INFO216:
Advanced Modelling

Theme, spring 2018:
Modelling and Programming the Web of Data

Andreas L. Opdahl
<Andreas.Opdahl@uib.no>

Lecture 2

- Themes:
  - Resource Description Framework (RDF)
    - some known from INFO116
    - some new stuff/more details
    - *all* of RDF
  - Jena's RDF API
    - creating and deleting models, input/output, listing statements, managing literals, type mappings
  - Semantic data sets and vocabularies *(if we have time!)*
    - useful web links
    - aid for project idea
    - *more later!*

Reading

- Sources:
  - Allemang & Hendler (2011): Semantic Web for the Working Ontologist chapter 3 (page 31-44)
  - materials in the wiki: *wiki.uib.no/info216*

Resource Description Framework (RDF)

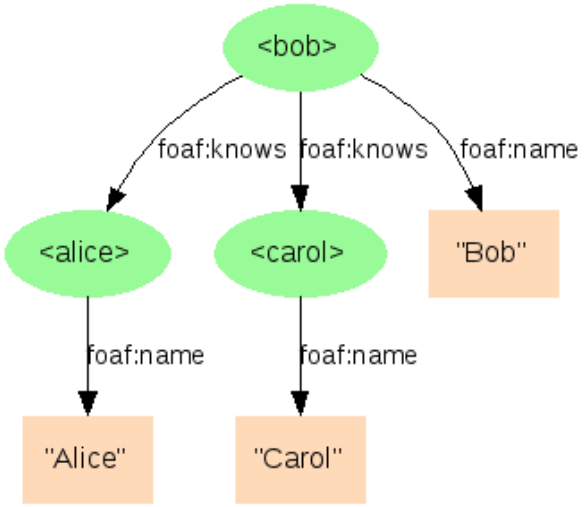
Resource Description Framework (RDF)

- Each data set:
  - treated as a set of *triples*
  - can be physical or virtual
  - exportation can be partial
- The relations form a *directed graph*:
  - "nodes" connected by "arrows"
- "Nodes":
  - either *are* or *represent resources*
  - "leaf nodes" can contain *literal values* (text, numbers, booleans...)
- "Arrows":
  - relations between resources and literals

Statements (triples)

- Triples of *subject predicate object* .
  - ...or *subject predicate literal* .
- The subject:
  - must be a *resource* (→ rdfs:Resource)
  - *named* by an IRI or *anonymous* (*blank node*)
- The predicate:
  - must be a *property* (→ rdf:Property)
  - properties are resources too!
- The object:
  - either a resource (named or anonymous/blank)
  - or a constant *value* (→ rdfs:Literal)

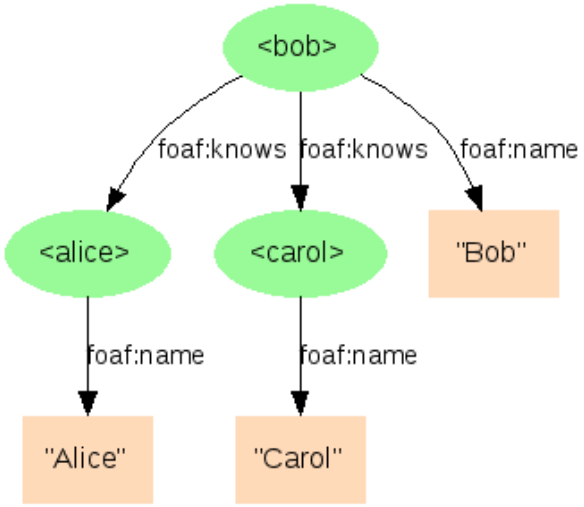
- Triples of *subject predicate object .*
  - ...or of *subject predicate literal* .
  - Int. Resource Identifiers (IRIs)
  - serialisation in *Turtle*:




```
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:bob    rdf:type        foaf:Person .
:bob    foaf:name       "Bob" .
:bob    foaf:mbox       <mailto:alice@example.org> .
:bob    foaf:knows      :alice .
:bob    foaf:knows      :carol .
```

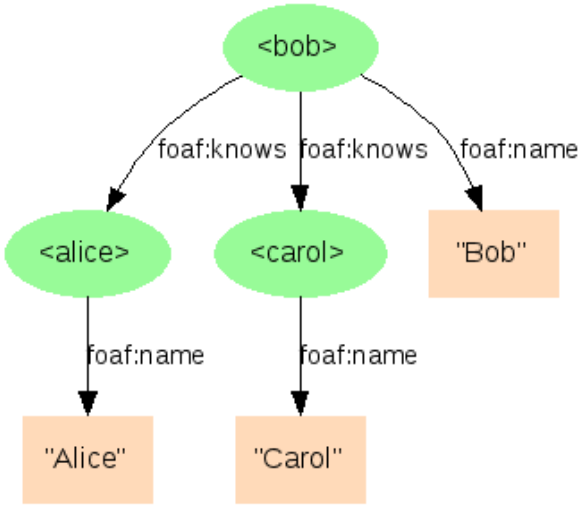
- Triples of *subject predicate object .*
  - ...or of *subject predicate literal* .
  - Int. Resource Identifiers (IRIs)
  - serialisation in *Turtle*:




```
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:bob    rdf:type        foaf:Person ;
        foaf:name       "Bob" ;
        foaf:mbox       <mailto:alice@example.org> ;
        foaf:knows      :alice ;
        foaf:knows      :carol .
```

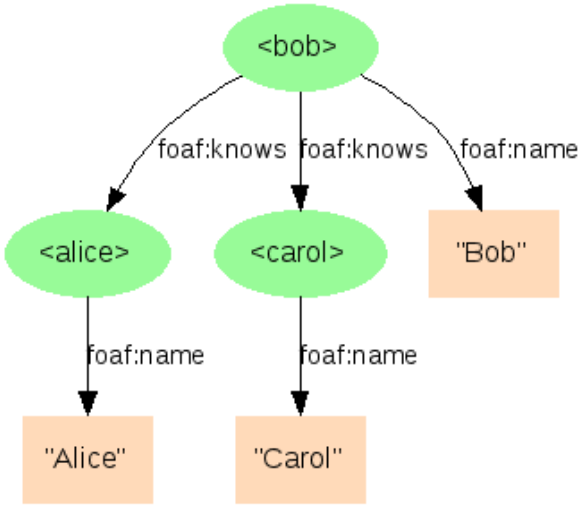
- Triples of *subject predicate object .*
  - ...or of *subject predicate literal* .
  - Int. Resource Identifiers (IRIs)
  - serialisation in *Turtle*:




```
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:bob    a           foaf:Person ;
        foaf:name   "Bob" ;
        foaf:mbox   <mailto:alice@example.org> ;
        foaf:knows  :alice ;
        foaf:knows  :carol .
```

