Blur>
12. </rdf:RDF>
RDF/XML Syntax – Bag, Seq, Alt

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/mitch/friends#">
  <rdf:Description rdf:about="http://example.org/mitch/10222006">
    <s:friends>
      <rdf:Bag>
        <rdf:li rdf:resource="http://example.org/mitch/Amy"/>
        <rdf:li rdf:resource="http://example.org/mitch/Eliyahu"/>
        <rdf:li rdf:resource="http://example.org/mitch/Malka"/>
        <rdf:li rdf:resource="http://example.org/mitch/Shanti"/>
        <rdf:li rdf:resource="http://example.org/mitch/Yoni"/>
      </rdf:Bag>
    </s:friends>
  </rdf:Description>
</rdf:RDF>
```

- `rdf:Bag`, `rdf:Seq`, and `rdf:Alt` are the allowed XML elements.
- `rdf:li` is used as a shorthand for the container elements. `rdf_1`, `rdf_2`, ... can be used instead.
RDF/XML Syntax – Collections

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/mitch/friends#">
  <rdf:Description rdf:about="http://example.org/mitch/10222006">
    <s:friends rdf:parseType="Collection">
      <rdf:Description rdf:about="http://example.org/mitch/Amy"/>
      <rdf:Description rdf:about="http://example.org/mitch/Yoni"/>
      <rdf:Description rdf:about="http://example.org/mitch/Malka"/>
    </s:friends>
  </rdf:Description>
</rdf:RDF>
```

