Semantic Web Engineering

Gerald Reif

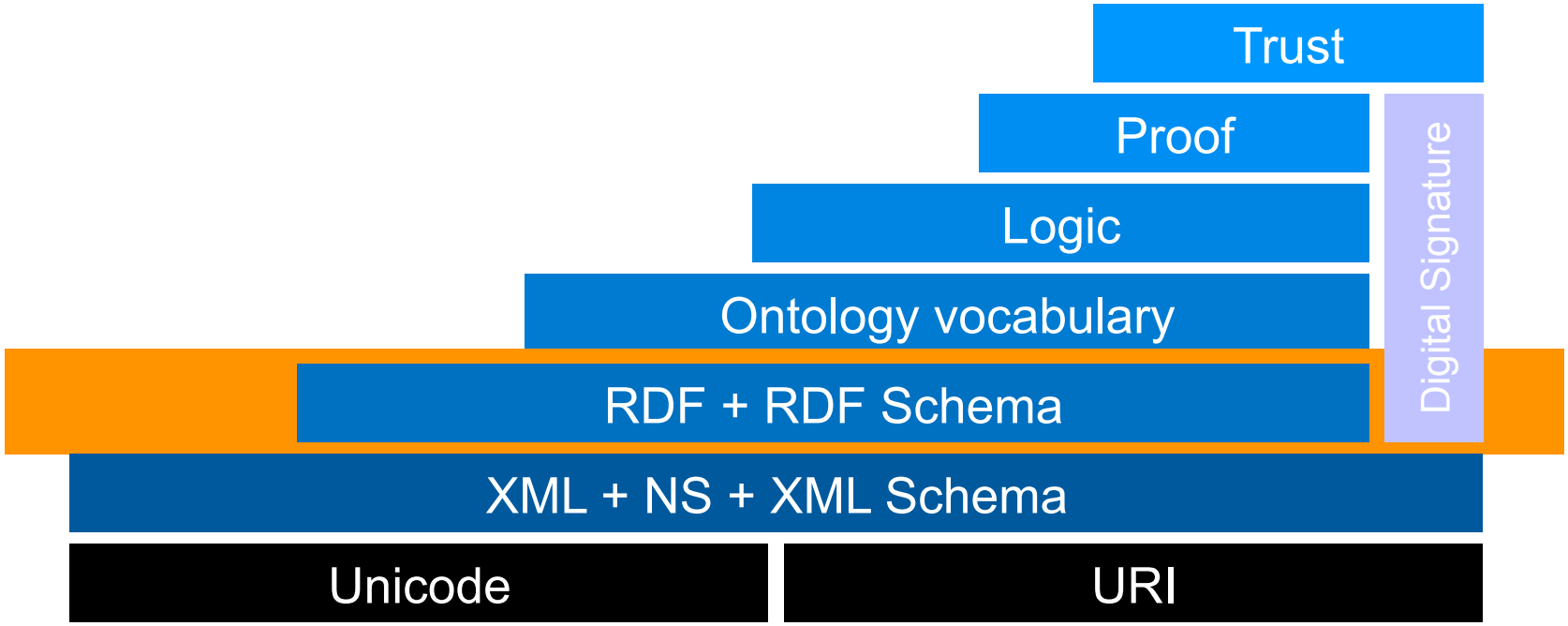
reif@ifi.unizh.ch

Fr. 10:15-11:45, Room 2.A.10

University of Zurich
Department of Informatics

s.e.a.l.
software evolution & architecture lab

The Resource Description Framework RDF



Suggested Reading

- The examples are taken for the W3C RDF Primer
  - http://www.w3.org/TR/rdf-primer/
- Further reading on the RDF/XML syntax
  - http://www.w3.org/TR/rdf-syntax-grammar/
- Detailed information:
  - Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation 10 February 2004 http://www.w3.org/TR/rdf-concepts/

Drawbacks of XML

- XML is a universal meta-language for defining markup
- It provides a uniform framework for interchange of data and metadata between applications
- However, XML does not provide any means of talking about the semantics (meaning) of data
- e.g., there is no intended meaning associated with the nesting of tags
  - It is up to each application to interpret the nesting.

Nesting of Tags in XML

Statement: *Gerald Reif is a lecturer of Semantic Web Engineering*

```
<course name="Semantic Web Engineering">
        <lecturer>Gerald Reif</lecturer>
</course>

<lecturer name="Gerald Reif">
        <teaches>Semantic Web Engineering</teaches>
</lecturer>

<lecturer>
        <name>Gerald Reif</name>
        <teaches>Semantic Web Engineering</teaches>
</lecturer>
```

Opposite nesting, same information!

RDF Goals 1/2

- RDF is a language for representing information about resources in the World Wide Web.
- By generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be *identified* on the Web.
  - Even when they cannot be directly retrieved on the Web.
  - e.g., persons, products, etc.

RDF Goals 2/2

- RDF is intended for situations in which this information needs to be processed by applications, rather being only displayed to people.
- RDF can be exchanged between applications without loss of meaning.
- The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created.
- RDF provides a common framework, therefore application designers can leverage the availability of common RDF parsers and processing tools.
  - Like XML for semi-structured Data.

RDF Triples

- RDF is based on the idea of identifying things (called resources) using URIs.
- Resources are described in triples.
- Triples are also called statements.
- The RDF triples consist of:
  - Subject - Predicate - Object
- Sample Statement:

  The CD with the item number “1234” has the title “Alanis Unplugged”.

Subject

- The Subject is the thing (the resource) we want to make a statement about.
- In our example we want to make a statement about a CD.
- To be able to make a statement about the CD we use the URI "`http://mytunes.com/album_id1234`" as resource identifier for the CD.
- The subject is always a resource.

Predicate

- The predicate defines the kind of information we want to express about the subject.
- In our example we want to make a statement about the title of the CD.
- To define the kind of information we want to state about a the subject we use the URIref "`http://mytunes.com/ontology#hasTitle`" that references the property defined in the ontology.
- How ontologies are defined we will see later.
- The predicate is also called the property that describes the subject resource.

Object

- The object defines the value of the predicate.
- In our example we want to state that the title of the CD is "Alanis Unplugged".
- The object can be a literal, like in our example, or another resource represented by the object's URI.

RDF Graph 1/2

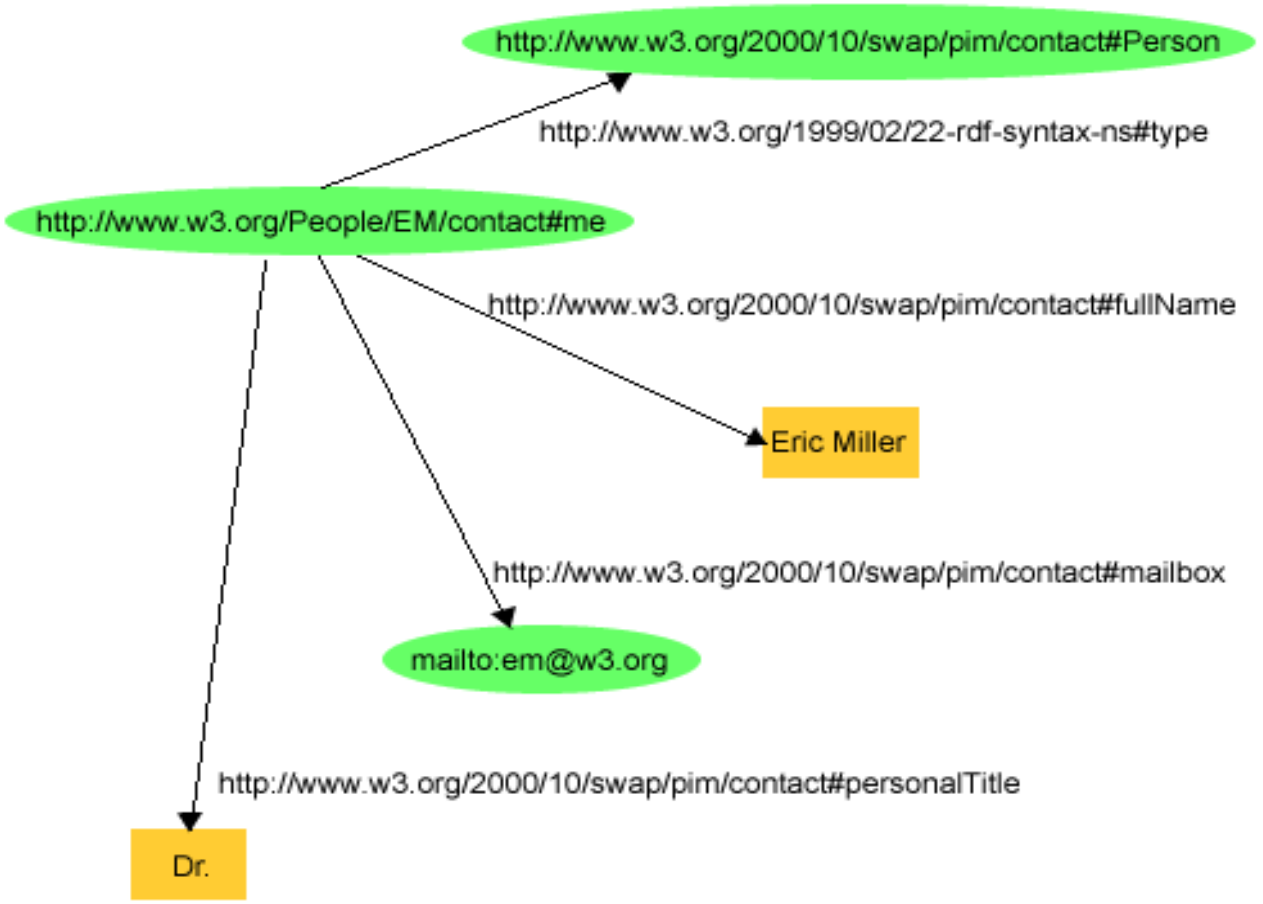
- RDF represents simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values.
- In the RDF data-model the statements are represented as nodes and arcs in a graph.
- In the RDF graph a statement is represented by:
  - a node for the subject,
  - a node for the object, and
  - a labeled arc for the predicate, directed from the subject node to the object node.

