الجمهوريـــــــــــــة الجزائريـــــــة الديمقراطيـــــة الشعبيـــة
وزارة التعليـــــم العـــــالي والبحـــــث العلـــمي
جامعة وهران للعلوم والتكنولوجيا محمد بوضياف
كلية الرياضيات والاعلام الالي
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur Et de la Recherche Scientifique
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF
Faculté des Mathématiques et Informatique
***Département** : Informatique*

Polycopié de Cours
Ontologies et Web Sémantique

Ougouti Naïma Souâd ¨

Avant-propos

Ce cours s'intitule "Ontologies et Web Sémantique". Il est adressé aux étudiants en Master2, semestre 3 (S3) de l'option Systèmes d'Information et Données (SID). Il appartient à l'Unité d'Enseignement Fondamentale UEF3.1 avec un nombre de crédits égal à quatre et un coefficient matière égal à deux.

Ce cours a pour premier objectif d'initier les étudiants aux technologies du Web Sémantique les plus connues comme :

— RDF (Resource Description Framework) et ses différents formats d'échange RDF/XML, N3, Turtle, N-Triples ;

— RDFS (Resource Description Framework Schema) qui représente le schema définissant les classes et propriétés utilisées dans RDF ;

— Les ontologies qui sont des structures indispensables dans le Web Sémantique chargées d'apporter des vocabulaires échangeables, réutilisables et formels facilitant ainsi leur utilisation par les humains et les machines ;

— SPARQL (SPARQL Protocol and RDF Query Language) : Le langage de requêtes du Web Sémantique ;

— OWL (Web Ontology Language) : Le langage de représentation des connaissances contenues dans les ontologies. Il permet d'avoir la version formelle des concepts, des relations entre concepts, des axiomes et des individus.

Le second objectif est d'initier les étudiants à la construction d'ontologies en suivant les différentes étapes dédiées à cet effet, c'est à dire de la récolte et l'acquisition des connaissances du domaine étudié jusqu'à leur formalisation. Un intérêt particulier sera donné à l'éditeur Protégé qui est l'outil le plus utilisé pour cette tâche.

Une autre manière de construire des ontologies existe, elle est décrite dans le chapitre 4, il s'agit de l'extraction automatique des connaissances qui se trouvent dans un texte pour les intégrer dans une ontologie de domaine.

le dernier objectif est de montrer l'intérêt de l'Utilisation du Web Sémantique pour la description des Services Web dans le but de faciliter les tâches de découverte et de composition de ces services.

Liste des abréviations

DC :	Dublin Core
FOAF :	Freind Of A Freind
IRI :	Internationalized Resource Identifier
JSON-LD :	JavaScript Object Notation for Linked Data
LOD :	Linked Open Data
OWL :	Web Ontology Language
OWL-S :	Web Ontology Language for Services
RDF :	Resource Description Framework
RDFS :	Resource Description Framework Schema
RIF :	Rule Interchange Format
RuleML :	Rule Markup Language
SAWDL :	Semantic Annotations for WSDL
SKOS :	Simple Knowledge Organization System
SOAP :	Simple Object Access Protocol
SPARQL :	Sparql Query Language
SWRL :	Semantic Web Rule Language
UDDI :	Universal Description, Discovery and Integration
URI :	Universal Resource Identifier
URL :	Universal Resource Locator
WSDL :	Web Services Description Language
WSML :	Web Service Modeling Language
WSMO :	Web Service Modeling Ontology
XML :	eXtended Markup Language
XSD :	XML Schema

Sommaire

1 Le Web Sémantique

1.1 Introduction

Le Web sémantique due à Tim Berners-Lee [5] au sein du W3C, peut être vu comme un vaste espace d'échange entre les machines et les êtres humains dans le but d'exploiter de grandes quantités d'information à travers le web ainsi que des services variés. Il a pour vocation de décharger l'utilisateur de certaines tâches jusqu'alors effectuées exclusivement par lui, comme par exemple de chercher des informations, combiner les résultats et même raisonner sur celles-ci pour inférer d'autres informations. Il est clair que pour réaliser cette vision du web, plusieurs concepts, formalismes, langages doivent être mis à la disposition de l'utilisateur et des professionnels.

1.2 Historique

En 1994, a eu l'annonce de la création du W3C (World Wide Web Consortium) à Genève. parmi les objectifs de cette institution est d'ajouter de la sémantique au Web futur. Tim Berners Lee, l'instigateur de cette initiative, montre alors en quoi les liens hypertextes ou, plus précisément, la façon dont on met en relation les documents sur le Web est trop limitée pour permettre aux machines de relier automatiquement les données contenues sur le Web à la réalité.

le W3C nouvellement créé entame les premières réflexions sur la mise en place du Web sémantique.

En octobre 1997, un premier draft de recommandations sur le Web sémantique a été publié puis un second en 1998. Dans la même année Tim Berners lee publie un document dans lequel, il présente les différentes technologies du Web sémantique.

en 1999, il publie le livre "Weaving the Web" dans lequel il dresse un portrait du Web et les pistes pour son avenir. Dans cette même année, il énonce dans un talk sa célèbre citation : "J'ai fait un rêve pour le Web dans lequel les ordinateurs deviennent capables d'analyser toutes les données sur le Web : le contenu, les liens et les transactions entre les personnes et les ordinateurs. Un « Web sémantique », qui devrait rendre cela possible, n'est pas encore sorti, mais,quand ce sera le cas, les mécanismes d'échange au jour le jour, la bureaucratie et notre vie quotidienne seront traitées par des machines qui parlent à d'autres machines. Certains nous ont vanté depuis des lustres les « agents intelligents » et cela va enfin se concrétiser".

1.3 Différence entre Web et Web sémantique

Le Web classique est essentiellement syntaxique, cela signifie que la structure des documents est bien définie, mais que son contenu est inaccessible aux traitements machines. Seuls les humains peuvent interpréter leurs contenus.

Le web sémantique a pour objectif de lever cette difficulté. En effet le but à atteindre est de rendre le web plus intelligent, où les informations seront comprises par la machine et de permettre un traitement automatique sur les documents non structurés et cela en ajoutant des informations sur les informations qu'on appelle les métadonnées. Le rôle de ces métadonnées est d'apporter une sémantique unique et sans ambiguïté à l'information utilisée, ceci permettra d'améliorer le résultat de recherche des différents moteurs de recherche, et d'alléger la tâche de l'utilisateur pour la recherche et l'exploitation des données.

Concrètement, on peut dire que le web sémantique est une infrastructure pour permettre l'utilisation des connaissances formalisées en plus du contenu qui reste encore informel du web.

1.4 Définitions

Définition 1 : D'après Tim Berners-Lee, Directeur du World Wide Consortium(W3C) et inventeur du web puis du web sémantique, Le web sémantique n'est pas un web distinct mais bien un prolongement du Web actuel, dans lequel on attribue à l'information une signification clairement définie, ce qui permet aux humains et machines de travailler en étroite collaboration [5].

Définition 2 : Le Web sémantique (plus techniquement appelé « le Web de données ») permet aux machines de comprendre la sémantique, la signification de l'information sur le Web. Il étend le réseau des hyperliens entre des pages Web classiques par un réseau de liens entre données structurées permettant ainsi aux agents automatisés d'accéder plus intelligemment aux différentes sources de données contenues sur le Web et, de cette manière, d'effectuer des tâches (recherche, apprentissage, etc.) plus précises pour les utilisateurs [16].

Définition 3 : Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C [28].

1.5 Objectifs du Web Sémantique

On peut dire que le web sémantique met à la disposition des utilisateurs des fonctionnalités très intéressantes, qu'on peut résumer dans les points suivants :

— Recherche d'information plus élaborée,

— Intégration de sources d'information,

— Découverte, exploitation et intégration des services,
— Rendre le contenu du Web exploitable par la machine
— Permettre le raisonnement sur les données.
— Lier les données entre elles pour former un Web de données

1.6 Architecture du web sémantique

Tim Berners-Lee a proposé une architecture qui s'appuie sur une pyramide de langages dont le but est de représenter des connaissances sur le web. Chaque langage d'une couche doit être une extension du langage de la couche immédiatement en dessous, comme représenté dans la figure 1.1

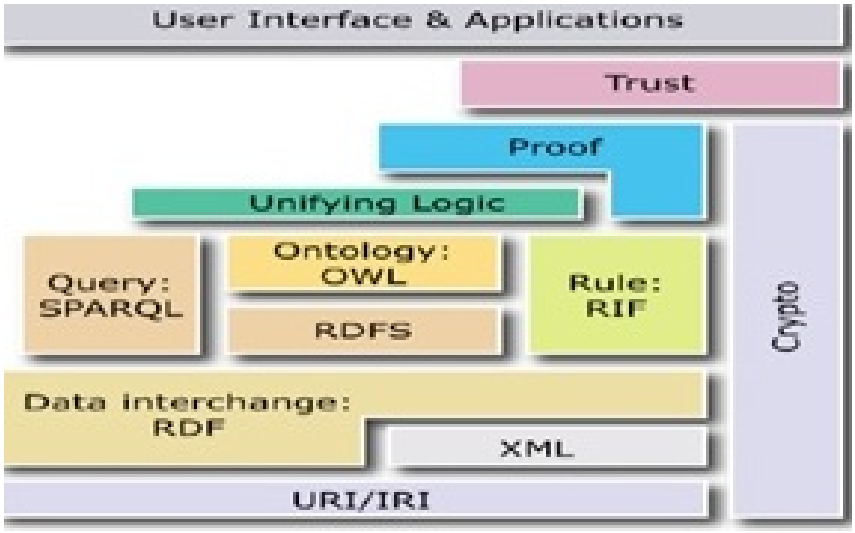


FIGURE 1.1 – Les couches du Web sémantique

Reprenons plus en détail ces différentes couches :

— **Un URI** (Uniform Resource Identifier) est une chaîne qui identifie une ressource ou un concept. Il existe plusieurs sortes d'URI, nous en citons :
  — URN Universal Resource Names : noms uniques
  — URL Universal Resource Locators : accès à des ressources : Exemple, une page Web est une ressource identifiée par une URL.

  Exemple : En langage naturel, on peut dire :

  <Charles De Gaulle> <est> <un homme> et <Charles De Gaulle> <est> <un porte-avion>.

  Dans le web sémantique, il faut écrire :

  <http ://example.org/Hommes/CharlesDeGaulle> <est> <un homme>

  <http ://example.org/Bateaux/CharlesDeGaulle> <est> <un porte-avion>

— **XML (eXtensible Markup Language)** : C'est un métalangage de représentation des données qui a été standardisé par le W3C en février 1998, il est apparu pour apporter des solutions aux limites du langage HTML. Il permet de structurer les données en utilisant des balises, qui donnent un sens aux données, elles sont fixées par l'application donc définies par l'utilisateur. Son intérêt réside dans le fait qu'il est indépendant de tout constructeur et qu'il peut s'intégrer facilement dans les diverses applications car il est indépendant de l'évolution technologique.

— **XML Schema** est un langage pour restreindre la structure des documents XML et étendre aussi XML avec des types de données. XML Fournit une surface syntaxique pour les documents structurés mais ne fournit aucune contrainte sémantique sur le sens de ces documents.

— **RDF** : Le web sémantique utilise les annotations sémantiques pour donner un sens aux pages Web ainsi qu'aux différents liens qui les lient. Le standard pour introduire ces transformations est RDF (Resource Description Framework), en effet en utilisant RDF, les ressources du Web sont annotées avec des informations sémantiques qui utilisent des modèles conceptuels (schémas) pour définir les classes et les propriétés utilisées pour les annotations sémantiques.

— **RDF Schema** est un vocabulaire pour décrire les propriétés et les classes des ressources RDF.

— **Ontology** est une structure qui ajoute plus de vocabulaire pour décrire les propriétés et les classes et les relations entre les classes. Elle offre aussi des contraintes de cardinalité, d'égalité, de typage de propriétés plus riche,de caractéristiques des propriétés et les hiérarchies des propriétés et des classes. Elle spécifie la sémantique des métadonnées fournies dans le web sémantique.

— **SPARQL** : Le W3C propose le langage SPARQL qui permet de :

  — Extraire l'information sous forme de URI, de nœuds vides ou de littéraux

  — Extraire des sous-graphes RDF

  — Construire de nouveaux graphes RDF à partir de l'information obtenue

— **La couche Logic** : donne la possibilité de construire des moteurs d'inférence logique afin de faire des liens entre les entités RDF sans que ceux-ci soient explicitement exprimés. Elle se baserait pour cela sur les règles définies par la couche Rules,

— **La couche Proof** a pour but de prouver la pertinence de l'information retournée par les couches de plus bas niveau et des déductions obtenues à partir des inférences,

— **La couche Trust** vient s'appuyer sur les deux autres et c'est elle qui permettra de contrôler la véracité d'une information en attribuant plus ou moins de confiance aux sources de données rencontrées lors de la recherche ,

— **La couche Cryptography** a pour but de s'assurer et de vérifier que les déclarations issues du Web sémantique proviennent d'une source sûre, ce qui peut être réalisé par la signature numérique des déclarations RDF and

— **User Interface** : Elle permet à l'utilisateur d'utiliser des applications en web sémantique

1.7 Le Web de données (Linked Data)

D'après le W3C, le web sémantique met en œuvre le web de données qui consiste à lier et structurer l'information pour accéder simplement à la connaissance qu'elle contient déjà. Tim Berners Lee le définit comme étant une infrastructure qui permet aux données d'être traitées directement ou indirectement par des machines pour aider leurs utilisateurs à créer de nouvelles connaissances. Les objectifs essentiels du Web de données sont résumés dans les points suivants :

— Mettre à disposition des données en utilisant des techniques standardisées qui garantissent l'interopérabilité ;

— Relier les données elles-mêmes et les rendre interprétables par les machines ;

— Permettre aux données d'être partagées et réutilisées au-delà des limites applicatives, organisationnelles ou communautaires.

Pour obtenir des données liées ou des Linked Data, quatre principes doivent être respectés :

1. Utiliser des adresses URI pour identifier les ressources ;
2. Utiliser des adresses URI HTTP pour que l'on puisse consulter ces identifications ;
3. Fournir des informations utiles sous forme de standards (RDF, SPARQL) lors d'une recherche d'adresse URI ;
4. Inclure des liens vers d'autres adresses URI qui permettent de découvrir d'autres informations.

Mais ce scénario n'est possible qu'à condition que les données soient disponibles sous licence ouverte. C'est pour cela que les Linked Data à elles seules ne suffisent pas dans l'optique du web sémantique ; elles doivent également être des Open Data, ou données ouvertes en français.

1.8 Le Web de données ouvertes (Linked Open Data)

l'Open Knowledge Foundation (2012) définit Une donnée ouverte comme étant une donnée qui peut être librement utilisée, réutilisée et redistribuée par quiconque - sujette seulement, au plus, à une exigence d'attribution et de partage à l'identique. Cette définition stipule donc qu'une donnée ouverte doit pouvoir être réutilisée à des fins commerciales également. De manière générale, pour faciliter la réutilisation des données, trois aspects doivent être pris en compte :

1. Aspect technique : un format le plus ouvert possible
2. Aspect juridique : une licence ouverte
3. Aspect économique : des redevances nulles ou limitées

L'Open Data s'inscrit dans une mouvance plus générale d'ouverture des connaissances, incluant entre autres l'Open Source (pour les logiciels), l'Open Research Data ou encore l'Open Access (pour les ressources d'information). Le concept inclut par ailleurs l'Open Government Data, qui concerne les données produites par des institutions de droit public. Un des projets

phares du LOD est Dbpedia qui a pour Objectif d'extraire les informations structurées (infobox) présentes dans Wikipedia pour les exposer en RDF.

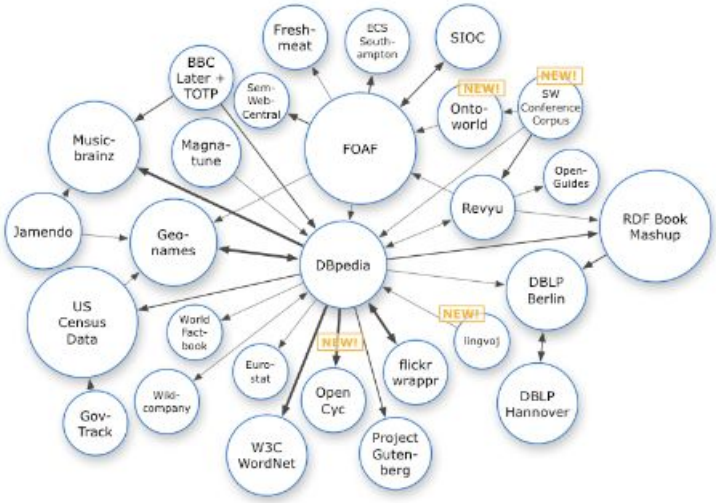


FIGURE 1.2 – Linked Open Data Cloud en 2007

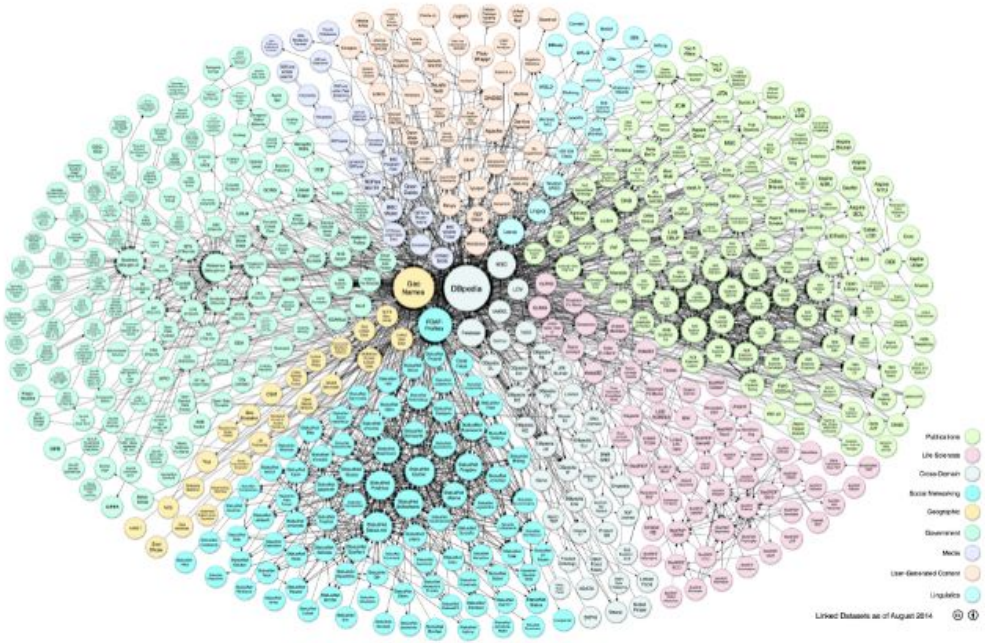


FIGURE 1.3 – Linked Open Data Cloud en 2014

1.9 Conclusion

Ce qu'il faut retenir de ce chapitre est que le Web sémantique est une extension du web qui ne remet en cause aucune des technologies existantes telles que HTTP et HTML. Il est venu pour renforcer le traitement automatique du contenu du Web et pour permettre le partage de toutes sortes de données et d'informations dans le but d'atteindre un niveau d'interopérabilité acceptable. Il est basé sur le framework RDF, la notion d'URI, RDFS, OWL et enfin le langage de requête SPARQL.

Avec l'apparition du langage SPARQL et des moteurs de gestion de données RDF qui l'accompagnent, le Web des données prend véritablement son essor. Le Linked Data qui est un ensemble de bases RDF dans des domaines variés a vu le jour et suit jusqu'à nos jours une forte croissance grâce à de nombreuses communautés actives dont le travail est l'alimentation et la mise à jour de bases dans différents domaines d'intérêts. Il devient ainsi la partie la plus visible du Web de données, contenant plusieurs milliards de triplets et est utilisé par les plus grandes et les plus puissantes entreprises et institutions telles que Google, Facebook, AMAZON et la NASA.

2 Les Langages du Web Sémantique

2.1 Introduction

Après avoir donné une idée générale sur les éléments de l'architecture du Web sémantique dans le chapitre précédent, nous allons reprendre plus en détail les constituants des langages RDF, RDF Schéma et SPARQL. L'objectif étant de pouvoir constituer une base de triplets RDF et de l'interroger par la suite.

2.2 RDF (Resource Description Framework)

RDF est un modèle de données pour décrire des ressources sur le web. Une ressource peut être n'importe quelle entité à décrire présente sur le Web comme une image, une vidéo, une page web, un lieu ou une personne. Le but de cette description est de permettre que les informations soient utilisées par d'autres applications.

RDF est l'un des piliers du Web sémantique, il s'appuie donc sur les URIs pour identifier d'une façon unique les ressources du Web et représenter les relations entre ces ressources.

La base de description dans RDF est le Triplet : Il s'agit d'une déclaration ou d'un énoncé (Statement) sous la forme (Sujet, Prédicat, Objet) ou le sujet est une ressource, le prédicat représente la propriété de cette ressource et l'objet est la valeur de cette propriété. Énoncer un triplet RDF, c'est dire qu'il y a une relation (prédicat) entre la ressource (le Sujet) et l'objet.

FIGURE 2.1 – Un triplet RDF

Un ensemble de tels triplets forme un graphe RDF. Le sujet d'un triplet peut être un URI ou un nœud vide, c'est-à-dire un nœud qui désigne une ressource sans la nommer, le prédicat est toujours un URI, par contre l'objet peut être un URI, un noeud vide ou un littéral. Ce graphe se compose de deux types de nœuds :

— Une entité référée par un URI et représentée par une ellipse ;

— Un littéral (chaine de caractère, entier, une date. . .) représenté par un rectangle.

Il présente les caractéristiques suivantes :

- Un multi-graphe, c'est-à-dire un graphe qui peut contenir plusieurs arcs et même des boucles entre deux mêmes sommets ;
- Un graphe orienté : chaque arc est orienté, allant du sommet représentant le sujet au sommet de l'objet ;
- Un graphe étiqueté : A chaque nœud et à chaque arc est associé une étiquette qui peut être soit un URI, soit un littéral.