- Triples of *subject predicate object .*
  - ...or of *subject predicate literal* .
  - Int. Resource Identifiers (IRIs)
  - serialisation in *Turtle*:




```
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:bob    a           foaf:Person ;
        foaf:name   "Bob" ;
        foaf:mbox   <mailto:alice@example.org> ;
        foaf:knows  :alice ,
                    :carol .
```

Semantic graphs and data sets

- *Graph (or Model)*:
  - a collection of triples/statements (possibly zero)
- *Data set*:
  - a collection of at least one graph
  - one of the graphs is *default/unnamed*
  - the others are *named*
  - from triples:
    - *(subject, predicate, object)*
  - to quadruples *(quads):*
    - *(graph, subject, predicate, object)*

Resources (→ rdfs:Resource)

- Resources may be physical phenomena (including people and artefacts), information resources, concepts, constructs...
  - ...most things, really :-)
  - ...and information about them
- Can be the *subject* or *object* in a statement
  - but only rdf:Property can be *predicate*
- Can be:
  - *named* by an IRI
  - *anonymous* (blank *node*)
- Resources can have one or more *rdf:type*-s
  - dbpedia:Magnus_Carlsen rdf:type dbpedia:ChessPlayer

Internationalized Resource Identifier (IRI)

- Used to name (identify) resources:
  - URI – Uniform Resource Identifier
  - URL – a URI that is *dereferencable* (“Locator”)
  - URN – a URI that is used to name something
  - initially based on limited ASCII-character sets...
- Generalised into *International Resource Identifier (IRI)*
  - based on a *Unicode-character set* (UTF-8)
    - major security issue: homographic attacks
    - which domain is this? uib.no
  - also IRL, IRN...

Prefixing

- XML Qualified Name (QName):
  - from “eXtensible Markup Language” (XML)
  - provides short forms for much used IRI bases
- Much used prefixes (here in Turtle syntax):

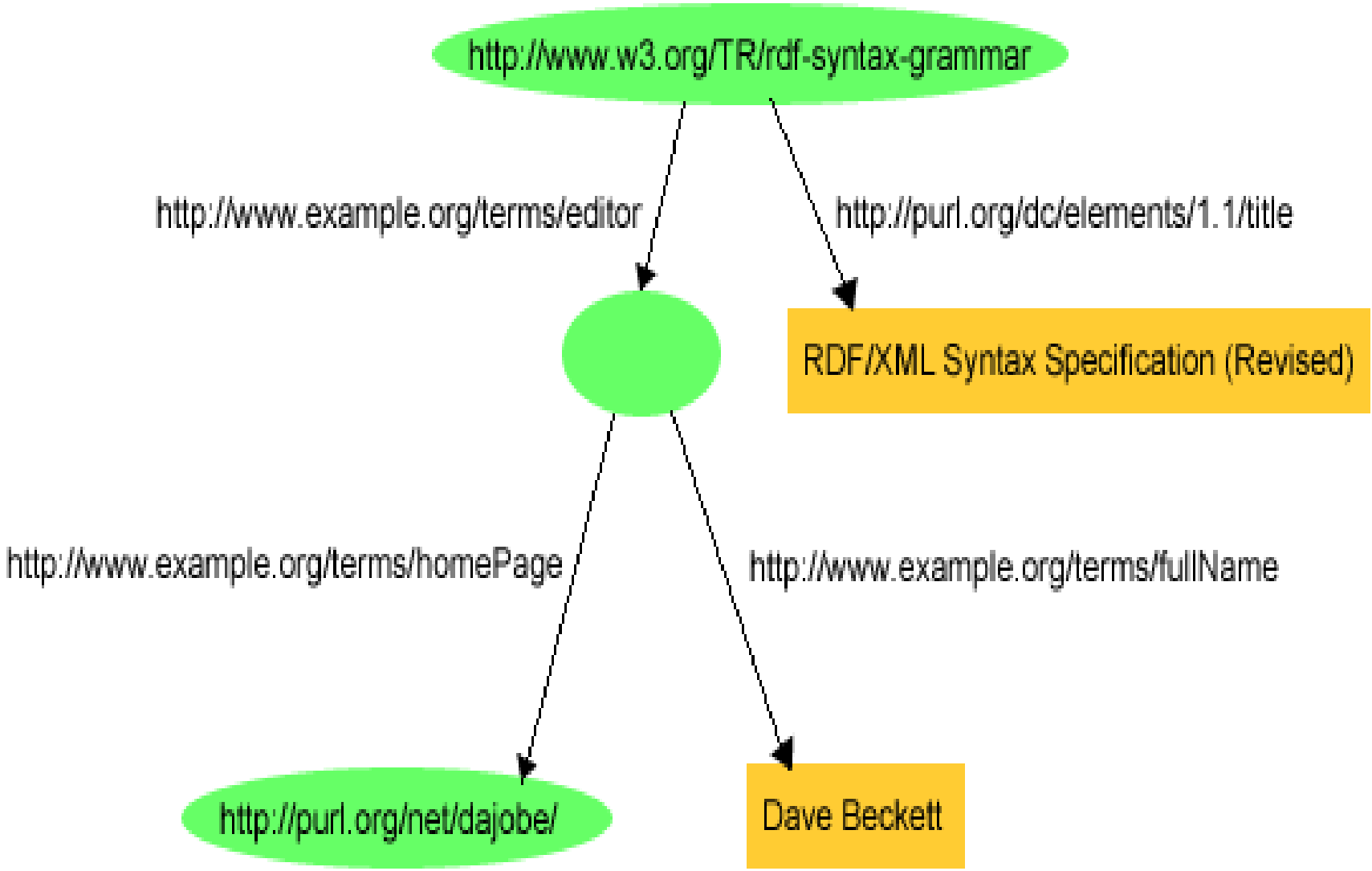
```
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# .
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema# .
@prefix dc: http://purl.org/dc/elements/1.1/ .
@prefix owl: http://www.w3.org/2002/07/owl# .
@prefix ex: http://www.example.org/ .
@prefix xsd: http://www.w3.org/2001/XMLSchema# .
```

