Introduction à OWL, un langage XML d'ontologies Web

auteur : Xavier Lacot <xavier@lacot.org>

Juin 2005

Résumé

En l'espace de 15 ans, le concept du « web » a fortement évolué. La source disparate de savoirs qu'était le « réseau des réseaux » dans les années 1990 cède progressivement la place, ces dernières années, à l'envie d'une meilleure architecture des échanges de connaissance que permet le web. Ce document présente sommairement quelques concepts du web sémantique, et s'attarde principalement sur le langage d'ontologies web recommandé par le W3C : OWL.

Mots Clés. Web sémantique, eXtensible Markup Language, Resource Description Framework, Web Ontology Language, Métadonnée.

Table des matières

Table des figures

Glossaire et définitions

DOM	Document Object Model
DTD	Document Type Definition
HTML	HyperText Markup language
Jena	Framework pour le web sémantique en Java, Jena fournit un environnement permettant de travailler avec RDF, RDFS et OWL.
Métadonnée	Une métadonnée est une information permettant de décrire une autre information, quels qu'en soient la nature et le support.
Ontologie	A l'origine domaine philosophique de la « science de l'être en tant qu'être », une ontologie est, selon l'entendement du Web sémantique, un ensemble structuré de savoirs. Une ontologie définit les termes employés pour décrire et représenter un domaine de connaissance.
OWL	Web Ontology Langage
Protégé	éditeur d'ontologies et framework de gestion des connaissances, développé en open-source au sein de l'université de médecine de Stanford.
RDF	Ressource Description Framework, permet de présenter des données et des métadonnées.
RDFS	RDF Schema, permet de définir des vocabulaires RDF.
SAX	Simple API for XML
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Introduction

A sa création par Tim Berners Lee, au début des années, l'objectif du web était de permettre des échanges rapides de savoirs entre individus distants. C'est dans ce but qu'a été créé le langage HTML, autorisant une mise en forme aisée et rapide documents en ligne.

Dix ans plus tard, au début du XXI^{ème} siècle, la « balkanisation » du Web a bien eu lieu, si on me permet d'emprunter un terme consacré par OpenWeb [OPE 2005]. Tant au niveau de la présentation que du balisage ou du respect des standards, les documents présents sur le web ne constituent plus, de par leur hétérogénéité, une source de savoir fiable et agréable à consulter. Les connaissances que proposent les pages Web ont peu à peu été enfermées, au grès des modes, dans une couche présentationnelle qui, si elle n'a pas dénaturé l'information, l'a en tout cas rendue moins accessible.

Depuis plusieurs années, toutefois, une nouvelle idée du Web prend corps : celle d'un Web Sémantique. Peu à peu, le World Wide Web Consortium (W3C) se dote de nouveaux langages et d'outils plus performants : XML, RDF, OWL, n'en sont que des exemples. Tous ont pour objectif commun de participer à une formalisation des savoirs, en permettant ainsi un meilleur partage et une transmission plus aisée. Les champs d'application éventuels sont vastes : raisonnement automatique, résolution de problèmes par inférences, représentation d edonées structurées, traduction automatisée, etc.

On présentera donc, dans une première partie, les langages XML et RDF, ainsi que les principes fondamentaux du Web sémantique. La deuxième partie, qui constitue l'introduction à OWL proprement dite, s'appuie largement sur les Recommandations du W3C. Enfin, une dernière partie propose quelques exemples de logiciels en lien direct avec OWL et le Web sémantique.

1 Introduction générale à XML, RDF et au web sémantique

1.1 Présentation succinte de XML

Recommandation du W3C depuis le 10 février 1998, le langage eXtensible Markup Language (XML) connait depuis ses débuts un succés indéniable. Défini dés son origine comme un méta-langage facilitant l'élaboration de langages à balises spécialisés, XML a conquis depuis de multiples formats, les nouveaux comme les anciens. De xhtml au format de documents openoffice.org, nombreux sont les documents qui profitent aujourd'hui du cadre de XML.

Les normes strictes qui gèrent la syntaxe et la structure de XML rendent le langage et son utilisation plus aisés, favorisant notamment la factorisation des travaux de développement d'analyseurs syntaxiques. XML fournit donc un cadre de structuration des données qui peut être employé pour la définition rapide et sans ambiguité syntaxique d'un format de document.

1.1.1 Structure d'un fichier XML

Une source de données est un document XML si elle est « bien formée », c'est à dire si elle correspond parfaitement à la spécification de XML [W3C 2004a]. Un document XML dispose alors de deux structures : une structure logique et une structure physique, les deux correspondant parfaitement.

Un document XML est représenté physiquement sous la forme d'un fichier texte structuré en éléments, à l'aide de balises éventuellement imbriquées. En en-tête du document doit figurer un « prologue », une déclaration qui identifie le document comme un document XML. Ce prologue indique la version de XML employée, le codage de caractères, et si le document est associé à une DTD ou s'il est autonome.

Il existe un élément particulier : l'élément « racine », encore appelé « élément document ». Cette racine doit contenir tous les autres éléments du document et ne peut apparaître qu'une fois dans un document XML. En conséquence, aucun autre élément ne contient la racine.

1.1.2 Elements syntaxiques

En complément des principes fondamentaux de XML expliqués ci-dessus, on peut ajouter quelques règles syntaxiques :

- XML est sensible à la casse
- chaque élément doit commencer par une balise ouvrante et se termine par une balise fermante. Eventuellement, un élément vide peut être représenté par une balise simple.
- l'imbrication des éléments du document se fait sans chevauchement. Concrètement, cela signifie que si la balise ouvrante d'un élément se trouve à l'intérieur d'un élement parent, la balise fermante se trouvera dans le même élément.
- La valeur d'un attribut doit être encadrée de guillemets, simples ou doubles.

1.1.3 Exemple de document XML simple

L'exemple suivant de document XML permet de décrire une personne :

```
<?xml version="1.0" encoding="UTF-8"?>
<personne>
      <nom>Dupond</nom>
      <prenom>Jean</prenom>
      <naissance>
            <lieu>
                  <ville>Paris</ville>
                  <pays>France</pays>
            </lieu>
            <date>
                  <jour>14</jour>
                  <mois>7</mois>
                  <annee>1789</annee>
            </date>
      </naissance>
</personne>
```

Illustration 1 : Exemple de fichier XML simple

1.1.4 Propriétés additionnelles

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. Il est à noter que certaines des déclarations de type de document – les DTD notamment – ne permettent pas de définir le type des attributs, limitant dans ces cas la notion de validité associée aux fichiers XML de ce type.

Un document XML peut être transformé en utilisant des feuilles de style XSLT en un nouveau document : fichier XML, évidemment, mais également tout autre format texte (pages html, fichier pdf, requête SQL, etc.).

Enfin, XML propose différentes interfaces de programmation (API) qui facilitent la manipulation de documents par les processeurs XML. Les deux API principales sont Document Object Model (DOM), qui procède par construction et manipulation d'un arbre logique complet, et Simple API for XML (SAX), qui permet de débuter le traitement d'un document XML avant d'en connaître le contenu complet.

1.2 Présentation de RDF et de RDFS

1.2.1 Contexte et Web sémantique

A sa création par Tim Berners Lee, au début des années 1990, le web était exclusivement destiné à partager des informations sous forme de pages html, affichables par un logiciel « navigateur web », et généralement destinées à être lues par un utilisateur humain.

Très rapidement, on s'est rendu compte que cette conception du web était bien trop limitée, et ne permettait pas un réel partage du savoir : tout au plus cela permettait-il de présenter des connaissances, mais en aucun cas de les rendre directement utilisables.

L'arrivée de XML, en 1998, a donné un cadre à la structuration des connaissances, rendant ainsi possible la création de nouveaux langages web destinés non plus à un rendu graphique à l'écran pour un utilisateur humain, mais à un réel partage et à une manipulation des savoirs. C'est dans cet esprit qu'a été créé en 1999 RDF, un langage XML permettant de décrire des métadonnées et facilitant leur traitement.

1.2.2 Objectif de RDF

Le développement de RDF par le W3C a été entre autres motivé par la perspective des applications suivantes :

- manipulation et classification des métadonnées Web, afin de fournir des informations sur les ressources Web et les systèmes qui les utilisent.
- développement de modèles d'information ouverts plutôt que fixés pour certaines applications (par exemple les activités de planification, de description de processus organisationnels, d'annotation de ressources web, etc.)
- faire avec l'information traitable par machine ce que le Web a fait pour l'hypertexte, en permettant aux informations d'être manipulées en dehors de l'environnement particulier dans lequel elles ont été créées, éventuellement à l'échelle d'Internet. C'est le concept d'interopérabilité des savoirs.
- optimisation de la coopération entre applications, en permettant de combiner les données de plusieurs applications, pour générer de nouvelles informations.
- faciliter le traitement automatique de l'information du Web par des agents logiciels, transformant ainsi le web d'un regroupement d'informations uniquement destinées aux humains, en un état de réseau de processus en coopération. Dans ce réseau, le rôle de RDF est de fournir une *lingua franca* compréhensible par tous les agents.

Pour ce faire, RDF procède par une description de savoirs (données tout comme métadonnées) à l'aide d'expressions de structure fixée. En effet, la structure fondamentale de toute expression en RDF est une collection de triplets, chacun composé d'un sujet, un prédicat et un objet. Un ensemble de tels triplets est appelé un graphe RDF. Ceci peut être illustré par un diagramme composé de noeuds et d'arcs dirigés, dans lequel chaque triplet est représenté par un lien noeud-arc-noeud (d'où le terme de "graphe").

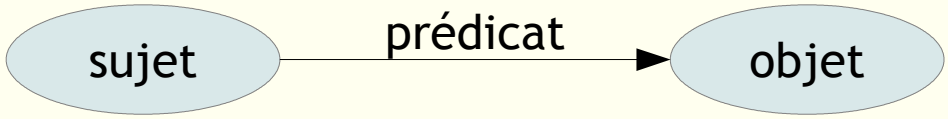


Illustration 2 : Triplet RDF

Dans un graphe, chaque triplet représente l'existence d'une relation entre les choses symbolisées par les noeuds qui sont joints.

1.2.3 Syntaxe de RDF

Jusqu'à présent, il n'a pas été fait mention de XML dans la description de RDF. RDF n'est, en soi, qu'un modèle de graphes à arcs orientés et labellés, dont le label est le « prédicat » (ou « propriété »), le noeud de départ le « sujet », et le noeud cible « l'objet ». RDF/XML propose une syntaxe de sérialisation de RDF. Pour cela, RDF/XML s'appuie sur XML par sa syntaxe, définie par une Recommandation du W3C de 1998, et mise à jour en 2004 [W3C 2004a].

