An Introduction to RDF

Acknowledgement

- This presentation is based on the excellent RDF primer by the W3C available at http://www.w3.org/TR/rdf-primer/ and http://www.w3.org/2007/02/turtle/primer/ .
- Much of the material in this presentation is verbatim from the above Web site.

Presentation Outline

- **Basic concepts of RDF**
- Serialization of RDF graphs: XML/RDF and Turtle
- Other Features of RDF (Containers, Collections and Reification).

What is RDF?

- The **Resource Description Framework (RDF)** is a data model for representing information (especially **metadata**) about **resources** in the Web.

- 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., a book or a person).

- RDF is intended for situations in which information about Web resources needs to be **processed by applications**, rather than being only displayed to people.

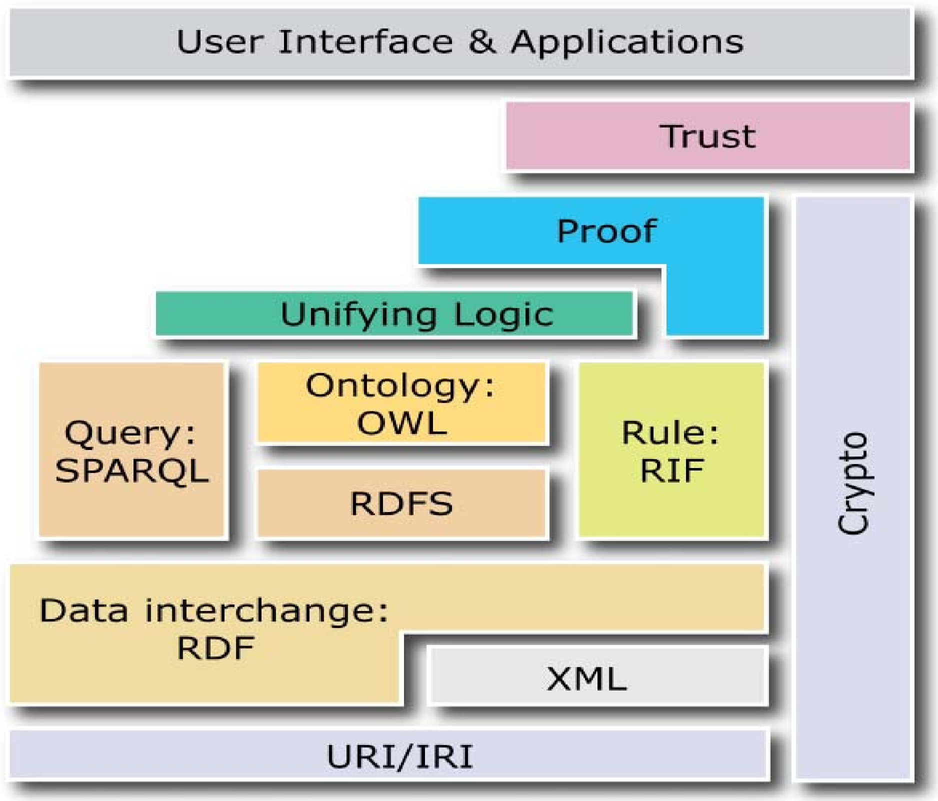
Some History

- RDF draws upon ideas from knowledge representation, artificial intelligence, and data management, including:
  - Semantic networks
  - Frames
  - Conceptual graphs
  - Logic-based knowledge representation
  - Relational databases

- **Shameless self-promotion ☺ :** The closest to RDF, pre-Web knowledge representation language is Telos:

  John Mylopoulos, Alexander Borgida, Matthias Jarke, Manolis Koubarakis: Telos: Representing Knowledge About Information Systems. ACM Trans. Inf. Syst. 8(4): 325-362 (1990).

The Semantic Web "Layer Cake"



RDF Basics

- RDF is based on the idea of identifying resources using **Web identifiers** and describing resources in terms of simple **properties** and property **values**.

- To identify resources, RDF uses **Uniform Resource Identifiers (URIs**) and **URI references (URIrefs).**

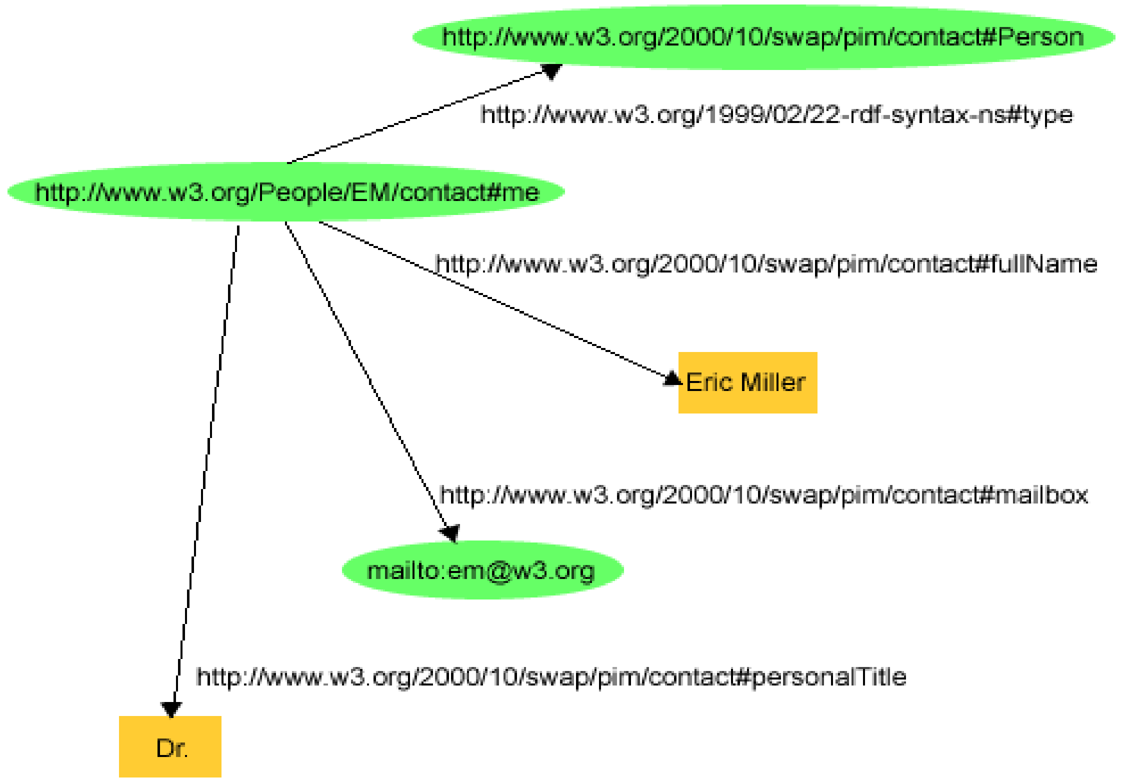
- **Definition:** A **resource** is anything that is identifiable by a URIref.

Example

- Consider the following information:

“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.”

Example (cont'd)


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

Basics (cont'd)

- Forget the long URIs for the moment!
- RDF is based on the idea that **the resources being described have properties which have values**, and that resources can be described by making **statements**, similar to the ones above, that specify those properties and values.
- **Terminology:**
  - The part that identifies the thing the statement is about is called the **subject**.
  - The part that identifies the property or characteristic of the subject that the statement specifies is called the **predicate**.
  - The part that identifies the value of that property is called the **object**.

Example

`http://www.example.org/index.html` has a creator whose value is “John Smith”

- The **subject** is the URL `http://www.example.org/index.html`
- The **predicate** is the word "creator"
- The **object** is the phrase “John Smith”

RDF Triples

- RDF statements can be written down using **triple notation.** In this notation, a statement is written as follows:

  subject predicate object .

- **Example:**

```
<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" .
```

- **Note:** In this notation URIs are written out completely, in angle brackets.

Uniform Resource Identifiers

- The Web provides a general form of identifier, called the **Uniform Resource Identifier (URI),** for identifying (naming) resources on the Web.

- Unlike URLs, URIs are not limited to identifying things that have network locations, or use other computer access mechanisms. A number of different **URI schemes** (**URI forms**) have been already been developed, and are being used, for various purposes.

- **Examples:**
  - `http:` (Hypertext Transfer Protocol, for Web pages)
  - `mailto:` (email addresses), e.g., `mailto:em@w3.org`
  - `ftp:` (File Transfer Protocol)
  - `urn:` (Uniform Resource Names, intended to be persistent location-independent resource identifiers), e.g., `urn:isbn:0-520-02356-0` (for a book)

- No one person or organization controls who makes URIs or how they can be used. While some URI schemes, such as URL's `http:,` depend on centralized systems such as DNS, other schemes, such as `freenet:,` are **completely decentralized.**

URIs (cont'd)

- A **URI reference** (or **URIref**) is a URI, together with an optional fragment identifier at the end.