RDF Graph 2/2

- Resources are shown as ellipses.
- Literals are shown as boxes.

More Complex Graph Example

There is a `Person` identified by `http://www.w3.org/People/EM/contact#me`, whose name is "Eric Miller", whose email address is `em@w3.org`, and whose title is "Dr.".



RDF Serialization 1/2

- To exchange RDF graphs we have to write the graph down. In other words, we have to serialize the RDF graph.
- Since RDF is the document format on the Web, there is an XML syntax for RDF called RDF/XML.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

RDF Serialization 2/2

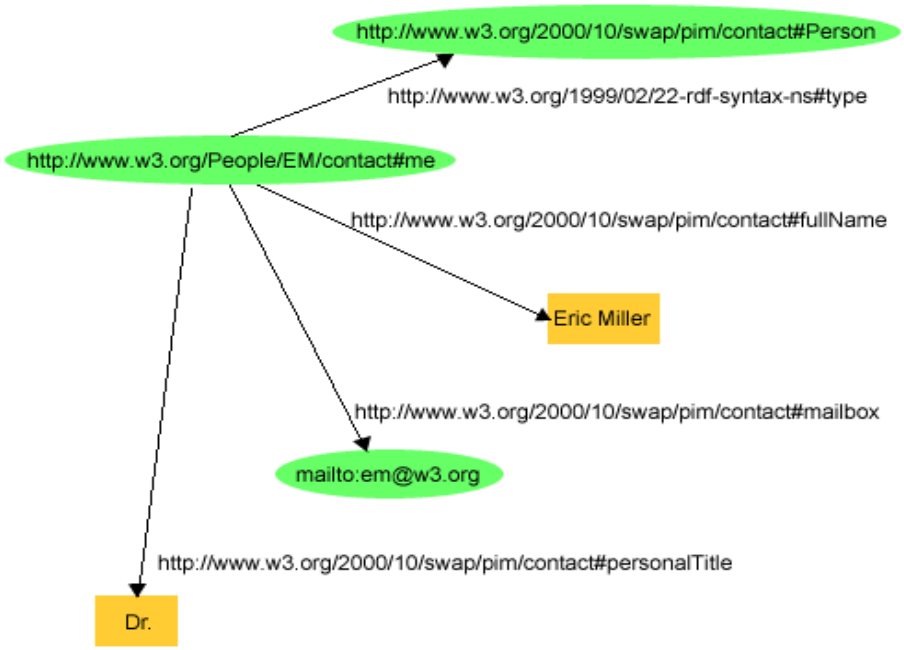
- The RDF/XML serialization represents exactly the same information as the drawn RDF graph.
- The important thing with RDF is the

  graph model!
- The serialization used to represent or depict the graph is secondary.
- Since RDF/XML is hard to read for humans, there exist other serializations for RDF.
  - e.g., N3, N-Triples, Turtle

RDF "N-Triples" Serialization

- N-Triples is a simple text notation for RDF triples.
- Each triple is written in a single line ending with a "."
- URIs are wirtten in angle brackets <>.
- Literals are written in "".
- There exists different names the the same serialization: NTriples, N-Triples

N-Triples Example




```
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/1999/02/22-rdf-syntax-
   ns#type> <http://www.w3.org/2000/10/swap/pim/contact#person> .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/2000/10/swap/pim/
   contact#fullName> "Eric Miller" .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/2000/10/swap/pim/
   contact#mailbox> <mailto:em@w3.org> .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/2000/10/swap/pim/
   contact#personalTitle> "Dr." .
```

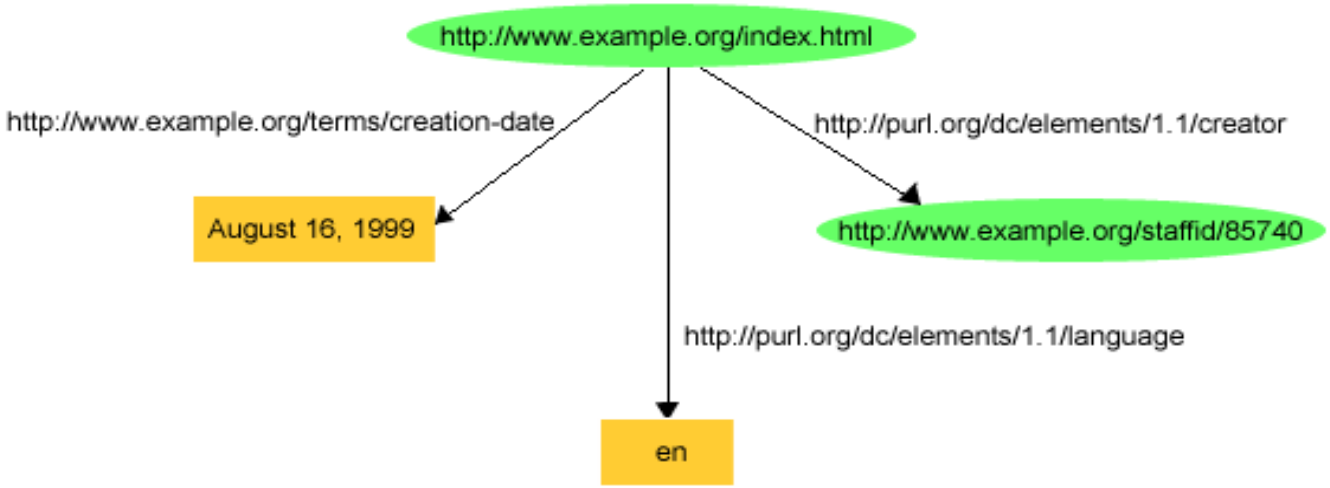
Namespaces for Shorter Serialization

- Writing URIs results in very long lines.
- For convenience we use a shorthand of writing triples.
- The XML qualified name without angle brackets <> is used as abbreviation for the full URI reference.
- A QName contains a prefix that has been assigned to a namespace URI, followed by a colon, and then a local name.
- The full URI is formed from the QName by appending the local name to the namespace URI assigned to the prefix.
  - So, for example, if the QName prefix `foo` is assigned to the namespace URI `http://example.org/somewhere/`, then the QName `foo:bar` is shorthand for the URIref `http://example.org/somewhere/bar`.