Exemple : Supposons que nous voulions décrire ce support de cours, nous dirons qu'il a pour auteur Souâd Ougouti qui travaille à l'Université des Sciences et de la Technologie D'Oran-USTO Mohammed Boudiaf. Supposons aussi que les URIs suivants existent :

- Le support de cours est identifié par un numéro d'identification attribué par l'université, son URI est donc : http ://www.supports-cours.usto.dz/Semweb2020YL198
- Souad Ougouti est identifiée par L'URI suivant : http ://www.usto.dz/Profs#SouadOugouti et a une page personnelle dont l'URL est : http ://www.professeurs.usto.dz/souâd.ougouti
- L'université est identifiée par son URL : http ://www.usto.dz
- Pour représenter la propriété a-pour-auteur, le Dublin Core définit cette relation par l'URI suivant : http ://purl.org/dc/elements/1.1/creator, qu'on peut abréger par dc :creator.
- Les autres prédicats sont tous définis dans un document local "monvocabulaire" dont l'URI est http ://www.usto.dz/monvocabulaire, il s'agit de :
  - http ://www.usto.dz/monvocabulaire#a-page-personnelle,
  - http ://www.usto.dz/monvocabulaire#a-nom,
  - http ://www.usto.dz/monvocabulaire#travaille-à

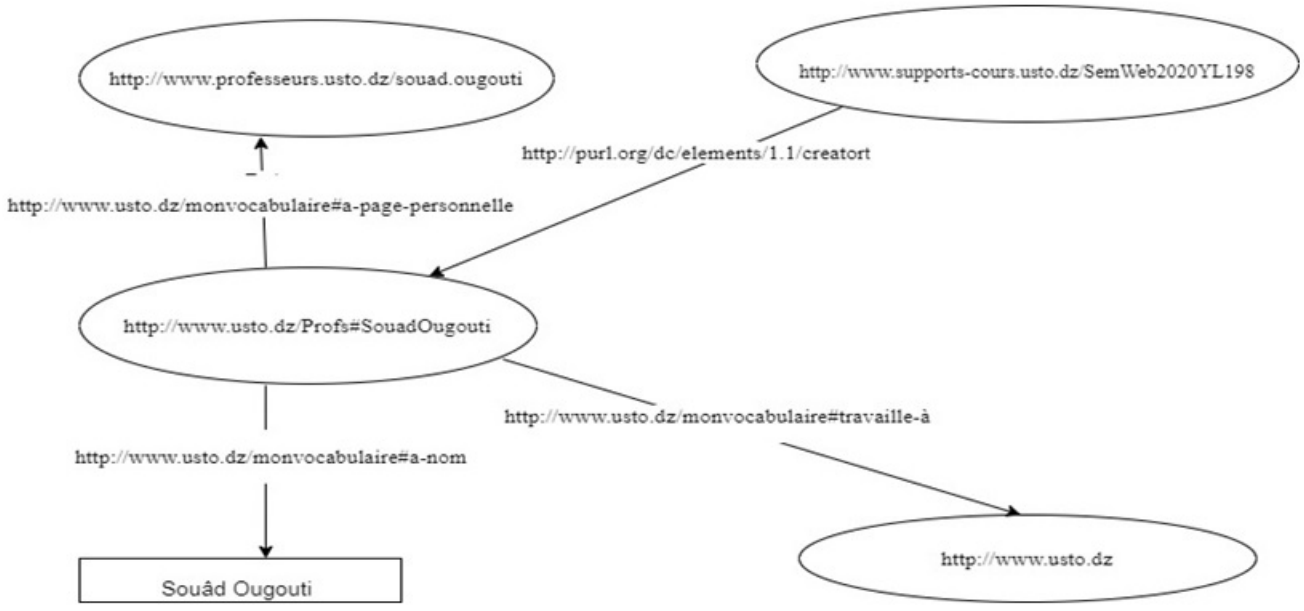


FIGURE 2.2 – Exemple de graphe RDF

Il est possible d'alléger la représentation en remplaçant les mêmes préfixes par des alias (namespaces ou espaces de nommage), par exemple :

Le préfixe http ://www.usto.dz/monvocabulaire# sera remplacé par "voclocal" et le préfixe http ://www.usto.dz/Profs# par "profs". Le schéma de la figure 2.2 deviendra :

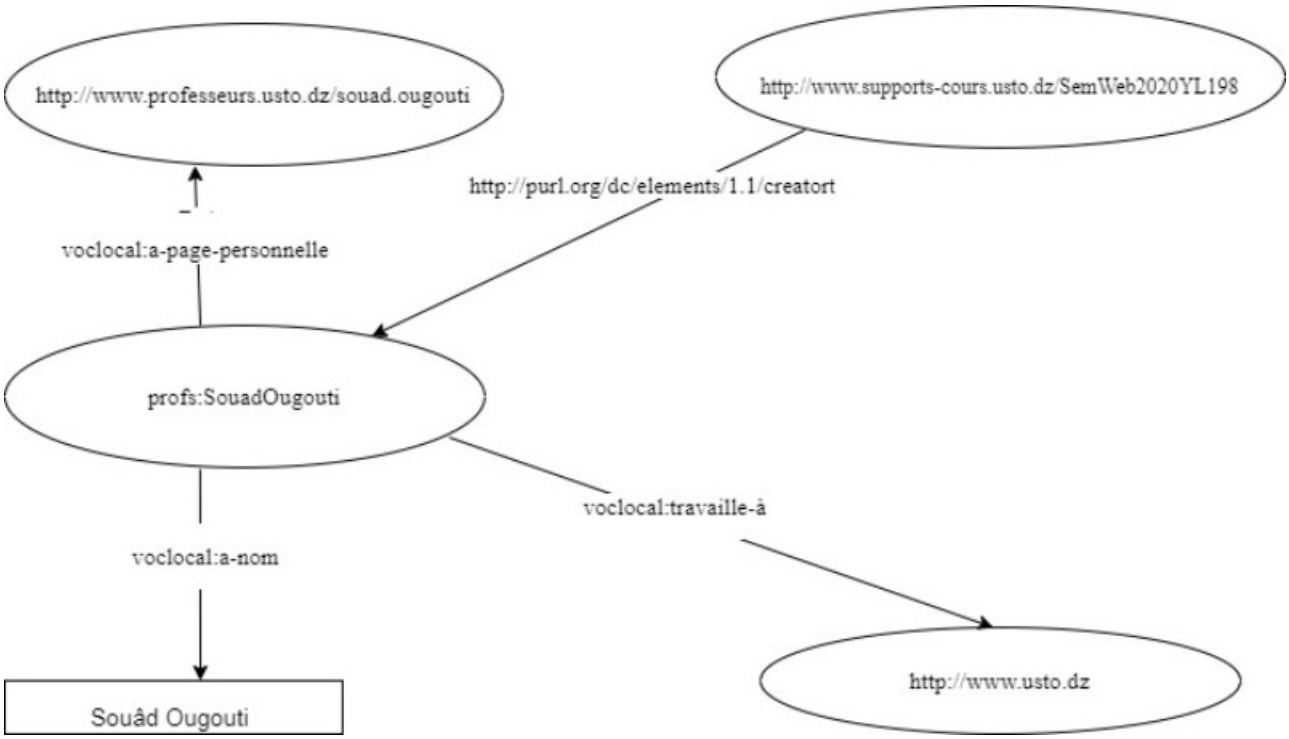


FIGURE 2.3 – Utilisation des préfixes

Dans la suite de ce support, nous utiliserons les espaces de noms suivants :

Préfixe	URI de l'espace de nommage
rdf	http ://www.w3.org/1999/02/22-rdf-syntax-ns
rdfs	http ://www.w3.org/2000/01/rdf-schema
foaf	http ://xmlns.com/foaf/0.1/
dc	http ://purl.org/dc/elements/1.1/
xsd	http ://www.w3.org/2001/XMLSchema
voclocal	http ://www.usto.dz/monvocabulaire

TABLE 2.1 – Espaces de noms

2.2.1 Syntaxe de RDF

Il y a plusieurs syntaxes possibles pour représenter une description RDF. La plus connue est la syntaxe RDF/XML qui utilise le méta-langage XML, d'autres existent comme N-Triples, Turtle et N3.

2.2.1.1 Syntaxe RDF/XML

Les documents RDF peuvent être écrits au format XML. Ce format est appelé RDF/XML, il est conçu pour être lu par les machines et bénéficie de tous les outils du monde XML. Un document RDF est un arbre XML dont l'élément racine a pour nom rdf :RDF. La description d'un triplet commence par l'élément <rdf :Description> avec l'attribut rdf :ID (avec un identificateur simple) ou rdf :about (avec un URI relatif ou absolu).

Exemple1 :

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:monDom="http://www.site.fr/mon-rdf-ns#">
        <rdf:Description rdf:about="http://www.site.fr/nomproprietaire.htm">
                <monDom:proprietaire> nompropriétaire </monDom:proprietaire>
        </rdf:Description>
</rdf:RDF>
```

Exemple2 :

Représentons en RDF/XML l'exemple de la figure 2.3

Représentation 1 :

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:profs="http://www.usto.dz/Profs#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
        xmlns:voclocal="http://www.usto.dz/monvocabulaire#">

<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator>
<rdf:Description rdf:about=" http://www.usto.dz/Profs#SouadOugouti">
        < Voclocal:a-nom> Souâd Ougouti</ Voclocal:a-nom>
        < Voclocal:travaille-à rdf:resource="http://www.usto.dz"/>
        < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
</dc:creator>
</rdf:Description>
</rdf:RDF>
```

Représentation 2 :

```
<?xml version="1.0"?>
<rdf:RDF

        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
        xmlns:voclocal="http://www.usto.dz/monvocabulaire#">
<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator  rdf:resource=" http://www.usto.dz/Profs#SouadOugouti/>
</rdf:Description>
<rdf:Description rdf:about=" http://www.usto.dz/Profs#SouadOugouti">
        < Voclocal:a-nom> Souâd Ougouti</ Voclocal:a-nom>
        < Voclocal:travaille-à rdf:resource="http://www.usto.dz"/>
         < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
</rdf:RDF>
```

Représentation 3 (Utilisation des entités) :

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
<!ENTITY profs  "http://www.usto.dz/Profs#">]>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator>
<rdf:Description rdf:about=" &profs;SouadOugouti">
        < Voclocal:a-nom> Souâd Ougouti</ Voclocal:a-nom>
        < Voclocal:travaille-à rdf:resource="http://www.usto.dz"/>
         < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
</dc:creator>
</rdf:Description>
</rdf:RDF>
```

Dans la syntaxe RDF/XML, un nœud vide est représenté tout simplement par une balise rdf :Description ne contenant aucun attribut rdf :about.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:profs="http://www.usto.dz/Profs#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
        xmlns:voclocal="http://www.usto.dz/monvocabulaire#">

<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator>
<rdf:Description >
        < Voclocal:a-nom> Souâd Ougouti</ Voclocal:a-nom>
        < Voclocal:travaille-à rdf:resource="http://www.usto.dz"/>
         < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
</dc:creator>
</rdf:Description>
</rdf:RDF>
```

Supposons maintenant que la propriété a-page-personnelle est fournie dans une description séparée. Il faudrait alors utiliser un identificateur pour le nœud vide :

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:profs="http://www.usto.dz/Profs#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
        xmlns:voclocal="http://www.usto.dz/monvocabulaire#">

<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator>
<rdf:Description  rdf:nodeID="pers25">
        < Voclocal:a-nom> Souâd Ougouti</ Voclocal:a-nom>
        < Voclocal:travaille-à rdf:resource="http://www.usto.dz"/>
         < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
<rdf:Description  rdf:nodeID="pers25">
        < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
</dc:creator>
</rdf:Description>
</rdf:RDF>
```

2.2.1.2 Syntaxe N-Triples

Ce format sérialise un graphe RDF à raison d'un triplet par ligne terminé par un point "." . Les URIs sont mis entre crochets < > et il ne permet pas les abréviations.

Dans le cas où le sujet ou l'objet est un nœud vide, on utilise la forme _ :id, où id est tout simplement un identificateur unique pour ce nœud vide.

Ce format est très simple par contre il est gourmand en taille en raison de l'énorme place que prennent les espaces de noms dans les URIs.

Exemple : Soit à représenter le même exemple précédent en notation N-Triples :

```
<http://www.supports-cours.usto.dz/SemWeb2020YL198> <http://purl.org/dc/elements/1.1/creator>
<http://www.usto.dz/Profs#SouadOugouti>.

<http://www.usto.dz/Profs#SouadOugouti> <http://www.usto.dz/monvocabulaire#a-nom> "Souâd Ougouti".

<http://www.usto.dz/Profs#SouadOugouti> <http://www.usto.dz/monvocabulaire#a-page-personnelle>
<http://www.professeurs.usto.dz/souad.ougouti>.

<http://www.usto.dz/Profs#SouadOugouti> <http://www.usto.dz/monvocabulaire#travaille-à>
<http://www.usto.dz>.
```

Si le nœud dénoté par l'URI http ://www.usto.dz/Profs#SouadOugouti était un nœud vide, on aurait la représentation suivante (à noter que l'identificateur du nœud vide est totalement arbitraire) :

```
_:pers25 <http://www.usto.dz/monvocabulaire#a-nom> "Souâd Ougouti".

_:pers25 <http://www.usto.dz/monvocabulaire#a-page-personnelle> <http://www.professeurs.usto.dz/souad.ougouti>.

_:pers25 <http://www.usto.dz/monvocabulaire#travaille-à> <http://www.usto.dz> .
```

2.2.1.3 La syntaxe Turtle

La syntaxe Turtle (Terse RDF Triple Language) permet de représenter un graphe RDF d'une manière similaire à N-Triples mais moins encombrante car il permet d'utiliser les préfixes. Les URIs sont toujours encadrés par les crochets < > par contre si on utilise les préfixes, on omet les crochets.

Dans ce format, on peut ne pas répéter le sujet ou le sujet et le prédicat s'ils sont communs à plusieurs triplets, il suffit de séparer les triplets par des points virgule " ;" , une fois la déclaration de l'ensemble de ces triplets terminée on utilise un point "." . Par contre, si nous sommes dans le cas de déclarations multiples ayant le même sujet et le même prédicat mais des objets différents, on peut séparer les objets par des virgules "," . Pour déclarer les types de ressources, Turtle offre l'article anglais "a" en remplacement de "rdf :type" .

Voici une manière de représenter l'exemple de la figure 2.3 avec la syntaxe Turtle/N3 :

Exemple :

```
@prefix  rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
@prefix  xsd="http://www.w3.org/2001/XMLSchema#"
@prefix  rdfs=http://www.w3.org/2000/01/rdf-schema#
@prefix  voclocal="http://www.usto.dz/monvocabulaire#">

<http://www.supports-cours.usto.dz/SemWeb2020YL198> dc:creator   <http://www.usto.dz/Profs#SouadOugouti> .

<http://www.usto.dz/Profs#SouadOugouti>
                    Voclocal:a-nom  "Souâd Ougouti" ;
                    Voclocal:travaille-à  <http://www.usto.dz>;
                    Voclocal:a-page-personnelle  <http://www.professeurs.usto.dz/souad.ougouti>.
```

Un nœud vide est représenté par les crochets []. Ainsi, si dans notre exemple l'URI de la personne n'était pas connue, nous aurions la représentation suivante :

```
[ ]     Voclocal:a-nom  "Souâd Ougouti" ;
        Voclocal:travaille-à  <http://www.usto.dz>;
        Voclocal:a-page-personnelle  <http://www.professeurs.usto.dz/souad.ougouti>.
```

2.2.2 Identification des types de ressources

On peut préciser les types des ressources décrites, c'est-à-dire à quelle classe appartient la ressource, ceci se fait avec l'élément **rdf :type**. **Exemple** :

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:profs="http://www.usto.dz/Profs#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
        xmlns:voclocal="http://www.usto.dz/monvocabulaire#">
<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<rdf:type  rdf:resource="http://www.usto.dz/monvocabulaire#supportscours"/>
<dc:creator>
<rdf:Description  rdf:about=" http://www.usto.dz/Profs#SouadOugouti>
<rdf:type  rdf:resource="http://www.usto.dz/monvocabulaire#MCB"/>
        < Voclocal:a-nom> Souâd Ougouti</ Voclocal:a-nom>
        < Voclocal:travaille-à rdf:resource="http://www.usto.dz"/>
         < Voclocal:a-page-personnelle rdf:resource=" http://www.professeurs.usto.dz/souad.ougouti"/>
</rdf:Description>
</dc:creator>
</rdf:Description>
</rdf:RDF>
```

2.2.3 Les Conteneurs

Un conteneur est une ressource qui contient d'autres ressources. RDF propose trois classes de conteneur : rdf :Bag, rdf :Seq et rdf :Alt.

— Rdf :Bag : groupe non ordonné de ressources ou de littéraux,

— Rdf :seq : groupe ordonné de ressources ou de littéraux,

— Rdf :Alt : groupe de ressources ou de littéraux qui sont des alternatives. Une seule des valeurs doit être sélectionnée.

Exemple : Supposons que le support de cours soit écrit par plusieurs auteurs dans ce cas, nous obtiendrons la syntaxe suivante :

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#">
<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator>
<rdf:Bag>
        <rdf:li rdf:resource="http://www.usto.dz/Profs#prenomnom1"/>
        <rdf:li rdf:resource="http://www.usto.dz/Profs#prenomnom2"/>
        <rdf:li rdf:resource="http://www.usto.dz/Profs#prenomnom3"/>
</rdf:Bag>
</dc:creator>
</rdf:Description>
</rdf:RDF>
```

Ou:

```
<rdf:Bag>
        <rdf:_1 rdf:resource="http://www.usto.dz/Profs#prenomnom1"/>
        <rdf:_2 rdf:resource="http://www.usto.dz/Profs#prenomnom2"/>
        <rdf:_3 rdf:resource="http://www.usto.dz/Profs#prenomnom3"/>
</rdf:Bag>
```

2.2.4 Collections fermées

Les collections sont des listes fermées de ressources et/ou de littéraux. Le principe de la collection est celui d'une liste chainée. La liste des membres d'une collection débute avec une ressource de type rdf :List qui porte une propriété rdf :first pointant vers le premier membre et une propriété rdf :rest pointant récursivement vers la liste des membres suivants ou sur rdf :nil si l'on a atteint le fin de la liste.

Dans RDF/XML, une collection est déclarée comme des éléments listés directement à l'intérieur d'une propriété portant l'attribut rdf :parseType="Collection".

Exemple :

Prenons le même exemple que celui utilisé pour les conteneurs, supposons que le support de cours soit co-écrit par uniquement ces trois professeurs, alors on peut utiliser les collections tel que c'est précisé dans ce qui suit avec les différentes syntaxes :

RDF/XML

```
<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator  rdf:parseType="collection">
        <rdf:Description  rdf:about="http://www.usto.dz/Profs#prenomnom1"/>
        <rdf:Description  rdf:about="http://www.usto.dz/Profs#prenomnom2"/>
        <rdf:Description  rdf:about="http://www.usto.dz/Profs#prenomnom3"/>
</dc:creator>
</rdf:Description>
Ou :
<rdf:Description rdf:about=" http://www.supports-cours.usto.dz/SemWeb2020YL198">
<dc:creator>
<rdf:Description>
        <rdf:first  rdf:resource ="http://www.usto.dz/Profs#prenomnom1"/>
        <rdf:rest>
        <rdf:Description>
                <rdf:first  rdf:resource="http://www.usto.dz/Profs#prenomnom2"/>
                <rdf:rest>
                        <rdf:Description>
                                <rdf:first  rdf:resource ="http://www.usto.dz/Profs#prenomnom3"/>
                                <rdf:rest  rdf:resource =http://www .w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                        </rdf:Description>
                </rdf:rest>
        </rdf:Description>
        </rdf:rest>
</rdf:Description>
</dc:creator>
</rdf:Description>
```

Turtle/N3

