

# LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES



## Sémantisation de jeux de données

Marjorie PETIT  
3<sup>ème</sup> année de licence Informatique  
Année universitaire 2017-2018  
Université de Nantes

*Maître de stage :*  
Patricia SERRANO ALVARADO  
*Tuteur :*  
Christophe JERMANN

Stage du 16 Avril 2018 — 8 Juin 2018

Remerciements à toute l'équipe et particulièrement à :

Patricia Serrano Alvarado,  
mon maître de stage qui m'a acceptée en stage et qui a su me partager sa passion et son enthousiasme pour ce sujet de recherche ;

Emmanuel Desmontils,  
qui a tenu le même rôle tout en partageant son bureau et ses connaissances sur le web sémantique ;

Benjamin Moreau,  
qui effectue sa thèse au sein de l'équipe et qui a su m'accompagner quand j'en ai eu besoin ;

Matthieu Perrin,  
qui a pris de son temps pour relire ce rapport et répondre à mes interrogations durant le stage ;

Christophe Jermann,  
mon tuteur, pour son suivi tout au long de ce stage ;

Chanez Amri,  
qui a effectué son stage au même titre que le mien et avec qui j'ai partagé cette expérience nouvelle du monde de la recherche.

Enfin, je remercie tout l'équipe GDD et l'ensemble du laboratoire pour avoir partagé leurs savoirs, leur bonne humeur et la réussite de mon intégration dans cette expérience très enrichissante aussi bien sur le plan professionnel que sur le plan humain.

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Laboratoire des Sciences du Numérique de Nantes (LS2N)</b>	<b>5</b>
2.1	Les différents pôles . . . . .	5
2.2	Équipe Gestion des Données Distribuées (GDD) . . . . .	5
<b>3</b>	<b>Contexte, objectifs et problématique</b>	<b>5</b>
3.1	Contexte et compréhension du sujet . . . . .	5
3.1.1	Les ontologies . . . . .	6
3.1.2	RDF, RDFS, OWL . . . . .	6
3.1.3	Introduction à SPARQL . . . . .	8
3.1.4	Le web des données . . . . .	8
3.2	La sémantisation des données à la volée . . . . .	9
3.2.1	Intérêt de la sémantisation . . . . .	9
3.2.2	Les solutions . . . . .	10
3.2.3	Problématique et missions . . . . .	10
<b>4</b>	<b>Travail réalisé</b>	<b>10</b>
4.1	Mise en place de la sémantisation . . . . .	11
4.1.1	Modèles conceptuels des jeux de données . . . . .	11
4.1.2	Recherche des ontologies . . . . .	13
4.1.3	<i>Mapping RML</i> . . . . .	16
4.2	Combinaison de jeux de données et requêtes fédérées . . . . .	18
4.2.1	Jeux de données compatibles . . . . .	19
4.2.2	Sémantisation des jeux de données trouvés . . . . .	19
4.2.3	Requêtes SPARQL fédérées . . . . .	21
4.3	Planning des tâches réalisées par semaine . . . . .	23
<b>5</b>	<b>Bilan</b>	<b>23</b>
<b>6</b>	<b>Conclusion et perspectives</b>	<b>24</b>
<b>7</b>	<b>Annexes</b>	<b>26</b>

# 1 Introduction

Dans le cadre de l'obtention de ma Licence Informatique, j'ai effectué un stage de huit semaines au sein du Laboratoire des Sciences du Numérique de Nantes. J'avais la possibilité de le faire soit en entreprise soit en laboratoire de recherche. Ayant déjà travaillé en entreprise, je voulais découvrir le monde de la recherche et ce stage était aussi pour moi l'occasion d'apprendre un nouveau sujet tout en étant en immersion.

La thématique de ce stage est avant tout le web sémantique. C'est une discipline qui se situe entre plusieurs autres thèmes tels que l'Intelligence Artificielle ou l'Interaction Homme Machine (IHM). C'est pour cela que j'ai choisi ce sujet de stage, j'aimerais poursuivre mes études en master Sciences Cognitives, en me spécialisant dans l'IHM et notamment les notions d'ergonomie. C'est donc le sujet de recherche qui se rapprochait le plus de mon projet professionnel et qui m'attirait le plus.

Le web sémantique [1] se situe au-dessus du web actuel (en complément) et a été proposé pour la première fois en 1994 par le créateur du web Tim Berners-Lee. La FIGURE 1 est une représentation assez parlante qui est tirée de la première présentation du web sémantique. Le but est de faire ressortir les concepts stockés sur le web en donnant un sens aux données. Les informations peuvent désormais être comprises et interprétées à la fois par les humains (comme dans le web actuel) mais aussi par les machines. Ce qui offre de nouvelles possibilités, par exemple, celle de pouvoir lier les concepts entre eux et ainsi de créer un véritable réseau de connaissances : c'est le web des données [2].

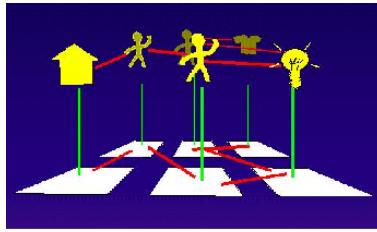


FIGURE 1 – diapositive de la présentation de Tim Berners-Lee à WWW94<sup>1</sup>

Aujourd'hui, peu d'applications utilisent le web sémantique car il n'y a pas assez de données sémantisées. À l'inverse, peu de personnes sémantisent leurs données car il n'y a pas assez d'applications qui les utilisent. Les jeux de données présents sur le web sont sémantisables et c'est peut-être une solution pour débloquer la situation.

Nous ne voulons pas dupliquer les données du web pour les intégrer au web sémantique. Il est donc important de trouver une solution pour pouvoir effectuer la transformation sans utiliser du stockage supplémentaire et ainsi limiter les frais de gestion qui en découleraient. C'est pourquoi, nous allons utiliser de la métainformation pour pouvoir compléter nos données et ainsi avoir la possibilité de sémantiser celles-ci au besoin, c'est-à-dire à la volée.

**Problématique :** Comment mettre en oeuvre la sémantisation à la volée d'un ou plusieurs jeux de données pour les intégrer dans le Web des données (*Linked Data*) ?

**Approche :** Pour répondre au mieux à la problématique, il a fallu travailler par étapes. La première était de comprendre le sens des noms des colonnes des jeux de données. Ensuite, il fallait trouver les bonnes ontologies et relier ces deux informations entre elles, c'est-à-dire effectuer un *mapping*.

Afin de mieux comprendre l'intérêt et le fonctionnement du web des données, il y a eu un travail de combinaisons de jeux de données et de réalisation de requêtes fédérées.

**Contribution :** À la fin de ce travail, les jeux de données sur lesquels j'ai travaillé ont désormais, dans leur métainformation, un *mapping*. Ceux-ci permettent de spécifier les concepts et les liens entre les données et ainsi de pouvoir les sémantiser seulement au besoin et sans contrainte de stockage.