  Example: the URIref `http://www.example.org/index.html#section2` consists of the URI `http://www.example.org/index.html` and (separated by the "#" character) the fragment identifier `section2`.

- URIrefs may be either **absolute** or **relative**.

- An **absolute** URIref refers to a resource independently of the context in which the URIref appears, e.g., the URIref `http://www.example.org/index.html`.

- A **relative** URIref is a shorthand form of an absolute URIref, where some prefix of the URIref is missing, and information from the context in which the URIref appears is required to fill in the missing information.

  Example: the relative URIref `otherpage.html`, when appearing in a resource `http://www.example.org/index.html`, would be filled out to the absolute URIref `http://www.example.org/otherpage.html`.

URIs (cont'd)

- A (relative) **URIref consisting of just a fragment identifier** is considered equivalent to the URIref of the document in which it appears, with the fragment identifier appended to it.

  Example: Within the document `http://www.example.org/index.html`, if `#section2` appeared as a URIref, it would be considered equivalent to the absolute URIref `http://www.example.org/index.html#section2` .

- A **URIref without a URI part** is considered a reference to the current document (the document in which it appears). So, an **empty URIref** within a document is considered equivalent to the URIref of the document itself.

- See the report from the Joint W3C/IETF URI Planning Interest Group at http://tools.ietf.org/html/rfc3305 for more information.

URIrefs in RDF

- RDF uses URIrefs to identify and name the subjects, predicates, and objects in statements.

- RDF URIrefs can contain **Unicode** characters, allowing many languages to be reflected in URIrefs. So, more precisely, RDF uses **Internationalized Resource Identifiers (IRIs) and IRIrefs.**

- See http://tools.ietf.org/html/rfc3987 which defines IRIs.

URIrefs in RDF (cont'd)

- Both RDF and Web browsers use URIrefs to **identify things**. However, RDF and browsers interpret URIrefs in slightly different ways:
  - RDF uses URIrefs **only** to identify things.
  - Browsers also use URIrefs to **retrieve** things.

- What is the difference?
  - When a URIref is used in a **browser**, there is the expectation that it identifies a resource that can actually be **retrieved**: that something is actually "at" the location identified by the URI.
  - In **RDF**, a URIref may be used to identify something, such as a person, that **cannot** be retrieved on the Web.

- But important uses of RDF, like **Linked Data** (http://linkeddata.org/), insist that we use HTTP URIs so data identified by a URI can be retrieved.

URIrefs in RDF (cont'd)

- Another difference is in the way URIrefs with **fragment identifiers** are handled. Consider the following URIrefs:

```
http://www.example.org/index.html
http://www.example.org/index.html#Section2
```

- In normal HTML usage, these **URIrefs are related** (they both refer to the same document, the second one identifying a location within the first one).

- RDF assumes **no particular relationship** between these two URIrefs. As far as RDF is concerned, they are syntactically different URI references, and hence may refer to unrelated things.

RDF Graphs

- Graphically, RDF models statements by **nodes** and **arcs** in a **graph**.

- In the RDF graph notation, a **statement** is represented by:
  - a node for the subject
  - a node for the object
  - an arc for the predicate, directed from the subject node to the object node.

- A **node** may be identified by a **URIref** or it can be a **literal** (it can also be a **blank node**; we will explain this later).

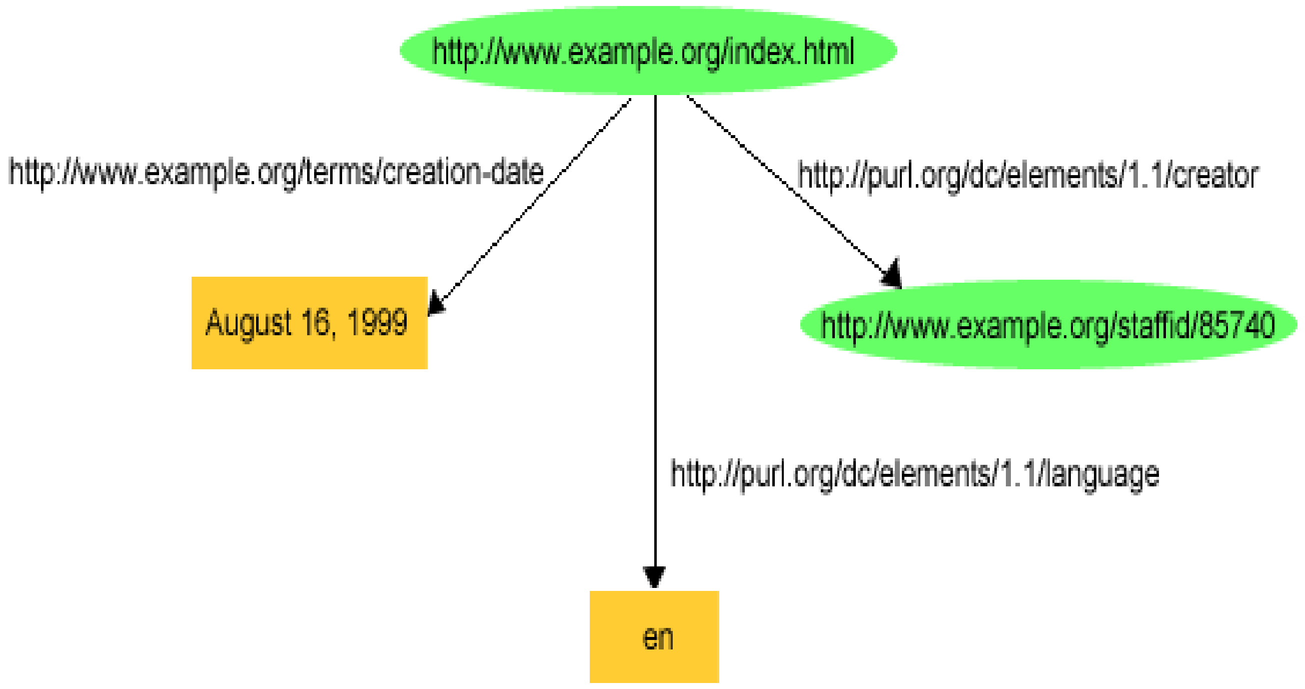
- An **arc** is identified by a **URIref**.

- **Note:** We will draw RDF graphs as **directed graphs**. But strictly speaking, directed graphs are not sufficient for capturing all of RDF (e.g., directed graphs assume that the sets of nodes and arcs are disjoint but RDF allows a property as a subject of a statement).

Example

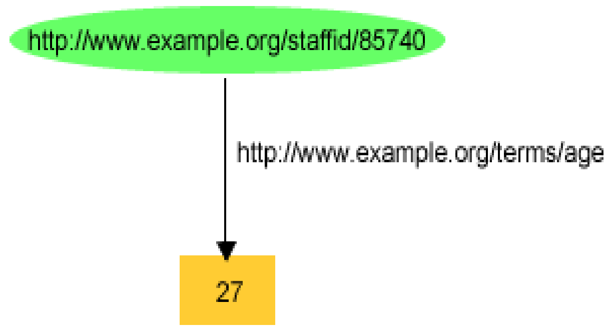
- Consider the following statements:
  - `http://www.example.org/index.html` has a creation-date whose value is August 16, 1999.
  - `http://www.example.org/index.html` has a language whose value is English.

The RDF Graph of the Example


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

Literals

- There are two kinds of literals:
  - **Plain (untyped).**
  - **Typed.**
- Example:


http://www.example.org/staffid/85740
http://www.example.org/terms/age
27

- What is 27? Number or string?

Plain Literals

- Plain literals have a **lexical form** (their lexical value) and optionally a **language tag**.

- **Example:** `“27”` , `“Hello world”@en`

Typed Literals

- An **RDF typed literal** is formed by pairing a string with a URIref that identifies a particular **datatype**.

- **Example:**

```
“27”^^http://www.w3.org/2001/XMLSchema#integer
```

Typed Literals (cont'd)

- RDF has **no built-in set of datatypes** of its own. RDF typed literals simply provide a way to explicitly indicate, for a given literal, what datatype should be used to interpret it. The datatypes used in typed literals are defined **externally to RDF, and identified by their datatype URIs**.

- **Exception:** RDF defines a built-in datatype with the URIref `rdf:XMLLiteral` to represent XML content as a literal value.

- RDF datatype concepts are based on a conceptual framework from **XML Schema datatypes.** This framework defines the **value space**, the **lexical space** and the **lexical-to-value** mapping for a datatype (see the RDF specifications for more details).

Triple Notation – QNames as Shorthands

- The full triple notation results in very long lines.
- **Shorthand:** We can use an **XML qualified name** (or **QName**) without angle brackets as an abbreviation for a full URI reference.
- A **QName** consists of a **prefix** that has been assigned to a **namespace URI**, followed by a **colon**, and then a **local name**. The full URIref is formed from the QName by appending the local name to the namespace URI assigned to the prefix.
- The concepts of **names** and **namespaces** used in RDF originate in XML.

