INFO-H-509 XML Technologies

Practical session 7: RDF and RDF Schema

Based on the corresponding lecture slides of
Stijn Vansummeren

April 19, 2019

Before We Begin

Disclaimer

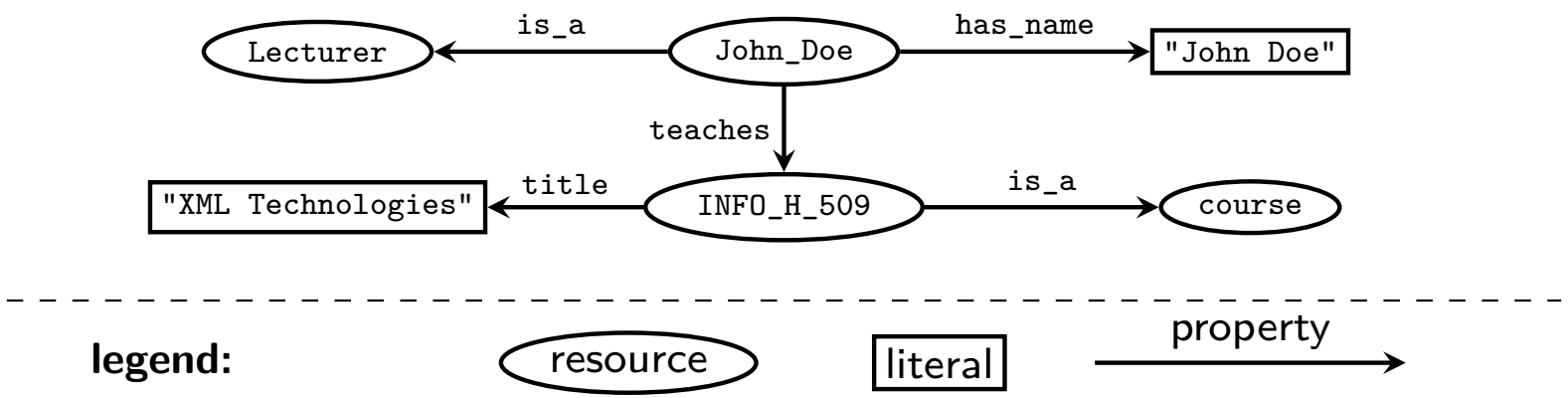
Some of the following examples have been copied from:

- The W3C RDF Primer at `http://www.w3.org/TR/rdf-primer/`
- Joshua Tauberer RDF intro at `http://rdfabout.com/intro/`
- The book Foundations of Semantic Web Technologies (Hitzler, Krötsch, Rudolph).

The RDF Graph: Resources and Literals

- The **Resource Description Framework** (RDF) is a general method for describing knowledge in the form of a directed, labeled **graph**.
- There are two kinds of nodes:
  1. Nodes that represent the 'things' we are describing: **resources** or **entities**
  2. Nodes that represent particular values of a property **literals** or **literal nodes**
- Literals can only have incoming edges
- The edge labels are called **predicates** or **properties**

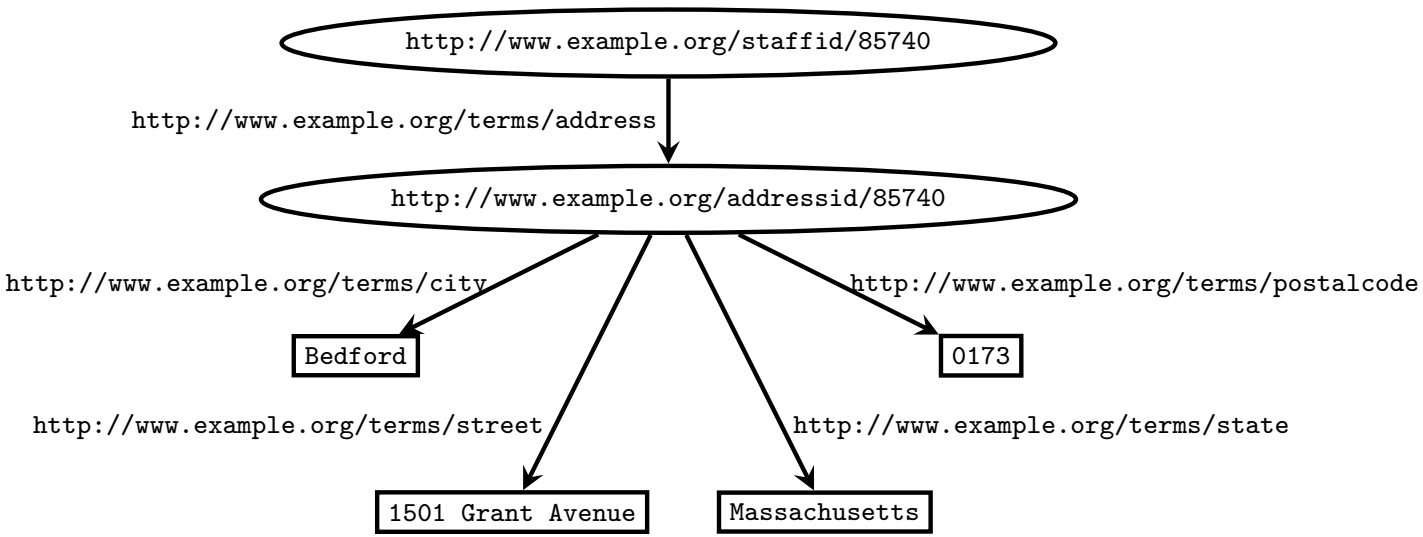
Example (not valid RDF):