  - ...or self-defined prefixes
  - see http://*prefix.cc*
- Example:

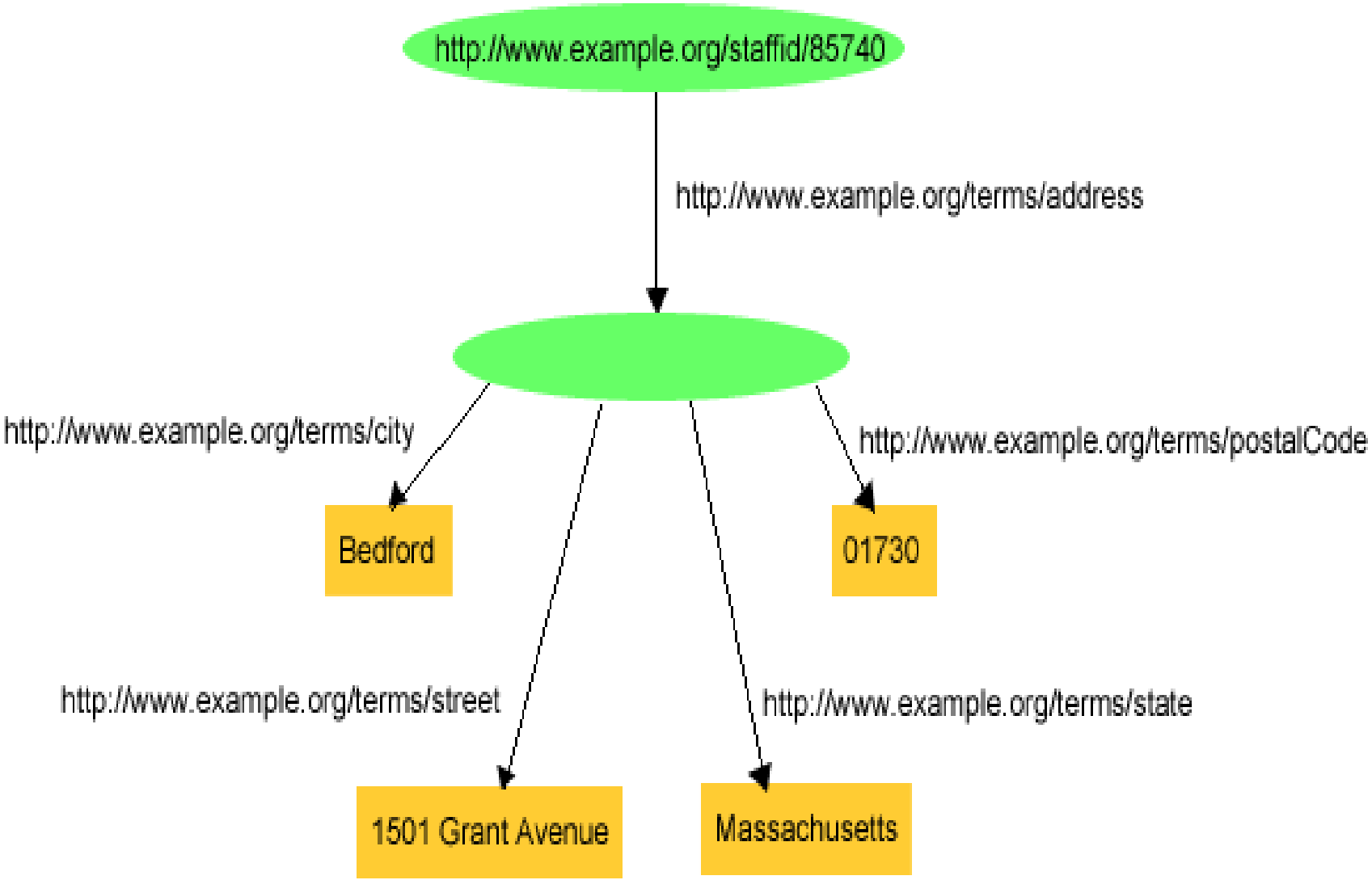
  http://www.w3.org/2001/XMLSchema#string
  can now be written *xsd:string*

Anonymous resources (blank nodes)

- Some nodes have no IRIs:
  - cannot be referenced by other data sets
  - *can* be referenced by graphs in same data set
- Can have a (non-IRI) identifier, but
  - local identifier only meaningful inside the data set
  - *cannot be used to merge nodes from different data sets*
- Uses:
  - you are not sure of the IRI
  - you do not want to provide an IRI
  - logical grouping of related properties
  - *not supported by all RDF technologies*



From http://www.w3.org/TR/rdf-primer/



From http://www.w3.org/TR/rdf-primer/

Turtle syntax for blank nodes

<http://www.w3.org/TR/rdf-syntax-grammar>
    <http://purl.org/dc/elements/1.1/title>
        "RDF/XML Syntax Specification (Revised)" .
<http://www.w3.org/TR/rdf-syntax-grammar>
    <http://www.example.org/terms/editor>
        [] .

[]

Each represents a *different* anon. node...

    <http://www.example.org/terms/homePage>
        <http://purl.org/net/dajobe> .

[]

    <http://www.example.org/terms/fullName>
        "Dave Beckett" .

Turtle syntax for blank nodes

```
<http://www.w3.org/TR/rdf-syntax-grammar>
        <http://purl.org/dc/elements/1.1/title>
                "RDF/XML Syntax Specification (Revised)" .
<http://www.w3.org/TR/rdf-syntax-grammar>
        <http://www.example.org/terms/editor>
                _:blank1 .
_:blank1
        <http://www.example.org/terms/homePage>
                <http://purl.org/net/dajobe> .
_:blank1
        <http://www.example.org/terms/fullName>
                "Dave Beckett" .
```

Turtle syntax for blank nodes

```
<http://www.w3.org/TR/rdf-syntax-grammar>
        <http://purl.org/dc/elements/1.1/title>
                "RDF/XML Syntax Specification (Revised)" ;
        <http://www.example.org/terms/editor>
                _:blank1 .
_:blank1
        <http://www.example.org/terms/homePage>
                <http://purl.org/net/dajobe> ;
        <http://www.example.org/terms/fullName>
                "Dave Beckett" .
```

Turtle syntax for blank nodes

<http://www.w3.org/TR/rdf-syntax-grammar>
    <http://purl.org/dc/elements/1.1/title>
        "RDF/XML Syntax Specification (Revised)" ;
    <http://www.example.org/terms/editor>
        [] .