- `rdf:first` and `rdf:rest` can be used instead (in conjunction with `rdf:nodeID`) to generate lists instead.
RDF/XML Syntax – Reification

```xml
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:externs="http://www.example.com/terms/"
     xml:base="http://www.example.com/2002/04/products">

  <rdf:Statement rdf:about="#triple12345">
    <rdf:subject rdf:resource="http://www.example.com/2002/04/products#item10245"/>
    <rdf:predicate rdf:resource="http://www.example.com/terms/weight"/>
    <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>
  </rdf:Statement>

</rdf:RDF>
```
RDF/XML Syntax – rdf:Value

```xml
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:exterms="http://www.sc.org/terms/">
    <rdf:Description rdf:about="http://www.sc.com/04/products#blr">
        <exterms:weight rdf:parseType="Resource">
            <rdf:value rdf:datatype="&xsd;decimal">5.1</rdf:value>
            <exterms:units rdf:resource="http://www.sc.org/units/pounds"/>
        </exterms:weight>
    </rdf:Description>
</rdf:RDF>
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xml:base="http://www.example.com/articles">
    <rdf:Description rdf:ID="article1245">
        <dc:title rdf:parseType="Literal">
            <span xml:lang="en">
                The <em>&lt;br /&gt;</em> Pluto still planet.
            </span>
        </dc:title>
    </rdf:Description>
</rdf:RDF>
RDF Schema

- Extension to RDF.
  - As we have seen, RDF has no notion of resource attributes or relationships between attributes and resources.
  - RDF Schema is defined as a series of RDF statements.
- Describes classes and properties somewhat similar to typical programming languages.
  - Based upon an open-world assumption.
  - Properties are described in terms of the classes of resource to which they apply.
RDF Schema (Continued)

- Provides information as additional descriptions of resources.
  - Just like RDF, not prescriptive.
- RDF Vocabulary Description Language is the official name for RDF Schema.
RDF Schema Vocabulary

- Classes
- Properties
- Container Classes and Properties
- RDF Collections
- Reification Vocabulary
- Utility Properties
RDFS Class – rdfs:Resource

- Class of everything – all classes subclass this class.
RDFS Class – rdfs:Class

- Class is an instance of class.
  - Formally, class is a type of class; resources are a type of class too.
    - rdfs:Class rdf:type rdfs:Class
    - rdfs:Resource rdf:type rdfs:Class
RDFS Class – rdfs:Literal

- The class of literal values.
  - Literals may be plain or typed.
    - A typed literal is an instance of the rdfs:Datatype class.
RDFS Class – rdfs:Datatype

- Class of all data types.
- Instances of rdfs:Datatype are the simple XML Schema datatypes, etc.

Definition:

- rdfs:Datatype rdfs:subClassOf rdfs:Class
- rdfs:Datatype rdf:type rdfs:Class
RDFS Class – rdfs:XMLLiteral

- rdfs:XMLLiteral is just the class of well-typed XML.

**Definition:**
- `rdfs:XMLLiteral rdf:type rdfs:Datatype`
- `rdfs:XMLLiteral rdfs:subClassOf rdfs:Literal`
RDFS Class – rdfs:Property

- Class of all properties.
- rdfs:Property is a relation between subject resources and object resources.

Definition:
- rdf:Property rdf:type rdfs:Class

Example:
- ex:age rdf:type rdf:Property
- ex:age rdfs:range xsd:integer

- Note that this is a global definition for the ex:age URI reference.
RDFS Property – rdfs:range

- Defines that values of a property are instances of one or more classes.

- Example:
  
  ```
  my:hasParts rdfs:range my:CarParts
  my:hasParts rdfs:range my:BusParts
  ```

- Note that classes are defined by the ranges that are applicable to them.
  
  ➢ This allows classes to grow w/o redefinition.

- Also, note that there are no cardinality restrictions.
RDFS Property – rdfs:domain

- Defines that any resource that has a given property is an instance of one or more classes.

- Example:
  - `my:hasParts rdfs:domain my:Car`
  - `my:color rdfs:domain my:Car`
  - `my:color rdfs:domain my:Bus`

- Note that classes are defined by the domains that are applicable to them.
  - This allows classes to grow w/o redefinition.

- Also, note that there are no cardinality restrictions.
RDFS Property – rdfs:type

- Defines that a resource is an instance of a class.

- Example:
  
  ```
  xs:string rdf:type rdfs:Datatype
  ```
RDFS Property – `rdfs:subClassOf`

- All the instances of one class are instances of another.
- This property is transitive.
- Example:
  
  `my:prius rdfs:subClassOf my:car`
RDFS Property – rdfs:subPropertyOf

- Resources related by one property are also related by another.
- This property is transitive.

\[ \text{my:hasCylinders} \ rdfs:subPropertyOf \ \text{my:hasParts} \]
RDFS Property – rdfs:label

- Provides a human-readable version of a resource's name.

\[
\text{exproduct:scblur rdfs:label} \quad \text{"Santa Cruz Blur"^^xsd:string}
\]
RDFS Property – rdfs:comment

- Provides a human-readable description of a resource.

