HMIN218 : Web sémantique et langages associés

I.Mougenot

UM

2021

Plan du cours

- Web sémantique et Layer cake
- Standards W3C : RDF, RDFS

Web Sémantique

Partager les données, voire les concepts, plutôt que les documents sur le web

Passer d'un système documentaire à un système de données de manière à en faciliter la manipulation et l'interprétation par des agents logiciels.
Le WD repose sur la description normalisée des ressources du Web et donc s'appuie en grande partie sur des standards de données, l'idée est donc d'aller vers des systèmes de mutualisation de ressources.

"Universalité du savoir"

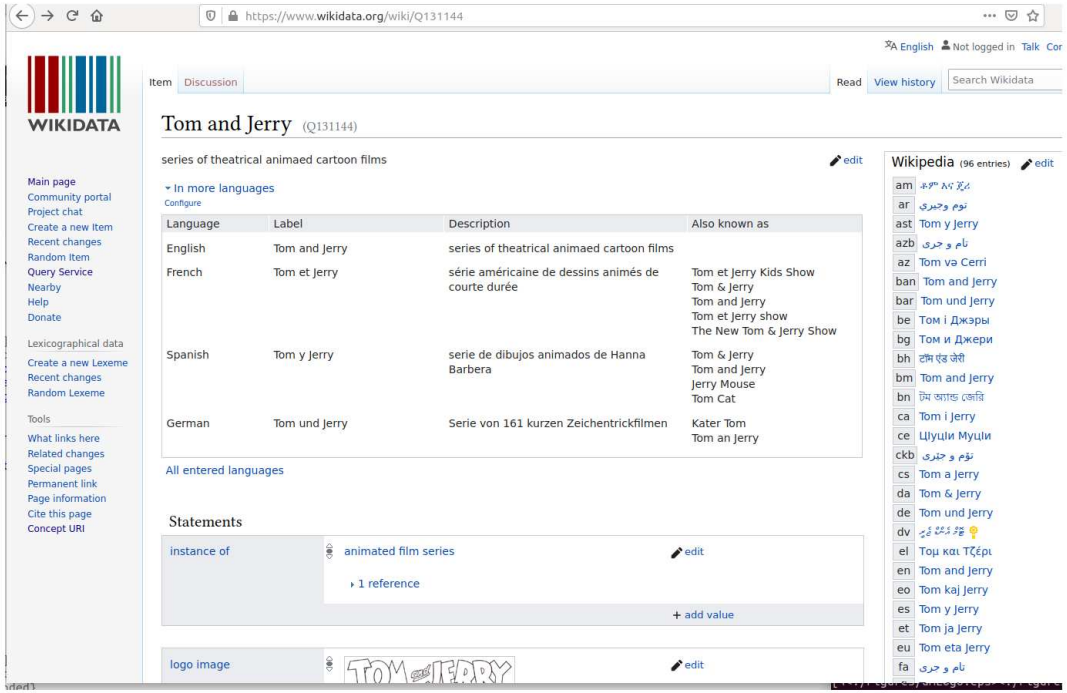


Figure: une page Wikidata

L'existant

Des systèmes de partage globalisés

- Cyc (travaux pionniers) et OpenCyc
- Freebase, Google Knowledge Graph
- Wikidata
- DbPedia
- Schema.org
- ...