[]

Each represents a *different* anon. node...

    <http://www.example.org/terms/homePage>
        <http://purl.org/net/dajobe> ;
    <http://www.example.org/terms/fullName>
        "Dave Beckett" .

Turtle syntax for blank nodes

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

Properties (rdf:Property)

- Properties are resources that
  - express a relationship between resources
  - ...or between resources and literal values
- Is used as a *predicate* (or as subject or object)
- Example:
  - *name* is a property in the Dublin Core vocabulary
  - it can also be the subject in RDF statements:
    - dc:name rdf:type rdf:Property .

- *Convention: predicates are written with small initial letters*

Literals (rdf:Literal)

- Untyped (simple) literals: only a character string
  - f eks “29”
  - *can have a language code!*
    “Göteborg”@”se”, “Gothenburg”@”en”
- Typed literals: a string + an IRI (ref)
  - the type is defined of the IRI
  - XML Schema Definition (XSD) language is common
  - two built-in RDF types: rdf:XMLLiteral, rdf:HTML
  - ...but other types can also be used
- Syntax depends on the serialisation, e.g., TURTLE:
  - "29"^^<http://www.w3.org/2001/XMLSchema#integer>
  - or with a prefix: "29"^^<xsd:integer>

XML Schema Definition (XSD)

- XSD types that can be used in RDF:

  xsd:string, xsd:boolean, xsd:decimal, xsd:integer, xsd:float, xsd:double, xsd:dateTime, xsd:dateTimeStamp, xsd:time, xsd:date, xsd:gYearMonth, xsd:gYear, xsd:gMonthDay, xsd:gDay, xsd:gMonth, xsd:duration, xsd:yearMonthDuration, xsd:dayTimeDuration, xsd:hexBinary, xsd:base64Binary, xsd:anyURI, xsd:normalizedString, xsd:token, xsd:language, xsd:NMTOKEN, xsd:Name, xsd:NCName, xsd:positiveInteger, xsd:nonPositiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:nonNegativeInteger, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte

- Not all XML Schema types can be used in RDF:
  - *must be a set of string values*
  - *...that can be mapped into*
  - *...a well-defined value space*

Literals with dimensions

- Some literals have dimensions:
  - a weight can be a flowing number and a weight unit
  - such literals can be represented by anonymous nodes
  - having one *rdf:value* and one *dimension* property
- Example:

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

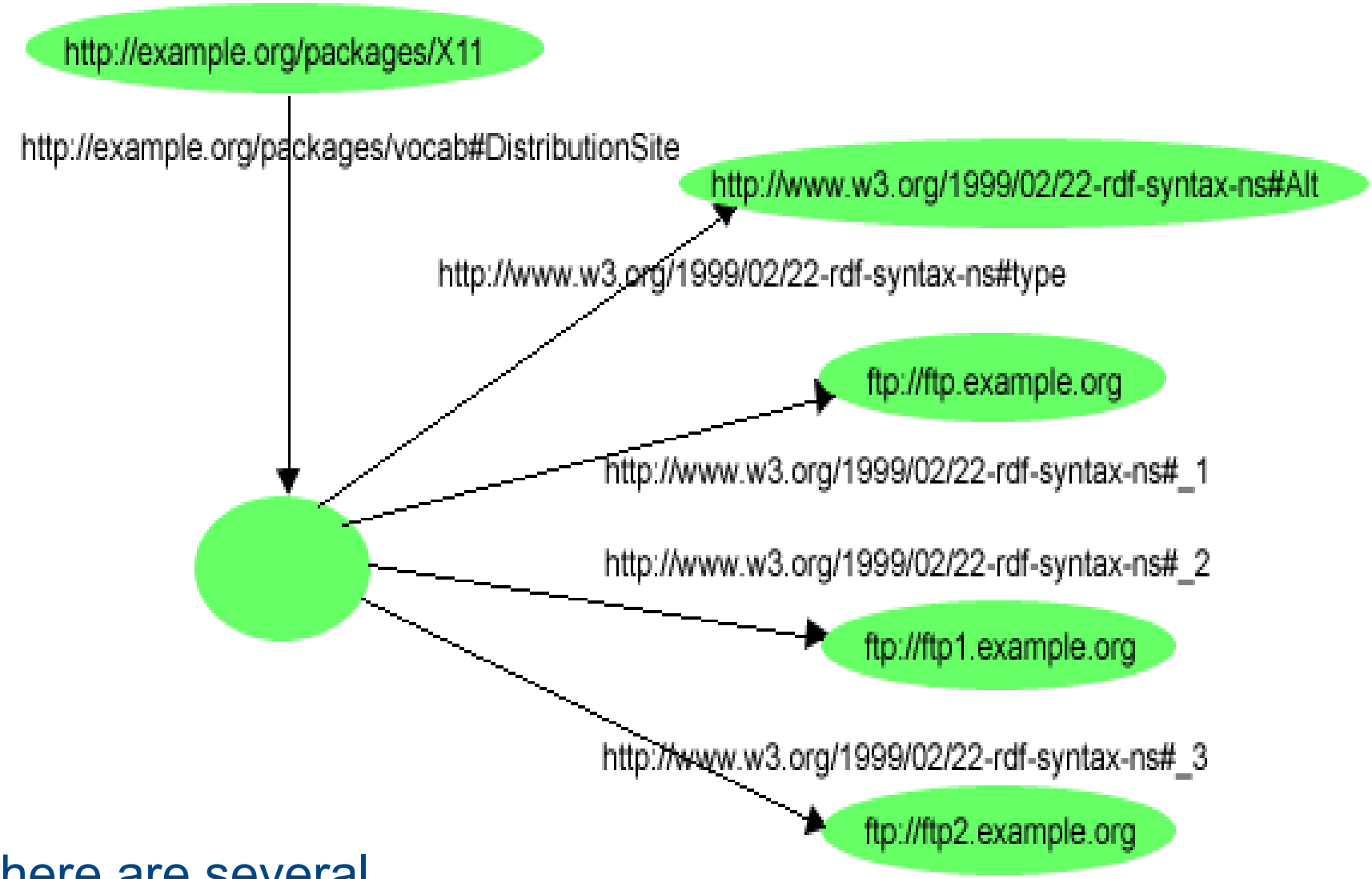
- Using rdf:value:

