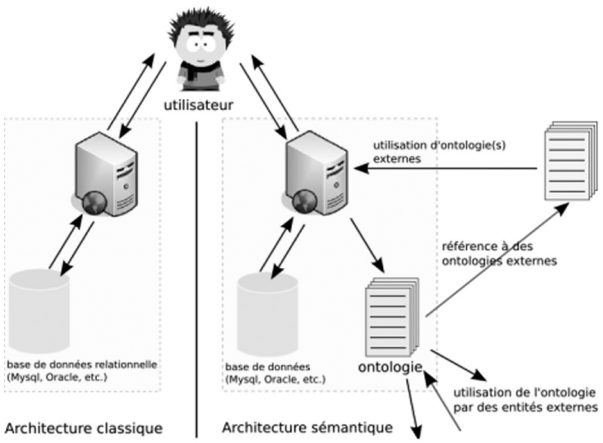
Le langage d'ontologie Web OWL

Francis.Lapique@epfl.ch, Domaine IT

Cet article est le deuxième d'une série consacrée aux technologies du Web sémantique (WS). Le but est de comprendre la place que ces technologies vont prendre **dans l'évolution des services autour du savoir-faire** de **communautés virtuelles**.

Même s'il est encore rare, l'emploi de langages du Web Sémantique au sein d'un site Internet est d'autant plus pertinent que les données concernées sont de nature structurantes ou descriptives. Un bon exemple concerne les systèmes de tags présents sur les weblogs et dans diverses applications, comme par exemple le site de partage de photos numériques Flickr[1] *ou la plateforme de marque-pages collaborative Blogmarks*[2]*, qui pourraient tout à fait être déclinés sous formes d'ontologies OWL. Un tel système permettrait de définir sans ambiguïté les notions utilisées pour taguer des ressources et, en allant plus loin, on pourrait même imaginer que les applications de tags reposent entièrement sur des ontologies, délaissant le modèle établi du recours aux bases de données relationnelles*[3].

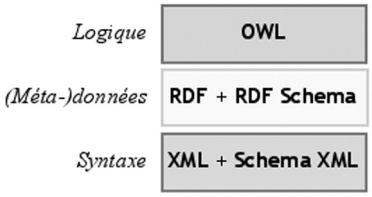


Dans un premier article (FI7/2006[4]), à titre d'introduction, nous avions présenté le couple RDF/RDFS destiné à satisfaire, comme l'indique la traduction du document du W3C[5], les objectifs suivants:

- avoir un modèle de données simple
- avoir une sémantique formelle et une inférence prouvable
- utiliser un vocabulaire extensif, basé sur les URI *(Uniform Resource Identifier)*
- utiliser une syntaxe basée sur XML
- supporter l'utilisation de types de données de schéma XML
- autoriser quiconque à faire des commentaires sur n'importe quelle ressource

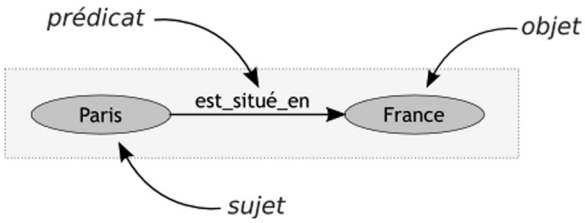
1 *www.flickr.com/*
2 *blogmarks.net/*
3 *www.clever-age.com/spip.php?page=article&id_article=498*
4 *dit.epfl.ch/publications-spip/article.php3?id_article=1175*
5 *lacot.org/w3c/REC-rdf-concepts-20040210*

Ce deuxième volet est consacré au langage OWL[6]. XML, RDF et OWL constituent les trois couches de base du Web Sémantique: XML est le support de sérialisation sur lequel s'appuient RDF et OWL pour définir des structures de données et les relations logiques qui les lient.



En première partie nous allons présenter le langage en suivant une traduction française du chapitre 4 du livre *A Semantic Web Primer*, Grigoris Antoniou et Frank van Harmelen et en seconde, un exemple d'ontologie en s'aidant de l'**éditeur Protégé**, distribué en *open source* par l'université en informatique médicale de Stanford.