The RDF Graph: n-ary relations

- In RDF, the only relationships that we can make is between two URIs (resources), or between a URI and a literal.
- To express *n*-ary relationships, we need to create extra resources.

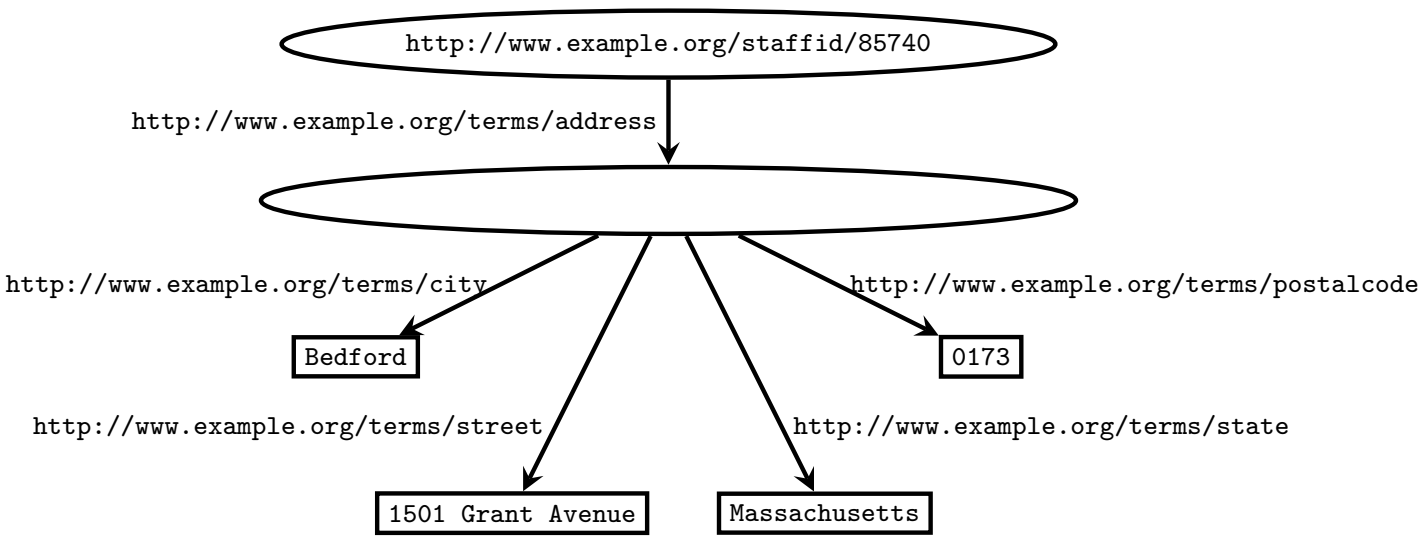
Example: street address as its own resource (n-ary relation)



The RDF Graph: Blank Nodes

- Sometimes inventing a new URI for every resource becomes tedious
- RDF also allows for nodes without URIs: these are called **blank nodes** or **anonymous resources**
- A blank node hence denotes an entity for which we do not have or do not want to create a globally unique name

Example: street address with blank node



The RDF Graph: Containers

Example: Course 509 has the students Amy, Mohamed, and John

The RDF Graph: Containers

There is often a need to describe **groups** of things.