Namespaces used in this lecture:

prefix `rdf:`, namespace URI:
`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
prefix `rdfs:`, namespace URI:
`http://www.w3.org/2000/01/rdf-schema#`
prefix `owl:`, namespace URI:
`http://www.w3.org/2002/07/owl#`
prefix `ex:`, namespace URI:
`http://www.example.org/` or `http://www.example.com/`
prefix `ex2:`, namespace URI:
`http://www.domain2example.org/`
prefix `xsd:`, namespace URI:
`http://www.w3.org/2001/XMLSchema#`
prefix `exterms:`, namespace URI:
`http://www.example.org/terms/`
prefix `exstaff:`, namespace URI:
`http://www.example.org/staffid/`
prefix `dc:`, namespace URI:
`http://purl.org/dc/elements/1.1/`

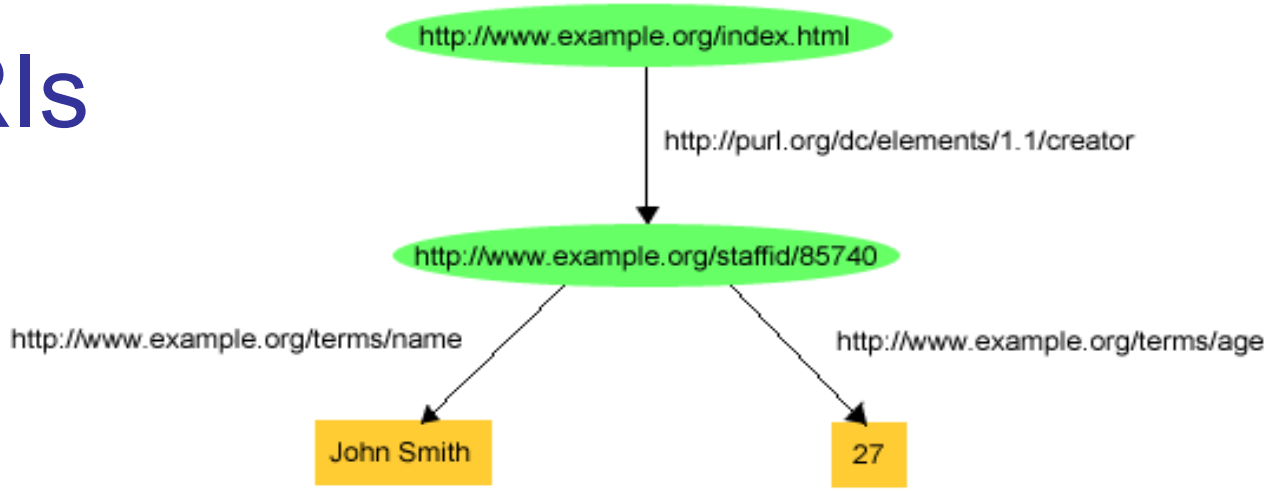
Example for Abbreviation




```
<http://www.example.org/index.html>
    <http://purl.org/dc/elements/1.1/creator>
    <http://www.example.org/staffid/85740> .
<http://www.example.org/index.html>
    <http://www.example.org/terms/creation-date> "August 16, 1999" .
<http://www.example.org/index.html>
    <http://purl.org/dc/elements/1.1/language> "en" .
```

```
ex:index.html  dc:creator             exstaff:85740 .
ex:index.html  exterms:creation-date  "August 16, 1999" .
ex:index.html  dc:language            "en" .
```

The use of URIs



- The example uses a resource as object.
- The creator was a specific, not any John Smith that is uniquely identified by `http://www.example.org/staffid/85740`
- The URIs used for the properties uniquely reference the meaning of the property that is defined in an ontology.
- The ontology defines the vocabulary used in the RDF graph.

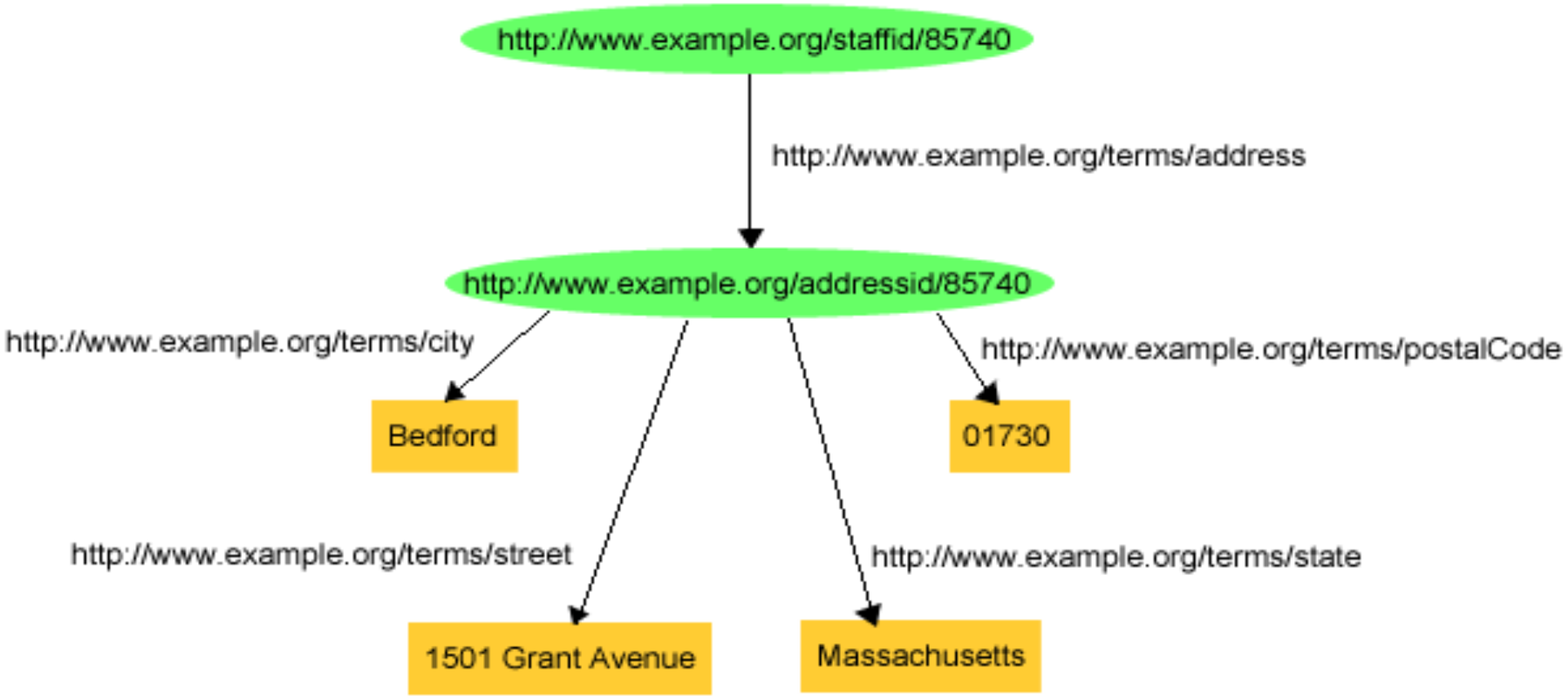
Blank Nodes 1/5

- In praxis we have to express more complex statements in RDF as we have seen before.
- For example:

```
exstaff:85740   exterms:address   "1501 Grant Avenue,
   Bedford, Massachusetts 01730" .
```

- However, suppose John's address needed to be recorded as a structure consisting of separate street, city, state, and postal code values?
- How would this be done in RDF?

