# Resource description in RDF and RDFS

Amel Bouzeghoub

CSC 4538

## RDF Model

- **An RDF document is a set of expressions or statements**
- **A statement is a triple Resource-Property-Value**
  - Each resource has a unique identifier (URI : Universal Resource Identifier).
  - A property is also a resource that has a special role. It is used to describe another resource.
  - A value can be a literal or a resource.
- **An RDF document is a directed graph with labelled nodes and arcs**

TELECOM SudParis

# Levels of the RDF model

- **Four levels of modelling:**
  - Physical level: URL triples
  - Basic types: resources, properties, statements
  - Complex types: collections, lists
  - Schemas (RDFS): classes, property types

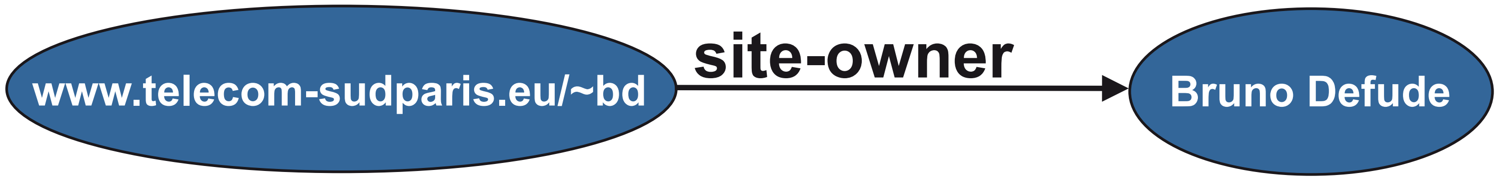
TELECOM SudParis

## 1- Physical layer

- **A statement is a triple (a; b; c) which means that "subject a has as value for property b the object c".**
  - a and b are URLs and c is a URL or a value.
- **Formally:**
  - Set of URLs: U
  - Set of literals (strings): V
  - Set of Statments:

    - $T \subseteq U \times U \times (U \cup V)$

## 1- Physical layer : example



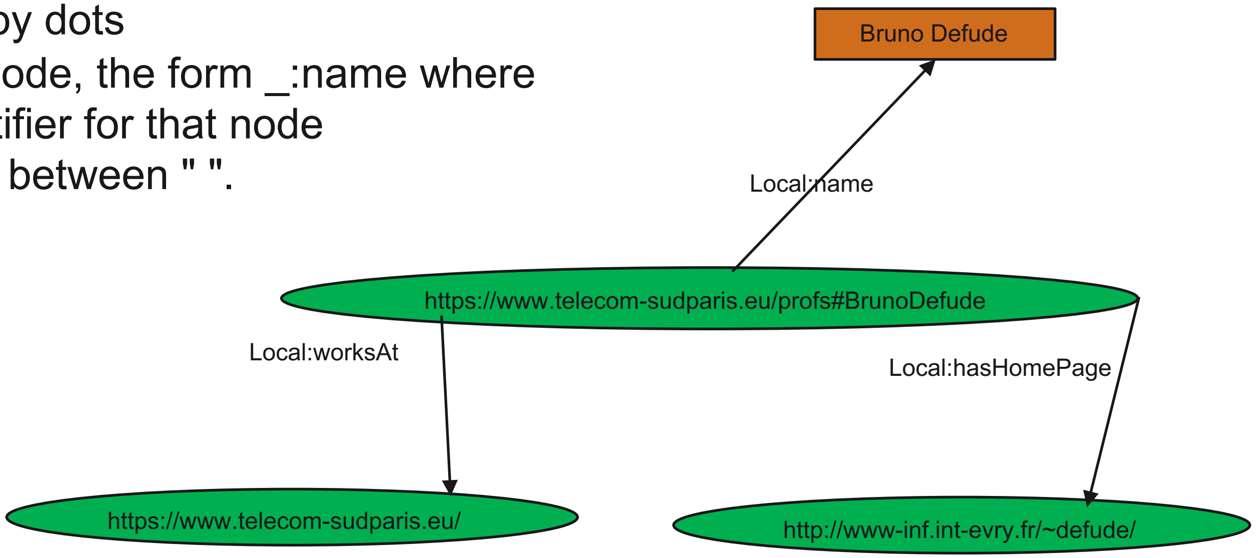
```
(www.telecom-sudparis.eu/~bd,
http://www.mydomain.org/site-owner,   #BrunoDefude)
```