Rappelons que la structure fondamentale de toute expression en RDF est une collection de triplets, chacun composé d'un sujet, un prédicat et un objet et que RDFS permet de donner du sens aux informations stockées sous la forme de ces triplets.



Limitations RDFS

Le W3C a mis au point OWL pour étendre RDF qui peut dans certain cas se révéler comme une approche insuffisante. Cette dernière, en effet, compte un certain nombre de limites comme:

- rdfs:range définit le domaine de valeurs d'une propriété quelle que soit la classe concernée. Par exemple, il ne permet pas d'exprimer que les vaches ne mangent que de l'herbe alors que d'autres sortes d'animaux mangent également de la viande.
- RDFS ne permet pas d'exprimer que deux classes sont disjointes. Par exemple, les classes des hommes et des femmes sont disjointes.
- RDFS ne permet pas de créer des classes par combinaison ensembliste d'autres classes (intersection, union, complément). Par exemple, on veut construire la classe **Personne**

6 *www.w3.org/2004/OWL*

comme l'union disjointe des classes des hommes et des femmes.
- RDFS ne permet pas de définir de restriction sur le nombre d'occurrences de valeurs que peut prendre une propriété. Par exemple, on ne peut pas dire qu'une personne a exactement deux parents.
- RDFS ne permet pas de définir certaines caractéristiques des propriétés: transitivité (par exemple: estPlusGrandQue), unicité (par exemple: estLePèreDe), propriété inverse (par exemple: mange est la propriété inverse de estMangéPar).

Si ces contraintes d'expressivité se montrent trop importantes, vous devez passer à OWL.

Le langage OWL

Le langage OWL se compose de trois sous-langages qui proposent une expressivité croissante, chacun conçu pour des communautés de développeurs et des utilisateurs spécifiques: OWL Lite, OWL DL, OWL Full. Chacun est une extension par rapport à son prédécesseur plus simple.

Le langage OWL Lite répond à des besoins de hiérarchie de classification et de fonctionnalités de contraintes simples de cardinalité 0 ou 1. Une cardinalité 0 ou 1 correspond à des relations fonctionnelles, par exemple, une personne a une adresse. Toutefois, cette personne peut avoir un ou plusieurs prénoms, OWL Lite ne suffit donc pas pour cette situation.

Le langage OWL DL concerne les utilisateurs qui souhaitent une expressivité maximum couplée à la complétude du calcul (cela signifie que toutes les inférences seront assurées d'être prises en compte) et la décidabilité du système de raisonnement (c'est- à -dire que tous les calculs seront terminés dans un intervalle de temps fini). Ce langage inclut toutes les structures OWL avec certaines restrictions, comme la séparation des types: une classe ne peut pas aussi être un individu ou une propriété. Il est nommé DL car il correspond à la logique descriptive (que nous présenterons dans un article futur, voir par exemple le raisonneur **Racer**[7]).

Le langage OWL Full se destine aux personnes souhaitant une expressivité maximale. Il a l'avantage de la compatibilité complète avec RDF/RDFS, mais l'inconvénient d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie.

Passons en revue les principaux éléments du langage. Il ne faut pas trop s'attarder à la syntaxe proprement dite qui sera dans la majorité des cas prise en charge par un éditeur.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
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#">
<owl:Ontology rdf:about="">
<rdfs:comment> Un exemple d'ontologie OWL </rdfs:comment>
<owl:imports rdf:resource="http://www.inapg.fr/Personnes"/>
</owl:Ontology>
…
</rdf:RDF>
```

OWL est construit sur RDF et RDFS et utilise la syntaxe RDF/XML (il existe d'autres syntaxes OWL). Tous les fichiers OWL doivent compter une référence à l'*espace de nommage: **http://www.w3.org/2002/07/owl#***.

Eléments classe

owl:Thing et owl:Nothing sont deux classes prédéfinies. Toute classe OWL est une sous-classe d'owl:Thing et une super-classe d'owl:Nothing. Les classes sont définies avec un élément owl:Class (owl:Class est une sous-classe de rdfs: Class).