Blank Nodes 2/5

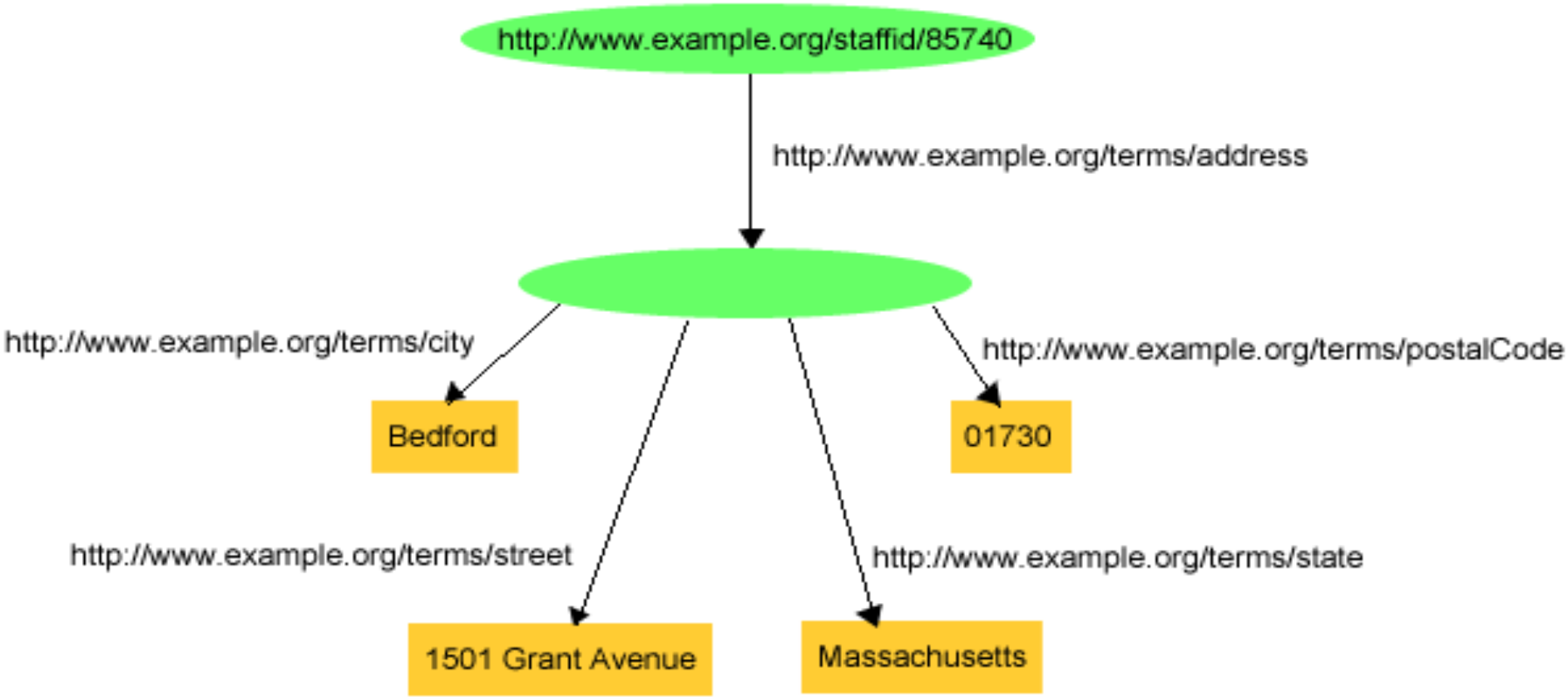


Blank Nodes 3/5

```
exstaff:85740        exterms:address          exaddressid:85740 .
exaddressid:85740    exterms:street           "1501 Grant Avenue" .
exaddressid:85740    exterms:city             "Bedford" .
exaddressid:85740    exterms:state            "Massachusetts" .
exaddressid:85740    exterms:postalCode       "01730" .
```

- This way of representing structured information in RDF can involve generating numerous "intermediate" URIrefs such as `exaddressid:85740` to represent aggregate concepts such as John's address.
- Such concepts may never need to be referred to directly from outside a particular graph, and hence may not require "universal" identifiers.
- The solution to this problem are blank nodes.

Blank Nodes 4/5



Blank Node 5/5

- The graph uses a node without a URIref to stand for the concept of "John Smith's address".
- This blank node serves its purpose in the drawing without needing a URIref, since the node itself provides the necessary connectivity between the various other parts of the graph.
- Blank nodes are used when a node is needed that never needs to be referenced from outside the RDF graph.
- Blank nodes were called anonymous resources.

Blank Node in N-Triples 1/2

```
exstaff:85740   exterms:address          ??? .
???             exterms:street           "1501 Grant Avenue" .
???             exterms:city             "Bedford" .
???             exterms:state            "Massachusetts" .
???             exterms:postalCode       "01730" .
```

- The ??? stands for something that indicates the presence of the blank node.

Blank Node in N-Triples 2/2

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

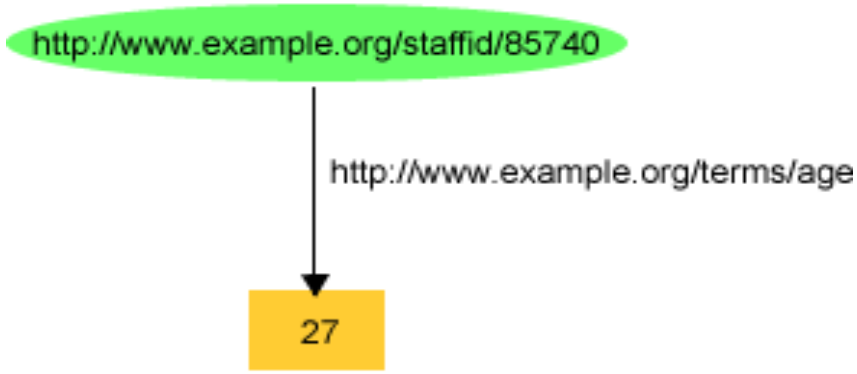
- Since a complex graph might contain more than one blank node, there also needs to be a way to differentiate between these different blank nodes in a triples representation of the graph.
- N-Triples use blank node identifiers, having the form `_:name`, to indicate the presence of blank nodes.
- Unlike URIrefs and literals, blank node identifiers are not considered to be actual parts of the RDF graph.
- Because blank node identifiers represent (blank) nodes, rather than arcs, in the N-Triples form of an RDF graph, blank node identifiers may only appear as subjects or objects in triples; blank node identifiers may not be used as predicates in triples.

Binary Relationship

- RDF directly represents only binary relationships.
  - e.g. the relationship between John Smith and the literal representing his address.
- Representing the relationship between John and the group of separate components of this address involves dealing with an n-ary (n-way) relationship (in this case, n=4) between John and the street, city, state, and postal code components.
- In order to represent such structures directly in RDF, this n-way relationship must be broken up into a group of separate binary relationships.
- Blank nodes provide one way to do this.
  - For each n-ary relationship, one of the participants is chosen as the subject (John in this case), and a blank node is created to represent the rest of the relationship (John's address in this case).
  - The remaining participants in the relationship (such as the city in this example) are then represented as separate properties of the new resource represented by the blank node.

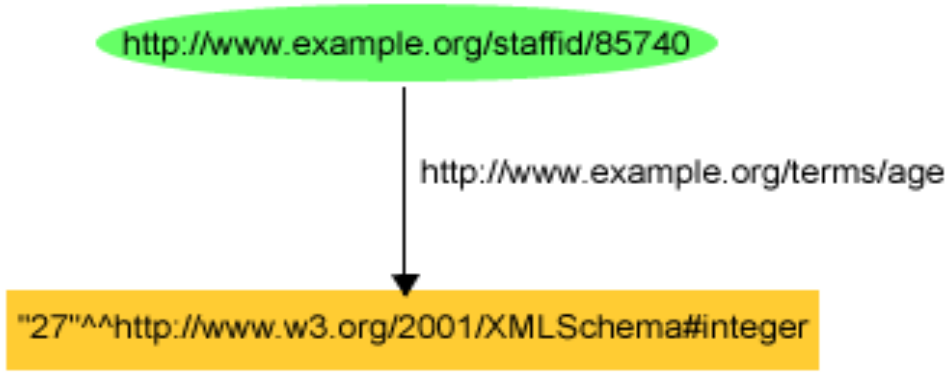
Typed Literals

- Literals in RDF graphs can have data types.




```
<http://www.example.org/staffid/85740>  <http://
www.example.org/terms/age> "27"^^<http://www.w3.org/2001/
XMLSchema#integer> .
```

- RDF uses the XML Schema Part 2: Datatypes
  - http://www.w3.org/TR/xmlschema-2/
- Overview of datatypes
  - http://www.w3.org/TR/xmlschema-0/#CreatDt



An XML Syntax for RDF: RDF/XML

- As mentioned before RDF's conceptual model is a graph.
- RDF provides an XML syntax for writing down and exchanging RDF graphs, called RDF/XML.
- Unlike N-Triples, which is intended as a shorthand notation, RDF/XML is the normative syntax for writing RDF.
- RDF/XML is defined in the RDF/XML Syntax Specification.
  - http://www.w3.org/TR/rdf-syntax-grammar/

Basic Principles

http://www.example.org/index.html has a **creation-date** whose value is **August 16, 1999**

```
ex:index.html   exterms:creation-date   "August 16, 1999" .
```

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.          xmlns:exterms="http://www.example.org/terms/">

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

7. </rdf:RDF>
```

Explanation 1/3

- Line 2 begins an `rdf:RDF` element. This indicates that the following XML content is intended to represent RDF.
- Following the `rdf:RDF` on this same line is an XML namespace declaration, represented as an `xmlns` attribute of the `rdf:RDF` start-tag. This declaration specifies that all tags in this content prefixed with `rdf:` are part of the namespace identified by the URIref `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.
- URIrefs beginning with the string `http://www.w3.org/1999/02/22-rdf-syntax-ns#` are used for terms from the RDF vocabulary.
- Line 3 specifies another XML namespace declaration, this time for the prefix `exterms:` defined as `http://www.example.org/terms/`.

Explanation 2/3

- Lines 1-3 are general "housekeeping" necessary to indicate that this is RDF/XML content, and to identify the namespaces being used within the RDF/XML content.
- Lines 4-6 provide the RDF/XML for the RDF graph. An obvious way to talk about any RDF statement is to say it is a description, and that it is about the subject of the statement (in this case, about `http://www.example.org/index.html`), and this is the way RDF/XML represents the statement.
- The `rdf:Description` start-tag in line 4 indicates the start of a description of a resource, and goes on to identify the resource the statement is `about` (the subject of the statement) using the `rdf:about` attribute to specify the URIref of the subject resource.

Explanation 3/3

- Line 5 provides a property element, with the QName `exterms:creation-date` as its tag, to represent the predicate and object of the statement.
- The content of this property element is the object of the statement, the plain literal `August 19, 1999`. The property element is nested within the containing `rdf:Description` element, indicating that this property applies to the resource specified in the `rdf:about` attribute of the `rdf:Description` element. Line 6 indicates the end of this particular `rdf:Description` element.
- Finally, Line 7 indicates the end of the `rdf:RDF` element started on line 2.
- Using an `rdf:RDF` element to enclose RDF/XML content is optional in situations where the XML can be identified as RDF/XML by context. However, it does not hurt to provide the `rdf:RDF` element in any case.

Two RDF/XML Statements

