XML, Langage de description

Resource Description Framework

Resource Description Framework Schema

Web Ontology Language

Problématique de l'accès aux ressources du Web

- *« Le World Wide Web a été conçu à l'origine pour la compréhension humaine, et bien que tout ce qui y réside est lisible par une machine, ces données ne sont pas compréhensibles par une machine. »*
- *« Il est très difficile d'automatiser quoi que ce soit sur le Web, et à cause du volume d'information que le Web contient, il est impossible de gérer cela manuellement ».*
- *« La solution proposée ici est d'utiliser les **métadonnées** pour décrire les **données** contenues sur le Web ».*
- *« Les **métadonnées** sont des "**données à propos des données**" (par exemple, un catalogue de bibliothèque est une compilation de métadonnées, puisqu'il décrit les publications) ou spécifiquement dans le contexte de cette spécification "des données décrivant les ressources Web". »*
- *« La distinction entre les "**données**" et les "**métadonnées**" n'est pas absolue ; c'est tout d'abord une distinction créée par une application particulière, et très souvent la même ressource traitée de plusieurs façons simultanément. »*

L'initiative du « Dublin Core »

- Le *Dublin Core* est un ensemble de 15 éléments de métadonnées ayant trait:
  - au *Contenu*: Title, Description, Subject, Source, Coverage, Type, Relation
  - à la *Propriété intellectuelle*: Creator, Contributor, Publisher, Rights
  - à la *Version*: Date, Format, Identifier, Language
- Les balises du Dublin core forment un vocabulaire non ambigu par leur association à l'espace de nom du Dublin Core
  - **http://purl.org/dc/elements/1.1/**

Nom de l'élément	**Identifiant**	**Définition**
titre	Title	Le nom donné à la ressource
créateur	Creator	L'entité principalement responsable de la création du contenu de la ressource
sujet, mots-clefs	Subject	Le sujet du contenu de la ressource
description	Description	Une description du contenu de la ressource
éditeur	Publisher	L'entité responsable de la diffusion de la ressource, dans sa forme actuelle, tels, un département universitaire, une entreprise.
contributeur	Contributor	Une entité qui a contribué à la création du contenu de la ressource
date	Date	Une date associée avec un événement dans le cycle de vie de la ressource
type	Type	La nature ou le genre du contenu de la ressource
format	Format	La matérialisation physique ou digitale de la ressource
identifiant	Identifier	Une référence non ambiguë à la ressource dans un contexte donné
source	Source	Une référence à une ressource à partir de laquelle la ressource actuelle a été dérivée
langue	Language	La langue du contenu intellectuel de la ressource
relation	Relation	Une référence à une autre ressource qui a un rapport avec cette ressource
couverture	Coverage	La portée ou la couverture spatio-temporelle de la ressource
droits	Rights	Information sur les droits sur et au sujet de la ressource

Introduction à RDF

- **RDF** (Resource Description Framework) est un moyen d'encoder, d'échanger et de réutiliser des métadonnées « structurées » pour ***décrire*** des données. C'est un idiome XML développé par le *W3C* et ayant fait l'objet d'une ***Recommandation*** en 1999.

- **RDF** ne précise pas la sémantique des ressources décrites par les différentes communautés d'utilisateurs de métadonnées; chacune de ces communautés peut y intégrer son propre vocabulaire via un espace de nom et un schema de description de ses termes.

- Fondé sur XML, **RDF** est aussi un langage extensible, un métalangage; c'est un cadre [*framework*] de description des ressources applicable à n'importe quel domaine d'application.

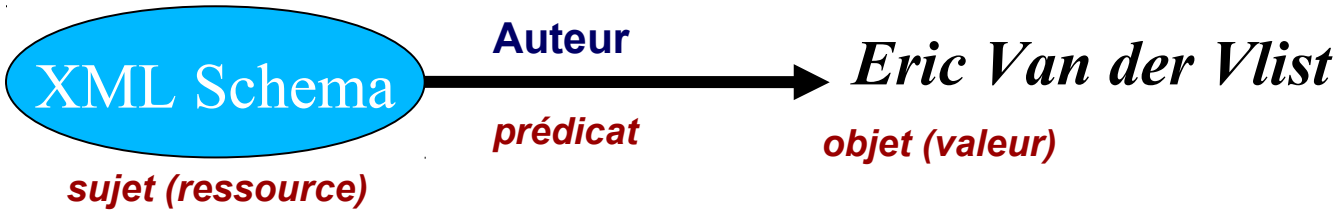
Principes de RDF

- **Les expressions RDF sont formées de triplets**
  - ***Sujet*** ***prédicat*** ***objet***
    ou
  - ***Ressource*** ***propriété*** ***valeur***

- **Exemple** :
  - Le document ***XML Schéma*** **a pour auteur** ***Eric Van der Vlist***

  sujet *prédicat* *objet*

- Ces triplets sont modélisés à l'aide de graphes orientés étiquetés



Identification des ressources

- Les ressources sont identifiées par des *URI* (Unified Ressource Identifier). Les *URI* peuvent être considérés comme un "stock de noms" utilisés pour désigner des choses ou des concepts. (Les *URL* habituels sont des *URI)*.
- Ainsi, dans notre exemple, le document ***XML Schema*** peut être identifié naturellement par l'URI:

  http://www.oreilly.fr/XML_Schema.pdf
- Les prédicats (propriétés) sont également représentés par des ***URI***
  - **L'URI** du prédicat "*a pour auteur*" est l'élément *Creator* du schéma *Dublin Core*:

    http://purl.org/dc/elements/1.1/creator

« Bootstrap » de la syntaxe RDF

- RDF est conçu pour être un langage de description de faits, et de leurs relations mutuelles. Bien que cela soit l'expression d'un graphe, et non d'un arbre, il est un fait que tout graphe peut être porté par un arbre, ce qui autorise une expression de RDF en langage XML, avec la racine :

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

- S'agissant de description de faits, l'élément initial des expressions RDF est « Description »

```
<rdf:Description/>
```

- Une description est comme une histoire successive attachée à des faits introduits une première fois par un identifiant, puis cités ensuite

```
<rdf:Description rdf:ID="unURI"/>
<rdf:Description rdf:about="#unURI"/>
```

- Une description lie des faits au moyen de propriétés dont la signification ne peut être connue que par l'association de leur désignation à un vocabulaire défini. Par exemple celui du Dublin Core :

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

on exprime un fait en associant une propriété et sa valeur à ce que l'on se propose de décrire

```
<rdf:Description rdf:ID="unURIdeLivre">
<dc:Title>un titre<dc:Title>
</rdf:Description>
```

Une propriété peut avoir un contenu littéral ou référer un autre fait, qu'il faut bien lui aussi … décrire

```
<rdf:Description rdf:ID="unURIdeLivre">
<dc:author> <rdf:Description rdf:ID="unURIdAuteur"> <dc:author>
</rdf:Description>
```

telle est la façon dont s'exprime en XML un triplet Sujet/Prédicat/Objet du langage RDF

Ensembles, catégories, caractères, et toutes ces sortes de choses

Ainsi est le caractère des langages des humains,
de par la constitution même de leur cerveau,
pour parler et se souvenir des faits, ils leur faut les catégoriser !

```
<rdf:Description rdf:ID="unURIdeLivre">
<rdf:type rdf:resource="#livre"/>
</rdf:Description>
```

Ainsi sont les usages des humains,
Qui pour simplifier leur langage
Désignent des faits par leur catégories
Et ces catégories par des noms de caractérisiques !

```
<blind rdf:ID="Ray_Charles">
     <rdf:type rdf:resource="#singer"/>
</blind>
```

Ou, ce qui est sémantiquement équivalent :

```
<singer rdf:ID="Ray_Charles">
      <rdf:type rdf:resource="#blind"/>
</singer>
```

