Summary of RDF/XML

The following is the basic way to represent a triple.

Pattern

```
<rdf:Description rdf:about=Subject>
  <Property>Object</Property>
</rdf:Description>
```

We refer to *Property* here as a "property element."

Example

```
<rdf:Description rdf:about="http://www.example.org/index.html">
  <exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>
```

Represent an RDF-graph with multiple triples by including multiple `rdf:Description` elements in the content of the `rdf:RDF` element

Pattern

```
<rdf:Description rdf:about=Subject>
  <Property>Object</Property>
</rdf:Description>
<rdf:Description rdf:about=Subject>
  <Property>Object</Property>
</rdf:Description>
```

Etc.

Example

```
<rdf:Description rdf:about="http://www.example.org/index.html">
      <exterms:creation-date>August 16, 1999</exterms:creation-date>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.example.org/index.html">
      <dc:language>en</dc:language>
  </rdf:Description>
```

When the same subject (resource) is described with several properties and values (objects), we can represent these property-object pairs inside the `rdf:Description` element identifying the subject.

Pattern

```
<rdf:Description rdf:about=Subject>
  <Property1>Object</Property1>
  <Property2>Object</Property2>
  Etc.
</rdf:Description>
```

Example

```
<rdf:Description rdf:about="http://www.example.org/index.html">
  <exterms:creation-date>August 16, 1999</exterms:creation-date>
  <dc:language>en</dc:language>
</rdf:Description>
```

If the value of a property element is a resource, we use an empty element

```
<Property rdf:resource="Object" />
```

The `rdf:resource` attribute indicates that the property element's value is another resource, identified by its URIref. Since the URIref is used as an attribute value, XML requires the URIref to be written out as an absolute or relative URIref and not abbreviated as a QName.

Example

```
<dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
```

Property Attributes

When a property element's content is a literal, we can capture it as an attribute on the containing `rdf:Description` element provided the property element name isn't repeated (since XML attribute names must be unique).

Pattern

```
<rdf:Description rdf:about="Subject"
                 Property1="Object1" Property2="Object2" …>
 …
</rdf:Description>
```

Example

```
<rdf:Description rdf:about="http://www.example.org/index.html"
                 dc:language="en"
                 exterms:creation-date="August 16, 1999">
  <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
</rdf:Description>
```

We can do this also when the property element is `rdf:type` and it has an `rdf:resource` attribute whose value is a URIref.

Example

```
<rdf:Description
       rdf:about="http://www.example.org/index.html"
       rdf:type="http://www.w3.org/1999/02/22-rdf-syntax-ns#Document">
  <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
</rdf:Description>
```

Blank Nodes

The most direct way to representing bnodes in RDF/XML is to assign a blank node (or bnode) identifier to each blank node in the document. A bnode identifier is not a URIref—you make up a (short) name. A statement with a bnode as subject is written using an `rdf:Description` element that has an `rdf:nodeID` (instead `rdf:about`) attribute with the bnode identifier as its value. A statement with a bnode as object is written using a property element that has an `rdf:nodeID` (instead of `rdf:resource`) attribute.

Pattern

```
<rdf:Description rdf:about="Subject">
  <Property1 rdf:nodeID="BNId"/>
  …
</rdf:Description>
…
<rdf:Description rdf:nodeID=" BNId">
  <Property2> Object2 </Property2>
  …
</rdf:Description>
```

Example

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
  <exterms:editor rdf:nodeID="abc"/>
</rdf:Description>
<rdf:Description rdf:nodeID="abc">
  <exterms:fullName>Dave Beckett</exterms:fullName>
</rdf:Description>
```

A bnode can be written so that the `<rdf:Description> … </rdf:Description>` tags are omitted from its representation, and it's absorbed into the `rdf:Description` element where it's referenced as object. For this, do the following.

1. Put an `rdf:parseType="Resource"` attribute on the (empty) property element (that has the bnode as object) and remove its `rdf:nodeID` attribute.
2. Put the property elements describing the bnode's properties as its content.
3. Eliminate the `rdf:Description` element for the bnode subject.

Pattern

The bnode pattern above is transformed to

```
<rdf:Description rdf:about="Subject">
  <Property1 rdf:parseType="Resource">
    <Property2> Object2 </Property2>
    …
  </Property1>
  …
</rdf:Description>
```

Example
The bnode example above is transformed to

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
  <exterms:editor rdf:parseType="Resource">
    <exterms:fullName>Dave Beckett</exterms:fullName>
  </exterms:editor>
</rdf:Description>
```

Further abbreviation may be possible. If (1) all property elements on a blank node element have string values, (2) each of these property elements appears at most once, and (3) there's at most one `rdf:type` property element with a URIref object node, then these property elements can be moved to be property attributes on the containing property element (introduced when the `rdf:Description` element for the bnode was eliminated). Then also remove the `rdf:parseType` attribute from the containing property element and make it empty.