1. Source : <https://www.w3.org/Talks/WWW94Tim/>

Pour commencer, nous allons voir une courte présentation du Laboratoire et de l'équipe dans laquelle j'ai évolué (Section 2). Nous allons ensuite poser le contexte précis, les missions ainsi que les objectifs de ce stage (Section 3). Par la suite, nous détaillerons les différents travaux réalisés durant ces quelques semaines (Section 4). Pour finir, nous dresserons le bilan (Section 5) de cette expérience, la conclusion et les perspectives possibles pour la suite (Section 6).

## 2 Laboratoire des Sciences du Numérique de Nantes (LS2N)

Le Laboratoire des Sciences du Numérique de Nantes (LS2N) est issu de la fusion de l'Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN) et du Laboratoire d'Informatique de Nantes Atlantique (LINA). Cette fusion, qui a eu lieu en janvier 2017, avait pour but de créer un seul et unique grand laboratoire de recherche interdisciplinaire (Robotique, données, calculs, systèmes, images...) sur la région nantaise. Le LS2N est une Unité Mixte de Recherche (UMR) soutenue par Université de Nantes (UN), l'Institut Mines-Télécom (IMT), l'Ecole Central de Nantes (ECN), le Centre National de la Recherche Scientifique (CNRS) et en collaboration avec l'INRIA (Institut National de Recherche en Informatique et en Automatique).

### 2.1 Les différents pôles

Le LS2N<sup>2</sup> est divisé en cinq pôles, ce qui permet de répartir l'ensemble des sujets de recherche et de rendre celles-ci les plus efficaces possibles.

**CSS :** Conception et Conduite de Systèmes

Étude des domaines de l'automatique et de l'automatisation, de la gestion de production, du génie industriel et de l'informatique embarquée.

**RPC :** Robotique, Procédés Calcul

Développement dans les domaines de l'action, de la perception, de la commande, dans les algorithmes de décision ou des interfaces logicielles.

**SDD :** Science des Données et de la Décision

Recherches sur les modèles de classification et les algorithmes de recherche.

**SIEL :** Signal, Image, Ergonomie, Langues

Travaux sur les signaux, images, sons, langues, écrits, mesures physiologiques et les facteurs humains.

**SLS :** Science du Logiciel et des Systèmes distribués

Recherches sur le domaine du génie de la programmation, du logiciel et des systèmes.

450 personnes réunies en une vingtaine d'équipes travaillent sur des sujets de recherche en lien avec ces pôles. Dans le cadre de mon stage, j'ai intégré l'équipe Gestion des Données Distribuées (GDD) qui travaille plus précisément en lien avec les pôles SDD et SLS.

### 2.2 Équipe Gestion des Données Distribuées (GDD)

L'équipe GDD<sup>3</sup> travaille en particulier sur les systèmes distribués fédérés. Les données distribuées sont des données qui sont réparties entre plusieurs machines. On dit qu'un système est fédéré quand il est partagé, accessible de manière collaborative pour un ensemble de personnes. GDD a pour objectif de proposer des modèles et des principes pour écrire des programmes fédérés sur des infrastructures fédérées en prenant en compte toutes les problématiques qui en découlent.

## 3 Contexte, objectifs et problématique

### 3.1 Contexte et compréhension du sujet

Pour s'imprégner du contexte et la compréhension du sujet, les références suivantes se sont avérées utiles :

---

2. Source : <https://www.ls2n.fr/>

3. Source : <https://www.ls2n.fr/equipe/gdd/>

- Les cours en ligne de la plate-forme Canal-u - Web sémantique et Web de données<sup>4</sup> : Fabien Gandon (INRIA).
- Les cours en ligne de la plate-forme OpenHPI - Linked Data Engineering<sup>5</sup> : Harald Sack.
- Le web sémantique, édition Dunod, de Fabien Gandon, Catherine Faron-Zucker et Olivier Corby.
- Les cours d'Emmanuel Desmontils sur le web sémantique, le RDF, le RDFS, le OWL, une introduction au langage de requêtes SPARQL et une autre sur le web des données.

Voici donc les grandes lignes de ces notions pour poser les bases du sujet.

### 3.1.1 Les ontologies

L'ontologie est une branche de la philosophie qui représente l'étude de l'être, au sens d'Aristote<sup>6</sup>, c'est "l'étude des propriétés générales de tout ce qui est".

En informatique, une ontologie est un ensemble de termes et de concepts structurés entre eux pour représenter le sens d'une donnée. Une ontologie contient des classes qui représentent des objets et qui sont ordonnées entre elles sous forme hiérarchique.

Liste 1 – Exemple de classes et de propriétés

<sup>1</sup>	Étudiant	prénom	"Pierre"
<sup>2</sup>	Étudiant	participe_à	Cours
<sup>3</sup>	Cours	de	"physique"

Dans l'exemple de la Liste 1, on peut retrouver la classe **Personne** ou bien la classe **Étudiant**. Hiérarchiquement, on peut dire qu'un étudiant est une personne, **Étudiant** est alors une sous-classe de **Personne**. Pour créer des liens entre les objets, on utilise des propriétés. Une propriété peut relier deux classes entre elles, les classes **Étudiant** et **Cours** sont reliées par la propriété **participe\_à**. Une propriété peut aussi permettre de relier une classe à un littéral comme une chaîne de caractères ou un entier.

Parmi les ontologies les plus connues, on peut citer **Dbpedia Ontology** [3] et **Schema** qui sont des ontologies qui couvrent beaucoup de concepts différents. Il faut mieux privilégier **Dbpedia** à **Schema** car **Schema** est une grosse ontologie mais très peu profonde : elle contient peu de relation de hiérarchie entre les classes. **Dbpedia** a beaucoup de données, est mieux organisée et maintenue par toute une communauté. Il est donc plus intéressant de mettre à jour les données ici et ainsi faire des liens avec les autres.

Il existe des ontologies plus précises sur un domaine, en voici deux exemples connus : l'ontologie **Foaf** destinée à représenter des personnes, leurs affiliations et leurs relations. L'ontologie **Dcterms**, qui permet de décrire des ressources (numériques ou physiques).

### 3.1.2 RDF, RDFS, OWL

#### Resource Description Framework

RDF<sup>7</sup> (*Resource Description Framework*) est le format de base du web sémantique destiné à décrire les données. Il permet de les représenter sous forme de graphes de données en décrivant ses arcs et ses sommets. Un document structuré en RDF se compose d'un ensemble de triplets RDF. Un triplet est constitué d'un sujet, d'une propriété et d'un objet.

- Sujet : c'est la ressource que l'on va décrire (le sommet d'origine de l'arc).
- Propriété (aussi appelée prédicat) : c'est ce qui va permettre de relier le sujet et l'objet selon le sens voulu de la description (le nom de l'arc).
- Objet : c'est la notion (donnée ou autre ressource) que l'on va relier à la ressource via la propriété (le sommet d'arrivée de l'arc).