Considérons l'information suivante : « l'auteur de http://www.lacot.org/ est Xavier Lacot ». On peut choisir de représenter cette information en triplet RDF par la chaine « {auteur de http://www.lacot.org/, Xavier Lacot, est} », ou encore « est(auteur de http://www.lacot.org/, Xavier Lacot) », ou encore, plus communément, « <auteur de http://www.lacot.org/><est><Xavier Lacot> ». En RDF/XML, cette information s'écrit :

```
<rdf:Description about="http://www.lacot.org/">
      <schema:auteur>Xavier Lacot</schema:auteur>
</rdf:Description>
```

Illustration 3 : Exemple de triplet RDF/XML

Dans l'exemple ci-dessus, 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. Un fichier RDF/XML ne contenant que l'information « l'auteur de http://www.lacot.org/ est Xavier Lacot » peut s'écrire de la façon suivante :

```
<?xml version="1.0"?>
<rdf:rdf xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                              xmlns:prefixe="http://source_de_schema/schema">
      <rdf:Description about="http://www.lacot.org/">
            <prefixe:auteur>Xavier Lacot</prefixe:auteur>
      </rdf:Description>
</rdf:rdf>
```

Illustration 4 : Exemple de fichier RDF/XML - première version

Ce fichier pourrait plus simplement s'écrire, en employant un espace de nom par défaut :

```
<?xml version="1.0"?>
<rdf xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:prefixe="http://source_de_schema/schema">
      <description about="http://www.lacot.org/">
            <prefixe:auteur>Xavier Lacot</prefixe:auteur>
      </description>
</rdf:rdf>
```

Illustration 5 : Exemple de fichier RDF/XML - deuxième version

La nécessité de déclarer l'espace de nom employé nous ammène au Schéma RDF (RDFS), présenté dans la partie suivante.

1.2.4 RDFS, pour donner du vocabulaire à RDF

Comme on le voit dans l'exemple précédent, la propriété « auteur » n'a de sens pour décrire la ressource « http://www.lacot.org/ » que dans un contexte bien défini :

- le lecteur (utilisateur) est humain
- ce lecteur comprend le français
- l'information transmise par le triplet RDF « {http://www.lacot.org/, Xavier Lacot, auteur} » est suffisamment triviale pour comprendre que sa signification est « Xavier Lacot est l'auteur de http://www.lacot.org/ ».

Il est donc nécessaire, pour donner un sens aux informations stockées sous forme de triplets RDF, de se donner un vocabulaire, de définir la signification de la propriété « auteur », ainsi que son type, son champ de valeurs, etc. C'est le rôle de RDF Schema, qui permet de créer des vocabulaires de métadonnées.

1.3 Le web sémantique et les ontologies

On a vu que RDF et RDFS permettent de définir, sous forme de graphes de triplets, des données ou des métadonnées. Cependant, de nombreuses limitations bornent la capacité d'expression des connaissances établies à l'aide de RDF/RDFS. On peut citer, par exemple, l'impossibilité de raisonner et de mener des raisonnements automatisés (automated reasoning) sur les modèles de connaissances établis à l'aide de RDF/RDFS. C'est ce manque que se propose de combler OWL.

1.3.1 Ontologie

Web Ontology Language (OWL) doit son nom au terme « ontologie », un mot emprunté à la philosophie qui, s'il est d'origine grecque, ne fut manifestement créé qu'au XVIIe siècle. « Discours sur l'être en tant qu'être » selon Aristote, l'ontologie prend un tout autre sens en informatique, où le terme désigne un ensemble structuré de savoirs dans un domaine de connaissance particulier.

On distingue généralement deux entités globales au sein d'une ontologie. La première, à objectif terminologique, définit la nature des élements qui composent le domaine de l'ontologie en question, un peu comme la définition d'une classe en programmation orientée objet définit la nature des objets que l'on va manipuler par la suite. La seconde partie d'une ontologie explicite les relations entre plusieurs instances de ces classes définies dans la partie terminologique.

Ainsi, au sein d'une ontologie, les concepts sont définis les uns par rapport aux autres (modèle en graphe de l'organisation des connaissances), ce qui autorise un raisonnement et une manipulation de ces connaissances.

1.3.2 De nombreux langages de définition et de manipulation d'ontologies

Evidemment, cette définition très générale laisse libre cours à l'imagination au sujet des applications potentielles : gestion de ressources, management, création de référentiels médicaux, etc. C'est sans doute la raison pour laquelle il existe de nombreux langages informatiques, plus ou moins récents, spécialisés dans la création et la manipulation d'ontologies. En voici quelques exemples :

- Open Knowledge Base Connectivity : OKBC 2.0, pubié en 1997, est une API permettant d'accéder à des bases de connaissance (Knowledge Representation System). [OKB 1998]
- Knowledge Interchange Format est un langage destiné à faciliter des échanges de savoirs entre des systèmes informatiques disparates. Son objetif n'est pas de permettre une interaction avec l'être humain (bien que cela reste possible), mais plutôt une coopération entre systèmes hétérogènes. [KIF 1998]
- LOOM est un langage de représentation des connaissances dont le but avoué est de « permettre la construction d'applications intelligentes ». [LOO 2004]
- DARPA Agent Markup Language : fondé sur XML et RDF, DAML-ONT est apparu en Octobre 2000 suite à un effort du DARPA (Defense Advanced Research Projects Agency), afin d'autoriser l'expression de classes RDF plus poussées que ce que permettait à l'époque RDFS. Le programme DAML se poursuit encore à l'heure actuelle. [DAR 2005].

En réaction à l'apparition de ces nombreux langages poursuivant pour la plupart des buts communs, le World Wide Web Consortium (W3C) a mis sur pieds, en Novembre 2001, le groupe de travail « WebOnt », chargé d'étudier la création d'un langage standard de manipulation d'ontologies web. Le premier Working Draft « OWL Web Ontology Language 1.0 Abstract Syntax » paraît en Juillet 2002 et, au final, OWL devient une Recommandation du W3C le 10 Février 2004. [W3C 2004a]

2 Le langage OWL

Cette section constitue l'introduction à OWL proprement dite. Elle s'appuie grandement sur les Recommandations du W3C du 10 février 2004 (cf. Bibliographie).

2.1 Objectifs et motivation

OWL est, tout comme RDF, un langage XML profitant de l'universalité syntaxique de XML. Fondé sur la syntaxe de RDF/XML, OWL offre un moyen d'écrire des ontologies web. OWL se différencie du couple RDF/RDFS en ceci que, contrairement à RDF, il est justement un langage d'ontologies. SI RDF et RDFS apportent à l'utilisateur la capacité de décrire des classes (ie. avec des contructeurs) et des propriétés, OWL intègre, en plus, des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc.

Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu web que RDF et RDFS, grace à un vocabulaire plus large et à une vraie sémantique formelle.

2.2 Différentes déclinaisons de OWL

Plus un outil est complet, plus il est, en général, complexe. C'est cet écueil qu'a voulu éviter le groupe de travail WebOnt du W3C en dotant OWL de trois sous-langages offrant des capacités d'expression croissantes et, naturellement, destinés à des communautés différentes d'utilisateurs :

- OWL Lite est le sous langage de OWL le plus simple. Il est destiné aux utilisateurs qui ont besoin d'une hiérarchie de concepts simple. OWL Lite est adapté, par exemple, aux migrations rapides depuis d'anciens thésaurus. [W3C2004j]
- OWL DL est plus complexe que OWL Lite, permettant une expressivité bien plus importante. OWL DL est fondé sur la logique descriptive (d'où son nom, OWL Description Logics), un domaine de recherche étudiant la logique, et conférant donc à OWL DL son adaptation au raisonnement automatisé. Malgré sa complexité relative face à OWL Lite, OWL-DL garantit la complétude des raisonnements (toutes les inférences sont calculables) et leur décidabilité (leur calcul se fait en une durée finie).
- OWL Full est la version la plus complexe d'OWL, mais également celle qui permet le plus haut niveau d'expressivité. OWL Full est destiné aux situations où il est plus important 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. OWL Full offre cependant des mécanismes intéressants, comme par exemple la possibilité d'étende le vocabulaire par défaut de OWL.

Il existe entre ces trois sous langage une dépendance de nature hiérarchique : toute ontologie OWL Lite valide est également une ontologie OWL DL valide, et toute ontologie OWL DL valide est également une ontologie OWL Full valide.

2.3 Document OWL

Tout comme RDF 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 :

2.3.1 Espaces de nommage de OWL

L'intégralité du vocabulaire de OWL provient de l'espace de nom de OWL, `http://www.w3.org/2002/07/owl#`.

2.3.2 Type MIME

Il n'existe aucun type MIME spécifique à OWL, un document OWL étant un document RDF. Il est donc 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`.

2.3.3 Extension

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 ».

2.4 Structure d'une ontologie OWL

Cette partie indique les principaux éléments constituant une ontologie OWL. Avant de poursuivre, il est bien nécessaire de comprendre qu'OWL n'a pas été conçu dans un esprit fermé permettant de borner les frontières d'une ontologie. Au contraire, la conception d'OWL a pris en compte la nature distribuée du web sémantique et, de ce fait, a intégré la possibilité d'étendre des ontologies existantes, ou d'employer diverses ontologies existantes pour compléter la définition d'une nouvelle ontologie.