Pattern
The last bnode pattern is transformed to

```
<rdf:Description rdf:about="Subject">
  <Property1 Property2=Object2 .../>
  ...
</rdf:Description>
```

Example
The last example is

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
    <exterms:editor exterms:fullName="Dave Beckett" />
</rdf:Description>
```

Typed Literals
A typed literal is represented by adding to the property element containing the literal an `rdf:datatype` attribute whose value is the datatype URIref. The datatype is usually an XML Schema datatype, in the namespace `http://www.w3.org/2001/XMLSchema#`, corresponding conventionally to the prefix `xsd`. Recall that an XML attribute value can't be written as a QName but must be written with the full URIref

Example

```
<rdf:Description rdf:about="http://www.example.org/index.html">
  <exterms:creation-date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
    1999-08-16
  </exterms:creation-date>
</rdf:Description>
```

Use an internal mixed-content entity with name `xsd` and value
`"http://www.w3.org/2001/XMLSchema#"`
so that the value of the `rdf:datatype` attribute here may be abbreviated. This is all that's in the internal subset, which, besides the root element's name (`rdf:RDF`), is all that's in the document type declaration:

```
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
```

We now abbreviate the attribute value "http://www.w3.org/2001/XMLSchema#date" as "&xsd;date". The entire document is as follows.

```
<?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.example.org/terms/">
  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date rdf:datatype="&xsd;date">
      1999-08-16
    </exterms:creation-date>
  </rdf:Description>
</rdf:RDF>
```

Abbreviating and Organizing RDF URIrefs

RDF doesn't specify or control how URIrefs are assigned to resources (which involves "resolving" URIs). But sometimes we want to achieve the effect of assigning a URIref (specifically, a URI and a fragment identifier) to a resource (identified by the fragment identifier) that's part of an organized group of resources (identified by the URI alone).

The value of attribute `rdf:ID` is a fragment identifier interpreted relative to a base URI. The `rdf:ID` attribute is like the `id` attribute in XML and HTML: it defines a name that's unique relative to the current base URI. Another element in the document can refer to the element with the `rdf:ID` attribute by using either the absolute URIref (with base URI and fragment identifier) or the relative URIref ('#' followed by the fragment identifier), which is relative to the base URI. RDF outside the document can refer to the element with the `rdf:ID` attribute with the full URIref.

Instead of `rdf:ID="fragID"`, we could use `rdf:about="fragID"` for the same effect. As an abbreviation mechanism, the 2 forms are synonyms: the full URIref is the same in either case. But `rdf:ID` provides an additional check: a given `rdf:ID` value must be unique relative to the same base URI.

Example

The following is a description of a tent in a sporting goods catalog whose URL is

```
   http://www.example.com/2002/04/products
```

```
<rdf:Description rdf:ID="item10245">
  <exterms:model rdf:datatype="&xsd;string">Overnighter</exterms:model>
  <exterms:sleeps rdf:datatype="&xsd;integer">2</exterms:sleeps>
  <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
  <exterms:packedSize rdf:datatype="&xsd;integer">784</exterms:packedSize>
</rdf:Description>
```

The following is the description of a review of this tent.

```
<rdf:Description rdf:about="http://www.example.com/2002/04/products#item10245">
  <sportex:ratingBy rdf:datatype="&xsd;string">
    Richard Roe
  </sportex:ratingBy>
  <sportex:numberStars rdf:datatype="&xsd;integer">
    5
  </sportex:numberStars>
</rdf:Description>
```

Base URI

Relative URIrefs are interpreted relative to a base URI. By default, the base URI is the URI of the resource where the relative URIref is used. RDF/XML, however, supports XML Base, which allows an XML document to specify a base URI other than the URI of the document itself.

Example

Here the `xml:base` declaration specifies that the base URI for the content within the `rdf:RDF` element (until another `xml:base` attribute is specified) is

`http://www.example.com/2002/04/products`

All relative URIrefs cited within that content are interpreted relative to that base.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.com/terms/"
         xml:base="http://www.example.com/2002/04/products">
  ...
</rdf:RDF>
```

Typed Nodes

RDF supports the concept of a class to which things belong (as instances) by providing predefined property `rdf:type`. RDF itself doesn't provide facilities for defining application-specific classes of things or their properties. Such classes are described using RDFS.

Resources with `rdf:type` properties are called typed nodes in the graph or typed node elements in the RDF/XML. RDF/XML has a special abbreviation for describing typed nodes:

1. The `rdf:type` property and its value are removed.
2. The `rdf:Description` element for the node is replaced by an element whose name is the QName corresponding to the value of the removed `rdf:type`.

Example

```
<rdf:Description rdf:ID="item10245">
    <rdf:type rdf:resource="http://www.example.com/terms/Tent"/>
  ...
</rdf:Description>
```

This can be transformed to

```
<exterms:Tent rdf:ID="item10245">
  …
</exterms:Tent>
```

A resource may have more than one rdf:type property, but only 1 of these rdf:type properties may be abbreviated.

The typed node abbreviation is also used in RDF/XML when describing instances of the built-in RDF classes (e.g., `rdf:Bag`) and the built-in RDF Schema classes (such as `rdfs:Class`)