Architecture du web

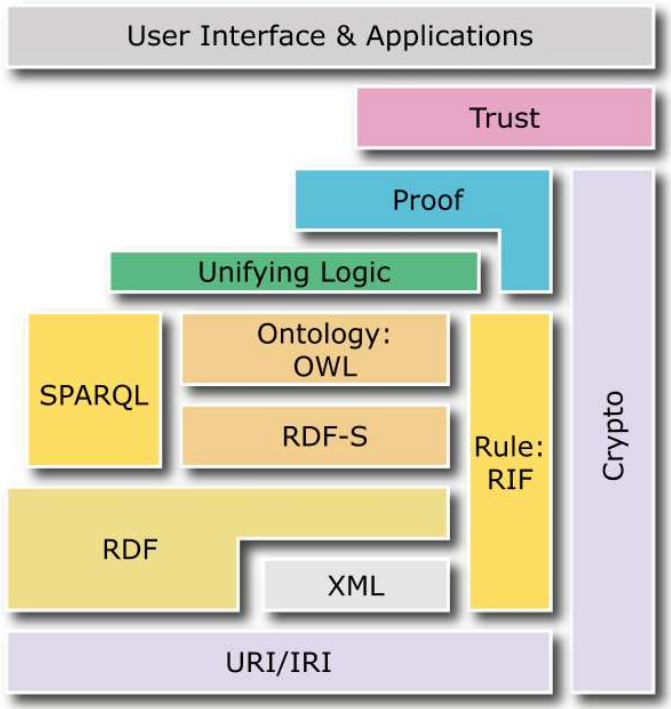


Figure: Empilement de couches (layer cake)

Resource Description framework

Langage RDF (Resource Description Framework) du W3C : initiative pour décrire des *ressources* (notamment Web) au travers de métadonnées

Principes :

- décrire une ressource ou des relations entre ressources
- apporter sa perception sur une ressource au travers d'annotations (couples propriété-valeur) sans modifier la ressource
- exploiter de manière décentralisée chaque annotation

Exemple de descriptions RDF

Espaces de nom :

xmlns:animal="http://www.ex.fr/animal#"
xmlns:rdf= 'http://www.w3.org/1999/02/22−rdf−syntax−ns#