XML Namespaces

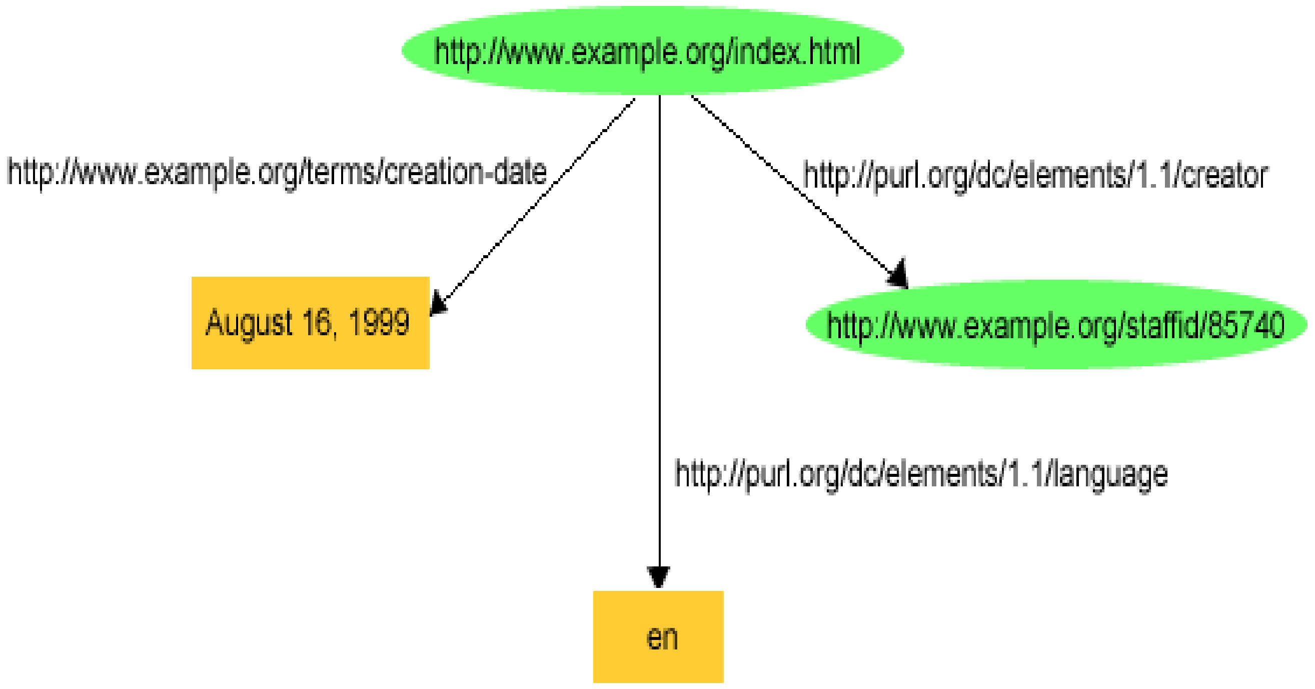
- A **namespace** is a way of identifying a subset of a set of names (e.g., the set of possible names of resources in the Web) which acts as a qualifier for the names in this subset.
- **XML namespaces** are used for providing uniquely named elements and attributes in an XML document.
- **XML namespaces help us eliminate ambiguity in an XML document.** For example, an XML document can use `id` to refer to both identifiers of customers and products if `id` is prefixed by an appropriate name space (e.g., http://customers.org and http://products.com).
- A **namespace** is created by creating a URI for it. By qualifying names with the URIs of their namespaces, **anyone can create their own names and properly distinguish them from names with identical spellings created by others**.
- See the **W3C Recommendation "Namespaces in XML 1.0"** available at http://www.w3.org/TR/REC-xml-names/.

QNames as Shorthands (cont'd)

- **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`.

- **More Examples:**
  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 `dc:`, namespace URI: `http://purl.org/dc/elements/1.1/`
  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 `xsd:`, namespace URI: `http://www.w3.org/2001/XMLSchema#`

Example


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

Example (cont'd)

- The previous graph can be written in triples notation using the shorthands we introduced earlier as follows:

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

- We will later see **Turtle**, a textual syntax for RDF graphs that makes use of this shorthand notation based on QNames.

URIrefs as Vocabulary

- Since RDF uses URIrefs instead of words to name things in statements, URIrefs define **vocabularies** in RDF.

- The URIrefs in RDF vocabularies are typically organized so that they can be represented as a **set of QNames with a common prefix:**
  - A common namespace URIref is chosen for all terms in a vocabulary, typically a URIref under the control of whoever is defining the vocabulary.
  - URIrefs that are contained in the vocabulary are formed by appending individual local names to the end of the common URIref.

Example

- DBpedia (http://wiki.dbpedia.org/About) is a large data set which has been derived from Wikipedia by extracting various kinds of structured information from Wikipedia editions in 14 languages and combining this information into a huge, cross-domain knowledge base.

- In the DBpedia data set, each thing is identified by a URIref of the form `http://dbpedia.org/resource/Name`, where `Name` is taken from the URL of the source Wikipedia article, which has the form `http://en.wikipedia.org/wiki/Name`. Thus, each resource is tied directly to an English-language Wikipedia article.

DBpedia (cont'd)

- The URIref `http://dbpedia.org/resource/Greece` is the DBpedia resource about Greece.
- The prefix `dbpedia` can be used instead of `http://dbpedia.org/resource/`
- For example: `dbpedia:Greece`

URIrefs as Vocabulary (cont'd)

- RDF uses this same approach to define its own vocabulary of terms with special meanings in RDF:
  - The URIrefs in the RDF vocabulary all begin with `http://www.w3.org/1999/02/22-rdf-syntax-ns#`, conventionally associated with the QName prefix `rdf:`.
  - The RDF Vocabulary Description Language defines an additional set of terms having URIrefs that begin with `http://www.w3.org/2000/01/rdf-schema#`, conventionally associated with the QName prefix `rdfs:`.

- Where a specific QName prefix is commonly used in connection with a given set of terms in this way, the QName prefix itself is sometimes used as the **name of the vocabulary**. For example, someone might refer to **"the `rdfs:` vocabulary".**

URIrefs as Vocabulary (cont’d)

- **Convention:** Organizations typically use a vocabulary's namespace URIref as the URL of a Web resource that provides further information about that vocabulary.

- **Example:** the QName prefix `dc:` with the namespace URIref `http://purl.org/dc/elements/1.1`refers to the **Dublin Core vocabulary**.
  - Accessing this namespace URIref in a Web browser will retrieve additional information about the Dublin Core vocabulary (specifically, **an RDF schema**).
  - **Reminder:** this is just a useful convention. RDF does not assume that a namespace URI identifies a retrievable Web resource.

URIrefs as Vocabulary (cont'd)

- Using URIrefs as subjects, predicates, and objects in RDF statements supports the **development and use of shared vocabularies** on the Web.

- People can **discover and begin using vocabularies** already used by others to describe things, reflecting a **shared understanding of those concepts.**

Example

- Consider the triple `ex:index.html dc:creator exstaff:85740 .`

  The predicate `dc:creator`, when fully expanded as a URIref, is an unambiguous reference to the "creator" attribute in the Dublin Core metadata attribute set, a widely-used set of attributes (properties) for describing a wide range of networked resources (see http://dublincore.org/documents/usageguide/).

  The writer of this triple is effectively saying that the relationship between the Web page and the creator of the page is exactly the concept identified by `http://purl.org/dc/elements/1.1/creator`.

  Another person familiar with the Dublin Core vocabulary, or who finds out what `dc:creator` means (say by looking up its definition on the Web) will know what is meant by this relationship. In addition, based on this understanding, people can **write programs to behave in accordance with that meaning** when processing triples containing the predicate `dc:creator`.

Another Example

- The Friend of a Friend (FOAF) vocabulary at http://xmlns.com/foaf/spec/.

- The FOAF project is creating a Web of machine-readable pages (written in RDF) describing people, the links between them and the things they create and do.

URIrefs as Vocabulary (cont'd)

- RDF gives meaning to the terms defined in the relevant RDF vocabularies `rdf:` and `rdfs:`.

- Others have defined the meaning of terms in other important vocabularies e.g., `dc:`

RDF and Related Data Models

- In terms of the **relational model**, a statement is similar to a **tuple in a relation** called *Triples* or *Graph* with attributes *Subject, Predicate* and *Object*.

- In terms of **first-order logic**, a statement is similar to an **atomic formula**

  triple(subj,pred,obj)

  where *triple* is a first-order logic predicate and *subj, pred* and *obj* are constants.