- RDF Defines three predefined types to denote containers:
  - Unordered groups with duplicates (a **Bag**)

    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag`
  - Ordered groups with duplicates (a **Sequence**)

    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq`
  - A group of alternatives (an **Alternative**)

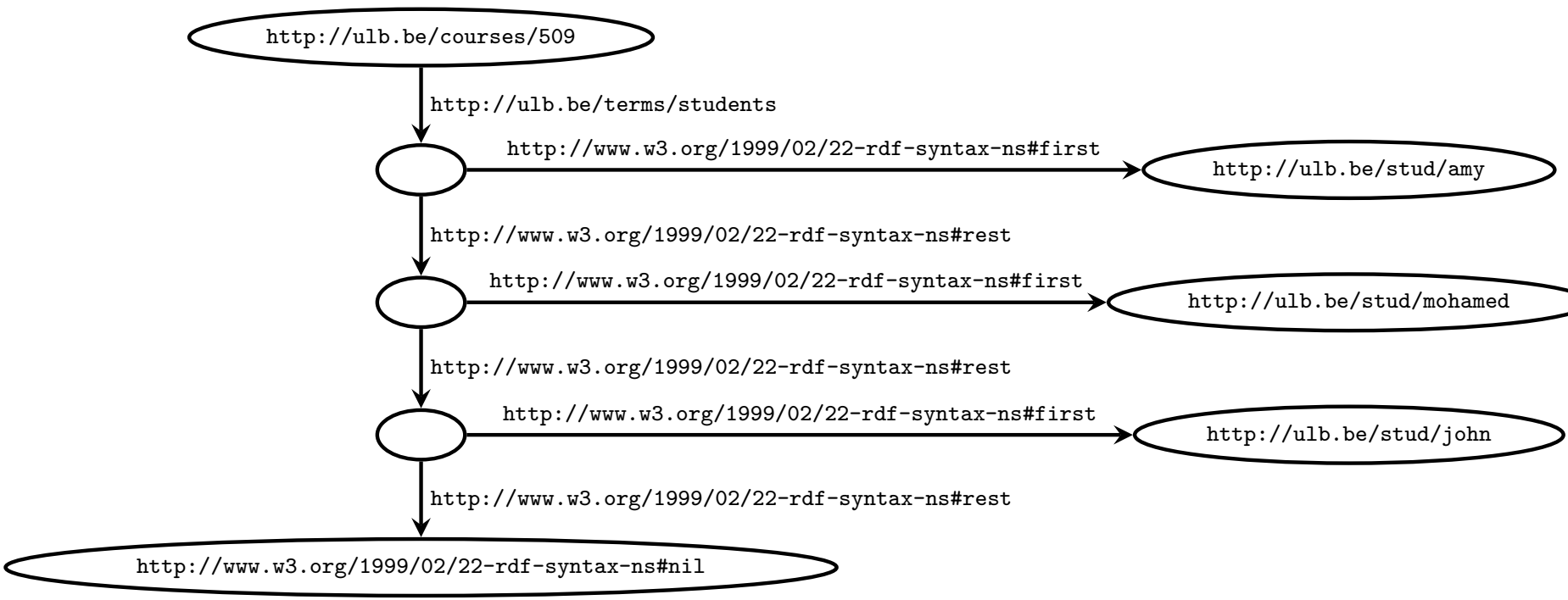
    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt`
- The *n*-th item in a collection is specified by the property