Ovale : Ressource
Arete : Propriete
Rectangle : Litteral

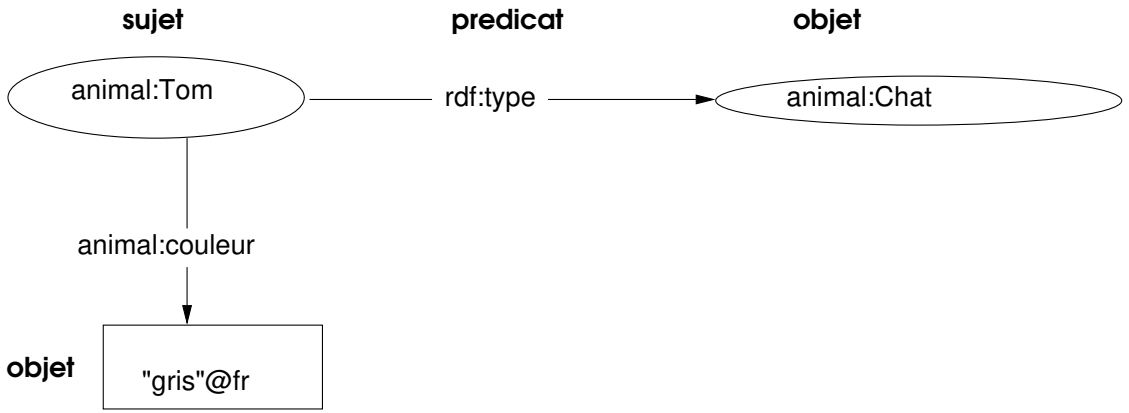


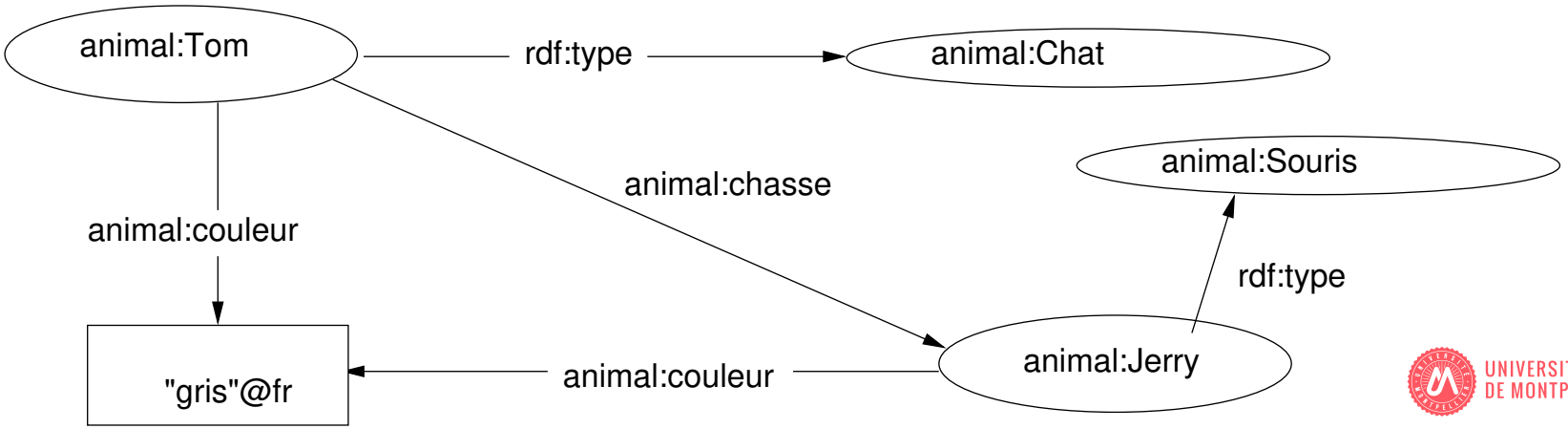
Figure: Tom est un chat gris

deux descriptions sous forme de triplets

Modèle RDF : graphe orienté, étiqueté (nœuds et arêtes)

Collection de triplets - Déclaration ou triplet RDF : {Sujet, Prédicat, Objet}

Espaces de nom : xmlns:animal="http://www.ex.fr/animal#"
xmlns:rdf= 'http://www.w3.org/1999/02/22–rdf–syntax–ns#



Espace de noms

Les espaces de noms sont des "conteneurs" de termes/objets

- notion de modèle ou de vocabulaire qui recouvre ces termes désignant des ressources
- possède un identifiant qui le rend unique (*Uniform Resource Identifier* ou URI)
- possède un qualificatif ou préfixe utilisé à la place de l'adresse URI

Exemples : espace de noms de RDF,
espace de noms des nouvelles de la BBC : xmlns:bbc="http://www.bbc.co.uk/ontologies/news/"
(http://prefix.cc/popular/all.rdf)

Identifiants de ressources

Les espaces de noms et les *Uniform Resource Identifier* jouent un rôle essentiel dans l'intégration des ressources

- Les ressources peuvent être nommées au travers d'URIs (à partir de l'espace de noms de la ressource)
- Les ressouces peuvent être anonymes (BNode) : structure composite, relation n-aire, entité non identifiée
- Les objets des triplets peuvent être des valeurs littérales (prenant un type de données XML Schema)
- Les prédicats (property) sont également étiquetés via un URI

Déclaration ou triplet RDF : $\{U \cup B \times U \times U \cup B \cup L\}$
Avec U : URI, L : Literal et B : Blank Node

Exemple de noeud anonyme

noeud composite sans étiquette

xmlns:rdf= 'http://www.w3.org/1999/02/22–rdf–syntax–ns#
xmlns:s="http://www.ex.fr/songs#"
xmlns:loc="http://www.example.org/location#"