Si l'on reprend nos exemples précédents de la Liste 1, on voit bien qu'ils sont sous forme de triplets avec :

- pour sujets : **Étudiant** lignes 1 et 2 ; **Cours** ligne 3 ;

4. [https://www.canal-u.tv/producteurs/inria/cours\\_en\\_ligne](https://www.canal-u.tv/producteurs/inria/cours_en_ligne)

5. <https://open.hpi.de/courses/semanticweb2016>

6. Source : [https://fr.wikipedia.org/wiki/Ontologie\\_\(informatique\)](https://fr.wikipedia.org/wiki/Ontologie_(informatique))

7. <https://www.w3.org/RDF/>

- pour propriétés : `prénom`, `participe_à`, `de` ;
- et pour objets : "Pierre", Cours à la ligne 2 et "physique".

Ces deux triplets des lignes 1 et 2 décrivent bien la ressource Étudiant.

### Internationalized Resource Identifier

Élément important pour la suite, nous utilisons des IRI (*Internationalized Resource Identifier*) ou des URI (*Uniform Resource Identifier*). Ce sont des identifiants qui vont permettre de définir de manière unique les sujets, les propriétés et les objets (sauf quand on a un littéral). Un URI va seulement être composé de caractères ASCII alors que l'IRI peut-être un ensemble de tous les caractères qui existent (accents, symboles japonais...). Les URI constituent un sous ensemble des IRI.

Définir un IRI pour identifier une ressource de manière unique est l'étape la plus importante de la sémantisation de données. C'est grâce à cet identifiant que l'on va pouvoir faire référence à cette ressource et ainsi l'intégrer et créer des liens dans le nuage de connaissances. Chaque chose possède un IRI : une place, un animal, une personne... et possède des liens avec d'autres choses.

Pour donner un exemple, une adresse URL est un IRI. Nos deux exemples précédents s'écrivent alors :

Liste 2 – Exemple d'IRI

1	<code>&lt;http://ex.org/Étudiant&gt;</code>	<code>&lt;http://ex.org/prénom&gt;</code>	"Pierre"
2	<code>&lt;http://ex.org/Étudiant&gt;</code>	<code>&lt;http://ex.org/participe_à&gt;</code>	<code>&lt;http://ex.org/Cours&gt;</code>
3	<code>&lt;http://ex.org/Court&gt;</code>	<code>&lt;http://ex.org/de&gt;</code>	"physique"

On peut simplifier l'écriture en définissant des préfixes. Pour notre exemple, nous pouvons définir le préfixe `ex` en écrivant `@prefix ex:<http://ex.org/>`. On pourra donc simplifier nos triplets de manière plus lisible comme sur la Liste 3 :

Liste 3 – Exemple de simplification avec les préfixes

1	<code>ex:Étudiant</code>	<code>ex:prénom</code>	"Pierre"
2	<code>ex:Étudiant</code>	<code>ex:participe_à</code>	<code>ex:Cours</code>
3	<code>ex:Cours</code>	<code>ex:de</code>	"physique"

### RDF Schema

RDFS, pour RDF Schema, est une extension de RDF qui permet de décrire le schéma d'une ontologie.

Liste 4 – Exemple pour RDFS

1	<code>ex:Étudiant</code>	<code>a</code>	<code>rdfs:Class</code>
2	<code>ex:Personne</code>	<code>rdfs:subClassOf</code>	<code>ex:Étudiant</code>
3	<code>ex:participe_à</code>	<code>a</code>	<code>rdf:Property</code>
4		<code>rdfs:domain</code>	<code>ex:Étudiant</code> ;
5		<code>rdfs:range</code>	<code>ex:Cours</code> .

Comme sur la Liste 4, on va ainsi pouvoir créer des classes avec `a` (ligne 1 : Étudiant est une classe), des relations entre elles (ligne 2 : Étudiant est une sous-classe de Personne) ou des relations entre propriétés avec `rdfs:subPropertyOf`.

On va pouvoir aussi spécifier pour une propriété la valeur attendue pour le sujet (l'origine de la propriété) en utilisant `rdfs:domain` et la valeur attendue pour l'objet (la cible) en utilisant `rdfs:range`. Les lignes 4 et 5 signifient que la propriété `participe_à` va former un arc qui va partir d'un sujet de classe Étudiant pour arriver à un objet de classe Cours.

### Web Ontology Language

On a vu que RDF servait à décrire les instances, et que RDFS permettait d'organiser les données les unes par rapport aux autres. Son pouvoir descriptif est néanmoins assez faible. En effet, nous ne pouvons pas indiquer si une propriété est symétrique, équivalente à une autre, transitive... OWL (*Web Ontology*

*Language*) est une extension de RDFS et permet d'aller plus loin dans la description d'une ontologie. OWL permet de spécifier des notions plus poussées comme celles de classes équivalentes (`owl:equivalentClass`), de propriétés équivalentes (`owl:equivalentProperty`), d'égalité entre deux ressources (`owl:sameAs`), de différences (`owl:differentFrom`), de contraire, de symétrie...  
Ainsi, avec `owl:sameAs`, on peut faire des liens entre des ressources si celles-ci sont représentées par plusieurs IRI alors qu'elles représentent la même chose.

#### Liste 5 – Exemple pour OWL

```
1 dbo:Person  owl:equivalentClass  schema:Person
```

Par exemple, dans la Liste 5, la classe Personne de Dbpedia est équivalent à la classe Personne de Schema.

Mais si la description est trop complexe, on peut vite se retrouver avec des traitements vraiment longs voire interminables, il faut donc faire attention et savoir rester moins expressif pour être plus rapide.

En effet, quand on commence à raisonner, on pourrait ne jamais terminer. On peut naviguer entre les ressources, les classes et les propriétés de façon éternelle. Chaque nouvelle information renvoyant vers une autre et ainsi de suite...

### 3.1.3 Introduction à SPARQL

SPARQL [4] (*Simple Protocol and RDF Query Language*) est un langage de requêtes. C'est-à-dire un langage permettant d'interroger une base de données pour en extraire l'information souhaitée. Il va permettre de faire une requête directement sur le sens des données (pas sur des mots-clefs) et de parcourir les informations contenues dans le web des données. SPARQL va utiliser des "Triple Pattern" (TP) qui sont des triplets RDF pouvant être constitués de une, deux ou trois variables parmi : ?sujet ?prédicat ?objet (?s ?p ?o). Une requête SPARQL est donc constituée d'une suite de combinaison de TP. Nous en verrons des exemples dans la Section 4.2.3 page 21.

### 3.1.4 Le web des données

Le web des données ou *Linked Data* est une initiative découlant du web sémantique et qui a pour but de favoriser la publication de données de façon structurée pour qu'elles puissent former un véritable réseau de connaissances au dessus du web actuel. Le rêve de Tim Berners-Lee serait que toutes les données présentes sur le web actuel puissent être sémantisées, reliées entre elles et accessibles pour que tout le monde puisse les utiliser<sup>8</sup>.

Pour cela, il instaure les quatre piliers suivants :

1. Utiliser des adresses URI pour l'identification unique de chaque ressource.
2. Utiliser des adresses URI HTTP qui existent sur le Web. Cela permet de lier une ressource à sa page web et donc à ses informations.
3. Fournir à travers l'URI des renseignements lisibles par les humains et par les machines. C'est à dire que c'est encore mieux si les données qui sont présentes sur la page web de la ressource sont déjà sémantisées.
4. Ajouter des URI externes aux données pour la découverte d'informations liées. Une ressource va alors être liée à plusieurs autres ressources et ainsi créer un réseau global.