2.4.1 Espaces de nommage

Afin de pouvoir employer des termes dans une ontologie, il est nécessaire d'indiquer avec précision de quels vocabulaires ces termes proviennent. C'est la raison pour laquelle, comme tout autre document XML, une ontologie commence par une déclaration d'espace de nom (parfois appelée « de nommage ») contenue dans une balise `rdf:RDF`. Supposons que nous souhaitons écrire une ontologie sur une population de personnes ou, d'une manière plus générale, sur l'humanite. Voici la déclaration d'espace de nom qui pourrait être employée :

```
<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 décalarations identifient l'espace de nommage propre à l'ontologie que nous sommes en train d'écrire. 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 déclaration 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.

Afin de simplifier l'écriture des URI dans la déclaration d'espace de nom et, surtout, dans les valeurs des attributs de l'ontologie, il est conseillé de définir des abbréviations au moyen d'entités de type de document :

```
<!DOCTYPE rdf:RDF [
    <!ENTITY humanite "http://domain.tld/path/humanite#" >
    <!ENTITY vivant "http://otherdomain.tld/otherpath/vivant#" >
]>
```

Ainsi, la déclaration d'espace de nom initiale devient :

```
<rdf:RDF
    xmlns         = "&humanite;"
    xmlns:humanite= "&humanite;"
    xmlns:base    = "&humanite;"
    xmlns:vivant  = "&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#">
```

2.4.2 En-têtes d'une ontologie

Tout comme il existe une section d'en-tête `<head>..</head>` en haut de tout document XHTML bien formé, on peut écrire, à la suite de la déclaration d'espaces de nom, un en-tête décrivant le contenu de l'ontologie courante. C'est la balise `owl:Ontology` qui permet d'indiquer ces informations :

```
<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>
    ...
```

Le contenu et la signification des différents éléments de cette section d'en-tête sont parfaitement décrits dans « OWL Web Ontology Language Reference » [W3C 2004k], et il est recommandé de s'y reporter pour plus de détails.

2.5 Eléments du langage

Cette partie ne va pas reprendre toutes les finesses de OWL Lite, OWL DL et OWL Full, mais uniquement les plus importantes. Pour des explications exhaustives du rôle de chacun des

éléments composant le vocabulaire d'OWL, il est recommandé de se reporter à la Recommandation du W3C « OWL Web Ontology Language Reference » [W3C 2004k].

2.5.1 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. Les trois versions d'OWL comportent les mêmes mécanismes de classe, à ceci près que OWL FULL est la seule version à permettre qu'une classe soit l'instance d'une autre classe (d'une métaclasse). A l'inverse, OWL Lite et OWL DL n'autorisent pas qu'une instance de classe soit elle-même une classe.

2.5.1.1 Déclaration de classe

La déclaration d'une classe se fait par le biais du mécanisme de « description de classe », qui se présente sous diverses formes. Une classe peut ainsi se déclarer de six manières différentes :

- 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 », crée en plaçant des contraintes sur son extension.

- l'énumération des individus composant la classe :

  Ce type de descrition 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>
  ```

  Si ce mécanisme est utilisable avec OWL DL et OWL FULL, il ne fait cependant pas partie de OWL Lite.

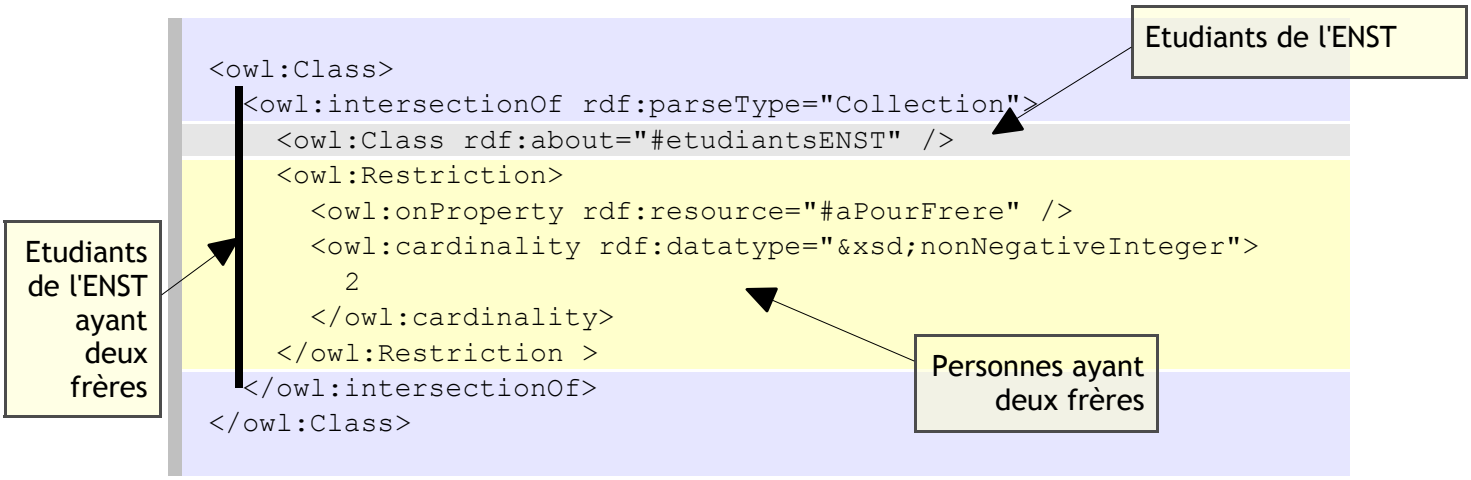
- La restriction de propriétés :

  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. Ces contraintes peuvent être de deux types : contrainte de valeur ou contrainte de

cardinalité. Une contrainte de valeur s'exerce sur la valeur d'une certaine propriété de l'individu (par exemple, pour un individu de la classe `Humain`, `sexe = Homme`), tandis qu'une contrainte de cardinalité porte sur le nombre de valeurs que peut prendre une propriété (par exemple, pour un individu de la classe `Humain`, `aPourFrere` est une propriété qui peut ne pas avoir de valeur, ou avoir plusieurs valeurs, suivant le nombre de frères de l'individu. La contrainte de cardinalité portant sur `aPourFrere` restreindra donc la classe décrite aux individus pour lesquels la propriété `aPourFrere` apparaît un certain nombre de fois).

Il existe naturellement différents opérateurs de comparaison pour établir les contraintes. Pour plus de détails, se reporter à « OWL Web Ontology Language Reference » [W3C 2004k].

- Enfin, 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 défition de la classe courante :




```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#etudiantsENST" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourFrere" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
        2
      </owl:cardinality>
    </owl:Restriction >
  </owl:intersectionOf>
</owl:Class>
```

2.5.1.2 Héritage

Il existe dans toute ontologie OWL une superclasse, nommée `Thing`, dont toutes les autres classes sont des sous-classes. Ceci nous ammène directement au 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>
```

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

2.5.2 Les instances de classe

2.5.2.1 Définition d'un individu

La définition d'un individu consiste à énoncer un « fait », encore appelé « axiome d'individu ». On peut distinguer deux types de faits :

- les faits concernant l'appartenance à une classe

  La plupart des faits concerne généralement la déclaration de l'appartenance à une classe d'in individu et les valeurs de propriété de cet individu. Un fait s'exprime de la manière suivante :

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

  Le fait écrit dans cet exemple 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.

  C'est la raison pour laquelle OWL propose un mécanisme permettant de lever cette ambiguité, à l'aide des propriétés `owl:sameAs`, `owl:diffrentFrom` et `owl:allDifferent`. L'exemple suivant permet de déclarer que les noms « `Louis_XIV` » et « `Le_Roi_Soleil` » désignent la même personne :

```
<rdf:Description rdf:about="#Louis_XIV">
  <owl:sameAs rdf:resource="#Le_Roi_Soleil" />
</rdf:Description>
```

2.5.2.2 Affecter une propriété à un individu

Une fois que l'on sait écrire une classe en OWL, la création d'une ontologie se fait par l'écriture d'instances de ces objets, et la description des relations qui lient ces instances.

2.5.3 Les propriétés

Maintenant que l'on sait écrire des classes OWL, il ne manque plus que la capacité à exprimer des faits au sujet de ces classes et de leurs instances. C'est ce que permettent de faire les propriétés OWL. OWL fait la distinction entre deux types de propriétés :

- les propriétés d'objet permettent de relier des instances à d'autres instances
- les propriétés de type de donnée 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 donnée étant une instance de la classe `owl:DatatypeProperty`. Ces deux classes sont elles-même sous-classes de la classe RDF `rdf:Property`.

La définition des caractéristiques d'une propriété se fait à l'aide d'un axiome de propriété qui, dans sa forme minimale, ne fait qu'affirmer l'existence de la propriété :

```
<owl:ObjectProperty rdf:ID="aPourParent" />
```

Cependant, il est possible de définir beaucoup d'autres caractéristiques d'une propriété dans un axiome de propriété.

2.5.3.1 Définition d'une propriété

Afin de spécifier une propriété, il existe différentes manières de restreindre la relation qu'elle symbolise. Par exemple, si on considère que l'existence d'une propriété pour un individu donné de l'ontologie constitue une fonction faisant correspondre à cet individu un autre individu ou une valeur de donnée, alors on peut préciser le domaine et l'image de la propriété. Une propriété peut également être définie comme la spécialisation d'une autre propriété.

```
<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`. 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, `anneDeNaissance` fait correspondre aux instances de la classe de `dateDeNaissance` des entiers positifs.

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.

2.5.3.2 Caractéristiques des propriétés

En plus de ce mécanisme d'héritage et de restriction du domaine et de l'image d'une propriété, il existe divers moyens d'attacher des caractéristiques aux propriétés, ce qui permet d'affiner grandement la qualité des raisonnements liés à cette propriété.

Parmi les caractéristiques de propriétés principales, on trouve la transitivité, la symétrie, la fonctionnalité, l'inverse, etc. L'ajout d'une caractéristique à une propriété de l'ontologie se fait par l'emploi de la balise OWL type :

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

Propriété non symétrique

```
<owl:ObjectProperty rdf:ID="aPourFrere">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>
```

Propriété symétrique

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

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).

2.6 Exemple d'ontologie OWL DL - description d'une population

L'ontologie OWL présentée dans ce paragraphe décrit un groupe de personnes et leurs relations de parenté, ainsi que leur lieu d'habitation. Cette ontologie ne fera pas l'objet de manipulations (édition, validation, raisonnement automatique, etc.) dans cette partie. Pour cela, il est préférable de se référer au chapitre « Outils disponibles ».

