cours 1 +2

# RDF
# Resource Description Framework

Gilles Falquet

Semantic Web Technologies

2021

# Contents

- The RDF graph model
- Blank nodes
- Representing collections
- Reification
- Concrete syntaxes

# A Graph Model for KR

RDF graphs express knowledge about resources

> a resource is anything that can be identified by a URI (a web page, a person, a country, an abstraction, ...)

The basic unit of knowledge is the triple


It represents the fact that a relation (predicate) holds between the subject and the object.

The canton of Vaud is a neighbour of the canton of Geneva

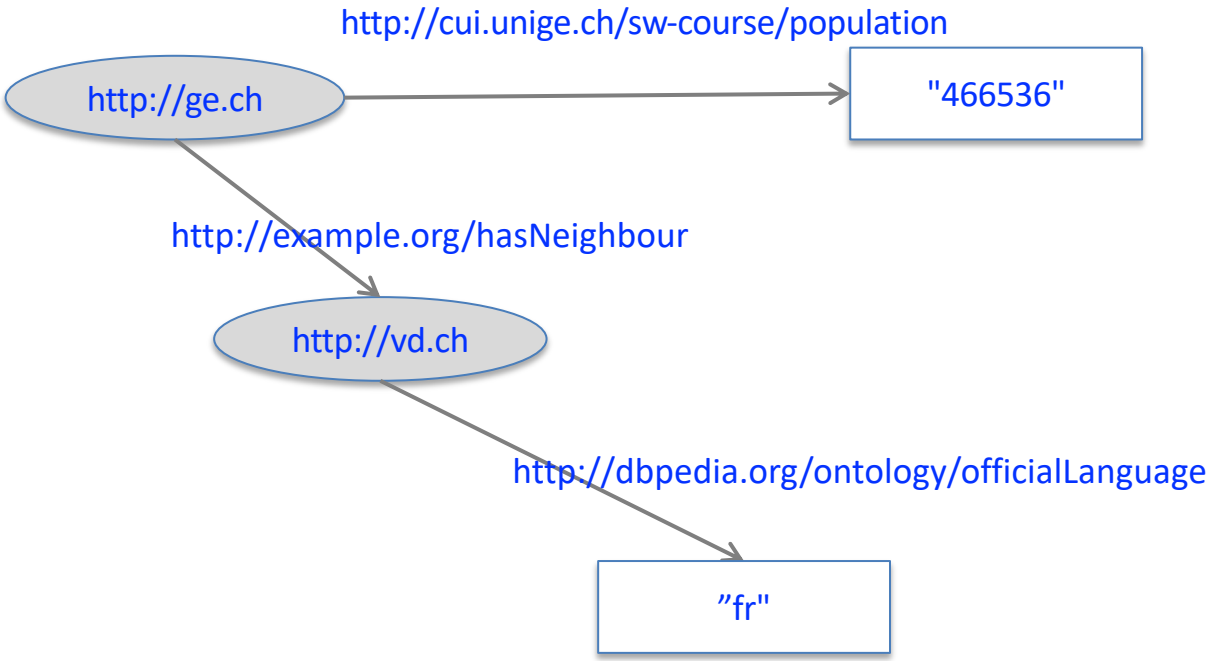
http://ge.ch http://example.org/hasNeighbour http://vd.ch

The official language of Vaud is French

http://vd.ch http://dbpedia.org/ontology/officialLanguage “fr”

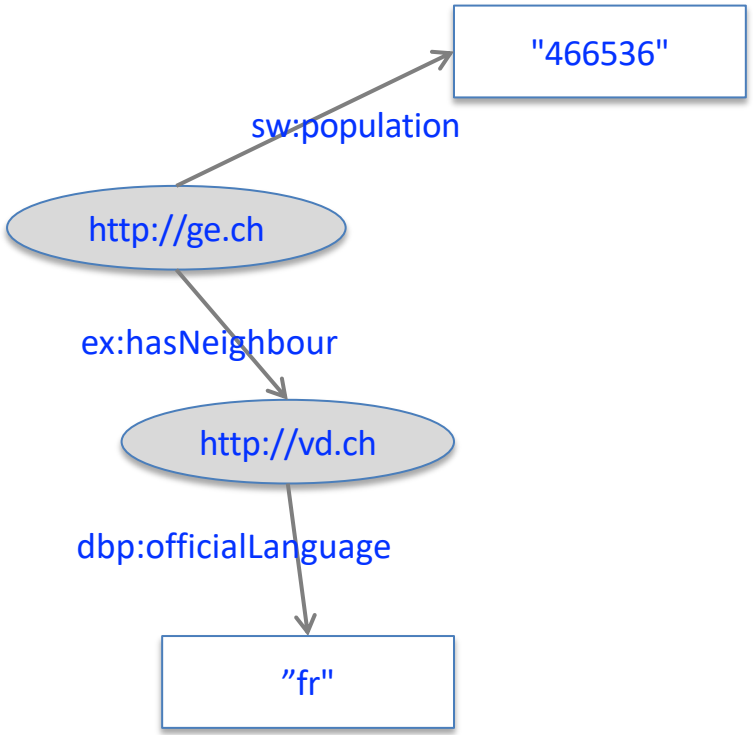
The object may be a literal value

# The triples form the edges of a knowledge graph



# Use of prefixes to simplify the expression of the graphs

sw ⇒ http://cui.unige.ch/sw-course/
dbp ⇒ http://dbpedia.org/ontology/
ex ⇒ http://example.org/