La FIGURE 2 représente le nuage formé par toutes les données actuelles du web sémantique. Ce graphe est régulièrement mis à jour, ainsi, on peut vraiment voir le réseau énorme de connaissances qui se développe au fil des mois. Chaque petit noeud représente un jeu de données au format RDF et publié selon les quatre principes précédents. On peut noter toutes les relations qu'ils ont entre eux par la multitude d'arcs qui vont former des ponts entre les ressources et ainsi permettre la navigation dans ce graphe. Ces relations peuvent être par exemple celles vues dans la Section 3.1.2 sur OWL.

Les couleurs indiquent le thème de chaque jeu de données, cela permet de bien visualiser dans quels domaines le web sémantique est le plus développé. On peut notamment remarquer un gros réseau dans la discipline "Life sciences" en bas en rose.

---

8. Source : <http://www.w3.org/DesignIssues/LinkedData.html>

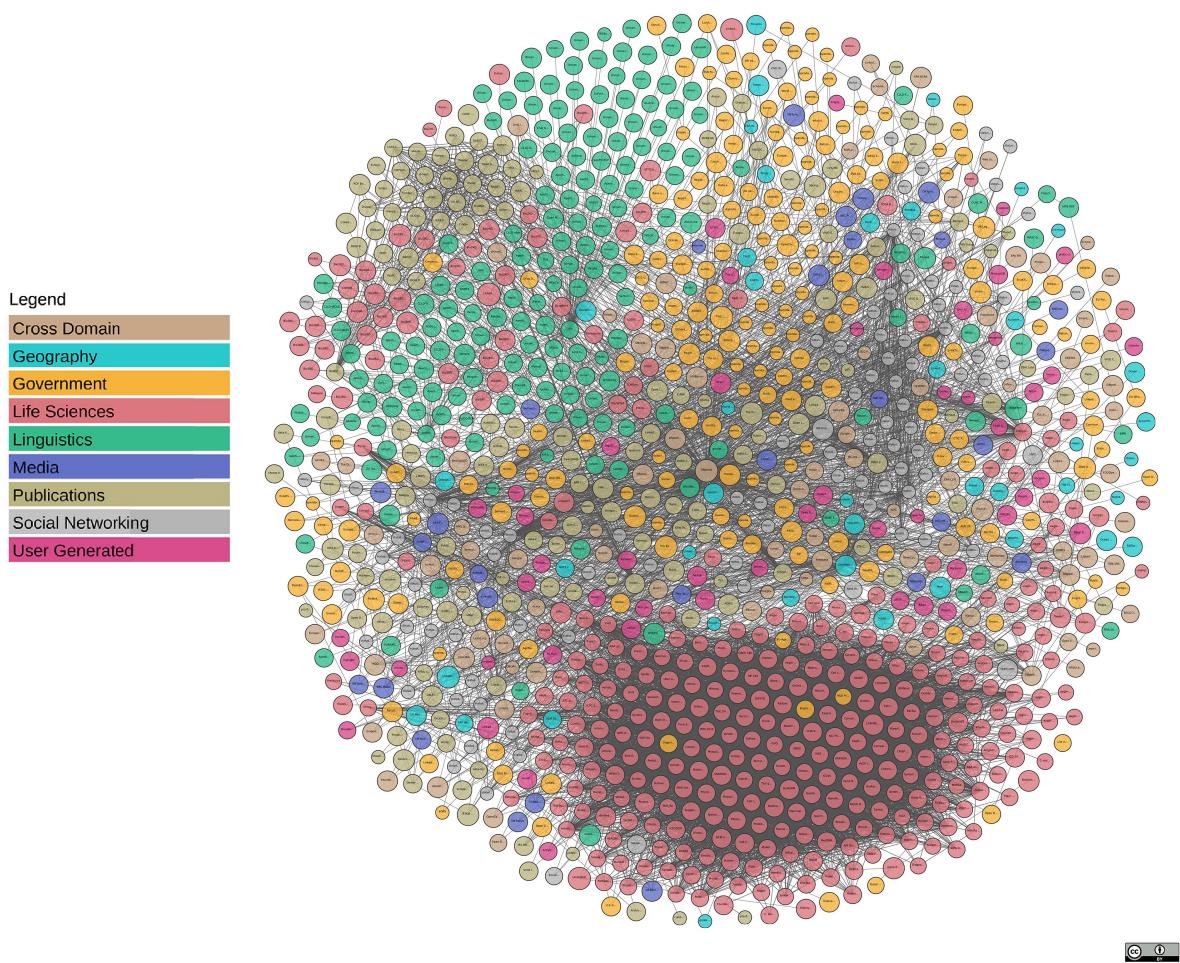


FIGURE 2 – Le nuage du web des données en 2018 sur <https://lod-cloud.net/>

### 3.2 La sémantisation des données à la volée

Rappelons que sémantiser veut dire donner un sens à quelque chose, attribuer une signification. Sémantiser des données à la volée signifie que l'on va sémantiser seulement les données que l'on veut au moment où on le veut. Nous allons voir l'intérêt et les solutions de la sémantisation et aborder la sémantisation à la volée.

#### 3.2.1 Intérêt de la sémantisation

Pendant ce stage, j'ai travaillé sur des données d'OpenDataSoft, cette société a pour but de publier des données en *opendata*. L'*opendata* consiste à ouvrir les données pour qu'elles soient accessibles à tous. Ils vont donc s'occuper de récupérer et partager des données publiques et des données d'entreprises ou d'organisations qui souhaitent rejoindre l'*opendata*.

Récemment, OpenDataSoft a commencé à s'intéresser aux technologies du web sémantique. Les utiliser leur permettrait d'avoir une meilleure structure et description de leurs données. La plate-forme pourra ainsi récolter des informations complémentaires sur les données, par exemple, savoir si un jeu de données contient la notion de personne, de ville ou de voiture.

Pour l'instant leurs données sont stockées sous forme de jeux de données en format tabulaire. Un jeu de données ou *dataset* est un ensemble de valeurs structurées entre elles. Mais il n'y a aucun lien entre les tables et les différents serveurs, on ne peut donc pas faire de requêtes communes. Sémantiser les données permettrait de les lier entre elles et donc de créer un véritable réseau de connaissances à la place de structure de données isolées les unes des autres.

### 3.2.2 Les solutions

Pour avoir leurs données sous forme sémantisée, il existe plusieurs solutions :

- La première serait de remplacer leurs bases de données orientées table directement par des bases de données orientées graphe. En plus du fait que cette solution ne soit pas valable économiquement, le service serait moins performant (moins rapide, moins optimisable) car on n'utilise pas d'index par exemple.
- Une autre solution serait de répliquer les données pour les avoir dans les deux formats. Mais ceci doublerait les coûts de stockage pour l'entreprise, ce qui n'est pas possible.
- Enfin, la solution qui semble la mieux adaptée serait la suivante : récupérer seulement les données qui nous intéressent, les transformer en RDF au besoin, et récupérer les données sémantisées. Cette solution est un peu coûteuse en calcul de données mais permet de ne pas engendrer des coûts de stockage. En effet, à chaque fois, il va falloir retransformer à la demande les données en RDF. Cette méthode se nomme *On-Demand Mapping using Triple Patterns* [5].