Structured Values in RDF

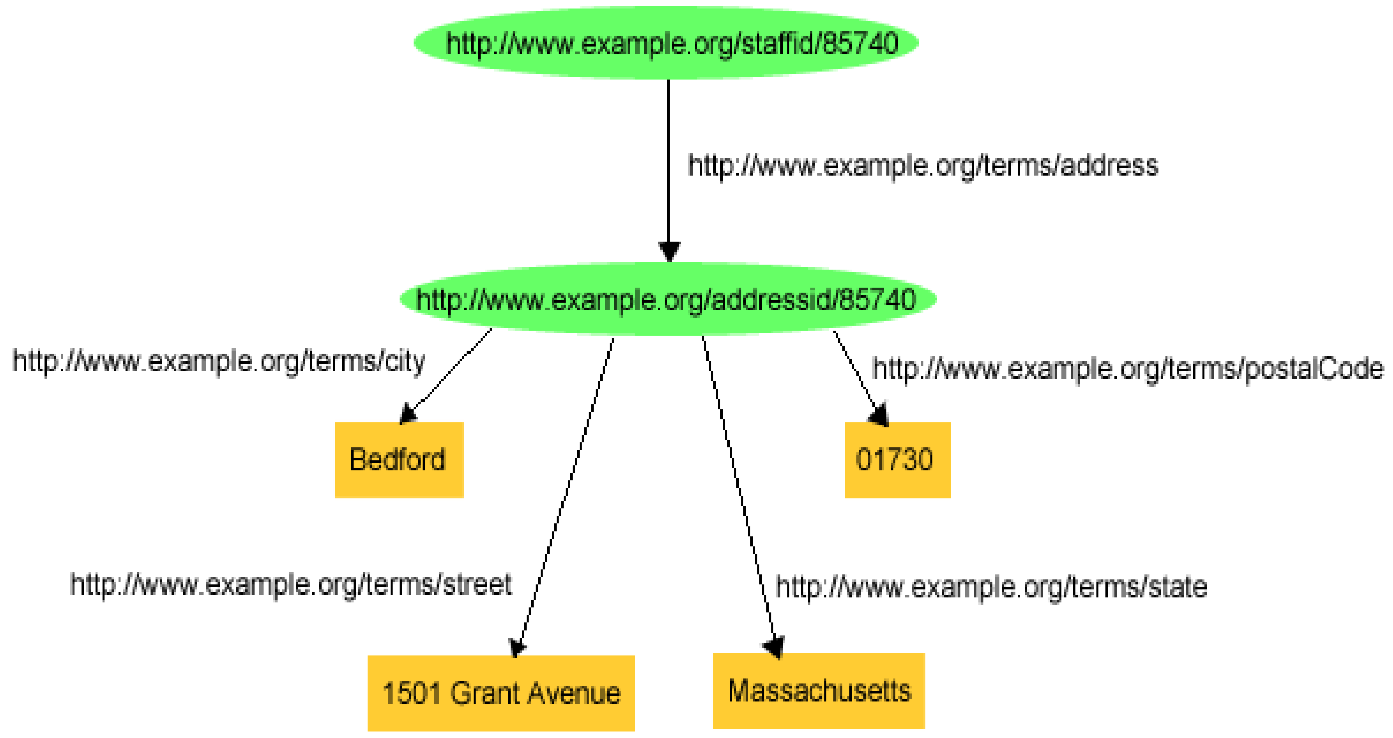
- Consider the triple:

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

- What if the address needs to be represented as a **structure** consisting of separate street, city, state, and postal code values? How would this be done in RDF?

- Structured information is represented in RDF by considering **the aggregate thing to be described** (like John Smith's address) **as a resource**, and then making statements about that new resource.

Structured Values in RDF (cont'd)


http://www.example.org/staffid/85740
http://www.example.org/terms/address
http://www.example.org/addressid/85740
http://www.example.org/terms/city
http://www.example.org/terms/postalCode
Bedford
01730
http://www.example.org/terms/street
http://www.example.org/terms/state
1501 Grant Avenue
Massachusetts

Structured Values in RDF (cont'd)

- Or in triples notation:

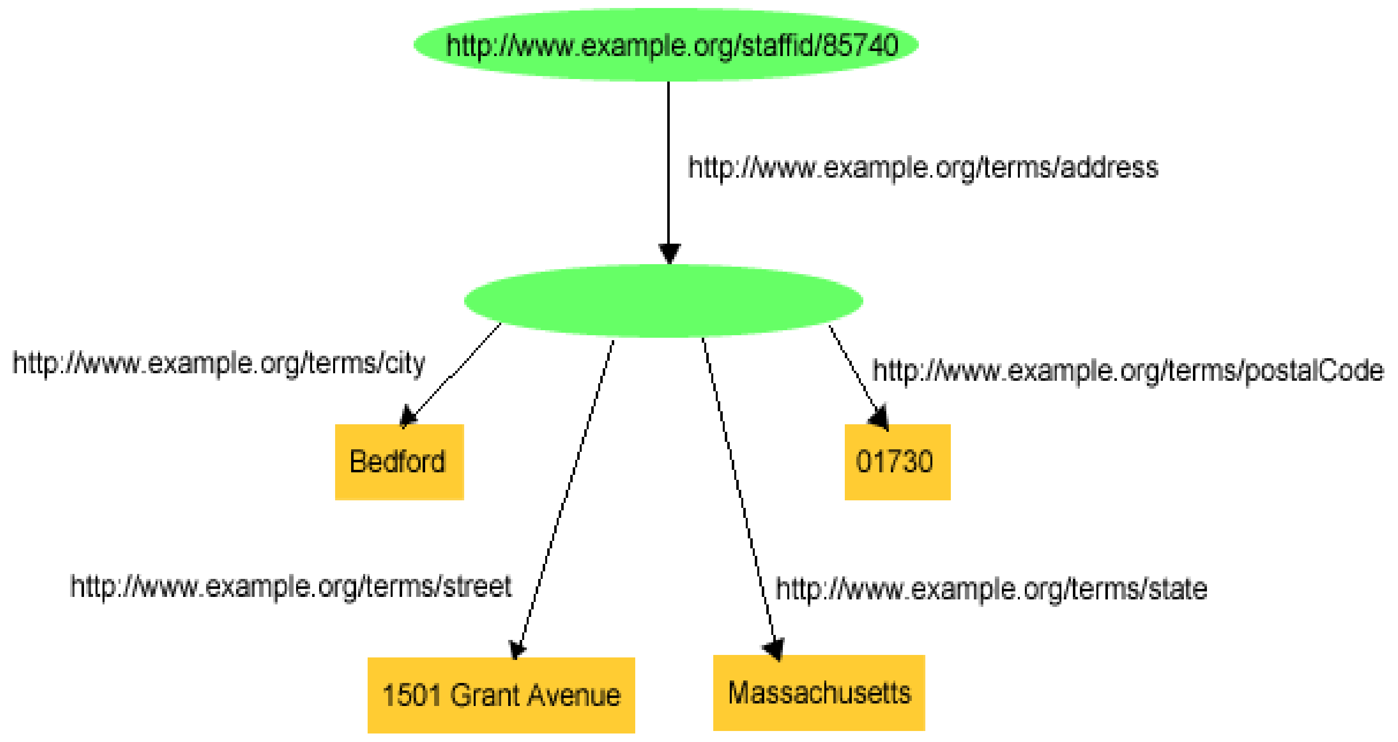
```
exstaff:85740 exterms:address exaddressid:85740 .
exaddressid:85740 exterms:street "1501 Grant Avenue" .
exaddressid:85740 exterms:city "Bedford" .
exaddressid:85740 exterms:state "Massachusetts" .
exaddressid:85740 exterms:postalCode "01730" .
```

Structured Values in RDF (cont’d)

- 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.**

- RDF allows us to use **blank nodes** and **blank node identifiers** to deal with this issue.

Blank Nodes


http://www.example.org/staffid/85740
http://www.example.org/terms/address
http://www.example.org/terms/city
Bedford
http://www.example.org/terms/postalCode
01730
http://www.example.org/terms/street
1501 Grant Avenue
http://www.example.org/terms/state
Massachusetts

Blank Nodes Using Triples

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

- But what about **different blank nodes**? Can we handle them using this notation?

Blank Node Identifiers

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

- In a triples representation of a graph, each distinct blank node must be given a **different** blank node identifier.
- Blank node identifiers have significance only within the triples representing a **single** graph.
- Blank node identifiers may only appear as subjects or objects in triples; blank node identifiers **may not be used as predicates** in triples.

RDF and Related Data Models (cont'd)

- In terms of the **relational model**, a blank node corresponds to a **marked null value**. Thus, a statement with a blank node identifier is similar to a **tuple in a relation** in the marked nulls model of Imielinski and Lipski:

  Tomasz Imielinski, Witold Lipski Jr.: Incomplete Information in Relational Databases. J. ACM 31(4): 761-791 (1984).