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

- rdf:value is not necessary here
  - ...offered by RDF as a c*onvention*
- *...distinct vocabularies for units of measure (QUDT, OM...)*

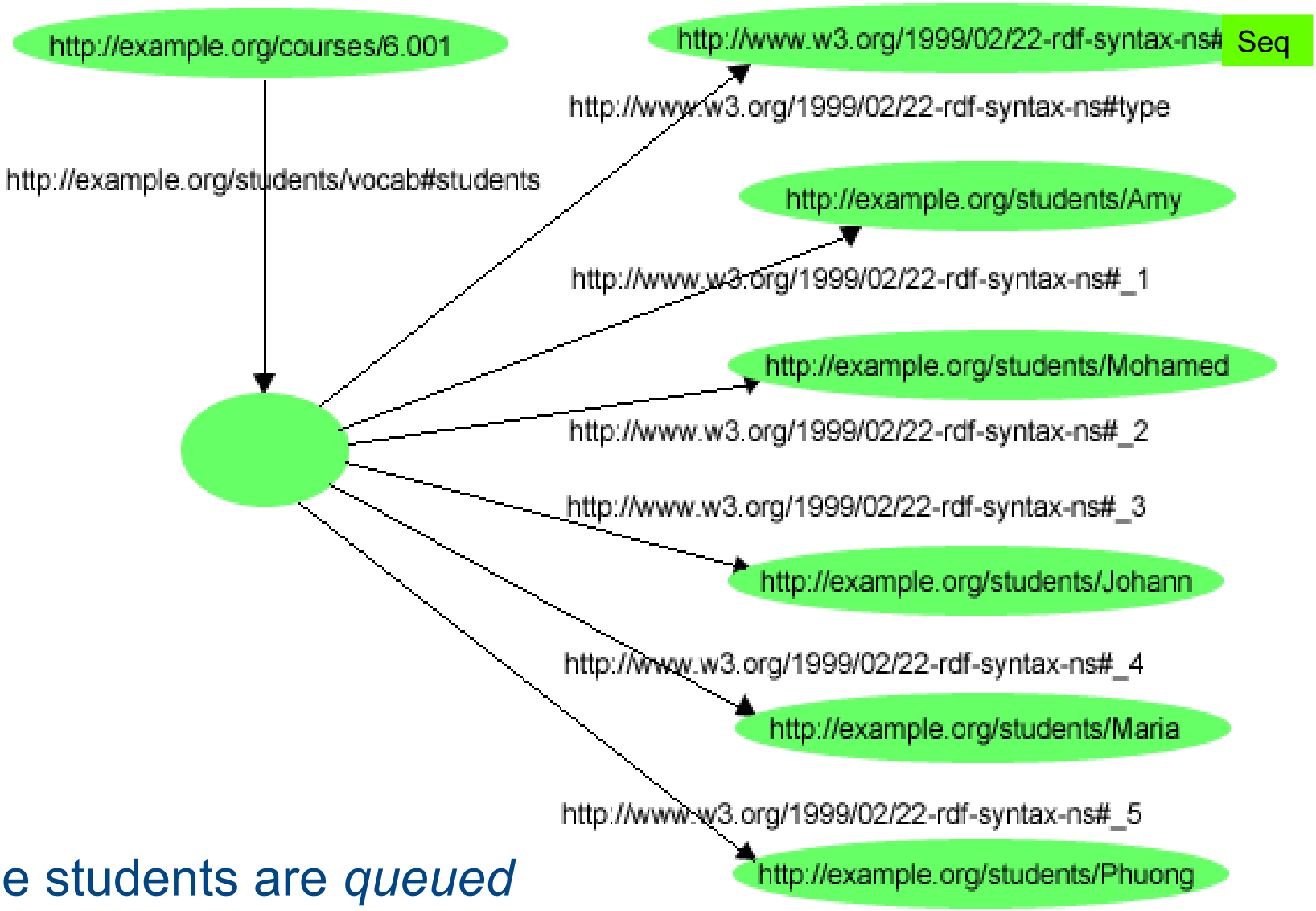
Containers (rdfs:Container)

- Containers can be used when a subject is related in the same way to many RDF nodes that are
  - ordered and/or duplicated
  - (*regular properties* suffice when the RDF nodes are unordered and there are no duplicates)
- Container nodes are often anonymous (can have an IRI)
  - have RDF nodes as members (rdfs:member)
  - have special properties rdf:_1, rdf:_2 etc. to pick out particular members
- rdf:Alt – several alternative resources
- rdf:Seq – lists of RDF nodes, can have duplicates
- rdf:Bag – orderless RDF nodes, can have duplicates



There are several
alternative distribution sites.

