

Web Sémantique

Master Informatique & Télécommunication
Fac Sciences Rabat
Année Universitaire : 2018/2019

Pr. Abderrahim EL QADI
Département Informatique
EST de Salé-Université Mohammed V de Rabat

Plan

1. Introduction
2. Sémantique : Représentation des concepts
3. Architecture du Web sémantique
 - La couche URI/IRI,
 - Langage XML,
 - Modèle de données RDF, et vocabulaire RDFS,
 - Langage OWL,
4. Structure d'une ontologie OWL DL
5. Langage SPARQL

Références

- Livre Web sémantique et modélisation ontologique (avec G-OWL), guide du développeur Java sous Eclipse, Edition eni.
- Livre Déployer un projet Web 2.0, Anticiper le Web sémantique (Web 3.0). Gbriel Képéklian, Jean-Louis Lequeux. Editions Eyrolles.
- Book Web Data Management and Distribution. <http://webdam.inria.fr/textbook>
- Polycopie, Raisonnement dans le web sémantique. H. Meziane
- Polycopie, SPARQL Langage d'interrogation du web sémantique. Anne-Cécile Caron.
- Introduction à OWL, un langage XML d'ontologies Web, http://lacot.org/public/introduction_a_owl.pdf
- Ontologies Langage OWL Application à la formalisation des concepts de description d'IMGT-ONTOLOGY avec l'éditeur Protégé (http://www.imgt.org/IMGTeDucation/Enseignements/_FR/archives/Ontologie_Protege_180511.pdf)
- Designing ETL Processes Using Semantic Web Technologies. Dimitrios Skoutas, Alkis Simitsis. DOLAP'06, Arlington, Virginia, USA.
- Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. Khouri S., B. Ilyes, Bellatreche L., Sardet E., J. Stéphane, Baron M.. Computers in Industry(2012)799–812.
- <http://www.professeurs.polymtl.ca/michel.gagnon/Publications/tutorielSWIG04.pdf>
- SPARQL, Langage d'interrogation du web sémantique. Anne-Cecile Caron.

1. Introduction

1.1. Web actuel

- Le Web actuel est une collection de document HTML, les documents ne sont affichés que par le Web browser.

```
<html>
<head>
  <title>....</title>
</head>
<body>
  <p>.....</p>
  ....
</body>
</html>
```

- Une des restrictions de HTML est qu'il est très limité sur le plan sémantique : le vocabulaire relatif aux types d'élément est très insuffisant pour capturer la signification de chaque pièce dans le texte.
- Avec les documents HTML, les Métadonnées (**Metadata**) sont utilisées pour attribuer des informations au contenu et affecter sa représentation.

Exemple :

- Fichier → les métadonnées associées sont :

Titre : Semantique Web : Technologies and Applications

Sujet : Semantique Web

Auteur : ...

- Afin de comprendre des données, l'ordinateur a besoin de métadonnées qui doivent être fournies par un certain type de langage de balisage. Parmi ces langages, le plus répandu est **XML**.
- **XML** est un langage de balisage extrême souple, qui permet aux utilisateurs de créer par eux même de nouvelles balises afin d'ajouter un nouveau sens syntaxique à l'information.

L'exemple suivant illustre une instance de XML :

```
<etudiant>
  <id> 100 </id>
  <name>Said </name>
</etudiant>
```

– Limite de XML :

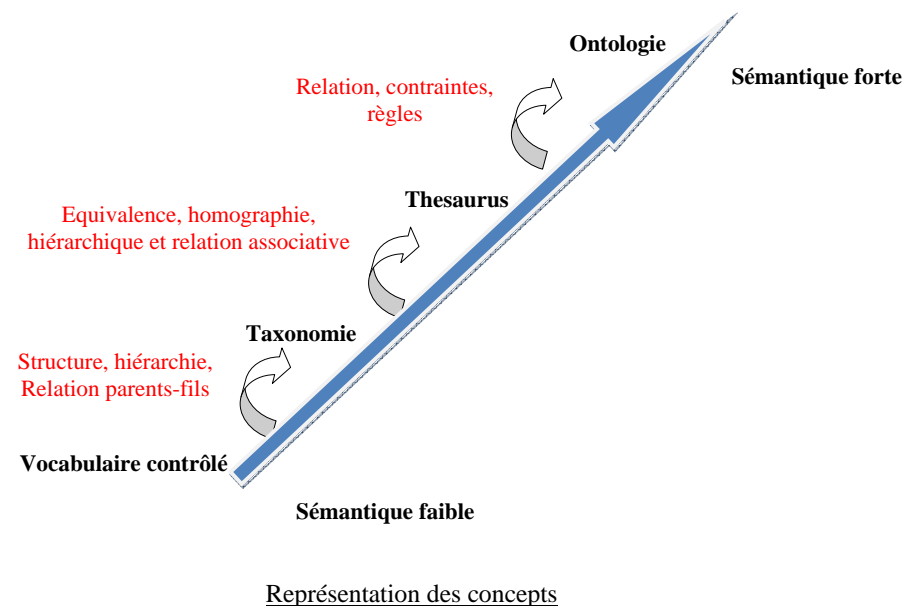
- Une des restrictions de XML est que ce langage ne peut définir que la syntaxe d'un document.
 - Les données XML n'incluent pas des informations qui peuvent être utilisées pour décrire le sens de chaque balise utilisée.
- L'exemple précédant indique qu'il existe un étudiant du *nom* de Said, son *id* est 100. Aucune information n'est fournie sur la signification de *id* ou d'autres champs qui composent le *id*.

1.2. Web sémantique

- Il ne représente pas un web différent mais une extension du web courant, dans lequel les informations sont sémantiquement bien définies,
- Décrit le contenu avec un formalisme à base de connaissances (RDF)
- Le web sémantique utilise des ontologies communes (RDF Schema, OWL) pour annoter les documents.
- L'ontologie est le document interopérable qui représente la sémantique des données du Web.
- Le projet du web sémantique englobe un ensemble d'outils nécessaires à la conception, la diffusion et l'exploitation de l'ontologie.

2. Sémantique : Représentation des concepts

- La sémantique est l'étude du sens de signes, telle que les termes ou les mots.
- Cette étude dépend des approches, des modèles ou les méthodes utilisées pour ajouter de la sémantique aux termes.
- Il existe quatre représentations qui peuvent être utilisées pour modéliser, organiser et décrire les termes sémantiquement : Vocabulaire contrôlé, Taxonomie, Thesaurus, Ontologie.



2.1. Vocabulaire contrôlé

- Le vocabulaire contrôlé est la méthode de métadonnées la plus simple, c'est une liste de termes (e.g., mots, phrases, notations) qui ont été énumérés explicitement.
- Tous les termes dans vocabulaire contrôlé doivent avoir une définition ni ambiguë, ni redondante.
- Le principal objectif est d'empêcher l'utilisateur de définir leur propre termes qui peuvent être ambigus, dénués de sens, ou mal orthographier.

2.2. Taxonomie

- Une taxonomie est une classification basée sur les thèmes ou sujets.
- Cette méthode classifie les termes en forme d'hiérarchie ou d'arbre. Elle décrit les mots en mettant en évidence leurs relations avec les autres mots.
- La représentation hiérarchique contient les relations de "est sous-classe de" ou "est super classe de".
- L'utilisateur ou l'ordinateur est en mesure de comprendre la sémantique des mots en analysant entre les mots.

Exemple de Taxonomie



2.3. Thesaurus

- Un thesaurus est un type particulier de langage documentaire.
- Il est constitué d'un ensemble structuré de concepts représentés par des termes, pouvant être utilisés pour l'**indexation** de documents textuels, à des fins de recherche documentaire.
- Un thesaurus représente une extension d'une taxonomie.