- In terms of **first-order logic**, a blank node corresponds to an existentially quantified variable (or a Skolem constant). Thus, a graph with blank nodes is similar to an **existentially quantified first order logic statement** or **a first-order database** in the sense of Reiter:

  Raymond Reiter: Towards a Logical Reconstruction of Relational Database Theory. Appears in the collection: Brodie M. L., Mylopoulos J., and Schmidt J. W. (eds.), On Conceptual Modelling: Perspectives from Artificial Intelligence, Database and Programming Languages, pp. 191-233, Springer-Verlag.

Blank Nodes (cont'd)

- Blank nodes are useful to represent **n-ary relationships** in RDF (e.g., the relationship between John Smith and the street, city, state, and postal code components of his address.
- Blank nodes are also useful to more accurately make statements about **resources that may not have URIs,** but that are described in terms of relationships with other resources that do have URIs.

Example

- When making statements about a person, say Jane Smith, is it natural to use a URI based on that person's email address as her URI, e.g., `mailto:jane@example.org` ?

- Well, if we do so, how are we going to record information both about **Jane's mailbox** (e.g., the server it is on) as well as about **Jane herself** (e.g., her current physical address)? Similarly, if we use her Web page URI etc.

Example (cont'd)

- **Blank nodes to the rescue:** When Jane herself does not have a URI, a blank node provides a more accurate way of modeling this situation.

```
_:jane exterms:mailbox <mailto:jane@example.org> .
 _:jane rdf:type exterms:Person .
_:jane exterms:name "Jane Smith" .
_:jane exterms:empID "23748" .
_:jane exterms:age "26" .
```

The Property `rdf:value`

- When we have a structured value, RDF provides a way to define the **“main value”** of this structured value, with the other parts providing additional contextual or other information that qualifies the main value.

- **Example:**

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

- In this case, it is better to have **2.4 kilograms** rather than just the decimal value 2.4.

Example (cont’d)

- Here we have a structured value (2.4 kilograms) with a **main value** (2.4) and another part (the unit of measure).

- RDF allows us to write:

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

- RDF does not have a special semantics for `rdf:value`; it simply offers it as a utility property.

Summary

- In RDF, **statements** about a domain are encoded by **triples**.
- **Triples** are of the form
  subject predicate object .
- The **subject** of a triple must be a **URIref or a blank node.**
- The **predicate** of a triple must be a **URIref**.
- The **object** of a triple must be a **URIref, a literal or a blank node**. Literals are not allowed as subjects or predicates.
- **Definition:** An **RDF graph** is a set of RDF triples.

Presentation Outline

- Basic concepts of RDF
- **Serialization of RDF graphs: XML/RDF and Turtle**
- Other Features of RDF (Containers, Collections and Reification).

Machine Readable Formats for RDF

RDF has a number of machine readable formats:

- XML/RDF
- Turtle (Terse RDF Triple Language)
- N3
- …

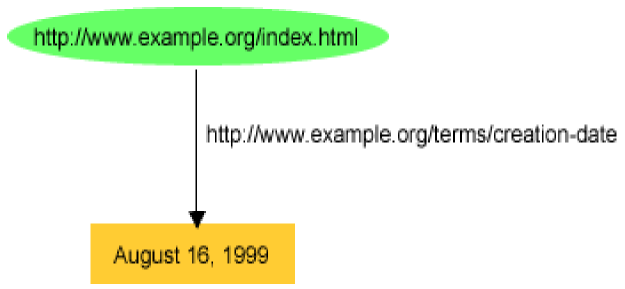
An XML Syntax for RDF: RDF/XML

- The conceptual model for RDF is a graph.
- RDF provides an XML syntax for **writing down and exchanging RDF graphs**, called **RDF/XML**.
- RDF/XML is the **normative syntax** for writing RDF.

Example

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" .
```


http://www.example.org/index.html
http://www.example.org/terms/creation-date
August 16, 1999

Example in XML/RDF

```
<?xml version="1.0"?>
<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>August 16, 1999</exterms:creation-date>
     </rdf:Description>
</rdf:RDF>
```

Turtle

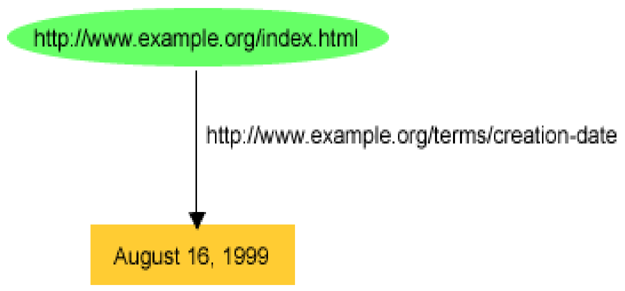
- **Turtle** provides another textual syntax for writing down and exchanging RDF graphs.

- **Turtle** is based on the **triple notation** we used so far, but also includes some additional syntactic means to **simplify the specification** of RDF graphs.

Example I

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" .
```


http://www.example.org/index.html
http://www.example.org/terms/creation-date
August 16, 1999

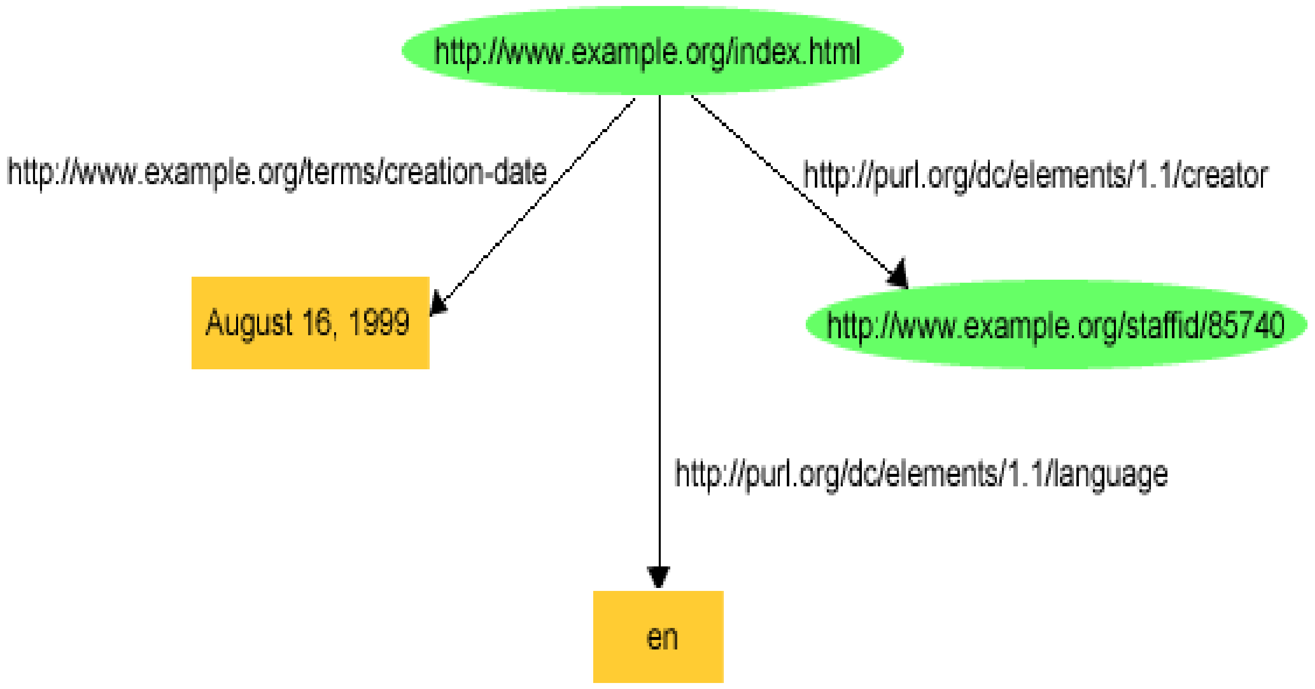
Example I in Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix exterms: <http://www.example.org/terms/>.