```
<owl:Class rdf:ID="ProfesseurAssistant">
  <rdfs:subClassOf rdf:resource="#Enseignant"/>
</owl:Class>
```

On peut également exprimer que la classe des ProfesseurAssistant est disjointe de la classe des professeurs et des ingénieurs.

```
<owl:Class rdf:about=" ProfesseurAssistant ">
 <owl:disjointWith rdf:resource="#Professeur"/>
 <owl:disjointWith rdf:resource="#Ingenieur"/>
</owl:Class>
```

On peut définir des équivalences entre classes.

```
<owl:Class rdf:ID="Enseignant">
 <owl:equivalentClass rdf:resource="#Intervenant"/>
</owl:Class>
```

Eléments propriété

Les propriétés OWL donnent la capacité d'exprimer des faits au sujet de ces classes et de leurs instances. Par exemple *la couleur d'un vin*, *son corps*, *sa teneur en sucre* sont des propriétés d'une classe *Vins*. OWL fait la distinction entre deux types de propriétés:
- les propriétés d'objet qui permettent de relier des instances à d'autres instances
- les propriétés de type de donnée qui permettent de relier des individus à des valeurs de données.

Une propriété d'objet est une instance de la classe owl: ObjectProperty, une propriété de type de données est une instance de la classe owl:DatatypeProperty. Ces deux classes sont elles-mêmes sous-classes de la classe RDF rdf:Property.

Exemple de propriété objet

```
<owl:ObjectProperty rdf:ID="estEnseignePar">
<rdfs:domain rdf:resource="#cours"/>
<rdfs:range rdf:resource="#enseignant"/>
<rdfs:subPropertyOf rdf:resource="#implique"/>
</owl:ObjectProperty>
```

La propriété *estEnseignePar* est attachée (attribut *domain*) à la classe cours, son étendue (attribut *range*) est la classe *enseignant*, c'est une extension de la propriété *implique*.

[7] ***www.racer-systems.com***

Exemple de propriété typée

```
<owl:DatatypeProperty rdf:ID="Age">
<rdfs:range rdf:resource="http://www.
w3.org/2001/XMLSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>
```

Dans ce cas, Age fait correspondre aux instances de la classe des entiers positifs.

Il est possible de définir des propriétés inverses:

```
<owl:ObjectProperty rdf:ID="Enseigne">
<rdfs:domain rdf:resource="#enseignant"/>
<rdfs:range rdf:resource="#cours"/>

<rdfs:inverseOf rdf:resource="#estEnseignePar"/>
</owl:ObjectProperty>
```

et des équivalences de propriétés:

```
<owl:ObjectProperty rdf:ID="faitCours">
<rdfs:equivalentProperty rdf:resource="#Enseigne"/>
</owl:ObjectProperty>
```

Restriction sur les propriétés

L'élément `owl:Restriction` permet de définir une classe anonyme (ie non définie par un ID). La restriction peut s'exprimer sur le *range* ou le *domain* d'une propriété ou sur la cardinalité d'une propriété.

Par exemple: on définit une classe anonyme regroupant des cours uniquement enseignés par des enseignants ayant le titre de professeur.

```
<owl:Restriction>
 <owl:onProperty rdf:resource="#estEnseignePar"/>
 <owl:allValuesFrom rdf:resource="#professeur"/>
</owl:Restriction>
```

On peut ensuite définir une classe comme sous-classe d'une classe anonyme définie par une restriction. Par exemple: chaque instance de cours de 1ère année ne peut être enseignée que par un enseignant professeur (contrainte avec quantificateur universel).

```
<owl:Class rdf:about="#cours1ereAnnee">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#estEnseignéPar"/>
<owl:allValuesFrom rdf:resource="#professeur"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

On peut exprimer une restriction de nature existentielle, par exemple: un enseignant doit enseigner au moins un cours de master.

