

# OWL, RDF, N3 : les langages de description des connaissances

XML, un langage d'arbre

Année 2008-09

## Représenter les connaissances en XML

### Enjeux

Actuellement

- le Web : une source de données extraordinaire
- lisibles (normalement) par l'homme
- mais encore difficilement accessibles par la machine

*...vers le web sémantique ?*

- contenu du web accessible et utilisable par les programmes et agents logiciels
- grâce à un système de métadonnées formelles

## Vers le Web Sémantique

### La vision du web sémantique du W3C

- les ressources sont toutes identifiées par une URI *Uniform Resource Identifier*
- un langage pour la description des relations entre les ressources *RDF : Resource Description Framework*
- une extension pour la description des propriétés des ressources *RDF-Schema*
- une extension pour la description des propriétés des relations : un langage d'ontologies *OWL : Web Ontology Language*

## Les objectifs du web sémantique

*Objectifs :*

- *générer* des données sémantiques à partir de la saisie d'information par les utilisateurs ;
- *agréger* des données sémantiques afin d'être publiées ou traitées ;
- *publier* des données sémantiques avec une mise en forme personnalisée ou spécialisée ;
- *échanger* automatiquement des données en fonction de leurs relations sémantiques ;

- *générer* des données sémantiques automatiquement, sans saisie humaine, à partir de règles d'inférences

source : *Wikipedia*

### Historique

- La notion de Web sémantique est proposée dès 1994 par Tim Berners-Lee, au congrès de création du W3C
- La première spécification de *RDF* date de 1999
- Les recommandations de *RDFS* et *OWL* datent de 2004

### Représenter les données

#### RDF : Resource Description Framework

- les données sont représentées sous la forme de triplets :  
*Sujet Verbe/Prédictat Objet*
- chacun des éléments est une ressource ;
- plusieurs syntaxes possibles : la vision XML est le langage *RDF/XML*, mais il existe d'autres langages (*NOTATION3*)
- comparable à d'autres formats. Pour un triplet  $(S,P,V)$  :
  - programmation logique :  $P(S,V)$
  - bases de données : *table P, attributs pour S et V*
  - ...

### Un premier exemple

Marc joue de l'accordéon.

en RDF :

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:mus="http://www.musiciens.org#">
  <mus:marc>
    <mus:joue>accordeon</mus:joue>
  </mus:marc>
</rdf:RDF>
```

en NOTATION3 :

```
@prefix mus: <http://www.musiciens.org#> .
mus:marc mus:joue mus:accordeon .
```

### Quelques variations

en RDF :

```

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:mus="http://www.musiciens.org#">

  <mus:marc mus:joue="accordeon" />
  <mus:marc mus:joue="guitare" />

</rdf:RDF>

```

en NOTATION3 :

```
mus:marc mus:joue mus:accordeon, mus:guitare .
```

### Toujours en N3

```

:marc :joue :accordeon, :guitare ;
      :nomme "Marc".
:chloe :joue :violon ;
       :age 10 ;
       :pere :marc ;
       :nomme "Chloé".

```

### URI par défaut, sans URI

- Par défaut, le préfixe est l'adresse du document lui-même. On peut écrire directement :

```
<#marion> <#joue> <#guitare> .
```

ce qui est équivalent à :

```
@prefix : <\#> .
:marion :joue :guitare .
```

- On peut désigner des objets sans URI :  

```
[ :nomme "Marion"; :age 12 ] .
:chloe :animal [ :nomme :grigri ] .
```

Les objets ainsi décrits ne peuvent pas être référencés.

### Associer des collections de valeurs

Différentes propositions suivant les langages :

- en *RDF*, différents connecteurs :
  - *rdf:Bag* pour une collection non ordonnée
  - *rdf:Seq* pour une collection ordonnée
  - *rdf:Alt* pour indiquer un choix
- en *N3* :
  - notion de liste (commune à *RDF*) :  

```
:weekEnd :est ("Samedi" "Dimanche") .
```

## Associer des collections de valeurs

```
<rdf:Description
  rdf:about="http://www.musiciens.org#pataques">
  <cd:artist>
    <rdf:Bag>
      <rdf:li>Marion</rdf:li>
      <rdf:li>Charlotte</rdf:li>
      <rdf:li>Pierre</rdf:li>
      <rdf:li>Kevin</rdf:li>
    </rdf:Bag>
  </cd:artist>
</rdf:Description>
```

## Dublin Core

Pour décrire ses connaissances, chacun peut développer son propre vocabulaire.

Mais certains projets ont proposé un vocabulaire, maintenant standardisé, pour décrire certaines données.

### Dublin Core Meta-Data Initiative

- est un projet engagé dans le développement de méta-données pour l'interopérabilité des documents sur le web
- plus particulièrement dédié à la description *de documents*

## Dublin Core

### Dublin Core Metadata Element Set