```
<http://www.supports-cours.usto.dz/SemWeb2020YL198>  dc:creator
        (<http://www.usto.dz/Profs#prenomnom1>
        <http://www.usto.dz/Profs#prenomnom2>
        <http://www.usto.dz/Profs#prenomnom3>).
```

Graphe correspondant

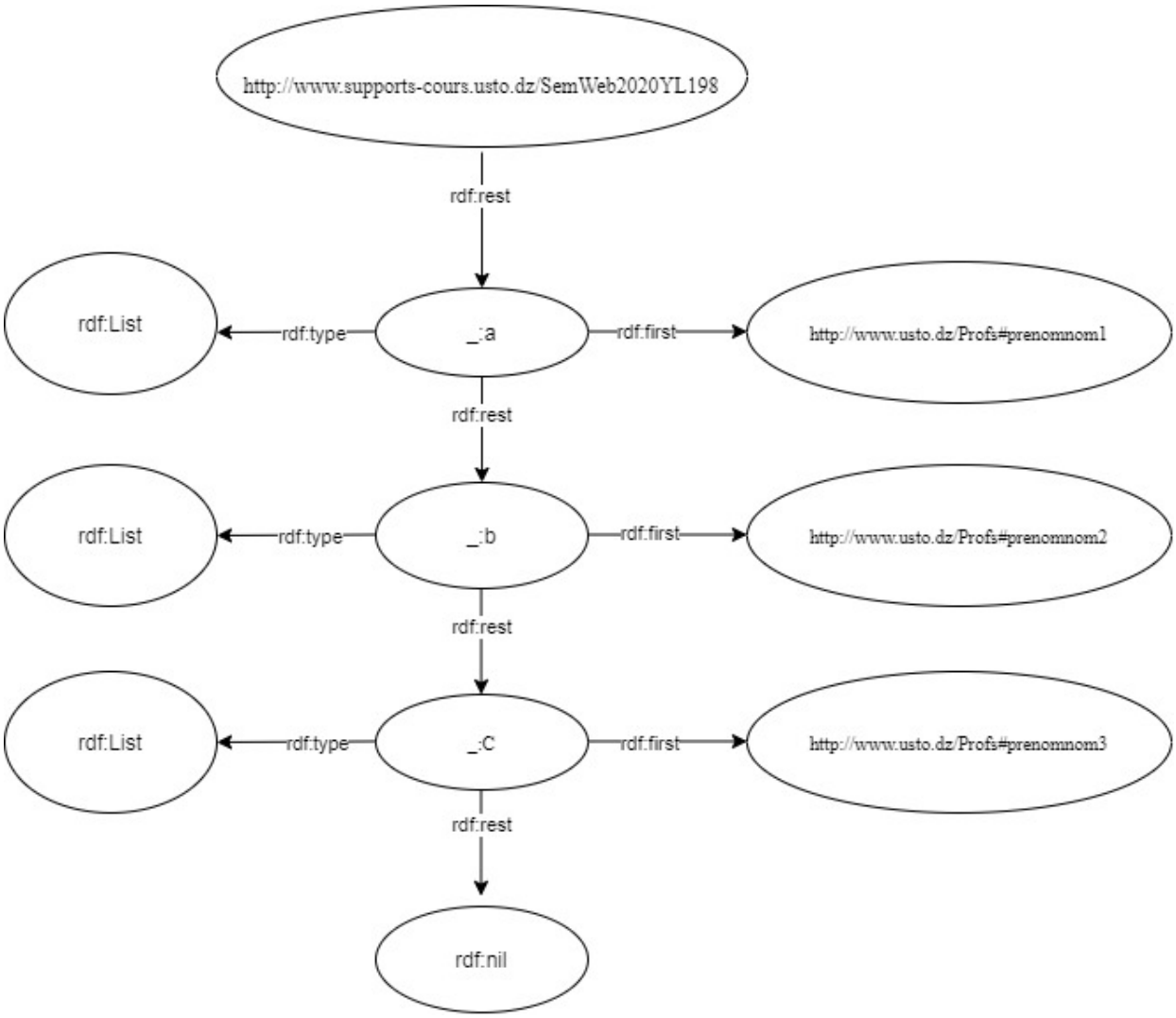


FIGURE 2.4 – Exemple de collections

2.2.5 Réification

Si on veut utiliser un triplet complet comme étant une ressource à décrire, on utilise la technique de la réification qui consiste à ajouter une ressource de type rdf :Statement pour remplacer le triplet.

On utilisera les propriétés rdf :subject, rdf :predicate et rdf :objet pour spécifier les différentes parties de ce triplet.

Graphe correspondant

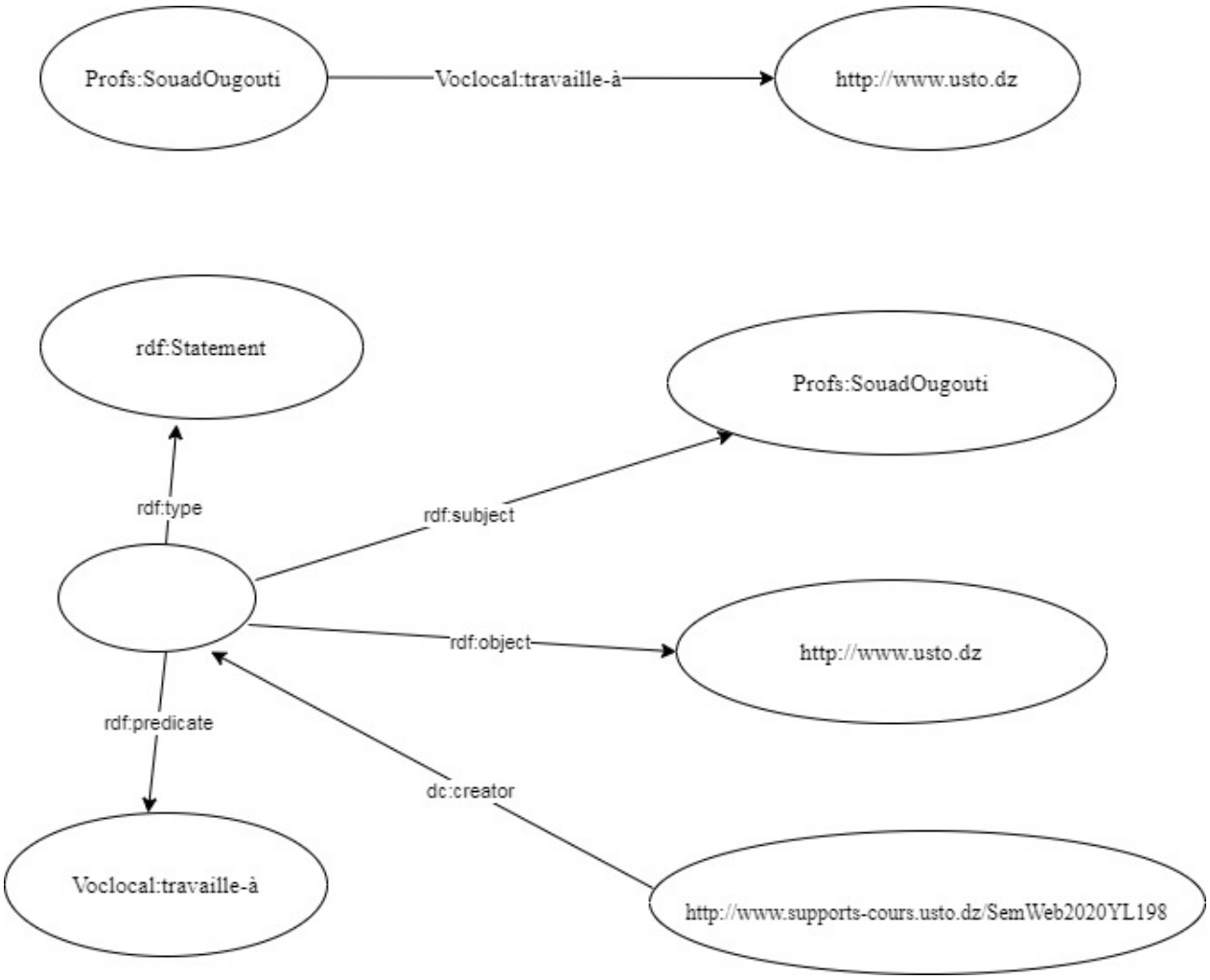


FIGURE 2.5 – Exemple de Réification

2.3 RDF Schema

RDFS ou RDF Schema permet de définir des vocabulaires RDF, principalement :

— des classes ;

— des sous-classes ;

— des prédicats avec leurs domaines et co-domaines ;

— des sous-propriétés.

RDFS est donc un premier langage de définition d'ontologie. Le prefixe pour tous les éléments du vocabulaire RDFS est http ://www.w3.org/2000/01/rdf-schema# .

2.3.1 Définition des Classes

Le premier élément important de ce vocabulaire est **rdfs :Class** qui représente justement le type de la ressource obtenue comme valeur lorsque la propriété **rdf :type** est appliquée à une ressource. RDFS permet de définir des hiérarchies de classes, en indiquant qu'une classe est sous-classe d'une autre classe, par le biais de la propriété **rdfs :subClassOf**.

Si une classe C est sous-classe de C', cela implique que si une ressource est décrite comme étant de type C, on peut déduire automatiquement qu'elle est aussi de type C'. À noter que la relation rdfs :subClassOf est transitive.

2.3.2 Définition des propriétés

En RDF, toutes les propriétés ont pour type la classe **rdf :Property**. on peut définir des hiérarchies de propriétés, en utilisant la relation **rdfs :subPropertyOf**. Mais le plus intéressant en RDFS est probablement la possibilité de spécifier le domaine ou l'image d'une propriété. On le fait en ajoutant un triplet qui indique un type de ressource qu'on peut retrouver comme sujet ou objet d'une propriété, en utilisant les relations **rdfs :domain** ou **rdfs :range**, respectivement.

2.3.3 Exemple1 de schéma RDF avec sa description RDF

Dans l'exemple suivant, trois classes sont définies : Personne, Véhicule et Voiture qui est une sous classe de Véhicule, et la propriété conducteur avec pour domaine la classe Véhicule et prend ses valeurs dans les instances de la classe Personne.

Document RDFS

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
        <rdfs:Class rdf:ID="Personne"/>
        <rdfs:Class rdf:ID="Vehicule"/>
        <rdfs:Class rdf:ID="Voiture">
                <rdfs:subclassof rdf:resource="#Vehicule"/>
        </rdfs:Class>
        <rdf:Property rdf:ID="conducteur">
                <rdfs:domain rdf:resource="#Vehicule"/>
                <rdfs:range rdf:resource="#Personne"/>
        </rdf:Property>
        </rdf:RDF>
```

Description RDF

Pour l'exemple précédent, nous pouvons avoir la description RDF suivante :

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<Voiture rdf:ID="vo001">
        <Conducteur>
                <Personne rdf:ID="p101"/>
        </Conducteur>
</Voiture>
<Voiture rdf:ID="vo002">
        <Conducteur>
                <Personne rdf:ID="p102"/>
        </Conducteur>
</Voiture>
</rdf:RDF>
```

2.3.4 Exemple2 de schéma RDF avec sa description RDF

Soit l'instruction RDF : « Java avancé est enseigné par Juliette Dibie ». Elle sera représentée graphiquement par le graphe étiqueté orienté suivant :

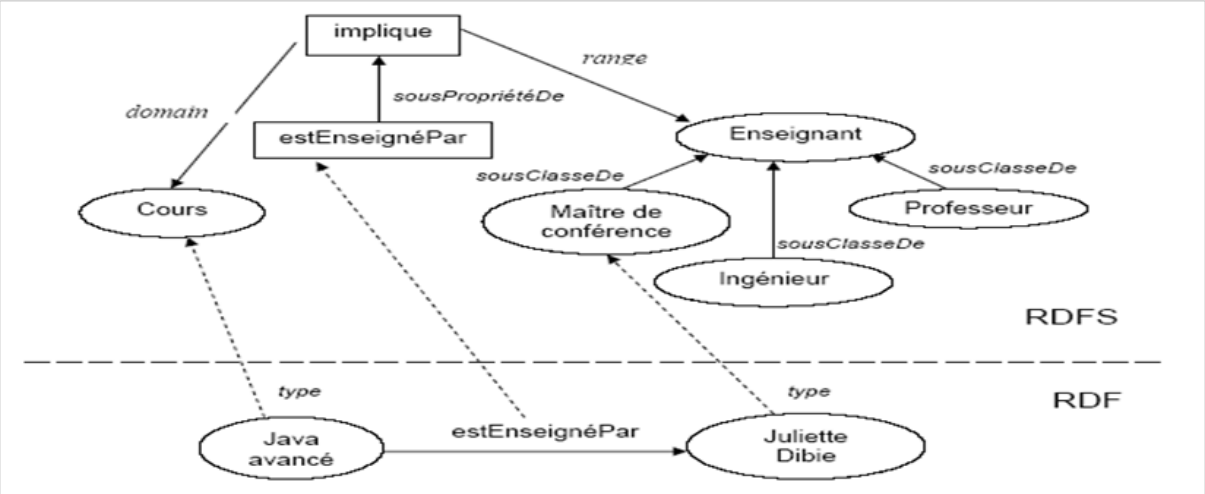


FIGURE 2.6 – Représentation de l'exemple en graphe RDF et RDFS

Elle sera traduite en RDF/XML comme suit :

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<rdfs:Class rdf:ID="MaitreDeConference">
<rdfs:subClassOf rdf:resource="#Enseignant"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Professeur">
        <rdfs:subClassOf rdf:resource="#Enseignant"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Ingenieur">
        <rdfs:subClassOf rdf:resource="#Enseignant"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Enseignant">
        <rdfs:comment> La classe de tous les enseignant </rdfs:comment>
</rdfs:Class>
<rdfs:Class rdf:ID="Cours">
        <rdfs:comment> La classe de tous les cours </rdfs:comment>
</rdfs:Class>
<rdf:Property rdf:ID="implique">
        <rdfs:domain rdf:resource="#Cours"/>
        <rdfs:range rdf:resource="#Enseignant"/>
</rdf:Property>
<rdf:Property rdf:ID="estEnseignePar">
        <rdfs:comment>
                hérite son domain et son range de la propriété implique
        </rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="#implique"/>
</rdf:Property>
</rdf:RDF>
```

2.4 SPARQL (Simple Protocol and RDF Query Language)

SPARQL est un langage de requête et un protocole pour accéder aux bases RDF. Il a été conçu par le groupe de travail du W3C RDF Data Access. SPARQL interroge des graphes RDF qui ne sont en fait qu'un un ensemble de triplets.

Il fournit un langage d'interrogation du Web Sémantique, et en cela il est à RDF ce que SQL est aux bases de données relationnelles.

2.4.1 Structure d'une requête SPARQL classique

Il y a plusieurs formes de requêtes SPARQL, mais la plus classique est celle du select .. where. Une requête SELECT se divise en 5 parties :

— définition des préfixes ; ;

— définition des résultats désirés en sortie ;

— définition des jeux de données sur lesquelles porte la requête.

— définition des conditions ;

— modificateurs de résultats (facultatif)..

```
# Déclaration des préfixes s'il y a utilisation d'IRI relatifs
PREFIX foo: <...> #Pas d'espace entre Le nom du préfixe et Les deux points
PREFIX ...
...
SELECT ... #Clause SELECT : définition des résultats
# (SPARQL 1.1) Définir le jeu de données (facultatif)
FROM <...>
FROM NAMED <...>
#Clause WHERE : conditions qui devront être respectées
WHERE {
...
}
#Modificateurs de résultats (facultatif)
GROUP BY ... #SPARQL 1.1
HAVING ... #SPARQL 1.1
ORDER BY ...
LIMIT ...
OFFSET ...
BINDINGS ... #SPARQL 1.1
```

FIGURE 2.7 – Structure d'une requête SPARQL

2.4.1.1 Les commentaires

Les commentaires commencent par un '#' en début de ligne.

Exemple : '#' Ceci est un commentaire pour donner plus de détails et d'explications.

2.4.1.2 Les préfixes

Les préfixes doivent être déclarés avant d'écrire une requête. Généralement avant la clause Select.

Exemple : Déclaration de préfixes

```
PREFIX rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc  :  <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ...
```

2.4.1.3 Abréviation de rdf :type

Dans les requêtes SPARQL le mot "a", représente l'abréviation du prédicat **rdf :type** ou de **<http ://www.w3.org/1999/02/22-rdf-syntax-ns#type>**. On peut écrire dans une requête Sparql **?x rdf :type foaf :person** ou **?x a foaf :Person**

2.4.1.4 La factorisation du Sujet et du Prédicat

le sujet des triplets est souvent répété dans une requête pour écrire plusieurs conditions portant sur lui. Ainsi, le caractère " ;" évite de retaper le sujet des triplets.

Exemple : Triplets avec répétition du sujet

```
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book1 foaf:homepage <http://example.org/book/sparql> .
```

Les mêmes triplets sans répétition du sujet et avec l'utilisation de " ;"

```
:book1 dc:title "SPARQL Tutorial" ;
       ns:price 42 ;
       foaf:homepage <http://example.org/book/sparql> .
```

Pour les mêmes raisons mais pour éviter d'écrire le même prédicat, on peut utiliser le caractère ",".

Exemple de triplets avec répétition du sujet et du prédicat

```
:book1 dc:author "Michel" .
:book1 dc:author "Paul" .
```

Le même exemple de triplets sans répétition du sujet et du prédicat

```
:book1 dc:author "Michel" , "Paul" .
```

2.4.1.5 Utilisation des Littéraux

Dans une requête en SPARQL qui utilise également des triplets, comme dans le format RDF/Turtle, les objets peuvent également être des littéraux, en ayant la même syntaxe que Turtle.

```
SELECT ...
WHERE {
... "2002-10-10T12:00:00Z"^^xsd:date .
}
```

2.4.1.6 Utilisation des Variables

Une variable dans une requête commence par le caractère " ?" et ce caractère ne fait pas partie du nom de la variable. Le nom d'une variable :

- ne doit pas commencer par un chiffre ;
- est sensible à la casse ;
- ne doit pas contenir d'espace ;
- doit être signifiant, car il sert de nom de colonne dans le résultat SPARQL 1.0.

2.4.1.7 Définir le tableau de résultats

Dans le protocole SPARQL, les résultats prennent la forme d'un tableau. C'est dans la clause SELECT que les colonnes sont définies en alignant les variables qui sont présentes dans la clause WHERE. Par exemple :

- "SELECT ?nom ?adresse" : affichera les lignes avec 2 champs : le nom et l'adresse.
- " SELECT * " : permet d'afficher dans les résultats toutes les variables présentes dans la clause WHERE.
- "SELECT DISTINCT ?pays" : permet de ne pas afficher les lignes qui ont un résultat identique.
- "SELECT ?pays (100 * ?taux AS ?pourcentage)" : permettra de faire des calculs et renommer les champs.

2.4.1.8 Modifier les résultats

La modifications des résultats se fait par l'utilisation des mots-clés ORDER BY, LIMIT et OFFSET.

ORDER BY permet d'ordonner le résultat de la requête. Il suffit de lui donner en argument la colonne selon laquelle il doit ordonner les résultats.

Exemple avec ORDER BY

```
SELECT ?Nomdefamille
WHERE
      {
            ?x  foaf:familyName  ?Nomdefamille.
       }
ORDER BY  ?Nomdefamille
```

Exemple avec ORDER BY avec plusieurs colonnes

```
SELECT ?Nomdefamille  ?Prénom
WHERE  {
       ?x       foaf:familyName ?Nomdefamille;
                foaf:givenName  ?Prénom.
 }
ORDER BY  ?Nomdefamille  ?Prénom
```

Pour afficher par ordre inverse, il suffit d'ajouter le mot-clef DESC devant le nom de la colonne concernée entre parenthèses.

Exemple avec ORDER BY avec les noms triés par ordre inverse

```
SELECT ?Nomdefamille  ?Prénom
WHERE  {
       ?x       foaf:familyName ?Nomdefamille;
                foaf:givenName  ?Prénom .
 }
ORDER BY  DESC(?Nomdefamille)  ?Prénom
```

LIMIT est utilisée pour afficher un nombre limité de réponses, dans l'exemple suivant uniquement les dix premières réponses seront affichées.

Exemple de requête avec un nombre de résultats affichés limité

```
SELECT ?Nomdefamille  ?Prénom
WHERE  {
        ?x      foaf:familyName ?Nomdefamille;
                foaf:givenName  ?Prénom .
 }
LIMIT 10
```

OFFSET permet d'afficher le résultat à partir d'une certaine position. Dans l'exemple suivant l'affichage des noms et prénoms se fera à partir de la cinquième ligne.

Exemple de requête avec OFFSET

```
SELECT ?Nomdefamille  ?Prénom
WHERE  {
        ?x      foaf:familyName ?Nomdefamille;
                foaf:givenName  ?Prénom .
 }
OFFSET 5
```

Exemple de requête avec OFFSET et LIMIT en même temps

```
SELECT ?Nomdefamille  ?Prénom
WHERE  {
        ?x      foaf:familyName ?Nomdefamille;
                foaf:givenName  ?Prénom .
 }
OFFSET 5
LIMIT 10
```

2.4.1.9 Définir les conditions

Les requêtes SPARQL contiennent généralement un ensemble de triplets ou chaque sujet, prédicat et objet peut être une variable. Chaque triplet devient alors une condition pour la requête. L'exemple ci-dessous montre une requête SPARQL qui se compose de deux parties :

— la clause SELECT qui identifie les variables à faire apparaître dans la réponse de cette requête,

— et la clause WHERE qui fournit les conditions à appliquer sur l'ensemble des triplets pour chercher une réponse.

La requête sera appliquée sur le triplet suivant :

<http ://example.org/Livre/livre1> <http ://purl.org/dc/elements/1.1/title> "SPARQL".

```
SELECT    ?titre
          WHERE
          {
          <http://example.org/Livre/Livre1> <http://purl.org/dc/elements/1.1/title>  ?titre.
          }
```

Cette requête sur les données ci-dessus a une seule solution.

titre
"SPARQL"

Le résultat d'une requête est un tableau où chaque ligne représente une solution différente. Il peut y avoir zéro, une ou plusieurs solutions à une requête.

Chaque ligne contient la liste des variables indiquées dans la clause SELECT de la requête. Pour chaque solution, il existe une ligne dans la solution et chaque ligne affichera la valeur qu'a dû prendre chaque variable de la clause SELECT pour trouver cette solution.

Soit l'ensemble de triplets suivant :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://jlow.me>  foaf:name "Johnny Lee Outlaw" .
<http://jlow.me>  foaf:mbox <mailto:jlow@jlow.me> .
<http://peter.me> foaf:name "Peter Goodguy" .
<http://peter.me> foaf:mbox <mailto:peter@peter.me> .
<http://carol.me> foaf:mbox <mailto:carol@carol.me> .
```

On désire afficher tous les noms avec leurs adresses mail, et pour cela on applique la requête SPARQL suivante :

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

Résultat de la requête :

name	**mbox**
"Johnny Lee Outlaw"	<mailto:jlow@jlow.me>
"Peter Goodguy"	mailto:peter@peter.me

2.4.1.10 Contraintes

On peut, à l'aide du mot clé FILTER, ajouter une contrainte à la requête. La clause FILTER ajoute une condition qui doit se vérifier pour valider une solution.
Les principales utilisations de la clause FILTER sont présentées dans les exemples suivants :

Soit le jeu de données RDF/Turtle suivant :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .

ex:example   dc:title     "SPARQL Title without tag lang" .
ex:book1     dc:title     "SPARQL Tutorial"@en .
ex:book1     ns:price     42 .
ex:book2     dc:title     "The Semantic Web"@en .
ex:book2     ns:price     23 .
```

regex() est une fonction qui permet d'utiliser une expression régulière pour vérifier une chaîne de caractères sans balise de langue.