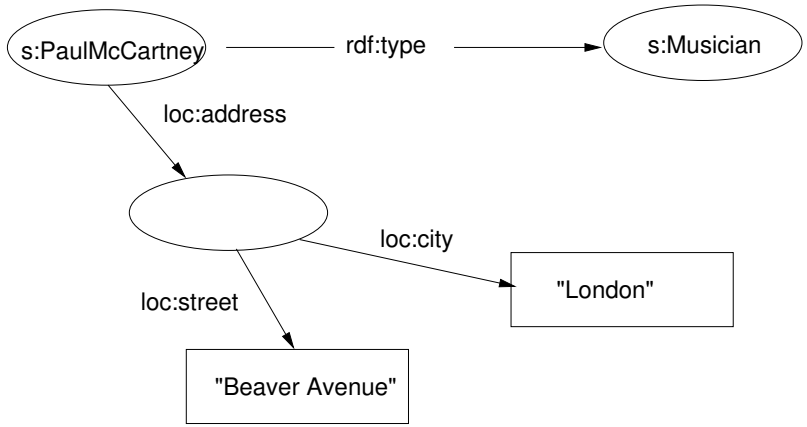


Figure: Exemple de noeud anonyme

Langages concrets

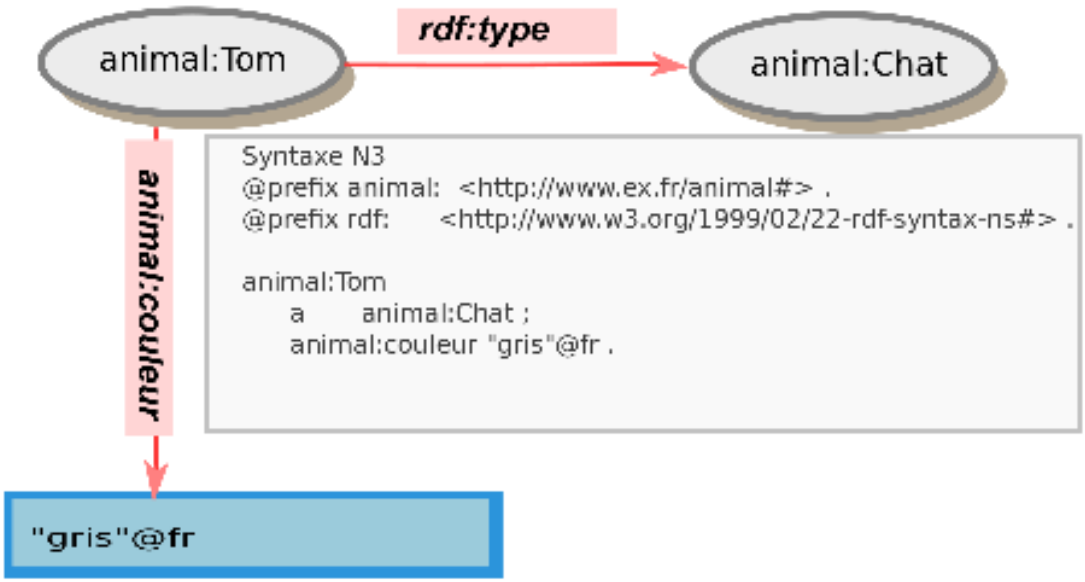
RDF est un langage abstrait, sérialisé en différents formats pour la manipulation et l'échange

- RDF/XML
- N3 et son sous-ensemble TURTLE (Terse RDF Triple Language)
- N-triple
- JSON-LD

Les plus lisibles et concis : N3 et TURTLE

Exemple de sérialisation

TURTLE



Sérialisation

Tom (format RDF/XML)

```
<http://www.ex.fr/animal#Tom> <http://www.ex.fr/animal#couleur> "gris"@fr .
<http://www.ex.fr/animal#Tom> <http://www.w3.org/2000/01/rdf-schema#label> "Tom" .
<http://www.ex.fr/animal#Tom> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
     <http://www.ex.fr/animal#Chat> .
```

Listing 1: Exemple N-TRIPLE

Sérialisation

Tom (format RDF/XML)

```
<rdf:RDF
   xmlns:animal="http://www.ex.fr/animal#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <animal:Chat rdf:about="http://www.ex.fr/animal#Tom">
    <animal:couleur xml:lang="fr">gris</animal:couleur>
    <rdfs:label>Tom</rdfs:label>
  </animal:Chat>
</rdf:RDF>
```

Listing 2: Exemple RDF/XML

RDF Schema (RDFS)

RDF Schema[a] (RDFS) : définir des vocabulaires RDF tout en s'exprimant en RDF (toute application supportant du RDF peut suppoter du RDFS)

[a] www.w3.org/TR/rdf-schema/

- notion centrale de Resource : tout objet concret ou abstrait (nommé ou non par une URI). Dans le méta-modèle RDFS : classe racine RDFS : Resource.
- décrit les ressources à partir des concepts de classe (class) et de propriété (property)
- Les classes comme les propriétés vont pouvoir être organisées en hiérarchies

Primitives de RDFS

Les classes et les propriétés sont définies de manière indépendante

classes	**propriétés**	**contraintes**
rdfs : Resource	rdf : type	rdfs : domain
rdf : Property	rdfs : subClassOf	rdfs : range
rdfs : Class	rdfs : subPropertyOf	rdfs : ConstraintProperty