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

- **Notation:** The `@prefix` keyword in the first two lines declares namespaces `rdf:` and `exterms:`

Example II


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

Example II Using Triples

```
<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" .
```

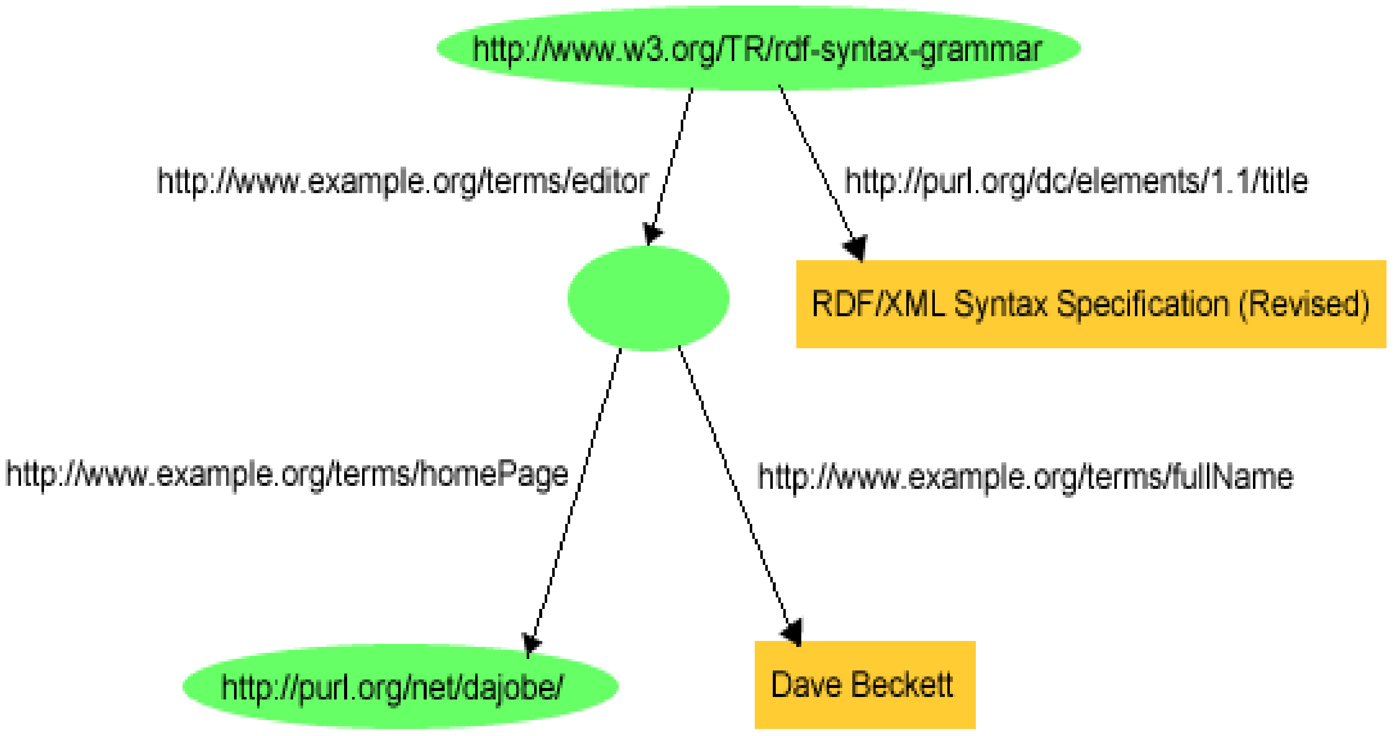
Example II Using Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/#>.
@prefix exterms: <hhttp://www.example.org/terms/>.

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

- **Notation:** In this case Turtle allows the semi-colon to separate **predicate-object pairs** for the same subject. A list of such pairs is terminated with a period.

Blank Nodes in Turtle


http://www.w3.org/TR/rdf-syntax-grammar
http://www.example.org/terms/editor
http://purl.org/dc/elements/1.1/title
RDF/XML Syntax Specification (Revised)
http://www.example.org/terms/homePage
http://www.example.org/terms/fullName
http://purl.org/net/dajobe/
Dave Beckett

Blank Nodes in Turtle (cont'd)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/#>.
@prefix exterms: <http://www.example.org/terms/>.

<http://www.w3.org/TR/rdf-syntax-grammar>
    dc:title "RDF/XML Syntax Specification (Revised)";
    exterm:editor _:abc.

 _:abc
    exterms:fullName "Dave Beckett";
    exterms:homePage <http://purl.org/net/dajobe/>.
```

- **Notation:** Blank node identifiers are used.

Anonymous Blank Nodes

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/#>.
@prefix exterms: <http://www.example.org/terms/>.

<http://www.w3.org/TR/rdf-syntax-grammar>
     dc:title "RDF/XML Syntax Specification (Revised)";
     exterm:editor [
          exterms:fullName "Dave Beckett";
          exterms:homePage <http://purl.org/net/dajobe/>.
     ].
```

- **Notation:** This is reminiscent of notations for complex objects from the area of database systems.

Presentation Outline

- Basic concepts of RDF
- Serialization of RDF graphs: XML/RDF and Turtle
- **Other Features of RDF (Containers, Collections and Reification).**

Other Features of RDF

- Containers (Bags, Sequences, Alternatives).
- Collections
- Reification

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 a **container vocabulary** consisting of three predefined types to describe groups of things.
- A **container** is a resource that contains things. The contained things are resources called **members**.

RDF Containers (cont'd)

- RDF defines three types of containers:
  - `rdf:Bag`
  - `rdf:Seq`
  - `rdf:Alt`
- A **bag** (a resource having type `rdf:Bag`) represents a group of resources or literals, possibly including duplicate members, where there is no significance in the order of the members.
- A **sequence** represents a group of resources or literals, possibly including duplicate members, where the order of the members is significant.
- An **alternative** represents a group of resources or literals that are alternatives (typically for a single value of a property).
- `rdf:Bag, rdf:Seq` **and** `rdf:Alt` are classes and they are a subclass of `rdfs:Container.`

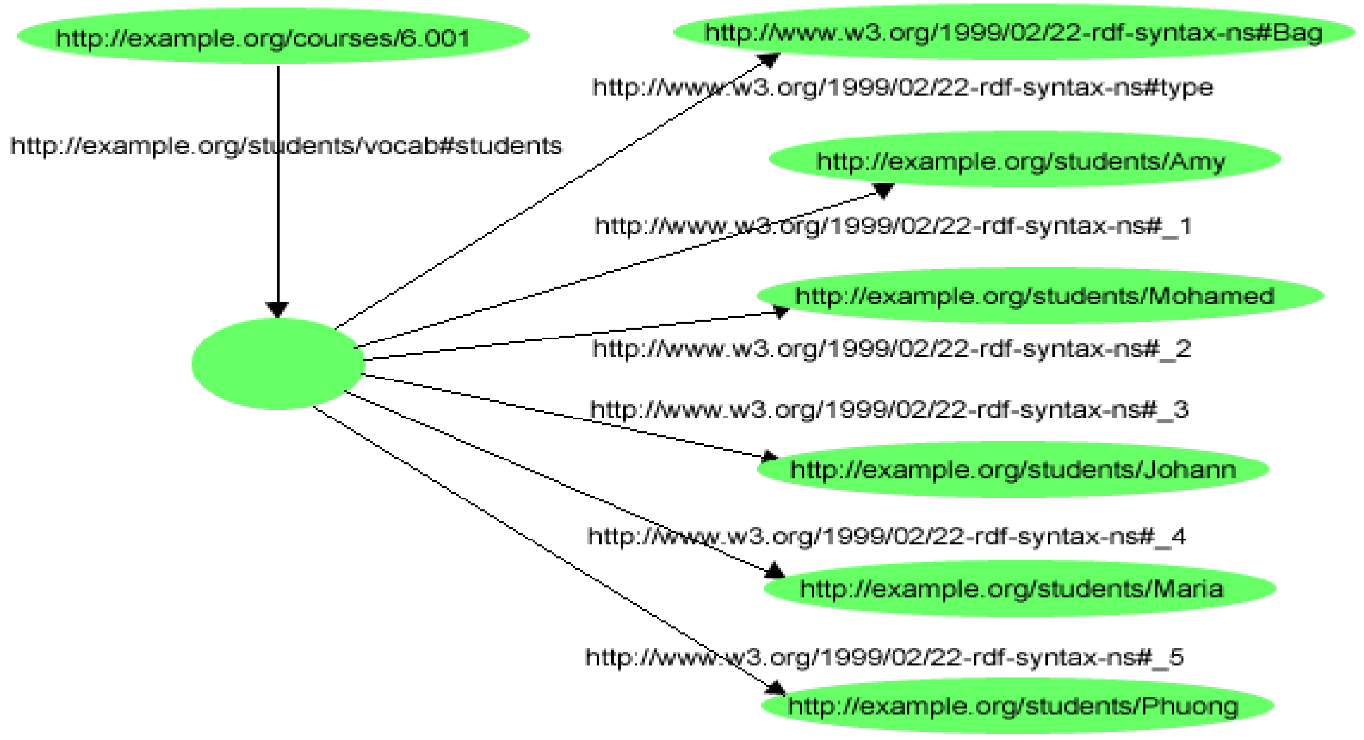
RDF Containers (cont'd)

- To describe a resource as being one of these types of containers, 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** (which may either be a blank node or a resource with a URIref) 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.
- The 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.

Example of Bag

- Consider the sentence "Course 6.001 has the students Amy, Mohamed, Johann, Maria, and Phuong" .