# Literals

A lexical form that identifies a value in a value space

strings

"value"

string in a specific language

"value"@language

typed value

"value"^^type

# Examples

XML Standard Types

prefix xsd: http://www.w3.org/2001/XMLSchema#

"Scrabble"
"vi povas legi ĉi tiun tekston"@eo
"567"^^xsd:number
"true"^^xsd:boolean
"2002-10-10T12:00:00+02:00"^^xsd:dateTime

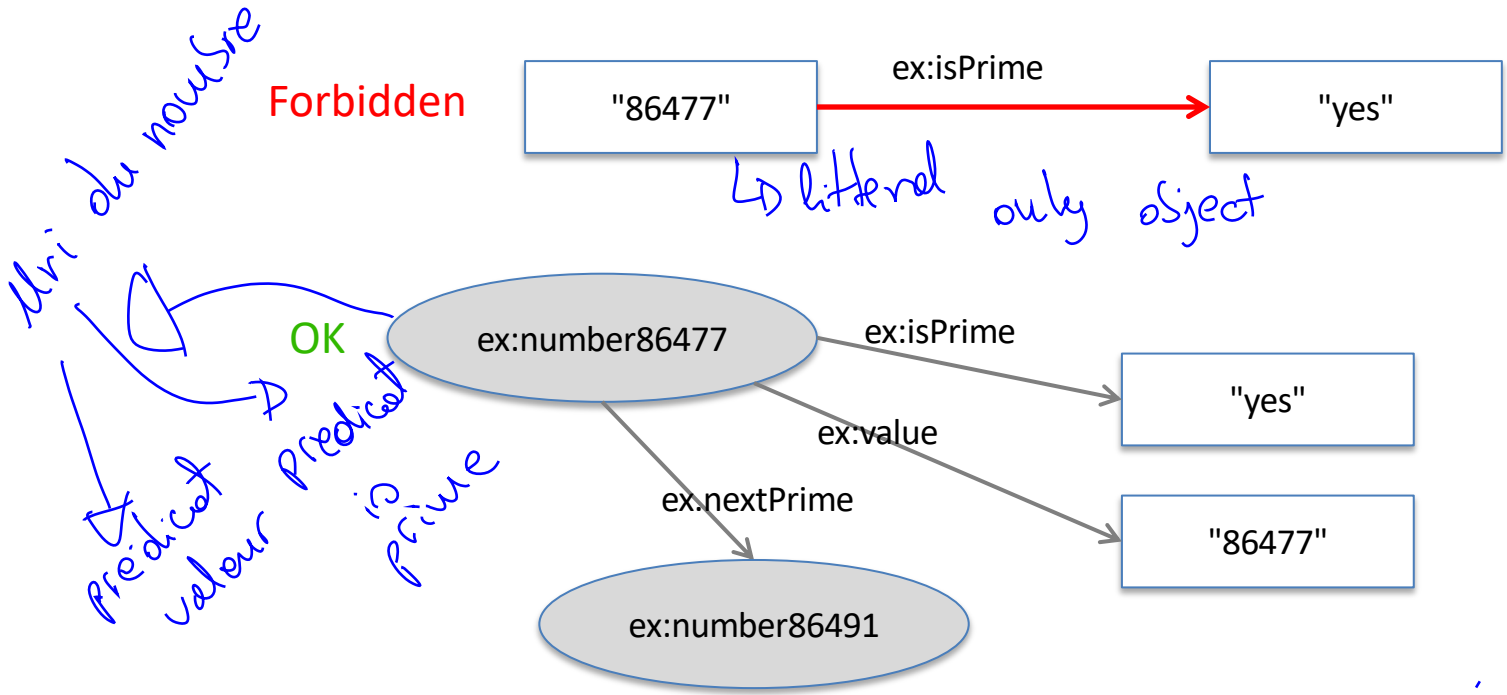
XML builtin datatypes are of common use, but not mandatory