2.6.1 Présentation de la population sur laquelle porte l'ontologie

Le diagramme de classes UML de la figure « Classes composant la population à décrire » indique les classes qui composent la population dont on souhaite écrire une ontologie OWL :

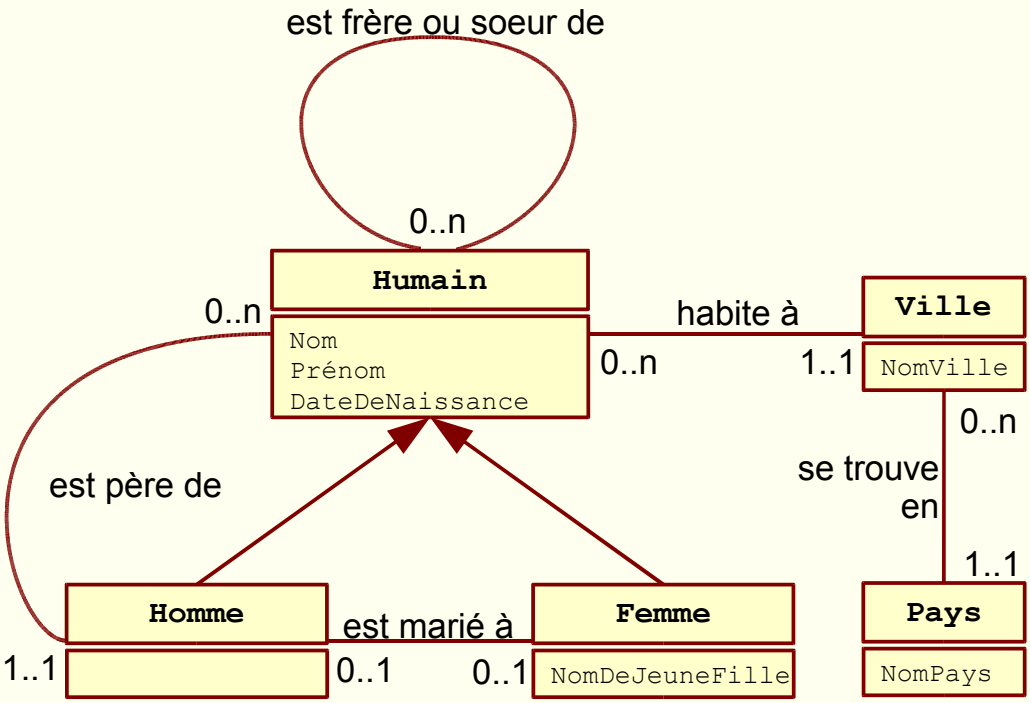


Illustration 6: Classes composant la population à décrire

La population que l'on souhaite décrire est composée d'humains, divisés en deux sous-classes `Homme` et `Femme`, et habitant dans une certaine ville. Un humain peut avoir un lien de fraternité avec un autre humain. Un homme peut être père d'un autre humain, et un homme et une femme peuvent être mariés.

Le schéma suivant indique plus précisémment les individus qui existent dans l'ontologie que nous allons écrire :

Le monde que l'on va décrire 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).

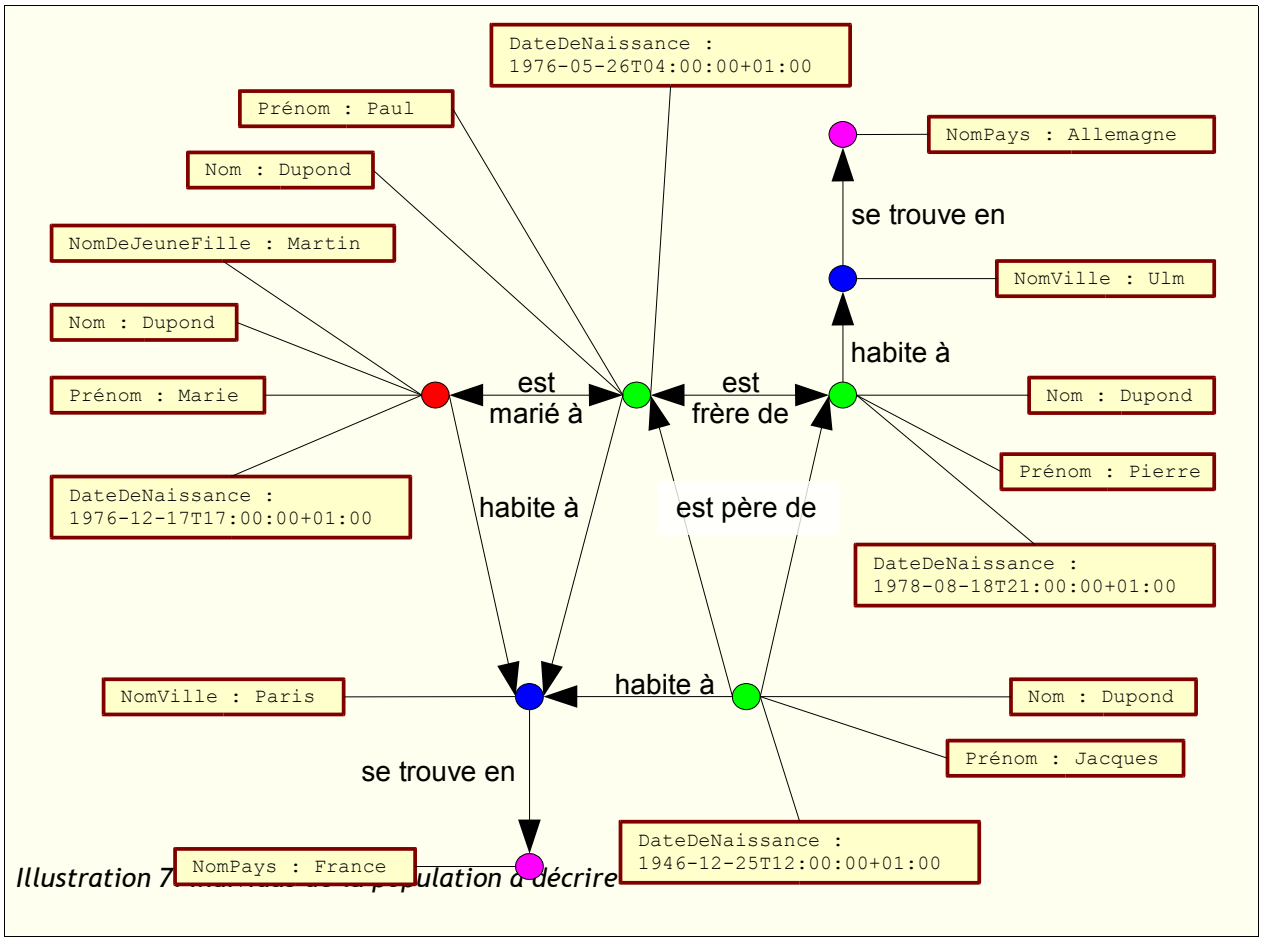


Illustration 7 : Humains de la population à décrire

2.6.2 Ecriture des classes

La première étape de l'écriture de l'ontologie OWL représentant cette population consiste à écrire les classes du monde. Il est possible de définir avec plus ou moins de précision la nature des éléments qui composent le monde que l'on décrit, en créant plus ou moins de classes. Par exemple, il serait possible d'écrire une classe « `date` » dont seraient instances toutes les dates de l'ontologie. Cela pourrait avoir un interêt dans le cadre d'une population plus vaste, pour rechercher, par exemple, les personnes nées un même jour.

On peut se poser la question du type de déclaration de classe à utiliser : par indicateur de classe ou par énumération ? L'énumération comporte, dans le cas de cette ontologie, l'inconvénient de brider l'extensibilité du monde. En conséquence, on choisit de déclarer `Humain` par le biais d'un indicateur de classe :

```
<!-- Défintion des classes -->
<owl:Class rdf:ID="Humain" />

<owl:Class rdf:ID="Homme">
  <rdfs:subClassOf rdf:resource="#Humain" />
</owl:Class>

<owl:Class rdf:ID="Femme">
  <rdfs:subClassOf rdf:resource="#Humain" />
</owl:Class>
```

```
<owl:Class rdf:ID="Ville" />
<owl:Class rdf:ID="Pays" />
```

Cette première définition des classes de l'ontologie est satisfaisante, mais elle est néanmoins perfectible. En effet, on peut ajouter des contraintes, notamment sur les classes Humain et Ville :

- un humain a toujours un père.
- Une ville se trouve forcément dans un pays.

D'autres contraintes sont imaginables, notamment les contraintes d'unicité des noms, dates de naissance, etc. Ces contraintes sont consultables dans le code complet de l'ontologie OWL, consultable en Annexe de ce document.

Voici la nouvelle écriture, plus détaillée, des deux classes « `Humain` » et « `Ville` » :

```
<owl:Class rdf:ID="Humain">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aPourPere" />
      <owl:cardinality
           rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Ville">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#seTrouveEn" />
      <owl:cardinality
           rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

2.6.3 Ecriture des propriétés

L'écriture des propriétés est l'étape qui va permettre de détailler la population que l'on veut décrire. Ecrivons tout d'abord les propriétés d'objet : habiteA, aPourPere, aUnLienDeFraternite, estMarieA et seTrouveEn :

```
  <!-- Propriétés d'objet -->
  <owl:ObjectProperty rdf:ID="habiteA">
    <rdfs:domain rdf:resource="#Humain" />
    <rdfs:range rdf:resource="#Ville" />
  </owl:ObjectProperty>
```

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

  <owl:ObjectProperty rdf:ID="aUnLienDeFraternite">
    <rdf:type rdf:resource="&owl;SymmetricProperty" />
    <rdfs:domain rdf:resource="#Humain" />
    <rdfs:range rdf:resource="#Humain" />
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="estMarieA">
    <rdf:type rdf:resource="&owl;SymmetricProperty" />
    <rdfs:domain rdf:resource="#Humain" />
    <rdfs:range rdf:resource="#Humain" />
  </owl:ObjectProperty>

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

Les propriétés `aUnLienDeFraternite` et `estMarieA` sont définies comme des propriétés symétriques : si une telle relation lie un individu A à un individu B, alors la même relation lie également l'individu B à l'individu A.

Passons maintenant aux propriétés de type de donnée. Ce sont ces propriétés qui, concrètement, permettent d'affecter des propriétés quantifiées aux instances des classes (par exemple, un nom, un prénom, une date de naissance, etc.) :

```
<!-- Propriétés de type de donnée -->
<owl:DatatypeProperty rdf:ID="nom">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="prenom">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="nomDeJeuneFille">
  <rdfs:domain rdf:resource="#Femme" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="dateDeNaissance">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="&xsd;date" />
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="nomVille">
  <rdfs:domain rdf:resource="#Ville" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="nomPays">
  <rdfs:domain rdf:resource="#Pays" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

L'image de ces propriétés est définie selon XML Schema (auquel fait référence la chaine « `&xsd;` »), ce qui permet de donner un type aux propriétés.