Exemple de Thesaurus : WordNet

- C'est un réseau lexical qui couvre la grande majorité des noms, verbes, adjectifs et adverbess de la langue anglaise.
- C'est un dictionnaire informatisé dont l'unité de base est le concept (non le mot).

- Il définit le sens des mots par deux moyens :
 - Les **synsets** (synonym set): ensembles de synonymes
 - Les **relations lexicales** entre les synsets : hyperonymie-hyponymie (is-a), méronymie, implication, dérivation morphologique
- **Synset** :
 - Le sens d'un mot est représenté par (i) l'ensemble des mots utilisés pour exprimer ce sens (les synonymes) et (ii) une définition.

Exemple : board

(i) Synset : { board, plank }

(ii) Définition : morceau de bois

• Relations lexicales :

- Le sens d'un mot est aussi déterminé par ses relations sémantiques avec d'autres sens.
- les relations suivantes sont utilisées :
- **Synonymie** : Deux mots sont synonymes s'ils sont interchangeables dans **certains** contextes linguistiques.
- **Antonymie** : relation lexicale entre mots, très importantes pour les adjectifs et les adverbess; Difficile à définir.
heavy est l'antonyme de **light** mais pas de **weightless**;
Des adjectifs ayant des sens similaires peuvent avoir des antonymes différents : **light/heavy**, **weight/weightless**.

- **Hyponymie** : structure les noms en une hiérarchie d'héritage, ce qui permet d'éviter les répétitions: Tout concept dans la hiérarchie hérite des propriétés de ses superconcepts.
- **Méronymie** : relation lexicale entre sens; X est un méronyme de Y si X **est une composante** de Y, X **est un élément** de Y, X **est le matériau** dont Y est constitué
- **Morphologie** : La morphologie est nécessaire pour permettre un accès simple aux lèmes (arbres --> arbre); le traitement de la morphologie inflectionnel fait partie de l'interface entre WN et utilisateur.
- Connaître la place d'un mot dans ce réseau de relations, c'est connaître son sens.

- Domaine d'utilisation de **WordNet** :

- Recherche d'informations: pour étendre la requête de l'utilisateur (ajout de synonymes, par exemple, pour augmenter le rappel, c'est-à-dire la proportion de documents pertinents rapportés),
- Acquisition de relations sémantiques,
- Désambiguïsation sémantique: pour l'étiquetage sémantique de corpus, pour la structuration et catégorisation des documents.

2.4. Ontologie

2.4.1. Définition

– Le terme ontologie est définie selon deux points de vue :

- **En philosophie** : une ontologie signifie science ou théorie de l'être, c'est-à-dire, l'étude des propriétés générales de ce qui existe.
- **En informatique** : une ontologie est une *spécification formelle et explicite d'une conceptualisation partagée*.

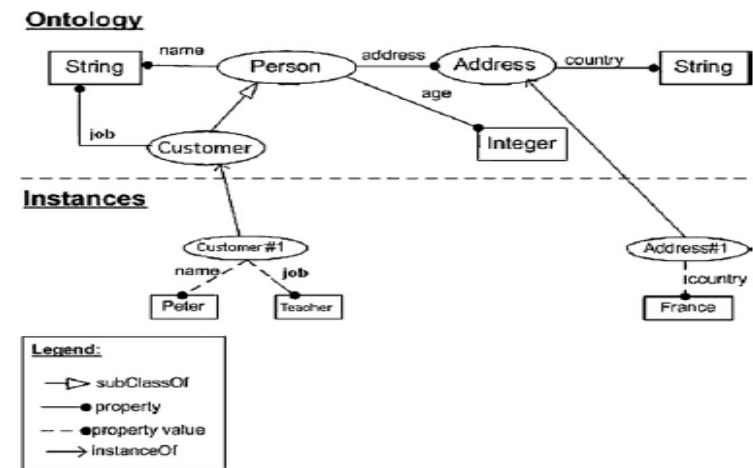
– Il s'agit d'un réseau sémantique qui regroupe un ensemble de concepts décrivant complètement un domaine.

– L'ontologie consiste à la définition des **entités** (objets), des **attributs**, les **relations** entre entités, le **vocabulaire du domaine**;

– Une ontologie est, non seulement le **repérage** et la **classification** des **concepts** mais c'est aussi des **caractéristiques** qui leur sont **attachées** et qu'on appelle les **propriétés** (appelées, aussi, les attributs).

– Les classes représentent les concepts pris au sens large.

Exemple d'une ontologie et leur instances :



- Une Ontologie est similaire à une taxonomie mais qui utilise des relations très riche en sémantique entre les termes et attributs, en plus des règles strictes qui guident la spécification des termes et des relations.
- Différence entre un thésaurus et une ontologie :
 - un thésaurus relie des concepts entre eux selon des relations précises : synonyme, homonyme, hiérarchie, terme associé.
 - L'ontologie ajoute des règles et des outils de comparaison sur et entre les termes, groupes de termes et relations : équivalence, symétrie, contraire, cardinalité, transitivité

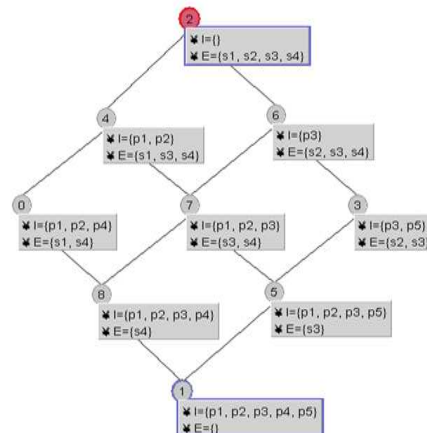
2.4.2. Les concepts d'ontologies

- Un concept peut être tout ce qui peut être évoqué : description d'une tâche, d'un objet matériel, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc.
- Un concept peut se diviser en trois parties :
 - **Terme ou Label** : est l'expression linguistique utilisée couramment pour y faire référence (ex. Customer).
 - **Notion** : correspond à la sémantique du concept, elle est définie à travers ses propriétés et ses attributs. La notion est appelée intention du concept (ex. job).
 - **Objets** : forme l'extension du concept. Il s'agit des objets auxquels le concept fait référence, autrement dit, ses instances (ex. Customer#1).

- Les concepts sont liés les uns aux autres par des relations taxonomiques (hiérarchisation des concepts) d'une part, et sémantiques d'autre part.

Exemple 1 :

$G \times M$	$p1$	$P2$	$p3$	$p4$	$p5$
$s1$	1	1	0	1	0
$s2$	0	0	1	0	1
$s3$	1	1	1	0	1
$s4$	1	1	1	1	0



Exemple 2:

pour décrire les concepts entrant en jeu dans la conception de cartes électroniques, on pourrait définir l'ontologie (simplifiée ici) suivante :

- une carte électronique est un ensemble de composants,
- un composant peut être soit un condensateur, soit une résistance, soit une puce,
- une puce peut être soit une unité de mémoire, soit une unité de calcul,
- une carte électronique qui contient une unité de calcul contient aussi au moins une unité de mémoire.

2.4.3. Relations entre les concepts

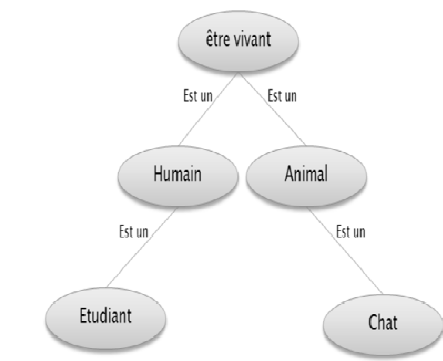
- Elles représentent un type d'association entre les concepts du domaine.