```
xmlns:dc="http://purl.org/dc/elements/1.1/"
```

Un ensemble d'éléments pour décrire une ressource :

- *dc:title*
- *dc:contributor*
- *dc:creator*
- *dc:publisher*
- *dc:date*
- *dc:type*
- *dc:isPartOf*
- ...

## Exemples avec Dublin Core

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<> dc:title "Dublin Core icon";
  dc:Identifier
```

```

    "http://purl.org/metadata/dublin_core/images/dc2.gif";
    dc:Type "image";
    dc:Format "image/gif 4kB".
-----
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<> dc:Subject "Saturn";
    dc:Type "image";
    dc:Format "image/gif 640 x 512 pixels";
    dc:Identifiant
        "http://www.not.iac.es/newwww/photos/images/satnot.gif" .

```

### Comparaison avec d'autres formats

Description sous forme de triplets se retrouvent sous d'autres formats :

- en langue naturelle!
- sous forme de graphe (les sujets, les objets sont des noeuds, les propriétés étiquettent les arcs);
- en programmation logique :

```

nomme(marion, "Marion").
nomme(chloe, "Chloé").
nomme(grigri, "grigri").
animal(chloe, grigri).
joue(marion, guitare).
joue(chloe, violon).

```

### Comparaison avec d'autres formats

- en bases de données :
  - avec une table pour associer les ressources à leur identifiant ;
  - une table par prédicat/propriété : il suffit de deux colonnes **sujet** et **objet**.

```

create table ressources(id-ressource serial primary key,
    uri-ressource URI);
create table animal(id-sujet int, id-objet int,
    primary key (id-sujet, id-objet));
create table joue(id-sujet int, id-objet int,
    primary key (id-sujet, id-objet));

```
- bien que n'ayant pas été conçu pour cela, *RDF* fournit un moyen de sérialiser une base de données vers du XML!

### Ontologies

Origine du mot :

- en philosophie, une ontologie c'est l'étude de l'être en tant qu'être
- *théorie de l'existence* : expliquer les concepts qui existent dans le monde et comment ces concepts s'imbriquent et s'organisent

En informatique

- outils qui permettent précisément de représenter un corpus de connaissances sous une forme utilisable par une machine
- **T. Gruber, 1993** : *Une ontologie est la spécification d'une conceptualisation d'un domaine de connaissance.*

*source : Wikipedia*

### Langages pour les ontologies

- classiquement basés sur la logique du premier ordre :
    - NOTATION3
    - N-Triples
    - RDF
  - basés sur les logiques de descriptions (graphes orientés avec arcs étiquetés) :
    - OWL : Web Ontology Language et ses trois sous langages, *OWL-Lite*, *OWL-DL* et *OWL-Full*.
- OWL est actuellement le standard utilisé pour les ontologies sur le web.

### RDF-Schema

**RDF-Schema** est une extension de RDF qui permet de

- typer un sujet ou un objet, i.e. *associer une classe*;
- typer une relation, i.e. lui associer *un domaine d'entrée et un domaine d'applications*;
- déclarer une hiérarchie de classes;
- déclarer une hiérarchie de propriétés.

### Typer des ressources

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

```
:InstrMus a rdfs:Class .
:Cordes rdfs:subClassOf :InstrMus .
:Vents rdfs:subClassOf :InstrMus .
```

```
:violon a :Cordes .
:guitare a :Cordes .
:accordeon a :Vents .
```

## Typage des propriétés

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix mus: <#> .
```

```
mus:Personne a rdfs:Class .
mus:Musicien rdfs:subClassOf mus:Personne .
mus:Homme rdfs:subClassOf mus:Personne .
```

```
mus:joue a rdfs:Property ;
    rdfs:domain mus:Musicien ;
    rdfs:range mus:InstrMus .
```

```
mus:pere a rdfs:Property ;
    rdfs:domain mus:Personne ;
    rdfs:range mus:Homme .
```

## Inférer des informations

De tous les documents précédents, un logiciel d'ontologies doit pouvoir faire des déductions :

- *:chloe* est un *:Musicien*
- *:marc* est un *:Musicien*
- *:marc* est un *:Homme*

## Langages de requêtes

- *SPARQL* est un langage de requêtes pour RDF
- c'est une recommandation depuis Janvier 2008

```
:paul :nom_usuel "Paulo" .
:paul :identite [ :nom_famille "Dupont"],
               [ :prenom "Paul" ] .
```

## Un exemple

```
:barbara :nom_usuel "Barbara" .
:barbara :identite [ :nom_famille "Dumoulin"],
                  [ :prenom "Barbara" ] .