  `http://www.w3.org/1999/02/22-rdf-syntax-ns#_`*n*

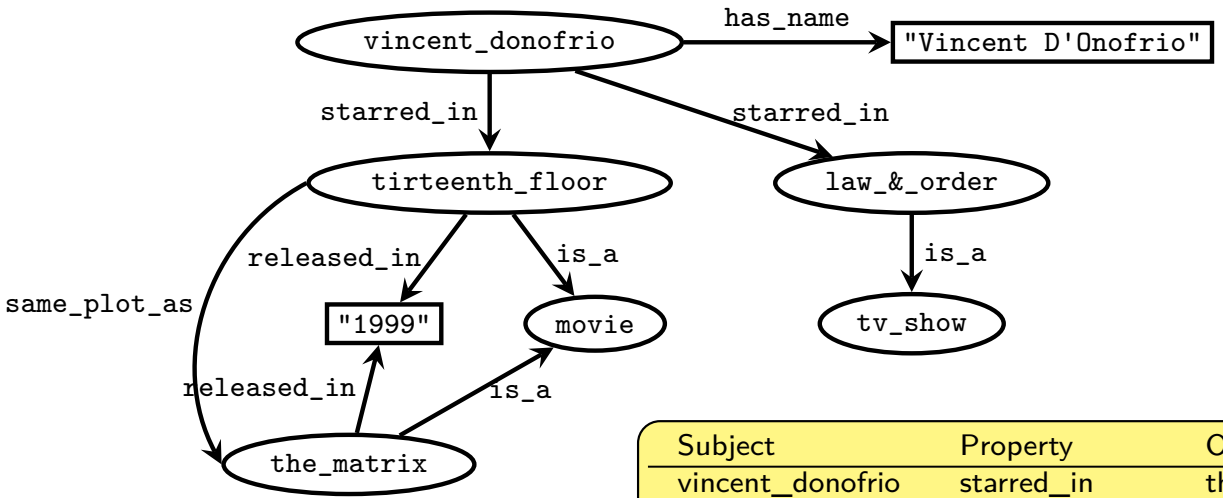
The RDF Graph: Collections

- Containers cannot be "closed"
- To specify "closed" groups RDF introduces **collections**, which are list structures formed by:
  - `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: Course 509 has exactly the students Amy, Mohamed, and John



An RDF Graph ≡ Set of Triples



Subject	Property	Object
vincent_donofrio	starred_in	thirteenth_floor
vincent_donofrio	starred_in	law_&_order
vincent_donofrio	has_name	"Vincent D'Onofrio"
thirteenth_floor	released_in	"1999"
thirteenth_floor	is_a	movie
the_matrix	is_a	movie
the_matrix	released_in	"1999"
thirteenth_floor	same_plot_as	the_matrix
law_&_order	is_a	tv_show

- A **triple** is also called a **statement**
- To store a (piece of) an RDF graph, it suffices to store its set of triples
- This can be done in multiple **serialization formats**

Turtle (1/2)

Turtle extends the NTriples format with some convenient features:

- the `@prefix` directive allows abbreviation of URIs
- the `a` property abbreviates `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
- XML Schema datatypes may be abreviated using the `xsd:` prefix
- The comma symbol can be used to repeat the subject and predicate of triples that only differ in the object
- The semicolon symbol can be used to repeat the subject of triples that only differ in the predicate and object v

Example:

```
@prefix staff: <http://www.example.org/staff id/> .
@prefix : <http://www.example.org/terms/> .

staff:85740    :address       _:addr                    .
_:addr         :city          "Bedford"^^xsd:string     ;
               :street        "1501 Grant Avenue"       ;
               :state         "Massachusetts"           ;
               :postalcode    "0713"                    .
staff:85740    a              :employee                 .
```

Turtle (2/2)

Turtle extends the NTriples format with some convenient features:

- Blank nodes can be described by nesting Turtle statements in `[ ]`
- Collections can be described by resources between parenthesis `(...)`

Example:

```
@prefix staff: <http://www.example.org/staff id/> .
@prefix : <http://www.example.org/terms/> .

staff:85740    :address    [ :city             "Bedford"^^xsd:string ;
                             :street           "1501 Grant Avenue" ;
                             :state            "Massachusetts" ;
                             :postalcode       "0713"                ] .
staff:85740    a           :employee                                     .
```

Turtle (2/2)

Turtle extends the NTriples format with some convenient features:

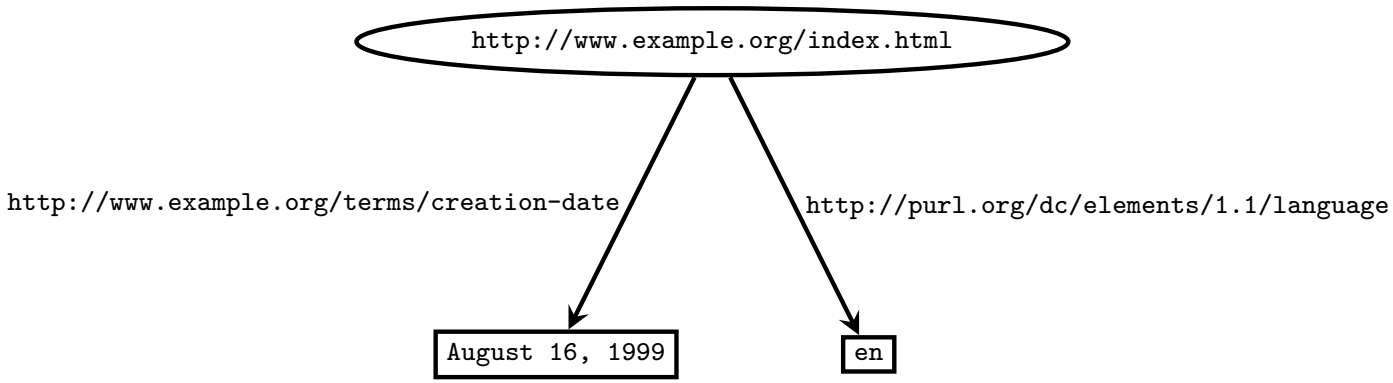
- Blank nodes can be described by nesting Turtle statements in `[ ]`
- Collections can be described by resources between parenthesis `(...)`