```rdfs
exproduct:scblur exterms:weight _:scblur
_:scblur rdf:value "5.1"^^xsd:decimal
_:scblur exterms:units exunits:pounds
_:scblur rdfs:comment
"weight is 5.1pds"^^xsd:string
```
RDFS Container Classes

- rdfs:Container – Superclass of all containers.
- rdf:Bag, rdf:Seq, and rdf:Alt definition:
  - rdf:Alt rdfs:subClassOf rdfs:Container
  - rdf:Bag rdfs:subClassOf rdfs:Container
  - rdf:Seq rdfs:subClassOf rdfs:Container
  - rdf:Seq rdf:type rdfs:Class
  - rdf:Bag rdf:type rdfs:Class
  - rdf:Alt rdf:type rdfs:Class
  - rdfs:Container rdf:type rdfs:Class

- All other semantics are the same as in RDF.
RDFS Container Properties

- `rdfs:member` - Super-property of all the container membership properties.
- `rdfs:ContainerMembershipProperty` – States that a resource is a member of a container.

**Definition:**

- `rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property`  
  `rdfs:ContainerMembershipProperty rdfs:subPropertyOf rdfs:member`  
  `rdf:_1 rdf:type rdfs:ContainerMembershipProperty`  
  `rdf:_2 rdf:type rdfs:ContainerMembershipProperty`
RDFS List

- **rdf:List, rdf:first, rdf:rest, rdf:.nil**

- **Definition:**
  - \( \text{rdf:List} \quad \text{rdf:type} \quad \text{rdfs:Class} \)
  - \( \text{rdf:first} \quad \text{rdf:type} \quad \text{rdfs:Property} \)
  - \( \text{rdf:last} \quad \text{rdf:type} \quad \text{rdfs:Property} \)
  - \( \text{rdf:first} \quad \text{rdfs:domain} \quad \text{rdf:List} \)
  - \( \text{rdf:rest} \quad \text{rdfs:domain} \quad \text{rdf:List} \)
  - \( \text{rdf:first} \quad \text{rdfs:range} \quad \text{rdfs:Resource} \)
  - \( \text{rdf:rest} \quad \text{rdfs:range} \quad \text{rdf:List} \)

  - All other semantics are the same as in RDF.
RDFS Reification

- **rdf:Statement, rdf:subject, rdf:predicate, rdf:object**
- **Definition:**
  
  ```
  rdf:Statement rdf:type    rdfs:Class
  rdf:subject   rdf:type    rdfs:Property
  rdf:predicate rdf:type    rdfs:Property
  rdf:object    rdf:type    rdfs:Property
  rdf:subject   rdfs:domain rdf:Statement
  rdf:predicate rdfs:domain rdf:Statement
  rdf:object    rdfs:domain rdf:Statement
  rdf:subject   rdfs:range  rdfs:Resource
  rdf:predicate rdfs:range  rdfs:Resource
  rdf:object    rdfs:range  rdfs:Resource
  ```

- All other semantics are the same as in RDF.
RDFS Utility Properties

- `rdfs:seeAlso` – indicates a resource that might provide additional information about the subject resource.
- `rdfs:isDefinedBy` – indicates a resource defining the subject resource. May be used to indicate an RDF vocabulary in which a resource is described.
- `rdf:value` – same semantics as in RDF.
- **Definition:**
  
  - `rdfs:seeAlso`      `rdf:type`     `rdf:Property`
  - `rdfs:isDefinedBy`  `rdf:type`     `rdf:Property`
  - `rdf:value`         `rdf:type`     `rdf:Property`
  - `rdf:value`         `rdfs:range`  `rdfs:Resource`
  - `rdf:value`         `rdfs:domain` `rdfs:Resource`
  - `rdfs:seeAlso`      `rdfs:range`  `rdfs:Resource`
  - `rdfs:seeAlso`      `rdfs:domain` `rdfs:Resource`
  - `rdfs:isDefinedBy`  `rdfs:range`  `rdfs:Resource`
  - `rdfs:isDefinedBy`  `rdfs:domain` `rdfs:Resource`
RDFS as RDF/XML – 1

```xml
<rdf:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Resource</rdfs:label>
  <rdfs:comment>The class resource, everything.</rdfs:comment>
</rdf:Class>

<rdf:Property rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>type</rdfs:label>
  <rdfs:comment>The subject is an instance of a class.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>

<rdf:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Class">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Class</rdfs:label>
  <rdfs:comment>The class of classes.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Class>
```
RDFS as RDF/XML – 2

```xml
<rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#subClassOf">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>subClassOf</rdfs:label>
  <rdfs:comment>The subject is a subclass of a class.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#subPropertyOf">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>subPropertyOf</rdfs:label>
  <rdfs:comment>The subject is a subproperty of a property.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>
```
RDFS as RDF/XML – 3