On peut en distinguer deux sortes :

2.4.3.1. la relation « is-a » (est un)

- **la relation de subsomption is-a** définit un lien de généralisation.
 - Elle est utilisée pour structurer les ontologies.
 - Elle permet formellement l'héritage de propriétés et doit être complétée par d'autres relations pour exprimer la sémantique du domaine.
- **La notion de subsomption** : un concept C1 subsume un concept C2 si toute relation sémantique de C1 est aussi relation sémantique de C2 (le concept C2 est plus spécifique que le concept C1).

Exemple :

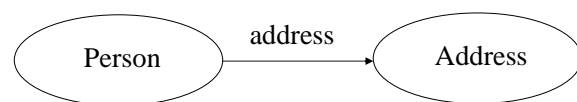


2.4.3.2. Relation associative

- est une **relation d'interaction entre deux concepts** qui n'est pas une relation de subsomption.

Elle correspond à la notion de rôle en Logique de description et permet de typer les concepts qu'elle relie.

Exemple de relation associative



2.4.4. Typologie des ontologies

- En fonction de leur usage, on distingue cinq catégories d'ontologies :

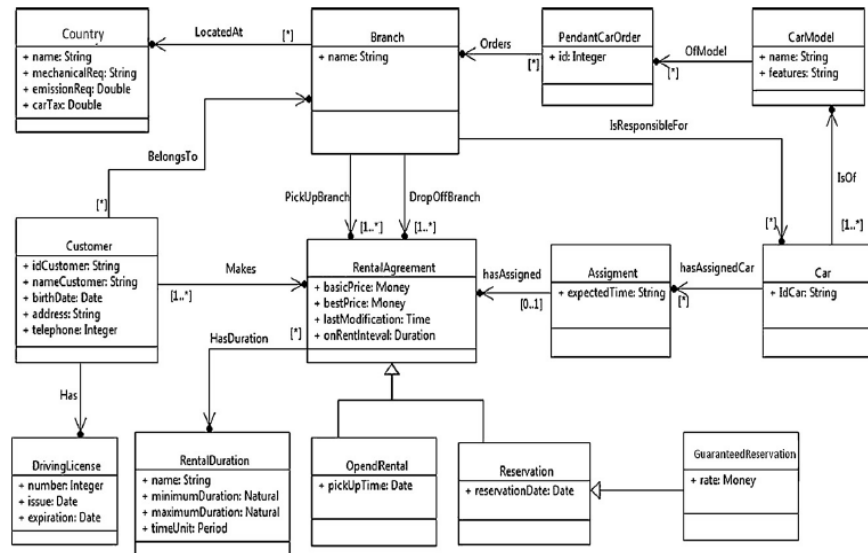
1. **Ontologie générique** : décrit des concepts généraux, indépendants d'un domaine ou d'un problème particulier. Il s'agit par exemple des concepts de temps, d'espace et d'événement.

2. **Ontologie de domaine** : spécifie un point de vue sur un domaine particulier à l'aide d'un vocabulaire lié à un domaine de connaissance générique.

- Les concepts d'une ontologie de domaine sont souvent définis comme une spécialisation des concepts des ontologies génériques.

- Une ontologie de domaine est constituée d'une description en extension du vocabulaire du domaine, d'une typologie, et d'une hiérarchie ou un treillis de classes.

Exemple d'ontologie de domaine (Khouri S., et al. 2012)



3. **Ontologie d'application** : décrit la structure des connaissances nécessaires à la réalisation d'une tâche particulière.

– Elle permet aux experts du domaine d'utiliser le même langage que celui de l'application.

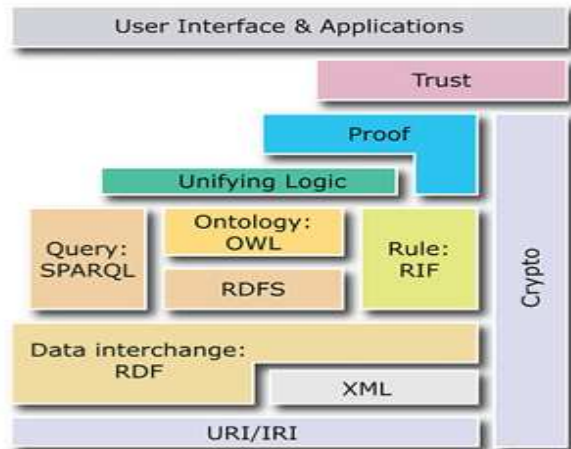
4. **Ontologie de représentation** : définit un ensemble de primitives de représentation des concepts, des relations des ontologies de domaine et des ontologies génériques.

5. **Ontologie de méthode**: décrit le processus de raisonnement d'une façon indépendante d'un domaine et d'une implémentation donnée.

– Elle spécifie des entités qui relèvent de la résolution d'un problème et fournit les définitions des concepts et relations utilisées pour spécifier un processus de raisonnement lors de la réalisation d'une tâche particulière.

3. Les couches du Web sémantique :

– La pile technologique est un standard du W3C comprenant :



3.1. URI (Universal Resource Identifier)

– est un outil de référencement qui permet d'identifier de manière unique chaque ressource du web.

– L'IRI et l'URI possèdent leurs propres normes d'encodage et ils se différencient dans la normalisation des caractères employés pour construire l'adresse.

- Si l'Unicode est employé pour construire l'URI, l'IRI utilise la norme ISO-3166 qui étend la liste des caractères employables aux caractères internationaux et spécifiques à une langue particulière (par ex. : le « é » pour le français).

3.2. XML (eXtensible Markup Language)

- XML est un langage développé par XML Working Group dirigé par le W3C (depuis 1996);
- Il s'agit plutôt d'un métalangage qui permet de définir des propres balises pour chaque document.
- **Exemple :**

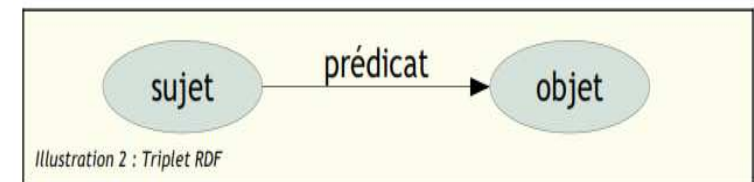
```
<livre>
  <language> fr </language>
  <title> Web sémantique et modélisation ontologique </title>
  <auteur> Michel HÉON </auteur>
  <edition> eni </edition>
  <isbn> 274608869X </isbn>
</livre>
```

- Un document XML bien formé est dit « valide » lorsqu'il est conforme à la déclaration de type de document (DTD, XML Schema, etc.) qui l'accompagne.
- XML propose différentes interfaces de programmation (API) qui facilitent la manipulation de documents par les processeurs XML.
 - DOM : Document Object Model, permet de naviguer dans le document (traversées d'arbre), d'accéder à des parties, etc.
 - SAX : Simple API for XML, interface plus simple, même usage.

3.3. RDF (Ressource Description Framework)

- RDF est un langage créé en 1999, Standard du W3C
- Il s'agit d'un modèle de données pour exprimer des métadonnées sémantiques sur des pages Web.
- Il permet interconnecter les ressources anonymes ou nommées par un URI/IRI du web, pour former un graphe orienté et étiqueté
- RDF se base sur une description des connaissances à l'aide de phrases simples :
 - C'est un moyen d'exprimer des relations.
 - Ces relations sont décrites sous forme de graphe.
 - Chaque nœud du graphe est une ressource ou une valeur.
 - Et chaque nœud est relié à un autre par un arc "nommé"

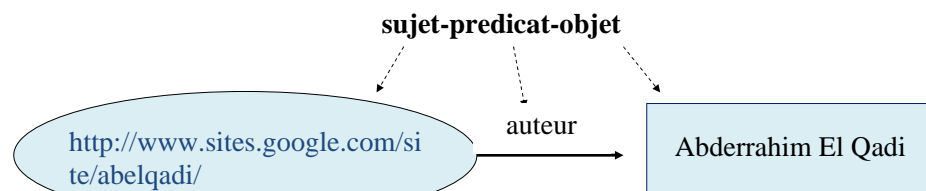
- Il est construit selon le modèle de triplets (sujet/prédicat/objet) :



- **Sujet** : une chose (ressource) identifiée par une URL
- **Predicate**: Type de metadata qui est aussi identifié par une URL (aussi appelé propriété)
- **Object**: est la valeur de ce type de metadata.