From http://www.w3.org/TR/rdf-primer/

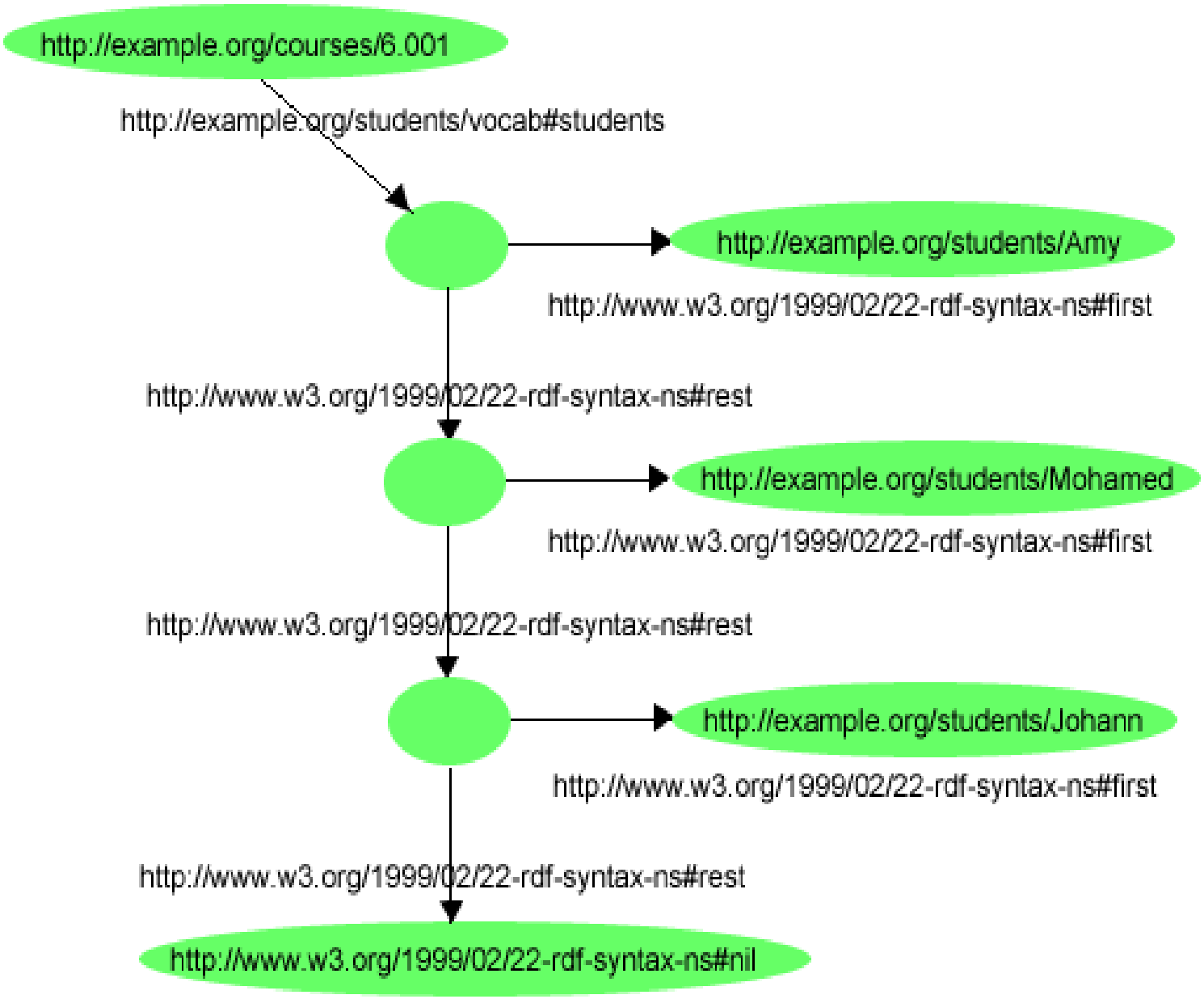


The students are *queued up* in order for the course.

From http://www.w3.org/TR/rdf-primer/

Collections (rdf:List)

- Containers are not closed
  - we *cannot assume it only has the members we know of*
  - others can add more members to the list *without deleting triples* (i.e., *monotonically*)
- Collections (rdf:List-s):
  - can only have the listed members
  - rdf:first gives the first RDF node in the list
  - rdf:rest gives the rest of the list
  - rdf:nil is an empty list



From http://www.w3.org/TR/rdf-primer/

Reified statements (triples)

- Regular statement:

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

- Reified statement (*reification quad*):

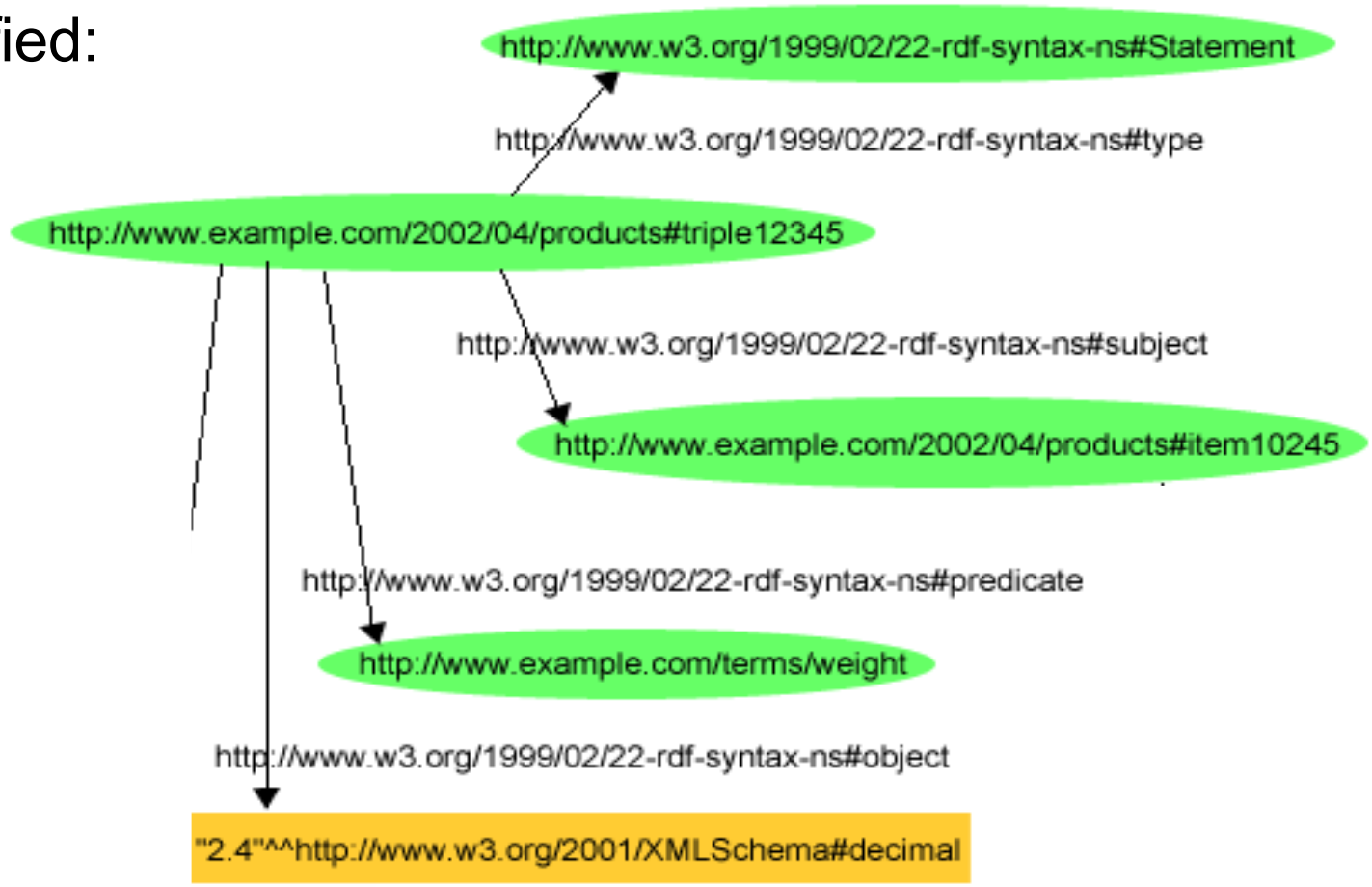
  exproducts:triple12345 rdf:type rdf:Statement .
  exproducts:triple12345 rdf:subject exproducts:item10245 .
  exproducts:triple12345 rdf:predicate exterms:weight .
  exproducts:triple12345 rdf:object "2.4"^^xsd:decimal .