Exemple de graphe RDFS

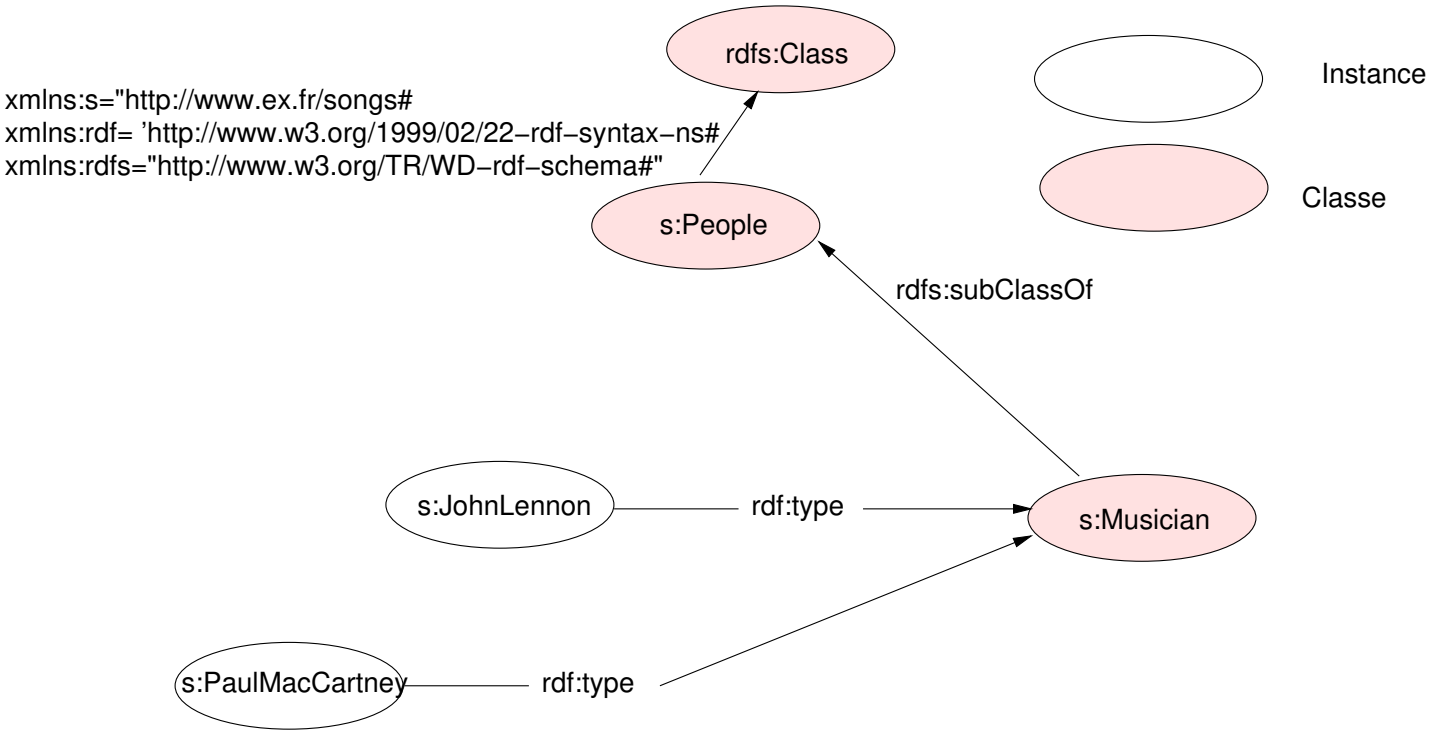


Figure: Classes RDFS

domaine de définition / cible

Enrichir la notion de *property* de RDF en précisant le concept jouant le rôle du sujet et/ou de l'objet des triplets

Deux propriétés sont définies :

1. rdfs:domain : domaine de définition d'un concept
   par ex. { animal:chasse, rdfs:domain, animal:Chat }
2. rdfs:range : concept cible (image)
   par ex. { animal:chasse, rdfs:range, animal:Souris }