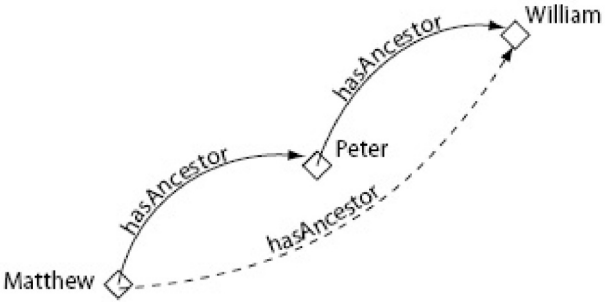
```
<owl:Class rdf:about="#Enseignant">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#enseigne"/>
<owl:someValuesFrom rdf:resource="#coursMaster"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

On peut exprimer une contrainte de cardinalité, par exemple, le fait qu'un cours doit être enseigné par au moins un enseignant et au plus par trois.

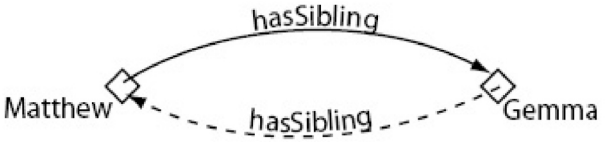
```
<owl:Class rdf:about="#cours">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#estEnseignePar"/>
<owl:minCardinality rdf:datatype="&xsd;nonNegati
veInteger">1
</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>

<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#estEnseignePar"/>
<owl:maxCardinality rdf:datatype="&xsd;nonNegati
veInteger">3
</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Propriétés spécifiques des propriétés

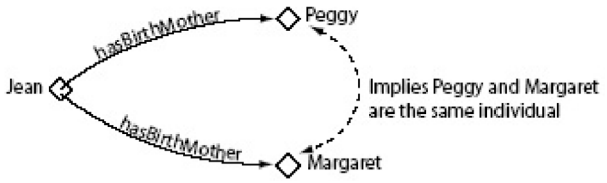
On peut définir des propriétés mathématiques sur les éléments propriétés. Une propriété P est dite transitive si pour tout x, y, z, P(x,y) et P(y,z) implique P(x,z). La propriété `hasAncestor` étant déclarée transitive vous pouvez déduire des deux relations entre Matthew et Peter et entre Peter et William celle entre Matthew et William.



La figure suivante illustre une propriété symétrique. Si Mathhew est relié à Gemma par la propriété symétrique `hasSibling`, vous pouvez en déduire que Gemma a un lien de fraternité avec Matthew.

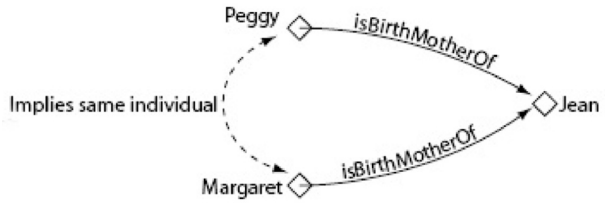


Si une propriété P est marquée comme fonctionnelle, alors pour tout x, y, z, P(x,y) et P(x,z) implique y = z.



Dans l'exemple de la figure précédente, si vous avez déclaré que Peggy et Margaret sont deux individus différents, alors vous avez un exemple d'inconsistance.

Si une propriété P est marquée comme inverse fonctionnelle, alors pour tout x, y, z, P(y,x) et P(z,x) implique y = z:



Opérateurs de combinaison de classes

On peut définir des classes par combinaison ensembliste (union, intersection, complément) d'autres classes. Par exemple, on définit la classe `PersonneUniversite` comme l'union de la classe des enseignants et de celle des étudiants.

```
<owl:Class rdf:about="#PersonneUniversite">
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Enseignant"/>
<owl:Class rdf:about="#Etudiant"/>
</owl:unionOf>
</owl:Class>
```

Un autre exemple mettant en œuvre trois opérateurs ensemblistes. Le personnel administratif est défini comme le personnel de l'Université n'étant ni personnel enseignant, ni personnel technique.

```
<owl:Class rdf:about="#PersonnelAdministratif">
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="#PersonnelUniversite"/>
<owl:Class>
<owl:complementOf>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Enseignant">
<owl:Class rdf:about="#Technicien">
</owl:unionOf>
</owl:Class>
</owl:complementOf>
</owl:Class>
</owl:intersectionOf>
<owl:Class>
```