- Reification *gives the triple an identifier (IRI)*
- Reification "unpacks" a triple into four new ones:
  - new type: rdf:Statement
  - new properties: rdf:subject, rdf:predicate, rdf:object
- We can now make *statements about statements*:
  - *"<Trippel-X> is valid from <dato> until <dato>."*
  - *"<dbpedia:Wikipedia> claims that <trippel-Y>."*

Non-reified:

From http://www.w3.org/TR/rdf-primer/

Reified:



From http://www.w3.org/TR/rdf-primer/

With other triples:

From http://www.w3.org/TR/rdf-primer/

RDF API

Possible plan for the lab

- Creating a Java project in Eclipse
- Creating a model (graph)
- Adding statements to a model (triple)
- Serialising (writing) and parsing (reading) RDF
- Listing statements in a model
- Prefix mapping
- Using vocabularies
- Using schemagen

Focus on *Jena*...
The Java-part does not have to be so hard!

(Expert in Java already? Try to do it in Scala!)

Creating models and statements

- Creation:
  - Model model = ModelFactory.createDefaultModel();
  - Resource res = model.createResource(*resIRI*);
  - Property prop = model.createProperty(*propIRI*);
  - Literal literal = model.createLiteral(...);
- Add triple:
  - res.addProperty(prop, *objectRes*);
  - res.addProperty(prop, *literalString*);
  - res.addLiteral(prop, *literalValue*);

Serialising and parsing

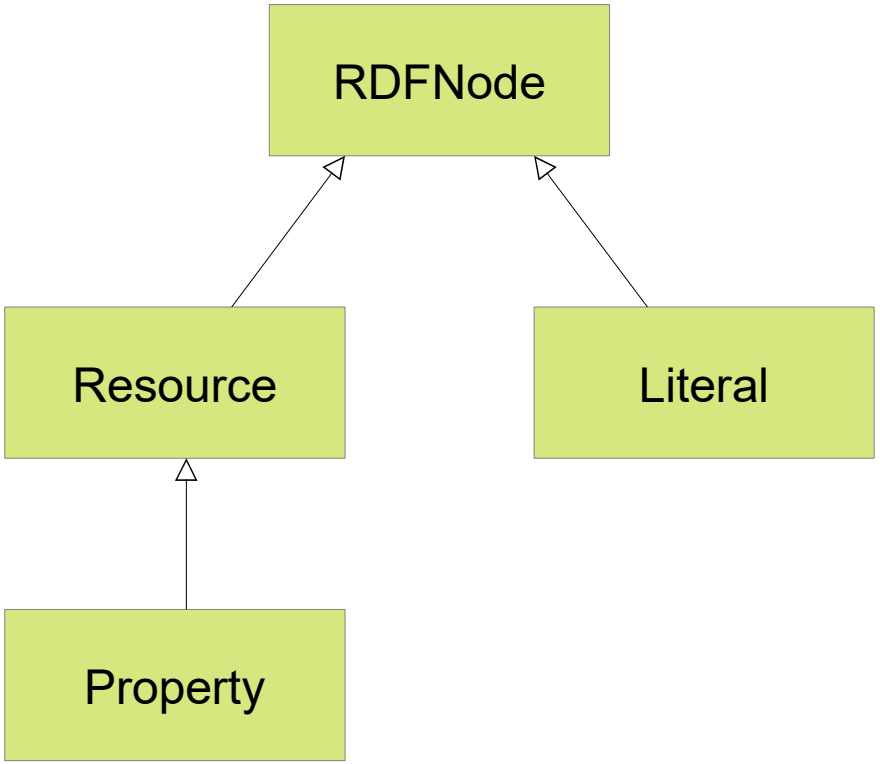
- Serialising:
  - model.write(System.out);
  - model.write(System.out, “N-TRIPLE”);
  - model.write(new FileOutputStream(*pathStr*), "TURTLE");

  ...needs enclosing try { ... } catch (Exception e) { } block
- Parsing:
  - model.read(new FileInputStreat(“file*:pathStr”), baseIRIstr*, "TURTLE");

  ...needs enclosing try { ... } catch (Exception e) { } block

Listing statements

- Retrieving statements (triples):
  - StmtIterator stmts = model.listStatements();
  - stmts = model.listStatements(*subj, pred, obj*);
- Iterating:
  - for (Statement stmt : stmts.toList()) {
    ...*do something with stmt...*
    }
  - Resource subj = stmt.getSubject();
  - Property pred = stmt.getPredicate();
  - RDFType obj = stmt.getObject();


RDFNode
Resource
Literal
Property

Deleting statements and closing models

- Remove triple(s):
  - res.removeAll(prop);
  - res.removeProperties();
- Close model when finished:
  - model.close();