prefix my ⇒ http://cui.unige.ch/TypeSystem#

"4.5+3i+2j-5k"^^my:quaternion

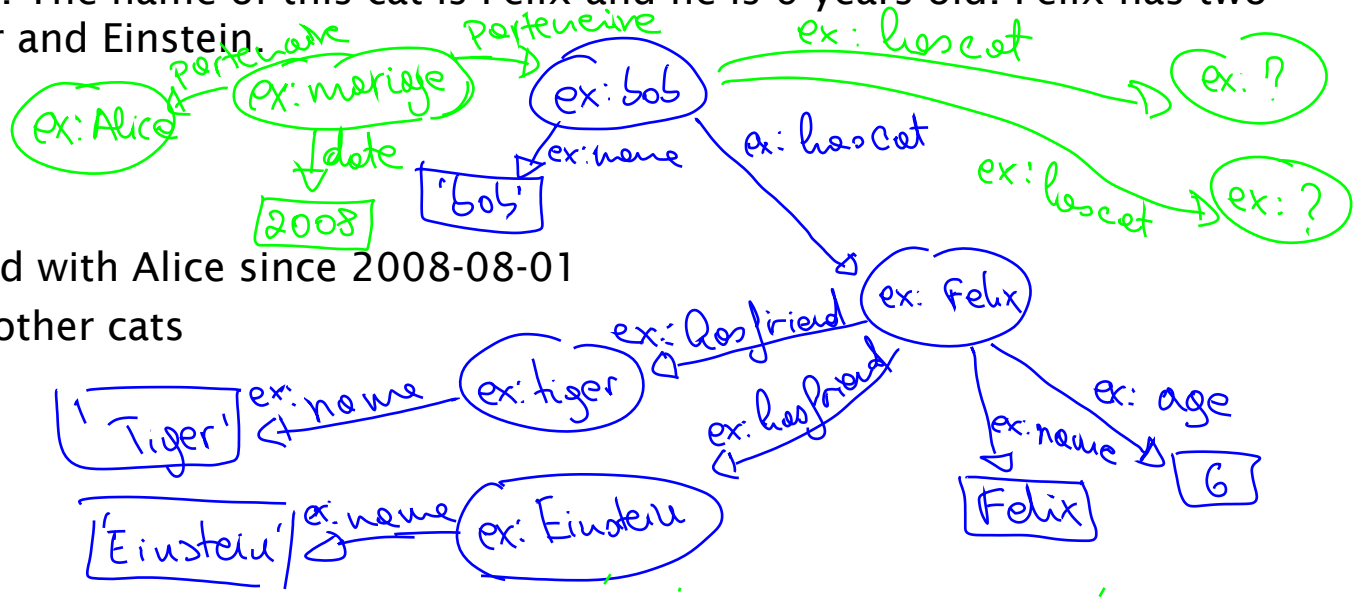
# Restriction on literal nodes

**Remark.** A literal may not be the subject of a triple (values cannot be described, they are supposed to be known)



# Exercises

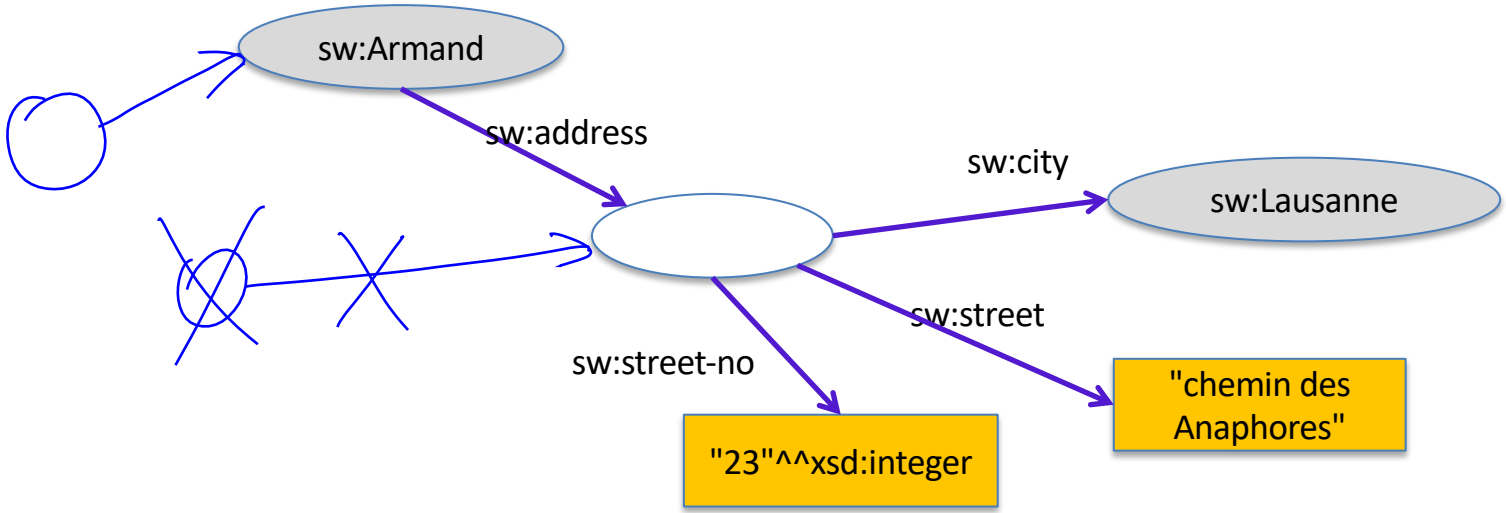
1. Draw an RDF graph that represents the following situation
   - Bob has a cat. The name of this cat is Felix and he is 6 years old. Felix has two friends: Tiger and Einstein.
2. Add the facts
   - Bob is married with Alice since 2008-08-01
   - Bob has two other cats