```
ex:index.html    exterms:creation-date    "August 16, 1999" .
ex:index.html    dc:language              "en" .
```

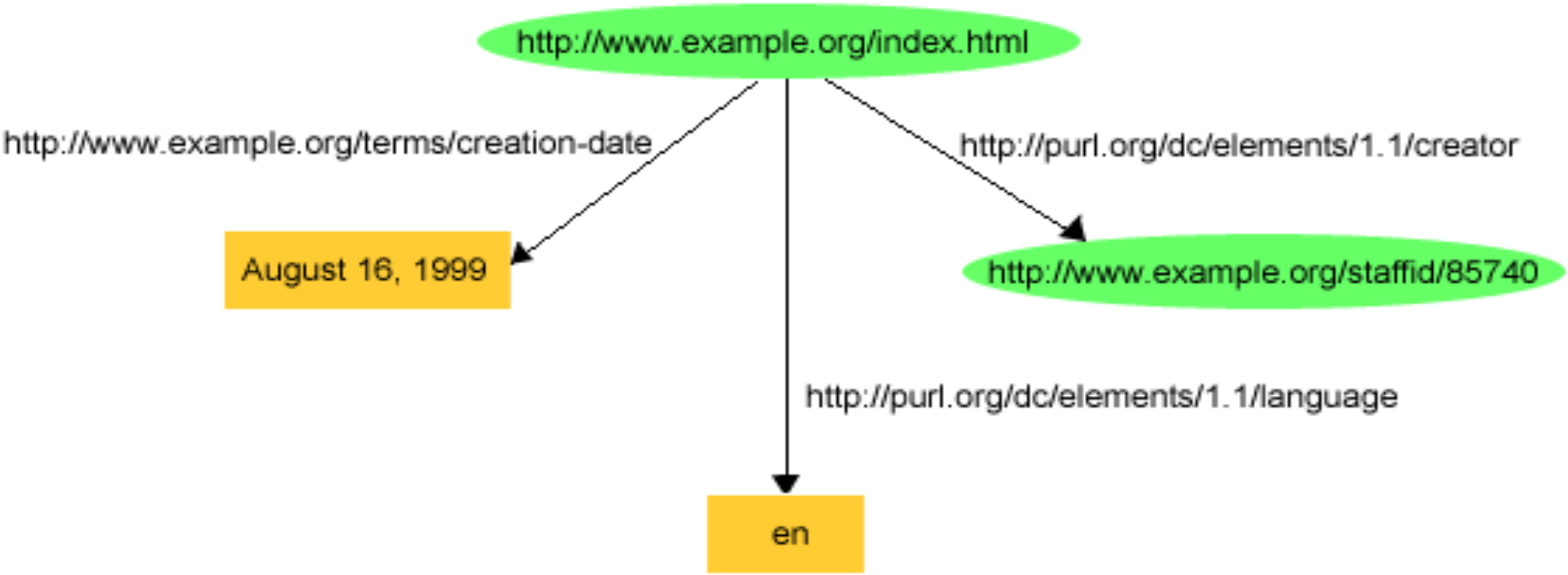
```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:dc="http://purl.org/dc/elements/1.1/"
4.           xmlns:exterms="http://www.example.org/terms/">

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

8.    <rdf:Description rdf:about="http://www.example.org/index.html">
9.        <dc:language>en</dc:language>
10.   </rdf:Description>

11. </rdf:RDF>
```

Several Statements about the Same Resource



Abbreviating Multiple Properties

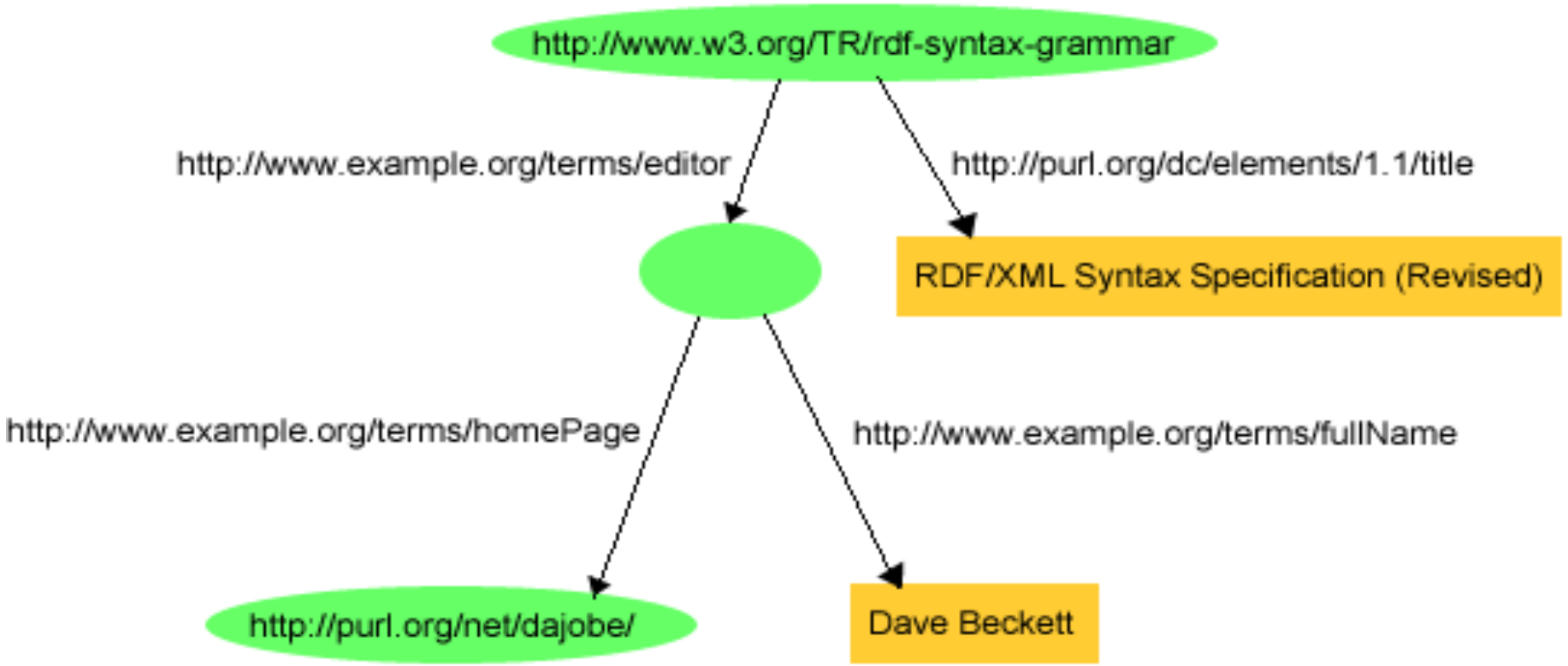
```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:dc="http://purl.org/dc/elements/1.1/"
4.           xmlns:exterms="http://www.example.org/terms/">

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

10. </rdf:RDF>
```

Ressource as object.

Graph Containing a Blank Node



RDF/XML Describing a 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>
11.        <exterms:homePage
                    rdf:resource="http://purl.org/net/dajobe/"/>
12.    </rdf:Description>
13. </rdf:RDF>
```

Alternative Syntax for Blank Nodes

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
  <ex:editor>
    <rdf:Description>
      <exterms:homePage>
        <rdf:Description rdf:about="http://purl.org/net/dajobe/">
        </rdf:Description>
      </exterms:homePage>
      <exterms:homePage rdf:resource="http://purl.org/net/dajobe/"/>
    </exterms:editor>
  <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
</rdf:Description>
```

Alternative Syntax for Blank Nodes

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
  <ex:editor>
    <rdf:Description>
      <exterms:homePage
          rdf:resource="http://purl.org/net/dajobe/"/>
      <exterms:fullName>Dave Beckett</exterms:fullName>
    </rdf:Description>
  </exterms:editor>
  <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
</rdf:Description>
```

Same Graph as on previous slide

Using typed Literals in RDF/XML

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.          xmlns:exterms="http://www.example.org/terms/">

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

7. </rdf:RDF>
```

Using XML entities

```
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#"
4.           xmlns:exterms="http://www.example.org/terms/">

5.   <rdf:Description rdf:about="http://www.example.org/index.html">
6.     <exterms:creation-date rdf:datatype="&xsd;date">1999-08-16
       </exterms:creation-date>
7.   </rdf:Description>