Exemple : Considérons l'information suivante : "l'auteur de <http://www.sites.google.com/site/abelqadi/> est Abderrahim El Qadi".

– On peut choisir de représenter cette information en triplet RDF par :



Ressource

Type propriété

Valeur propriété

Le site est la ressource. Cette ressource a un URI: <http://www.sites.google.com/site/abelqadi/>, et sa propriété est "Auteur" avec la valeur "Abderrahim El Qadi".

– En RDF/XML, cette information s'écrit :

```
<rdf:Description about=" http://www.sites.google.com/site/abelqadi/">
  <schema:auteur>Abderrahim El Qadi</schema:auteur>
</rdf:Description>
```

Dans l'exemple, le préfixe d'espace de nom « schema » correspond à un espace de nom spécifique, qui doit être indiqué par l'auteur du document RDF/XML dans une déclaration XML d'espace de nom.

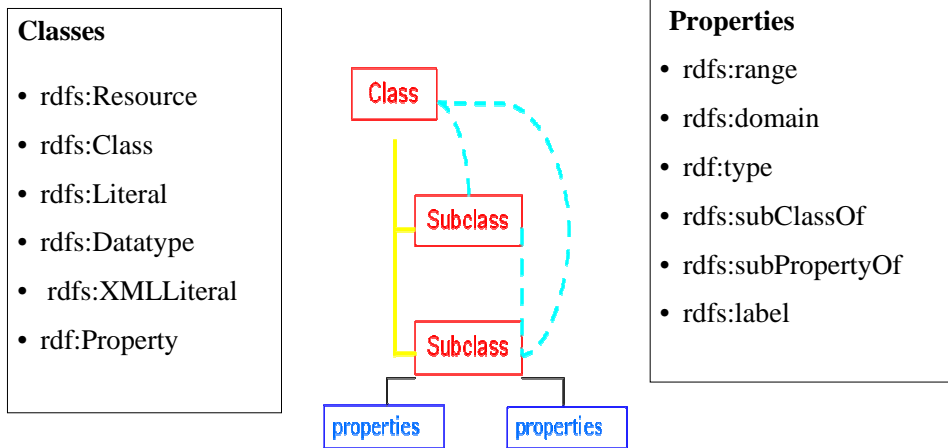
3.4. RDFS

- RDF est **indépendant du domaine**, en d'autres mots, il ne pose aucun présupposé quant à un domaine spécifique. C'est donc à l'utilisateur que revient le rôle de définir sa propre terminologie dans un langage schéma appelé **RDF Schema** (RDFS).
 - Le vocabulaire RDFS (RDF Schema) définit la terminologie de base du web sémantique.
 - Il contient les éléments fondamentaux à la construction d'une ontologie notamment les concepts de classe, de propriété et de hiérarchie.
- Comme on le voit dans l'exemple précédent (`<schema:auteur>Abderrahim El Qadi</schema:auteur>`), la propriété **auteur** n'a de sens pour décrire la

ressource <http://www.sites.google.com/site/abelqadi/> que dans un contexte bien défini :

- le lecteur (utilisateur) est humain, et comprend le français
- l'information transmise par le triplet RDF "{ <http://www.sites.google.com/site/abelqadi/>, Abderrahim El Qadi, auteur }" est suffisamment triviale pour comprendre que sa signification est "Abderrahim El Qadi est l'auteur de <http://www.sites.google.com/site/abelqadi/>".

– Schéma du Modèle RDFS



– **Classe** : Le concept de la classe RDF est similaire au concept de classe dans les langages de la programmation orienté objet.

- L'héritage est autorisé. Ceci permet de définir les ressources comme étant des instances de classes, et sous-classes de classes.
- RDF permet d'organiser les classes de manière hiérarchique.

– **Propriété** : les propriétés RDFS peuvent être vues comme étant des attributs de classes.

3.5. OWL (Web Ontology Language)

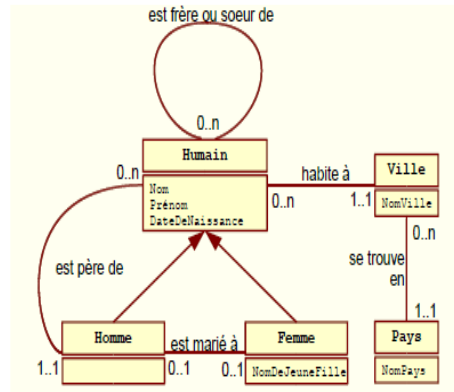
- est un langage standard pour définir des ontologies sur le Web.
- En 2004, OWL devient une recommandation du W3C.
- Basé sur RDF schema (RDFS).
- Étend les constructions de base pour améliorer :
 - L'interopérabilité (équivalences)
 - Le raisonnement (description logic)
 - Les évolutions (intégration)
- OWL intègre, en plus, des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc.

– Les 3 sous-langages OWL:

- OWL LITE: permet d'établir une hiérarchie de concepts simples, contraintes simples.
- OWL DL (DL pour description logic): comprend toutes les structures de OWL, possède une expressivité plus importante, avec complétude de calcul.
- OWL FULL expressivité maximale, liberté syntaxique sans garantie de calcul, une classe peut aussi correspondre à l'instance d'une autre classe.

Exemple d'ontologie OWL DL : l'ontologie OWL présentée décrit un groupe de personnes et leurs relations de parenté, ainsi que leur lieu d'habitation.

- Le diagramme de classes UML ci-dessous indique les classes qui composent la population :



- Le schéma suivant indique plus précisément les individus qui existent dans l'ontologie :
 - Il comporte trois instances de la classe « Homme » (en vert), une instance de la classe « Femme » (en rouge), ainsi que deux instances de « Ville » (en bleu) et de « Pays » (en mauve).

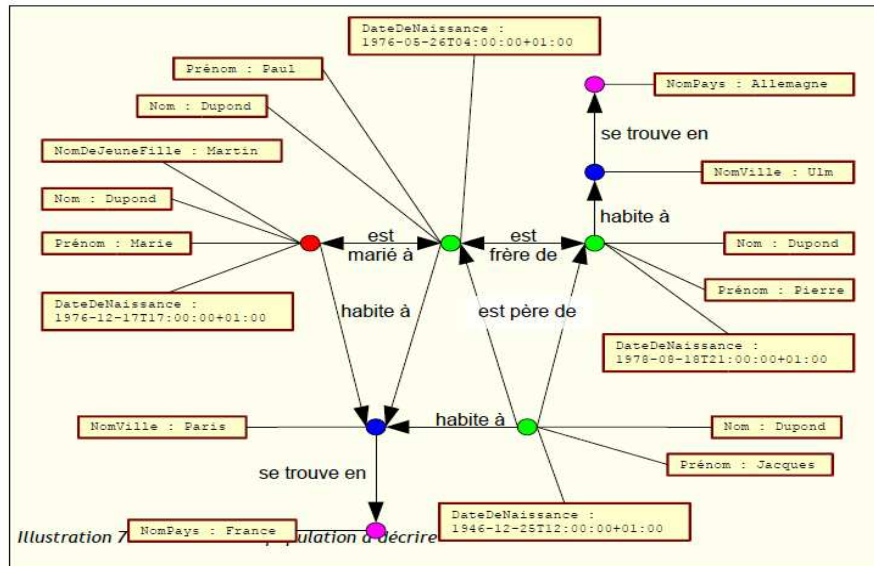


Illustration 7 : Population d'instance

4. Structure d'une ontologie OWL

- Tout comme RDF, OWL dispose d'une structure logique dont RDF/XML est la sérialisation, les ontologies OWL se présentent sous forme de fichiers texte, de "documents OWL".
- La création d'un document OWL fait l'objet de diverses recommandations:
 - Il est recommandé d'employer le type MIME lié à RDF, à savoir application/rdf+xml ou, à la rigueur, le type MIME de XML, application/xml.