Enumérations

On peut définir une classe par énumération avec l'élément `owl:oneOf`. Par exemple, on définit la classe des jours de la semaine.

```
<owl:Class rdf:about="#jourDeLaSemaine">
<owl:oneOf rdf:parseType="Collection">
<owl:Thing rdf:about="#Lundi">
<owl:Thing rdf:about="#Mardi">
<owl:Thing rdf:about="#Mercredi">
<owl:Thing rdf:about="#Jeudi">
<owl:Thing rdf:about="#Vendredi">
<owl:Thing rdf:about="#Samedi">
<owl:Thing rdf:about="#Dimanche">
</owl:oneOf>
</owl:Class>
```

Instances

Les instances d'une classe OWL se déclarent avec l'élément RDF rdf:type.

```
<rdf:Description rdf:ID="1234">
<rdf:type rdf:resource="#enseignant"/>
</rdf:Description>
```

ou, de manière équivalente,

```
<enseignant rdf:ID="1234"/>
```

On peut également compléter la déclaration. Par exemple:

```
<enseignant rdf:ID="1234">
<uni:age rdf:datatype="&xsd;integer">43</uni:age>
</enseignant>
```

A la différence des bases de données, OWL ne fait pas l'hypothèse de l'unicité de nommage. Le fait que deux instances aient un ID différent n'implique pas qu'ils soient des individus différents. Par exemple, la propriété `estEnseignePar` indique qu'un cours est enseigné par au plus un enseignant:

```
<owl:ObjectProperty rdf:ID="#estEnseignePar">
<rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

Si l'on déclare que le cours Bases de données est enseigné par les enseignants 1234 et 1235, un interpréteur OWL ne conclut pas à une erreur mais à l'égalité des ressources 1234 et 1235.

```
<cours rdf:ID="Bases de données">
<estEnseignéPar rdf:resource="#1234"/>
<estEnseignéPar rdf:resource="#1235"/>
</cours>
```

Pour indiquer que deux individus sont différents, il faut le spécifier explicitement avec l'élément `differentFrom`:

```
<enseignant rdf:ID="#1234">
<owl:differentFrom rdf:resource="#1235"/>
</enseignant>
```

L'élément `AllDifferent` permet de définir les inégalités de couples d'individuels d'une liste donnée. Par exemple,