```xml
<rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#subClassOf">
  <rdfs:isDefinedBy
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>subClassOf</rdfs:label>
  <rdfs:comment>The subject is a subclass of a class.</rdfs:comment>
  <rdfs:range
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Property>
```
Semantic Web Layering

From: Berners-Lee XML 2000
Tool Demos
SPARQL – RDF Query Language

- Facilities include:
  - To extract information in the form of URIs, blank nodes, plain and typed literals.
  - To extract RDF subgraphs.
  - To construct new RDF graphs based on information in the queried graphs.

- Working Draft (was Candidate Recommendation)
SPARQL – Example

- **RDF Triplet:**

  ```
  <http://sc.com/products/Blur>
  <http://purl.org/dc/elements/1.1/title>
  "The Best Bike ever made"
  ```

- **Query:**

  ```
  SELECT ?title
  WHERE
  {
    <http://sc.com/products/Blur>
    <http://purl.org/dc/elements/1.1/title>
    ?title
  }
  ```

- **Query Result:**

  ```
  title
  "The Best Bike ever made"
  ```
SPARQL – Syntax

- **IRI (generalization of URI)**
  - Can be prefixed or relative to a base.

- **Literals**
  - Enclosed in double or single quotes.
  - Integers, floats, doubles and booleans can be written w/o quotes or a datatype for convenience.

- **Variables**
  - Have global scope.
  - Prefixed by ? or $. ($a and ?a refer to the same variable).
SPARQL – Blank Nodes Results

- Blank nodes can be selected in the query results.

  - RDF Triplets:
    
    ```
    _:a <http://purl.org/dc/elements/1.1/title> "The Best Bike ever made"
    _:a rdf:Value "$1500.00"
    _:b <http://purl.org/dc/elements/1.1/title> "The Best Mountain Bike ever made"
    _:b rdf:Value "$1600.00"
    ```

  - Query:
    
    ```
    SELECT ?x ?title
    WHERE
    {
    }
    ```

  - Query Results:
    
    ```
    x   title
    _:q "The Best Bike ever made"
    _:a "The Best Mountain Bike ever made"
    ```
SPARQL – Blank Nodes Query

- A blank node only used once in a query can be specified with the [] syntax.
  - Example:  [] rdf:Value "fence"
  - Could have written:  _:b rdf:Value "fence"

- A blank node used many times in a query can be specified with the [:x :y] syntax.
  - Examples:  [rdf:Value "fence"]
               [rdf:Value "fence"] rdf:Value "post"
SPARQL – Collections

- Can enclose collection elements in "()".
  - Example: ("a", ?x)
  - Short for:
    
    _:a rdf:first "a"
    _:a rdf:rest _:b
    _:b rdf:first ?x
    _:b rdf:rest rdf:nil
SPARQL – Value Constraints

- Can filter RDF statements based upon the value of an object.
  - RDF Triplets:
    
    ```
    <http://sc.com/products/Blur>  
    <http://purl.org/dc/elements/1.1/title>  
    "The Best Mountain Bike sold!"
    
    <http://example.org/products/Blur>  
    <http://purl.org/dc/elements/1.1/price>  
    "1500.00"
    ```
  
  - Query:
    ```
    SELECT ?title ?price  
    WHERE  
    {
      FILTER (?price < 2000.00)
    }
    ```
  
  - Query Results:
    ```
    title                           price  
    "The Best Mountain Bike sold!" "1500.00"
    ```
SPARQL – Optional Pattern Matching

- Can specify optional patterns and optional filters (similar to a SQL outer join).

  RDF Triplets:
  ```
  @prefix dc:   <http://purl.org/dc/elements/1.1/> .
  @prefix ns:   <http://sc.com/ns#> .
  @prefix :     <http://sc.com/products/> .