- La recommandation du groupe de travail WebOnt indique qu'il est préférable d'employer, comme suffixe de nom de fichier, les extensions ".rdf" ou ".owl"
- Une ontologie formalisée en OWL comprend :
un espace de nom, l'entête pour décrire l'ontologie, la définition des classes, des propriétés et des instances

4.1. Espaces de nommage

- Une ontologie commence par une déclaration d'espace de nom (parfois appelée « de nommage ») contenue dans une balise `rdf:RDF`.

Exemple : déclaration d'espace de nom (ontologie sur une population de personnes ou, d'une manière plus générale, sur l'humanité):

```
<rdf:RDF
  xmlns      = "http://domain.tld/path/humanite#"
  xmlns:humanite= "http://domain.tld/path/humanite#"
  xmlns:base  = "http://domain.tld/path/humanite#"
  xmlns:vivant = "http://otherdomain.tld/otherpath/vivant#"
  xmlns:owl   = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf   = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs  = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd   = "http://www.w3.org/2001/XMLSchema#">
...
```

- Les deux premières déclarations identifient l'espace de nommage propre à l'ontologie.
- La première déclaration d'espace de nom indique à quelle ontologie se rapporter en cas d'utilisation de noms sans préfixe dans la suite de l'ontologie.
- La troisième déclaration identifie l'URI de base de l'ontologie courante.
- La quatrième déclaration signifie simplement que, au cours de la rédaction de l'ontologie humanité, on va employer des concepts développés dans une ontologie vivant, qui décrit ce qu'est un être vivant.
- Les quatre dernières déclarations introduisent le vocabulaire d'OWL et les objets définis dans l'espace de nommage de RDF, du schéma RDF et des types de données du Schéma XML.

4.2. En-têtes d'une ontologie

- L'en-tête est la partie du document ontologique qui contient les métadonnées de l'ontologie. (balise `owl:Ontology`) :

```
<owl:Ontology rdf:about="">
  <rdfs:comment>Ontologie décrivant l'humanité</rdfs:comment>
  <owl:imports
    rdf:resource="http://otherdomain.tld/otherpath/vivant"/>
  <rdfs:label>Ontologie sur l'humanité</rdfs:label>
...
```

4.3. Les classes

- Une classe définit un groupe d'individus qui sont réunis parce qu'ils ont des caractéristiques similaires.
- L'ensemble des individus d'une classe est désigné par le terme "extension de classe", chacun de ces individus étant alors une "instance" de la classe.

4.3.1. Déclaration de classe

- Une classe peut ainsi se déclarer de six manières différentes :

4.3.1.1. L'indicateur de classe

- La description de la classe se fait, dans ce cas, directement par le nommage de cette classe.

Une classe « humain » se déclare de la manière suivante :

```
<owl:Class rdf:ID="Humain" />
```

- Il est à noter que ce type de description de classe est le seul qui permette de nommer une classe.
- Dans les cinq autres cas, la description représente une classe dite « anonyme », créée en plaçant des contraintes sur son extension.

4.3.1.2. L'énumération des individus composant la classe

- Ce type de description se fait en énumérant les instances de la classe, à l'aide de la propriété **owl:oneOf** :

```
<owl:Class>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Damien" />
    <owl:Thing rdf:about="#Olivier" />
    <owl:Thing rdf:about="#Philippe" />
    <owl:Thing rdf:about="#Xavier" />
    <owl:Thing rdf:about="#Yves" />
  </owl:oneOf>
</owl:Class>
```

4.3.1.3. La restriction de propriétés

- La description par restriction de propriété permet de définir une classe anonyme composée de toutes les instances de **owl:Thing** qui satisfont une ou plusieurs propriétés.

4.3.1.4. Les descriptions par intersection, union ou complémentaire

permettent de décrire une classe par, comme leur nom l'indique, l'intersection, l'union ou le complémentaire d'autres classes déjà définies, ou dont la définition se fait au sein même de la définition de la classe courante.

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#etudiantsFS" />

    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourFrere" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
        2
      </owl:cardinality>
    </owl:Restriction>

  </owl:intersectionOf>
</owl:Class>
```

4.3.1.5. Héritage

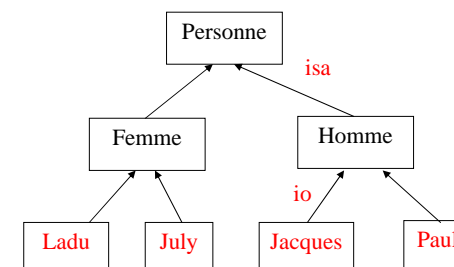
- Il existe dans toute ontologie OWL une superclasse, nommée **Thing**, dont toutes les autres classes sont des sous-classes. Ceci représente le concept d'héritage, disponible à l'aide de la propriété **subClassOf** :

```
<owl:Class rdf:ID="Humain">
  <rdfs:subClassOf rdf:resource="&etre;#EtreVivant" />
</owl:Class>
<owl:Class rdf:ID="Homme">
  <rdfs:subClassOf rdf:resource="#Humain" />
</owl:Class>
<owl:Class rdf:ID="Femme">
  <rdfs:subClassOf rdf:resource="#Humain" />
</owl:Class>
```

- Il existe également une classe nommée **noThing**, qui est sous-classe de toutes les classes OWL. Cette classe ne peut avoir aucune instance.

4.3.2. Les instances de classe : Individu

- Ce sont les « individus » qui peuplent les classes.



- Un individu est un « fait », encore appelé « axiome d'individu ».

– On peut distinguer deux types de faits :

- **les faits concernant l'appartenance à une classe** : le fait s'exprime de la manière suivante :

```
<Humain rdf:ID="Pierre">
  <aPourPere rdf:resource="#Jacques" />
  <aPourFrere rdf:resource="#Paul" />
</Humain>
```

Dans cet exemple le fait exprime l'existence d'un Humain nommé «Pierre» dont le père s'appelle «Jacques», et qu'il a un frère nommé «Paul».

- On peut également instancier un individu anonyme en omettant son identifiant :

```
<Humain>
  <aPourPere rdf:resource="#Jacques" />
  <aPourFrere rdf:resource="#Paul" />
</Humain>
```

Ce fait décrit, dans ce cas, l'existence d'un Humain dont le père se nomme «Jacques» et qui a un frère nommé «Paul».

- **les faits concernant l'identité des individus** : Une difficulté qui peut éventuellement apparaître dans le nommage des individus concerne la non-unicité éventuelle des noms attribués aux individus.

Par exemple, un même individu pourrait être désigné de plusieurs façons différentes.

OWL propose un mécanisme permettant de lever cette ambiguïté, à l'aide des propriétés **owl:sameAs**, **owl:diffrentFrom** et **owl:allDifferent**.

L'exemple suivant permet de déclarer que les noms "Smith" et "Black" désignent la même personne :