2.6.4 Assertion de faits caractéristisant la population

La dernière étape concerne l'assertion des faits caractérisant la population, présentés sur la figure « Individus de la population à décrire ». Il s'agit donc non seulement de l'instanciation des individus de la population, mais également de leur description par l'énonciation de leurs propriétés :

```
<!-- Humains -->
<Homme rdf:ID="unknown">
  <aPourPere rdf:resource="#unknown" />
</Homme>

<Homme rdf:ID="Pierre">
  <nom>Dupond</nom>
  <prenom>Pierre</prenom>
  <dateDeNaissance>1978-08-18</dateDeNaissance>
  <aUnLienDeFraternite rdf:resource="#Paul" />
  <aPourPere rdf:resource="#Jacques" />
  <habiteA rdf:resource="#Ulm" />
</Homme>

<Homme rdf:ID="Paul">
  <nom>Dupond</nom>
  <prenom>Paul</prenom>
  <dateDeNaissance>1976-05-26</dateDeNaissance>
  <estMarieA rdf:resource="#Marie" />
  <aPourPere rdf:resource="#Jacques" />
  <habiteA rdf:resource="#Paris" />
</Homme>

<Homme rdf:ID="Jacques">
  <nom>Dupond</nom>
  <prenom>Jacques</prenom>
  <dateDeNaissance>1946-12-25</dateDeNaissance>
  <aPourPere rdf:resource="#unknown" />
  <habiteA rdf:resource="#Paris" />
```

```
</Homme>

<Femme rdf:ID="Marie">
  <nom>Dupond</nom>
  <prenom>Marie</prenom>
  <dateDeNaissance>1976-12-17</dateDeNaissance>
  <nomDeJeuneFille>Martin</nomDeJeuneFille>
  <aPourPere rdf:resource="#unknown" />
  <habiteA rdf:resource="#Paris" />
</Femme>

<!-- Pays -->
<Pays rdf:ID="Allemagne">
  <nomPays>Allemagne</nomPays>
</Pays>
<Pays rdf:ID="France">
  <nomPays>France</nomPays>
</Pays>

<!-- Villes -->
<Ville rdf:ID="Paris">
  <nomVille>Paris</nomVille>
  <seTrouveEn rdf:resource="#France" />
</Ville>

<Ville rdf:ID="Ulm">
  <nomVille>Ulm</nomVille>
  <seTrouveEn rdf:resource="#Allemagne" />
</Ville>
```

On peut observer l'existence d'un individu « `unknown` », dont la présence est nécessitée par la contrainte de parenté qui caractérise un humain. Sans cet individu, « `Jacques` » et « `Marie` », dont le père n'apparait pas dans la population à décrire, poseraient un problème d'inconsistance vis-à-vis de la définition de la classe « `Humain` ».

Cette ontologie exemple est maintenant écrite. La partie suivante, qui traite des outils existants, utilisera cette ontologie comme référence. Le contenu complet de cette ontologie peut être consulté en Annexe de ce document.

3 Outils disponibles

Les outils présentés dans cette section ne sont pas les seuls outils disponibles, ni forcément les meilleurs outils du domaine. Ils sont cités ici uniquement à titre d'exemple de la richesse logicielle qui accompagne RDF et OWL, malgré la jeunesse relative de ces deux langages.

3.1 Editeur d'ontologies Protégé

Protégé [STA 2005] est un éditeur d'ontologies distribué en open source par l'université en informatique médicale de Stanford. Protégé n'est pas un outil spécialement dédié à OWL, mais un éditeur hautement extensible, capable de manipuler des formats très divers. Le support d'OWL, comme de nombreux autres formats, est possible dans Protégé grace à un plugin dédié.

3.1.1 Définition des classes et des propriétés

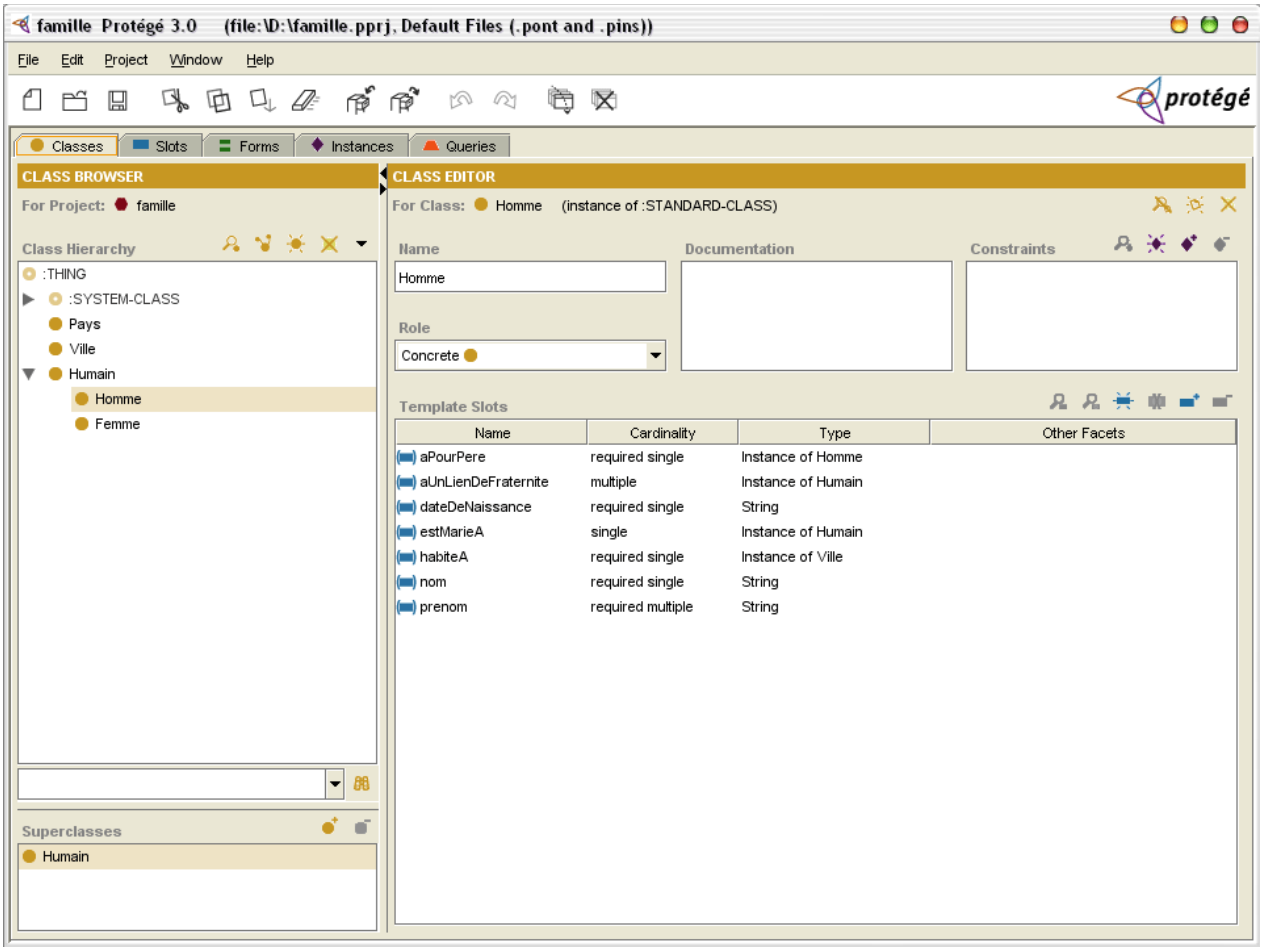


Illustration 8 : Capture de l'écran principal de Protégé, onglet « classes »

L'interface de Protégé est assez simple, l'ensemble des fonctionnalités de l'éditeur étant regroupé en cinq onglets. Le premier onglet présente les classes de l'ontologie. Il est possible de créer, modifier, supprimer une classe, et de lui attacher des propriétés. Ces propriétés

peuvent elles-même être caractérisées. La capture d'écran de la figure 8 : Capture de l'écran principal de Protégé, onglet « classes » montre l'exemple de l'ontologie développée dans la partie précédente. Cette ontologie, écrite à la main, a été importée dans Protégé à l'aide de son plugin OWL, et peut maintenant être modifiée de façon bien plus confortable.

3.1.2 Gestion des instances de classe et de leurs propriétés

L'onglet « Instances » permet de créer des instances et de leur affecter des propriétés, conformément à la définition des classes et des propriétés effectuée dans l'onglet « Classes » :

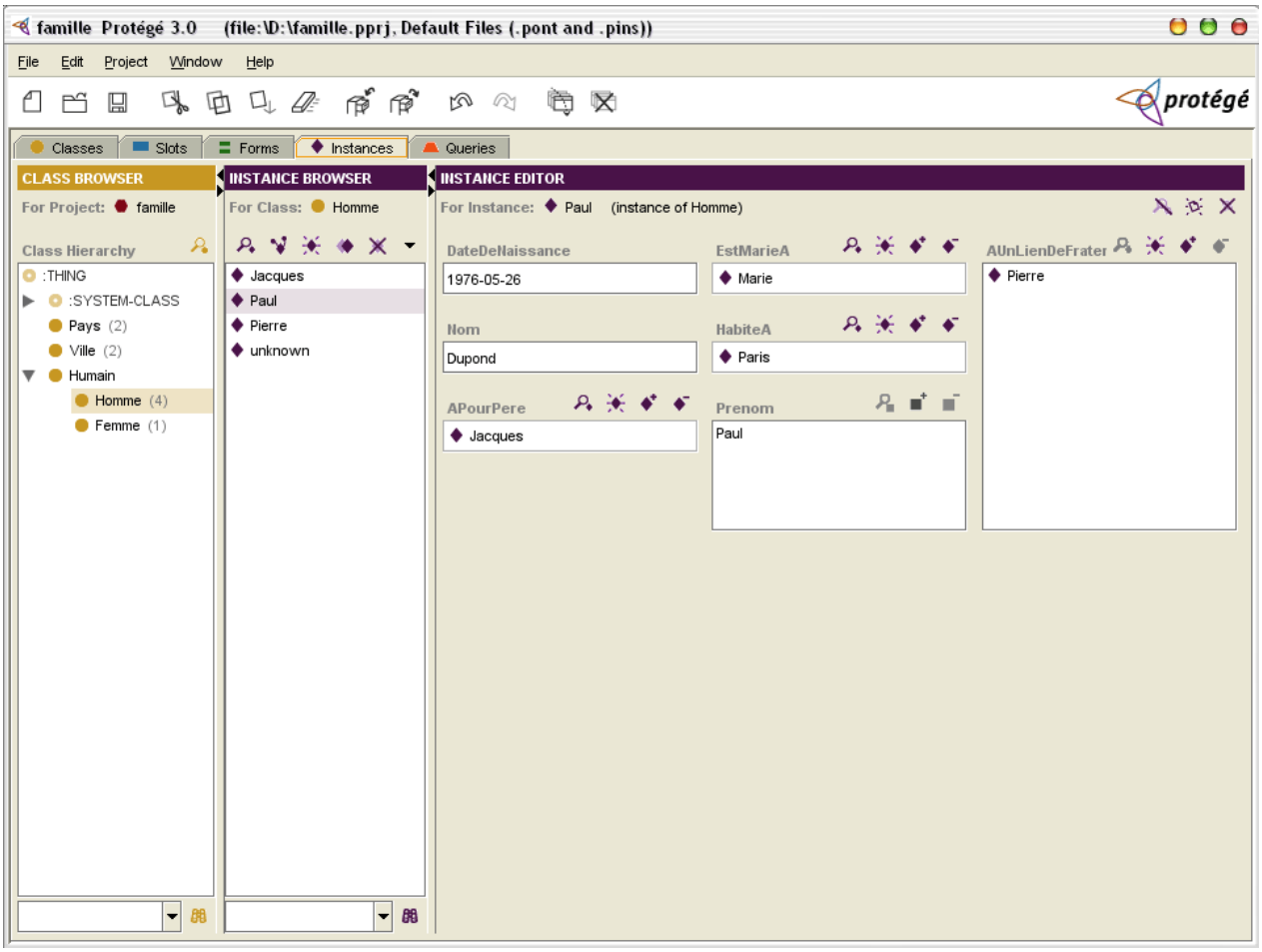


Illustration 9 : Editeur d'instances intégré à Protégé

Sur l'écran présenté, il est par exemple possible d'éditer les informations concernant l'individu « `Paul` ». Il est important de comprendre que, dans l'Instance Browser, un individu est désigné par son identifiant (son « `rdf:ID` »), et non par son prénom. En ce sens, le nommage des individus de l'ontologie écrite en exemple est sans doute un peu ambigu, bien que tout à fait correct. Pour des raisons pédagogiques, il aurait été plus indiqué de nommer ces instances « `homme1` », « `homme2` », etc.