8. </rdf:RDF>
```

Abbreviating and Organizing URIrefs

- So far, the examples have assumed that the resources being described have been given URIrefs already.
- For instance, the initial examples provided descriptive information about example.org's Web page, whose URIref was `http://www.example.org/index.html`.
- This resource was identified in RDF/XML using an `rdf:about` attribute citing its full URIref.
- Although RDF does not specify or control how URIrefs are assigned to resources, sometimes it is desirable to achieve the effect of assigning URIrefs to resources that are part of an organized group of resources.

Outdoor Store Example

- Suppose a sporting goods company, `example.com`, wanted to provide an RDF-based catalog of its products, such as tents, hiking boots, and so on, as an RDF/XML document, identified by (and located at) `http://www.example.com/2002/04/products`.
- In that resource, each product might be given a separate RDF description.
- This catalog, along with one of these descriptions, the catalog entry for a model of tent called the "Overnighter", might be written in RDF/XML as follows...

The `rdf:ID` Attribute 1/3

```
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#"
4.            xmlns:exterms="http://www.example.com/terms/">

5.     <rdf:Description rdf:ID="item10245">
6.          <exterms:model
               rdf:datatype="&xsd;string">Overnighter</exterms:model>
7.          <exterms:sleeps
               rdf:datatype="&xsd;integer">2</exterms:sleeps>
8.          <exterms:weight
               rdf:datatype="&xsd;decimal">2.4</exterms:weight>
9.          <exterms:packedSize
               rdf:datatype="&xsd;integer">784</exterms:packedSize>
10.    </rdf:Description>
  ...other product descriptions...
11.  </rdf:RDF>
```

The `rdf:ID` Attribute 2/3

- An important *difference* from previous examples is that, in line 5, the `rdf:Description` element has an `rdf:ID` attribute instead of an `rdf:about` attribute.
- Using `rdf:ID` specifies a fragment identifier, given by the value of the `rdf:ID` attribute (`item10245`), as an abbreviation of the complete URIref of the resource being described.
- The fragment identifier `item10245` will be interpreted relative to a base URI, in this case, the URI of the containing catalog document.
- The full URIref for the tent is formed by taking the base URI, and appending the character "`#`" (to indicate that what follows is a fragment identifier) and then `item10245` to it, giving the absolute URIref `http://www.example.com/2002/04/products#item10245` if the document is stored at `http://www.example.com/2002/04/products`.

The `rdf:ID` Attribute 3/3

- The `rdf:ID` attribute is somewhat similar to the ID attribute in XML and HTML, in that it defines a name which must be unique relative to the current base URI (in this example, that of the catalog `http://www.example.com/2002/04/products`).
- In this case, the `rdf:ID` attribute appears to be assigning a name (`item10245`) to this particular kind of tent. Any other RDF/XML within this catalog could refer to the tent by using either the absolute URIref `http://www.example.com/2002/04/products#item10245`, or the relative URIref `#item10245`.
- The relative URIref would be understood as being a URIref defined relative to the base URIref of the catalog.

Using the IDs

- A given value of the `rdf:ID` attribute can only appear once relative to the same base URI (the catalog document, in our example).
- RDF located outside the catalog could refer to this tent by using the full URIref, i.e., by concatenating the relative URIref `#item10245` of the tent to the base URI of the catalog, forming the absolute URIref `http://www.example.com/2002/04/products#item10245`.
- For example, an outdoor sports Web site exampleRatings.com might use RDF to provide ratings of various tents.

exampleRatings.com Example

```
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#"
4.          xmlns:sportex="http://www.exampleRatings.com/terms/">

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

exampleRatings.com Example

```
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#"
4.          xmlns:sportex="http://www.exampleRatings.com/terms/">

5. <rdf:Description rdf:about="#item10245">
6.       <sportex:ratingBy rdf:datatype="&xsd;string">
            Richard Roe
         </sportex:ratingBy>
7.       <sportex:numberStars
             rdf:datatype="&xsd;integer">5</sportex:numberStars>
8.     </rdf:Description>
9.  </rdf:RDF>
```

`rdf:ID` and Document Mirrors 1/2

- When a document is stored on several mirrors with different DNS names, the same document can be reached via different URLs.
- As a consequence, the URIrefs for a resource build by the URL of the document and the `rdf:ID` fragment identifier are different on each mirror.
- This is not the indented interpretation, since the same resource should always be referenced with the same identifier.
- If the `xml:base` attribute is specified, not the URL of the document is used to build the URIref, but the value specified by the `xml:base` attribute.

rdf:ID and Document Mirrors 2/2

```
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#"
4.            xmlns:exterms="http://www.example.com/terms/"
5.            xml:base="http://www.example.com/2002/04/products">
6.     <rdf:Description rdf:ID="item10245">
7.          <exterms:model
                rdf:datatype="&xsd;string">Overnighter</exterms:model>
8.          <exterms:sleeps rdf:datatype="&xsd;integer">2</exterms:sleeps>
9.          <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
10.         <exterms:packedSize
                rdf:datatype="&xsd;integer">784</exterms:packedSize>
11.    </rdf:Description>
  ...other product descriptions...
12.  </rdf:RDF>
```

rdf:type to Define Classes

- So far, the examples have used a single product description, a particular model of tent, from `example.com`'s catalog.
- However, `example.com` will probably offer several different models of tents, as well as multiple instances of other categories of products (e.g. backpacks, hiking boots, etc.).
- This idea of things being classified into different kinds or categories is similar to the programming language concept of objects having different types or classes.
- RDF supports this concept by providing a predefined property, `rdf:type`.
- When an RDF resource is described with an `rdf:type` property, the value of that property is considered to be a resource that represents a *category* or *class* of things, and the subject of that property is considered to be an *instance* of that category or class.
- The available classes are defines in ontologies.

Describing a Tent with `rdf:type`

```
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#"
4.               xmlns:exterms="http://www.example.com/terms/"
5.               xml:base="http://www.example.com/2002/04/products">
6.     <rdf:Description rdf:ID="item10245">
7.          <rdf:type
                rdf:resource="http://www.example.com/terms/Tent"/>
8.          <exterms:model
                rdf:datatype="&xsd;string">Overnighter</exterms:model>
9.          <exterms:sleeps
                rdf:datatype="&xsd;integer">2</exterms:sleeps>
10.         <exterms:weight
                rdf:datatype="&xsd;decimal">2.4</exterms:weight>
11.         <exterms:packedSize
                rdf:datatype="&xsd;integer">784</exterms:packedSize>
12.    </rdf:Description>
13.  </rdf:RDF>
```

Abbreviated Syntax for `rdf:type`

```
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#"
4.            xmlns:exterms="http://www.example.com/terms/"
5.            xml:base="http://www.example.com/2002/04/products">

6.     <exterms:Tent rdf:ID="item10245">
7.          <exterms:model
                rdf:datatype="&xsd;string">Overnighter</exterms:model>
8.          <exterms:sleeps
                rdf:datatype="&xsd;integer">2</exterms:sleeps>
9.          <exterms:weight
                rdf:datatype="&xsd;decimal">2.4</exterms:weight>
10.         <exterms:packedSize
                rdf:datatype="&xsd;integer">784</exterms:packedSize>
11.    </exterms:Tent>
12.  </rdf:RDF>
```

RDF Containers

- There is often a need to describe groups of things:
  - For example, to say that a book was created by several authors, or to list the students in a course, or the software modules in a package.
- RDF provides several predefined (built-in) types and properties that can be used to describe such groups.
- RDF provides a container vocabulary consisting of three predefined types.
- A container is a resource that contains things.
- The contained things are called members.
- The members of a container may be resources (including blank nodes) or literals.

RDF Container Types 1/2

- A Bag (`rdf:Bag`) represents a group of resources or literals, possibly including duplicate members, where there is no significance in the order of the members.
  - For example, a Bag might be used to describe a group of part numbers in which the order of entry or processing of the part numbers does not matter.
- A Sequence or Seq (`rdf:Seq`) represents a group of resources or literals, possibly including duplicate members, where the order of the members is significant.
  - For example, a sequence might be used to describe a group that must be maintained in alphabetical order.