A historical syntax in XML and other syntaxes : Turtle, TriG, JSON-LD, N-Triples, N-Quads

```
<rdf:Description about="www.telecom-sudparis.eu/~bd">
      <site-owner>"Bruno Defude"</site-owner>
</rdf:Description>
```

# Serialization N – Triples

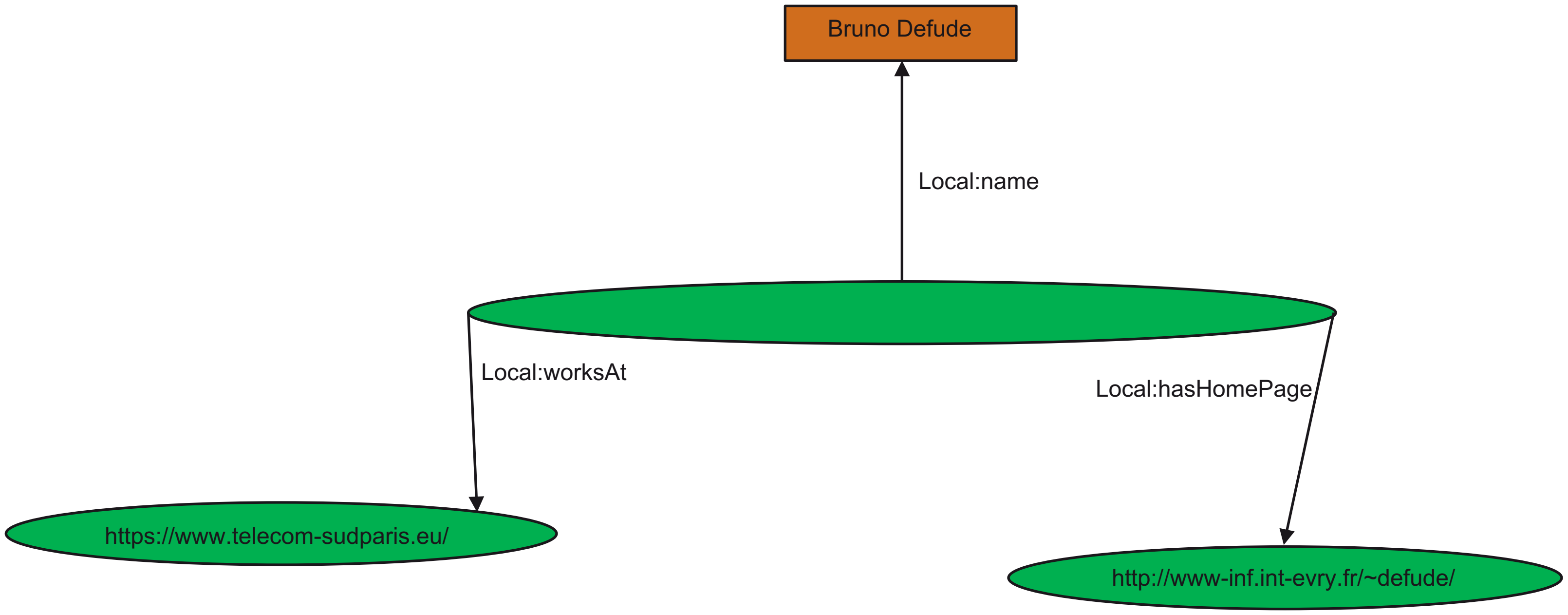
- An RDF graph is represented by a collection of triples
  **sujet prédicat objet** .
- Easy to load (parser)
- If an element is a URI, it is enclosed in chevrons: <>
- Triples are separated by dots
- If it is an anonymous node, the form _:name where name is a unique identifier for that node
- A literal is represented between " ".