Example:

```
@prefix courses: <http://ulb.be/courses/> .
@prefix terms: <http://ulb.be/terms/> .
@prefix : <http://ulb.be/students/> .

courses:509    terms:students    ( :amy :mohamed :john ) .
```

RDF/XML

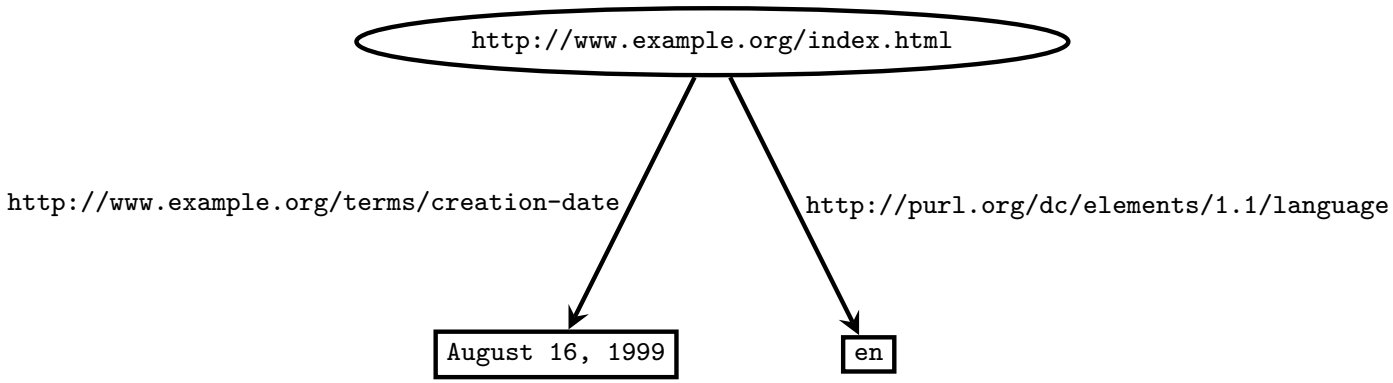
Example of multiple statements :




```
<?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/"
         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:Description rdf:about="http://www.example.org/index.html">
    <dc:language>en</dc:language>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML

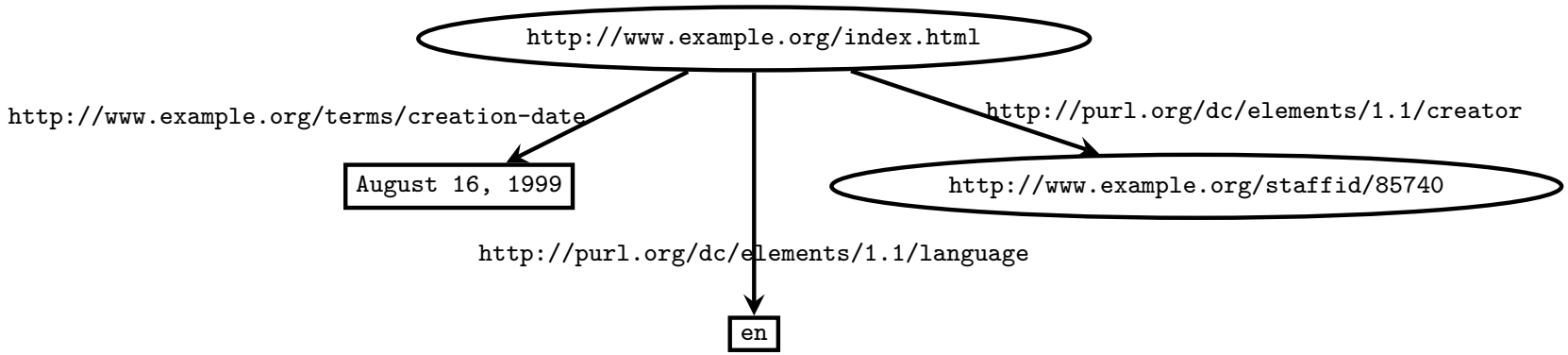
Example of multiple statements with abbreviation:




```
<?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/"
         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>
    <dc:language>en</dc:language>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML

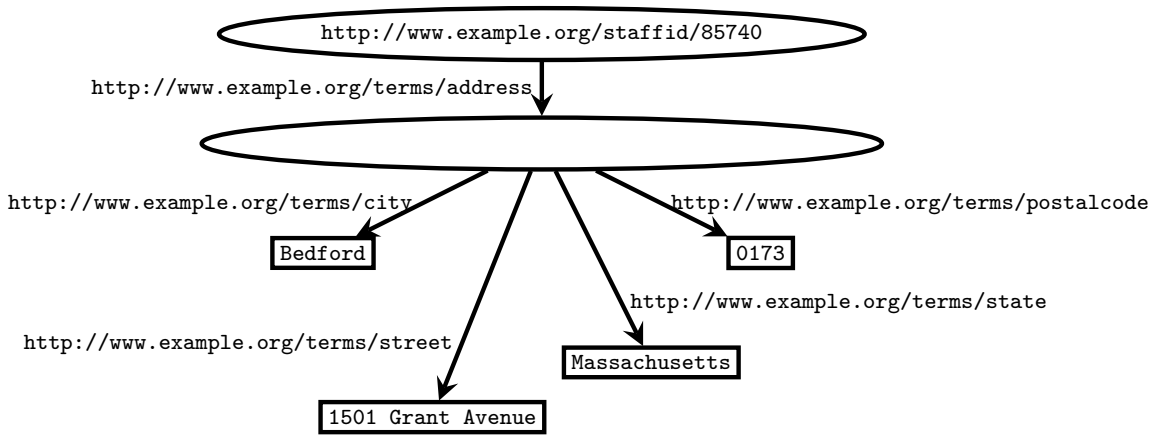
Example of a statement with a resource as object:




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

</rdf:RDF>
```

RDF/XML

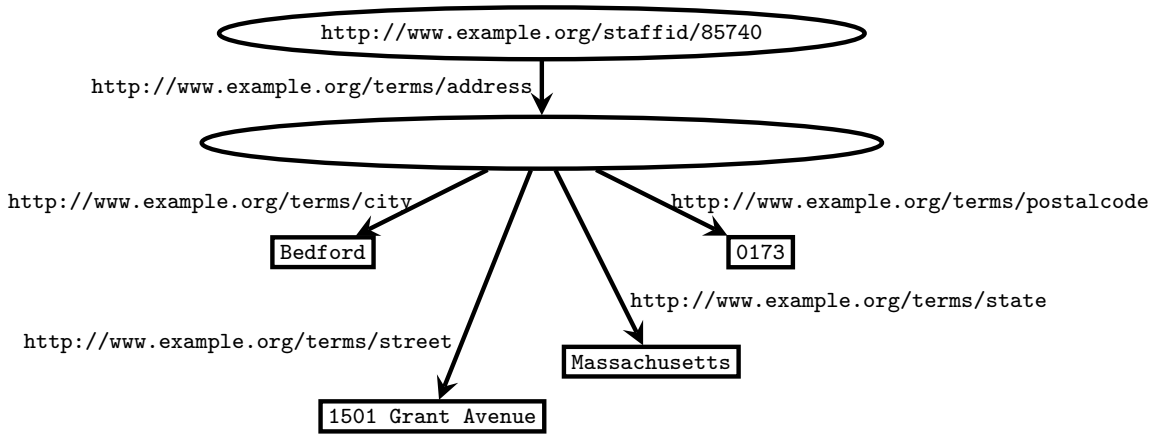
Example of a blank node:




```
<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/staffid/85740">
    <exterms:addresss rdf:nodeID="abc"/>
  </rdf:Description>

  <rdf:Description rdf:nodeID="abc">
    <exterms:city>Bedford</exterms:city>
    <exterms:street>1501 Grant Avenue</exterms:street>
    <exterms:state>Massachusetts</exterms:state>
    <exterms:postalcode>0173</exterms:postalcode>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