Pour la troisième solution et celle qui nous intéresse pour ce stage, il faut bien comprendre que les données sont sémantisées au besoin et chez le client, pas sur le serveur. Le serveur se sert juste des *mappings* (voir Section 4.1.3) pour savoir quelles classes et quelles propriétés contiennent un jeu de données. Il peut ainsi créer un filtre pour trier les jeux de données de la plate-forme. Ceci est en train de se mettre en place et pas encore visible sur le site, car il n'y a, pour l'instant, pas assez de *mappings* pour que cela soit intéressant.

On voit donc que la sémantisation des données à la volée semble la plus pertinente. Elle permet à moindre coût de pouvoir récolter des informations sur un ensemble de jeux de données. Mon travail s'est principalement concentré sur une partie de la transformation des données tabulaires en données sémantisées.

### 3.2.3 Problématique et missions

Voici la problématique de ce stage redéfinie plus en détail, connaissant désormais le but du web sémantique et le principe du web des données : compte-tenu des énormes quantités d'information présentes sur le web actuel, comment faire pour contribuer au web des données et augmenter ce nouveau réseau de connaissances qui se met en place et ainsi profiter de ses avantages ?

Deux missions se sont définies pendant ce stage :

1. Comprendre les enjeux et les problématiques lors de la sémantisation de jeux de données.
2. Comprendre les enjeux et les problématiques de la combinaison de jeux de données à travers des requêtes fédérées.

Les activités et objectifs globaux prévus dès le début de mon stage étaient les suivants :

- Connaître le langage qui permet de décrire des ontologies.
- Se familiariser avec les ontologies les plus connues et les plus utilisées.
- Savoir décrire des données en RDF.
- Connaître des langages de *mapping* comme le RML.
- Implémenter le *mapping* d'un ou plusieurs jeux de données non sémantisés afin de les intégrer dans le Web des données.
- Vérifier l'intégration avec des requêtes du langage SPARQL.

## 4 Travail réalisé

Dans un premier temps (Section 4.1), nous allons voir la réalisation de la sémantisation de plusieurs jeux de données, étape après étape. Ensuite, une explication de la démarche réalisée pour effectuer des requêtes fédérées sur plusieurs sources de données sera faite dans la Section 4.2.

## 4.1 Mise en place de la sémantisation

La principale mission a été de travailler sur quatre jeux de données présents sur le site d'OpenDataSoft afin de les sémantiser. Pour cela, la démarche a été la suivante pour chaque jeu de données :

1. Comprendre le sens des noms des colonnes des jeux de données.
2. Effectuer un modèle EAP pour visualiser les données entre elles.
3. Trouver les bonnes ontologies, classes et propriétés.
4. Écrire le *mapping* correspondant.

### 4.1.1 Modèles conceptuels des jeux de données

#### Modèle Entité-Association-Propriété

Un modèle EAP<sup>9</sup> pour Entité-Association-Propriété permet de modéliser un système d'information, c'est-à-dire un ensemble organisé de ressources. Une propriété est une information élémentaire associée à une entité ou à une association. Une entité, sous la forme d'un encadré, est la représentation d'un objet décrit par des propriétés. C'est donc un ensemble de propriétés. Une association permet de relier deux entités entre elles selon un sens précis et unique. Chaque occurrence d'une entité possède un identifiant unique (qui peut être l'URI dans notre cas). Cet identifiant est appelé clé primaire et est représenté par une ou plusieurs propriétés soulignées.

La dernière notion importante pour ce modèle est celle de la cardinalité. Constituée de deux bornes, elle précise le minimum et le maximum du nombre de fois qu'une occurrence d'une entité participe aux occurrences de l'association. Exemple sur la FIGURE 3, pour l'association "publie" (tout en haut du schéma), un tribunal peut publier 0 ou plusieurs annonces (d'où le 0,n). Une annonce est forcément publiée par un tribunal (d'où le 1,1).

#### Les quatre jeux de données

Ces quatre jeux de données sont constitués d'informations concernant des annonces légales publiées au Bulletin Officiel des Annonces Civiles et Commerciales (BODACC). Au vu du thème des jeux de données, nous retrouvons donc des éléments concernant des greffes des tribunaux de commerce, des actes enregistrés au Registre du Commerce et des Sociétés (RCS), des ventes et cessions, des immatriculations, des créations d'établissements, des modifications et radiations de personnes physiques ou morales...

Voici ces jeux de données qui ont été produit par la Direction de l'Information Légale et Administrative (DILA) et qui sont sous la licence ouverte Etalab<sup>10</sup> :

*Jeu de données 1 : Immatriculations au registre du commerce et des sociétés.*

Ce jeu de données regroupe des actes d'immatriculation, de création et de ventes d'établissement. Il possède 77 colonnes (degré) et 3 167 828 lignes (cardinalité).

*Jeu de données 2 : Modifications et radiations d'établissements au registre du commerce et des sociétés.*

Liste des annonces publiées au journal officiel qui concernent des radiations au Registre du Commerce et des Sociétés (RCS). On y trouve 27 colonnes et 6 051 835 lignes.

*Jeu de données 3 : Dépôt des comptes de sociétés.*

On y trouve les annonces publiées au journal officiel qui concernent des dépôts de comptes annuels, de rapports de sociétés. Ce jeu de données à un degré de 26 et une cardinalité de 11 305 836.

*Jeu de données 4 : Procédures Collectives.*

Celui-ci regroupe les annonces publiées au journal officiel qui concernent des jugement de personnes physiques ou de personnes morales. Il possède un degré de 37 et une cardinalité de 1 962 671.

#### Démarche

La première étape après avoir lu les jeux de données à sémantiser a été de réaliser un schéma de chacun d'entre eux afin de mieux visualiser les liens entre chaque colonne. En visualisant mieux les données, il est plus facile de savoir quelles ontologies chercher, quels concepts utiliser et comment les relier entre eux.

9. Source : cours de L2 de Mr Desmontils

10. [https://www.etalab.gouv.fr/wp-content/uploads/2014/05/Licence\\_Ouverte.pdf](https://www.etalab.gouv.fr/wp-content/uploads/2014/05/Licence_Ouverte.pdf)

Afin de réaliser un modèle EAP, on va suivre la démarche suivante :

1. Analyser les noms des colonnes.

Pour mieux comprendre le sens de chaque nom, on peut s'aider des valeurs présentes dans les colonnes ainsi que leur type (text, decimal...).

2. Analyser les instances.

L'étape consiste à trouver des liens entre les colonnes. Pour cela, il faut regarder une ligne précise dans le jeu de données et essayer de voir quelles autres colonnes ne sont pas vides.