Formalisation EBNF (Extended Backus-Naur form) de la grammaire RDF

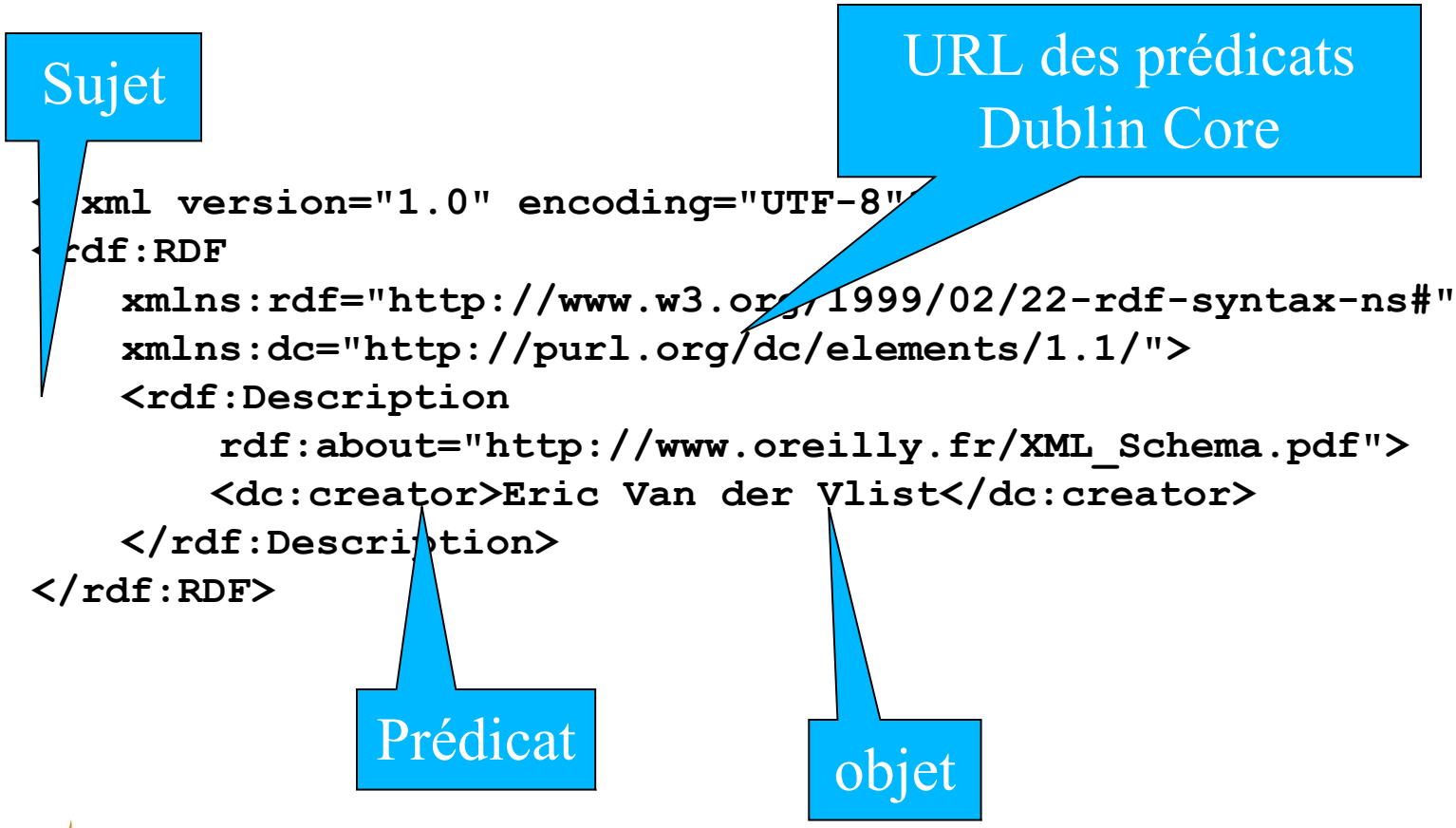
[1] RDF ::= ['<*rdf*:RDF>'] description* ['</*rdf*:RDF>']
[2] description ::= '<*rdf*:Description' idAboutAttr? '>' propertyElt* '</*rdf*:Description>'
[3] idAboutAttr ::= idAttr | aboutAttr
[4] aboutAttr ::= 'about="' URI-reference '"'
[5] idAttr ::= 'ID="' IDsymbol '"'
[6] propertyElt ::= '<' propName '>' value '</' propName '>' | '<' propName resourceAttr '/>'
[7] propName ::= Qname
[8] value ::= description | string
[9] resourceAttr ::= 'resource="' URI-reference '"‘
[10] Qname ::= [NSprefix ':'] name
[11] URI-reference ::= string, interpreted per [URI]
[12] IDsymbol ::= (any legal XML name symbol)
[13] name ::= (any legal XML name symbol)
[14] NSprefix ::= (any legal XML namespace prefix)
[15] string ::= (any XML text, with "<", ">", and "&" escaped)

[2a] description ::= '<*rdf*:Description' idAboutAttr? propAttr* '/>' | '<*rdf*:Description' idAboutAttr? propAttr* '>' propertyElt* '</*rdf*:Description>' | typedNode
[6a] propertyElt ::= '<' propName '>' value '</' propName '>' | '<' propName resourceAttr? propAttr* '/>'
[16] propAttr ::= propName '="' string '"' (with embedded quotes escaped) [17] typedNode ::= '<' typeName idAboutAttr? propAttr* '/>' | '<' typeName idAboutAttr? propAttr* '>' property* '</' typeName '>'

syntaxe XML de RDF

- Un sujet (ressource) peut posséder plusieurs prédicats (propriétés)

Ce qui se traduit en syntaxe *RDF*:



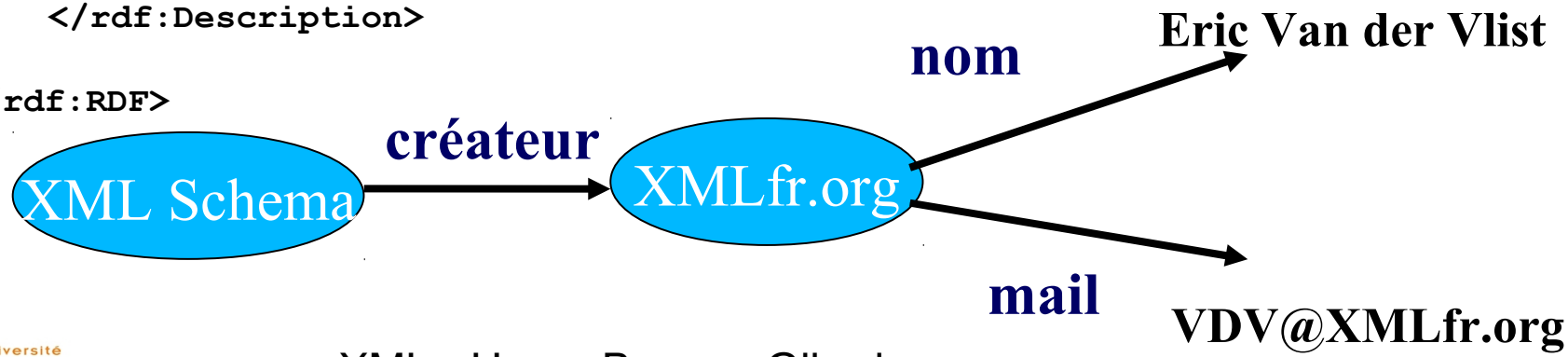
RDF : un moyen d'expression de graphes

- Les ressources décrites peuvent être associées, imbriquées:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:v="http://description.org/people/"
     xmlns:s="http://description.org/schema/">

     <rdf:Description about="http://www.oreilly.fr/XML_Schema.pdf">
       <dc:Creator>
          <rdf:Description about="http:XMLfr.org">
               <rdf:type resource="http://description.org/schema/Person"/>
               <v:Nom>Eric Van der Vlist</v:Nom>
               <v:Email>VDV@XMLfr.org</v:Email>
          </rdf:Description>
       </dc:Creator>
     </rdf:Description>