Bruno Defude
Local:name
https://www.telecom-sudparis.eu/profs#BrunoDefude
Local:worksAt
Local:hasHomePage
https://www.telecom-sudparis.eu/
http://www-inf.int-evry.fr/~defude/

```
<https://www.telecom-sudparis.eu/profs#BrunoDefude> <https://www.telecom-sudparis.eu/vocab#hasHomePage> <http://www-inf.int-evry.fr/~defude/>
<https://www.telecom-sudparis.eu/profs#BrunoDefude> <https://www.telecom-sudparis.eu/vocab#worksAt > <https://www.telecom-sudparis.eu/> .
<https://www.telecom-sudparis.eu/profs#BrunoDefude> <https://www.telecom-sudparis.eu/vocab#name>   " Bruno Defude" .
```

# Serialization N – Triples


Bruno Defude
Local:name
Local:worksAt
Local:hasHomePage
https://www.telecom-sudparis.eu/
http://www-inf.int-evry.fr/~defude/

_:p43 <http://www.polymtl.ca/vocab#hasHomePage> <http://www-inf.int-evry.fr/~defude/> .
_:p43 <http://www.polymtl.ca/vocab#worksAt > <http://www.telecom-sudparis.eu> .
_:p43 <http://www.polymtl.ca/vocab#name> »Bruno Defude" .

## Serialization RDF/XML

Uses namespaces

- rdf:Description tag to group descriptions of a resource
- For an empty node, remove the about attribute
- To label an empty node, use the rdf:nodeID tag
- To represent a typed literal, use the rdf:datatype attribute in the predicate that links the resource to that literal
- There are often several ways to represent the same RDF graph

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:local="https://www.telecom-sudparis.eu/vocab#">

<rdf:Description rdf:about="http://www.telecom-sudparis.eu/profs#BrunoDefude">
<local:hasHomePage resource=" http://www-inf.int-evry.fr/~defude"/>
<local:worksAt resource=" http://www.telecom-sudparis.eu"/>
<local:name> Bruno Defude</local:name>
</rdf:Description>
</rdf:RDF>
```

# Serialization Turtle

- Allows prefixes to be specified
- Allows you to combine descriptions of the same resource:
- Use ; to group triples of the same subject
- Use , to group multiple instances of a property on the same subject
- Anonymous node represented by square brackets [ ].
- All descriptions related to an empty node can be placed inside the square brackets

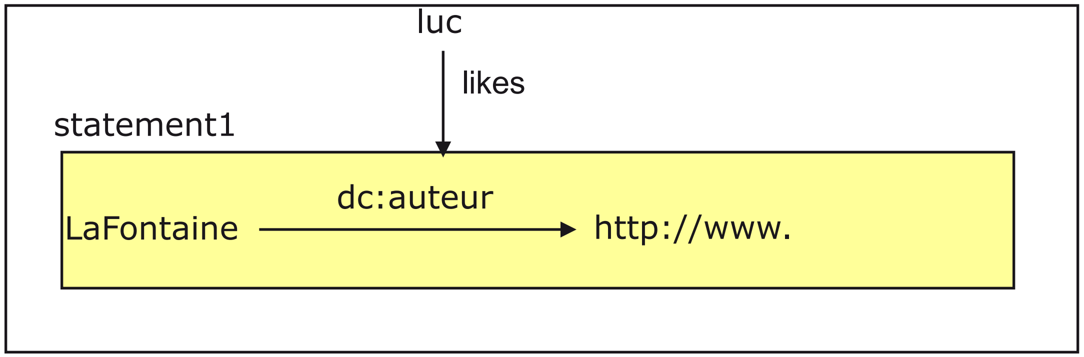
```
@prefix local: <https://www.telecom-sudparis.eu/vocab#> .
@prefix prof: <https://www.telecom-sudparis.eu/profs#> .

prof:BrunoDefude local:hasHomePage <http://www-inf.int-evry.fr/~defude> .
prof:BrunoDefude local:worksAt < http://www.telecom-sudparis.eu> .
prof:BrunoDefude local:name "Bruno Defude" .
```

```
@prefix local: <https://www.telecom-sudparis.eu/vocab#> .
@prefix prof: <https://www.telecom-sudparis.eu/profs#> .

prof:BrunoDefude local:hasHomePage < http://www-inf.int-evry.fr/~defude> ;
                 local:worksAt <http://www.telecom-sudparis.eu> ;
                 local:name   "Bruno Defude" .
```

```
@prefix local: <https://www.telecom-sudparis.eu/vocab#> .
@prefix prof: <https://www.telecom-sudparis.eu/profs#> .