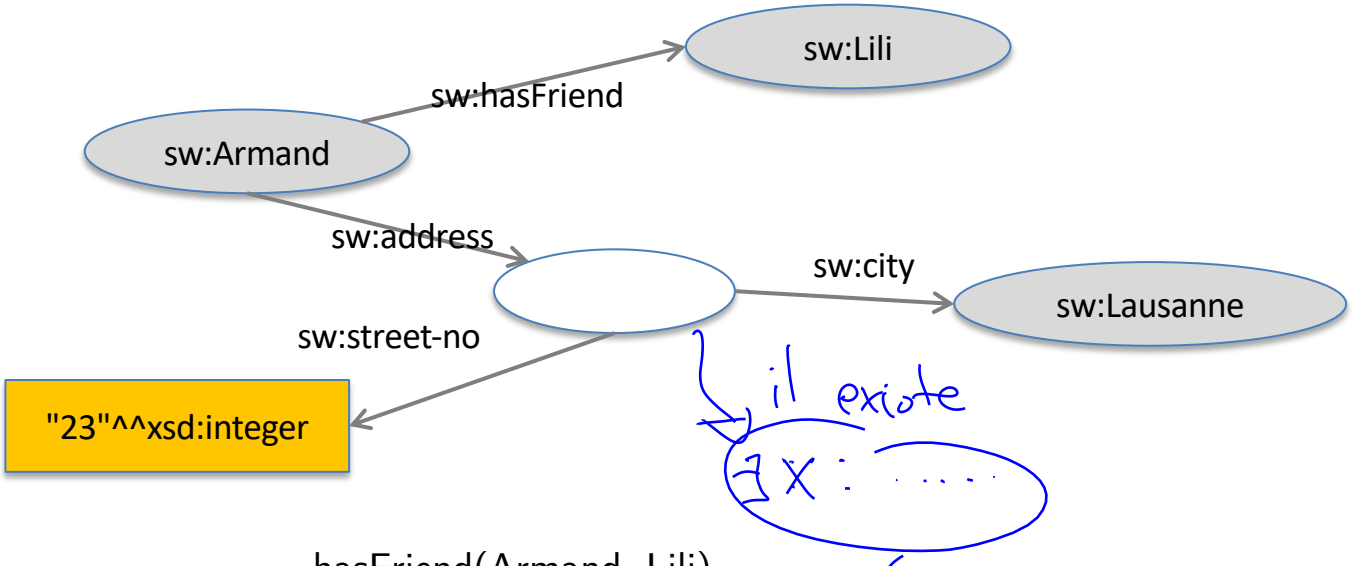
# Blank nodes

- Nodes that are anonymous, not identified by a URI
- Only locally identified

"The address of Armand is 23 chemin des Anaphores, Lausanne"



# blank nodes correspond to existentially quantified variables



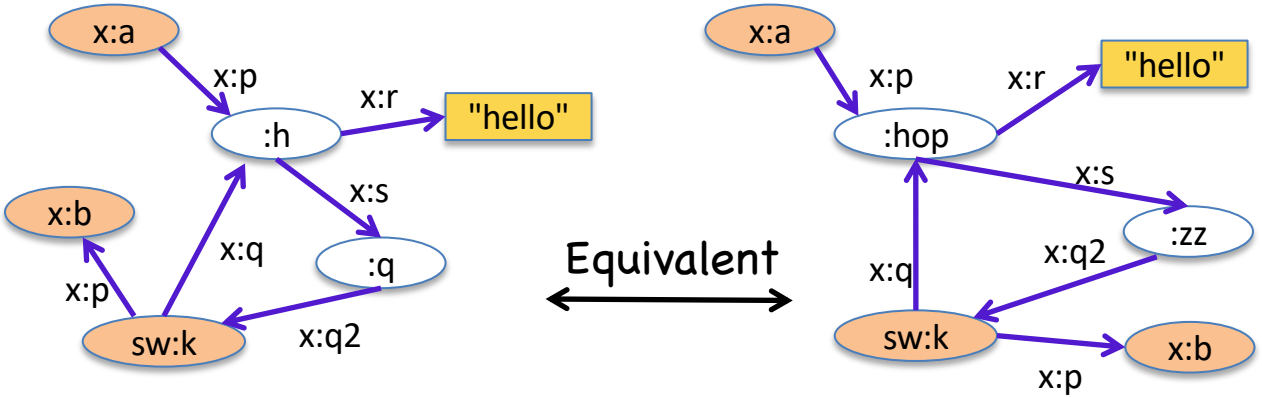

hasFriend(Armand, Lili)

∧

$\exists x$ : address(Armand, $x$) ∧ city($x$, Lausanne) ∧ street-no($x$, 23)

# Graph equivalence

- The internal identifiers of blank node are interchangeable
- Two RDF graphs have the same meaning if their only differences are the blank node identifiers.



# Graph equivalence

**The official definition**

Two RDF graphs *G* and *G'* are equivalent if there is a bijection *M* between the sets of nodes of the two graphs, such that:

- *M* maps blank nodes to blank nodes.
- *M*(*lit*)=*lit* for all RDF literals *lit* which are nodes of *G*.
- *M*(*uri*)=*uri* for all RDF URI references *uri* which are nodes of *G*.
- The triple (*s*, *p*, *o*) is in *G* iff (*M*(*s*), *p*, *M*(*o*)) is in *G'*

In fact, *M* shows how each blank node in *G* can be replaced with a new blank node to obtain *G'*.

# RDF standard vocabulary

A standard vocabulary for defining

- resource typing
- data structures (containers and collections)
- RDF graph schemas
  - resource classification (schemas)
  - constraints on properties