```
<rdf:Description rdf:about="#Smith">
  <owl:sameAs rdf:resource="#Black" />
</rdf:Description>
```

4.3.3. Classes équivalentes et classes disjointes

- 2 classes sont équivalentes **«equivalentClass»** lorsqu'elles ont les mêmes instances exemple: classe "Voiture" et "Automobile".
- Inversement 2 classes disjointes **«disjointWith»** lorsqu'elles ne peuvent avoir des instances communes. exemple: classe "Women" et "Man".

4.3.4. Les propriétés

- OWL fait la distinction entre deux types de propriétés :

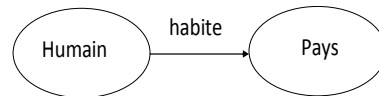
4.3.4.1. Les propriétés d'objet

- permettent de relier des instances à d'autres instances.

Une propriété d'objet est une instance de la classe **owl:ObjectProperty**.

Exemple :

```
<owl:ObjectProperty rdf:ID="habite">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Pays" />
</owl:ObjectProperty>
```



Dans l'exemple ci-dessus, on apprend que la propriété **habite** a pour domaine la classe Humain et pour image la classe Pays : elle relie des instances de la classe Humain à des instances de la classe Pays.

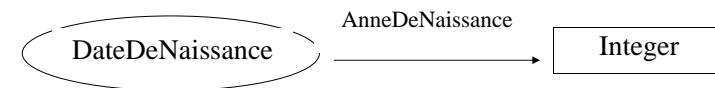
4.3.4.2. Les propriétés de données

- permettent de relier des individus à des valeurs de données.

- une propriété de type de donnée étant une instance de la classe **owl:DatatypeProperty**.

- Dans le cas d'une propriété de type de donnée, l'image de la propriété peut être un type de donnée, comme défini dans le Schéma XML.

Par exemple, on peut définir la propriété de type de données **anneeDeNaissance**:



```
<owl:Class rdf:ID="dateDeNaissance" />
<owl:DatatypeProperty rdf:ID="anneeDeNaissance">
  <rdfs:domain rdf:resource="#dateDeNaissance" />
  <rdfs:range rdf:resource="&xsd:positiveInteger"/>
</owl:DatatypeProperty>
```

Dans ce cas, anneeDeNaissance fait correspondre aux instances de la classe de dateDeNaissance des entiers positifs (restriction de propriété).

- On peut également employer un mécanisme de hiérarchie entre les propriétés, exactement comme il existe un mécanisme d'héritage sur les classes :

```
<owl:Class rdf:ID="Humain" />
<owl:ObjectProperty rdf:ID="estDeLaFamilleDe">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="aPourFrere">
  <rdfs:subPropertyOf rdf:resource="#estDeLaFamilleDe" />
  <rdfs:range rdf:resource="#Humain" />
  ...
</owl:ObjectProperty>
</owl:Class>
```

La propriété aPourFrere est une sous-propriété de estDeLaFamilleDe, ce qui signifie que toute entité ayant une propriété aPourFrere d'une certaine valeur a aussi une propriété estDeLaFamilleDe de même valeur.

4.3.4.3. Restrictions sur les propriétés

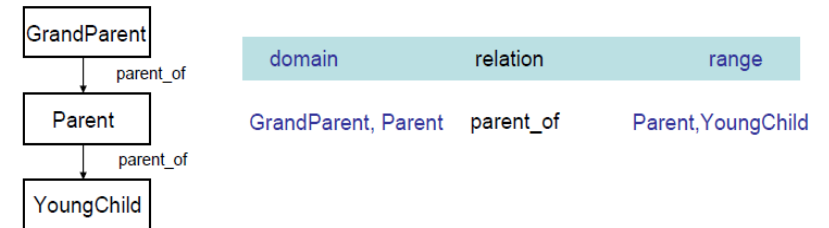
a. Restrictions globales :

- **domain** : classes pour lesquelles est définie la propriété,
- **range** : classes reliées par la propriété au domain.

Exemple :