prof:BrunoDefude local:hasHomePage < http://www-inf.int-evry.fr/~defude> ;
                 local:worksAt <http://www.telecom-sudparis.eu>,
                <https://digicosme.lri.fr/tiki-index.php> ;
                 local:name "Bruno Defude" .
```

## 1- Physical layer : reification

- Consider a triple as a resource
- Describe this resource
- A triple is reified (considered as an object) by a Statement
  - *(#statement1, rdf:subject, #LaFontaine)*
  - *(#statement1, rdf:predicate, dc:author)*
  - *(#statement1, rdf:object, http://www.)*
  - *(#luc, #likes, #statement1)*



## 2- Basic types

- **Rdf:Resource**

A resource is defined as anything that has a URI

- **Rdf:Property**

A property is a resource used as a predicate of a triple

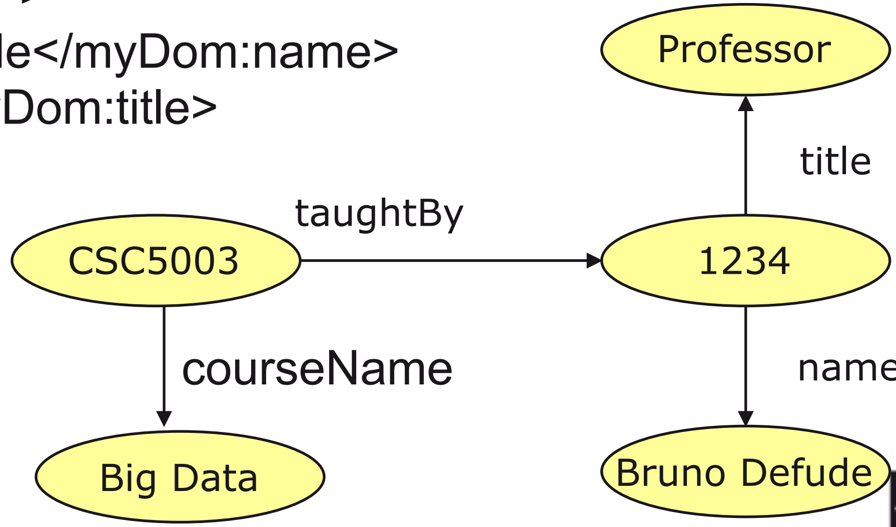
- **Rdf:Statement**

A statement is a resource that reifies a triplet, i.e., a statement that assigns a value to a property of a resource

# rdf:resource attribute

- **This attribute is used to reference, in an RDF statement, a value of type resource.**

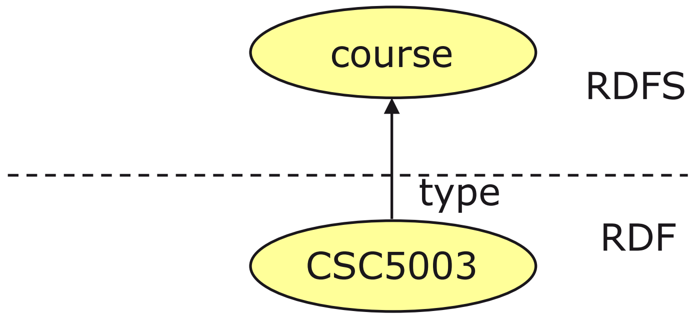
```
<rdf:Description rdf:about="http://www.telecom-sudparis/CSC5003.htm">
    <myDom:courseName > Big Data</myDom:courseName>
    <myDom:taughtBy rdf:resource="1234"/>
</rdf:Description>
<rdf:Description rdf:about="1234">
    <myDom:name> Bruno Defude</myDom:name>
    <myDom:title> Professor</myDom:title>
</rdf:Description>
```



# rdf:type (Inheritance)

- **The rdf:type element is used to define that a resource is an instance of an RDFS class (instance-of).**

<rdf:Description rdf:about="http://www.inapg.fr/omip/CSC5003.htm">
*<**rdf:type** rdf:resource="course"/>*
<myDom:CourseName> Big Data</myDom:CourseName>
<myDom:taughtBy rdf:resource="1234"/>
</rdf:Description>



## 3- Complex types : Containers and Lists

- **A container is a resource of type rdfs:Container**
- **This class has  three sub-classes :**
  - rdf:Bag: multi-set of resources (unordered)
  - rdf:Sequence: sequence of resources (ordered)
  - rdf:Alt: enumeration of resource (set of alternatives)
- **The membership of a collection is encoded by properties**
  - rdf:_1, rdf:_2, rdf:_3, . . .
- **A collection is a resource : we can have collection of collections.**
- **A list is a resource of type rdf:List :**
  - constructeurs : rdf:first, rdf:rest, rdf:nil

# Containers : example