3.1.3 Flexibilité de l'interface

Un des atouts majeurs de Protégé est la flexibilité des formulaires de saisie de l'onglet « Instances ». En effet, ce formulaire dynamique s'adapte en fonction des classes décrites dans le premier onglet du logiciel. Par ailleurs, l'onglet « Forms » permet de modifier l'organisation des formulaires d'instances, par un simple Drag and Drop :

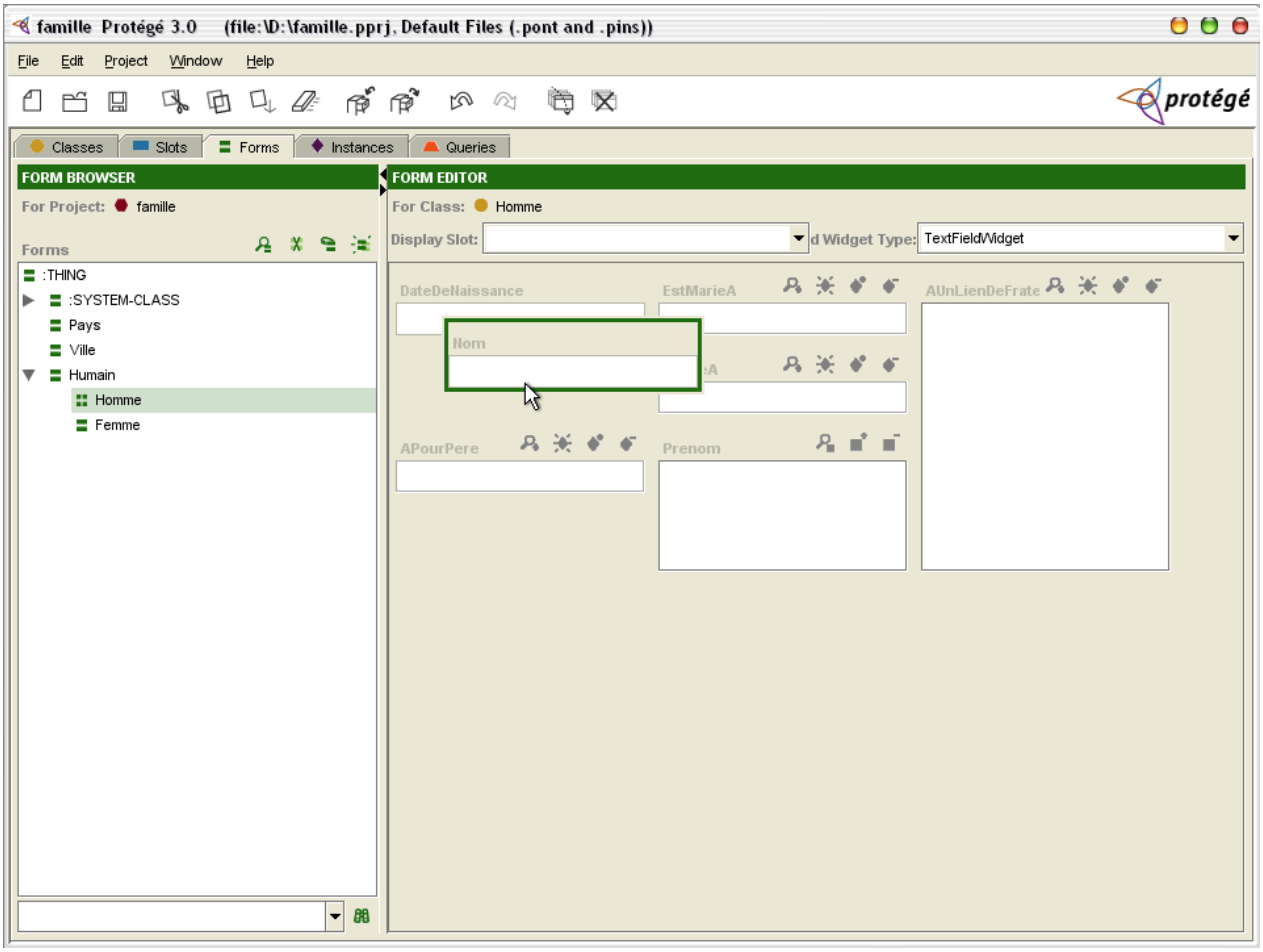


Illustration 10 : Editeur de formulaires de Protégé

3.1.4 Possibilité d'effectuer des requêtes

Enfin, une fonctionnalité intéressante de Protégé concerne la possibilité d'effectuer des requêtes sur l'ontologie en cours d'édition. La capture d'écran de la figure « Illustration 11 : Gestionnaire de requêtes de Protégé » présente une requête. En l'occurence, il s'agit de chercher toutes les instances de la classe « `Humain` » dont le « `prenom` » commence par « P ».

Le résultat de cette requête est évidemment l'ensemble `{Pierre, Paul}`. Le gestionnaire de requêtes de Protégé, bien que pratique, reste cependant limité en fonctionnalités.

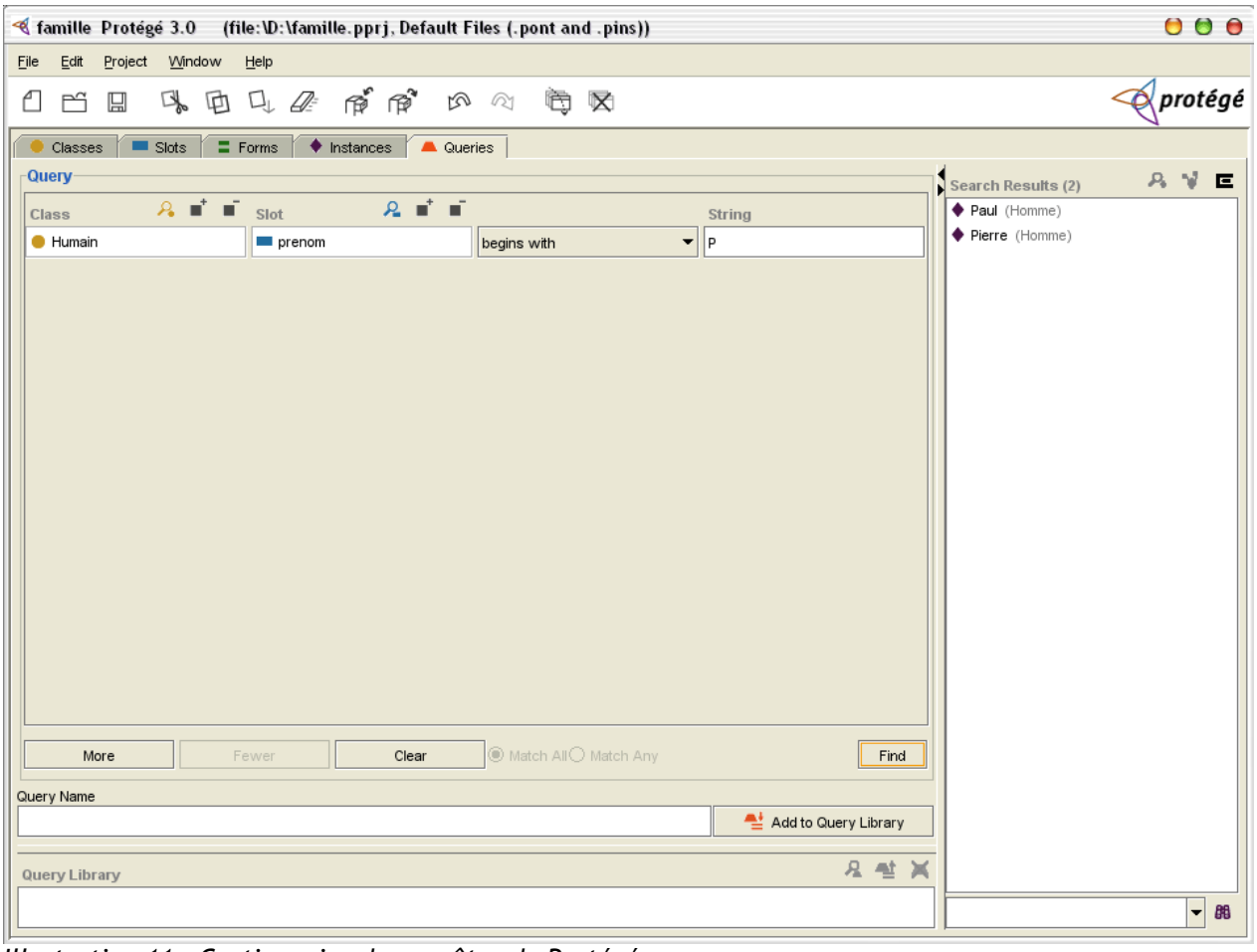


Illustration 11 : Gestionnaire de requêtes de Protégé

3.2 Framework Jena

Jena [JEN 2005] est un framework écrit en Java, dont l'objectif est de fournir un environnement facilitant le développement d'applications dédiées au web sémantique. Jena permet de manipuler des documents RDF, RDFS et OWL, et fournit en plus un moteur d'inférences permettant des raisonnements sur les ontologies.

3.3 OWL validator

Une fois qu'un document OWL est écrit, que ce soit à la main ou à l'aide d'un éditeur tel que Protégé, c'est une bonne idée de s'assurer de sa validité et de la cohérence des concepts qu'il exprime. D'une manière plus générale, le respect d'un standard ou de la définition d'un format favorise l'interopérabilité, en permettant au développeur de s'assurer de l'intégrité des données contenues dans le document qu'l vient d'écrire.