```
<owl:ObjectProperty rdf:ID="seTrouveEn">
  <rdfs:domain rdf:resource="#Ville" />
  <rdfs:range rdf:resource="#Pays" />
</owl:ObjectProperty>
```

b. Déclaration au niveau des classes :



- **allValuesFrom**: au niveau d'une classe, permet de restreindre les classes liées par une relation.

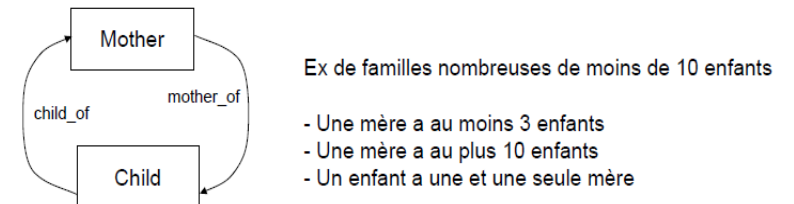
Exemple : un individu de la classe Parent ne peut être relié par la relation parent_of qu'à un individu de la classe YoungChild

- **someValuesFrom**: au niveau d'une classe, indique qu'un individu a au moins une relation avec un individu de la classe indiquée dans la restriction.

Un individu de la classe GrandParent a au moins une relation avec un individu de la classe Parent.

- **hasValue**: au niveau d'une classe, permet de restreindre la relation à un individu donné.

c. Déclaration au niveau des classes (cardinalités):



- **minCardinality**: toute instance de la classe est liée par la propriété à au moins x individus

Exemple: propriété mother_of de la classe Mother: minCardinality=3

Restriction sur les Propriétés : déclaration au niveau des classes (cardinalités)

- **maxCardinality**: toute instance de la classe est liée par la propriété à au plus x individus

Exemple: propriété mother_of de la classe Mother: maxCardinality=10

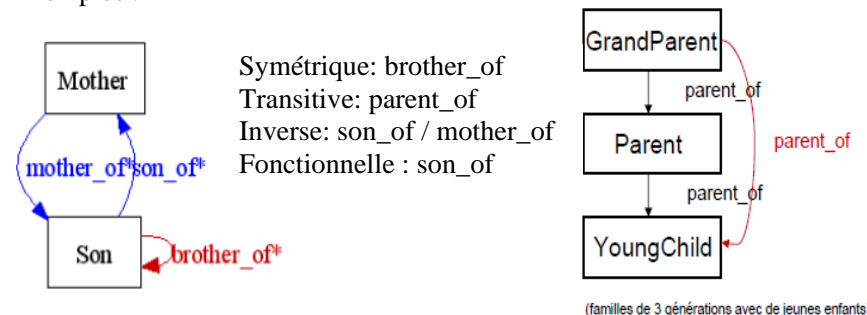
- **cardinality**: toute instance de la classe est liée par la propriété à exactement x individus

Exemple: propriété son_of de la classe Mother : cardinality=1

4.3.4.4. Caractéristiques des propriétés

- Parmi les caractéristiques de propriétés principales, on trouve la transitivité, la symétrie, la fonctionnalité, l'inverse, etc.

Exemples :



Exemple:

<pre><owl:ObjectProperty rdf:ID="aPourPere"> <rdfs:domain rdf:resource="#Humain" /> <rdfs:range rdf:resource="#Humain" /> </owl:ObjectProperty></pre>	Propriété non symétrique
<pre><owl:ObjectProperty rdf:ID="aPourFrere"> <rdf:type rdf:resource="&owl;SymmetricProperty" /> <rdfs:domain rdf:resource="#Humain" /> <rdfs:range rdf:resource="#Humain" /> </owl:ObjectProperty></pre>	Propriété symétrique
<pre><Humain rdf:ID="Pierre"> <aPourFrere rdf:resource="#Paul" /> <aPourPere rdf:resource="#Jacques" /> </Humain></pre>	

L'Humain Pierre a pour frère Paul, de même que (symétrie) l'Humain Paul a pour frère Pierre. Par contre, si Pierre a pour père Jacques, l'inverse n'est pas vrai (aPourPere n'est pas symétrique).

4.4. Syntaxe graphique (G-OWL)

- Le langage G-OWL est un langage de modélisation typé et graphique qui vise à représenter le contenu d'une ontologie de manière simple et efficace.

4.4.1. Sémantique des entités de G-OWL

- Les entités graphiques de G-OWL sont de cinq types : le rectangle, le conteneur, l'hexagone, le rectangle hachuré et l'hexagone hachuré:

Entité	Représentation graphique	Description
Le rectangle	Concept	Le rectangle est employé pour désigner un concept dans le domaine du discours et pour définir la classe dans une ontologie. Le rectangle est aussi employé pour désigner un type de donnée (<i>DataType</i>).
Le conteneur	Conteneur	Le conteneur est employé pour désigner un regroupement de faits dans un domaine et pour définir des expressions ou des axiomes dans l'ontologie.
L'hexagone	Propriété	L'hexagone est utilisé pour désigner une caractéristique, un attribut dans le domaine et il sert à définir une propriété dans l'ontologie.
Le rectangle hachuré	Fait	Le rectangle hachuré désigne un fait dans le domaine du discours et il définit l'individu dans l'ontologie. Le rectangle hachuré est aussi employé pour désigner la valeur d'une donnée.
L'hexagone hachuré	Propriété de donnée	L'hexagone hachuré désigne un attribut de donnée dans le domaine et il décrit la propriété de donnée dans l'ontologie.

4.4.2. Sémantique des relations de G-OWL

- Le langage G-OWL comporte un certain nombre de relations typées qui unissent les entités de l'ontologie.
- Le tableau suivant présente les principales relations employées dans une ontologie.

Relation Exemple de représentation graphique	Description
<p>Lien S</p>	Le lien S (<i>sorte de</i>) est employé entre deux rectangles pour représenter la relation de hiérarchisation entre deux concepts. On dira que <i>Femme</i> est une sorte de <i>Personne</i> .
<p>Lien double S</p>	Le lien double S (<i>synonyme de</i>) est employé entre deux rectangles pour représenter la relation d'équivalence entre deux concepts. On dira que <i>Personne</i> et <i>Humain</i> sont synonymes.
<p>Lien I</p>	Le lien I (<i>a pour instance</i>) est employé entre un rectangle (un concept) et un rectangle hachuré (un fait) pour représenter la relation d'instanciation entre deux classes. On dira que <i>Marie</i> est une instance de <i>Personne</i> .
<p>Lien A</p>	Le lien A est notamment employé entre un hexagone et un rectangle ou vice versa afin d'axiomatiser une propriété. On dira que <i>La Femme</i> a le rôle d'être l'épouse de l' <i>Homme</i> .
<p>Prédicat</p>	Le prédicat est employé entre deux rectangles hachurés. Il sert à unir un individu (le sujet) à un autre individu (l'objet) par un prédicat déclaré par une propriété.

5. Langage SPARQL

- C'est le langage de requêtes pour interroger les graphes RDF/RDFS.
- Il est adapté à la structure spécifique de triplets qui les constituent.
- SPARQL permet d'exprimer des requêtes interrogatives ou constructives :
 - une requête SELECT, de type interrogative, permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE ;
 - une requête CONSTRUCT, de type constructive, engendre un nouveau graphe qui complète le graphe interrogé.

– La structure d'une requête SPARQL est très similaire à celle employée dans le langage SQL :

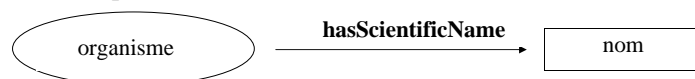
```
BASE pre_x :<namespace-uri >
PREFIX pre_x :<namespace-uri >
SELECT variable-list
FROM source-list
WHERE pattern
ORDER BY expression
LIMIT integer > 0
OFFSET integer > 0
```

- **PREFIX** suivi d'une déclaration d'un raccourci pour des espaces de nom. Il est courant d'avoir plusieurs préfixes.
- **BASE** suivi d'un raccourci pour une URI de base auxquelles les URIS seront concaténées. Une seule base par requête.
- La clause **SELECT** permet de choisir les variables du résultat, parmi les variables de la clause WHERE
- La clause **WHERE** contient des triplets qui définissent le motif (pattern)..

Exemple : la requête qui donne les noms scientifiques des individus qui en possèdent un.

L'URI, dénotant l'ontologie exploitée, est renommée **jungle** afin de rendre la requête plus lisible. Deux variables (**?organisme** et **?nom**) sont liées dans la clause WHERE au travers du prédicat (propriété)

hasScientificName et vont permettre de retourner les individus et leurs noms scientifiques.



```
PREFIX jungle: <http://www.owl-
ontologies.com/Ontology1225188668.owl#>
SELECT ?organisme ?nom
WHERE { ?organisme jungle:hasScientificName ?nom }
```

- La réponse aura la forme d'une table avec pour en-têtes de colonnes organisme et nom.
- Ce type de requêtes est bien adapté à la structure spécifique des graphes RDF et permet d'interroger sur tous les triplets qui les constituent (sujet/propriété/valeur).
- Le **.** permet d'exprimer une concaténation de liaisons de variables (l'exemple, présenté, restitue les organismes qui ont un nom scientifique et un âge) :

```
PREFIX jungle: <http://www.owl-
ontologies.com/Ontology1225188668.owl#>
SELECT ?Organisme ?NomScientifique ?Age
WHERE { ?Organisme jungle:hasScientificName ?NomScientifique
. ?Organisme jungle:hasAge ?Age }
```

5.1. Motifs de triplets

- On parle de motif de triplet car le triplet peut contenir des variables.
- Les termes utilisés dans les motifs de triplets sont :
 - des IRIs (Internationalized Resource Identifiers),
 - des littéraux
 - des variables, préfixées par ? ou par \$
 - des nœuds blancs

5.1.1. IRI et espaces de noms

- Comme en XML, on peut définir un préfixe associé à un espace de noms, et un espace de noms **”base”** de la requête.
- Les requêtes suivantes sont équivalentes :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { <http://example.org/book/book1> dc:title ?title }
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/book/>
SELECT $title
WHERE { :book1 dc:title $title }
BASE <http://example.org/book/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT $title
WHERE { <book1> dc:title ?title }
```

5.1.2. Littéraux

- il s'agit, de chaîne de caractères entre simples quotes ou guillemets.
- Pour simplifier,
- 1 est synonyme de "1"^^xsd:integer
 - 1.3 est synonyme de "1.3"^^xsd:decimal
 - true est synonyme de "true"^^xsd:boolean

5.1.3. Nœuds blancs (Nœuds anonymes)

- Ils jouent le rôle de variables non distinguées,
- On les utilise sous deux formes :
 - désigné par un label (ex : `_:b57`)
 - ou forme abrégée `[]` dans les motifs de triplets :
- `[:p "v"]` . ou `[] :p "v"` . permettent d'allouer un nœud blanc (par exemple de label `b57`) et sont équivalents à `_:b57 :p "v"` .

`[:p "v"] :q "w"` .

est équivalent à

`_:b57 :p "v"` .

`_:b57 :q "w"` .