This vocabulary has URIs of the form

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

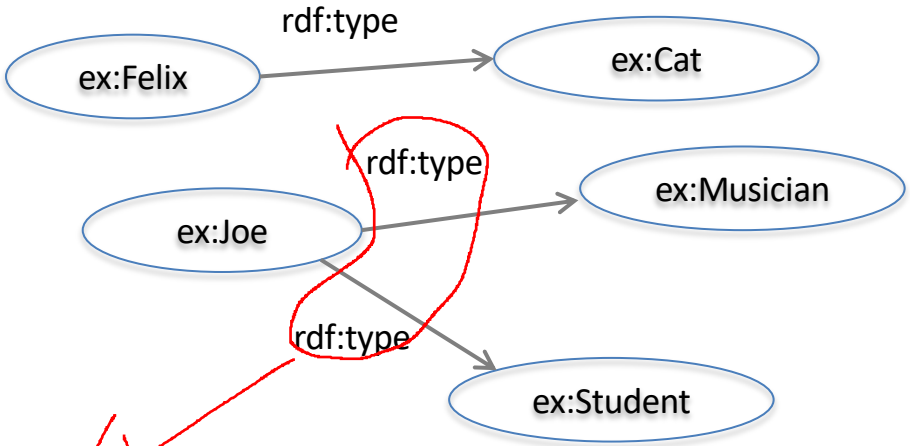
the usual prefix definition is

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

# rdf:type

Assign a type to a resource

- Felix *is a* cat and Joe *is a* musician and a student




peut avoir 2 classes

# Containers

peu utilisé → skip

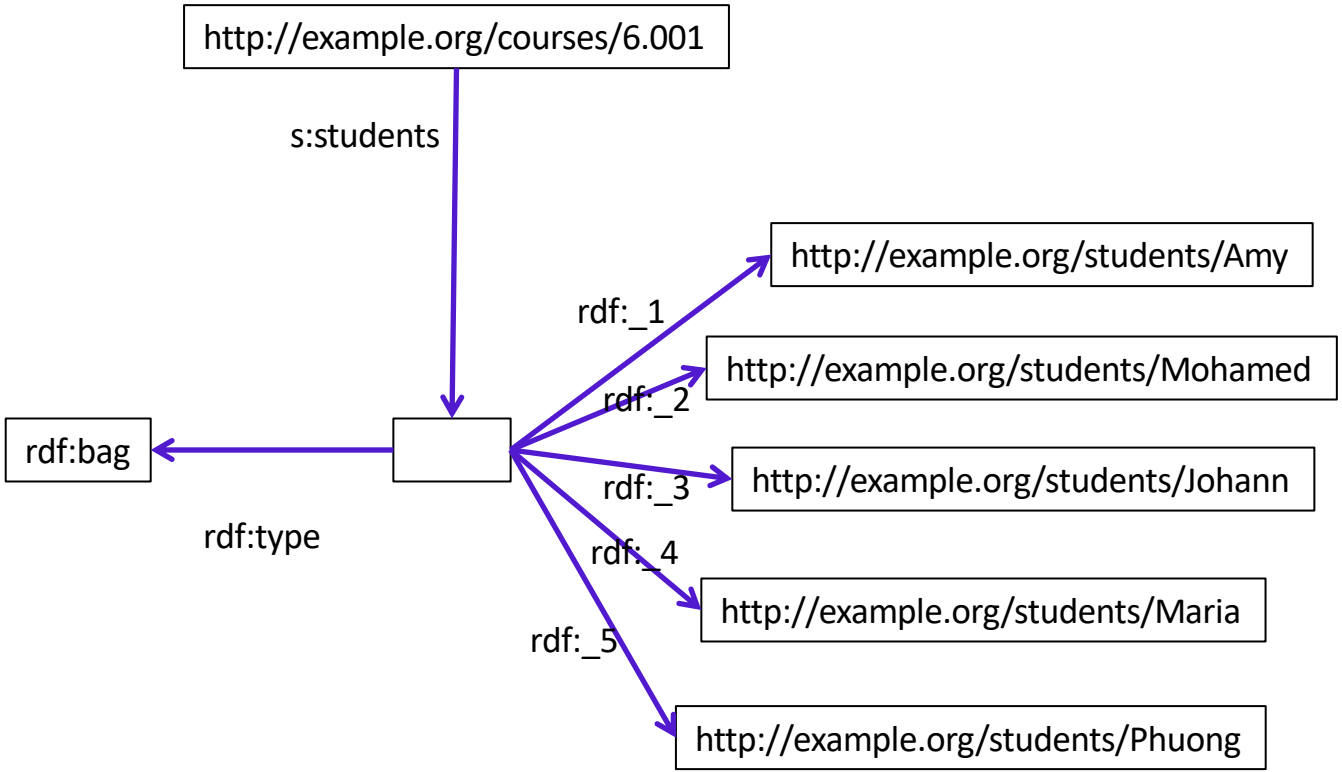
To consider a group of resources as a whole

- assign global properties to the group

Three types of containers

- rdf:Bag (a set with repetitions)
- rdf:Seq (an ordered set)
- rdf:Alt (represents choices)

Properties rdf:_1, rdf:_2, rdf:_3, ... to link a container with its first, second, third, ... member.


http://example.org/courses/6.001
s:students
http://example.org/students/Amy
rdf:_1
http://example.org/students/Mohamed
rdf:_2
rdf:bag
rdf:_3
http://example.org/students/Johann
rdf:type
rdf:_4
http://example.org/students/Maria
rdf:_5
http://example.org/students/Phuong

# Remarks

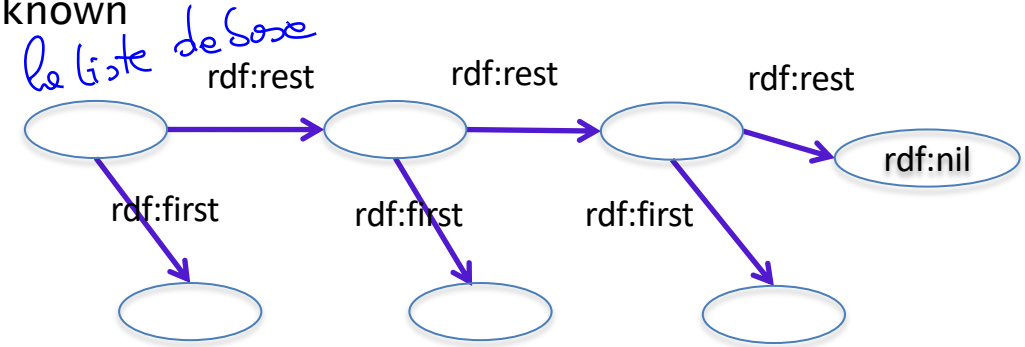
- Bag, Seq, Alt are indications about the intended meaning
- There is not specific way to "close" a container, i.e. to say that is doesn't have any other member.
  - the Bag of students in the previous example may have more than 5 members, in reality

# Build lists with rdf:first, rdf:rest

Closed collections: all the members are known

A collection is made of
- a first element (any resource)
- a rest, which is a list

rdf:nil is the empty list



# Exercise

Represent the following facts

- p1 and p2 are political parties
- c1, c2, c3, c4 were candidates for p1
- d1, d2, d3 were candidates for p2
- c3, c1 have been elected (in this order) for p1
- no one from p2 has been elected
- elected candidates have become members of the parliament (MP)

# Reification

How to represent statements about statements?