Tout comme le W3C propose un validateur HTML (http://validator.w3c.org/), il existe différents validateurs d'ontologies OWL. Certains valident uniquement la syntaxe du document, tandis que d'autres vérifient également la cohérence des informations contenues dans l'ontologie.

3.3.1 Validateur RDF du W3C

La validateur RDF du W3C (http://www.w3.org/RDF/Validator/) permet de valider des documents RDF. Il permet donc également de s'assurer qu'un document OWL respècte la

syntaxe de RDF, ce qui donne déjà une première indication de la validité d'une ontologie.

3.3.2 WonderWeb OWL Ontology Validator

Le validateur WonderWeb OWL Ontology Validator [BEC 2004] a été développé par Sean Bechhofer et Rafael Volz dans le cadre du projet WonderWeb (http://wonderweb.semanticweb.org/). L'ontologie écrite en exemple dans ce document valide évidemment :

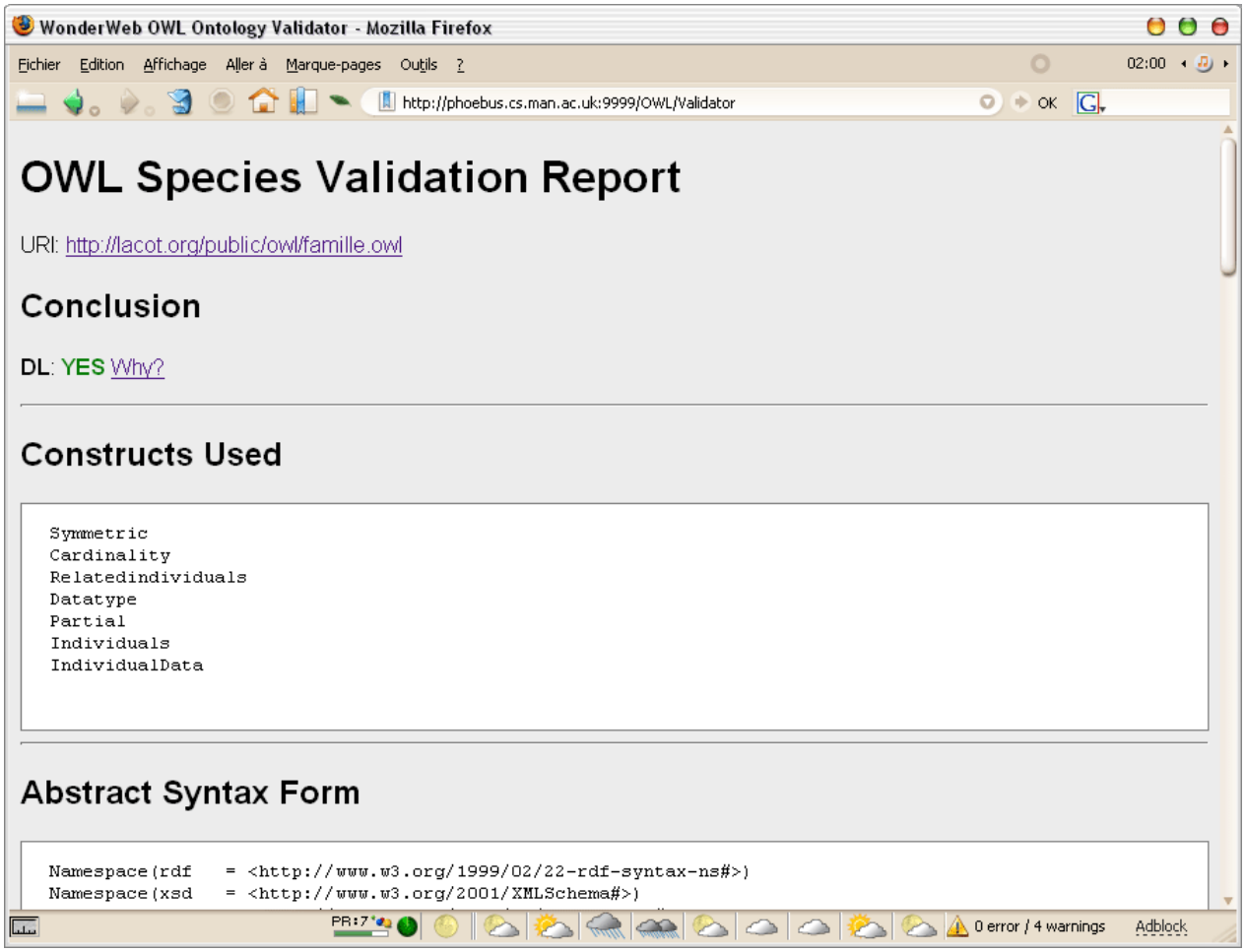


Illustration 12 : Validation de l'ontologie exemple à l'aide de WonderWeb validator

D'autres outils en relation avec le web sémantique sont proposés par WonderWeb; ils sont disponibles à l'adresse http://phoebus.cs.man.ac.uk:9999/OWL. En outre du convertisseur d'ontologies OWL (http://phoebus.cs.man.ac.uk:9999/OWL/Converter), on notera surtout l'outil de génération de documentation « OWL Ontology HTML Presentation » (http://phoebus.cs.man.ac.uk:9999/OWL/Presentation), qui permet de créer des documentations agréables à consulter à partir de toute ontologie valide.

3.3.3 Validateur vOWLidator

Le validateur OWL vOWLidator [RAG 2005] effectue la validation de documents OWL en s'appuyant sur le framework Jena. Publié en open source, vOWLidator est disponible comme exécutable (Java est naturellement requis). Une version en ligne (http://owl.bbn.com/validator/) du validateur est également disponible et, même si elle ne semble plus maintenue depuis Octobre 2003, elle fournit un moyen de valider rapidement des ontologies en ligne.

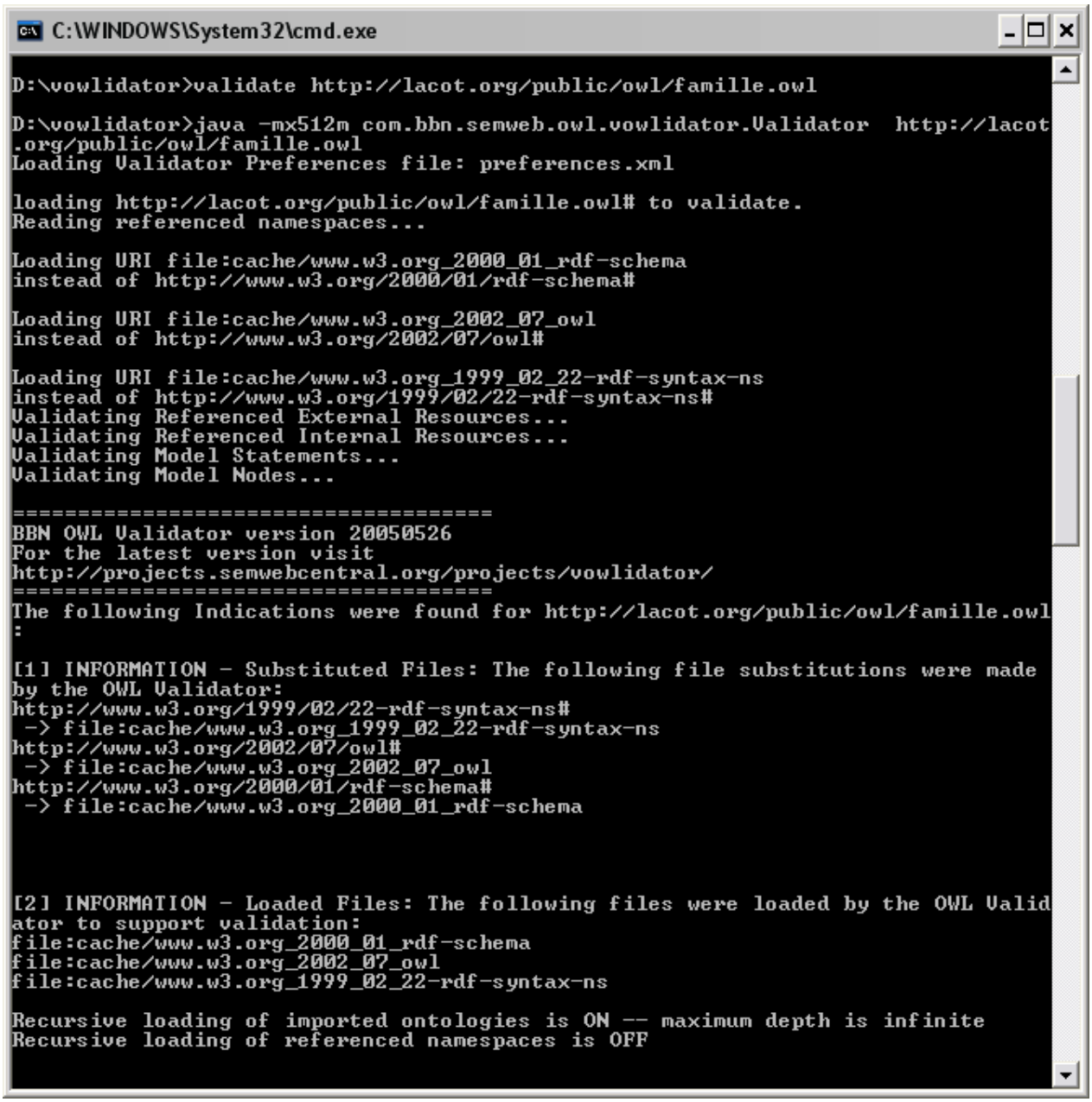


Illustration 13 : Capture d'écran du validateur OWL vOWLidator

Encore une fois, l'ontologie écrite en exemple est valide. VOWLidator retourne deux informations : la première concerne l'utilisation de fichiers de cache internes au validateur, la seconde donne des indications sur le chargement de fichiers lors de la validation.

Conclusion

Quelques quinze ans après sa création, le Web semble enfin comprendre son objectif initial : le partage rapide de savoirs précis. Pas uniquement leur présentation anarchique mais, plutôt, leur mise à disposition dans un format non ambigu, compréhensible par tous et extensible. Créés dans un objectif de partage des connaissances en réseau, les dernires langages du Web sémantique que sont RDF et OWL sont en vérité les premières pierres d'une nouvelle forme de Web : le Web sémantique, le Web de la connaissance. Facilitant l'appropriation des savoirs en les libérant de la couche présentationnelle qui les enfermait jusqu'alors, le Web sémantique contribue, en ce sens, à rejoindre les objectifs initiaux de Tim Berners Lee.