5.2. Collections RDF

5.2.1. RDFS

- Le prédicat `rdfs:subClassOf` permet d'associer une classe à l'une de ses super-classes

```
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object }
```

La réponse sera de la forme : subject avec pour object

- Le prédicat `rdfs:subPropertyOf` permet d'associer une propriété à l'une de ses super-propriétés

```
SELECT ?subject ?object
WHERE { ?subject rdfs:subPropertyOf ?object }
```

5.2.2. RDF:type

- Le prédicat `rdf:type` permet d'associer une ressource à sa classe (?x `rdf:type` ex:Personne)

```
SELECT ?subject ?object
WHERE { ?subject rdf:type ?object }
```

Un exemple de réponse sera notamment : subject is part of avec pour object owl transitive property


Exemple2: Soit un schéma avec les relations "pere_de", "mere_de" et les types "Homme" et "Femme", demandez les femmes et leurs parents:

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?femme ?pere ?mere
WHERE {
    ?femme rdf:type ex:Femme .
    ?mere ex:mere_de ?femme .
    ?pere ex:pere_de ?femme .
}
```

- Les triplets ayant une racine commune peuvent être simplifiés ainsi que la relation de typage;
- Le mot **a** permet de préciser qu'une ressource est d'un certain type:

?x a :Class1 .
est équivalent à
?x rdf:type :Class1

Exemple 3

<pre>SELECT ?nom ?prenom WHERE { ?x a Person; nom ?nom ; prenom ?prenom ; auteur ?y . }</pre>		<pre>SELECT ?nom ?prenom WHERE { ?x rdf:type Person . ?x nom ?nom . ?x prenom ?prenom . ?x auteur ?y . }</pre>
---	---	--

- Une liste de valeurs
?x prenom "Fabien", "Lucien" .
- Resource anonyme dans pattern requête
[prenom "Fabien"] ou [] prenom "Fabien"
- **Question:** ?x a Document; auteur [nom "Lo"] .
- **Réponse:** les documents ?x écrits par un auteur ayant pour nom "Lo"

5.2.3. OWL

- Les prédicats owl :disjointWith, owl :equivalentClass, owl :hasValue permettent respectivement de retourner les concepts qui sont disjoints, les concepts qui sont équivalents et enfin les valeurs exactes des propriétés (qui en possèdent une).

```
SELECT ?ressource ?ressourceDisjointe
WHERE { ?ressource owl:disjointWith ?ressourceDisjointe }
```

```
SELECT ?ressource ?ressourceEquivalente
WHERE { ?ressource owl:equivalentClass ?ressourceEquivalente }
```

```
SELECT ?propriete ?valeurPropriete
WHERE { ?propriete owl:hasValue ?valeurPropriete }
```

5.3. Filtres

- On peut ajouter des conditions à la requête, sous la forme de filtres.
- Un filtre s'applique à un motif de graphe.

FILTER regex(?x, "pattern" [, "flags"])

- Exemple : Les titres du livre qui commencent par "SPARQL"

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE
{
    ?x dc:title ?title
    FILTER regex(?title, "^SPARQL")
}
```

- Les titres et prix des livres qui coûtent moins de 30.5.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
        FILTER (?price < 30.5)
        ?x dc:title ?title . }
```

5.3.1. Opérateurs de Filtre

- Dans la clause FILTER:
 - Comparateurs: <, >, =, <=, >=, !=
 - Tests sur les binding des variables: isURI(?x), isBlank(?x), isLiteral(?x), bound(?x)
 - Filtres à base d'expressions regulieres regex(?x, "A.*")
 - Accès aux attributs/valeur lang(), datatype(), str()
 - Fonctions de (re-)typage (casting) xsd:integer(?x)
 - Fonctions externes / extensions
 - Combinaisons &&, ||
- Dans la clause WHERE: @fr , ^^xsd:integer
- Dans la clause SELECT: distinct

Exemple1 : Soit un schéma avec le type "Personne", et la propriété "taille" (en centimètres). demandez les personnes entre 140 et 170 centimètres.

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?x
WHERE {
  ?x rdf:type ex:Personne
  ?x ex:taille ?t
  FILTER ( ?t >= 140 && ?t <= 170 )
}
```

xsd:integer(?t)

5.3.2. Valeurs optionnelles

- Un motif de graphe basique permet de rechercher les solutions qui correspondent entièrement à ce motif.
- On peut définir des parties optionnelles dans le motif : clause OPTIONAL

Exemple1 : Soit un schéma avec les relations "marie_avec", "nom" et les types "Homme" et "Femme", afficher le nom des hommes et optionnellement le nom de leur femme

```
PREFIX ex: <http://www.exemple.abc#>

SELECT ?nom_homme ?nom_femme
WHERE {
  ?homme rdf:type ex:Homme .
  ?homme ex:nom ?nom_homme .
  OPTIONAL {
    ?homme ex:marie_avec ?femme .
    ?femme ex:nom ?nom_femme . } }
```

Exemple2: Soit un schéma avec le type "Personne", et la propriété "marie_avec" demandez les personnes non mariées

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?x
WHERE {
  ?x rdf:type ex:Personne .
  OPTIONAL { ?x ex:marie_avec ?y . }
  FILTER (! bound(?y))
}
```

Exemple 3:

Le négation par l'échec n'est pas une négation absolue

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?personne
WHERE {
  ?personne rdf:type ?type .
  FILTER ! ( ?type = ex:Homme )
}
```

5.3.3. Union

- Permet de satisfaire un motif OU un autre.
- On peut différencier les résultats des deux réponses, en utilisant des variables différentes dans les motifs.
- L'opérateur d'union n'enlève pas les doublons

Exemple:

Soit un schéma avec les types "Homme", "Femme", et "Enfant" demandez le nom des femmes et des enfants:

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?nom
WHERE {
    ?femme_ou_enfant ex:nom ?nom .
    {
        ?femme_ou_enfant rdf:type ex:Femme .
    }
    UNION
    {
        ?femme_ou_enfant rdf:type ex:Enfant .
    }
}
```

5.3.4. Modification de la séquence des solutions

- Il existe 6 opérateurs permettant de modifier la séquence des solutions:

1. order by : trier les solutions

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY ASC(?name) DESC(?emp)
```

2. projection pour choisir les variables

3. distinct : éliminer les doublons parmi les solutions

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name WHERE { ?x foaf:name ?name }
```

4. reduced : autoriser l'élimination des doublons par le moteur de requête (pas d'obligation)

5. offset : indiquer à partir de quelle position dans la séquence on démarre

6. limit : borner la taille de la séquence des solutions.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name }
ORDER BY ?name
LIMIT 5
OFFSET 10
```

Cette requête donne au maximum 5 solutions, à partir de la 11ème dans la séquence des solutions.

5.3.5. Autres formes de requêtes :

- **ASK** : retourne un booléen indiquant l'existence d'une solution qui satisfait le motif.

Exemple : Est ce que l'âge de Paul est supérieure à 30 ? . L'URI

de Paul est <http://www.semanticweb.org/vaio/ontologies/2015/10/untitled-ontology-14#Paul>

```
PREFIX px: <http://www.semanticweb.org/vaio/ontologies/2015/10/untitled-ontology-14#>
```

```
ASK
WHERE {
  <http://www.semanticweb.org/vaio/ontologies/2015/10/untitled-ontology-14#Paul> px:age ?age
  FILTER ( ?age>30 )
}
```

- **CONSTRUCT** : pour construire un graphe RDF solution, à partir des valuations des variables.

Exemple . Construire tous les triplets de l'instance de type Homme en utilisant les hommes connus et les personnes masculin connues.

```
PREFIX px: <http://www.semanticweb.org/vaio/ontologies/2015/10/untitled-ontology-14#>
```

```
CONSTRUCT
```

```
{
  ?x rdf:type px:Homme
}
```

```
WHERE
```

```
{
  {
    ?x rdf:type px:Homme
  }
  UNION
```

```
{
  ?x rdf:type px:Masculin .
  ?x rdf:type px:Humain
}
```

- **DESCRIBE** : retourne un graphe RDF décrivant les ressources trouvées. Le résultat dépend du service qui publie la donnée.

Exemple : On peut demander une description générale:

```
DESCRIBE ?Ville
```

```
WHERE {
```

```
  ?Ville px:seTrouveEn ?x .
```

```
  ?x px:nomVille
```

```
http://www.semanticweb.org/vaio/ontologies/2015/10/untitled-ontology-14#Paris
```

```
}
```