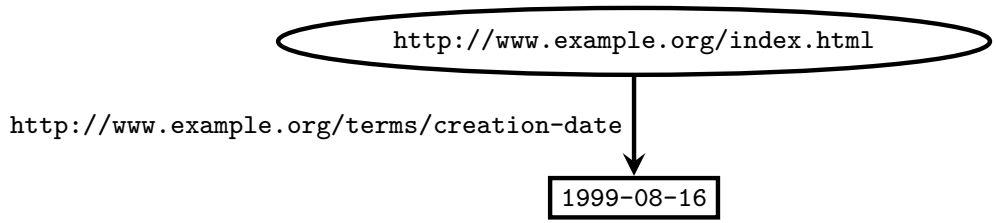
Example of a blank node (alternate syntax):




```
<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/staffid/85740">
    <exterms:addresss rdf:parseType="Resource">
      <exterms:city>Bedford</exterms:city>
      <exterms:street>1501 Grant Avenue</exterms:street>
      <exterms:state>Massachusetts</exterms:state>
      <exterms:postalcode>0173</exterms:postalcode>
    </exterms:addresss>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

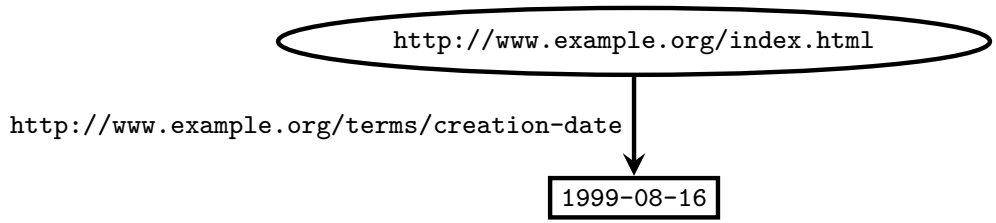
Example of a typed literal:




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

</rdf:RDF>
```

RDF/XML

Example of a typed literal:

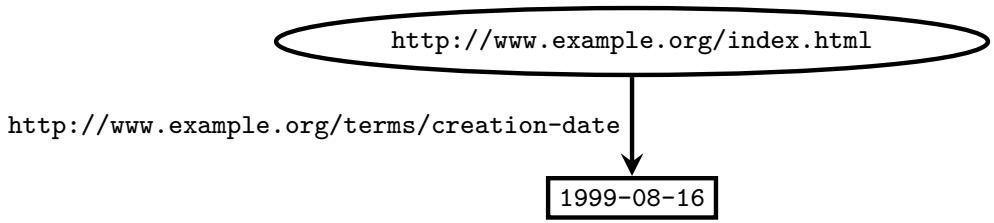



```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

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

</rdf:RDF>
```

Note that URIs that occur in attributes must be written in full!

RDF/XML

Example of using entities to abbreviate URIs:




```
<?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>
```

RDF/XML

Example of container - first possiblity:

```
<?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:_1 rdf:resource="http://example.org/students/Amy"/>
        <rdf:_2 rdf:resource="http://example.org/students/Mohamed"/>
        <rdf:_3 rdf:resource="http://example.org/students/Johann"/>
        <rdf:_4 rdf:resource="http://example.org/students/Maria"/>
        <rdf:_5 rdf:resource="http://example.org/students/Phuong"/>
      </rdf:Bag>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

Example of container - second possiblity:

```
<?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>
```

How do we specify ontologies?

- RDF Schema and the Ontology Web Language (OWL) allow ontologies to be specified **in RDF itself**: they provide resources and properties in a **standard namespace** to talk about class relationships, properties of classes, ...
- They also specify how these resources and properties should be interpreted (in terms of inference)
- OWL allows more fine-grained knowledge of the world to be specified, at the cost of less efficient inference and reasoning

Example:

```
@prefix terms: <http://www.example.org/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