Illustration

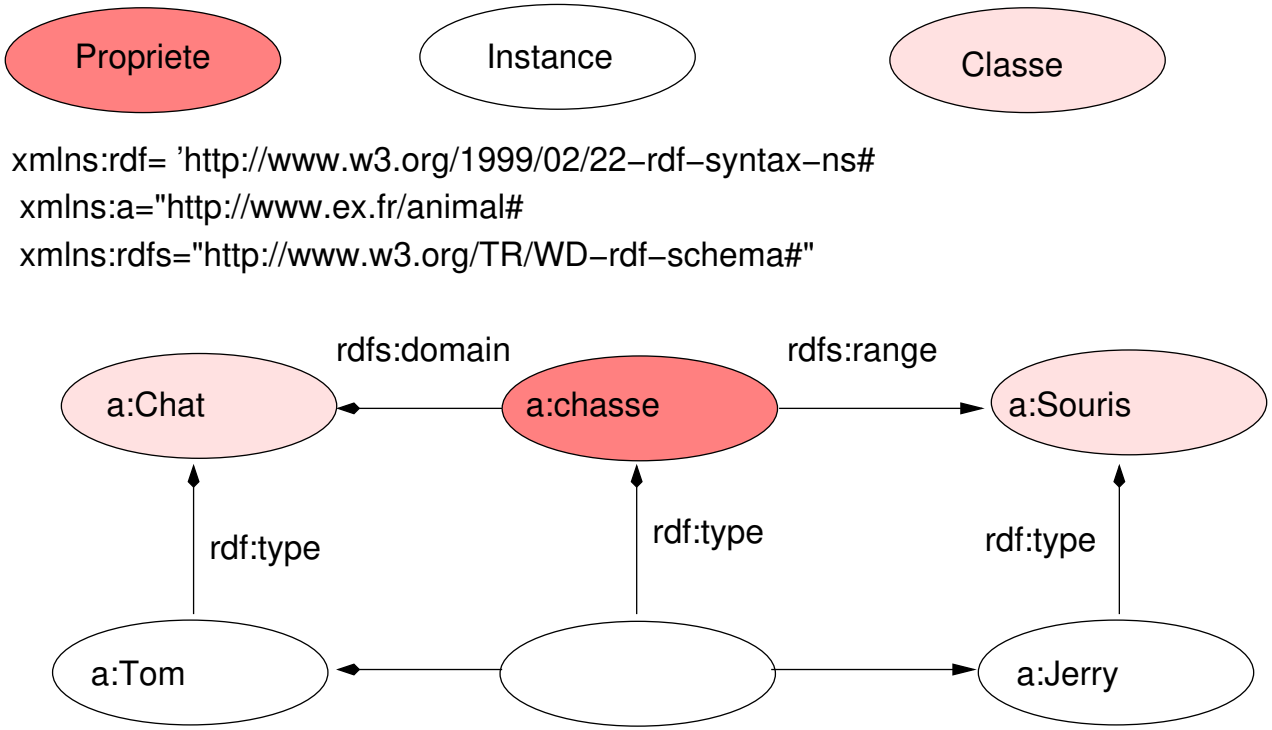


Figure: Graphe RDFS illustrant rdfs:domain et rdfs:range

Fixer des interprétations au travers de RDFS

Exploiter le typage des classes et des propriétés

1. si (c2, rdfs:subClassOf, c1) et (x, rdf:type, c2) alors (x, rdf:type:c1)

   par ex. { a:Chat, rdfs:subClassOf, a:Félin }{ a:Tom, rdf:type, a:Chat } $\mapsto$ { a:Tom, rdf:type, a:Félin }
2. si (p2 rdfs:subPropertyOf p1) et (x, p2, y) alors (x, p1, y)

   par ex. { a:effraie, rdfs:subPropertyOf, a:chasse } { a:Tom, a:effraie, a:Jerry } $\mapsto$ { a:Tom, a:chasse, a:Jerry }
3. si (p1, rdfs:domain, c1) et (x, p1, y) alors (x, rdf:type, c1)

   par ex. { a:chasse, rdfs:domain, a:Chat } { a:Tom, a:chasse, a:Jerry } $\mapsto$ { a:Tom, rdf:type, a:Chat }