Vocabularies

- Predefined resources, properties (and some other fields) for common vocabulary terms (IRIs):
  - VCARD.NAME, VCARD.EMAIL, VCARD.PHOTO
  - DC_11.title, DC_11.subject, DC_11.creator
  - XSD.xstring, XSD.xint, XSD.date, XSD.anyURI
  - RDF.type, RDF.Property
  - RDFS.Resource, RDFS.Class, RDFS.Literal, RDFS.Datatype, RDFS.subClassOF, RDFS.subPropertyOf
  - OWL.sameAs, OWL.differentFrom, OWL.equivalentClass, OWL.disjointWith, OWL.equivalentProperty
- Jenas schemagen-verktøy kan brukes til å generere flere vokabularklasser

Prefix mapping

- Qualified names (qnames):
  - IRI written as prefix + identifier
  - http://purl.org/dc/elements/1.1/author
    - written as dc:author
  - http://www.w3.org/2001/XMLSchema#string
    - written as xsd:string
- model.setNsPrefix(*prefixStr*, *iriStr*);
- model.getNsPrefixIRI(*prefixStr*); *// returnerer iriStr*
- model.getNsIRIPrefix(*iriStr*); *// returnerer prefixStr*
- model.removeNsPrefix(*prefixStr*);

Semantic data sets and vocabularies

(quick overview to help you find project ideas!)

Quick overview of...

- *Semantic vocabularies*
  - semantic resources (in RDFS, OWL...) that define:
    - standard IRIs for *types of resources*
    - standard IRIs for *properties*
    - standard types for *literals*
- *Semantic data sets*
  - free/open RDF-graphs that define:
    - standard IRIs for *individual resources*
    - perhaps also also their own vocabularies

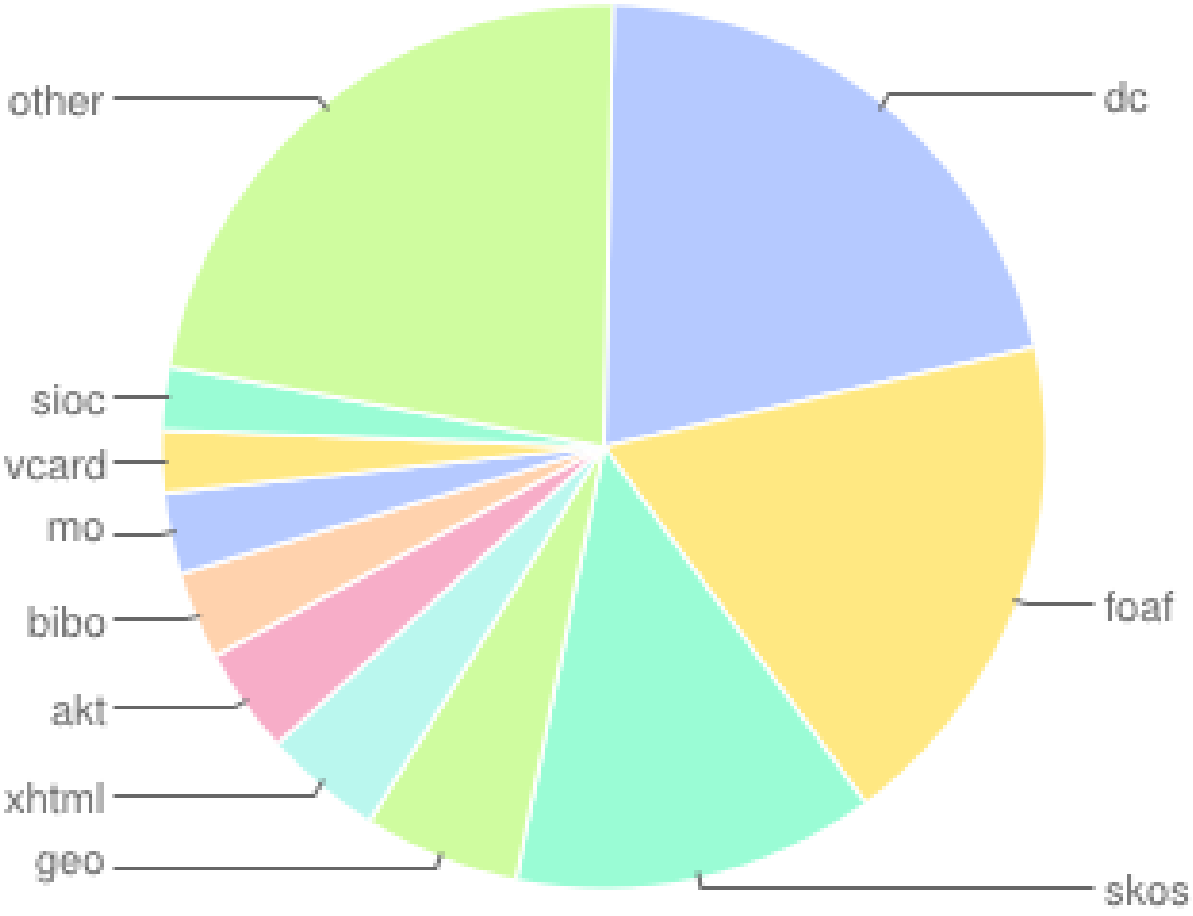
Places to start

- Open and semantic:
  - open semantic data sets: *http://lod-cloud.net*
  - vocabularies: *http://lov.okfn.org/dataset/lov/*
  - statistics and overviews: *http://stats.lod2.eu/*
- Open data in general:
  - internationally: *http://datahub.io* or *http://ckan.net*
  - Norge: *data.norge.no*
  - EU: *https://open-data.europa.eu*
  - Storbritannia: *data.gov.uk*
  - USA: *data.gov*

The LOD cloud...

- *http://www.lod-cloud.net/*
  - statistics at www.lod-cloud/net/state
  - 570 data sources (LOD-cloud, 2014)
  - based on data from DataHub (+ some crawling)
    - datahub.io or ckan.net
    - an open data portal
    - not necessarily semantic
    - LOD cloud group: www.ckan.net/group/lodcloud
  - ...also based on LOD crawling
- *http://stats.lod2.eu/ is more recent (& less restrictive?!)*
  - 154 000M triples from 2005 data sets

Important vocabularies



Pie chart from www.lod-cloud.net/stats