3. Grouper les colonnes en entités.

Après avoir trouvé le sens de chaque champ, un regroupement des noms de colonnes par entité est réalisé. Ensuite, on nomme chacune d'entre-elle avec un nom qui se rapproche le plus du sens trouvé du concept en question.

4. Lier les entités avec des associations.

Grâce aux informations récoltées à l'étape 2, il est possible d'indiquer les cardinalités facilement sur notre modèle.

### Exemple

Nous allons maintenant voir en exemple le modèle réalisé pour le jeu de données 2 : Modifications et radiations d'établissements au registre du commerce et des sociétés (FIGURE 3). Chaque propriété présente dans les entités correspond à un titre de colonne dans le jeu de données d'OpenDataSoft.

Les 3 autres modèles EAP réalisés sont disponibles dans les **ANNEXES A, B et C**.

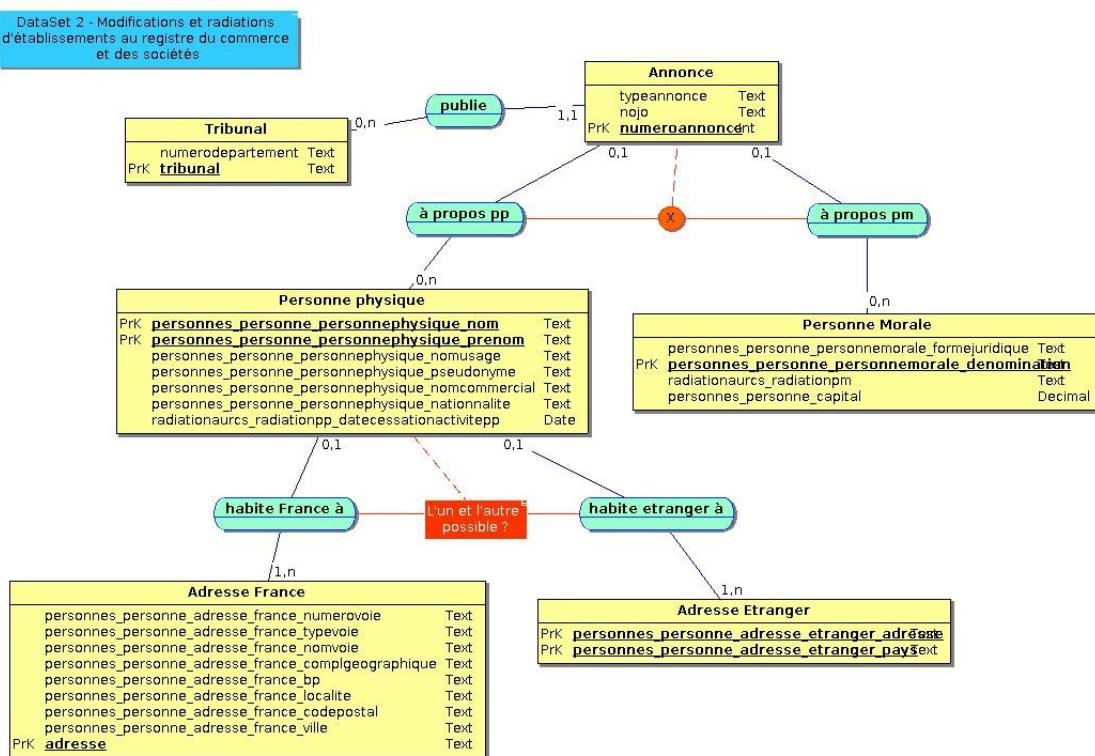


FIGURE 3 – Modèle EAP du jeu de données 2

Voici brièvement un raisonnement pour montrer comment les entités ont été liées entre-elles : à chaque fois qu'il y a une information dans les colonnes **personnes\_personne\_personnephysique\_nom** et **prenom**, la colonne **personnes\_personne\_adresse\_france\_[...]** n'est jamais vide. Nous pouvons donc en conclure que les deux entités auxquelles appartiennent ces propriétés sont liées. À contrario, quand il y a une information dans la colonne **personnes\_personne\_personnemorale\_denomination**, les colonnes commençant par **personnes\_personne\_adresse\_france\_[...]** sont toujours vides. Donc on peut savoir qu'il n'y a pas de

lien entre les entités correspondantes.

Les noms d'associations ont été indiqués de façon très génériques mais cela permet avant tout de visualiser les données.

## Difficultés rencontrées

Ce travail de modélisation a fait ressortir quelques difficultés. Pour commencer, certaines colonnes étaient entièrement vides. Par exemple, toutes les colonnes concernant l'adresse étrangère. Il a fallu les relier par logique et déduction sans certitude que cela soit exact. Pour notre exemple, seule l'entité personne physique possède une adresse française, relier l'adresse étrangère à cette même entité était le choix le plus pertinent. Seulement, on ne peut pas savoir s'il est possible pour une personne d'avoir une adresse française ainsi qu'une adresse étrangère : une note a été laissée à ce sujet sur le modèle.

De plus, la plupart des jeux de données ne sont pas assez documentés. Pour certains, des informations concernant chaque colonne sont spécifiées. Malheureusement, pour les quatre jeux de données du BODACC, ce n'était pas le cas. Il fallait donc déduire le sens des colonnes comme indiqué plus haut. Il est compréhensible que ces informations ne soient pas indiquées pour tous les jeux de données d'OpenDataSoft, cela demanderait trop de temps et de travail sachant la quantité énorme qu'ils ont de jeux de données.

Certaines informations comme les dates sont totalement fausses. On peut y trouver de l'an 1000 comme de l'an 5000 pour des dates de création d'entreprises. Cela n'est pas dû à OpenDataSoft mais aux personnes qui ont rempli la base de données de manière incorrecte. Je me suis interrogée sur la pertinence de sémantiser des données fausses, sachant toutefois que ces erreurs n'étaient pas de mon fait et qu'il n'est pas mon ressort de vérifier la cohérence des données. D'autant plus que mon rôle est de réaliser un *mapping* RML qui permettra de sémantiser les données à chaque fois que l'on en aura besoin. Cela veut dire que si les données sont corrigées en aval, le *mapping* s'appliquera quand même et sémantisera sans problème les données corrigées. J'ai donc continué les étapes de la sémantisation sans tenir compte de ces erreurs.

### 4.1.2 Recherche des ontologies

Il faut maintenant conceptualiser, donner un sens à nos entités et à leurs propriétés. C'est pourquoi la recherche d'ontologies est l'étape primordiale.

#### Démarche et discussion

La démarche pour trouver les ontologies utilisées a été la suivante :

1. Rechercher les ontologies sur LOV<sup>11</sup>.

Ce site les regroupe et permet de faire une recherche rapide sur toutes les ontologies existantes. On peut y faire des recherches par mots-clefs et celui-ci nous propose des classes ou des propriétés.

2. Consulter la documentation.

Pour chaque propriété ou classe proposée par LOV, il faut ensuite aller voir dans la documentation de l'ontologie à laquelle elle appartient pour trouver ce qu'elle représente vraiment et être sûr que celle-ci correspond au sens que l'on veut.