</rdf:RDF>
```



Les termes du Vocabulaire de base de RDF

- Entités de base du vocabulaire RDF
  http://www.w3.org/1999/02/22-rdf-syntax-ns
  - Resource :
    - n'importe quelle « chose » référée par une expression RDF, identifiée par une URI
  - Property :
    - N'importe quel aspect, caractéristique propre d'une *Resource*, sa relation à une autre *Resource* ;
    - est elle-même une sorte de *Resource*
  - Statement :
    - expression de la valeur d'une propriété nommée rattachée à une *Resource* donnée.
    - Elle est la réification d'un triplet sujet/prédicat/valeur

Schémas et espaces de noms

- Ce qui est crucial pour la compréhension des déclarations pour un humain l'est, aussi pour toute application de **RDF**, pour que le traitement en cours soit correct.

- Il est crucial que le rédacteur et le lecteur d'une déclaration comprennent de la même manière la signification des termes utilisés, tel que ***Créateur***, ***ApprouvéPar***, ***DroitDeCopie***. Toute confusion, tout recouvrement de concepts est prohibé.

- La ***signification*** en **RDF** est exprimée par la référence à un ***schéma***.

- un schéma est une sorte de dictionnaire. Un schéma définit les appellations à utiliser dans les déclarations **RDF** et leur associe des significations uniques.

- Une variété de formes de schéma peut être utilisée avec **RDF**, y compris une forme spécifique définie dans un document **XML** séparé [**RDF Schema**] qui possède quelques caractéristiques spécifiques pour aider l'automatisation des tâches utilisant **RDF**.

Des notions différentes

- Il faut faire la distinction entre:
  - *Schéma XML* qui exprime des contraintes sur la *structure* et la syntaxe XML de « documents »
  - *Schéma RDF* qui exprime des contraintes sur la *sémantique* des expressions d'un modèle *RDF*

- Notion de *schéma RDF*
  - Un ***schéma RDF*** permet de décrire un vocabulaire et une sémantique des types de propriétés utilisées par une communauté d'utilisateurs.
  - Un ***schéma RDF*** précise les propriétés valides pour une description *RDF* particulière, ainsi que les caractéristiques et contraintes du vocabulaire descriptif.
    Exemple : **le schéma RDF du Dublin Core, version 1.1**

Schema RDF

- Un schéma est l'endroit où les définitions et les restrictions d'usage pour les propriétés sont documentées.
- Afin d'éviter la confusion entre les définitions indépendantes -- et potentiellement conflictuelles -- du même terme, **RDF** utilise les possibilités des **espaces de noms XML**.
- Les espaces de noms sont simplement une manière de lier l'utilisation spécifique d'un mot dans un contexte au dictionnaire (schéma) où la définition est supposée se trouver.
- En **RDF**, chaque prédicat utilisé dans une déclaration doit être identifié avec exactement un espace de nom, ou un schéma.
- Cependant, un élément Description peut contenir des déclarations avec des prédicats issues de plusieurs schémas.

syntaxe abrégée

- **Description pleine:**

```
<rdf:Description rdf:ID='Henry' >
   <rdf:type resource='#Researcher' />
   <vehicle>
     <rdf:Description>
   <rdf:type resource='#Car' />
   <brand>Renault</brand>
   </rdf:Description>
   </vehicle>
</rdf:Description>
```

- **Syntaxe abrégée, avec exploitation des types :**

```
<Researcher rdf:ID='Henry'>
    <vehicle>
     <car brand='Renault'/>
   </vehicle>
</Researcher>
```

- **Autre forme de Syntaxe abrégée avec des attributs :**

```
<rdf:Description rdf:ID='Henry' rdf:type='#Researcher' />
```

Conteneurs

- Il est fréquemment nécessaire de faire référence à une collection de ressources ; par exemple, pour dire qu'un travail a été créé par plus d'une personne, ou pour lister des étudiants dans un cours, ou les modules d'un logiciel. Les conteneurs RDF sont utilisés pour contenir de telles listes de ressources ou de littéraux.

- RDF définit trois types d'objets conteneurs :
  - **Bag** Une liste non ordonnée de ressources ou de littéraux. *Bag* est utilisé pour déclarer qu'une propriété possède plusieurs valeurs et qu'il n'y a pas de sens pour l'ordre dans lequel elles sont données. *Bag* pourra être utilisé pour donner les nombres d'une liste de partie, où l'ordre de traitement des parties importe peu. Les valeurs identiques sont permises.

  - **Sequence** Une liste ordonnée de ressources ou de littéraux. *Sequence* est utilisé pour déclarer qu'une propriété a plusieurs valeurs et que l'ordre de ces valeurs a un sens. *Sequence* pourra être utilisé, par exemple, pour préserver un ordre alphabétique de valeurs. Les valeurs identiques sont permises.

  - **Alternative** Une liste de ressources ou de littéraux qui représentent des alternatives pour la valeur (unique) d'une propriété. *Alternative* pourra être utilisé pour fournir des traductions dans une autre langue pour le titre d'un travail, ou pour fournir une liste de sites Internet miroirs dans lesquels une ressource pourra être trouvée. Une application utilisant une propriété dont la valeur est une collection d'*Alternative* sait qu'elle peut choisir l'un des éléments de la liste lorque c'est approprié.

exemples

```
<rdf:Description rdf:about="http://www.recshop.fake/cd/Beatles">
   <cd:format>
    <rdf:Alt>
        <rdf:li>CD</rdf:li>
         <rdf:li>Record</rdf:li>
        <rdf:li>Tape</rdf:li>
    </rdf:Alt>
   </cd:format>
</rdf:Description>

<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:cd="http://www.recshop.fake/cd#">
   <rdf:Description rdf:about="http://www.recshop.fake/cd/Beatles">
    <cd:artist>
        <rdf:Seq>
            <rdf:li>George</rdf:li>
            <rdf:li>John</rdf:li>
            <rdf:li>Paul</rdf:li>
            <rdf:li>Ringo</rdf:li>
        </rdf:Seq>
    </cd:artist>
   </rdf:Description>
</rdf:RDF>
```

Les collections fermées

- L'attribut **`rdf:parseType="Collection"`**

Définit une liste fermée non ordonnée d'éléments

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:cd="http://recshop.fake/cd# "
        xmlns:bt="http://recshop.fake/cd/Beatles#"
        xmlns="http://recshop.fake/cd/Beatles#" >
   <rdf:Description rdf:about="http://recshop.fake/cd/Beatles">
    <cd:artist rdf:parseType="Collection">
        <rdf:Description rdf:about="#George"/>
        <rdf:Description rdf:about="#John"/>
        <rdf:Description rdf:about="#Paul"/>
        <rdf:Description rdf:about="#Ringo"/>
    </cd:artist>
   </rdf:Description> </rdf:RDF>
```

N3 : Une autre modalité d'expression pour RDF

- La notation N3 est simple :

  sujet verbe objet ponctuation

  <#pat> <#knows> <#jo> .

  (Tout y est URL ; le # situe la ressource dans l'URI du document courant, dont le namespace est par défaut)
- Seul l'objet peut être littéral :

  <#pat> <#aged> 24.
- De façon équivalente plus lisible :

  <#pat> is <#aged> 24.
- La ponctuation ";" et"," permet de mettre le sujet en facteur commun pour le verbe et pour l'objet respectivement

  <#pat> is <#child>of <#jo> ; is <#parent> of <#fred>.
- Les [ ] identifient des ressource à partir de propriétés

  <#pat> has<#child> [<#aged> 21], [<#aged> 24],
- Autre exemple d'expression :

  [<#name> "Pat"; <#age> 24; <#eyecolor> "blue"].

Le document courant se cite ainsi

<> <#title> "A simple example of N3".