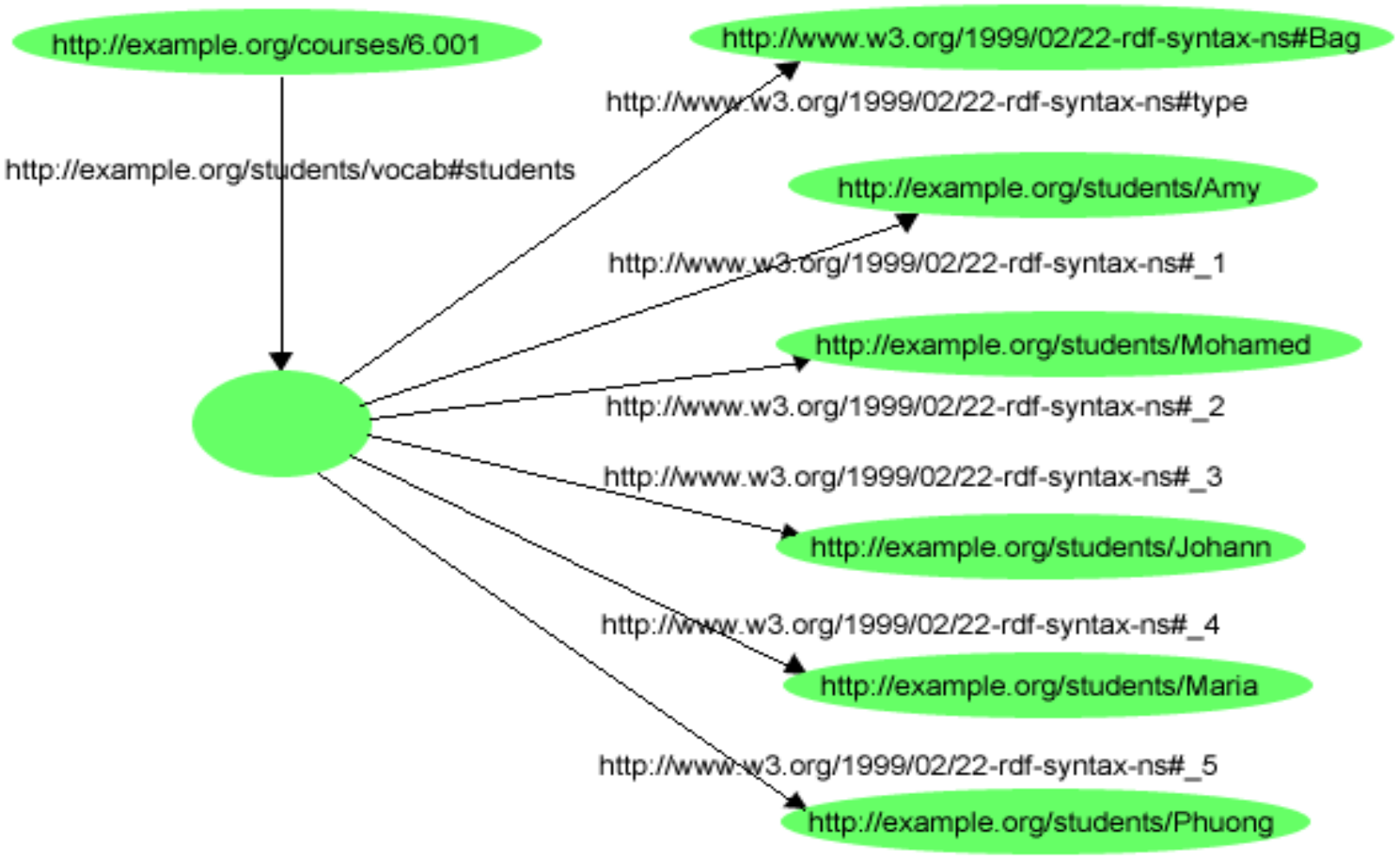
RDF Container Types 2/2

- An Alternative or Alt (`rdf:Alt`) represents a group of resources or literals that are alternatives (typically for a single value of a property).
  - For example, an Alt might be used to describe alternative language translations for the title of a book, or to describe a list of alternative Internet sites at which a resource might be found. An application using a property whose value is an Alt container should be aware that it can choose any one of the members of the group as appropriate.

RDF Container Membership

- To describe a resource as being a container, the resource is given an `rdf:type` property whose value is one of the predefined resources `rdf:Bag`, `rdf:Seq`, or `rdf:Alt`.
- The container resource denotes the group as a whole.
- The members of the container can be described by defining a container membership property for each member with the container resource as its subject and the member as its object.
  - These container membership properties have names of the form `rdf:_n`, where n is a decimal integer greater than zero, with no leading zeros, e.g., `rdf:_1`, `rdf:_2`, `rdf:_3`, and so on, and are used specifically for describing the members of containers.
  - Container resources may also have other properties that describe the container, in addition to the container membership properties and the `rdf:type` property.

RDF Container Example



RDF/XML for an RDF Container 1/2

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

   <rdf:Description rdf:about="http://example.org/courses/6.001">
      <s:students>
         <rdf:Bag>
            <rdf:li rdf:resource="http://example.org/students/Amy"/>
            <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
            <rdf:li rdf:resource="http://example.org/students/Johann"/>
            <rdf:li rdf:resource="http://example.org/students/Maria"/>
            <rdf:li rdf:resource="http://example.org/students/Phuong"/>
         </rdf:Bag>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

RDF/XML for an RDF Container 2/2

- RDF/XML provides `rdf:li` as a convenience element to avoid having to explicitly number each membership property.
- The numbered properties `rdf:_1`, `rdf:_2`, and so on are generated from the `rdf:li` elements in forming the corresponding graph.
- The graph structure for an `rdf:Seq` and `rdf:Alt` container, and the corresponding RDF/XML, are similar to those for an `rdf:Bag` (the only difference is in the type, `rdf:Seq` bzw. `rdf:Alt`).

`rdf:Alt` Default Value

- An Alt container is intended to have at least one member, identified by the property `rdf:_1`.
- This member is intended to be considered as the default or preferred value.
- Other than the member identified as `rdf:_1`, the order of the remaining elements is not significant.

When to use RDF Containers? 1/3

- Sue has written "Anthology of Time", "Zoological Reasoning", and "Gravitational Reflections".
- Without container:

```
exstaff:Sue   exterms:publication   ex:AnthologyOfTime .
exstaff:Sue   exterms:publication   ex:ZoologicalReasoning .
exstaff:Sue   exterms:publication   ex:GravitationalReflections .
```

- Each statement is an individual fact and can also be written with a container:

```
exstaff:Sue   exterms:publication   _:z .
_:z           rdf:type              rdf:Bag .
_:z           rdf:_1                ex:AnthologyOfTime .
_:z           rdf:_2                ex:ZoologicalReasoning .
_:z           rdf:_3                ex:GravitationalReflections .
```

When to use RDF Containers? 2/3

- The resolution was approved by the Rules Committee, having members Fred, Wilma, and Dino.
- The committee as a whole approved the resolution; it does not necessarily state that each committee member individually voted in favor of the resolution. In this case, it would be potentially misleading to model this sentence as three separate `exterms:approvedBy` statements, one for each committee member, as shown below:

```
ex:resolution   exterms:approvedBy   ex:Fred .
ex:resolution   exterms:approvedBy   ex:Wilma .
ex:resolution   exterms:approvedBy   ex:Dino .
```

When to use RDF Containers? 3/3

- In this case, it would be better to model the sentence as a single `exterms:approvedBy` statement whose subject is the resolution and whose object is the committee itself.
- The committee resource could then be described as a Bag whose members are the members of the committee, as in the following triples:

```
ex:resolution       exterms:approvedBy   ex:rulesCommittee .
ex:rulesCommittee   rdf:type             rdf:Bag .
ex:rulesCommittee   rdf:_1               ex:Fred .
ex:rulesCommittee   rdf:_2               ex:Wilma .
ex:rulesCommittee   rdf:_3               ex:Dino .
```