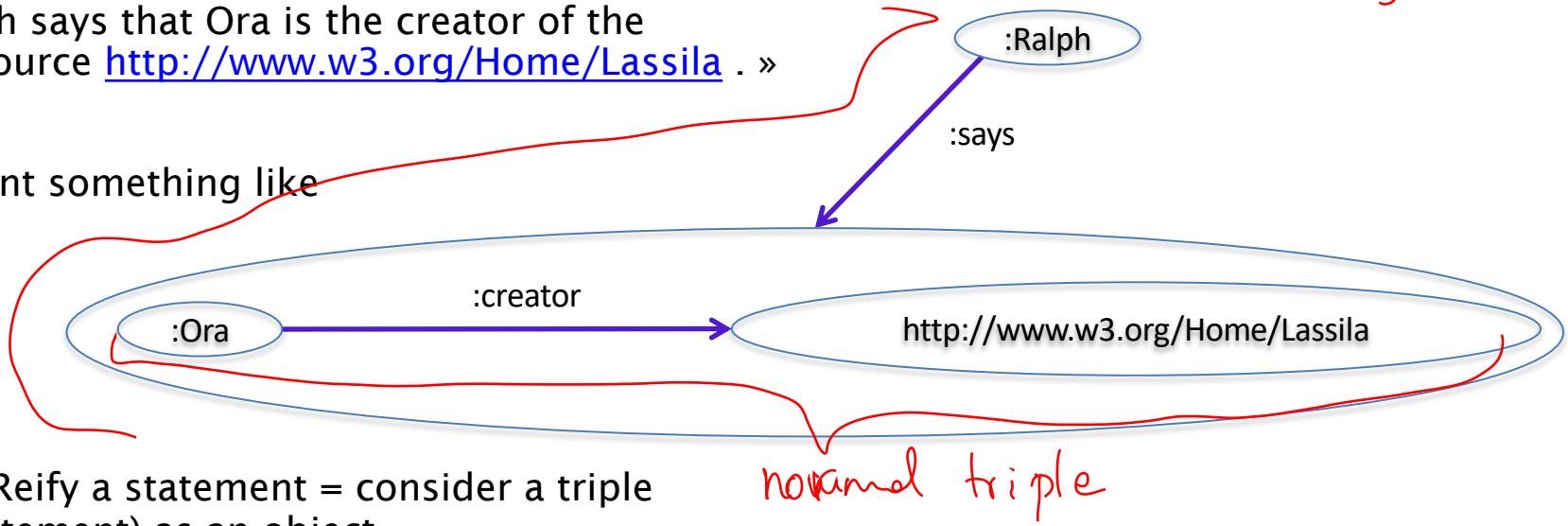
« Ralph Swick says that Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila . »

« Albert says that document 345 confirms that Ralph Swick says that Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila ».

# Reification

« Ralph says that Ora is the creator of the resource http://www.w3.org/Home/Lassila . »

We want something like




Goal: Reify a statement = consider a triple (statement) as an object

Remark. *res* = *thing* in Latin

# Reification

Ralph Swick says that Ora Lassila is the creator of the resource
http://www.w3.org/Home/Lassila . »

The rdf standard vocabulary contains a reification vocabulary.

rdf:Statement .