- **Cependant le rattachement à un espace de nom formel est préférable :**

  <> <http://purl.org/dc/elements/1.1/title> " titre de ce document ".

N3 : Une autre modalité d'expression pour RDF (suite)

- Le mécanisme de prefixe pour référer l'URI d'un espace de noms :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<> dc:title " titre de ce document ".
```

  (où le fait que le nom soit colonisé dispense de <>)
- Pour déclarer l'usage d'un prefixe par défaut : `@prefix : <#> .`
- Par la suite on peut écrire : `:pat :aged 24.  au lieu de <#pat> <#aged> 24 .`
- L'équivalence de vocabulaire s'exprime par le signe =

```
:woman  =  foo:human_female .
:Title a rdf:Property; = dc:title .
```

N3 permet nativement d'écrire des règles, ce que la notation XML de RDF ne permet pas :

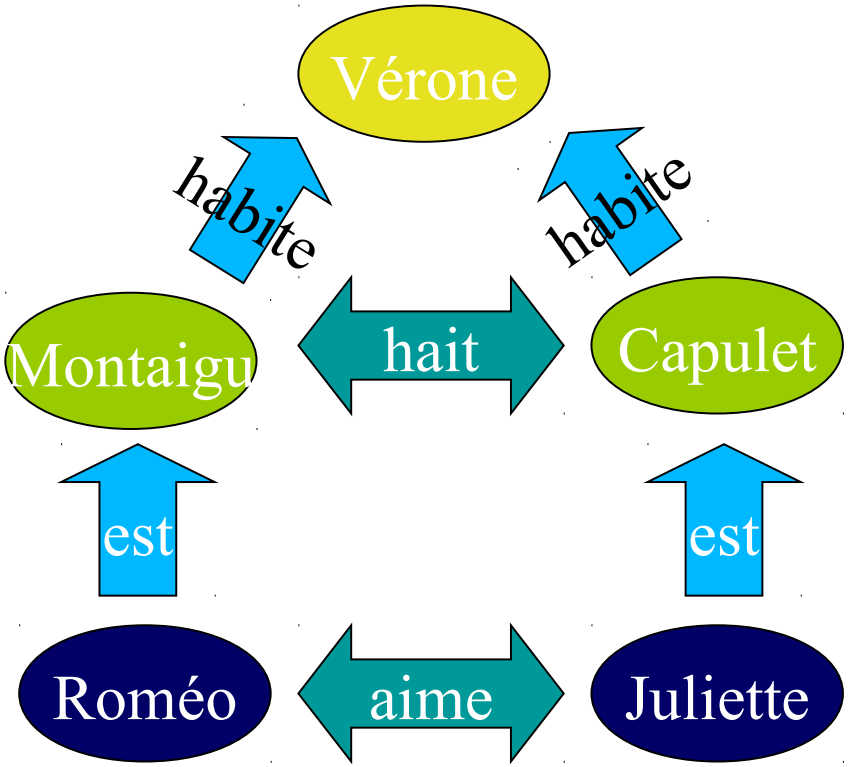
```
@prefix : <#>.
# Tous les hommes sont mortels
{?x a :homme} => {?x a :mortel}.
# Socrate est un homme
:Socrate a :homme.
```

Ceci permet à un moteur d'inférence de déduire

```
:Socrate a :mortel.
```

Exercice

- Décrire en RDF les relations qui se sont tissées à Vérone entre Montaigu et Capulets, Roméo et Juliette…



```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:a="http://www.MonVocabulaire#"
   xmlns:kb="http://www.MonVocabulaire#"
   xmlns:rdfs="&rdfs;">
<kb:Personne rdf:about="http://www.MonVocabulaire#KB_155146_Instance_4"
   kb:Label="Romeo"
   rdfs:label="Romeo"/>
<kb:Personne
  rdf:about="http://www.MonVocabulaire#KB_155146_Instance_6"
   kb:Label="Juliette"
  rdfs:label="Juliette"/>
<kb:Famille
  rdf:about="http://www.MonVocabulaire#KB_155146_Instance_7"
   kb:Label="Montaigu"
  rdfs:label="Montaigu"/>
<kb:Famille
   rdf:about="http://www.MonVocabulaire#KB_155146_Instance_8"
   kb:Label="Capulet"
  rdfs:label="Capulet"/>
<kb:Ville
  rdf:about="http://www.MonVocabulaire#KB_155146_Instance_9"
  kb:Label="Verone"
   rdfs:label="Verone"/>
<kb:aime
  rdf:about="http://www.MonVocabulaire#Shakespeare_Instance_0"
   rdfs:label="Shakespeare_Instance_0">
  <a:_to rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_4"/>
  <a:_from rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_6"/>
</kb:aime>
```

```
<kb:aime
  rdf:about="http://www.MonVocabulaire#Shakespeare_Instance_1"
   rdfs:label="Shakespeare_Instance_1">
  <a:_from rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_4"/>
  <a:_to rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_6"/>
</kb:aime>
<kb:hait
  rdf:about="http://www.MonVocabulaire#Shakespeare_Instance_2"
  rdfs:label="Shakespeare_Instance_2">
 <a:_to rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_7"/>
 <a:_from rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_8"/>
</kb:hait>
<kb:hait
  rdf:about="http://www.MonVocabulaire#Shakespeare_Instance_3"
   rdfs:label="Shakespeare_Instance_3">
  <a:_from rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_7"/>
  <a:_to rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_8"/>
</kb:hait>
<kb:habite
  rdf:about="http://www.MonVocabulaire#Shakespeare_Instance_4"
  rdfs:label="Shakespeare_Instance_4">
  <a:_from rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_8"/>
  <a:_to rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_9"/>
</kb:habite>
<kb:habite
  rdf:about="http://www.MonVocabulaire#Shakespeare_Instance_5"
  rdfs:label="Shakespeare_Instance_5">
  <a:_from rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_7"/>
  <a:_to rdf:resource="http://www.MonVocabulaire#KB_155146_Instance_9"/>
</kb:habite>
</rdf:RDF>
```

RDFS et son méta-modèle

- **RDFS** est la première initiative de description de l'organisation de ressources **RDF**
- **RDFS** est une extension de **RDF**, dans la mesure où **RDF** définit un jeu de classes et de propriétés de base.
- Au commencement est la « ressource » dont tout est issu…
  - **rdfs:Resource** : classe dont toute classe est un sous-type
  - **rdfs:Class**, instance d'elle-même, sous-type de rdfs:Resource
  - **rdfs:Literal** instance de rdfs:Class, sous-type de rdfs:Resource
  - **rdfs:Datatype** est la classe des attributs. Ses instances correspondent au modèle d'une donnée décrit dans la spécification des concepts de RDF.
    rdfs:Datatype est à la fois une instance et une sous-classe de rdfs:Class.
    Chaque instance de rdfs:Datatype est un sous-type de rdfs:Literal.
  - **rdfs:Property** est la classe des propriétés RDF, instance de rdf:Class
  - **rdfs:range** et **rdfs:domain** sont des instances de **rdfs:Property** attachées à **rdfs:Property**, de façon récursive :
    le domaine rdfs:domain de rdfs:range et de rdfs:domain est rdfs:Property…
  - **rdf:type** est une instance de **rdf:property** utilisée pour définir une ressource comme instance d'une classe.

RDFS (suite)

- **rdfs:subClassOf** est une instance de **rdf:property** qui définit que les instances d'une classe sont instances d'une autre classe dont elle hérite

- **rdfs:subPropertyOf** est une instance de **rdf:property** qui définit que toutes les ressources indiquées par une propriété sont aussi ressources d'une autre propriété dont elle hérite.

- **rdfs:label** est une instance de **rdf:property** utilisée pour donner d'une ressource une version lisible par un humain.

- **rdfs:comment** est une instance de **rdf:property** utilisée pour documenter une ressource par un commentaire

Class name	comment
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:XMLLiteral	The class of XML literals values.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements.
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.
rdf:List	The class of RDF Lists.

Property name	comment	domain	range
rdf:type	The subject is an instance of a class.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF list after the first item.	rdf:List	rdf:List
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values (see the RDF Primer for an example of its usage).	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:a="http://www.MonVocabulaire#"
         xmlns:kb="http://www.MonVocabulaire#"
         xmlns:rdfs="&rdfs;">
<rdfs:Class rdf:about="http://www.MonVocabulaire#Famille"
         rdfs:label="Famille">
        <rdfs:subClassOf rdf:resource="http://www.MonVocabulaire#Objet_tragique"/>
</rdfs:Class>
<rdf:Property rdf:about="http://www.MonVocabulaire#Label"
         a:maxCardinality="1"
         rdfs:label="Label">
        <rdfs:domain rdf:resource="http://www.MonVocabulaire#Famille"/>
        <rdfs:domain rdf:resource="http://www.MonVocabulaire#Personne"/>
        <rdfs:domain rdf:resource="http://www.MonVocabulaire#Ville"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdfs:Class rdf:about="http://www.MonVocabulaire#Objet_tragique"
         rdfs:label="Objet_tragique">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://www.MonVocabulaire#Personne"
         rdfs:label="Personne">
        <rdfs:subClassOf rdf:resource="http://www.MonVocabulaire#Objet_tragique"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://www.MonVocabulaire#Ville"
         rdfs:label="Ville">
        <rdfs:subClassOf rdf:resource="http://www.MonVocabulaire#Objet_tragique"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://www.MonVocabulaire#aime"
         rdfs:label="aime">
        <rdfs:subClassOf rdf:resource="http://www.MonVocabulaire#_directed_binary_relation"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://www.MonVocabulaire#habite"
         rdfs:label="habite">
        <rdfs:subClassOf rdf:resource="http://www.MonVocabulaire#_directed_binary_relation"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://www.MonVocabulaire#hait"
         rdfs:label="hait">
        <rdfs:subClassOf rdf:resource="http://www.MonVocabulaire#_directed_binary_relation"/>
</rdfs:Class>
</rdf:RDF>
```