```
<owl:AllDifferent>
<owl:distinctMembers rdf:parseType="Collection">
<enseignant rdf:ID="#1234"/>
<enseignant rdf:ID="#1235"/>
<enseignant rdf:ID="#1236"/>
</owl:distinctMembers>
</owl:AllDifferent>
```

Editeur d'ontologie Protégé

Protégé[8] est un éditeur distribué en *open source* par le Stanford Medical Informatics[9] au sein de la Stanford University School of Medicine. Protégé offre un *plugin* dédié à OWL. Dans sa dernière version il propose également un moteur SPARQL, dont l'objectif est de fournir un protocole et un langage de requêtes similaire à SQL, mais adapté à RDF.

[8] *protege.stanford.edu*
[9] *smi.stanford.edu*

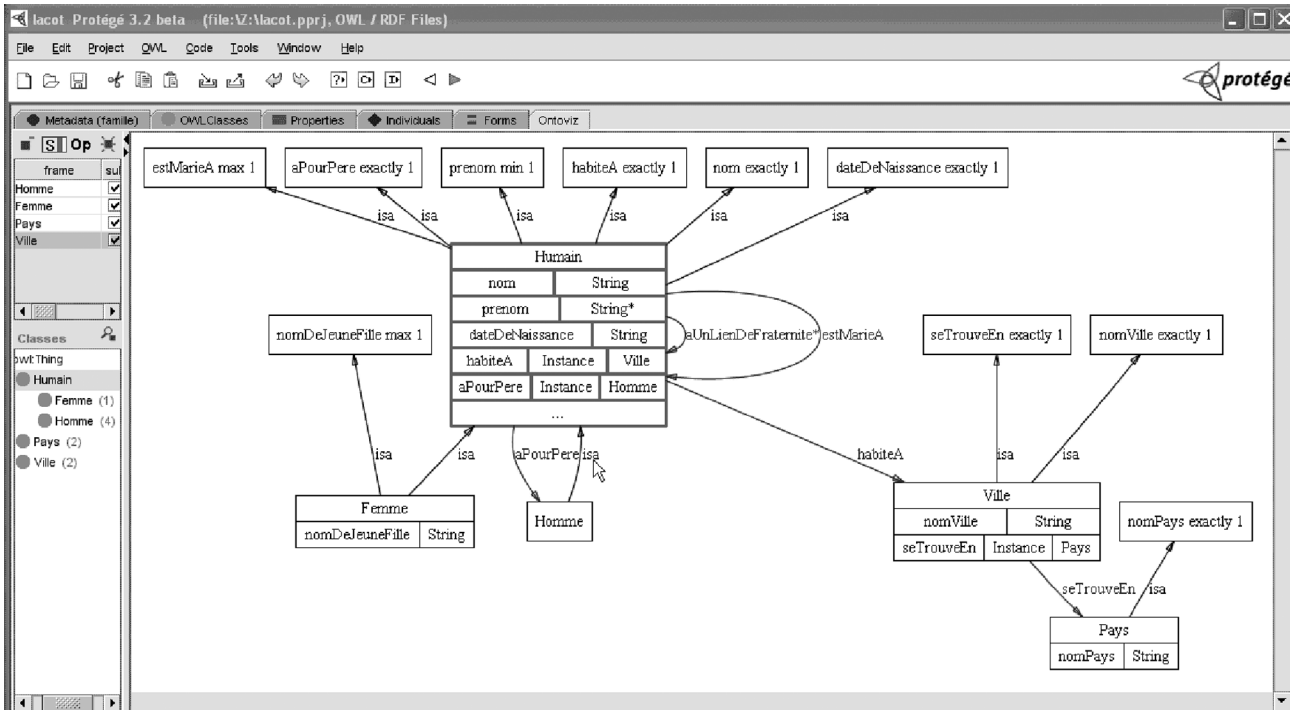


Fig. 1

La figure 1 visualise l'exemple de description d'une population que X. Lacot[10] donne dans son introduction à OWL. Les onglets OWL Classes, Properties, Individuals et Forms font respectivement référence à l'éditeur de classes, de propriétés, d'instances, et aux formulaires de saisie des instances.

Vous y trouvez l'ensemble des classes *Humain*, *Femme, Homme, Pays* et *Ville* l'ensemble des propriétés d'objets *habiteA, aPourPere, aUnLienDeFraternite, estMarieA, seTrouveEn* et propriétés de types de données, *nom, prenom, nomDeJeuneFille, dateDeNaissance, nomVille et nomPays.*