  :scblur   dc:title        "The Best Mountain Bike sold!"
  :scblur   ns:price        "1500.00"^^xsd:decimal
  :scxcblur dc:title        "Best XC Mountain Bike Sold"
  :scxcblur ns:specialprice "Call for quote"^^xsd:string
  ```

  Query:
  ```
  PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
  PREFIX  ns:  <http://example.org/ns#>
  SELECT ?title ?price
  WHERE {
    OPTIONAL { ?x ns:price ?price . FILTER (?price < 100) }
  }
  ```

  Query Results:
  ```
  title                                    price
  "Best XC Mountain Bike Sold"             
  ```
**SPARQL – UNION**

- Unions together two or more subgraphs.

  - **RDF Triplets:**
    
    ```
    @prefix dc:   <http://purl.org/dc/elements/1.1/> .
    @prefix ns:   <http://sc.com/ns#> .
    @prefix :     <http://sc.com/products/> .
    
    :scblur   dc:title        "The Best Mountain Bike Sold!"
    :scblur   ns:price        "1500.00"^^xsd:decimal
    :scxcblur dc:title        "Best XC Mountain Bike Sold!"
    :scxcblur ns:specialprice "Call for quote"^^xsd:string
    ```

  - **Query:**
    
    ```
    PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
    PREFIX  ns:  <http://example.org/ns#>
    SELECT ?title ?price
          {?x ns:price ?price} UNION {?x ns:specialprice ?price } }
    ```

  - **Query Results:**
    
    ```
    title                                    price
    "The Best Mountain Bike Sold!"           "1500.00"^^xsd:decimal
    "Best XC Mountain Bike Sold!"            "Call for quote"^^xsd:string
    ```
A RDF data store can hold multiple RDF graphs.

A RDF dataset can represent this collection of graphs.

Each RDF graph can be identified by an IRI.

By definition, a RDF dataset has one unnamed graph.
SPARQL – GRAPH

GRAPH – used to match patterns against named graphs.

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?src ?AmandaBall
WHERE {
  GRAPH ?src
  { ?x foaf:mbox <mailto:mandyball@gmail.com> .
    ?x foaf:nick ?AmandaBall } }

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX data: <http://example.org/foaf/>
SELECT ?AmandaBall
WHERE {
  GRAPH data:mitchFoaf
  { ?x foaf:mbox <mailto:mandyball@gmail.com> .
    ?x foaf:nick ?AmandaBall } }
SPARQL – FROM, FROM NAMED

- The FROM clause specifies the IRI for the default graph.
- The FROM NAMED clause specifies the IRI for the named graph.
  - Multiple FROM NAMED clauses can exist in a single query.
SPARQL – FROM, FROM NAMED

Example (1)

# Default graph (stored at http://any.org/dft.ttl)
@prefix dc: <http://purl.org/dc/elements/1.1/> .


########################################

# Named graph: http://any.org/jim
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Jim" .
_:a foaf:mbox <mailto:jim@gmail.com> .

########################################

# Named graph: http://any.org/bob
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@gmail.com> .
QUERY:
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?who ?g ?mbox
FROM <http://any.org/dft.ttl>
FROM NAMED <http://any.org/jim>
FROM NAMED <http://any.org/bob>
WHERE
{
    ?g dc:publisher ?who .
    GRAPH ?g { ?x foaf:mbox ?mbox }
}
RESULTS:

who     g                      mbox
"Jim Johnson"   <http://any.org/jim>   <mailto:jim@gmail.com>
"Bob Smith"     <http://any.org/bob>   <mailto:bob@gmail.com>
SPARQL – Solution Modifiers

- ORDER BY – order the solutions.
  
  ```sparql
  SELECT ?name
  WHERE { ?x foaf:name ?name ; :foaf:nick ?nick }
  ORDER BY DESC(?nick)
  ```

- DISTINCT – create unique solutions.

- LIMIT – restrict the number of solutions returned.
  
  ```sparql
  SELECT ?name WHERE { ?x foaf:name ?name } LIMIT 20
  ```

- OFFSET – determine from where, in the overall sequence of solutions, to start.
  
  ```sparql
  SELECT ?name WHERE { ?x foaf:name ?name } LIMIT 20
  OFFSET 15
  ```
SPARQL – CONSTRUCT

- Generates a RDF graph from the query solution.
  - Blank nodes are handled as well.