Example of Bag (cont’d)


http://example.org/courses/6.001
http://example.org/students/vocab#students
http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://example.org/students/Amy
http://www.w3.org/1999/02/22-rdf-syntax-ns#_1
http://example.org/students/Mohamed
http://www.w3.org/1999/02/22-rdf-syntax-ns#_2
http://www.w3.org/1999/02/22-rdf-syntax-ns#_3
http://example.org/students/Johann
http://www.w3.org/1999/02/22-rdf-syntax-ns#_4
http://example.org/students/Maria
http://www.w3.org/1999/02/22-rdf-syntax-ns#_5
http://example.org/students/Phuong

Example of Bag (cont’d)

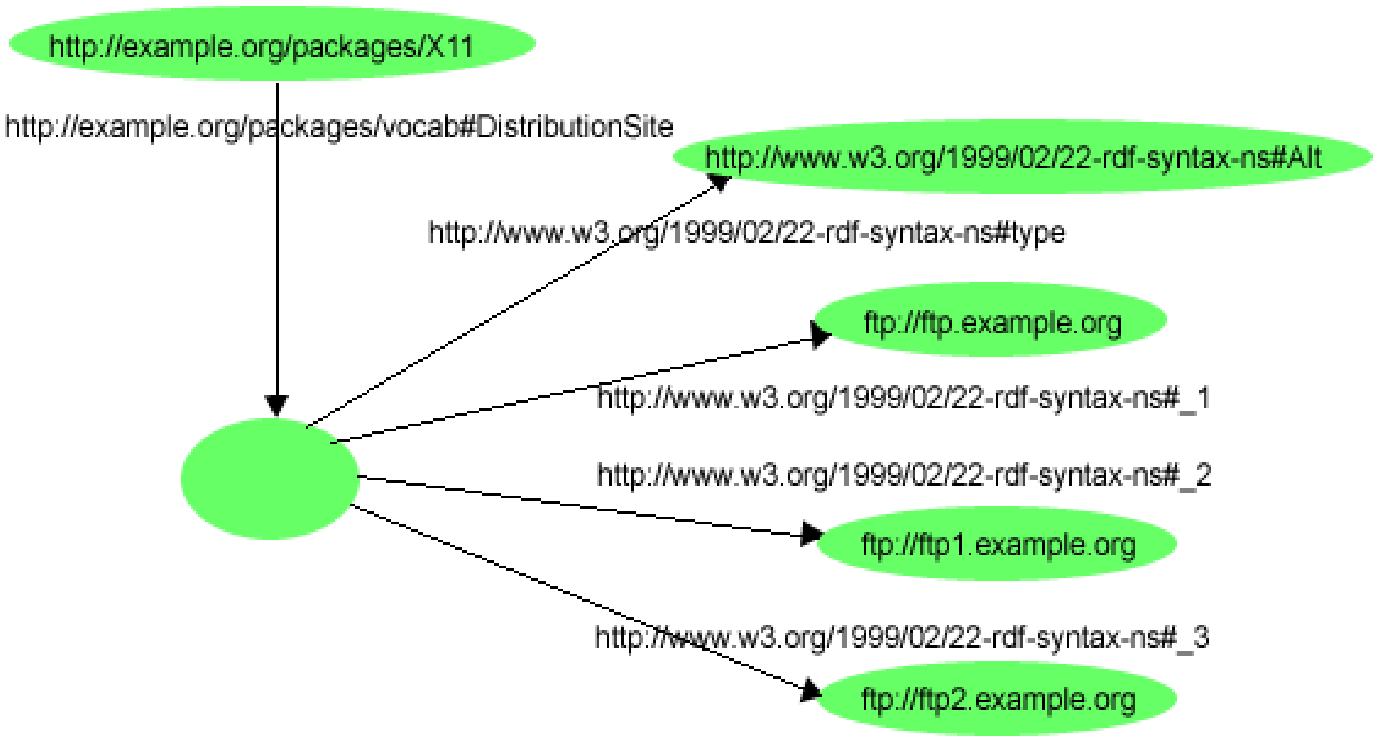
- In Turtle notation:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix s: <http://example.org/students/vocab#>.
<http://example.org/courses/6.001>
    s:students [
        a rdf:Bag;
        rdf:_1 <http://example.org/students/Amy>;
        rdf:_2 <http://example.org/students/Mohamed>;
        rdf:_3 <http://example.org/students/Johann>;
        rdf:_4 <http://example.org/students/Maria>;
        rdf:_5 <http://example.org/students/Phuong>.
    ].
```

Example of Alternative

- Consider the sentence:
  - "The source code for X11 may be found at ftp.example.org, ftp1.example.org, or ftp2.example.org" .

Example of Alternative (cont'd)


http://example.org/packages/X11
http://example.org/packages/vocab#DistributionSite
http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
ftp://ftp.example.org
http://www.w3.org/1999/02/22-rdf-syntax-ns#_1
http://www.w3.org/1999/02/22-rdf-syntax-ns#_2
ftp://ftp1.example.org
http://www.w3.org/1999/02/22-rdf-syntax-ns#_3
ftp://ftp2.example.org

Example of Alternative (cont'd)

- In Turtle notation:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix s: <http://example.org/packages/vocab#>.

<http://example.org/packages/X11>
   s:DistributionSite [
      a rdf:Alt;
      rdf:_1 <ftp://ftp.example.org>;
      rdf:_2 <ftp://ftp1.example.org>;
      rdf:_3 <ftp://ftp2.example.org>.
   ].
```

Containers vs. simple triples

- Consider the statement:
  - Sue has written "Anthology of Time", "Zoological Reasoning", and "Gravitational Reflections".

- The above statement can be expressed in RDF using **three triples**:
  - `exstaff:Sue exterms:publication ex:AnthologyOfTime .`
  - `exstaff:Sue exterms:publication ex:ZoologicalReasoning .`
  - `exstaff:Sue exterms:publication ex:GravitationalReflections .`

Containers vs. simple triples (cont'd)

- Alternatively, a bag can be used:

```
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 .
```

Containers vs. simple triples (cont'd)

- However there cases where using a container is the most prominent modeling option.
- Consider the statement:
  - The resolution was approved by the Rules Committee, having members Fred, Wilma, and Dino.
- The statement says that the committee *as a* **whole** approved the resolution; it does not necessarily state that each committee member **individually** voted in favor of the resolution.

Containers vs. simple triples (cont'd)

```
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 .
```