Malgré leur jeunesse relative, RDF et OWL sont déjà source d'un réel enthousiasme. Naturellement objet de l'attention des universitaires, RDF, OWL, et les autres développements du Web sémantique semblent également convaincre certaines grosses entreprises. L'exemple de Jena, soutenu par Hewlett Packard, est flagrant.

D'une manière plus générale, la richesse logicielle qui entoure les deux dernières créations du W3C ne trompe pas : le Web de demain sera sémantique.

Bibliographie

Références normatives :

[W3C 2004a] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau et John Cowan. *Extensible Markup Language (XML) 1.1*. http://www.w3.org/TR/2004/REC-xml11-20040204/ (en ligne au 16 juin 2005). Traduction française : *Le langage de balisage extensible (XML) 1.1*, http://www.yoyodesign.org/doc/w3c/xml11/.

[W3C 2004b] Frank Manola, Eric Miller, W3C et Brian McBride. *RDF Primer*. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (en ligne au 16 juin 2005).

[W3C 2004c] Graham Klyne, Jeremy J. Carroll et Brian McBride. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (en ligne au 16 juin 2005).

[W3C 2004d] Dave Beckett et Brian McBride. *RDF/XML Syntax Specification*. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ (en ligne au 16 juin 2005).

[W3C 2004e] Patrick Hayes et Brian McBride. *RDF Semantics*. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/ (en ligne au 16 juin 2005).

[W3C 2004f] Jan Grant, Dave Beckett et Brian McBride. *RDF Test Cases*. http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/ (en ligne au 16 juin 2005).

[W3C 2004g] Dan Brickley, R.V. Guha et Brian McBride. *RDF Vocabulary Description Language 1.0: RDF Schema*. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/ (en ligne au 16 juin 2005).

[W3C 2004h] Michael K. Smith, Chris Welty et Deborah L. McGuinness. *OWL Web Ontology Language - Guide*. http://www.w3.org/TR/2004/REC-owl-guide-20040210/ (en ligne au 16 juin 2005). Traduction française : *Le langage d'ontologie Web OWL - Guide*, http://www.yoyodesign.org/doc/w3c/owl-guide-20040210/.

[W3C 2004i] Jeff Heflin. *OWL Web Ontology Language - Use Cases and Requirements*. http://www.w3.org/TR/2004/REC-webont-req-20040210/ (en ligne au 16 juin 2005). Traduction française : *Les cas et conditions d'utilisation du langage d'ontologie Web OWL*, http://www.yoyodesign.org/doc/w3c/webont-req-20040210/.

[W3C 2004j] Deborah L. McGuinness et Frank van Harmelen. *OWL Web Ontology Language - Overview*. http://www.w3.org/TR/2004/REC-owl-features-20040210/ (en ligne au 16 juin 2005). Traduction française : *Vue d'ensemble du langage d'ontologie Web OWL*, http://www.yoyodesign.org/doc/w3c/owl-features-20040210/.

[W3C 2004k] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider et Lynn Andrea Stein. *OWL Web Ontology Language - Reference*. http://www.w3.org/TR/2004/REC-owl-ref-20040210/ (en ligne au 16 juin 2005). Traduction française : *La référence du langage d'ontologie Web OWL*, http://www.yoyodesign.org/doc/w3c/owl-ref-20040210/.

[W3C 2004l] Peter F. Patel-Schneider, Patrick Hayes et Ian Horrocks. *OWL Web Ontology Language - Semantics and Abstract Syntax*.http://www.w3.org/TR/2004/REC-owl-semantics-20040210/ (en ligne au 16 juin 2005). Traduction française : *La sémantique et la syntaxe abstraite du langage d'ontologie Web OWL*, http://www.yoyodesign.org/doc/w3c/owl-semantics-20040210/.

[W3C 2004m] Jeremy J. Carroll, Jos De Roo. *OWL Web Ontology Language - Test Cases*. http://www.w3.org/TR/2004/REC-owl-test-20040210/ (en ligne au 16 juin 2005). Traduction française : *Les jeux d'essais du langage d'ontologie Web OWL*, http://www.yoyodesign.org/doc/w3c/owl-test-20040210/.

Références non normatives :

[BEC 2004] Sean Bechhofer et Raphael Volz. *WonderWeb OWL Ontology Validator*. http://phoebus.cs.man.ac.uk:9999/OWL/Validator (en ligne au 16 juin 2005).

[BOR 2004] Xavier Borderie, Journal Du Net. *RDF et OWL pour donner du sens au Web*. http://developpeur.journaldunet.com/tutoriel/xml/040322-xml-web-semantique-rdf-owl1a.shtml (en ligne au 16 juin 2005).

[DAR 2005] DARPA. *The DARPA Agent Markup Language Homepage (DAML)*. http://www.daml.org/ (en ligne au 16 juin 2005).

[FOR 2005] Foundation for Research and Technology – Hellas (FORTH). *The RDF Query Language (RQL)*. http://139.91.183.30:9090/RDF/RQL/(en ligne au 16 juin 2005).

[HOR 2005] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens et Chris Wroe. *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools*. http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf (en ligne au 16 juin 2005).

[JEN 2005] *Jena - A Semantic Web Framework for Java*. http://jena.sourceforge.net/ (en ligne au 16 juin 2005).

[LOO 2004] University of Southern California's - Information Sciences Institute. *LOOM Project Home Page*. http://www.isi.edu/isd/LOOM/ (en ligne au 16 juin 2005).

[OKB 1998] *Open Knowledge Base Connectivity*. http://www.ai.sri.com/~okbc/ (en ligne au 16 juin 2005).

[OPE 2005] *OpenWeb, pour les standards du Web*. http://openweb.eu.org/ (en ligne au 16 juin 2005).

[RAG 2005] David Rager, Troy Self. *Information sur le projet vOWLlidator*. http://projects.semwebcentral.org/projects/vowlidator/ (en ligne au 16 juin 2005).

[STA 1998] Stanford Logic group. *Knowledge Interchange Format (KIF)*. http://logic.stanford.edu/kif/ (en ligne au 16 juin 2005).

[STA 2005] Université de Stanford. *The Protégé Ontology Editor and Knowledge Acquisition System*. http://protege.stanford.edu/ (en ligne au 16 juin 2005).

[WEB 2004] *Wiki francophone du web sémantique*. http://websemantique.org/ (en ligne au 16 juin 2005).

Licence

Code complet de l'ontologie exemple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY famille "http://lacot.org/public/owl/famille#" >
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>

<rdf:RDF
  xmlns         = "&famille;"
  xmlns:famille = "&famille;"
  xml:base      = "&famille;"
  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>Ontologie décrivant la famille Dupond</rdfs:comment>
    <rdfs:label>Ontologie de la famille Dupond</rdfs:label>
  </owl:Ontology>

  <!-- Défintion des classes -->
  <owl:Class rdf:ID="Humain">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#aPourPere" />
        <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#nom" />
        <owl:cardinality
             rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#prenom" />
        <owl:minCardinality
          rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
```

```
        <owl:onProperty rdf:resource="#dateDeNaissance" />
        <owl:cardinality
             rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#estMarieA"/>
        <owl:maxCardinality
          rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#habiteA" />
        <owl:cardinality
             rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="Homme">
    <rdfs:subClassOf rdf:resource="#Humain" />
  </owl:Class>

  <owl:Class rdf:ID="Femme">
    <rdfs:subClassOf rdf:resource="#Humain" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#nomDeJeuneFille" />
        <owl:maxCardinality
          rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="Ville">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#seTrouveEn" />
        <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#nomVille" />
        <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:ID="Pays">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#nomPays" />
      <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<!-- définition des propriétés -->

<!-- Propriétés d'objet -->
<owl:ObjectProperty rdf:ID="habiteA">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Ville" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="aPourPere">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Homme" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="aUnLienDeFraternite">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="estMarieA">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>

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

<!-- Propriétés de type de donnée -->
<owl:DatatypeProperty rdf:ID="nom">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="prenom">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="nomDeJeuneFille">
  <rdfs:domain rdf:resource="#Femme" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="dateDeNaissance">
  <rdfs:domain rdf:resource="#Humain" />
  <rdfs:range rdf:resource="&xsd;date" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="nomVille">
  <rdfs:domain rdf:resource="#Ville" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="nomPays">
  <rdfs:domain rdf:resource="#Pays" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<!-- Assertion de faits -->

<!-- Humains -->
<Homme rdf:ID="unknown">
  <nom>unknown</nom>
  <prenom>unknown</prenom>
  <dateDeNaissance>0001-01-01</dateDeNaissance>
  <aPourPere rdf:resource="#unknown" />
  <habiteA rdf:resource="#Paris" />
</Homme>

<Homme rdf:ID="Pierre">
  <nom>Dupond</nom>
  <prenom>Pierre</prenom>
  <dateDeNaissance>1978-08-18</dateDeNaissance>
  <aUnLienDeFraternite rdf:resource="#Paul" />
  <aPourPere rdf:resource="#Jacques" />
  <habiteA rdf:resource="#Ulm" />
</Homme>

<Homme rdf:ID="Paul">
  <nom>Dupond</nom>
  <prenom>Paul</prenom>
  <dateDeNaissance>1976-05-26</dateDeNaissance>
  <estMarieA rdf:resource="#Marie" />
  <aPourPere rdf:resource="#Jacques" />
  <habiteA rdf:resource="#Paris" />
</Homme>
```

```
  <Homme rdf:ID="Jacques">
    <nom>Dupond</nom>
    <prenom>Jacques</prenom>
    <dateDeNaissance>1946-12-25</dateDeNaissance>
    <aPourPere rdf:resource="#unknown" />
    <habiteA rdf:resource="#Paris" />
  </Homme>

  <Femme rdf:ID="Marie">
    <nom>Dupond</nom>
    <prenom>Marie</prenom>
    <dateDeNaissance>1976-12-17</dateDeNaissance>
    <nomDeJeuneFille>Martin</nomDeJeuneFille>
    <aPourPere rdf:resource="#unknown" />
    <habiteA rdf:resource="#Paris" />
  </Femme>

  <!-- Pays -->
  <Pays rdf:ID="Allemagne">
    <nomPays>Allemagne</nomPays>
  </Pays>
  <Pays rdf:ID="France">
    <nomPays>France</nomPays>
  </Pays>

  <!-- Villes -->
  <Ville rdf:ID="Paris">
    <nomVille>Paris</nomVille>
    <seTrouveEn rdf:resource="#France" />
  </Ville>

  <Ville rdf:ID="Ulm">
    <nomVille>Ulm</nomVille>
    <seTrouveEn rdf:resource="#Allemagne" />
  </Ville>
</rdf:RDF>
```