3. Pour les propriétés, il faut aussi bien faire attention au *domain* et au *range* indiqués.

Reprendons l'exemple de la Liste 4, si j'utilise la propriété **participe\_à** avec **Pierre** **participe\_à** **Physique**, il faut être sûre que Pierre est un **Étudiant** et que Physique est un **Cours**.

4. L'héritage est aussi à prendre en compte.

Par exemple, la classe **Étudiant** est sous-classe de **Personne** qui est elle-même sous-classe de **Thing**. Cela veut dire que l'on peut utiliser sur l'objet "Pierre" les propriétés issues de **Étudiant**, de **Personne** et de **Thing**. Il faut donc penser à regarder les propriétés des classes mères quand on ne trouve pas ce que l'on cherche tout de suite.

On ne peut donc pas utiliser n'importe quelles propriétés sur la classe que l'on veut. Il faut respecter la cohérence et les contraintes de chaque ontologie. Ceci complique la tâche, nombreuses sont les fois où l'on trouve des propriétés qui correspondent à ce que l'on cherche, mais qui ne sont pas utilisables pour cette raison.

---

11. <https://lov.okfn.org>

Il existe deux types d'ontologies, les ontologies "métiers" (comme eli, osp, insee) qui sont vraiment spécifiques pour un concept et les ontologies "générales" (Dbpedia ou Schema) qui couvrent un peu tous les sujets, mais de manière moins précise. Les générales sont beaucoup plus accessibles pour le grand public et permettent de couvrir une multitude de domaines, on est donc facilement tenté d'utiliser celle-ci quitte à être moins précis. Il y a un travail de génie des données pour savoir comment vont être utilisées nos données RDF par la suite. Si elles sont destinées à des professionnels comme des juristes par exemple, il sera plus adapté d'utiliser l'ontologie métier qui couvrira vraiment les termes techniques et qui sera très précise dans les descriptions. À l'inverse, si nos données sont destinées au grand public, il vaut mieux utiliser des concepts qui parlent à tout le monde sans terme technique spécifique à un type de population. On voit donc qu'il est important de réaliser son *mapping* en fonction de comment les données vont être utilisées et par qui; quitte à faire deux versions, une avec les ontologies métiers et une avec les générales.

### Exemple

Voici la liste, en FIGURE 4, de l'ensemble des ontologies utilisées lors de la sémantisation des jeux de données. Je propose une sémantisation pour chacun des jeux de données mais celles-ci auraient pu être réalisées d'une autre manière. Les ontologies choisies n'étaient pas les seules candidates et chaque choix dépend de la personne et de sa perception.

Prefixes	Ontologie	IRI
osp	Ontologie du Service Public	<a href="http://data.lirmm.fr/ontologies/osp#">http://data.lirmm.fr/ontologies/osp#</a>
eli	The European Legislation Identifier	<a href="http://data.europa.eu/eli/ontology#">http://data.europa.eu/eli/ontology#</a>
foaf	Friend of a Friend vocabulary	<a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a>
odsbodacc	Extention Dbpedia pour ODS	<a href="http://dbpedia.org/ontology-ext/ods/bodacc/">http://dbpedia.org/ontology-ext/ods/bodacc/</a>
dbo	The DBpedia Ontology	<a href="http://dbpedia.org/ontology/">http://dbpedia.org/ontology/</a>
schema	Schema.org vocabulary	<a href="http://schema.org/">http://schema.org/</a>
geo	WGS84 Geo Positioning	<a href="http://www.w3.org/2003/01/geo/wgs84_pos#">http://www.w3.org/2003/01/geo/wgs84_pos#</a>
bibo	The Bibliographic Ontology	<a href="http://purl.org/ontology/bibo/">http://purl.org/ontology/bibo/</a>
wlo	BBC Wildlife Ontology	<a href="http://purl.org/ontology/wo/">http://purl.org/ontology/wo/</a>
dcterm	DCMI Metadata Terms	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>

FIGURE 4 – Listes des ontologies utilisées

La FIGURE 5 permet de suivre l'exemple du dataset 2 et de mieux visualiser le travail de recherche d'ontologies. Pour cela, reprenons le schéma du jeu de données de la FIGURE 3 et rajoutons les classes et les propriétés utilisées pour la sémantisation que je propose. Le même type de schéma révisé pour deux autres jeux de données sémantisés sont disponibles dans les **ANNEXES D et E**.

Discussion autour de deux ontologies utilisées sur les ressources / entités de ce jeu de données :

**Ressource tribunal** : Pour représenter cette ressource, l'ontologie du service public (osp<sup>12</sup>) et notamment la classe `osp:Tribunal` correspondait parfaitement au sens recherché. De cette même ontologie, la propriété `ost:aArrondissement` a été utilisée pour définir le numéro du département et `osp:typeTribunal` pour indiquer si c'est un tribunal de commerce, d'instance ou autre.

J'aurais pu prendre l'ontologie Dbpedia et la classe `dbo:Organization` pour conceptualiser un tribunal mais celle-ci aurait été moins précise. De plus, pour cette classe, il n'y avait pas de propriété pertinente et surtout aussi proche du sens des colonnes "numerodepartement" et "tribunal".

**Ressource annonce** : L'ontologie eli<sup>13</sup> permet de définir les informations possibles autour de la publication de textes juridiques. Pour le concept d'annonce dans le sens d'une annonce légale publiée dans un journal officiel (JO), la classe `eli:LegalResource` rejoint totalement cette idée. La propriété `eli:number` a été utilisée pour les numéros de journaux officiels et les numéros d'annonces puisque la documentation indique : *"This can be the number of a legislation, the number of an article, or the issue number of an official journal"*.

12. <http://data.lirmm.fr/ontologies/osp>

13. [http://publications.europa.eu/mdr/eli/documentation/ELI\\_Ontology-v1.1.htm](http://publications.europa.eu/mdr/eli/documentation/ELI_Ontology-v1.1.htm)

Là encore, j'aurais pu faire un autre choix d'ontologie, de classe et de propriétés pour cette ressource. Par ailleurs, la classe `dbo:Document` de Dbpedia peut correspondre mais celle-ci est encore une fois moins précise. La propriété `dbo:documentNumber` aurait pu servir pour le "numeroannonce".