:fx :nom_usuel "Fx" .
:pierre :identite [ :nom_famille "Dumoulin" ],
                 [ :prenom "Francois-Xavier" ] .
```

```

:sarah :nom_usuel "Sarah" .
:sarah :identite [ :nom_famille "Dupont" ],
               [ :prenom "Sarah" ] .

```

### Un exemple

```

SELECT ?x
WHERE { ?x :nom_usuel "fx" }
-----
| x      |
=====
| :fx    |
-----
SELECT ?x ?nom
WHERE {?x :nom_usuel ?nom}
-----
| x      | nom      |
=====
| :paul  | "Paulo"  |
| :barbara | "Barbara" |
| :sarah | "Sarah"  |
| :fx    | "Fx"     |
-----

```

### Un exemple

```

SELECT ?prenom
WHERE
  { ?y :nom_famille "Dumoulin" .
    ?y :prenom ?prenom .
  }
-----
| prenom      |
=====
| "Francois-Xavier" |
| "Barbara"      |
-----

```

### Aller plus loin...

- ...dans la description des relations et des classes.
- OWL : Web Ontology Language**
- définir des cardinalités pour chaque relation
- définir des équivalences entre relations

- définir des règles de subsomption entre relations
- définir des équivalences entre classes
- définir des égalités entre ressources
- définir des différences
- ...

### Exemples

- Définir des cardinalités sur une relation `:pere a owl:FunctionalProperty`
- Définir des synonymes `:marionnette owl:sameAs :marion . :chlochlo = :chloe .`
- Définir des équivalences de classes `:Sympathisants owl:equivalentClassAs :Militants`
- Définir des différences `:marion owl:differentFrom :marc .`
- Définir des ensembles disjoints `:Cordes owl:disjointFrom :Vents .`

### Un exemple

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:InstrMus a rdfs:Class .

:Cordes rdfs:subClassOf :InstrMus .

:CordesPincees rdfs:subClassOf :Cordes ;
  rdfs:label "Cordes Pincées" ;
  owl:disjointFrom :CordesFrappees, :CordesFrottees .
```

### Un exemple

```
:CordesFrappees rdfs:subClassOf :Cordes ;
  rdfs:label "Cordes Frappées" ;
  owl:disjointFrom :CordesPincees, :CordesFrottees .

:CordesFrottees rdfs:subClassOf :Cordes ;
  rdfs:label "Cordes Frottées" ;
  owl:disjointFrom :CordesFrappees, :CordesPincees .

:Vents rdfs:subClassOf :InstrMus .

:Percussions rdfs:subClassOf :InstrMus .
```

## Vocabulaire de OWL

Une partie du vocabulaire de OWL

### Opérateurs sur les classes

```
owl:intersectionOf owl:unionOf
owl:complementOf owl:oneOf
```

### Restrictions sur les propriétés

```
owl:Restriction owl:onProperty
owl:someValuesFrom owl:allValuesFrom
owl:hasValue owl:cardinality
owl:minCardinality owl:maxCardinality
```

### Contraintes sur les classes

```
owl:equivalentClass owl:subClassOf
owl:disjointWith
```

### Exemple

```
<owl:Class rdf:ID="Human">
  <rdfs:subClassOf="Anything"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasParents"/>
      <owl:allValuesFrom rdf:resource="#Human"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

### Encore un peu plus loin...

au-delà des possibilités actuelles de RDF/RDFS/OWL...

Définir des règles d'implications en NOTATION3

```
@prefix log: <http://www.w3.org/2000/10/swap/log#>.
:violon :possede :archet .
@forall :x, :y.
{ :x :possede :archet } log:implies
  { :x a :CordesPincees } .
{ :x :pere :y } log:implies { :y :enfant :x } .
```

### Encore un peu plus loin...

*Définir des règles d'implications en NOTATION3*

Le connecteur *log:implies* s'abrege en =>, ?x désigne une variable quantifiée universellement.

On peut écrire :

```
:violon :possede :archet .  
  
{ ?x :possede :archet } => { ?x a :CordesPincees } .  
  
{ ?x :possede :archet . ?x :possede :touches .}  
=> { ?x = :nyckelHarpa .} .  
  
{ ?x :pere ?y } => { ?y :enfant ?x } .
```

### Exercices

1. redéfinir les relations : *rdfs:subClassOf*, *rdfs:sameAs* avec des règles d'équivalence ;
2. proposer une représentation des membres d'une famille (plusieurs générations).  
On veut connaître (ou déduire) les relations entre membres de la famille.