terms:digital-camera      rdfs:subClassOf   terms:digital-device .
terms:netbook             rdfs:subClassOf   terms:digital-device .
terms:book                rdfs:subClassOf   terms:media .
terms:digital-device      rdfs:subClassOf   terms:electronic-device .
terms:electronic-device   rdfs:subClassOf   terms:device .
```

Classes and Class Hierarchies

- Classes stand for sets of things (in RDF: sets of URIs)
- Use `rdf:type` to indicate class membership
- A URI can belong to several classes.
- `X rdfs:subClassOf Y` is used to specify that all members of a class X are also members of a class Y
- This allows to arrange classes in hierarchies.
- And allows expressing that two classes are the same by mutual inclusion: C_1 subClassOf C_2 and C_2 subClassOf C_1

Deduction Rule

If	u `rdfs:subClassOf` x .
And	y `rdf:type` u .
Then infer	y `rdf:type` x .

Example

:mary	rdf:type	:Woman .
:mary	rdf:type	:Student .
:Woman	rdfs:subClassOf	:Person .
:Person	rdfs:subClassOf	:Animal .
:mary	rdf:type	:Woman .
:mary	rdf:type	:Animal .

Properties and Property Hierarchies

- Everything that occurs in the middle of a triple is a property.
- `X rdfs:subPropertyOf Y` is used to specify that all resources with property X to some object also have property Y to that object
- This allows to arrange properties also in hierarchies.
- And expressing that two properties are the same (by mutual inclusion).

Deduction Rule

If	a `rdfs:subPropertyOf` b .
And	x a y .
Then add	x b y.

Example

:john	:happilyMariedTo	:mary	.
:happilyMariedTo	rdfs:subPropertyOf	:marriedTo	.
:john	:marriedTo	:mary	.

Property Restriction

- **Property restrictions** allow expressing that a certain property can only be between things of a certain rdf:type.
- `P rdfs:domain C` is used to specify that all subjects with (outgoing) property *P* belong to class *C*
- `P rdfs:range C` is used to specify that all objects with (incoming) property *P* belong to class *C*

Deduction Rule

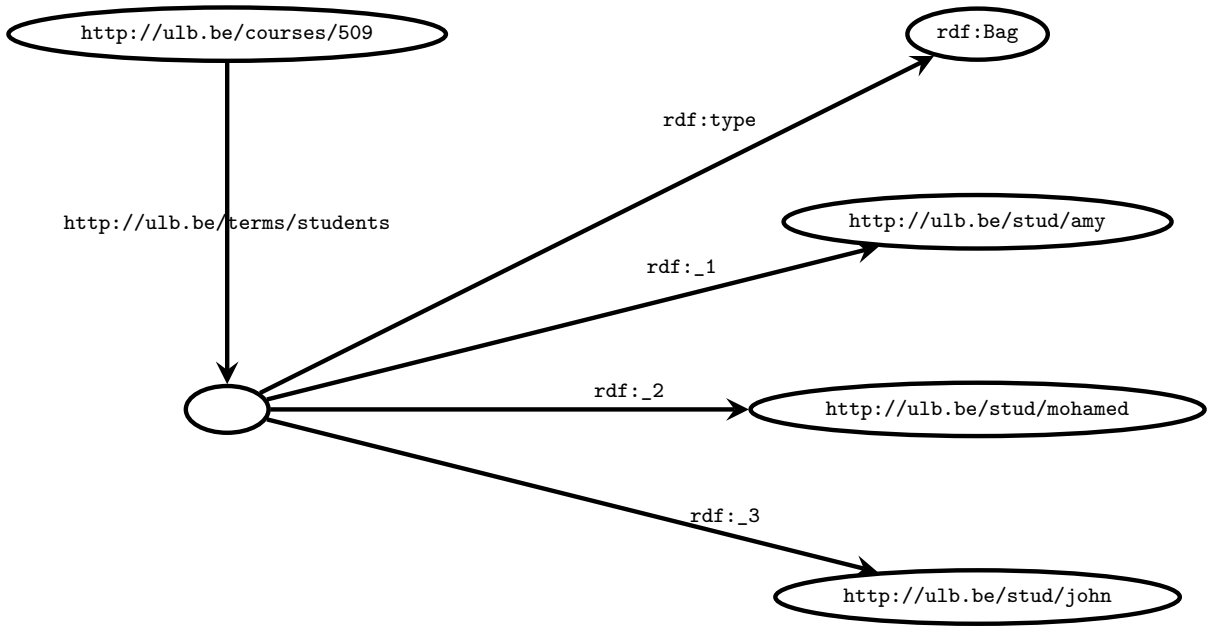
If	*a* `rdfs:domain` *x* .
And	*u a y* .
Then add	*u* `rdf:type` *x* .

If	*a* `rdfs:range` *x* .
And	*u a v* .
Then add	*v* `rdf:type` *x* .

Example

:john	:isMarriedTo	:mary .
:isMarriedTo	rdfs:domain	:Person .
:isMarriedTo	rdfs:range	:Person .
:john	rdf:type	:Person .
:mary	rdf:type	:Person .

Open lists in RDFs



For modeling open lists, RDF schema introduces:

- `rdfs:Container` as the superclass of `rdf:Seq`, `rdf:Bag`, `rdf:Alt`
- `rdfs:ContainerMembershipProperty` as the property that contains all properties used with containers (`rdf:_1`, `rdf:_2`, ...are all of this type).

Open lists in RDFs

In addition, it introduces

- a new property, `rdfs:member`
- which is the superproperty of all properties contained in `rdfs:ContainerMembershipProperty`
- this allows one to introduce their own properties to describe membership in a container.
- and to easily determine whether something is a member of a container without knowing the property used (just look for *u* `rdfs:member` *x*).

If	*a* `rdf:type rdfs:ContainerMembershipProperty .`
And	*u a x .*
Then add	*a* `rdfs:member` *x .*