```
<myDom:prof rdf:about="1234">
<myDom:name> Bruno Defude</myDom:name>
<myDom:title> Professor</myDom:title>
<myDom:coursRealises>
<rdf:Bag>
<rdf:_1 rdf:resource="#CSC5003"/>
<rdf:_2 rdf:resource="#CSC5006"/>
</rdf:Bag>
</myDom:coursRealises>
</myDom :prof>
```

B. Defude teaches CSC5003 and CSC5006 courses.

## Containers : example

```
<monDom:course rdf:about=  "CSC5003" monDom:nameCourse=  "Big Data">
<monDom:taughtBy>
<rdf:Alt>
<rdf:li rdf:resource="#1234"/>
<rdf:li rdf:resource="#1235"/>
</rdf:Alt>
</monDom:taughtBy>
</monDom:course>
```

- The course CSC5003 is taught either by Bruno Defude (URI 1234), or Amel Bouzeghoub(URI 1235).

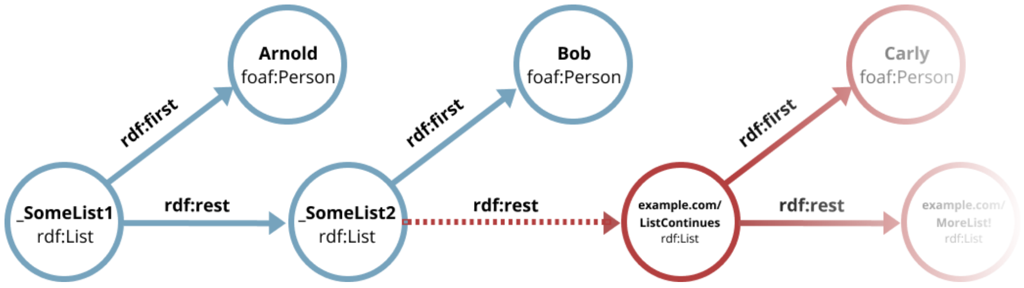
# Collection : example

- Members of Diego team:

*(#students, rdf:type, rdf:Bag)*
*(#students, rdf:_1, #Katleen)*
*(#students, rdf:_2, #Wafaa)*
*(#students, rdf:_3, #Greg)*
*(#diegomembers, rdf:type, rdf:Bag)*
*(#diegomembers, rdf:_1, #Bruno)*
*(#diegomembers, rdf:_2, #Walid)*
*(#diegomembers, rdf:_3, #Chourouk)*
*(#diegomembers, rdf:_4, #Amel)*
*(#diegomembers, rdf:_5, #students)*

# List : example

```
<ex:Group>
  <ex:members>
    <rdf:List>
              <rdf:first rdf:resource="Arnold"/>
              <rdf:rest>
                <rdf:List>
                  <rdf:first rdf:resource="Bob" >
                  <rdf:rest rdf:resource="&rdf;nil"/>
                </rdf:List>
          </rdf:rest>
    </rdf:List>
     </ex:members>
</ex:Group>
```



## Anonymous Resource

- **Intermediate Resource , without identifier**
- **Existential semantics**