Exemple de requête SPARQL avec FILTER

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

Résultat de la requête

title
"SPARQL Title without tag lang"

Pour appliquer cette contrainte à tout littéral sans se soucier de la balise de langue, on utilise la fonction str(). Ainsi, la requête SPARQL devient :

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

le résultat de la requête fait apparaitre deux résultats :

title
"SPARQL Title without tag lang"
"SPARQL Tutorial"@en

On peut également appliquer l'option "insensible à la casse" à notre expression régulière, avec le paramètre "i", la requête SPARQL devient :

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

Pour appliquer cette contrainte à tout littéral sans se soucier de la balise de langue, on utilise la fonction str(). Ainsi, la requête SPARQL devient :

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

Et le résultat de la requête fait apparaitre deux résultats :

title
"SPARQL Title without tag lang"
"SPARQL Tutorial"@en

FILTER sur un Littéral numérique

La clause FILTER peut vérifier des conditions numériques. Par exemple la requête SPARQL suivante :

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

donnera le résultat suivant :

title	price
"The Semantic Web"@en	23

2.4.1.11 UNION

L'union permet de faire l'addition ou l'union de deux graphes RDF. Par exemple, avec le graphe RDF/Turtle suivant :

```
@prefix ex: <http://example.org/book/> .
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
ex:a    dc10:title      "SPARQL Query Language Tutorial" .
ex:a    dc10:creator    "Alice" .
ex:b    dc11:title      "SPARQL Protocol Tutorial" .
ex:b    dc11:creator    "Bob" .
ex:c    dc10:title      "SPARQL" .
ex:c    dc11:title      "SPARQL (updated)" .
```

la requête suivante :

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { { ?book dc10:title ?title } UNION { ?book dc11:title ?title } }
```

donnera le résultat suivant :

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

2.4.1.12 Requête ASK

Une requête ASK est très proche d'une requête SELECT sauf qu'il n'y a que deux résultats possibles : true (vrai) ou false (faux). Elle se divise en 3 parties :

1. Définition des préfixes.
2. Définition des jeux de données sur lesquels porte la requête
3. Définition des conditions.

Une requête renvoie vrai si les conditions de la requête trouvent au moins une réponse dans les données. S'il n'y a aucune réponse possible, la requête renvoie faux.

Soit le jeu de données RDF/Turtle suivant :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://Alice.me> foaf:name "Alice" .
<http://Alice.me> foaf:homepage <http://work.example.org/alice/> .
<http://Bob.me> foaf:name "Bob" .
<http://Bob.me> foaf:mbox <mailto:bob@work.example> .
```

On applique la requête SPARQL :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK WHERE { ?x foaf:name "Alice" }
```

Résultat de la requête : **TRUE**

En revanche, avec la requête SPARQL :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK WHERE {     ?x foaf:name "Alice" ;
                foaf:mbox <mailto:alice@work.example> }
```

Résultat de la requête : **FALSE**

2.4.1.13 Requêtes DESCRIBE

La requête DESCRIBE a pour fonction de décrire une référence.

```
DESCRIBE <http://example.org/>
```

Donnera un résultat qui contiendra les triplets dont la racine sera la référence à décrire :

```
@prefix a: <http://www.w3.org/2000/10/annotation-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example.org/>
        a:annotates <http://www.w3.org/TR/rdf-sparql-query/> ;
        a:annotates <http://www.w3.org/TR/rdf-sparql/> ;
        a:annotates <http://www.w3.org/TR/xsd> ;
        dc:date "2004-12-31T19:00:00-05:00".
```

On peut également décrire plusieurs références sélectionnées en même temps et qui respectent des conditions.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
DESCRIBE ?x
WHERE { ?x foaf:name "Alice" }
```

Le résultat sera :

```
<http://Alice.me> foaf:name "Alice" .
<http://Alice.me> foaf:homepage <http://work.example.org/alice/> .
```

2.5 Conclusion

Dans ce chapitre, nous avons introduit l'essentiel à connaître sur les langages du web sémantique, à savoir, RDF , RDFS et SPARQL. Ce sont des langages standardisés par le W3C qui constituent un prérequis obligatoire pour intégrer la communauté des linked data. Plusieurs exemples ont été intégrés pour illustrer et faciliter la compréhension de ces éléments. le langage OWL n'a pas été abordé ici, nous avons jugé préférable de l'introduire dans le chapitre sur les ontologies.

3 Les Ontologies

3.1 Introduction

Les ontologies représentent un pilier essentiel et obligatoire pour la construction du web sémantique. Elles ont pour rôle de fournir les concepts nécessaires pour le marquage sémantique des données pour pouvoir être exploitées par ce dernier. Nous essaierons dans ce qui suit de mettre la lumière sur leurs fondements et principes de constitution.

3.2 Définition

A l'origine l'ontologie est un 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 qui définit les termes employés pour décrire et représenter un domaine de connaissances. Elle fournit une sémantique formelle pour l'information qui lui permet d'être exploitée par un ordinateur. Son but est d'avoir une compréhension partagée entre des personnes ou des agents logiciels du sens des termes et de leurs relations, un vocabulaire contrôlé et commun, ce qui implique une définition formelle des concepts. Les ontologies ont donc deux rôles symétriques

— Définir / fournir une sémantique formelle pour l'information permettant son exploitation par un ordinateur ;

— Définir / fournir une sémantique d'un domaine du monde réel fondée sur un consensus et permettant de lier le contenu exploitable par la machine avec sa signification pour les humains.

On distingue généralement deux entités globales au sein d'une ontologie :

— La première, à objectif terminologique, définit la nature des éléments 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.

3.3 Autres définitions

Une ontologie fournit une base solide pour la communication entre les machines mais aussi entre humains et machines en définissant le sens des objets tout d'abord à travers les symboles (mots ou expressions) qui les désignent et les caractérisent et ensuite à travers une représentation structurée ou formelle de leur rôle dans le domaine.

Une des définitions les plus célèbres et la plus utilisée est celle de Gruber en 1995 [**gruber**] : « Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée ». Une conceptualisation est la représentation d'un domaine par un ensemble de concepts. Elle est le résultat d'une analyse ontologique du domaine étudié. C'est une spécification formelle et explicite car elle représente la conceptualisation dans une forme concrète et dans un langage compréhensible par les machines où tous les concepts et les contraintes utilisées doivent être explicitement définis. Enfin, la connaissance décrite par l'ontologie doit être consensuelle et refléter un point de vue général accepté et partagé par un groupe.

Guarino [14], en 1995, affine cette définition en considérant une ontologie comme un vocabulaire partagé, plus une spécification du sens convenu de ce vocabulaire.

Charlet [6], en 1998, définie une ontologie comme une certaine vue du monde par rapport à un domaine donné. Cette vue est souvent conçue comme un ensemble de concepts (entités, attributs, processus), leurs définitions et leurs interrelations. On appelle cela une conceptualisation.

D'après Bachimont [2],en 1998, définir une ontologie pour la représentation des connaissances, c'est définir, pour un domaine et un problème donnés, la signature fonctionnelle et relationnelle d'un langage formel de représentation et la sémantique associée.

3.4 Les caractéristiques d'une ontologie

Les ontologies doivent offrir certaines caractéristiques fondamentales pour le web sémantique, nous pouvons citer les points suivants :

Définition précise des termes et de leur sens : la signification des termes doit être suffisamment précise pour que l'ontologie puisse servir de référentiel et offrir un vocabulaire partagé par les communautés. C'est la partie «terminologique» des connaissances d'un domaine : elle définit les connaissances de nature « structurelle », c'est-à-dire les concepts du domaine, leurs propriétés, et leurs relations (souvent plus aisément représentables par des frames ou des logiques de description), par opposition à la partie « déductive » qui définit les connaissances de nature inférentielle (auxquelles les règles se prêtent mieux).

Fondement sur des principes formels rigoureux : les approches formelles améliorent les représentations conceptuelles «intuitives» ou pragmatiques d'un domaine et permettent de bâtir des systèmes logiquement cohérents à partir desquels il est possible de raisonner, ce qui est indispensable dans le cadre d'un Web Sémantique, où les concepts à utiliser pour le marquage sémantique des ressources, doivent avoir une signification partagée et être réutilisables pour différentes applications.

Etre multi-usage : elle doit être suffisamment générique pour pouvoir être réutilisable pour

différents usages, contrairement aux ontologies définies pour une application particulière, ce qui nécessite de satisfaire les deux conditions précédentes.

3.5 Les constituants d'une ontologie

Une ontologie est constituée de concepts, de relations entre concepts, d'axiomes et d'individus :

— **Un concept** peut représenter un objet matériel, une notion ou une idée. Il peut être divisé en trois parties : un terme (ou plusieurs), une notion et un ensemble d'objets. La notion, également appelée intention du concept, contient la sémantique du concept, exprimée en termes de propriétés et d'attributs, de règles et de contraintes. L'ensemble d'objets, également appelé extension du concept, regroupe les objets manipulés à travers le concept ; ces objets sont appelés instances du concept. L'extension d'un concept peut être vide lorsqu'il s'agit d'un concept générique correspondant généralement à une notion abstraite comme par exemple le concept « vérité ».

— **Les relations** représentent des liens sémantiques entre les connaissances du domaine. Elles peuvent être des propriétés ou des attributs.

1. **Les propriétés** décrivent des interactions entre concepts ;
2. **les attributs** correspondent à des caractéristiques qui permettent de définir le concept de manière unique dans le domaine.

— **Les axiomes** constituent des assertions (ou des prédicats) acceptées comme vraies. Ils permettent de déduire de nouveaux faits à partir des faits connus.

— **Les instances (ou objets ou individus)** représentent des éléments spécifiques d'un concept (ou d'une classe).

3.6 Les langages de définition et de manipulation d'ontologies

il existe de nombreux langages informatiques, spécialisés dans la création et la manipulation d'ontologies. En voici quelques exemples :

3.6.1 DARPA Agent Markup Language

Ce langage est 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.
En réaction à l'apparition de ces nombreux langages, le World Wide Web Consortium (W3C) a mis sur pieds un nouveau langage standard de manipulation d'ontologies web « OWL Web Ontology Language ».

3.6.2 Le langage DAML+OIL

[DAML+OIL] est un langage conçu par le groupe du W3C WebOnt dans le but de dépasser la simple « présentation » d'informations sur le Web pour aller vers l'interopérabilité, la compréhension et le raisonnement sur ces informations. DAML+OIL est le résultat de la fusion de OIL résultant du projet européen OntoKnowledge, et de DAML-ONT, issu du projet DARPA DAML (American Agent Markup Language). Il doit ses primitives de modélisation intuitives aux « frames », sa syntaxe aux standards XML et RDF, sa sémantique formelle et ses mécanismes de raisonnement aux logiques de description. D'un point de vue formel DAML+OIL est basé sur la logique de description expressive SHIQ étendue du constructeur oneOf. DAML+OIL permet de définir en outre un ensemble d'axiomes, il est aussi accompagné de différents outils : un éditeur mais surtout un classifieur FaCT (Fast Classification of Terminology) qui permet de détecter automatiquement les incohérences, et classer automatiquement les concepts d'une ontologie.

3.6.3 OWL

Le langage OWL (Web Ontology Language) est un langage basé sur RDF, il permet de l'enrichir en fournissant un vocabulaire supplémentaire avec une sémantique formelle définie par une syntaxe rigoureuse. Il peut être utilisé pour représenter le sens des termes dans des vocabulaires ainsi que les relations qui existent entre ces termes, et cela par le biais de classes et de propriétés telle que le font les approches orientés objets.

C'est un langage beaucoup plus riche qui, aux notions définies par RDF Schéma, ajoute les propriétés de classe équivalente, de propriété équivalente, d'identité de deux ressources, de différences de deux ressources, de contraire, de symétrie, de transitivité, de cardinalité, etc., permettant de définir des rapports complexes entre des ressources.

OWL fournit des sous langages de plus en plus expressifs conçus pour faire un compromis entre son pouvoir expressif et son pouvoir de raisonnement, il s'agit des langages OWL Lite, OWL DL, OWL Full.

3.7 Les outils d'édition et de validation des ontologies

La construction d'une ontologie passe par trois étapes essentielles :

1. Définition du domaine et de l'objectif ;
2. Construction de l'ontologie ;
   - — Acquisition des connaissances
   - — Représentation formelle
3. Vérification et Validation.

Il existe plusieurs outils dédiés à la construction et la validation des ontologies tels que protégé et Framework Jena.

3.7.1 Protégé

Protégé 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 grâce à un plugin dédié.
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. 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». 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. Une fonctionnalité intéressante de Protégé concerne la possibilité d'effectuer des requêtes sur l'ontologie en cours d'édition. [1] [2]

3.7.2 OILEd

OILEd est un éditeur graphique développé par l'Université de Manchester qui permet à l'utilisateur de construire des ontologies représentées en DAML+OIL. Le modèle de OILEd est basé sur DAML+OIL, tout en offrant une interface de modélisation de type « frame », paradigme plus familier aux utilisateurs que celui de la logique de description, il supporte toute l'expressivité de la logique de description OIL et DAML+OIL. Les classes sont définies en terme de leurs super-classes, de leurs propriétés avec restrictions de type, et en outre la possibilité de définir des axiomes, par exemple pour définir des classes disjointes. Le modèle permet de définir des descriptions complexes comme valeur des attributs, à l'opposé de la plupart des éditeurs de frames où les classes doivent être nommées pour pouvoir être utilisées.

3.7.3 Framework Jena

Jena 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.

3.7.4 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'il vient d'écrire. Tout comme le W3C propose un

1. tutoriel protégé :http ://www.iro.umontreal.ca/ lapalme/ift6282/OWL/EtapesCreationOntologie.html.
2. Introduction à protégé : https ://inf6070.teluq.ca/ressources/introduction-a-protege/

validateur, 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.

3.8 Représentation des connaissances d'une ontologie

Les langages permettant la représentation des ontologies sont les langages à base de Frame, les logiques de description et les graphes conceptuels. Mais avant de présenter ces langages, nous sommes dans l'obligation de donner un aperçu sur leur ancêtre qu'est le langage des réseaux sémantiques.

3.8.1 Les réseaux sémantiques

L'idée centrale des réseaux sémantiques est de décrire la réalité sous formes de graphes (réseaux) composés de nœuds qui représentent les concepts, reliés par des arcs, exprimant la relation entre les concepts. Les nœuds et les arcs sont en général étiquetés. Aux premiers sont associés les objets (concepts, événement, situation), aux seconds les relations entre les objets, d'où la désignation de " structure objet - relation ". Les liens les plus spécifiques de ce type de réseau sont les liens " sorte - de " exprimant la relation d'inclusion des classes , et les liens " est - un " qui représente la relation d'appartenance d'un élément à une classe lorsqu'elle n'est pas exprimée dans le lien " sorte - de ". D'autres liens sont utilisés comme " à pour partie ", " instrument pour "...Ces différents liens permettent de définir l'une des notions les plus importante des réseaux sémantiques qui est la déduction par héritage des propriétés.

L'héritage est essentiellement basé sur la transitivité de la relation " sorte - de " qui combinée aux autres relations, devient un mécanisme d'héritage très complexe. Lorsqu'un concept est décrit comme un des composants des classes, cela signifie que cette classe regroupe un certain nombre d'objets (concepts) qui partage avec lui certaines propriétés. Quand le champs " sorte - de " est monovalué, on dit que les mécanismes d'héritage sont simples. Contrairement, lorsque le champs " sorte - de " est engendré par plus de deux concepts, on dit que le mécanisme d'héritage est multiple.

De nombreuses études ont montré que ce type de graphe manque de précision sémantique et mène à des confusions entre les relations et aussi entre les classes et individus. Elles ont mené à la définition de nouveaux formalismes tels que les frames, les logiques de description et les graphes conceptuels.

3.8.2 Les graphes conceptuels

Le modèle des Graphes Conceptuels (GC) est un modèle opérationnel de représentation de connaissances, qui appartient à la famille des réseaux sémantiques. Ce modèle est mathématiquement fondé sur la logique et la théorie des graphes. Cependant, pour raisonner à l'aide de GC, deux approches peuvent être distinguées :

1. considérer les GC comme une interface graphique pour la logique et donc raisonner à l'aide de la logique ;

2. considérer les GC comme un modèle de représentation à part entière disposant de ses propres mécanismes de raisonnement fondés sur la théorie des graphes. ;

Les graphes conceptuels, sont des graphes finis, connectés et bipartites.

— Les deux sortes de nœuds d'un graphe bipartite sont les concepts et les relations conceptuelles.

— Chaque relation conceptuelle possède un arc ou plus, chacun d'entre eux doit être lié à un concept.

— Si une relation a n arcs, elle est dite n-adique.

— Un unique concept peut former un graphe conceptuel, mais chaque arc de chaque relation conceptuelle doit être lié à un concept quelconque.

3.8.3 Les frames

On définit un " frame " comme une structure de données regroupant l'ensemble des connaissances relatives à un concept. Un frame est un prototype décrivant une situation ou un objet standard. Il sert de référence pour comparer des objets que l'on désire reconnaître, analyser ou classer. Les prototypes doivent prendre en compte toutes les formes possibles d'expression de la connaissance. Les frames possèdent par conséquent une richesse descriptive que n'offrent pas les classes.

Un " frame " est composé d'un ensemble d'attributs qui sont les propriétés caractérisant le concept. Ces attributs contiennent des facettes qui décrivent l'ensemble des valeurs possibles pour cet attribut. Ces facettes peuvent être de deux formes : déclaratives et procédurales. Les premières associent des valeurs aux attributs, alors que les secondes décrivent les procédures appelées réflexes, qui sont activées lors des accès à ces valeurs. Un frame n'a donc pas de comportement propre décrit par des méthodes. Les " frames " appartenant à un ensemble de " frames " propres à un même thème sont organisés dans une même structure hiérarchisée d'héritage d'attributs.

Un sous frame est une spécialisation d'un ou plusieurs frames pères, appelés superframes dont il hérite les couples attribut - facette. L'héritage est dynamique. Les couples hérités ne sont pas recopiés dans les frames. De cette façon aucune mise à jour n'est nécessaire lorsqu'une propriété d'un frame, héritée et non masquée, est modifiée dans l'un des superframes.

3.8.4 Les logiques de description

Les logiques de description issues des frames reposent sur trois notions de base : les concepts représentant des classes (ensemble d'objets), les rôles (relations liant deux objets) et les individus (objets représentant les classes qu'ils instancient). Pour décrire ces éléments, deux structures sont utilisées : la T-BOX et la A-BOX.

La **T-BOX** (boîte terminologique) comprend la description des concepts et des rôles. Cette description est structurée à l'aide du lien hiérarchique sorteDe. Deux concepts particuliers figurent au minimum dans la T-BOX : le concept le plus générique (anything) et le concept le plus spécifique (nothing).

La **A-BOX** (boîte assertionnelle) est constituée des individus, de leur description et des règles qui leur sont attachés. Les inférences reposent sur la reconnaissance d'instances de concepts à partir de leur définition, la détection des concepts plus généraux ou plus spécifiques, et la classification ordonnant les concepts dans la hiérarchie. Les logiques de description sont plus flexibles que les frames et reposent sur une sémantique et une syntaxe rigoureuses.

La plupart des logiques de description divisent la connaissance en deux parties :

— **les informations terminologiques :** définition des notions basiques ou dérivées et de comment elles sont reliées entre elles. Ces informations sont "génériques" ou "globales", vraies dans tous les modèles et pour tous les individus.

— **les informations sur les assertions :** ces informations sont "spécifiques" ou "locales", vraies pour certains individus particuliers.

3.9 Les différents types d'ontologies

Une classification des ontologies peut être faite selon leur sujet de conceptualisation, de ce fait plusieurs types d'ontologies existent. Elles sont classifiées comme suit :

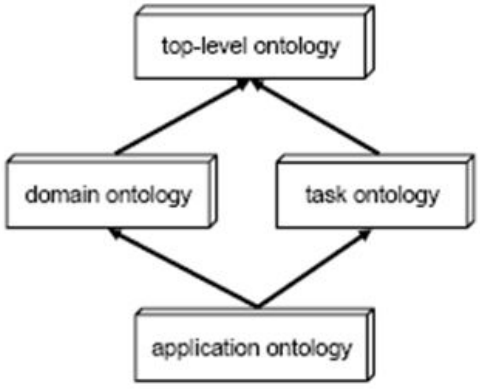


FIGURE 3.1 – Les types d'ontologies

3.9.1 Les ontologies de haut niveau

Elles sont aussi appelées les ontologies supérieures ou de représentation des connaissances, elles essaient de décrire des concepts généraux et abstraits. Elles proviennent des notions philosophiques, décrivant des concepts de haut niveau de toutes les choses qui existent dans le monde tel qu'un objet physique ou un objet abstrait comme le temps ou l'espace.

3.9.2 Les ontologies de domaine

Ce type d'ontologie capture la connaissance dans un domaine spécifique. De ce fait, elles modélisent les connaissances réutilisables dans des domaines précis. Ces ontologies fournissent les concepts et les relations permettant de couvrir les vocabulaires, activités et théories

de ces domaines. Les concepts des ontologies de domaine sont souvent des spécialisations de concepts définis dans des ontologies de niveau supérieur. Plusieurs ontologies de domaines ont étés développés, nous citons par exemple des ontologies dans le domaine de la médecine, la génétique (gene ontology) et dans le tourisme (DATAtourisme).

3.9.3 Les ontologies de tâches

Les ontologies de tâches modélisent les vocabulaires relatifs à une tâche ou une activité générique en spécialisant certains termes des ontologies de niveau supérieur, tel que diagnostiquer une maladie

3.9.4 Les ontologies d'application

Les ontologies d'application fournissent le vocabulaire nécessaire pour décrire une certaine tâche dans un contexte d'application particulier. Elles utilisent les ontologies de domaine et de tâches, et décrivent le rôle qu'une certaine entité de domaine joue dans une tâche spécifique.

3.10 Le cycle de vie des ontologies

1. Détection initiale des besoins et Évaluation : tâches et méthodologies de recueil (entretiens, questionnaires, sondages), identification et analyse, le but étant de dresser un état des lieux initial approfondi. **2.Conception initiale et Évolution :** Cette étape englobe les tâches suivantes :

— spécification des solutions

— acquisition des connaissances nécessaires

— conceptualisation/modélisation

— formalisation

— intégration de ressources existantes

— implantation et gestion des versions

3.Diffusion : déploiement et mise en place de l'ontologie et formation des utilisateurs sur la mise à disposition ou mise à jour.

4.Utilisation : activités reposant plus ou moins directement sur l'utilisation de 'ontologie.

5.Gestion et Planification : suivi et politique globale pour détecter ou déclencher, préparer et évaluer les itérations du cycle.

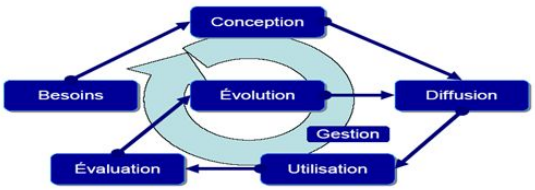


FIGURE 3.2 – Cycle de vie des ontologies

3.11 Le langage OWL

OWL (Web Ontology language) est le langage de définition d'ontologies pour le web de données. Il est beaucoup plus expressif que RDF Schema, qui ne permet que la définition d'ontologies dites légères. OWL permet d'exprimer les notions d'équivalence de classes ou de propriétés, d'égalité de ressources, de différence, de contraire, de symétrie, de cardinalité, etc.

Le vocabulaire du langage OWL est un ensemble de primitives, décrites dans le langage RDF, qui étendent le vocabulaire du langage RDFS. Ces primitives sont identifiées par des URIs dans l'espace de nommage suivant : http ://www.w3.org/2002/07/owl#.

La définition d'une ontologie dans le langage OWL est constituée de définitions de classes, de définitions de propriétés, une méta-description de l'ontologie et éventuellement des descripteurs d'individus.

3.11.1 Définition de classes

Dans OWL, il existe six types de descriptions de classes que nous citons dans ce qui suit :

— le nommage d'une classe ;
— l'énumération des instances d'une classe ;
— l'intersection de descriptions de classe ;
— l'union de descriptions de classe ;
— le complémentaire d'une description de classe ;
— une restriction de propriétés pour la classe décrite.

3.11.1.1 Nommage d'une classe

Le nommage d'une classe consiste en son identification par un URI typé comme étant une instance de owl :Class, et/ou la combinaison de descriptions de classe représentée pas trois propriétés :

— La propriété rdfs :subClassOf qui exprime que l'extension d'une description de classe est un sous-ensemble de l'extension d'une autre description de classe.
— La propriété owl :equivalentClass permet d'exprimer qu'une description de classe a exactement la même extension qu'une autre description de classe.

— La propriété owl :disjointWith qui exprime que l'extension d'une description de classe est totalement disjointe de celle d'une autre description de classe.

3.11.1.2 Intersection de classes

Une intersection de classes est définie par une ressource anonyme ou identifiée, instance de la classe owl :Class et sujet d'une propriété **owl :intersectionOf** dont la valeur est une liste de descriptions de classe. L'extension de la classe ainsi décrite contient exactement les individus qui appartiennent à chacune des extensions des classes listées.

```
<owl:Class rdf:ID="Homme">
        <owl:intersectionOf rdf:Type="Collection">
                <owl:Class rdf:about="#Humain"/>
                <owl:Class rdf:about="#Masculin"/>
        </owl:intersection>
</owl:Class>
```

3.11.1.3 Union de classes

Une union de classes est définie par une ressource anonyme ou identifiée, instance de la classe owl :Class et sujet d'une propriété **owl :unionOf** dont la valeur est une liste de descriptions de classe. L'extension de la classe ainsi décrite contient exactement les individus qui appartiennent à l'extension d'au moins une des classes listées.