Portion du métamodèle RDFS

RDFS au travers de son métamodèle permet la construction de vocabulaires RDF

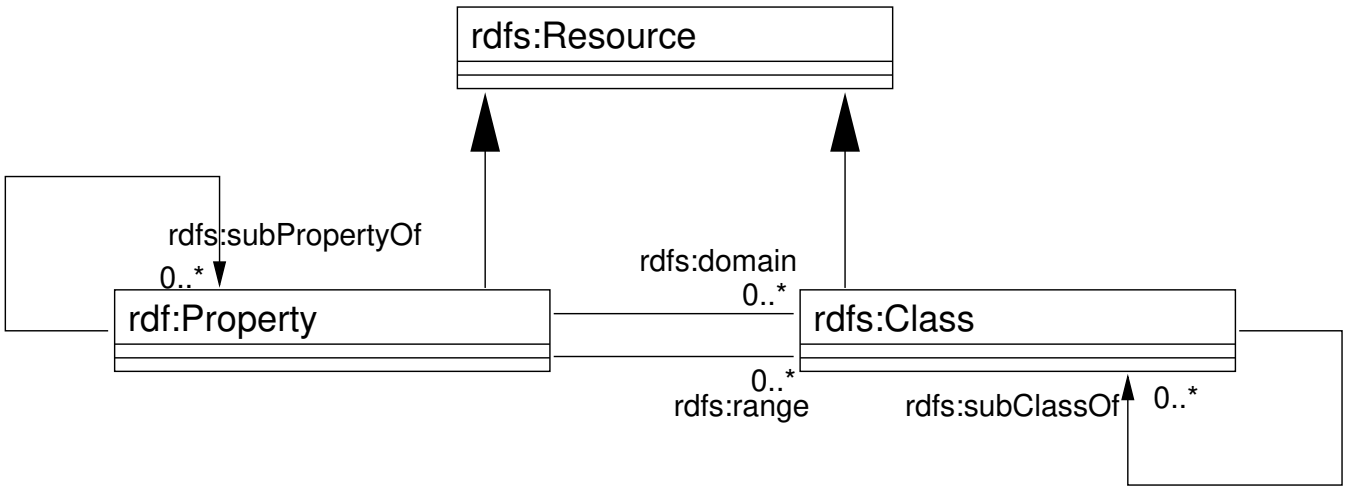


Figure: Diagramme de classes UML : portion du métamodèle RDFS

Des outils pour exploiter RDF et RDFS

Liste non exhaustive

- APIs Java : Jena (Apache), OWL-API
  - Jena http://jena.sourceforge.net
  - SPARQL
  - Paquetage pour l'exploitation de graphes RDF (org.apache.jena.rdf.model)
- API PHP : RAP RDF API for PHP
- RDFLib : API RDF pour Python

Jena - 1 modèle, 1 espace de noms, 1 ressource, 1 propriété

```
static final String animal_ns = "http://www.ex.fr/animal#";
...
        Model m = ModelFactory.createDefaultModel();
        m.setNsPrefix("animal", animal_ns);
        Resource tom = m.createResource(animal_ns + "Tom");
        // Propriete couleur
        Property color = m.createProperty(animal_ns + "couleur");
        tom.addProperty(color, m.createLiteral("gris", "fr"));
        m.write(System.out, "TURTLE");
```

Listing 3: Construction et affichage de triplets

Jena - Enrichir le modèle

```
import org.apache.jena.vocabulary.RDF;
...
        Model m = ModelFactory.createDefaultModel();
        m.setNsPrefix("animal", animal_ns);
        m.setNsPrefix("rdfs", RDF.getURI());
        // Ressource Chat
        Resource cat = m.createResource(animal_ns + "Chat");
        // Ressource Tom
        Resource tom = m.createResource(animal_ns + "Tom");
        tom.addProperty(RDF.type, cat);
        m.write(System.out, "RDF/XML");
```

Listing 4: Construction et affichage de triplets

Jena - Littéraux typés

```
import org.apache.jena.vocabulary.RDF;
import org.apache.jena.vocabulary.XSD;
...
        Model m = ModelFactory.createDefaultModel();
        m.setNsPrefix("animal", animal_ns);
        m.setNsPrefix("rdfs", RDF.getURI());
        m.setNsPrefix("xsd", XSD.getURI());
        // Ressource Tom
        Resource tom = m.createResource(animal_ns + "Tom");
        Property color = m.createProperty(animal_ns + "couleur");
        tom.addProperty(color, m.createTypedLiteral("gris", XSD.getURI() +
             "string"));
        Property age = m.createProperty(animal_ns + "age");
           tom.addProperty(age, m.createTypedLiteral("4", XSD.getURI() + "int"));
```