OWL

- **OWL** : acronyme de Web Ontology Language

(c'est plus chouette!)

« *le langage **OWL** est conçu comme une extension de RDF et RDFS ; OWL est destiné à la description de classes (par des constructeurs) et de types de propriétés. De ce fait, il est plus expressif que RDF et RDFS, auxquels est reprochée une insuffisance d'expressivité due à la seule définition des relations entre objets par des assertions. OWL apporte aussi une meilleure intégration, une évolution, un partage et une inférence plus facile des ontologies* ».

(source ***Wikipedia***)

  - Aux concepts de **classe**, de **ressource**, de **littéral** et de propriétés des sous-classes, de sous-propriétés, de champs de valeurs et de domaines d'application déjà présents dans RDFS, OWL ajoute les concepts de **classes équivalentes**, de **propriété équivalente**, **d'égalité** de deux ressources, de leurs **différences**, du **contraire**, de **symétrie** et de **cardinalité**...

OWL : 3 niveaux de conformité

- Liés à ce que l'on peut exprimer à un niveau et aux garanties d'inférence

OWL Full
- Pouvoir tout exprimer mais sans garantir des *raisonnements en un temps fini*

OWL DL (Description Logic)
- Avoir le maximum d'expressivité possible
- Tout en pouvant garantir la "décidabilité" de l'ontologie : les calculs se feront en un temps "fini"

Méthode :
- Des restrictions sur l'utilisation des constructeurs
  - *Une classe ne peut être instance d'une autre classe*

OWL Lite
- Pouvoir exprimer des thesaurus et autres index
- Pouvoir construire aisément des outils

Méthode :
- Hiérarchisation de classes
- Contraintes simples

Classes OWL

- **Owl:Thing** est la classe mère de toutes les classes
- **Owl:noThing** est classe fille de toutes les classes

- Une classe OWL peut être définie
  - par une référence (URI)

```
<owl:Class rdf:ID="Toto"  rdf:resource= "http://toto.org">
```

  - par l'énumération de ses instances:

```
<owl:Class rdf:ID="continents">
 <owl:oneOf rdf:parseType="Collection">
 <owl:Thing rdf:about="#Eurasie"/>
 <owl:Thing rdf:about="#Afrique"/>
 <owl:Thing rdf:about="#Amerique_du_Nord"/>
 <owl:Thing rdf:about="#Amerique_du_Sud "/>
 <owl:Thing rdf:about="#Australie"/>
 <owl:Thing rdf:about="#Antarctique"/>
 </owl:oneOf>
</owl:Class>
```

Classe OWL

- Une classe OWL peut être définie,
  - par ses propriétés (définition intensionnelle)
    - Types de propriétés :
      `owl:allValuesFrom,owl:someValuesFrom`
    - Valeurs de propriétés :
      `owl:hasValue`
    - Cardinalité de propriétés :
      `owl:maxCardinality, owl:minCardinality, owl:Cardinality`

Exemple : classe des ressources qui ont au moins une propriété #author de type #novelist

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#author" />
  <owl:allValuesFrom rdf:resource="#novelist" />
</owl:Restriction>
```

Classe OWL

- Une classe OWL peut être définie
  - comme *union*, *intersection*, *complément* d'autres classes

```
<owl:Class>
 <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="continents">
   <owl:Class>
       <owl:oneOf rdf:parseType="Collection">
           <owl:Thing rdf:about="#Europe" />
           <owl:Thing rdf:about="#Asie" />
           <owl:Thing rdf:about="#Afrique" />
       </owl:oneOf>
   </owl:Class>
 </owl:intersectionOf>
</owl:Class>
```

Classe OWL

- **`rdfs:subClassOf`** :
  l'extension d'une classe est incluse dans l'extension de l'autre
- **`owl:equivalentClass :`**

  les deux classes ont la même extension, mais ne désignent pas le même concept.

  <footballTeam owl:equivalentClass us:soccerTeam />
- **`owl:disjointWith :`**
  deux classes sont disjointes.

Propriétés

- RDF Schema définit les notions:
  - **rdfs:subPropertyOf, rdfs:domain et rdfs:range**

- OWL enrichit les relations entre propriétés :
  - **owl:equivalentProperty :**
    les deux propriétés ont la même extension, mais ne sont pas identiques
  - **owl:inverseOf :**
    une propriété est l'inverse de l'autre.

```
<owl:ObjectProperty rdf:ID="enfant">
  <owl:inverseOf rdf:resource="#parent"/>
</owl:ObjectProperty>
```

  - Contraintes de cardinalité :
    owl:FunctionalProperty (mono-valuées), et
    owl:InverseFunctionalProperty
  - Contraintes logiques :
    owl:SymmetricProperty (exemple époux)
    owl:TransitiveProperty (exemple ascendant)

Exemple (ontologie)

```
<owl:ObjectProperty rdf:ID="habite">
   <rdfs:domain rdf:resource="#famille"/>
   <rdfs:range rdf:resource="#ville"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:about="#a_pour_amant">
   <owl:inverseOf rdf:resource="#a_pour_amante"/>
   <rdfs:range rdf:resource="#garçon"/>
   <rdfs:domain rdf:resource="#fille"/>
 </owl:ObjectProperty>
 <owl:DatatypeProperty rdf:ID="nom">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
   <rdfs:domain>
     <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#personne"/>
         <owl:Class rdf:about="#famille"/>
         <owl:Class rdf:about="#ville"/>
       </owl:unionOf>
     </owl:Class>
   </rdfs:domain>
 </owl:DatatypeProperty>
```

Exemple ontologie (suite)