```
<owl:Class rdf:ID="Fruits">
        <owl:unionOf rdf:Type="Collection">
                <owl:Class rdf:about="#FruitRouge"/>
                <owl:Class rdf:about="#Agrumes"/>
                ...
        </owl:unionOf>
</owl:Class>
```

3.11.1.4 Le complémentaire d'une classe

Le complémentaire d'une classe est représenté par une ressource anonyme ou identifiée, instance de la classe owl :Class et sujet d'une propriété **owl :complementOf** dont la valeur est une description de classe. L'extension de la classe ainsi décrite contient exactement les individus qui n'appartiennent pas à l'extension de la classe valeur de la propriété owl :complementOf.

```
<owl:Class rdf:ID="Gentil"/>
<owl:Class rdf:ID="Mechant">
        <owl:complementOf rdf:resource="#Gentil"/>
</owl:Class>
```

3.11.1.5 Enumération des instances d'une classe

La propriété **owl :oneOf** permet d'énumérer les instances d'une classe. La valeur de cette propriété est une liste, de type rdf :List, d'éléments de type owl :Thing. L'extension de la classe contient exactement les éléments de cette liste.

```
<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>
```

3.11.1.6 Restriction de propriété

Une classe OWL peut être décrite par l'ensemble des individus qui satisfont certaines restrictions sur certaines propriétés. Ces restrictions sont soit des contraintes sur les valeurs possibles d'une propriété pour un individu de la classe, soit des contraintes sur le nombre d'occurrences d'une propriété et donc le nombre de valeurs possibles de cette propriété pour un individu de la classe. Une restriction est représentée par une ressource anonyme de type **owl :Restriction**, sujet d'exactement deux propriétés :

— une propriété **owl :onProperty**, qui a pour valeur la propriété sur laquelle porte la restriction,

— une seconde propriété, qui précise la nature de la contrainte.

Dans OWL, il existe trois propriétés qui permettent d'exprimer une contrainte sur une propriété et trois propriétés qui permettent d'exprimer une contrainte de cardinalité :

La propriété **owl :hasValue** a pour valeur un individu ou une valeur littérale. Elle permet de restreindre la relation à un seul individu.

Dans l'exemple suivant, le cours de maths ne peut être enseigné que par un seul enseignant spécifié par son identifiant :

```
<owl:Class rdf:about="#CoursMaths">
      <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#estenseignépar"/>
                <owl:hasValue rdf:resource="#949352"/>
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

La propriété **owl :allValuesFrom** a pour valeur une description de classe ou un type de données, elle permet de restreindre les classes liées par une relation. En d'autres termes toutes les valeurs d'une propriété doivent appartenir à une classe spécifique.

Dans l'exemple suivant, Chaque instance de cours de première année ne peut être enseignée que par un enseignant professeur.

```
<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>
```

La propriété **owl :someValuesFrom** a pour valeur une description de classe ou un type de données, elle permet de restreindre les classes en exprimant qu'au moins une valeur d'une propriété provient d'une classe spécifique.

Dans l'exemple suivant est exprimée l'idée qu'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>
```

La propriété **owl :maxCardinality** a pour valeur un entier positif qui exprime que le nombre d'occurrences d'une certaine propriété avec des valeurs différentes ne dépasse pas ce nombre.

La propriété **owl :minCardinality** a pour valeur un entier positif qui exprime que le nombre d'occurrences d'une certaine propriété avec des valeurs différentes est au moins égale à ce nombre.

La propriété **owl :Cardinality** a pour valeur un entier positif qui exprime que le nombre d'occurrences d'une certaine propriété avec des valeurs différentes est au moins égale à ce

nombre. Dans l'exemple suivant, les cours sont enseignés par au moins un enseignant et par au plus trois enseignants :

```
<owl:Class rdf:about="#cours">
   <rdfs:subClassOf>
      <owl:Restriction>
            <owl:onProperty rdf:resource="#estEnseignePar"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
             </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
            <owl:onProperty rdf:resource="#estEnseignePar"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">3
            </owl:maxCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

3.11.2 Définition de propriétés

Une définition de propriété détermine les caractéristiques d'une propriété. La plus simple consiste à lui associer un nom et un URI. OWL distingue deux classes de propriétés :

— La classe **owl :Objectproperty** regroupe les propriétés dont la valeur est une ressource.

— La classe **owl :DatatypeProperty** regroupe les propriétés dont la valeur est un littéral.

Remarque :

OWL réutilise les primitives du langage RDFS rdfs :domain et rdfs :range pour préciser le domaine et la portée d'une propriété.

Dans ce qui suit, un exemple de propriété d'objet qui permet de relier deux classes entre elles :

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

Dans l'exemple suivant, la propriété de type de données permet de relier des individus à des valeurs de données.

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

3.11.2.1 Relations entre propriétés

La propriété **owl :inverseOf** permet d'exprimer que deux propriétés sont inverses l'une de l'autre.

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

La propriété **owl :equivalentProperty** permet d'exprimer que deux propriétés ont la même extension.

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

3.11.2.2 Contraintes globales de cardinalité

La classe **owl :FunctionnalProperty** permet de caractériser les propriétés fonctionnelles, c'est-à-dire les propriétés qui ne peuvent avoir plusieurs valeurs pour un même sujet.

```
<owl:FunctionalProperty rdf:about="époux"/>
```

La classe **owl : InverseFunctionalProperty** permet de caractériser les propriétés qui ne peuvent avoir plusieurs sujets pour une même valeur.

```
<owl:InverseFunctionalProperty rdf:ID="mèreBiologique">
        <rdfs:domain rdf:resource="femme"/>
        <rdfs:range rdf:resource="personne"/>
</owl: InverseFunctionalProperty>
```

3.11.2.3 Caractéristiques logiques

Les classes **owl :SymmetricProperty** et **owl :TransitiveProperty** permettent de caractériser les propriétés respectivement symétriques et transitives. OWL2 introduit trois classes supplémentaires pour exprimer des caractéristiques logiques : **owl :AsymmetricProperty** représente la classe des propriétés asymétriques, **owl :ReflexiveProperty** celle des propriétés refléxives et **owl :IrreflexiveProperty** celle des propriétés irréfléxives.

3.12 Exemple d'une ontologie OWL DL- Description d'une population

L'ontologie OWL présentée dans le diagramme UML suivant décrit un groupe de personnes et leurs relations de parenté, ainsi que leur lieu d'habitation. Pour plus de détails sur cet exemple, il faut se référer à [**23**].

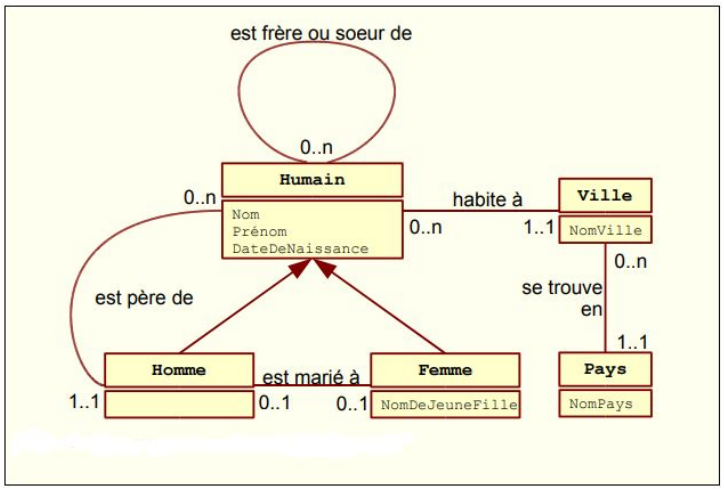


FIGURE 3.3 – Classes composant la population à décrire

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. Dans ce qui suit, le code complet de l'ontologie exemple avec OWL.

< ?xml version="1.0" encoding="ISO-8859-1" ?>

<rdf :RDF

xmlns = "http ://lacot.org/public/owl/famille#"

xmlns :famille = "http ://lacot.org/public/owl/famille#"