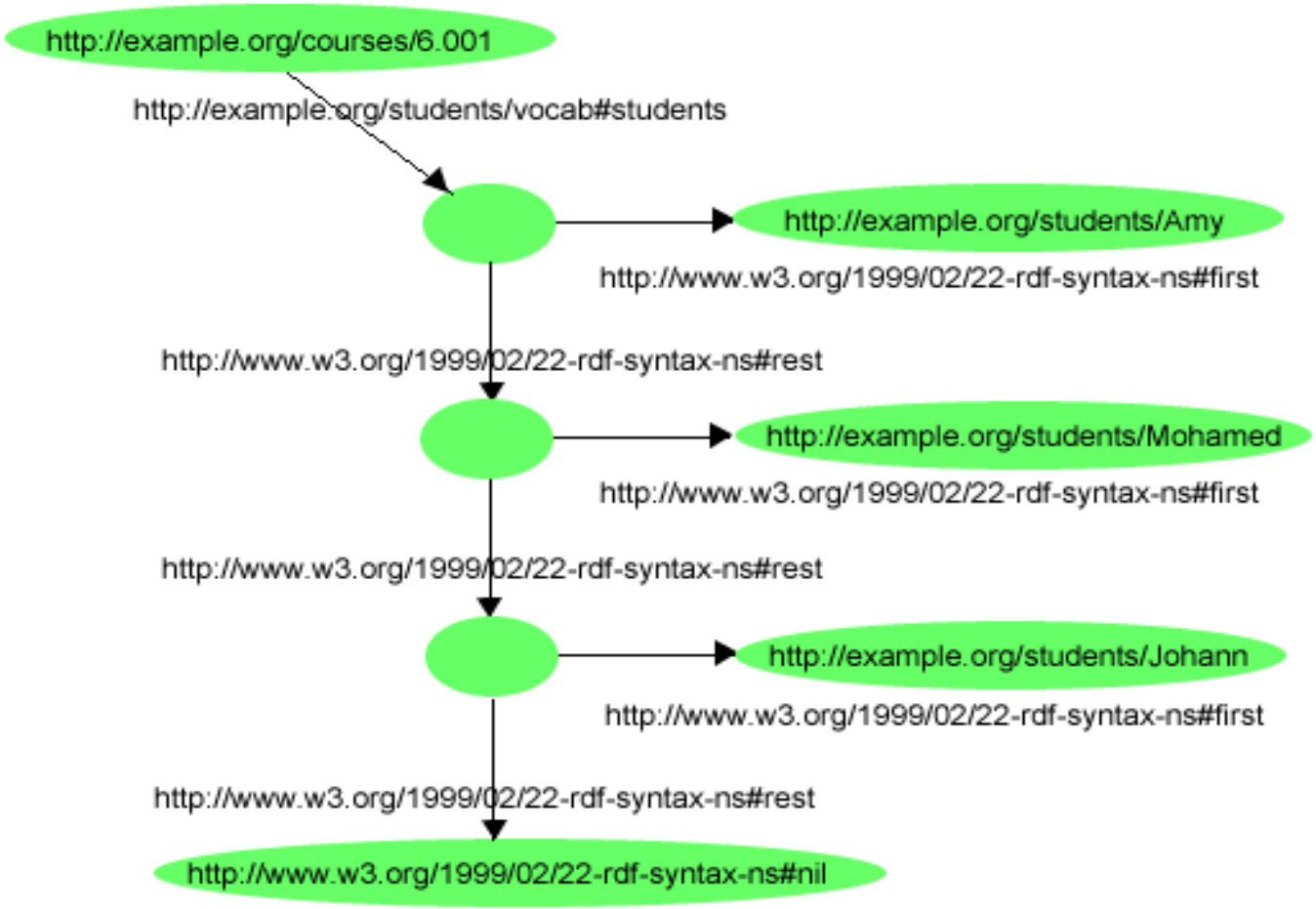
RDF Only *"Intended"* Meaning

- These types of containers are described using predefined RDF types and properties, any special meanings associated with these containers are only intended meanings.
  - These specific container types, and their definitions, are provided with the aim of establishing a shared convention among those who need to describe groups of things.
- All RDF does is provide the types and properties that can be used to construct the RDF graphs to describe each type of container.
- RDF has no more built-in understanding of what a resource of type `rdf:Bag` is than it has of what a resource of type `ex:Tent` is.
- Applications must be written to behave according to the particular meaning involved for each type.

RDF Collections

- A limitation of containers is that there is no way to close them, i.e., to say "these are all the members of the container".
- While one graph may describe some of the members, there is no way to exclude the possibility that there is another graph somewhere that describes additional members.
- Closed groups can be described using RDF collections.
- An RDF collection is a group of things represented as a list structure in the RDF graph.
- The collection vocabulary consists of the predefined type `rdf:List`, the predefined properties `rdf:first` and `rdf:rest`, and the predefined resource `rdf:nil`.
  - The blanke node and the `rdf:nil` node (see next slide) are from the implicitly from the type `rdf:List`.

RDF Collection Example



RDF/XML Syntax for RDF Collections

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

   <rdf:Description rdf:about="http://example.org/courses/6.001">
     <s:students rdf:parseType="Collection">
       <rdf:Description
            rdf:about="http://example.org/students/Amy"/>
       <rdf:Description
            rdf:about="http://example.org/students/Mohamed"/>
       <rdf:Description
            rdf:about="http://example.org/students/Johann"/>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

RDF Reification 1/2

- RDF applications sometimes need to describe other RDF statements using RDF.
  - For instance, to record information about when statements were made or who made them.

```
exproducts:item10245   exterms:weight   "2.4"^^xsd:decimal .
```

- It might be useful to record who provided that particular piece of information.
- RDF provides a built-in vocabulary intended for describing RDF statements. A description of a statement using this vocabulary is called a reification of the statement.
- The RDF reification vocabulary consists of the type `rdf:Statement`, and the properties `rdf:subject`, `rdf:predicate`, and `rdf:object`.

RDF Reification 2/2

- Using the reification vocabulary, a reification of the statement about the tent's weight would be given by assigning the statement a URIref such as `exproducts:triple12345`, and then describing the statement using the statements:

```
exproducts:triple12345   rdf:type         rdf:Statement .
exproducts:triple12345   rdf:subject      exproducts:item10245 .
exproducts:triple12345   rdf:predicate    exterms:weight .
exproducts:triple12345   rdf:object       "2.4"^^xsd:decimal .
exproducts:triple12345   dc:creator       exstaff:85740 .
```

- The (first) four statements are sometimes referred to as a "reification quad" for this reason.

RDF Reification Graph

RDF Reification RDF/XML Syntax 1/2

```
<?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:exterms="http://www.example.com/terms/"
            xml:base="http://www.example.com/2002/04/products">

  <rdf:Description rdf:ID="item10245">
     <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
  </rdf:Description>

  <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>

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

RDF Reification RDF/XML Syntax 2/2

```
<?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:exterms="http://www.example.com/terms/"
            xml:base="http://www.example.com/2002/04/products">

  <rdf:Description rdf:ID="item10245">
     <exterms:weight rdf:ID="triple12345" rdf:datatype="&xsd;decimal">2.4
     </exterms:weight>
  </rdf:Description>

  <rdf:Description rdf:about="#triple12345">
     <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
  </rdf:Description>
</rdf:RDF>
```

RDF Reification Considerations

- Asserting the reification is not the same as asserting the original statement, and neither implies the other.
  - When someone says that John said something about the weight of a tent, they are not making a statement about the weight of a tent themselves, they are making a statement about something John said.
  - When someone describes the weight of a tent, they are not also making a statement about a statement they made.
- Both statements are needed: The original statement and its reification.

RDF Reification Problem

- To reify one RDF statement, four additional statements are needed.
- Leads to an explosion in the number or RDF statements.
- Recently proposed RDF named graphs try to solve this problem.
- With RDF named graphs it is possible to make statements about graphs (instead of single statements).

More structured values: `rdf:value`

- RDF can only express binary relationships.
- More complex relations can be expressed using blank nodes.
- The `rdf:value` to express more complex relationships.

```
exproduct:item10245   exterms:weight   "2.4"^^xsd:decimal .
```

- We want to express that the weight is given in kilograms.

```
exproduct:item10245   exterms:weight   _:weight10245 .
_:weight10245         rdf:value        "2.4"^^xsd:decimal .
_:weight10245         exterms:units    exunits:kilograms .
```

RDF/XML rdf:value

```
<?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.com/2002/04/products#item10245">
     <exterms:weight rdf:parseType="Resource">
       <rdf:value rdf:datatype="&xsd;decimal">2.4</rdf:value>
       <exterms:units
        rdf:resource="http://www.example.org/units/kilograms"/>
     </exterms:weight>
  </rdf:Description>

</rdf:RDF>
```

XML Literals

- If the value of a property is a fragment of XML:

```
<?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/books">

  <rdf:Description rdf:ID="book12345">
     <dc:title rdf:parseType="Literal">
       <span xml:lang="en">
         The <em>&lt;br /&gt;</em> Element Considered Harmful.
       </span>
     </dc:title>
  </rdf:Description>

</rdf:RDF>
```

Visualizing and Editing RDF: IsaViz

- IsaViz tool to visualize and edit RDF.
  - http://www.w3.org/2001/11/IsaViz/
- Features:
  - RDF/XML, Notation 3 and N-Triple import
  - RDF/XML, Notation 3 and N-Triple export, but also SVG and PNG export
- Requires graphviz/dot
  - http://www.research.att.com/sw/tools/graphviz/

RDF GRAVITY

- RDF Graph Visualization Tool
  - Graph Visualization (does not use the W3C recommended visualization)
  - Global and Local Filters (enabling specific views on a graph)
  - Full text Search
  - Generating views from RDQL Queries
  - http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html

RDF Validator

- RDF/XML validation service from the W3C
  - http://www.w3.org/RDF/Validator/
- Output:
  - Explicit triples of the graph.
  - Graphical representation of the graph.