```
<owl:ObjectProperty rdf:ID="a_pour_amante">
   <rdfs:range rdf:resource="#fille"/>
   <owl:inverseOf>
     <owl:ObjectProperty rdf:ID="a_pour_amant"/>
   </owl:inverseOf>
   <rdfs:domain rdf:resource="#garçon"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="hait_de_génération_en_génération">
   <rdfs:range rdf:resource="#famille"/>
   <rdfs:domain rdf:resource="#famille"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="appartient_à_la_famille">
   <rdfs:domain rdf:resource="#personne"/>
   <rdfs:range rdf:resource="#famille"/>
   <owl:inverseOf>
     <owl:ObjectProperty rdf:ID="inverse_of_appartient_à_la_famille"/>
   </owl:inverseOf>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:about="#inverse_of_appartient_à_la_famille">
   <owl:inverseOf rdf:resource="#appartient_à_la_famille"/>
   <rdfs:range rdf:resource="#personne"/>
   <rdfs:domain rdf:resource="#famille"/>
 </owl:ObjectProperty>
```

Exemple (ontologie fin)

```
<owl:ObjectProperty rdf:ID="habite">
   <rdfs:domain rdf:resource="#famille"/>
   <rdfs:range rdf:resource="#ville"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#a_pour_amant">
   <owl:inverseOf rdf:resource="#a_pour_amante"/>
   <rdfs:range rdf:resource="#garçon"/>
   <rdfs:domain rdf:resource="#fille"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="nom">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
   <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#personne"/>
      <owl:Class rdf:about="#famille"/>
      <owl:Class rdf:about="#ville"/>
     </owl:unionOf>
    </owl:Class>
   </rdfs:domain>
  </owl:DatatypeProperty>
```

Exemple (peuplement de l'ontologie)

```
<personne rdf:ID="personne_9">
   <a_pour_amant>
    <garçon rdf:ID="personne_7">
     <nom rdf:datatype="http://www.w3.org/2001/XMLSchema#string" >Roméo</nom>
     <a_pour_amante rdf:resource="#personne_9"/>
     <rdf:type rdf:resource="#personne"/>
     <appartient_à_la_famille>
      <famille rdf:ID="Montaigu">
       <habite>
        <ville rdf:ID="ville_13">
         <nom rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Vérone</nom>
        </ville>
       </habite>
       <inverse_of_appartient_à_la_famille rdf:resource="#personne_7"/>
      </famille>
     </appartient_à_la_famille>
    </garçon>
   </a_pour_amant>
   <nom rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Juliette</nom>
   <rdf:type rdf:resource="#fille"/>
   <appartient_à_la_famille>
    <famille rdf:ID="famille_11">
     <inverse_of_appartient_à_la_famille rdf:resource="#personne_9"/>
     <habite rdf:resource="#ville_13"/>
     <hait_de_génération_en_génération rdf:resource="#Montaigu"/>
     <nom rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Capulet</nom>
    </famille>
   </appartient_à_la_famille>
  </personne>
</rdf:RDF>
```

SPARQL: Simple Protocol And Rdf Query Language

- Un langage de requête pour ressources RDF, semblable à SQL dans son expression, produisant des résultats exprimés en XML.

Exemple de ressource RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rss="http://purl.org/rss/1.0/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
        <foaf:Person rdf:about="http://example.net/Paul_Dupont">
              <foaf:name>Paul Dupont</foaf:name>
              <foaf:img rdf:resource="http://example.net/Paul_Dupont.jpg"/>
              <foaf:knows
                 rdf:resource="http://example.net/Pierre_Dumoulin"/>
        </foaf:Person>
        <foaf:Person rdf:about="http://example.net/Pierre_Dumoulin">
              <foaf:name>Pierre Dumoulin</foaf:name>
              <foaf:img
                 rdf:resource="http://example.net/Pierre_Dumoulin.jpg"/>
        </foaf:Person>
        <foaf:Image rdf:about="http://example.net/Paul_Dupont.jpg">
              <dc:description>Photo d'identité de Paul
                 Dupont</dc:description>
        </foaf:Image>
        <foaf:Image rdf:about="http://example.net/Pierre_Dumoulin.jpg">
              <dc:description>Photo d'identité de Pierre
                 Dumoulin</dc:description>
        </foaf:Image>
</rdf:RDF>
```

Exemple de Requête SPARQL

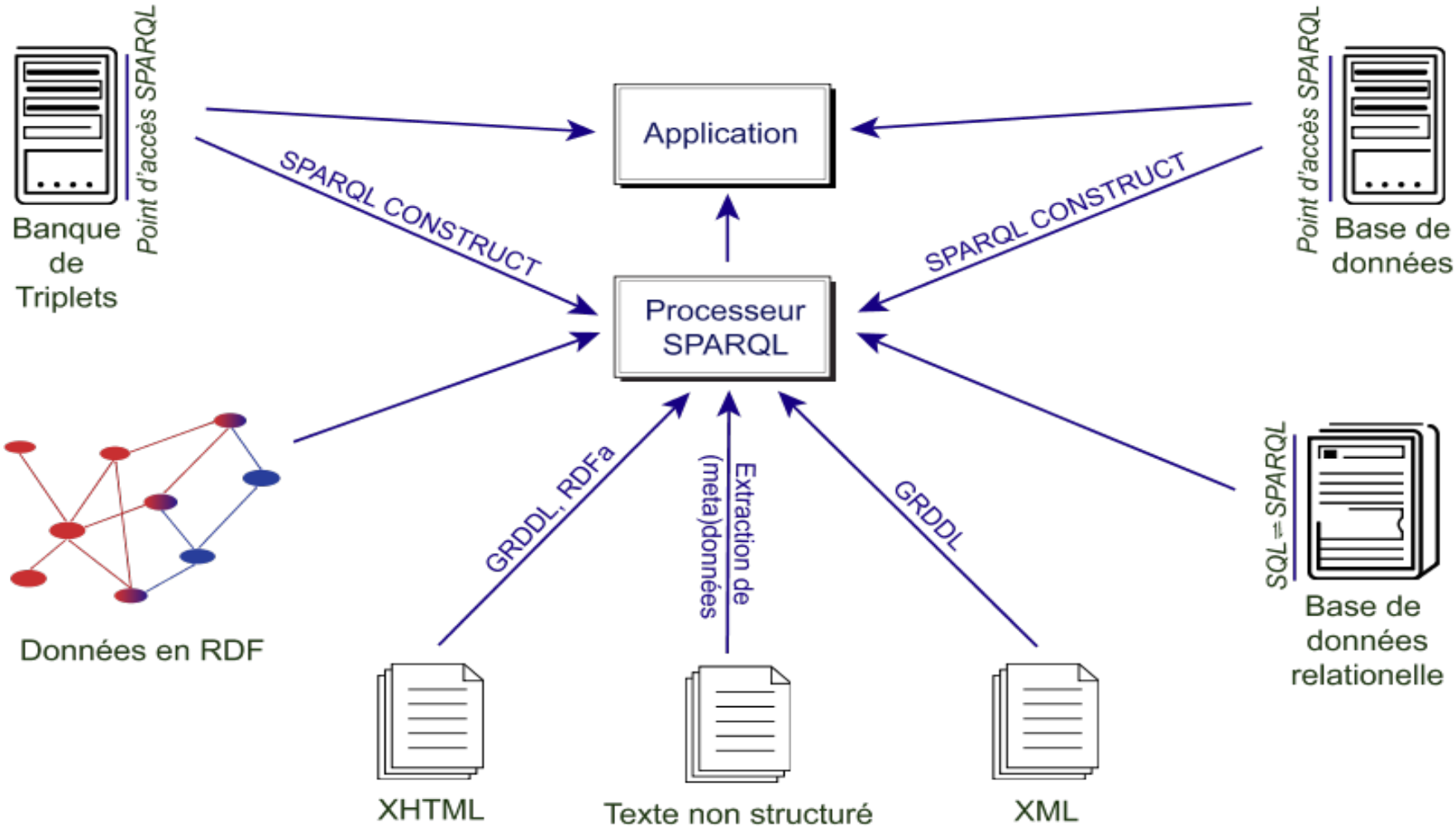
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?nom ?image ?description
WHERE {
    ?personne rdf:type foaf:Person .
    ?personne foaf:name ?nom .
    ?image rdf:type foaf:Image .
    ?personne foaf:img ?image .
    ?image dc:description ?description
}
```

(Source: Wikipedia)

Résultat de requête SPARQL (source Wikipedia)

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
	<head>
		<variable name="nom"/>
		<variable name="image"/>
		<variable name="description"/>
	</head>
	<results ordered="false" distinct="true">
		<result>
			<binding name="nom">
				<literal>Pierre Dumoulin</literal>
			</binding>
			<binding name="image">
				<uri>http://example.net/Pierre_Dumoulin.jpg</uri>
			</binding>
			<binding name="description">
				<literal>Photo d'identité de Pierre Dumoulin</literal>
			</binding>
		</result>
		<result>
			<binding name="nom">
				<literal>Paul Dupont</literal>
			</binding>
			<binding name="image">
				<uri>http://example.net/Paul_Dupont.jpg</uri>
			</binding>
			<binding name="description">
				<literal>Photo d'identité de Paul Dupont</literal>
			</binding>
		</result>
	</results>
</sparql>
```