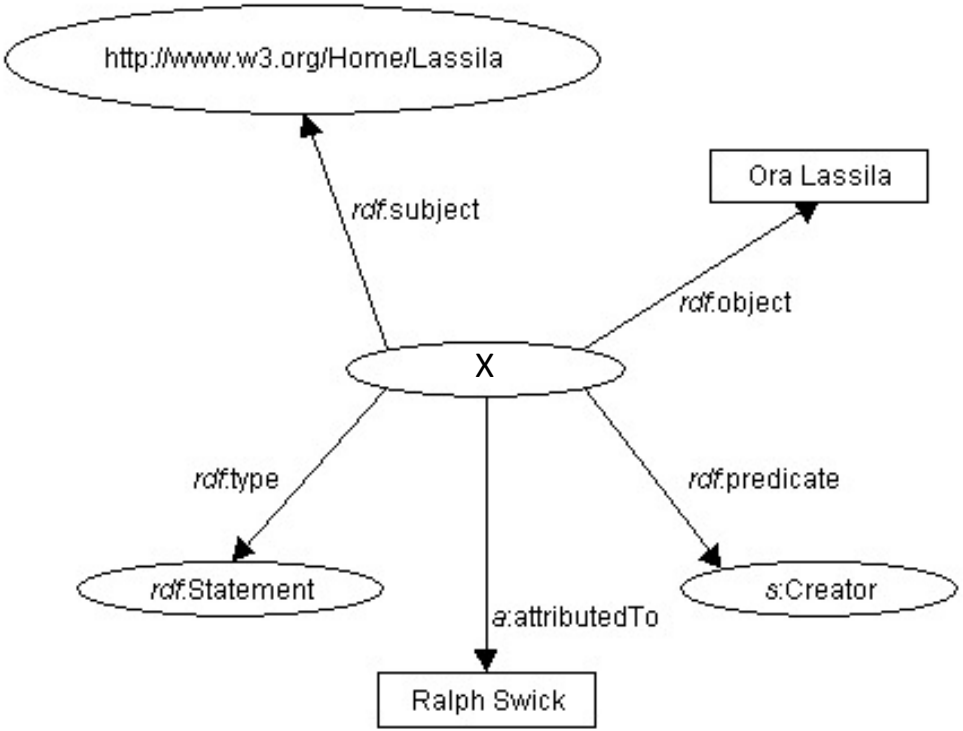
rdf:predicate

rdf:subject

rdf:object

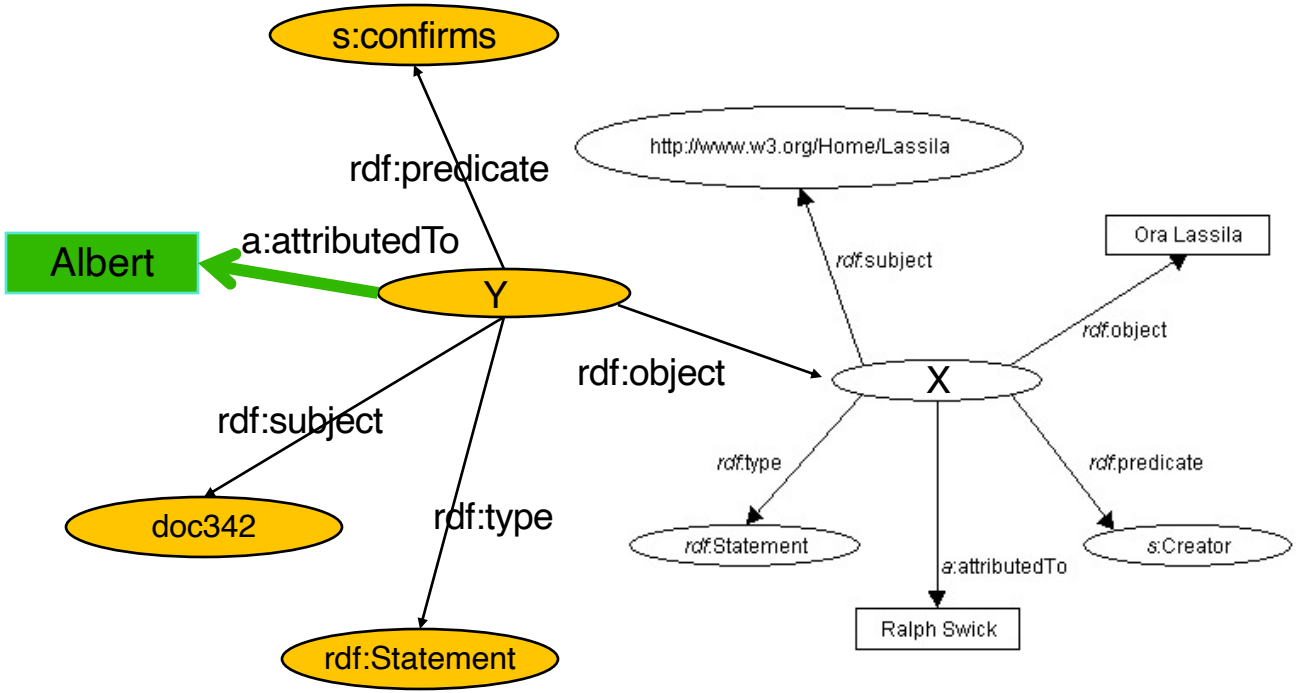
# Reified statement

la solution pour slide 23



# A statement about a statement about a statement

Albert says that doc342 confirms that Ralph Swick says that Ora Lassila is the creator of the resource.



# Practical syntax for RDF

How to represent a RDF graph with characters (in a text file)

RDF data can be expressed with different notations

- XML (for machine interchange)
- N3 and Turtle (human readable)
- JSON-LD

# XML Syntax

Principle: there are alternating node and property elements

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:sw="http://cui.unige.ch/sw-course/">
<rdf:Description rdf:about="http://cui.unige.ch/sw-course/Geneva">
   <sw:population>466536</sw:population>
   <sw:neighbour>
      <rdf:Description
         rdf:about="http://cui.unige.ch/sw-course/Vaud">
      </rdf:Description>
   </sw:neighbour>
</rdf:Description>
 ...
</rdf:RDF>
```

# N3 notation

An N3 file has

1. prefix definitions
2. triples

```
@prefix sw: <http://cui.unige.ch/sw-course/> .
@prefix xsd: <http://www.w3c.org/2001/XMLSchema#> .

sw:Geneva sw:population "466536"^^xsd:integer .
sw:Geneva sw:neighbour sw:Vaud .
sw:Vaud sw:official-language <http://id.loc.gov/vocabulary/iso639-2/fra> .
```

# Turtle: Abbreviations

subject $pred_1$ $obj_1$ ; $pred_2$ $obj_2$ ; ... ; $pred_n$ $obj_n$ .

for

subject $pred_1$ $obj_1$ . subject $pred_2$ $obj_2$ . ... . subject $pred_n$ $obj_n$ .

```
sw:Geneva
   sw:population "466536"^^xsd:integer ;
   sw:neighbour sw:Vaud .
```

# Turtle: Abbreviations

subject predicate $obj_1$ , $obj_2$ , ... , $obj_n$ .

for

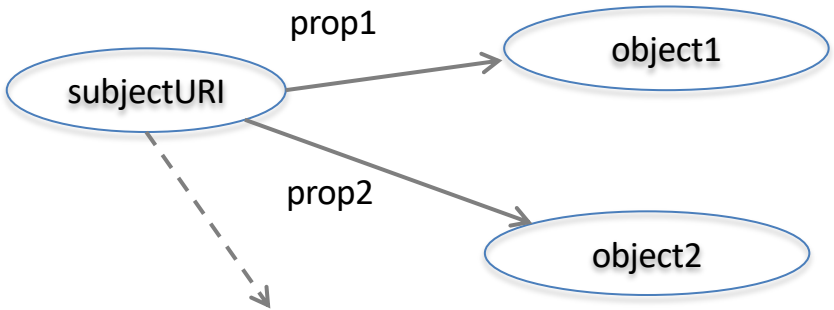
subject predicate $obj_1$ . subject predicate $obj_2$ . ... . subject predicate $obj_n$ .