La figure 2 montre la définition de la classe *Humain*, elle hérite de la classe `owl:Thing`, elle est caractérisée par un ensemble de propriétés dont *aPourPere* de cardinalité exactement égale à 1.

La propriété symétrique *aUnLienDeFraternite* est attachée à la classe *Humain* et a pour image cette même classe *Humain* (fig. 3)*:*

Figure 4: un exemple d'instance de la classe *Homme* qui hérite de la classe *Humain* avec un ensemble de propriétés.

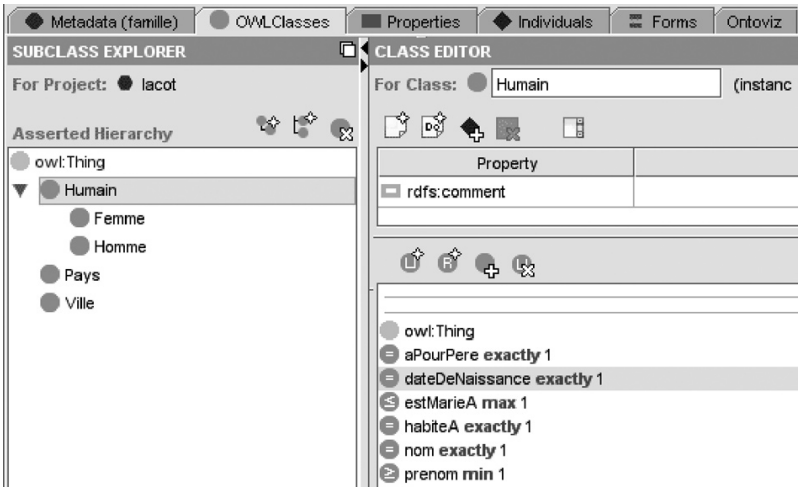


Fig. 2

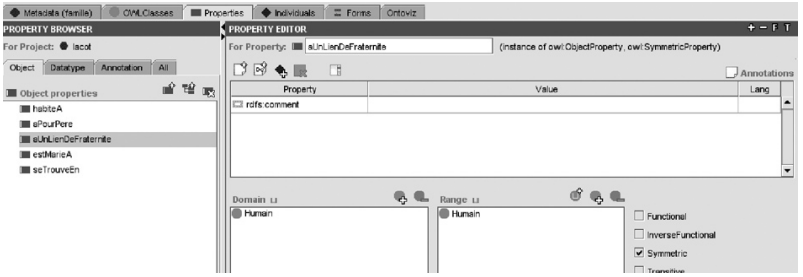


Fig. 3

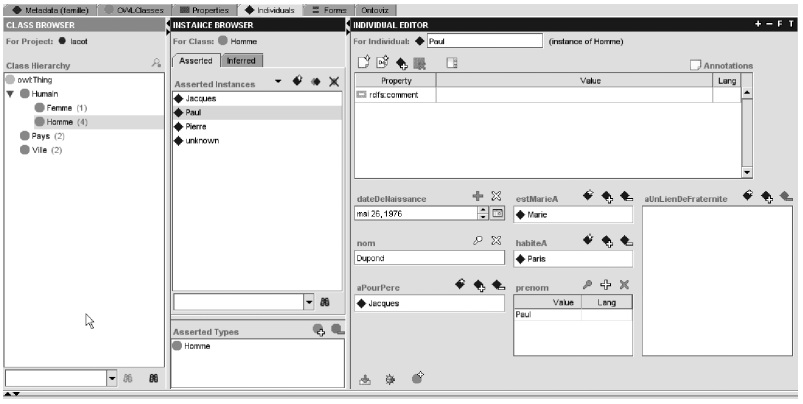


Fig. 4

[10] *lacot.org/public/introduction_a_owl.pdf*

Sous une forme graphique (fig. 5) l'ensemble des instances de cet exemple.

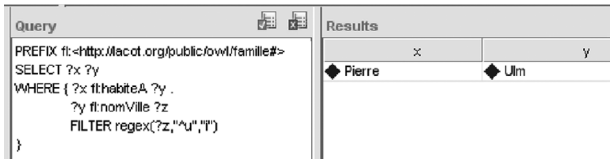
Sous la rubrique owl on peut demander l'ouverture d'une fenêtre pour exécuter des requêtes SPARQL. Comme exemple de requête demandons l'ensemble des instances *Humain* qui habitent dans une ville commençant par le caractère U ou u.

Sa traduction en langage SPARQL:

```
PREFIX fl:<http://lacot.org/public/owl/famille#>
SELECT ?x ?y
WHERE { ?x fl:habiteA ?y .
   ?y fl:nomVille ?z
    FILTER regex(?z,"^U","i")
}
```

Le mot-clé PREFIX associe un label ici fl à une URI (Uniform Resource Identifier).

?x fl:habiteA ?y, on va s'intéresser au sujet ?x lié à l'objet ?y par le prédicat propriété habiteA et au prédicat nomVille que l'on filtre par une expression régulière.



Conclusion

Avant de vous lancer dans un projet OWL consultez les bibliothèques d'ontologies comme: ***http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary***, ***http://www.schemaweb.info/*** ou plus largement faites appel à Google: ***http://www.google.ch/search?q=filetype:owl+owl***.

Nous allons revenir plus complètement, dans un troisième article, sur ce langage de requête SPARQL du W3C qui sera illustré avec le projet *The Friend of a Friend (FOAF)* une ontologie *amicale*[11]. Puis nous poursuivrons la route par un quatrième sur les outils de raisonnement. ■

[11] ***www.foaf-project.org***

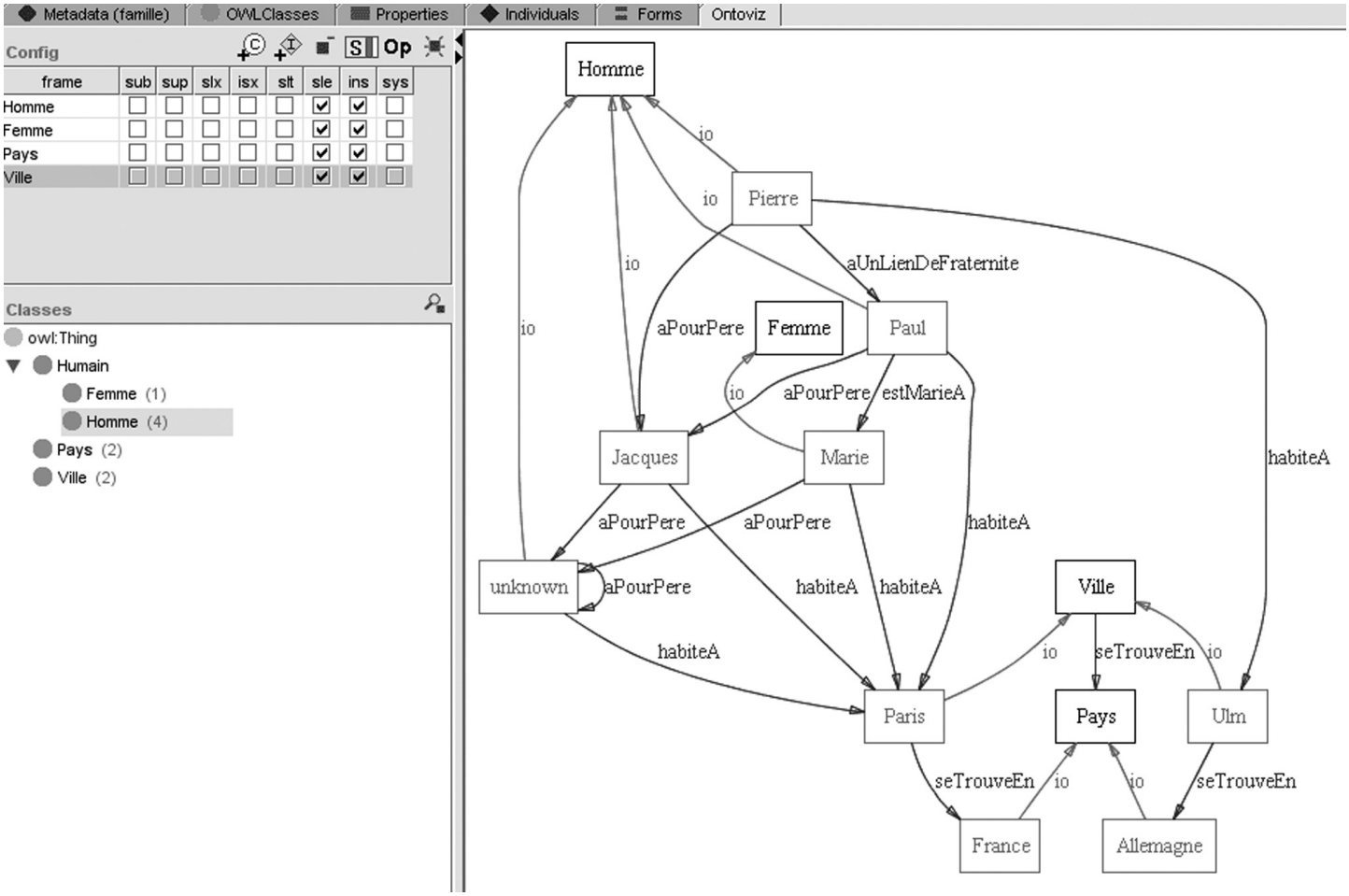


fig. 5