xml :base = "http ://lacot.org/public/owl/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="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>
    <habiteA rdf :resource="#Paris" />
</Homme>
<Femme rdf :ID="Marie">
    <nom>Dupond</nom>
    <prenom>Marie</prenom>
    <dateDeNaissance>1976-12-17</dateDeNaissance>
    <nomDeJeuneFille>Martin</nomDeJeuneFille>
    <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>
```

3.13 Conclusion

Les ontologies sont un élément essentiel du web sémantique, elles apportent des vocabulaires partagés nécessaires pour assurer l'interopérabilité entre les systèmes.

Actuellement, le langage OWL du W3C est le langage qui prime pour la représentation des ontologies, c'est un standard qui s'appuie sur RDFS et apporte aux langages du web sémantique, l'équivalent d'une logique de description tout en disposant d'une syntaxe XML. De plus, ces descriptions peuvent être utilisées par la suite par un raisonneur.

Une relation très étroite existe donc entre le web sémantique et le développement d'ontologies, c'est pour cette raison qu'il faut faciliter l'accès et l'utilisation des ontologies à l'ensemble des utilisateurs.

4 Construction d'ontologies à partir de pages Web : Terminae, des textes à une ontologie

4.1 Introduction

Dans le domaine de l'ingénierie des ontologies, la construction d'ontologies à partir de textes constitue un sous-domaine très convoitée par la communauté des chercheurs. L'idée est venue du fait que le passage automatique des textes vers une ontologie ferait gagner du temps aux experts d'autant plus que les textes sont souvent porteurs de connaissances facilement exploitables grâce aux avancées dans le domaine du traitement automatique du langage naturel (TALN).

Plusieurs travaux ont été développés dans ce domaine, Toutefois, il n'existe aucun outil entièrement automatisé qui, à partir d'un corpus initial décrivant un certain domaine, produise l'ontologie de ce domaine. Les techniques existantes sont qualifiées de semi-automatiques et nécessitent l'intervention d'un expert.

4.2 Le texte et ses constituants

Un texte est un ensemble de phrases et de mots cohérents et ordonnés qui permettent d'interpréter et de transmettre les idées d'un auteur. Des textes sont écrits tous les jours : Un message instantané, une recette, le corps d'un courrier électronique est constitué d'un texte, pourvu qu'il réponde à certaines caractéristiques. Les textes contiennent des éléments linguistiques qui peuvent servir dans la construction d'ontologies. Ces éléments sont les candidats-termes, les entités nommées, les relations lexicales ou spécialisées et les classes de mots. Dans la suite de cette section, nous présentons chacun de ces éléments :

4.2.1 les candidats-termes

L'analyse terminologique d'un corpus permet d'identifier des mots simples ou composés qui semblent avoir un fonctionnement terminologique. Dans le processus de construction d'ontologies, l'extraction terminologique permet de repérer la trace des concepts qui sont mentionnés

dans le corpus. Elle sert donc au repérage du vocabulaire conceptuel.

4.2.2 les entités nommées

On désigne par « entités nommées » des expressions textuelles qui s'apparentent à des noms propres. Il désignait au départ les noms de personnes, de lieux, de compagnies, etc., mais a été élargi depuis aux expressions numériques, urls, noms de gènes ou de médicaments, etc. Les entités nommées sont utilisées comme un moyen pour peupler les ontologies avec des instances de concepts.

4.2.3 les relations lexicales

Les relations lexicales sont des relations linguistiques fréquentes. Comme exemple, nous avons la relation d'hyperonymie qui est exprimée au niveau de la phrase (« un chat est un mammifère »), la relation de synonymie qui exprime une proximité sémantique entre mots ou expressions, La méronymie, relation de partie à tout et enfin L'antonymie, qui exprime les contraires peut être utile pour le repérage de concepts disjoints

4.2.4 les relations spécialisées

Les relations spécialisées sont des relations sémantiques qui sont dépendantes d'un domaine particulier. Elles permettent d'enrichir l'ontologie avec majoritairement des relations entre concepts.

4.2.5 les classes sémantiques

Les classes sémantiques sont des regroupements de termes faisant référence à un même terme.

4.2.6 axiomes et règles

Certaines phrases dans les textes peuvent être modélisées en tant qu'axiomes ou règles dans une ontologie. Exemple :Le chemin rural est avant tout une voie privée communale affectée à la desserte des riverains et ouverte au public.

4.3 Processus général de passage d'un texte vers une ontologie

Pour pouvoir construire des ontologies à partir de textes, la plupart des méthodes dans ce domaine s'appuient sur quatre étapes : constitution d'un corpus de documents, analyse linguistique du corpus, conceptualisation et opérationnalisation de l'ontologie. Ceci permet de passer du niveau textuel vers un niveau conceptuel ou la connaissance est décrite à travers des concepts et des relations entre ces concepts.

en d'autres termes, il faut passer en premier lieu par l'identification des :

— Termes pertinents pour l'ontologie à construire ;

— Relations de synonymie entre ces termes ;

— Concepts et de leurs attributs ;

— Relations taxonomiques (hiérarchie) entre concepts ;

— Relations non taxonomiques (génériques) entre concepts ;

— Règles spécifiant les contraintes sur les propriétés des concepts et les relations entre concepts.

A la fin, une formalisation et une intégration des concepts au sein d'une base de connaissances formelle sera nécessaire.

4.4 L'analyse linguistique des textes

Dans le processus de construction d'ontologies, l'extraction terminologique permet de repérer la trace des concepts qui sont mentionnés dans le corpus. Elle sert donc au repérage du vocabulaire conceptuel.

Cette tâche requière au préalable une analyse linguistique des textes. Dans cette phase préparatoire, on distingue trois opérations principales : l'identification des termes, leur regroupement en classes sémantiques et leur structuration en réseau terminologique.

4.4.1 Identification des termes du domaine

Les outils du TAL (Traitement Automatique de la Langue) sont généralement utilisés pour extraire un ensemble de termes candidats (mots ou groupe de mots). Ces termes candidats deviennent termes après validation de l'utilisateur. Cette validation se fait de deux manières, soit par une analyse statistique, soit par une analyse distributionnelle.

L'analyse statistique consiste à mesurer la pertinence d'un terme dans un document en s'intéressant à sa fréquence d'apparition. L'analyse distributionnelle consiste à trouver des éléments du discours et éclairer les relations entre eux afin de « repérer des régularités et de construire des classes paradigmatiques à partir de là". Une fois les candidats termes validés, il est important de leur associer un poids et de les trier par ordre de pertinence pour guider le processus de conceptualisation sans toutefois l'automatiser.

4.4.2 Construction des classes sémantiques

Dans cette étape les termes trouvés doivent être regroupés dans des classes sémantiques. Chaque classe représentera ainsi un concept du domaine.

4.4.3 Construction de la structure

Pour structurer l'ontologie, il faut identifier les relations sémantiques entre les termes. Ces dernières pourront jouer le rôle des relations conceptuelles. Différents types de relations sémantiques sont exploitées pour la construction d'ontologies : relation hiérarchiques ou relations plus spécialisées. Cependant, l'extraction des relations terminologiques est une tâche difficile.

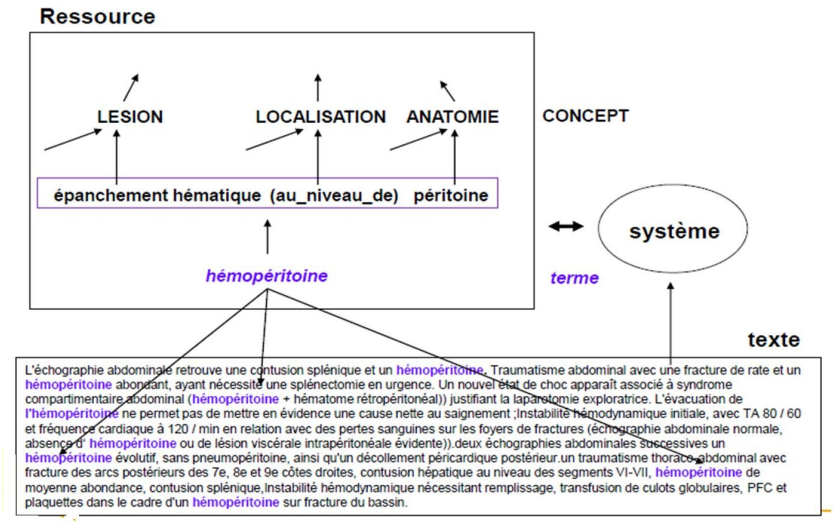


FIGURE 4.1 – Exemple d'identification des termes candidats

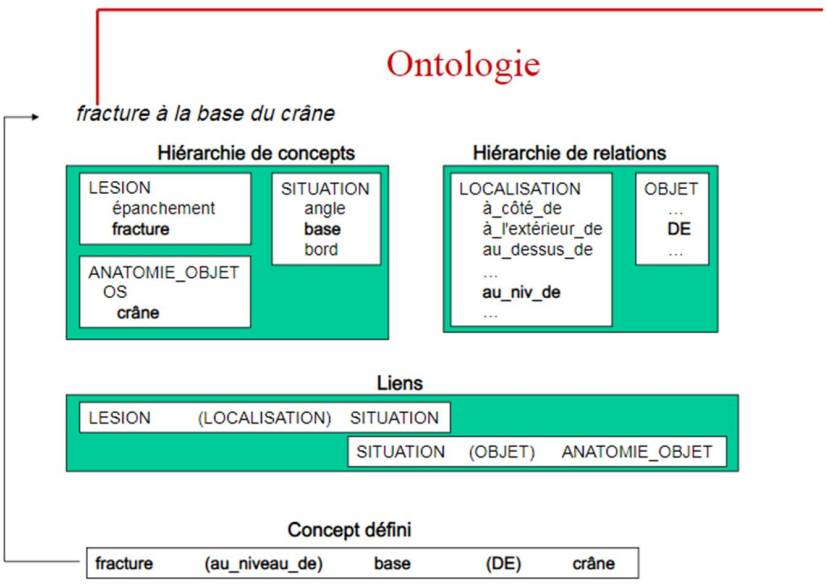


FIGURE 4.2 – ontologie résultante de la figure 4.1

4.5 Quelques méthodes de construction d'ontologies à partir de textes

4.5.1 TexttoOnto

Text2Onto est un outil de construction d'ontologies qui implémente des algorithmes de fouille de textes sur des corpus textuels pour construire des ontologies de manière semi automatique. Cet outil inclut plusieurs traitements comme, par exemple, l'extraction des termes en utilisant soit des calculs statistiques, soit des expressions régulières ou encore par identification des relations à l'aide de patrons lexico-syntaxiques ou de calculs de proximités.

4.5.2 OntoCase

OntoCASE est un outil de conception supervisée d'ontologies impliquant un processus de transformation d'un modèle de connaissances semi-formel en ontologie. OntoCASE s'appuie sur une démarche d'Ingénierie Dirigée par les Modèles. Contrairement à Text2Onto et Terminae où les étapes de construction d'ontologies sont reliées entre elles par des programmes dits "codés en dur", OntoCASE utilise des règles de transformation décrites au travers d'une ontologie dite de transformation écrite en OWL-SWRL.

4.5.3 Terminae

Terminae est à la fois une méthode et un outil d'aide à l'élaboration de ressources terminologiques et ontologiques à partir de textes. Cet outil intègre un environnement d'étude terminologique, un environnement d'aide à la conceptualisation et un système de gestion d'ontologies.

C'est une méthode qui repose sur des bases théoriques issues de la linguistique et de la représentation des connaissances (extracteur de termes, détecteur de synonymie, analyseur syntaxique). Une fiche terminologique affiche les occurrences d'un terme dans le corpus et lui associe sa définition, les termes synonymes et les concepts terminologiques correspondants. La phase de conceptualisation est manuelle.

La figure 4.3 résume les étapes de construction d'une ontologie dans l'approche Terminae.

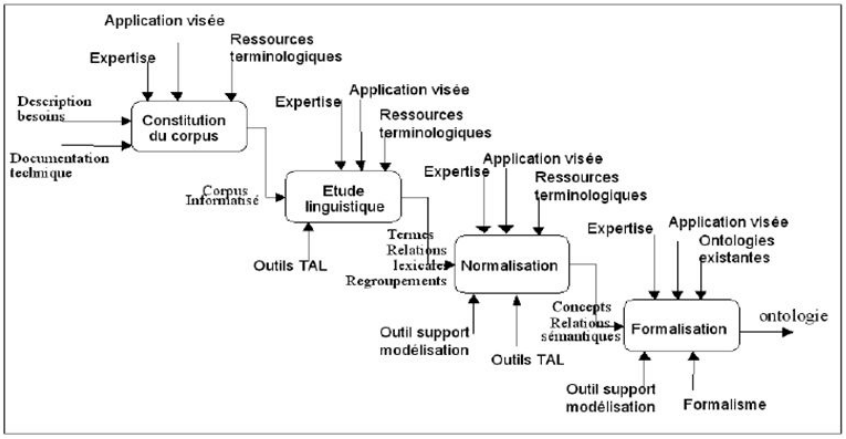


FIGURE 4.3 – Le processus de construction d'Ontologie de Terminae

Après la constitution du corpus et une étude linguistique, menée à l'aide d'outils du TAL, les connaissances sont normalisées puis formalisées. Dans ce qui suit, les deux dernières étapes seront reprises avec plus de détails :

la normalisation qui est constituée de deux parties :

1. La première étape étudie les candidats-termes et les relations lexicales déterminés par les outils linguistiques. Le cogniticien choisit ceux qui ont un sens dans le corpus et qui présentent un intérêt par rapport aux objectifs du modèle. Ces éléments font l'objet de fiches terminologiques.

2. La deuxième étape de la normalisation consiste à définir des concepts et des relations sémantiques à partir des termes et des relations lexicales précédentes.

La formalisation qui comprend l'élaboration et la validation de l'ontologie. Les concepts et relations sémantiques des fiches de modélisation sont traduits en concepts et rôles formels dans le langage de description de l'ontologie. Les concepts peuvent être classifiés en hiérarchie.

4.5.4 Illustration de la méthode Terminae par un exemple

L'étude de cas présentée dans ce qui suit est tirée de [**terminae1**]. Pour illustrer la méthode proposée par TERMINAE et l'utilisation du logiciel, les auteurs ont pris comme exemple la constitution d'une terminologie à partir d'un texte portant sur la fabrication du verre. Ce texte court (3 pages, 1 425 mots) est extrait d'un manuel pédagogique destiné à des élèves du secondaire. L'étude du texte sert de point de départ pour organiser en une terminologie structurée des connaissances sur la nature des matières vitreuses et sur la fabrication du verre. Pour cet exemple, les étapes de la figure 4.4 seront suivies.

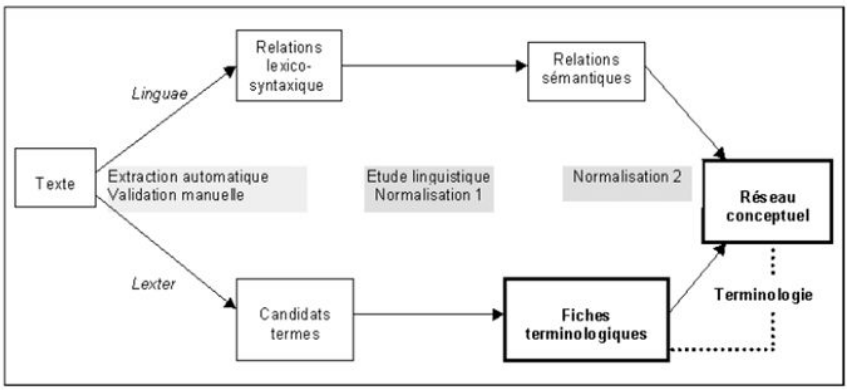


FIGURE 4.4 – Processus de constitution d'une terminologie dans Terminae

4.5.4.1 Extraction des candidats termes

TERMINAE utilise Lexter pour dégager des termes pouvant donner lieu à la définition de concepts. Les résultats de Lexter sont regroupés selon trois fichiers dont l'un contient les occurrences des candidats-termes et un autre les candidats-termes retenus. un expert concerné par cette terminologie peut valider les candidats-termes, afin de ne retenir que ceux pertinents pour le domaine considéré puis les regrouper avec leur synonymes. A partir du texte sur le verre, 472 candidats-termes (correspondant à 790 occurrences dans le texte) ont été extraits. Parmi les plus fréquents, on retrouve des mots simples très significatifs du contenu du texte : verre, température, liquide, oxyde, élément, atome, solide... La fenêtre de la figure 4.5 montre les occurrences du candidat-terme liquide dans le texte.

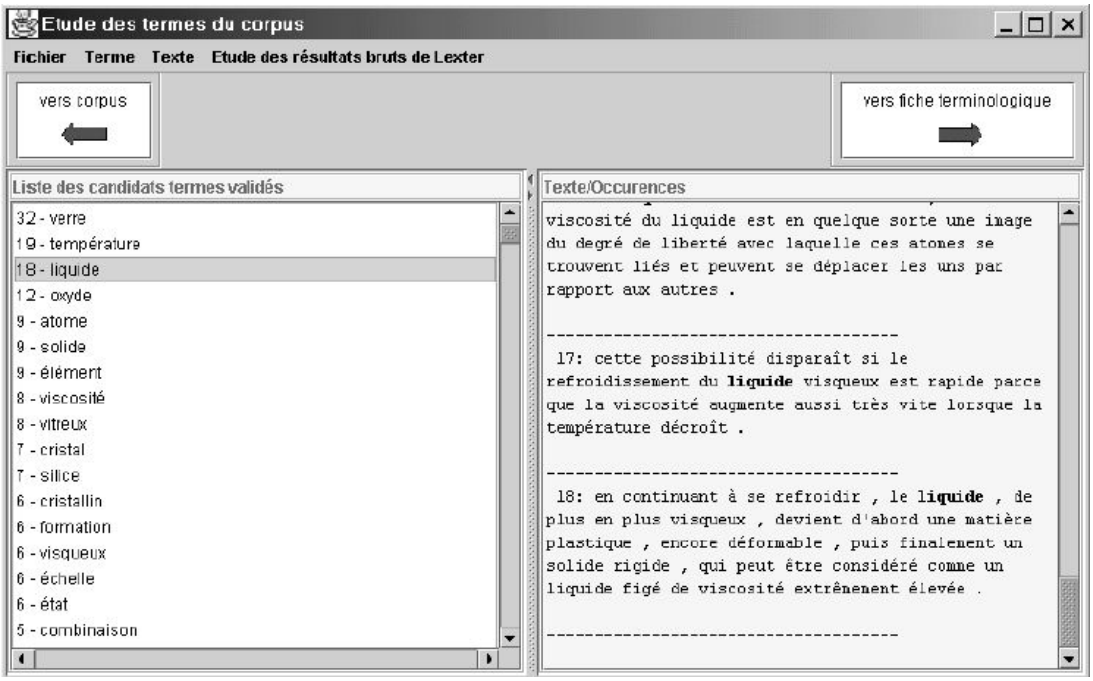


FIGURE 4.5 – Fenêtre d'étude des termes du corpus

Parmi les groupes nominaux extraits, on trouve état vitreux, température de liquidus,oxyde formateur, solide rigide, verre courant, composition chimique, matières minérales ou encore composé cristallin. TERMINAE crée ensuite le fichier des candidats-termes validés et de leurs occurrences. Une fenêtre spécifique permet à l'utilisateur de sélectionner, parmi les candidats-termes simples et composés, ceux pour lesquels il souhaite créer une fiche terminologique.

4.5.4.2 Elaboration des fiches terminologiques

Un candidat-terme validé accède au statut de terme lorsqu'une fiche terminologique est créée pour le décrire. Cette fiche comporte des informations relatives à sa création et ses mises à jour (auteur et date de création) et des informations lexicales associées au terme sous forme de rubriques. la figure 4.6 montre la fiche terminologique du terme liquide.

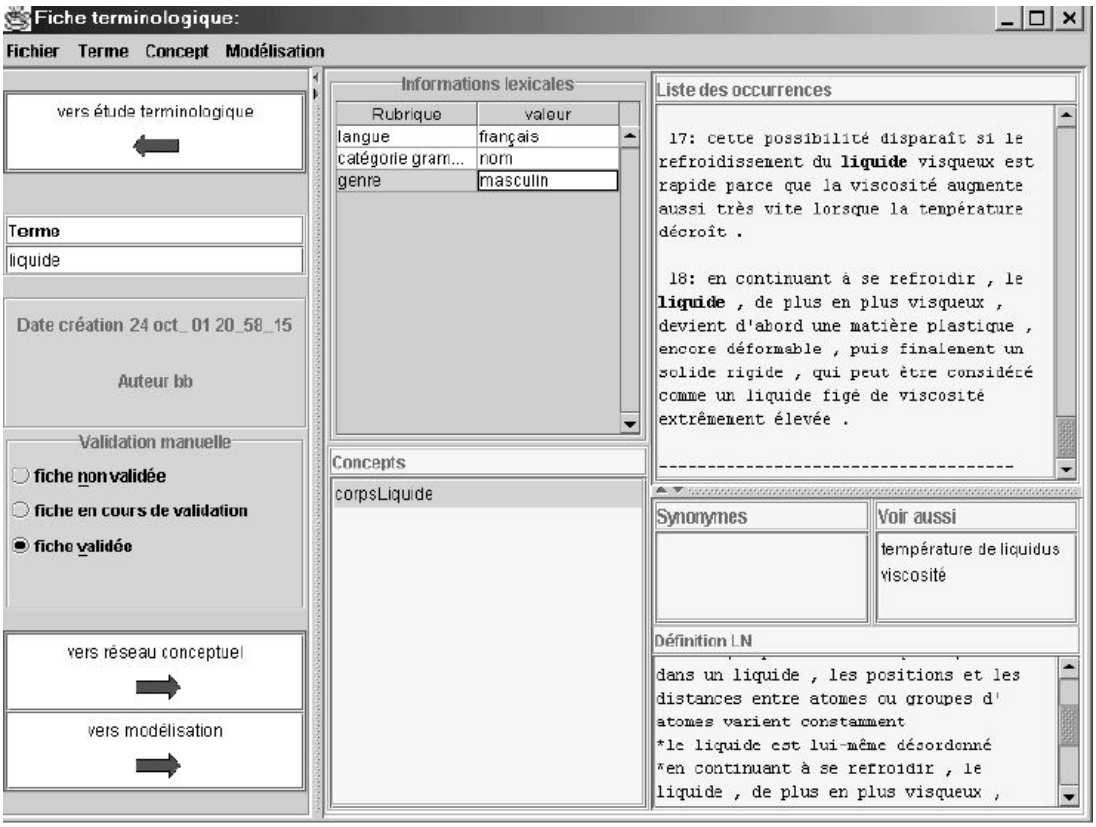


FIGURE 4.6 – Fiche terminologique du terme liquide

Une fiche terminologique rend compte de la première partie du processus de normalisation.

Pour décider de la création d'une fiche terminologique, l'un des points de départ possibles consiste à considérer d'abord les candidats-termes les plus fréquents (verre, température, liquide, oxyde dans notre cas). Un autre pourrait être de se focaliser sur un candidat-terme renvoyant à un concept central, comme état vitreux ou viscosité ou encore des groupes nominaux dont fait partie le candidat-terme. Par exemple pour verre, nous trouvons verre de silice, verre d'oxydes, verre courant. On peut aussi considérer des verbes dont verre est sujet : verre

peut contenir des formateurs. Enfin, on peut lister des actions possibles sur le verre : réchauffer un verre, former des verres, etc..

Comme étape suivante, il est nécessaire d'étudier les relations entre les concepts, via la recherche de relations entre les termes grâce au module Linguae.

4.5.4.3 Recherche de relations entre les termes

La recherche de relations entre les termes est réalisée par le module Linguae qui permet de projeter des marqueurs lexico-syntaxiques sur un corpus et de déterminer ainsi des relations entre les termes. Dans Linguae, un marqueur est décrit par un ou plusieurs motifs, selon sa complexité et sa variabilité.

Linguae offre un jeu de relations prédéfinies (hyperonymie, meronymie, synonymie) auxquelles sont associés des marqueurs connus ; il est possible de rajouter des relations spécifiques et des marqueurs supplémentaires.

L'ensemble des motifs associés aux marqueurs d'une relation peut être appliqué automatiquement, et les résultats sauvegardés pour une interprétation ultérieure.

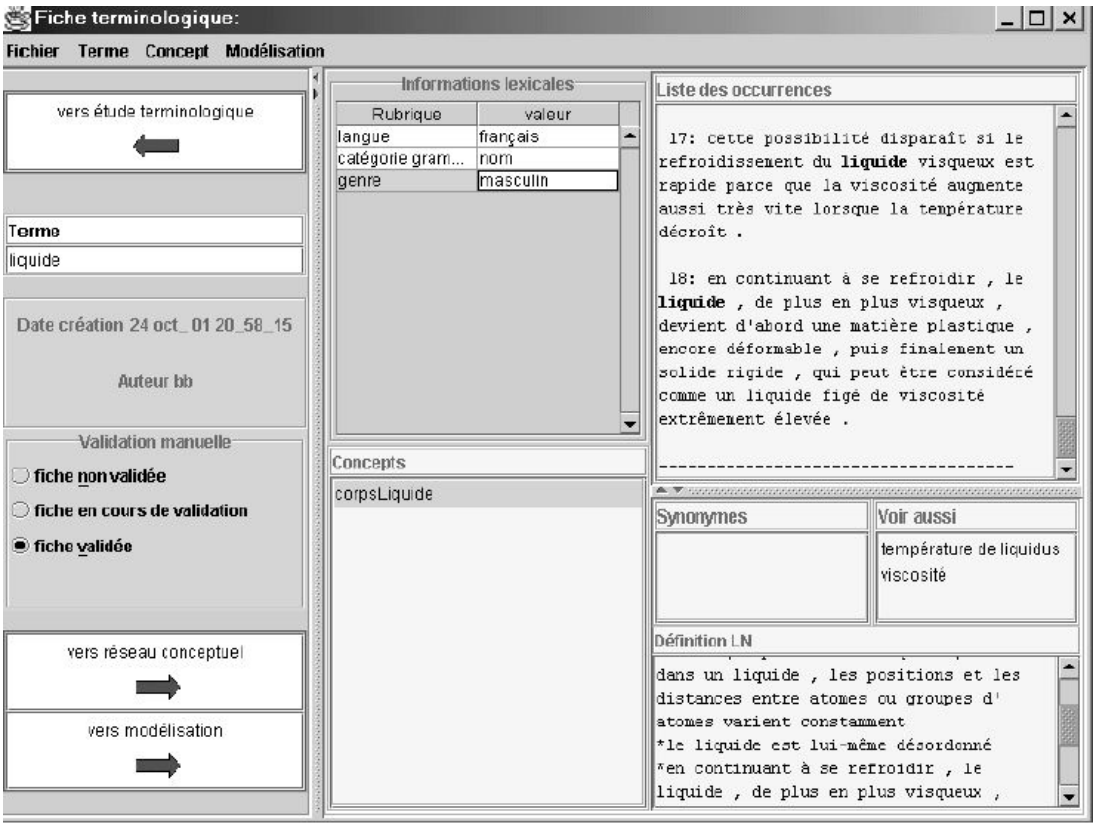


FIGURE 4.7 – Fenêtre des relations lexico-syntaxiques

Linguae permet de visualiser les résultats et de passer des relations entre termes à des relations sémantiques entre concepts. La figure 4.8 montre un exemple de relations sémantiques.

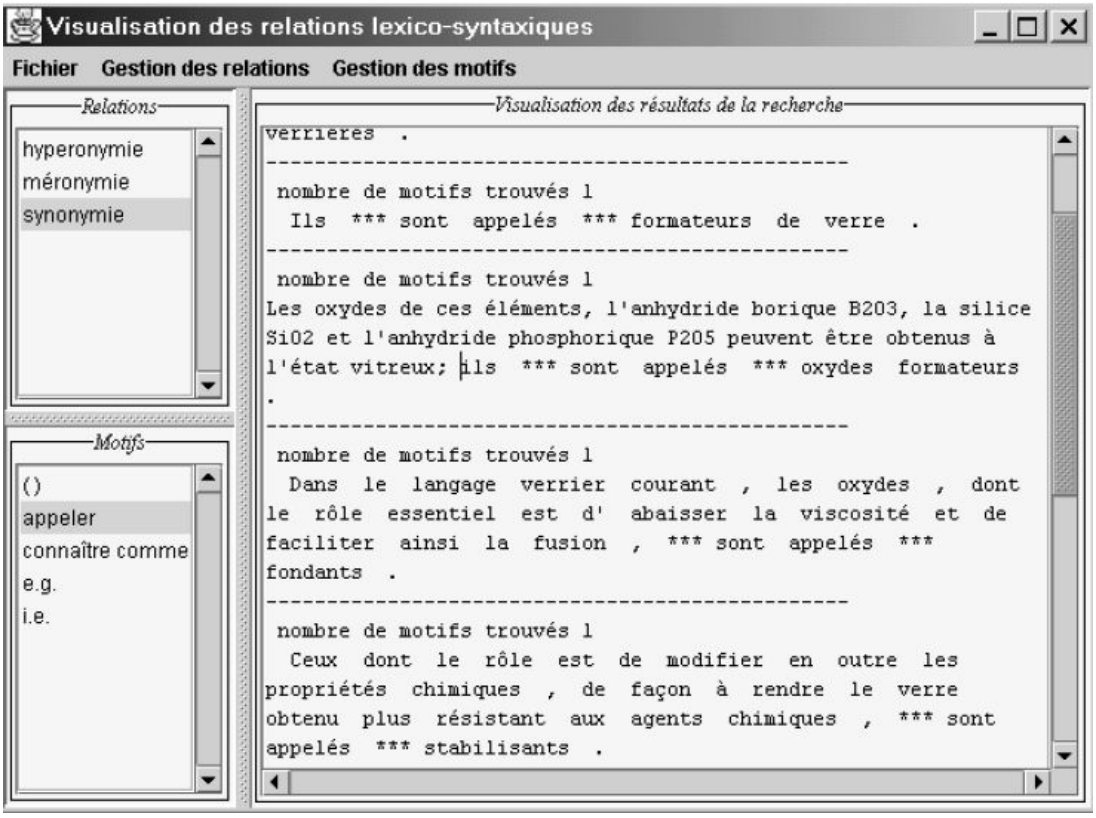


FIGURE 4.8 – Fenêtre d'étude des relations sémantiques

4.5.4.4 Construction du réseau conceptuel

cette étape permet de structurer les concepts correspondant aux termes, appelés « concepts terminologiques », en utilisant les relations mises à jour et les définitions des fiches terminologiques.

La structuration va faire apparaître de nouveaux concepts et relations, et conduire au renommage de certains pour faciliter l'interprétation du réseau.

Par exemple, les concepts FUSION et SOLIDIFICATION sont plus compréhensibles s'ils sont regroupés sous un concept qui exprime la notion de processus chimique qui lui n'existe pas dans le corpus d'origine mais qui a été crée.

Construire ce réseau revient à structurer hiérarchiquement les concepts normalisés comme le montre La figure 4.9 .

la vue d'ensemble du réseau conceptuel permet d'abord de choisir les concepts et rôles primitifs, puis de construire l'ontologie selon une démarche descendante en validant incrémentalement chaque définition de concept formel décrite dans une fiche de modélisation.

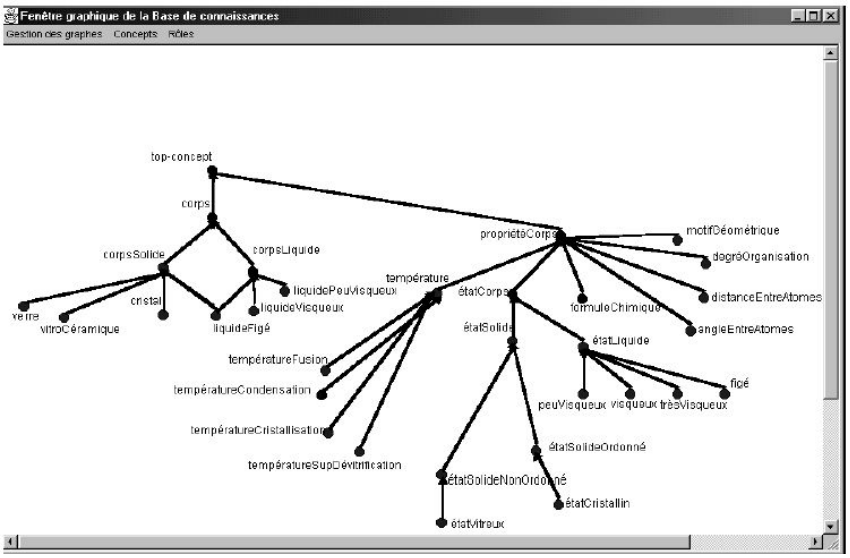


FIGURE 4.9 – Hiérarchie des concepts

4.5.4.5 Passage à une ontologie

TERMINAE permet aussi de générer le réseau conceptuel sous différents formats comme OIL, RDF , et également de récupérer des ontologies en provenance de ces formats. Il devient possible d'échanger des ontologies, d'utiliser des logiques de description plus riches que celle de TERMINAE pour enrichir une ontologie, de compléter une ontologie existante d'éléments textuels facilitant une interprétation à partir de textes. La figure 4.10 montre l'ontologie obtenue en format RDF.

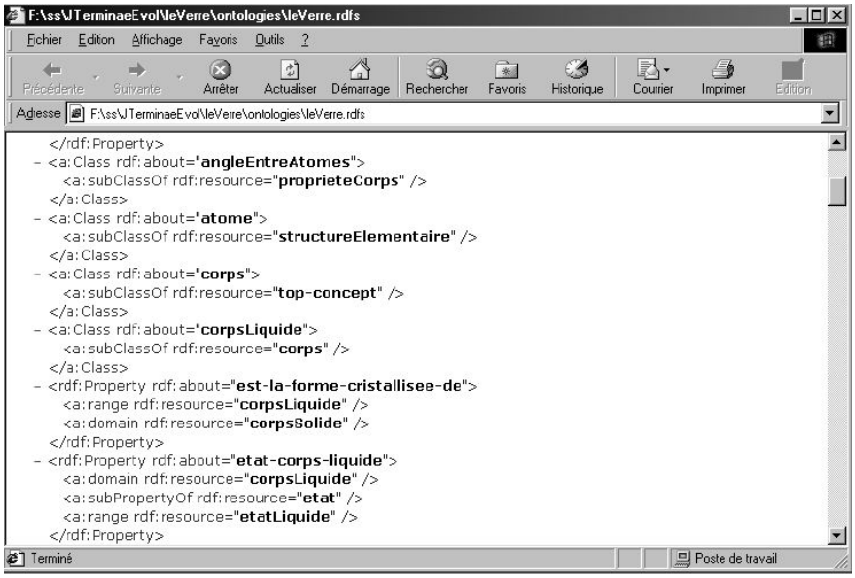


FIGURE 4.10 – Ontologie obtenue en format RDF

4.6 Conclusion

Pour résumer ce chapitre, nous pouvons dire que le passage d'un texte vers une ontologie passe obligatoirement par :

- Une étape d'étude linguistique qui consiste à analyser le corpus pour en extraire les termes porteurs de connaissances ainsi que les relations qui les lient ;
- Une étape de conceptualisation qui consiste à transformer les termes obtenus en concepts et les relations lexicales en relations sémantiques ;
- Une étape de formalisation qui consiste à exprimer le modèle conceptuel obtenu avec un langage formel qui sera compris par la machine.

Terminae s'est imposée dans ce domaine et reste la plateforme d'aide à la construction de ressources termino-ontologiques à partir de ressources textuelles la plus complète de nos jours.

5 Utilisation des ontologies dans les services web sémantiques

5.1 Introduction

Le web sémantique et les services web sont deux domaines de recherche très importants qu'on peut associer aux technologies de l'Internet. Quand ces deux domaines s'associent en mettant les technologies du web sémantique au service des services web, on parle alors des services web sémantiques. Dans ce chapitre, nous présentons brièvement les principaux frameworks de description des services Web Sémantiques, qui sont les standards SAWSDL, OWL-S et WSMO.

5.2 Les services Web

Un service web est une application se trouvant sur internet, utilisée par un client et mise à la disposition des utilisateurs par un fournisseur de services. Les services web les plus connus sont dans le domaine du tourisme comme la réservation de chambres d'hôtel en ligne. Selon le W3C, un service web est une application ou un composant logiciel qui vérifie les propriétés suivantes :

— Il est identifié par un URI ;
— Ses interfaces et ses liens (binding) peuvent être décrits en XML ;
— Sa définition peut être découverte par d'autres web services ;
— Il peut interagir directement avec d'autres web services à travers le langage XML et en utilisant des protocoles Internet.

L'idée derrière les web services et de remplacer les tâches effectuées par les humains par des machines en vue de permettre une découverte et/ou une composition automatique. L'architecture de référence des web services comme le montre la figure 5.1 s'articule autour des trois rôles suivants :

— **Le fournisseur de service** : correspond au propriétaire du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service ;
— **Le client** : correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle-même un web service ;

— **L'annuaire des services** : correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

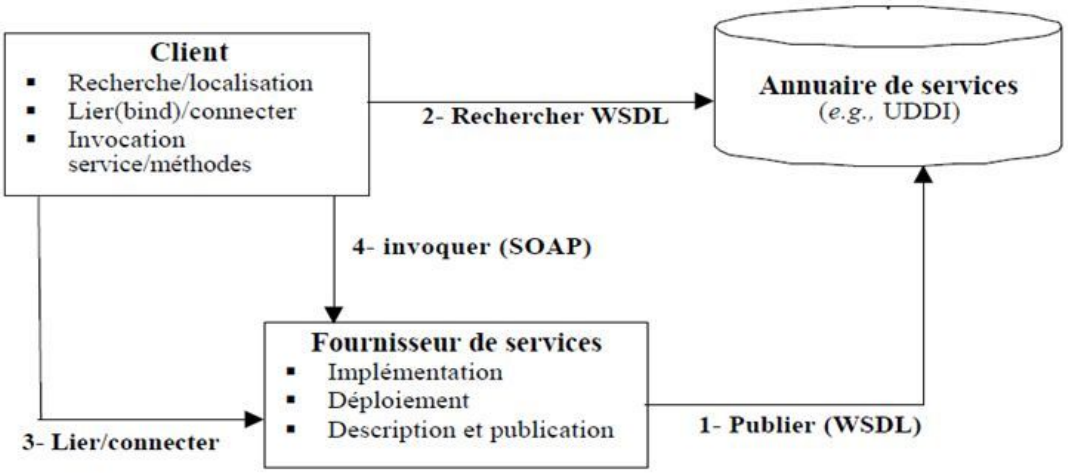


FIGURE 5.1 – Architecture des web services

Les services Web fournissent des standards qui facilitent le transport, l'invocation, la description et la recherche :

— Le protocole SOAP (Simple Object Access Protocol) transporte les messages XML entre les services Web et permet également d'invoquer des services Web ;

— Le langage WSDL (Web Services Description Language) offre la possibilité de décrire l'interface d'un service Web ;

— Le standard UDDI (Universal Description Discovery and Integration) décrit l'annuaire dans lequel les services Web sont répertoriés d'une manière homogène. La recherche et la découverte automatique des services devraient être supportées par ce protocole.

Le problème qui existait est que l'usage des web services était limité aux utilisateurs humains plutôt qu'aux machines. Ceci est dû aux nombreuses connaissances nécessaires à l'automatisation des services qui étaient soit absentes, soit décrites pour être interprétées et exploitées par des humains. Il a donc été nécessaire de s'orienter vers des services intelligibles pour des machines, ce qui a amené vers la notion de web service sémantique.

Il fallait donc réfléchir à la façon de décrire formellement les connaissances de manière à les rendre exploitables par des machines. Ceci peut être réalisé par les technologies et les outils du web sémantique et plus spécialement par les ontologies qui peuvent apporter une sémantique commune exploitable par les humains et par les machines et permettant l'automatisation des fonctionnalités suivantes :

— Processus de description et de publication des services ;

— Découverte des services ;

— Sélection des services ;

— Composition des services ;

— Fourniture et administration des services.

5.3 Principales attentes des services web sémantiques

Le manque de descriptions sémantiques des entrées et des sorties dans le WSDL rend impossible le développement de clients logiciels qui peuvent dynamiquement et sans assistance humaine, trouver et appeler avec succès un service. Les spécifications WSDL des services doivent être interprétés par les programmeurs, qui interprètent les noms des éléments du message en utilisant d'autres moyens pour intégrer des services spécifiques avec leurs applications clientes.

L'objectif des services Web sémantiques est d'accompagner les clients pour trouver et utiliser correctement les services nouvellement découverts sans fournir un effort supplémentaire de programmation. Le problème réside donc dans la découverte et la composition des services web car l'infrastructure de base qui repose uniquement sur les standards SOAP, WSDL et UDDI est insuffisante pour rendre automatique certaines tâches.

Pour apporter des solutions à ce problème, le SWS doit :

— Offrir aux fournisseurs la possibilité de publier leurs services ;

— Faciliter la tâche de l'annuaire afin qu'il puisse fournir et composer, de façon automatique, les services recherchés ;

— Permettre au client d'invoquer et surveiller, de façon automatique, un service sélectionné par l'annuaire .

Plusieurs approches ont été développées pour les SWS. Nous passons en revue les plus connues parmi elles OWL-S, SAWSDL et WSMO.

5.4 Les approches de description sémantique pour les services Web

5.4.1 OWL Web Ontology Language for Services (OWL-S)

OWL-S (anciennement connu sous le nom de DAML-S) définit une ontologie supérieure pour la description, l'invocation et la composition des services Web. Elle comprend trois sous-ontologies principales : le service profile, le service model et le service grounding.

Le service Profile est utilisé pour décrire ce que fait le service. Il permet la description, la publication et la découverte des services, en spécifiant une description textuelle à destination des utilisateurs "humains", des propriétés fonctionnelles et des propriétés non fonctionnelles.

Le service Model est utilisé pour décrire comment est utilisé le service, en détaillant le contenu sémantique des demandes, les conditions dans lesquelles des résultats particuliers se produiront et, si nécessaire, les processus étape par étape menant à ces résultats. C'est-à-dire qu'il décrit comment demander le service et ce qui se passe lorsque le service est effectué. Pour les services non triviaux (ceux composés de plusieurs étapes dans le temps), cette description peut être utilisée par un agent demandeur de service d'au moins quatre manières différentes :

1. pour effectuer une analyse plus approfondie afin de savoir si le service répond à ses besoins ;

2. pour composer des descriptions de services à partir de plusieurs services pour effectuer une tâche spécifique ;
3. au cours de l'acte de service, coordonner les activités des différents participants ;
4. pour surveiller l'exécution du service.

Le service Grounding est utilisé pour décrire comment accéder puis interagir avec le service. Généralement, il spécifiera le protocole de communication, les formats de message et d'autres détails spécifiques au service tels que les numéros de port utilisés pour contacter le service.

En résumé, le Service Profile fournit les informations nécessaires à un agent pour découvrir un service, tandis que le Service Model et le Service Grounding, pris ensemble, fournissent suffisamment d'informations pour qu'un agent puisse utiliser un service une fois trouvé.

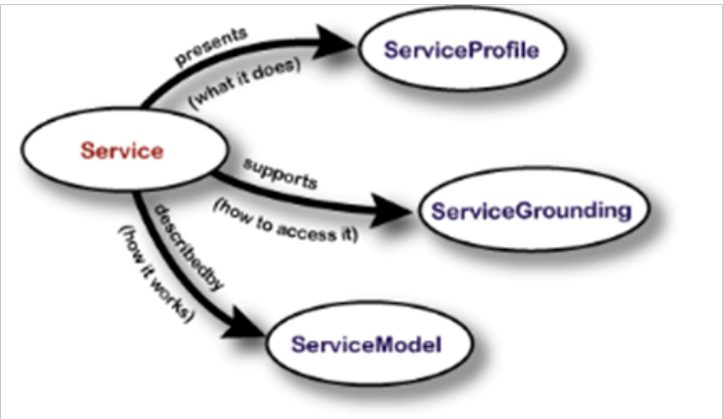


FIGURE 5.2 – Top level of the service ontology

5.4.2 SAWSDL

Les annotations sémantiques pour WSDL et XML Schema (SAWSDL) définissent comment ajouter des annotations sémantiques à un document WSDL telles que les structures de message d'entrée et de sortie, les interfaces et les opérations. Les annotations sur les types de schéma peuvent être utilisées lors de la découverte et de la composition du service Web. De plus, SAWSDL définit un mécanisme d'annotation pour spécifier le mappage de données des types de schéma XML vers et depuis une ontologie ; de tels mappages pourraient être utilisés pendant l'invocation, en particulier lorsqu'une médiation est requise.

Pour réaliser l'annotation sémantique, SAWSDL définit des attributs d'extension qui peuvent être appliqués à la fois aux éléments WSDL et aux éléments XML Schema.

Les annotations sémantiques référencent un concept dans une ontologie. Le mécanisme d'annotation est indépendant du langage d'expression d'ontologie et cette spécification n'exige et n'impose aucun langage d'ontologie particulier.

L'annotation sémantique des documents WSDL est possible grâce à l'extensibilité de WSDL 2.0. En effet, conceptuellement WSDL 2.0 est doté des constructions suivantes : interface, opération, message, binding, service et endpoint. Les trois premiers à savoir interface, operation

et message concernent la définition abstraite du service tandis que les trois autres sont relatifs à l'implémentation du service.

SAWSDL fournie des mécanismes pour référencer des concepts de modèles définis à l'extérieure du document WSDL. Cela se fait grâce à l'attribut «sawsdl». Il existe trois extensions de cet attribut.

La première est **modelReference** et permet d'associer un composant WSDL ou XML Schema à un concept d'une ontologie. Les deux autres sont **liftingSchemaMapping** et **loweringSchemaMapping** et permettent de spécifier le mapping entre les données sémantiques et les éléments XML. **Les SchemaMapping** sont utilisés pour établir la correspondance entre les structures des entrées et des sorties, et sont utiles lorsque les structures XML demandées par le client et celles fournies par le service sont différentes. L'annotation des interfaces, opérations, entrées/sorties et les types XML simple s'effectue en leurs associant un concept dans une ontologie par le biais de l'attribut **modelReference**.

L'exemple suivant illustre bien les annotations ajoutées dans un document WSDL.