Containers (cont’d)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix s: <http://example.org/packages/vocab#>.
<http://example.org/packages/X11>
    s:DistributionSite [
        a rdf:Alt;
        a rdf:Bag;
        rdf:_2 <ftp://ftp.example.org>;
        rdf:_2 <ftp://ftp1.example.org>;
        rdf:_5 <ftp://ftp2.example.org>
    ].
```

- The above example is a well-formed RDF desciption although the resource has been defined as an instance of both `rdf:Alt` and `rdf:Bag`. Also, the property `rdf:_2` has two values**.**

- RDF does not enfoce any “well-formedness constraint” here. RDF applications that require containers to be "well-formed" should be written to check that the container vocabulary is being used appropriately, in order to be fully robust.

RDF Collections

- A limitation of the containers is that there is no way to **close** them, i.e., to say "these are all the members of the container". A container only says that certain identified resources are members; it does not say that other members do not exist.

- RDF provides support for describing groups containing **only the specified members**, in the form of RDF **collections**.

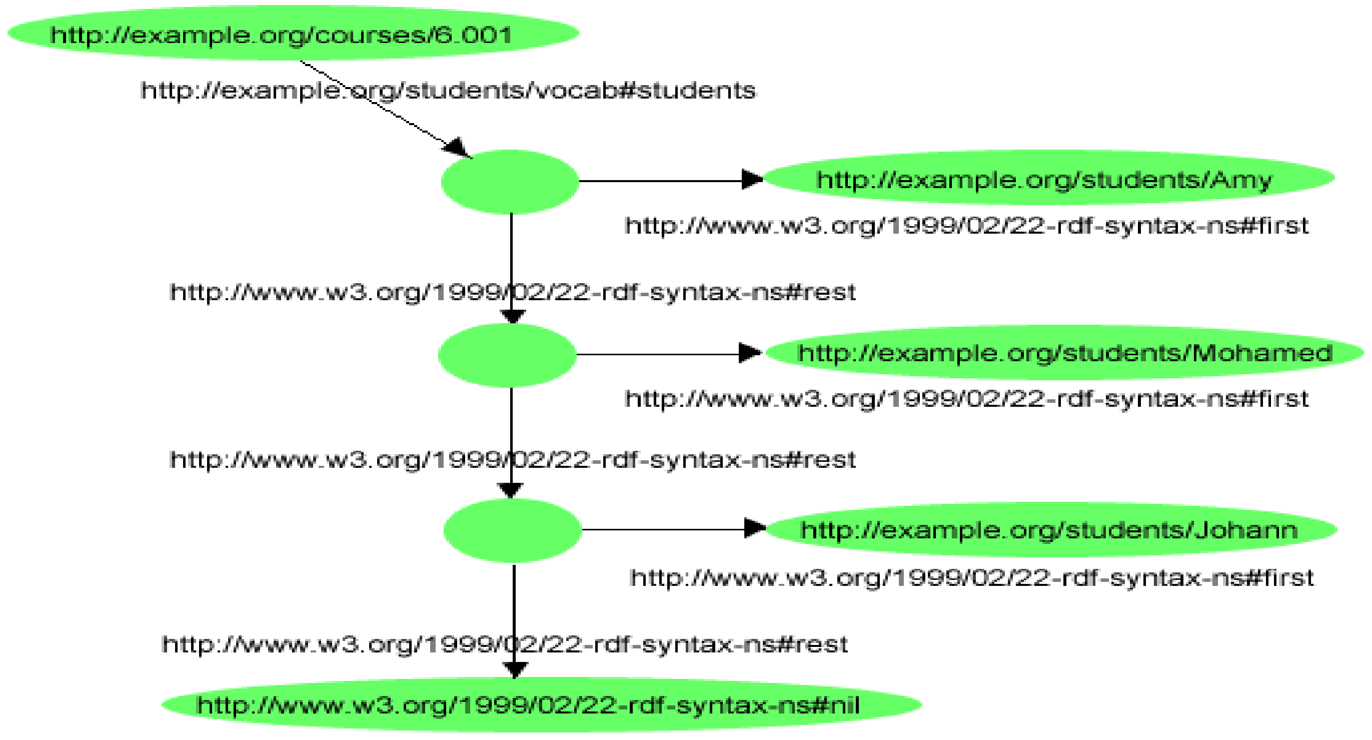
- An **RDF collection** is a group of things represented as a **list structure** in the RDF graph. This list structure is constructed using a predefined **collection vocabulary** consisting of the predefined type `rdf:List`, the predefined properties `rdf:first` and `rdf:rest`, and the predefined resource `rdf:nil`.

- RDF does not enforce well-formedness constraints here too; this is left to applications.

Example

- Consider the sentence “The students in course 6.001 are Amy, Mohamed and Johann" .

Example as Graph


http://example.org/courses/6.001
http://example.org/students/vocab#students
http://example.org/students/Amy
http://www.w3.org/1999/02/22-rdf-syntax-ns#first
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
http://example.org/students/Mohamed
http://www.w3.org/1999/02/22-rdf-syntax-ns#first
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
http://example.org/students/Johann
http://www.w3.org/1999/02/22-rdf-syntax-ns#first
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
http://www.w3.org/1999/02/22-rdf-syntax-ns#nil

Example in Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
  syntax-ns#>.
@prefix s: <http://example.org/students/vocab#>.

<http://example.org/courses/6.001>
   s:students (
      http://example.org/students/Amy
      http://example.org/students/Mohamed
      <http://example.org/students/Johann>
   ).
```

Reification

- RDF applications sometimes need to **describe other RDF statements** using RDF, for instance, to record information about when statements were made, who made them, or other similar information (this is sometimes referred to as "**provenance" information**).

- **Example:** The company example.com might want to record who made the statement

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

Reification (cont'd)

- 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 **class** `rdf:Statement`, and the properties `rdf:subject, rdf:predicate`, and `rdf:object`.

Example (cont'd)

- Assign the statement about the tent's weight a URIref such as `exproducts:triple12345.`

- Now statements can be written describing the statement. For example:

```
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 .
```

- Four statements like the above are usually called "**the reification quad**".

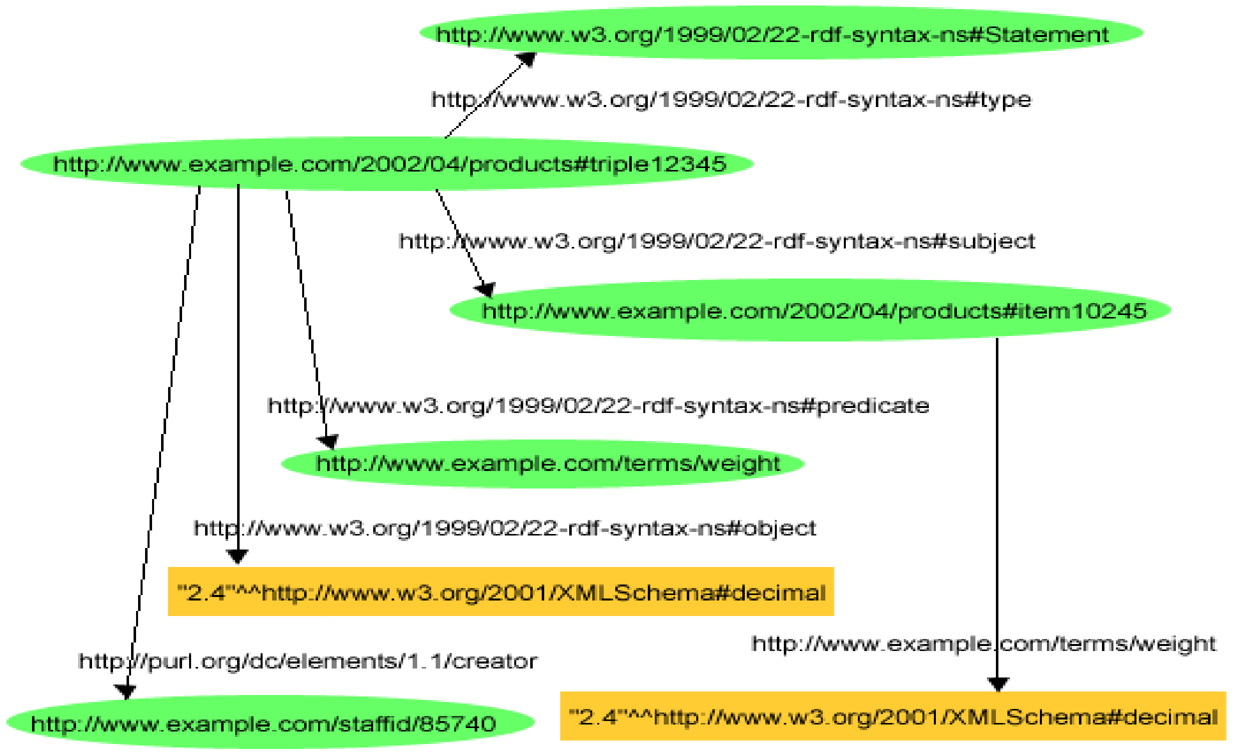
Example (cont'd)

- Now we can add information about who made the statement:

```
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 Statement, its Reification and its Attribution


http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.example.com/2002/04/products#triple12345
http://www.w3.org/1999/02/22-rdf-syntax-ns#subject
http://www.example.com/2002/04/products#item10245
http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate
http://www.example.com/terms/weight
http://www.w3.org/1999/02/22-rdf-syntax-ns#object
"2.4"^^http://www.w3.org/2001/XMLSchema#decimal
http://www.example.com/terms/weight
http://purl.org/dc/elements/1.1/creator
http://www.example.com/staffid/85740
"2.4"^^http://www.w3.org/2001/XMLSchema#decimal

Reification (cont’d)

- In the previous graph, nothing indicates that the original statement describing the tent's weight is the resource `exproducts:triple12345`, the resource that is the subject of the four reification statements and the statement that `exstaff:85740` created it.
- Asserting the reification is not the same as asserting the original statement, and neither implies the other.

Reification (cont'd)

- The reification mechanism of RDF is **weak.**
- Applications are left to interpret and use the reification vocabulary in a proper manner.
- What RDF offers is **not enough to identify a particular triple** that is being reified and assert information about that triple (e.g., who wrote it, when etc.)
- There is more recent work on **named graphs** that offers a better approach to these issues. See the paper "Named Graphs, Provenance and Trust" available at http://www2005.org/cdrom/docs/p613.pdf .

Readings

- Chapters 2 and 3 of the Semantic Web Primer available from http://www.csd.uoc.gr/~hy566/SWbook.pdf .

- The following material from the Semantic Web Activity Web page on RDF http://www.w3.org/RDF/ :
  - RDF Primer. The version on the above Web page uses RDF/XML; don't forget to see the version based on Turtle at http://www.w3.org/2007/02/turtle/primer/ .
  - Resource Description Framework (RDF): Concepts and Abstract Syntax

- Check out the content published at the RDF namespace URI:
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#

  where you will find an RDF Schema description of the RDF vocabulary given in RDF/XML!

- The DBpedia project (http://dbpedia.org/About), a nice application of RDF and Linked Data (http://linkeddata.org/).