```
<ns:Cours>
          <ns:auteur>
                    <ns:Person>
                              <ns:name>B. Defude</ns:name>
                    </ns:Person>
          </ns:auteur>
</ns:Cours>
```

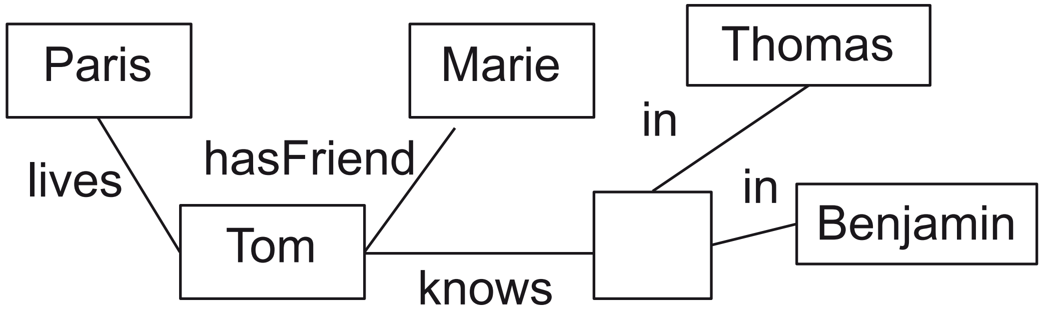
## n-aire Relation

- **Properties are binary relations**
- **To describe a n-aire relation,  we can use**
  - An anonymous resource (blank nodes)
  - N binary relations
  - Collection

# n-aire Relation : *Blank nodes*

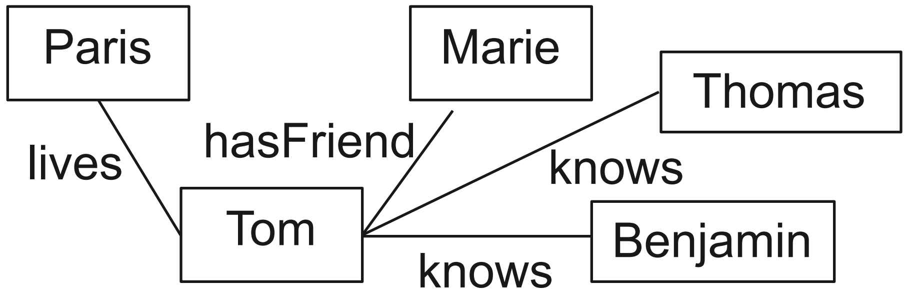
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex="http://ex.com/#">
<rdf:Description rdf:about=  "Tom">
    <ex:hasFriend rdf:resource= "Marie"/>
    <ex:lives rdf:resource=  " Paris"/>
    <ex:knows rdf:parseType="Resource">
            <ex:in rdf:resource= "Thomas"/>
            <ex:in rdf:resource= "Benjamin"/>
    </ex:knows>
</rdf:Description>
</rdf:RDF>
```