Sparql : point d'unification



Décrire pour Raisonner

Ontologie

En Intelligence Artificielle et pour le web, une ontologie est un modèle d'information qui définit formellement les termes et les relations entre termes.

- Ces relations permettent de raisonner sur les objets respectant ce modèle
- Les termes représentent des concepts qui sont organisés dans un graphe avec :
  - Des relations sémantiques
  - Des contraintes qui sont autant de règles d'inférence.

Des ressources (individus/instances) sont liées à une ontologie

- Ils sont en conformité avec l'ontologie
- L'ontologie permet d'inférer pour découvrir de nouvelle propriétés des individus

L'inférence permet de d'agréger des bases d'individus du "monde ouvert"

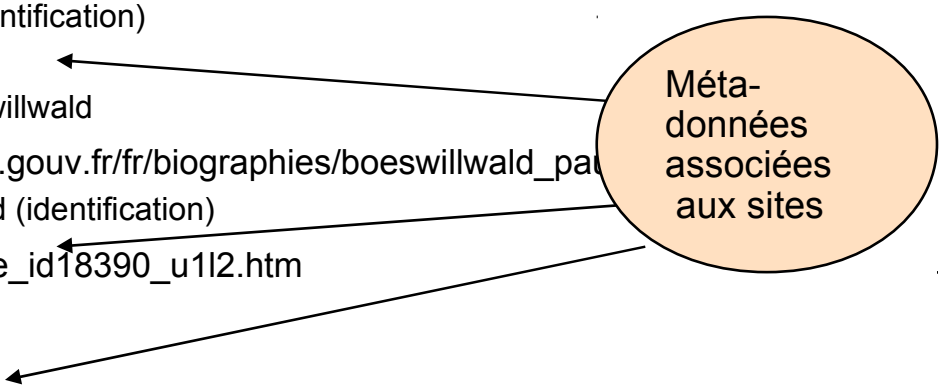
- En inférant sur les équivalences entre les différentes ontologies dont les individus sont issus.

Les présupposés du web sémantique

- AAA : "Anyone can say Anything about Any topic"
- Un monde ouvert
  - Il y a certainement plus d'information que celle que l'on connait
  - Il ne peut pas manquer une information : on ne l'a pas encore "découverte"
    - "Un parent à au moins un enfant"
    - Cela ne veut pas dire que je sais exhiber l'enfant de Jean. En revanche, je sais qu'il existe
  - Les informations trouvées peuvent être contradictoires
- Les ressources ne sont pas nommées de façon unique
  - "Le philosophe rieur", "Démocrite"
  - Ou exister par leur propriétés : "l'inventeur de l'atome"
  - possibilité
    - D'inférer des ressources identiques
      - Pierre A pour mère Estelle. Pierre a pour mère Jeanne.
      - Estelle et Jeanne sont les mêmes personnes
    - De déceler des contradictions

Opportunité

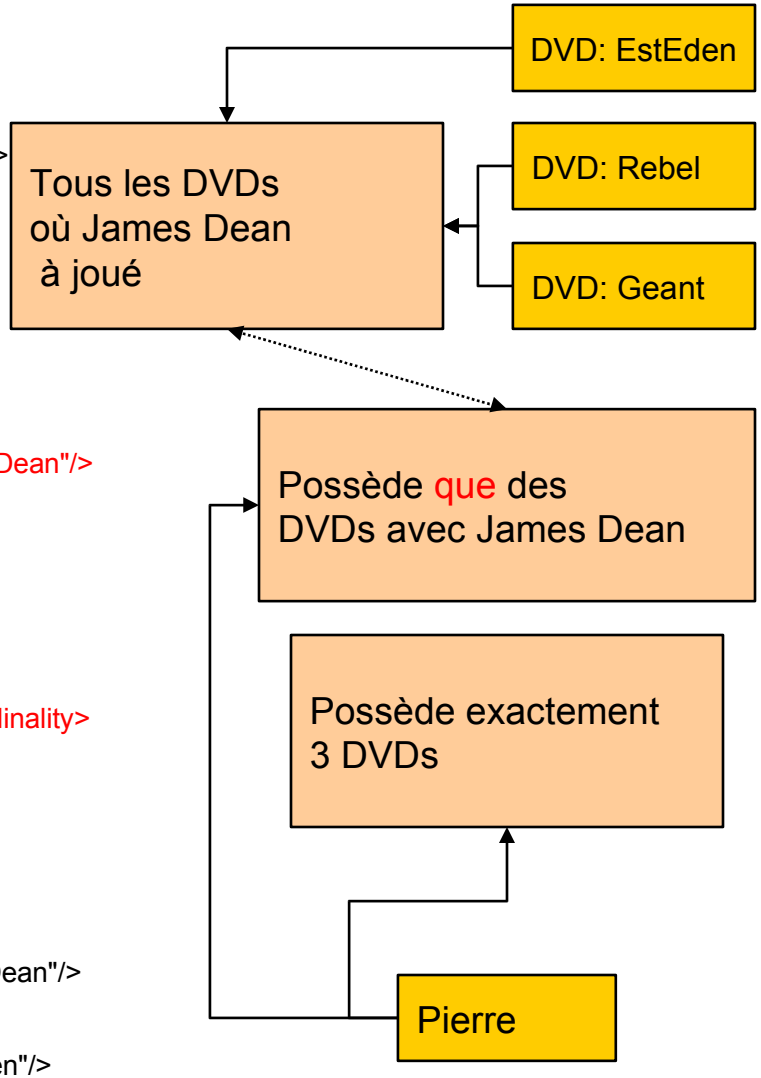
- moteur de recherche sémantique
  - http://web.mit.edu/museum/ware/viollet_le_duc.html
    - Le document concerne Viollet le Duc (identification)
    - Viollet le Duc est un architecte
    - Viollet le Duc a comme élève Paul Boeswillwald
  - http://www.mediatheque-patrimoine.culture.gouv.fr/fr/biographies/boeswillwald_pa
    - Le document concerne Paul Boeswillwald (identification)
  - http://www.musee-moyenage.fr/pages/page_id18390_u1l2.htm
    - A comme nom Hotel de Cluny
    - Restauré par Paul Boeswillwald
    - A comme lieu Paris
    - …



- Question : trouver des informations sur des monuments à Paris restaurés par un architecte élève de Viollet le Duc
  - Triple clé:
    - Identifier (Viollet Le Duc, Paul Boeswillwald, etc.)
      - Définir des équivalences entre ressources
    - Normaliser les concepts (personne, site, métier, a comme élève, etc.)
      - Définir des équivalences entre concepts si nécessaire
    - Inférer
      - Si Paul Boeswillwald est élève de Viollet Le Duc ; Si Viollet Le Duc est architecte
      - Paul Boeswillwald est architecte …
  - En utilisant des descriptions agrégées de différentes bases du Web