```sparql
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@gmail.com> .

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT
{ <http://example.org/person#Bob> vcard:FN ?name } WHERE
{ ?x foaf:name ?name }

Result:

@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
<http://example.org/person#Bob> vcard:FN "Bob"
```
Can return a whole graph, subgraph, or filtered subgraph.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX app: <http://sc.com/ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

CONSTRUCT { ?s ?p ?o } WHERE {
  { ?g xsd:decimal ?cost } .
  FILTER ( app:toEuros(?cost) = "200"^^xsd:decimal ) .
```
SPARQL – Operators, Tests, and Accessors

- SPARQL has a subset of the XPath tests
  - >, <, !=, and, or, etc.
  - Can also determine if two RDF terms are equal or not.

- Basic math operations

- Regular expression matching

- Also, has its own test and accessors
  - BOUND, isURI, isIRI, isBLANK, isLITERAL, etc.

- Can also call external extension functions.
SPARQL Protocol for RDF (1)

- Send SPARQL queries to query processors.
- Standard has abstract interface and SOAP bindings.
- WSDL has one interface and one operation (WSDL fragment):

```xml
<!-- Abstract SparqlQuery Interface -->
<interface name="SparqlQuery"
    styleDefault="http://www.w3.org/2006/01/wsdl/style/iri">

    <!-- the Interface Faults -->
    <fault name="MalformedQuery"
        element="st:malformed-query"/>
    <fault name="QueryRequestRefused"
        element="st:query-request-refused"/>

```
SPARQL Protocol for RDF (2)

<!-- the Interface Operation -->
<operation name="query"
  pattern="http://www.w3.org/2006/01/wSDL/in-out">
  <input messageLabel="In"
    element="st:query-request"/>
  <output messageLabel="Out"
    element="st:query-result"/>

  <!-- the interface faults are out faults -->
  <outfault ref="tns:MalformedQuery"
    messageLabel="Out"/>
  <outfault ref="tns:QueryRequestRefused"
    messageLabel="Out"/>
</operation>
</interface>
Query request XML Schema fragment:

```xml
<xsd:element name="query-request">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="query" type="xsd:string"/>
      <xsd:element minOccurs="0" maxOccurs="unbounded" name="default-graph-uri" type="xsd:anyURI"/>
      <xsd:element minOccurs="0" maxOccurs="unbounded" name="named-graph-uri" type="xsd:anyURI"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```
SPARQL Protocol for RDF (3)

- Query response XML Schema fragment:

```xml
<xs:element name="query-result">
  <xs:complexType>
    <xs:choice>
      <xs:element maxOccurs="1" ref="vbr:sparql"/>
      <xs:element maxOccurs="1" ref="rdf:RDF"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

- Either use SPARQL XML result format or just plain RDF.
SPARQL Protocol for RDF (4)

**SPARQL Query Results XML Format**

```xml
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="name"/>
    <variable name="nextversion"/>
  </head>
  <results ordered="false" distinct="false">
    <result>
      <binding name="name">
        <literal datatype="http://www.w3.org/2001/XMLSchema#string">perl</literal>
      </binding>
      <binding name="nextversion">
        <literal datatype="http://www.w3.org/2001/XMLSchema#string">6</literal>
      </binding>
    </result>
  </results>
</sparql>
```
SPARQL Query Engines Demo

- Opera SPARQL Query Engine
  ➢ http://www.myopera.com/community/sparql

- GovTrack.us - Independently Tracking the United States Congress
  ➢ http://www.govtrack.us/sparql.xpd

- XML Army Knife
  ➢ http://xmlarmyknife.org/api/rdf/sparql/query

- Redland Rasqal RDF Query Demonstration
  ➢ http://librdf.org/query/
Questions