Tom knows people who are Thomas and Benjamin



# n-aire Relation : N binary relations

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex="http://ex.com/#">
<rdf:Description rdf:about= "Tom">
    <ex:friend rdf:resource=  "Marie"/>
    <ex:lives rdf:resource= "Paris"/>
    <ex:knows rdf:resource=  " Thomas"/>
    <ex:knows rdf:resource=  " Benjamin"/>
</rdf:Description>
</rdf:RDF>
```



Tom knows Thomas
Tom knows Benjamin

## RDF Schema

## Introduction

- RDF is a universal language for describing resources. It is independent of any application.
- The vocabulary used to describe resources is defined in RDFS: it is composed of a set of classes and a set of properties.
- RDFS allows the definition of classes and a hierarchy of specialisation on classes.
- An RDF resource can be an instance of an RDFS class (rdf:type).
- RDFS also allows defining properties and a specialisation hierarchy on properties.
- RDFS defines restrictions on the value of a property (range) and on the type of resource described by the property (domain)

## Introduction

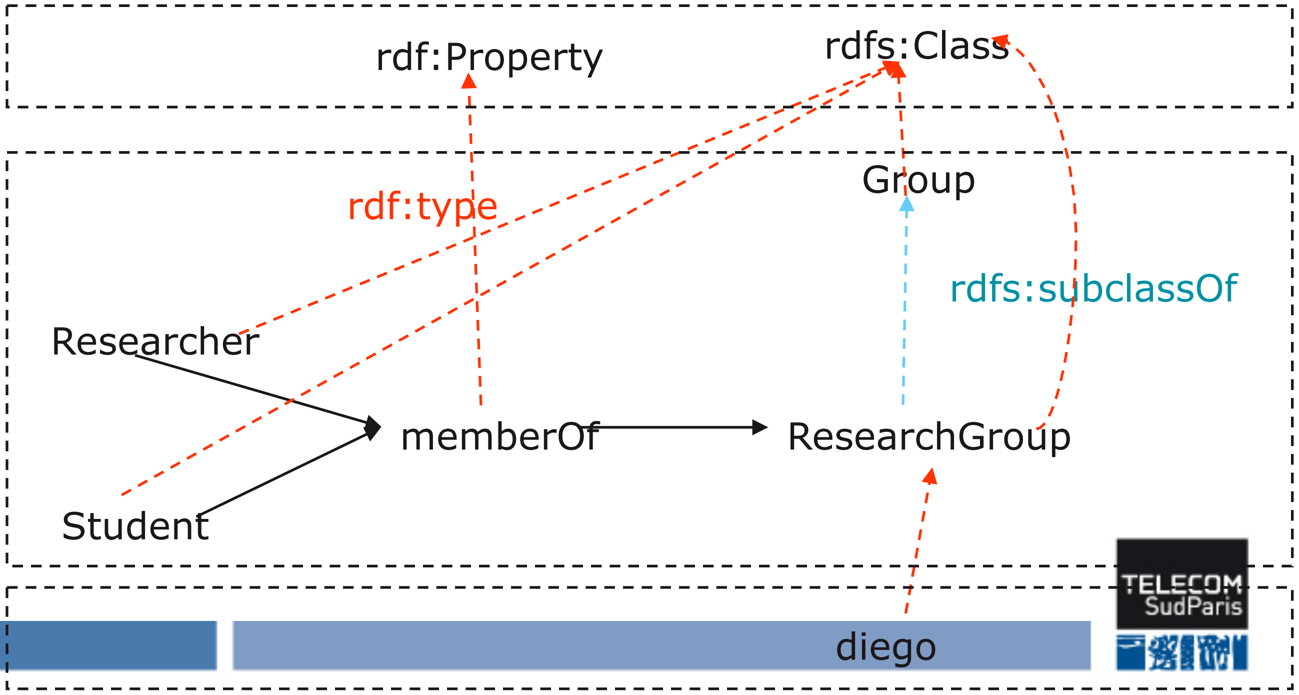
- **Towards a more object-oriented approach**
  - Class concept (group of resources)
  - Relations (properties) defined with domain and range
    - OOP: classe Book {Person author;}
      - The class is defined by its properties (attributes)
    - RDFS: author: Book-> Person
      - The relation (property) is defined by its classes

# Basic Classes and Properties

- **Basic Classes**
  - rdfs:Class, the class of all the classes
    - Class is an instance of the class Class
    - Class is a sub-class of Resource
  - rdf:Property, the class of relations.
  - rdfs:Ressource, the class of all the ressources (*Object* in Java).
  - rdfs:Literal, the class of primitives types.
  - rdf:Statement, the class of all reified instructions.
- **Properties to define relations**
  - rdf:type, to associate a resource with its class (class-instance relation).
  - rdfs:subClassOf, to associate a class with one of its superclasses.
  - rdfs:subPropertyOf, to associate a property with one of its superproperties.
- **Properties to define restrictions on properties**
  - rdfs:domain, specifies the domain of a Property, i.e. the class of all resources that can occur as subjects in a Subject-Property-Value triple.
  - rdfs:range, specifies the class of all resources that can appear as a property value in a Subject-Property-Value triple.

# Class : example

(#Group, rdf:type, rdfs:Class)
(#ResearchGroup, rdf:type, rdfs:Class)
(#ResearchGroup, rdfs:subclassOf, #Group)
(#memberOf, rdf:type, rdf:Property)
(#memberOf, rdfs:domain, #Student)
(#memberOf, rdfs:domain, #Researcher)
(#memberOf, rdfs:range, #ResearchGroup)
(#diego, rdf:type, #ResearchGroup)



## Limits of RDFS

- RDFS cannot be used to express that two classes are disjoint.
- RDFS does not allow classes to be created by combining other classes (intersection, union, complement).
- RDFS does not allow for any restriction on the number of occurrences of values that a property can take. For example, one cannot say that a person has exactly two parents.
- RDFS does not allow the definition of certain property features: transitivity, unicity, inverse property

➔ OWL!