```
sw:Vaud sw:neighbour
  sw:Geneva , sw:Fribourg , sw:Valais ,
  sw:Neuchatel , sw:Bern .
```

# JSON-LD

Expression in JSON

```
{"@id" : "subjectURI",
  "prop1" : object1 ,
  "prop2" : object2,
  ...
}
```



# JSON-LD

```
{
“graph”: [
  { "@id" : "http://ge.ch",
    "http://cui.unige.ch/ex#neighbour" : {"@id" : "http://vd.ch"},
    "http://cui.unige.ch/ex#population" : "466536"
  },
  { "@id" : "http://vd.ch",
    "http://cui.unige.ch/ex#official-language" : ”fr"
  }]
}
```

# JSON-LD – with typed values

```
{
“graph”: [
{ "@id" : "http://ge.ch",
  "http://cui.unige.ch/ex#neighbour" : {"@id" : "http://vd.ch"},
  "http://cui.unige.ch/ex#population" : {
          "@type" : "http://www.w3.org/2001/XMLSchema#integer",
          "@value" : "466536"
           }
},
{ "@id" : "http://vd.",
  "http://cui.unige.ch/ex#official-language" : "fr"}
}]
}
```

# JSON-LD – with context

```
{
"@context": {"@vocab" : "http://cui.unige.ch/ex#"},
“graph”: [
{ "@id" : ”http://ge.ch",
  "neighbour" : {"@id" : ”http://vd.ch"},
  "population" : {
          "@type" : "http://www.w3.org/2001/XMLSchema#integer",
          "@value" : "466536"
           }
},
{ "@id" : ”http://vd.ch",
  "official-language" : {"@id" : "fra"}
}]
}
```

# Blank nodes in Turtle, with the _: prefix

```
@prefix sw: <http://cui.unige.ch/sw-course/> .
@prefix xsd: <http://www.w3c.org/2001/XMLSchema#> .

sw:Armand sw:address _:aa .
_:aa sw:street "chemin des Anaphores" .
_:aa sw:street-no "23"^^xsd:integer .
_:aa sw:city sw:Lausanne .
```

_:aa acts like an internal variable, within the RDF file/graph. It is invisible from the outside (no URI).

Possible abbreviation: [ blank node description ]

```
sw:Armand sw:address
  [sw:street "chemin des Anaphores" ;
   sw:street-no "23"^^xsd:integer ;
   sw:city sw:Lausanne] .
```

# In JSON-LD: @id value of the form _:…

```
{"@context" : {
  "sw" : "http://cui.unige.ch/sw-course/",
  "xsd" : "http://www.w3.org/2001/XMLSchema#"
},
"@id": "http://example.org/graphs/73",
"@graph": [
  {"@id" : "sw:Armand",
   "sw:address" : {"@id": "_:aa"}},

  {"@id": "_:aa" ,
    "sw:street" : "chemin des Anaphores",
    "sw:street-no" : {"@type": "xsd:integer", "@value" : "33"},
    "sw:city" : "sw:Lausanne"
  }]
}
```

# or no @id

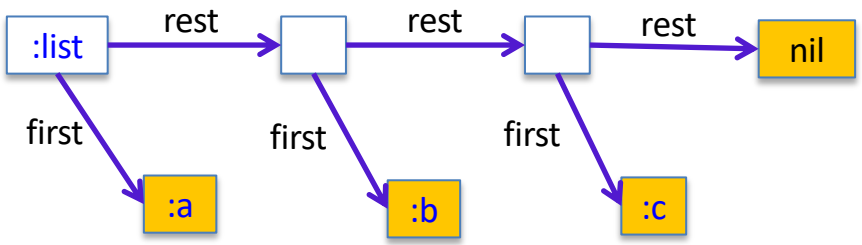
```
{
"@context" : {
 "sw" : "http://cui.unige.ch/sw-course/",
 "xsd" : "http://www.w3.org/2001/XMLSchema#”
}

{"@id" : "sw:Armand",
 "sw:address" : {
          "sw:street” : "chemin des Anaphores",
          "sw:street-no” : {"@type" : ”xsd:integer", "@value" : "466536"}
          "sw:city" : "sw:Lausanne"
  }
}}
```

no @id specified
⇒ blank node

# Lists in Turtle

```
:list rdf:first :a ;
      rdf:rest [rdf:first :b ;
                rdf:rest [rdf:first :c ;
                          rdf:rest rdf:nil]]
```

```
Abbreviated
(:a :b :c)
```



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

c:6.001  s:students  ( std:Amy, std:Mohamed, std:Johann ) .

{

# Lists in JSON-LD are JSON lists

```
{
"@context" : {
 "sw" : "http://cui.unige.ch/sw-course/",
 "xsd" : "http://www.w3.org/2001/XMLSchema#”
}

{"@id" : "sw:Anna",
 "sw:preferredLanguages" : [
          {"@id": "sw:python} ,
         {"@id": "sw:c"} ,
         {"@id": "sw:julia"J
  ]
}}
```

# Summary

- RDF is a graph data model
  - nodes are either resources (URI), literals, or blank nodes
- The RDF standard vocabulary helps modeling (among others)
  - the *is_a* (type) relationship
  - collections
  - statements (reification)
- There are different syntaxes: XML, N3, Turtle, JSON-LD, ...