```
<wsdl:types>
 <xs:schema targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#" elementFormDefault="qualified">

    <xs:element name="OrderRequest"
          sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#OrderRequest"
          sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/RDFOnt2Request.xml">
          <xs:complexType>
              <xs:sequence>
              <xs:element name="customerNo" type="xs:integer" />
              <xs:element name="orderItem" type="Item" minOccurs="1" maxOccurs="unbounded" />
             </xs:sequence>
         </xs:complexType>
   </xs:element>

<xs:complexType name="item">
     <xs:all> <xs:element name="UPC" type="xs:string" /> </xs:all>  <xs:attribute name="quantity" type="xs:integer" />
</xs:complexType>

<xs:element name="OrderResponse" type="confirmation" />
    <xs:simpleType name="confirmation"
          sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#OrderConfirmation">
         <xs:restriction base="xs:string">
         <xs:enumeration value="Confirmed" />
         <xs:enumeration value="Pending" />
         <xs:enumeration value="Rejected" />
   </xs:restriction>
   </xs:simpleType>
</xs:schema>
</wsdl:types>
```

FIGURE 5.3 – Exemple d'annotations sémantiques ajoutées à un document WSDL dans SAWSDL

5.4.3 WSMO

The Web Service Modeling Ontology (WSMO) complète les normes des services Web syntaxiques existantes en fournissant un modèle conceptuel et un langage (WSML) pour le balisage sémantique de tous les aspects pertinents de Web services généraux. Elle fournit un cadre pour les descriptions sémantiques des services Web et agit comme un méta-model pour ces services sur la base de la fonction de méta-objet (MOF). Le but ultime d'un tel balisage est de permettre l'automatisation totale/partielle des tâches (par exemple, la découverte, sélection, composition, médiation, exécution, suivi, etc.) impliquées dans l'intégration intra- et inter-entreprises des services Web. De tels éléments incluent ontologies, buts, services Web et médiateurs :

— Les ontologies définissent la terminologie utilisée par les autres éléments ;

— Les buts indiquent ce que l'utilisateur attend du service ;

— Les services Web définissent les fonctionnalités offertes ;

— Les médiateurs lient les différents éléments afin de permettre l'interopérabilité entre les composants hétérogènes ;

Le langage WSML (Web Service Modeling Language) est utilisé pour décrire formellement tous les éléments de WSMO et l'environnement d'exécution WSMX permet la découverte, la sélection, la médiation et l'invocation des services Web sémantiques.

5.5 Conclusion

Pour conclure ce chapitre, une petite synthèse s'impose. On peut dire que OWL-S prend en charge partiellement la découverte et l'invocation automatiques des services, par contre WSMO et SAWSDL permettent uniquement l'invocation automatique. WSMO est plus proche de OWL-S et ils sont tous les deux différents de SAWSDL.

S'agissant de SAWSDL, celui-ci s'intéresse à la découverte et l'invocation automatique des services mais il ne s'occupe pas de la composition. Etant proche de WSDL, SAWSDL ne nécessite pas beaucoup d'efforts pour les développeurs habitués à WSDL contrairement à OWL-S et WSMO.

Les outils pour WSMO s'avèrent plus difficiles à développer car celui-ci se base sur WSML, un langage qui n'a pas été utilisé auparavant, tandis que OWL-S et SAWSDL s'appuient sur RDF et XML ce qui fait leur succès.

6 Exercices Corrigés

6.1 RDF

6.1.1 Exercice

6.1.1.1 Enoncé

Donner le Graphe RDF des connaissances suivantes, le document RDF/XML puis la notation N3 correspondante :

L'URIref suivante "http ://www.usto.dz/dptmaths#samiraabid" représente une enseignante qui a pour nom de famille "Abid" et pour prénom "Samira". Elle a pour adresse mail "samira.abid@univ-usto.dz". Elle enseigne au département dont l'uri est "https ://www.univ-usto.dz/faculte/fac-mathinfo/Mathématiques" et dont le nom est "Département Mathématiques". Elle enseigne les cours suivants : "Analyse" et "Algèbre".

6.1.1.2 Corrigé

1. Graphe RDF

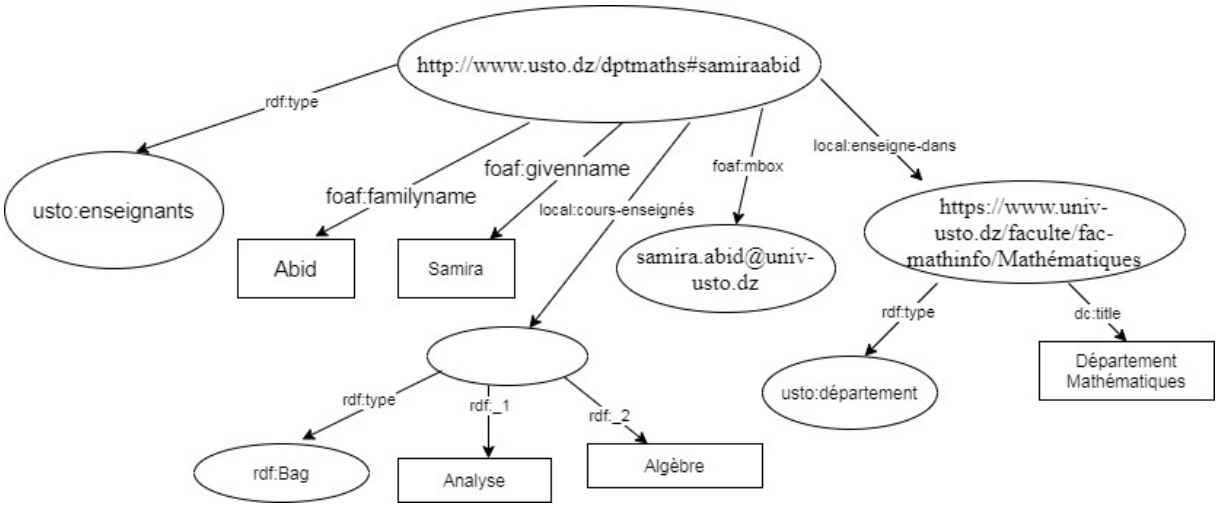


FIGURE 6.1 – Graphe RDF correspondant

2. RDF/XML

Solution 1

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns=http://purl.org/rss/1.0/
        xmlns:dc="http://purl.org/dc/elements/1.1/#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/#"
        xmlns:local="http://www.usto.dz/monvocabulaire#"
        xmlns:usto="http://www.usto.dz/dptmaths#">
<rdf:Description  rdf:about=" http://www.usto.dz/dptmaths#samiraabid ">
        <rdf:type rdf:resource="usto:enseignants"/>
        <foaf:familyname>Abid</foaf:familyname >
        <foaf:givenname>Samira</foaf:familyname >
        <foaf:mbox rdf:resource=" samira.abid@univ-usto.dz "/>
        <local:enseigne-dans rdf:resource=" https://www.univ-usto.dz/faculte/fac-
mathinfo/Mathématiques  "/>
        <local:cours-enseignés>
        <rdf:bag>
                <rdf:_1 > Analyse </rdf:_1>
                <rdf:_2 > Algèbre </rdf:_2>
        </rdf:bag>
        </local:cours-enseignés>
</rdf:Description>
<rdf:Description  rdf:about=" https://www.univ-usto.dz/faculte/fac-
mathinfo/Mathématiques  ">
         <rdf:type rdf:resource="usto:département"/>
        <dc:title> Département Mathématiques</dc:title>
</rdf:Description>
</rdf:RDF>
```

Solution 2

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns=http://purl.org/rss/1.0/
        xmlns:dc="http://purl.org/dc/elements/1.1/#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/#"
        xmlns:local="http://www.usto.dz/monvocabulaire#"
        xmlns:usto="http://www.usto.dz/dptmaths#">
<rdf:Description  rdf:about=" http://www.usto.dz/dptmaths#samiraabid ">
        <rdf:type rdf:resource="usto:enseignants"/>
        <foaf:familyname>Abid</foaf:familyname >
        <foaf:givenname>Samira</foaf:familyname >
        <foaf:mbox rdf:resource=" samira.abid@univ-usto.dz "/>
        <local:enseigne-dans>
                <rdf:Description  rdf:about=" https://www.univ-usto.dz/faculte/fac-
        mathinfo/Mathématiques  ">
                        <rdf:type rdf:resource="usto:département"/>
                        <dc:title> Département Mathématiques</dc:title>
                </rdf:Description>
        </local:enseigne-dans>
        <local:cours-enseignés>
        <rdf:bag>
                <rdf:_1 > Analyse </rdf:_1>
                <rdf:_2 > Algèbre </rdf:_2>
        </rdf:bag>
        </local:cours-enseignés>
</rdf:Description>
</rdf:RDF>
```

Solution 3

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns=http://purl.org/rss/1.0/
        xmlns:dc="http://purl.org/dc/elements/1.1/#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/#"
        xmlns:local="http://www.usto.dz/monvocabulaire#"
        xmlns:usto="http://www.usto.dz/dptmaths#">
<usto:enseignants  rdf:about=" http://www.usto.dz/dptmaths#samiraabid ">
        <foaf:familyname>Abid</foaf:familyname >
        <foaf:givenname>Samira</foaf:familyname >
        <foaf:mbox rdf:resource=" samira.abid@univ-usto.dz "/>
        <local:enseigne-dans>
                <usto:département  rdf:about=" https://www.univ-usto.dz/faculte/fac-
        mathinfo/Mathématiques  ">
                        <dc:title> Département Mathématiques</dc:title>
                </usto:département>
        </local:enseigne-dans>
        <local:cours-enseignés>
        <rdf:bag>
                <rdf:_1 > Analyse </rdf:_1>
                <rdf:_2> Algèbre </rdf:_2>
        </rdf:bag>
        </local:cours-enseignés>
</usto:enseignants>
</rdf:RDF>
```

3. NOTATION N3

Solution 1

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc:<http://purl.org/dc/elements/1.1/#>.
@prefix foaf:<http://xmlns.com/foaf/0.1/#>.
@prefix local:<http://www.usto.dz/monvocabulaire#>.
@prefix usto:< usto="http://www.usto.dz/dptmaths#>.
Usto:samiraabid   a   usto:enseignants;
        foaf:familyname  "Abid"^^ xsd:string;
        foaf:givenname  "Samira"^^xsd:string;
        foaf:mbox  <samira.abid@univ-usto.dz> ;
        local:enseigne-dans <https://www.univ-usto.dz/faculte/fac-
mathinfo/Mathématiques >;
        local:cours-enseignés
                [a rdf:bag;
                rdf:_1  "Analyse"^^xsd:string;
                rdf:_2  " Algèbre"^^xsd:string].

<https://www.univ-usto.dz/faculte/fac-mathinfo/Mathématiques >
dc:title  "Département Mathématiques"^^xsd:string.
```

Solution 2

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc:<http://purl.org/dc/elements/1.1/#>.
@prefix foaf:<http://xmlns.com/foaf/0.1/#>.
@prefix local:<http://www.usto.dz/monvocabulaire#>.
@prefix usto:< usto="http://www.usto.dz/dptmaths#>.
Usto:samiraabid   a   usto:enseignants;
        foaf:familyname  "Abid"^^ xsd:string;
        foaf:givenname  "Samira"^^xsd:string;
        foaf:mbox  <samira.abid@univ-usto.dz> ;
        local:enseigne-dans <https://www.univ-usto.dz/faculte/fac-
mathinfo/Mathématiques >;
        local:cours-enseignés   _:x.
                _:x      a rdf:bag;
                         rdf:_1  "Analyse"^^xsd:string;
                         rdf:_2  " Algèbre"^^xsd:string.
<https://www.univ-usto.dz/faculte/fac-mathinfo/Mathématiques >
dc:title  "Département Mathématiques"^^xsd:string.
```

6.2 RDF Schéma

6.2.1 Exercice1

6.2.1.1 Enoncé

Définir un vocabulaire RDFS (document RDFSchema et le graphe RDFS) puis un document RDF/XML (sans utiliser rdf :Description) modélisant les informations décrites dans les assertions suivantes : L'Algérie est un pays qui se situe en Afrique. Elle a des frontières avec d'autres pays tels que Le Maroc et la Tunisie. L'Algérie contient plusieurs villes telles que Alger, Oran et Mostaganem. Alger est sa capitale. Sa superficie est de 2,382 millions km². La Tunisie est aussi un pays de l'Afrique dont la capitale est Tunis.

Remarque : Il faut privilégier l'utilisation des URIs pour les ressources et l'utilisation du prédicat a-pour-nom pour les lier à leurs noms. Pour toutes les classes et les propriétés définies, il faut utiliser le préfixe local.