OWL : Un langage de modélisation construit pour inférer

- Définition des classes
  - <owl:Class rdf:ID="DVDAvecJamesDean">
    <owl:equivalentClass>
    <owl:Class>
    <owl:oneOf rdf:parseType="Collection">
    <rdf:Description rdf:about="&www;EstEden"/>
    <rdf:Description rdf:about="&www;Rebel"/>
    <rdf:Description rdf:about="&www;Geant"/>
    </owl:oneOf>
    </owl:Class>
    </owl:equivalentClass>
    </owl:Class>
  - <owl:Class rdf:ID="SeulementProprietaireDeDVDAvecJamesDean">
    <rdfs:subClassOf>
    <owl:Restriction>
    <owl:onProperty rdf:resource="#possedeDVD"/>
    <owl:allValuesFrom rdf:resource="#DVDAvecJamesDean"/>
    </owl:Restriction>
    </rdfs:subClassOf>
    </owl:Class>
  - <owl:Class rdf:ID="ProprietaireDe3DVD">
    <rdfs:subClassOf>
    <owl:Restriction>
    <owl:onProperty rdf:resource="#possedeDVD"/>
    <owl:cardinality
    rdf:datatype="&xsd;nonNegativeInteger">3</owl:cardinality>
    </owl:Restriction>
    </rdfs:subClassOf>
    </owl:Class>
- Inférons
  - Si
    <owl:Thing rdf:ID="Pierre">
    <rdf:type rdf:resource="#ProprietaireDe3DVD"/>
    <rdf:type rdf:resource="#SeulementProprietaireDeDVDAvecJamesDean"/>
    </owl:Thing>
  - Alors
    <owl:Thing rdf:about="#Pierre"><dvd:possedeDVD rdf:about="&www;EstEden"/>



Semantic Web Rule Language (swrl)

SWRL (Semantic Web Rule Language) est un langage de règles pour le web sémantique, combinant des sous-langages OWL (OWL DL et OWL Lite) avec ceux du Rule Markup Language (Unary/Binary Datalog).

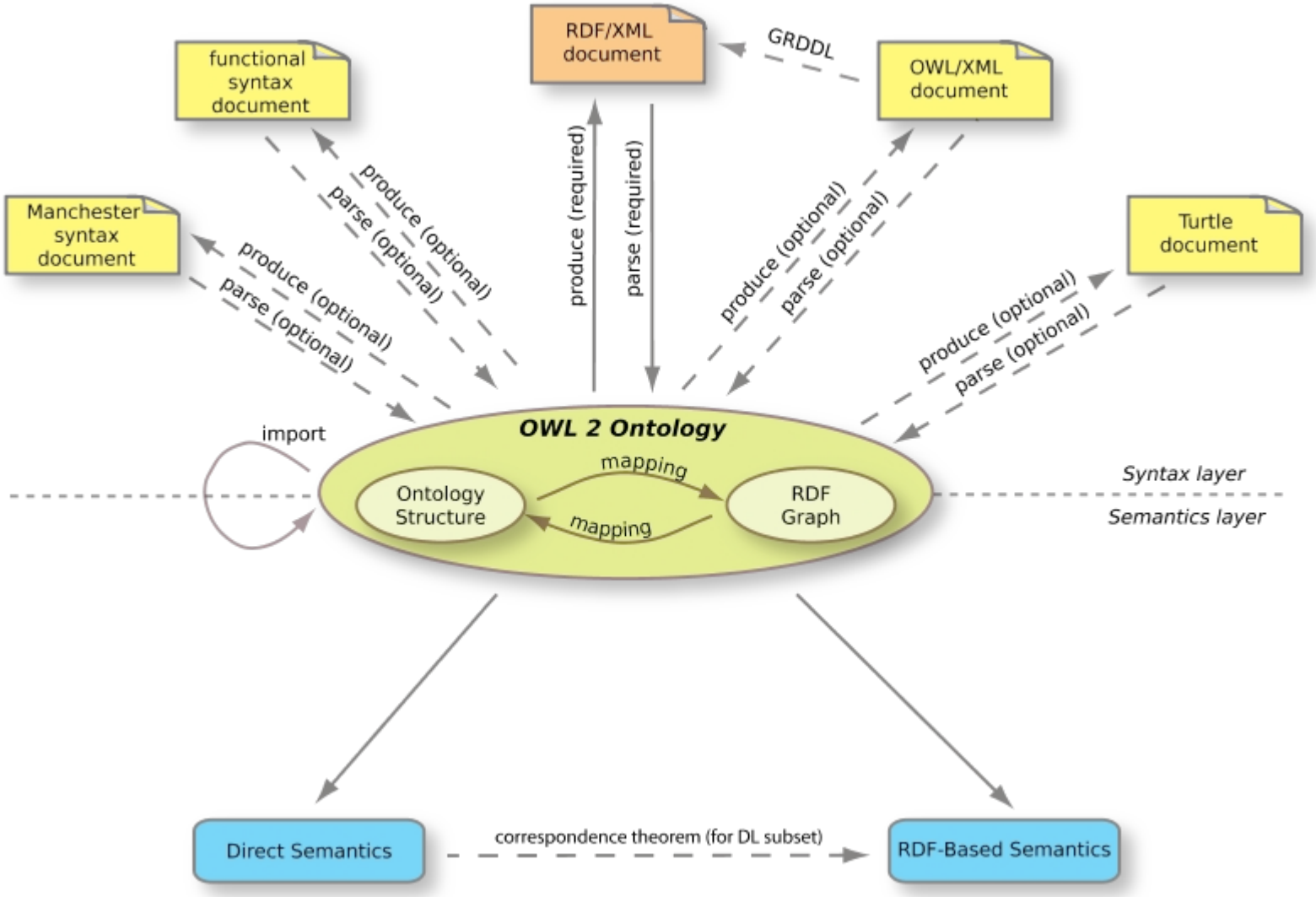
Exemple d'une règle de parenté telle que:
hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

```
<ruleml:imp> <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
     <swrlx:individualPropertyAtom swrlx:property="hasParent">
 <ruleml:var>x1</ruleml:var>
         <ruleml:var>x2</ruleml:var>
     </swrlx:individualPropertyAtom>
<swrlx:individualPropertyAtom swrlx:property="hasBrother">
<ruleml:var>x2</ruleml:var>
         <ruleml:var>x3</ruleml:var>
     </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
     <swrlx:individualPropertyAtom swrlx:property="hasUncle">
          <ruleml:var>x1</ruleml:var>
         <ruleml:var>x3</ruleml:var>
      </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Propositions d'évolutions en cours : OWL 2.00

1. Des facilités syntaxiques, quelques assertions plus concises à écrire,
2. Des nouvelles sortes d'assertions pour étendre l'expressivité ,
3. Un support étendu des Datatypes,
4. Des fonctionnalités simples de meta modelisation,
5. Des extensions des possibilités d'annotation,
6. Des modalités supplémentaires de sérialisation
7. Quelques autres innovations, et fonctionnalités mineures.

La structure de OWL 2



Quelques outils pour passer de la théorie à la pratique

- « Protégé » de l'université de Stanford :
  - **http://protege.stanford.edu/**
- « Swoop » de l'université du Maryland :
  - **http://code.google.com/p/swoop/**
- « KAON » de l'université de Karlsuhe :
  - **http://www.sourceforge.net/projects/kaon**
- « OntoEdit » de l'université de Karlsuhe :
  - **http://www.ontoknowledge.org/tools/ontoedit.shtml**
- « XMLSpy » outil du commerce :
  - **http://www.altova.com/download_components.html**

Perspective, en guise de conclusion

” People keep asking what Web 3.0 is.

I think maybe when you've got an overlay of scalable vector graphics - everything rippling and folding and looking misty - on Web 2.0 and access to a semantic Web integrated across a huge space of data, you'll have access to an unbelievable data resource. ”

Tim Berners-Lee,

Fin du module