Listing 5: Enrichir le modèle

Jena - Persister le modèle construit dans un fichier

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
..
   try { FileOutputStream outStream = new FileOutputStream("tom.n3"); //
           m.write(outStream, "N3");
           outStream.close(); }
           catch (FileNotFoundException e)
           {System.out.println("File not found");} catch (IOException e)
           {System.out.println("IO problem");}
      }
```

Listing 6: Exporter le contenu

Exploiter RDFS

Tom individu de la classe Chat

```
// Chat
Resource cat = m.createResource(animal_ns + "Chat");
m.add(cat, RDFS.subClassOf,RDFS.Class);
// Tom
Resource tom = m.createResource(animal_ns + "Tom");
tom.addProperty(RDF.type, cat);
tom.addProperty(RDFS.label, "Tom");
```

Listing 7: Chat est une classe

Jena - hiérarchie de classes RDFS

```
// Personnage
Resource personnage = m.createResource(animal_ns + "Personnage");
m.add(personnage, RDFS.subClassOf,RDFS.Class);
// Chat
Resource cat = m.createResource(animal_ns + "Chat");
m.add(cat, RDFS.subClassOf,RDFS.Class);
// sous-classe de
cat.addProperty(RDFS.subClassOf, personnage);
// Tom
Resource tom = m.createResource(animal_ns + "Tom");
tom.addProperty(RDF.type, cat);
tom.addProperty(RDFS.label, "Tom");
```

Listing 8: Arborescence de classes

Jena - RDFS et N3

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix animal: <http://www.ex.fr/animal#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

animal:Tom a      animal:Chat ;
      rdfs:label  "Tom" ;
      animal:age  "4"^^xsd:int ;
      animal:couleur "gris" .

animal:Chat rdfs:subClassOf animal:Personnage , rdfs:Class .

animal:Personnage rdfs:subClassOf rdfs:Class .
```

Listing 9: Résultat N3

Jena - Explorer le contenu d'un fichier : lire

```
public static final String rdf_file = "tom.n3";
@SuppressWarnings("deprecation")
public static void main(String args[]) {

   try {
      Model m = ModelFactory.createDefaultModel();
      FileManager.get().readModel( m, rdf_file );
      String a_ns = m.getNsPrefixURI("animal");
      Resource cat = m.getResource(a_ns + "Chat");
      Property color = m.getProperty(a_ns + "couleur");
      System.out.println(cat.getLocalName());
      System.out.println(cat.getNameSpace());
      System.out.println(color.getURI());
   }
```

Listing 10: Lire

Jena - Explorer le contenu d'un fichier : la couleur de Tom

```
public static final String rdf_file = "tom.n3";
   @SuppressWarnings("deprecation")
   public static void main(String args[]) {

      try {
         Model m = ModelFactory.createDefaultModel();
         FileManager.get().readModel( m, rdf_file );
         String a_ns = m.getNsPrefixURI("animal");
         Resource tom = m.getResource(a_ns + "Tom");
         Property color = m.getProperty(a_ns + "couleur");
         NodeIterator nod_i1 = m.listObjectsOfProperty(tom, color);
          while (nod_i1.hasNext())
          { RDFNode n = nod_i1.nextNode();
               Literal l = (Literal) n;
               System.out.println(l.getLexicalForm());
               }
      }
 catch (Exception e) {
         System.out.println("failure" + e);
      }
      }
```

Listing 11: naviguer dans le graphe

Jena - Explorer les statements (déclarations)

```
  public static final String rdf_file = "tom.n3";
  @SuppressWarnings("deprecation")
  public static void main(String args[]) {
     try {
        Model m = ModelFactory.createDefaultModel();
        FileManager.get().readModel( m, rdf_file );
        String a_ns = m.getNsPrefixURI("animal");
        System.out.println("Nombre de triplets du graphe = " + m.size());
        StmtIterator stmt_i = m.listStatements();
        while (stmt_i.hasNext()) {
           Statement stmt = stmt_i.nextStatement();
           Resource s = stmt.getSubject();
           if (s.isAnon()) {
              System.out.println(" Noeud anonyme " + s.getId());
           } else {
              System.out.println(" Nom Qualifie : " + s.getLocalName());
           }
        }
     }
catch (Exception e) {
        System.out.println("failure" + e);
     }
     }
```

Listing 12: Statements