6.2.1.2 Corrigé

Document RDFS

```
<rdfs:Class rdf:ID="Pays"/>
<rdfs:Class rdf:ID="Continent"/>
<rdfs:Class rdf:ID="Ville"/>
<rdfs:Class rdf:ID="Capitale">
<rdfs:subClassof rdf:resource="#Ville"/>
</rdfs:class>

<rdf:Property rdf:ID="se-situe">
        <rdfs:domain rdf:resource="#Pays"/>
        <rdfs:range rdf:resource="#Continent "/>
</rdf:Property>

<rdf:Property rdf:ID="avoir-frontières">
        <rdfs:domain rdf:resource="#Pays"/>
        <rdfs:range rdf:resource="#Pays "/>
</rdf:Property>

<rdf:Property rdf:ID="contient-villes">
        <rdfs:domain rdf:resource="#Pays"/>
        <rdfs:range rdf:resource="#Ville"/>
</rdf:Property>

<rdf:Property rdf:ID="a-pour-capitale">
        <rdfs:domain rdf:resource="#Pays"/>
        <rdfs:range rdf:resource="#Capitale "/>
</rdf:Property>
```

```
<rdf:Property rdf:ID="superficie">
        <rdfs:domain rdf:resource="#Pays"/>
        <rdfs:range rdf:resource="xsd:Decimal"/>
</rdf:Property>
<rdf:Property rdf:ID="a-pour-nom">
        <rdfs:domain rdf:resource="#Pays"/>
        <rdfs:domain rdf:resource="#Ville"/>
        <rdfs:domain rdf:resource="#Continent"/>
        <rdfs:range rdf:resource="xsd:String"/>
</rdf:Property>
<local:Pays  rdf:about=" #Algérie ">
        <local:se-situe rdf:resource="#Afrique/>
        <local:nom>Algérie</local:nom>
        <local:superficie>2382000  </local:superficie>
        <local:a-pour-capitale  rdf:resource="#Alger">
        <local:contient-villes >
        <rdf:bag>
                <rdf:li  rdf:resource="#Alger"/>
                <rdf:li  rdf:resource="#Oran"/>
                <rdf:li  rdf:resource="#Mostaganem"/>
        </rdf:bag>
        </local:contient-villes>
        <local:avoir-frontières >
        <rdf:bag>
                <rdf:li  rdf:resource="#Maroc"/>
                <rdf:li  rdf:resource="#Tunisie"/>
        </rdf:bag>
        </local:avoir-frontières>
</local:pays>
```

Document RDF

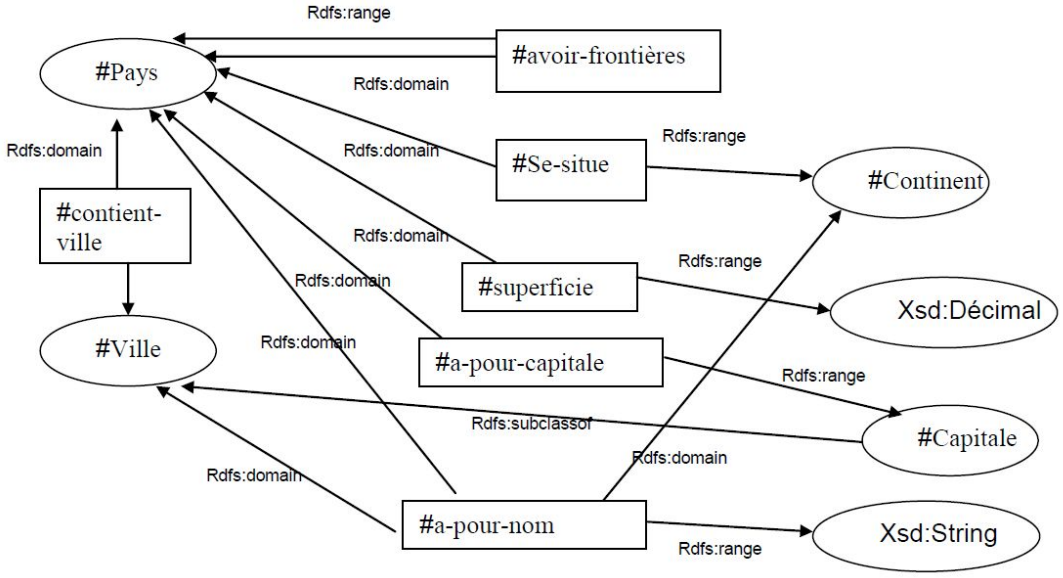
```
<local:Pays  rdf:about=" #Tunisie ">
        <local:se-situe rdf:resource="#Afrique"/>
        <local:nom>Tunisie</local:nom>
        <local:a-pour-capitale  rdf:resource="#Tunis">
</local:pays>

<local:Capitale  rdf:about=" #Alger ">
        <local:a-pour-nom>Alger</local:a-pour-nom>
</local:Capitale>
<local:Pays  rdf:about=" #Maroc ">
        <local:a-pour-nom>Maroc</local:a-pour-nom>
</local:pays>
<local:Ville  rdf:about=" #Oran ">
        <local:a-pour-nom>Oran</local:a-pour-nom>
</local:Ville>


<local:Ville  rdf:about=" #Mostaganem ">
        <local:a-pour-nom>Mostaganem</local:a-pour-nom>
</local:Ville>

</rdf:RDF>
```

Graphe RDFS sans partie RDF



6.2.2 Exercice 2

6.2.2.1 Enoncé

1) Donnez le document RDF/XML qui décrit le contenu du texte suivant en utilisant rdf :Description.

La ressource dont l'URI est http ://www.example.org/music#ABBA est un groupe de musique fondé à "Stockholm" en 1972. Le groupe se nomme "ABBA", il est composé de 4 chanteurs (qui sont des personnes) cités ci-dessous par ordre alphabétique :

— http ://www.example.org/music#Agnetha Fältskog

— http ://www.example.org/music#Anni-FridLyngstad

— http ://www.example.org/music#BennyAndersson

— http ://www.example.org/music#BjörnUlvaeus.

Le genre musical de ce groupe est soit "Pop" ou "Pop Rock" ou "Disco". Ses ventes sont estimées à 380 millions de disques. Ce groupe a chanté la chanson http ://www.example.org/music#Waterloo dont le titre est "Waterloo" et qui est sortie le 04 Mars 1974. Des informations sur ce groupe sont aussi disponibles sur freebase :ABBA et sur Dbpedia :ABBA.

2) Traduire le document obtenu en notation N3 et ajouter à la fin les triplets (N3) qui expriment la phrase suivante :

Le site Wiképedia dont l'URI est https ://fr.wikipedia.org/wiki/ABBA nous apprend que le groupe ABBA a obtenu "le prix de L'Eurovision de la chanson ".

6.2.2.2 Corrigé

3.Document RDF sous la forme RDF/XML

```
<rdf:RDF
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#"
xmlns:dc="http://purl.org/dc/elements/1.1/#"
xmlns:foaf="http://xmlns.com/foaf/0.1/#"
xmlns:local="http://www.usto.dz/monvocabulaire#"
xmlns:ms="http://www.example.org/music#"
xmlns:freebase=" http://rdf.freebase.com/ns#"
xmlns: dbpedia="http://dbpedia.org/ontology#">
<rdf:Description rdf:about=" http://www.example.org/music#ABBA ">
        <rdf:type rdf:resource="local:groupe-musique"/>
        <local:fondé-à>Stockholm</local:fondé-à>
        <local:fondé-en>1972</local:fondé-en>
        <local:a-pour-nom>ABBA</local:a-pour-nom>
        <local:nombre-chanteurs>4</local:nombre-chanteurs>
        <local:est-composé-de>
        <rdf:seq>
                <rdf:li rdf:resource=" http://www.example.org/music#Agnetha Fältskog "/>
                <rdf:li rdf:resource=" http://www.example.org/music#Anni-FridLyngstad "/>
                <rdf:li rdf:resource=" http://www.example.org/music#BennyAndersson "/>
                <rdf:li rdf:resource=" http://www.example.org/music#BjömUlvaeus"/>
        </rdf:seq>
        </local:est-composé-de>
        <local:genre-musical>
        <rdf:alt>
                <rdf:li > Pop</rdf:li >
                <rdf:li >Pop Rock </rdf:li >
                <rdf:li >Disco </rdf:li >
        </rdf:alt>
        </local:genre-musical>
        <local:nbre-de-ventes>380000000</local:nbre-de-ventes>
        <local:a-chanté rdf:resource=" http://www.example.org/music#Waterloo/>
        <rdfs:seeAlso rdf:resource=" http://rdf.freebase.com/ns#"/>
        <rdfs:seeAlso rdf:resource=" http://dbpedia.org/ontology#"/>
</rdf:Description>
<rdf:Description rdf:about=" http://www.example.org/music#Waterloo">
        <rdf:type rdf:resource="local:chanson"/>
        <local:titre> Waterloo </local:titre>
        <local:sortie-le> 04/03/1974 </local:sortie-le>
</rdf:Description>
</rdf:RDF>
```

2) Notation N3

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/#>.
@prefix local: <http://www.usto.dz/monvocabulaire#> .
@prefix ms: <http://www.example.org/music#> .
@prefix freebase: <http://rdf.freebase.com/ns#> .
@prefix dbpedia: <http://dbpedia.org/ontology#>.
ms:ABBA rdf:type local:groupe-musique ;
    local:a-chante ms:Waterloo ;
    local:fonde-a "Stockholm"^^xsd:string ;
    local:a-pour-nom "ABBA"^^xsd:string ;
    local:fonde-en "1972"^^xsd:gYear ;
    local:compose-de _:x;
    local:genre-musical _:y;
    local:nombrechanteurs "4"^^xsd:positiveInteger;
    local:ventes "380000000"^^xsd:positiveInteger ;
    rdfs:seeAlso <https://dbpedia.org/page/ABBA>;
    rdfs:seeAlso <http://rdf.freebase.com/ns>.
_:x rdf:type rdf:seq;
        rdf:_1 ms:AgnetaFaltskog;
        rdf:_2 ms:Annifridlyngstad;
        rdf:_3 ms:bennyanderson;
        rdf:_4 ms:bjornulvaeus.
_:y rdf:type rdf:alt;
        rdf:_1 "Disco"^^xsd:string ;
        rdf:_2 "Pop"^^xsd:string ;
        rdf:_3 "Pop Rock"^^xsd:string.
ms:Waterloo rdf:type local:chanson;
       local:sortie-le "1974-03-04"^^xsd:date ;
       local:titre "Waterloo"^^xsd:string .

https://fr.wikipedia.org/wiki/ABBA local:nous-apprend-que _:z.
_:z rdf:type rdf:statement.
_:z rdf:subject ms:ABBA.
_:z rdf:predicate local:a-obtenu.
_:z rdf:object "Prix de l'Eurovision de la chanson".
```

6.3 SPARQL

6.3.1 Exercice

6.3.1.1 Enoncé

Soit la base de connaissances suivante :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix local: <http://local.org/> .

local:p1    foaf:name "Karim" ;
            foaf:age "17"^^xsd:nonNegativeInteger ;
            foaf:mbox "Karim@abc.com" .
local:p2    foaf:name "Farah" ;
            foaf:age "27"^^xsd:nonNegativeInteger ;
            foaf:mbox "Farah@abc.com" .
local:p3    foaf:name "Ibtissem" ;
            foaf:age "21"^^xsd:nonNegativeInteger ;
            foaf:mbox "Ibtissem@uuu.com" .
local:p4    foaf:name "Salim" ;
            foaf:age "19"^^xsd:nonNegativeInteger ;
            foaf:mbox "Salim@uuu.com" .
local:p5    foaf:name "Ali" ;
            foaf:age "20"^^xsd:nonNegativeInteger ;
            foaf:mbox "Ali@xyz.ca" .
local:p6    foaf:name "Reda" ;
            foaf:age "30"^^xsd:nonNegativeInteger ;
            foaf:mbox "Reda@xyz.ca" .
local:p1    foaf:knows    local:p2 .
local:p1    foaf:knows    local:p3 .
local:p2    foaf:knows    local:p1 .
local:p2    foaf:knows    local:p3 .
```

Exprimez en SPARQL les requêtes suivantes :

1. Les URI et les noms de toutes les personnes dans la base de connaissances.
2. Les noms des personnes qui ont une adresse mail dans le domaine "abc.com"
3. Les noms de personnes qui connaissent des personnes âgées de plus de 20 ans.
4. Le nom et l'URI des personnes qui connaissent Karim.
5. Extraire le nom des personnes qui ont plus de 18 ans et moins de 30 ans, qui connaissent quelqu'un, et dont le courriel n'est pas dans le domaine "uuu.com".
6. Y a-t-il une personne qui connait Salim.
7. Pour toutes les personnes de la base, ajoutez foaf :homepage avec en objet la même chaîne de caractère utilisée avec foaf :mbox.

6.3.1.2 Corrigé

Requête 1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?uri ?nom
                WHERE { ?uri  foaf:name ?nom }
```

Requête 2

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?nom ?courriel
                WHERE { ?x foaf:name ?nom.
?x local:mbox ?courriel.
FILTER regex(?courriel, "abc.com", "i")
}
```

Requête 3

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT distinct ?nom
                WHERE { ?x foaf:name ?nom.
?x foaf:knows ?y.
?y foaf:age  ?z.
FILTER (?z>=20)
}
```

Requête 4

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?uri  ?nom
                WHERE { ?uri foaf:name ?nom.
?uri foaf:knows ?y.
?y foaf:name  "Karim".

}
```

Requête 5

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT   distinct ?nom
                 WHERE { ?x foaf:name ?nom.
?x  foaf:age ?y.
?x  foaf:knows ?z.
?x  local:mbox  ?courriel.
FILTER ( (?y >18) && (?y<30) && !regex("?courriel","uuu.com","i"))

}
```

Requête 6

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
ASK
Where {
?x  foaf:knows ?z .
?z foaf:name "Karim".
}
```

Requête 7

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX local:<http://www.semanticweb.org/firstone/ontologies/2021/2/untitled-ontology-120#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
Construct
 {
?x local:homepage ?z .
}

Where {
?x local:mbox ?z .

}
```

6.4 OWL

6.4.1 Exercice

6.4.1.1 Enoncé

Exprimez en utilisant OWL les classes et les propriétés suivantes :

— La classe Animal.

— La classe Plante qui est une classe disjointe de la classe Animal

— La classe Arbre qui est une sous classe de Plante.

— La classe Branche qui est une partie de l'arbre.

— La classe Feuille qui est une partie de la branche.

— La classe basilic qui est une plante.

— La propriété mange entre animal et plante ou animal.

— La classe des Herbivores qui sont des animaux qui ne mangent que des plantes, elle est disjointe de la classe des carnivores.

— La classe des carnivores qui sont des animaux qui ne mangent que des animaux.

— Un animal a un âge de type positiveInteger.

— La classe des girafes qui sont des herbivores qui ne mangent que des feuilles.

— La classe des lions qui sont des carnivores qui mangent des herbivores.

— La classe des herbivores basilicéens qui ne mange que du basilic.

6.4.1.2 Corrigé

• **La classe Animal.**

<owl :Class rdf :ID="Animal"/>

•**La classe Plante qui est une classe disjointe de la classe Animal.**

<owl :Class rdf :ID="Plante">
<owl :disjointwith rdf :resource= "# Animal/>
</owl :Class>

• **La classe Arbre qui est une sous classe de Plante.**

<owl :Class rdf :ID="Arbre">
<rdfs :subClassOf rdf :resource="#Plante"/>
</owl :Class>

• **La classe Branche qui est une partie de l'arbre.**

<owl :Class rdf :ID="Branche"/>
<owl :ObjectProperty rdf :about="partie-de">
<rdfs :domain rdf :resource="#Branche"/>
<rdfs :range rdf :resource="#Arbre"/>
</owl :ObjectProperty>

• **La classe Feuille qui est une partie de la branche.**

<owl :Classrdf :ID="Feuille"/>
<owl :ObjectProperty rdf :about="partie-de1">
<rdfs :domain rdf :resource="#Feuille"/>
<rdfs :range rdf :resource="#Branche"/>
</owl :ObjectProperty>

• **La classe basilic qui est une plante.**

<owl :Class rdf :ID="Basilic">
<rdfs :subClassOf rdf :resource="#Plante"/>
</owl :Class>

• **La propriété mange entre animal et plante ou animal.**

<owl :ObjectProperty rdf :about="mange">
<rdfs :domain rdf :resource="#Animal"/>
<rdfs :range rdf :resource="#Animal"/>
<rdfs :range rdf :resource="#Plante"/>
</owl :ObjectProperty>

• La classe des Herbivores qui sont des animaux qui ne mangent que des plantes, elle est disjointe de la classe des carnivores.

```
<owl :Class rdf :ID="Herbivore">
    <rdfs :subClassOf rdf :ressource="#Animal"/>
    <owl :disjointWith rdf :ressource="#Carnivore"/>
    <rdfs :subClassOf>
    <owl :Restriction>
    <owl :onProperty rdf :resource="Mange"/>
    <owl :AllValuesFrom rdf :resource="#Plante"/>
    </owl :Restriction>
    </rdfs :subClassOf>
</owl :Class>
```

• La classe des carnivores qui sont des animaux qui ne mangent que des animaux.

```
<owl :Class rdf :about="#Carnivore">
    <rdfs :subClassOf rdf :ressource="#Animal"/>
    <rdfs :subClassOf>
    <owl :Restriction>
    <owl :onProperty rdf :resource="Mange"/>
    <owl :someValuesFrom rdf :resource="#Animal"/>
    </owl :Restriction>
    </rdfs :subClassOf>
</owl :Class>
```

• Un animal a un âge de type positiveInteger.

```
<owl :DataTypeProperty rdf :Id="age">
    <rdfs :domain rdf :resource="#Animal"/>
    <rdfs :range rdf :resource="xsd :positiveinteger"/>
</owl :ObjectProperty>
```

• La classe des girafes qui sont des herbivores qui ne mangent que des feuilles.

```
<owl :Classrdf :ID="Girafe">
    <rdfs :subClassOf rdf :ressource="#Herbivore"/>
    <rdfs :subClassOf>
    <owl :Restriction>
    <owl :onProperty rdf :resource="Mange"/>
    <owl :AllValuesFrom rdf :resource="#feuille"/>
    </owl :Restriction>
```

```
    </rdfs :subClassOf>
</owl :Class>
```

• **La classe des lions qui sont des carnivores qui mangent des herbivores.**

```
<owl :Class rdf :ID="Lions">
    <rdfs :subClassOf rdf :ressource="#Carnivores"/>
    <rdfs :subClassOf>
    <owl :Restriction>
    <owl :onProperty rdf :resource="Mange"/>
    <owl :AllValuesFrom
    rdf :resource="#Herbivores"/>
    </owl :Restriction>
    </rdfs :subClassOf>
</owl :Class>
```

• **La classe des herbivores basilicéens qui ne mange que du basilic.**

```
<owl :Class rdf :ID=" basilicéens ">
    <rdfs :subClassOf rdf :ressource="#Herbivores"/>
    <rdfs :subClassOf>
    <owl :Restriction>
    <owl :onProperty rdf :resource="Mange"/>
    <owl :hasvalue rdf :resource="#Basilic"/>
    </owl :Restriction>
    </rdfs :subClassOf>
</owl :Class>
```

7 Bibliographie

[1] ANTONIOU, GRIGORIS AND VAN HARMELEN, FRANK,*A semantic web primer* , MIT press, 2004.

[2] BACHIMONT B.,*Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en Ingénierie des connaissances*,Rapport Interne Institut National de l'Audiovisuel, 1998.

[3] BERNARD ESPINASSE,*Introduction au Web Sémantique*, Aix-Marseille Université, Septembre 2019.

[4] BERNARD ESPINASSE,*Introduction au langage OWL (Ontology Web Language)*, Aix-Marseille Université, Septembre 2019.

[5] BERNERS-LEE TIM, HENDLER JAMES, LASILLA ORA,*The Semantic Web*,Scientific American, 2001.

[6] CHARLET J., BACHIMONT B., BRUNIE V., EL KASSAR S., ZWEIGENBAUM P., BOISVIEUX J.,*Hospitexte : towards a document-based hypertextual electronic medical record*, In Proc. AMIA Symp., 1998.

[7] D. GENEST,*Cours : web Sémantique*,Université d'Angers, 2008.

[8] DAVID MARTIN, MASSIMO PAOLUCCI, SHEILA MCILRAITH, MARK BURSTEIN, DREW MCDERMOTT, DEBORAH MCGUINNESS, BIJAN PARSIA, TERRY PAYNE, MARTA SABOU, MONIKA SOLANKI, NAVEEN SRINIVASAN, AND KATIA SYCARA, *Bringing Semantics to Web Services : The OWL-S Approach*,International Workshop on Semantic Web Services and Web Process Composition, 26–42, 2004.

[9] DUCHARME, BOB,*Learning SPARQL : querying and updating with SPARQL 1.1* , " O'Reilly Media, Inc.", 2013.

[10] E. LUCZAK,*A Guide to the Semantic Web*, eading Edge Forum Technology Grant, 2004.

[11] FABIEN GANDON, CATHERINE FARON-ZUCKER, OLIVIER CORBY,*Le web sémantique Comment lier les données et les schémas sur le web ?*, DUNOD, 2012.

[12] G. LAPALME,*Tutoriel protégé : Création d'une petite ontologie*, Université de Montreal.

[13] GRUBER T.,*Towards principles for the design of ontologies used for knowledge sharing*,International Journal of Human-Computer Studies, 43(5/6), 907-928, 1995.

[14] GUARINO N., GIARETTA P.,*Ontologies and knowledge bases, towards a terminological clarification*,In MARS N. I. P. Towards very large knowledge bases : knowledge building and knowledge sharing, 1995.

[15] HITZLER, PASCAL AND KRÖTZSCH, MARKUS AND PARSIA, BIJAN AND PATEL-SCHNEIDER, PETER F AND RUDOLPH, SEBASTIAN AND OTHERS,*OWL 2 web ontology language primer* , W3C recommendation, 2009.

[16] JULIEN PLU,*Introduction au Web sémantique*, 29 décembre 2015.

[17] KIM, HONG-GEE,*Semantic Web*, 2003.

[18] MATTHEWS, BRIAN,*Semantic web technologies*, Journal of E-learning,volume 6, page 8, 2005 .

[19] M. GAGNON,*Cours : Introduction au web sémantique*, Ecole Polytechnique de Montréal, 2007.

[20] M. GAGNON,*Cours : Brève introduction à RDF*.

[21] OWL WORKING GROUP AND OTHERS,*OWL 2 Web Ontology Language Document Overview : W3C Recommendation 27 October 2009* , 2009.

[22] S. GARLATTI,*Cours :Méta-données et annotations dans le Web sémantique*, ENST, Paris, 2006.

[23] STUDER, RUDI AND GRIMM, STEPHAN AND ABECKER, ANDREAS,*Semantic web services*,Springer, 2007.

[24] SZULMAN, SYLVIE AND BIÉBOW, BRIGITTE AND AUSSENAC-GILLES, NATHALIE, *Structuration de terminologies à l'aide d'outils de TAL avec TERMINAE*,Revue Traitement Automatique des Langues, volume 43, 103–128, 2002.

[25] SZULMAN, SYLVIE,*Une nouvelle version de l'outil Terminae de construction de ressources termino-ontologiques* , IC 2011. 22èmes Journées francophones d'Ingénierie des Connaissances, 2011.

[26] T. B. PASSIN,*Explorer's guide to the Semantic Web*, Manning Ed., 2008.

[27] XAVIER LACOT,*cours : Introduction à OWL, un langage XML d'ontologies Web*, Juin 2005.

[28] ,*https ://www.techno-science.net/definition/1450.html*.

[29] STANFORD UNIVERSITY,*https ://protege.stanford.edu/*.

[30] W3C, *https ://www.w3.org/TR/rdf11-concepts/*.

[31] W3C,*https ://www.w3.org/TR/rdf-schema/*.

[32] W3C,*https ://www.w3.org/TR/owl-features/*.

[33] W3C,*https ://www.w3.org/TR/sparql11-query/*.