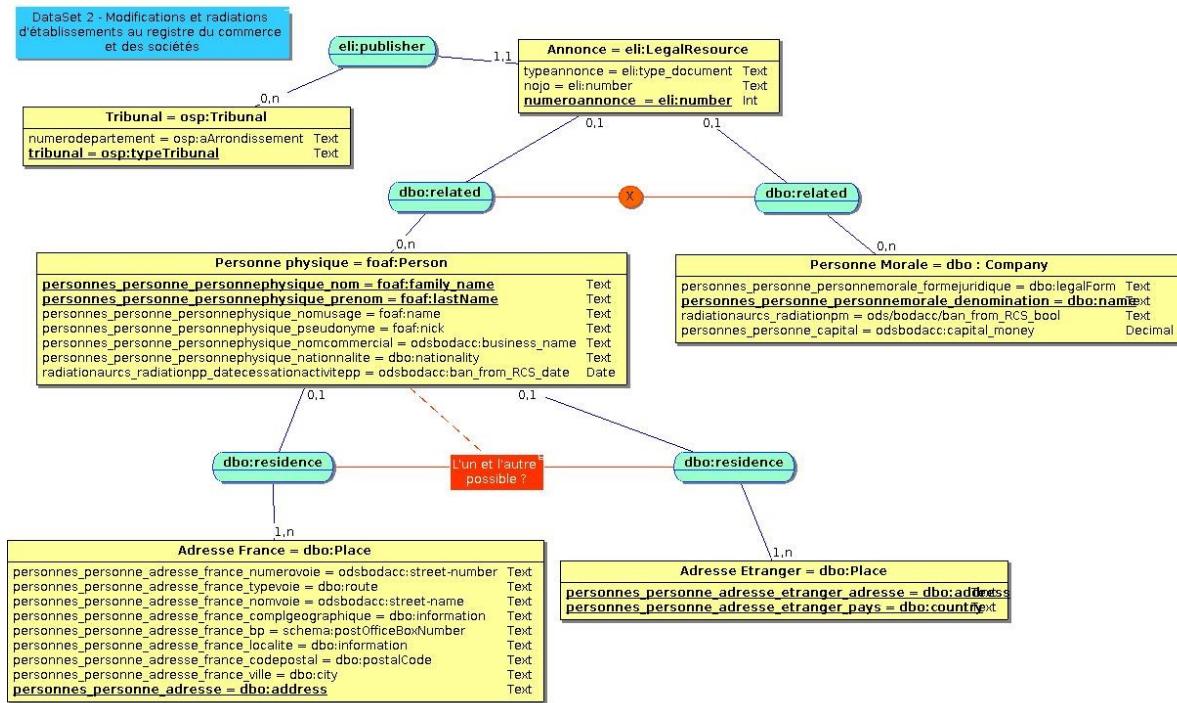


FIGURE 5 – Modèle du précédent jeu de données avec les propriétés et classes utilisées pour la sémantisation

### Difficultés rencontrées

À partir du milieu du stage, le site LOV est devenu inaccessible. De ce fait, il a fallu rechercher les ontologies à la main en parcourant les classes et les propriétés une par une des ontologies les plus connues (Dbpedia, foaf, Schema). Cela a demandé encore plus de temps qu'avec le site.

Conceptualiser l'adresse française a soulevé un problème. En effet, celle-ci est composée de plusieurs informations (noms de colonnes) : "adresse", "numerovoie", "typevoie", "nomvoie", "complgeographique", "bp", "localite", "codepostal", "ville".

Les données dans "numerovoie", "typevoie" et "nomvoie" se regroupent pour former "adresse" qui est au final l'adresse complète. Il fallait donc trouver une ontologie qui décrivait une adresse avec des propriétés très spécifiques d'une adresse française. Malgré le temps passé à parcourir les ontologies, aucune ne couvrait toutes ces informations et notamment le numéro et le nom de la voie.

Plusieurs solutions étaient possibles pour résoudre ce problème :

- Une solution aurait été de ne pas sémantiser les données présentes dans les colonnes "numerovoie", "typevoie" et "nomvoie" puisque l'on peut les retrouver dans "adresse". L'avantage de cette solution est que l'on utilisait une seule ontologie (Dbpedia) pour conceptualiser l'adresse. Seulement, si l'on choisit cette solution, il faut savoir que l'on ne pourra pas faire de recherche dans ce jeu de données sur ces informations-là.
- L'autre solution, qui a été celle choisie, est la suivante :
  1. Essayer de prendre une seule ontologie pour conceptualiser l'adresse.
  2. S'il nous manque une propriété, essayer de la trouver dans une autre ontologie.
  3. Enfin, si l'on ne trouve pas ce que l'on recherche, notre dernier recours est de créer la propriété ou la classe en la rajoutant à l'ontologie que l'on a le plus utilisé pour notre ressource et ainsi l'enrichir.

Pour cette solution, l'ontologie JUSO<sup>14</sup> couvre presque tout ce qui compose l'adresse à l'exception du type de voie et du complément géographique. Mais cette ontologie semble peu connue, peu utilisée et n'est pas entretenue (pas de mise à jour depuis sa création en 2015). L'idée a donc été abandonnée pour privilégier des ontologies plus connues et surtout qui possèdent un suivi.

## Mise en place de la solution

Dans les cas où les propriétés représentant le sens voulu ne semblait pas exister, j'ai dû les créer :

- En complétant l'ontologie de Dbpedia, car c'est celle qui a été le plus utilisée dans les quatre *mapping*.
- Quand elles concernaient quelque chose de très spécifique : le numéro de rue, des informations concernant le Registre du Commerce et des Sociétés (RCS), le nom commercial d'une personne...

Pour cela il a fallu créer un fichier pour y regrouper toutes les propriétés inventées. Dans la Liste 6, on retrouve un extrait suivi d'une explication :

Liste 6 – Exemple de code pour définir une propriété

```

1  @prefix : <http://dbpedia.org/ontology-ext/ods/bodacc/> .
2  @prefix owl: <http://www.w3.org/2002/07/owl#> .
3  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
5  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6  @prefix dbo: <http://dbpedia.org/ontology/> .

7
8  :ban_from_RCS_date a owl:DatatypeProperty ;
9    rdfs:label "date of the RCS ban"@en ;
10   rdfs:label "Date de radiation au RCS"@fr ;
11   rdfs:comment "Date of the ban from the Registre du Commerce et des Sociétés (RCS) in
France. For a physical person who has a small business (a shopkeeper or a farmer...
not a moral person)"@en ;
12   rdfs:comment "Date de radiation au Registre du Commerce et des Sociétés (RCS). Pour
une personne physique qui possède un commerce (boutique, agriculteur... pas une
personne physique)"@fr ;
13   rdfs:isDefinedBy <http://dbpedia.org/ontology-ext/ods/bodacc/> ;
14   rdfs:domain dbo:Person ;
15   rdfs:range xsd:date .

```

1. Tout d'abord, il a fallu définir un URI spécialement pour le projet. Celui-ci se trouve à la ligne 1 et servira pour définir les URI des futures propriétés. Le choix de reprendre le début de l'URI de Dbpedia en spécifiant `ext` est pour indiquer que ceci est une extension. Les mots `ods` et `bodacc` font référence aux jeux de données de ce thème et à OpenDataSoft.
2. De la ligne 2 à 6, on retrouve les préfixes qui servent à simplifier la clarté du code.
3. Ligne 8, on va définir un nom et un type, ici, c'est une propriété que l'on nomme `ban_from_RCS_date`.
4. En utilisant RDFS pour décrire l'ontologie, on lui donne un label et un commentaire (en anglais et en français).
5. Ligne 13, on dit quel va être le préfixe de notre URI pour cette propriété.
6. Puis, ligne 14 et 15, on définit le `domain` et le `range` comme vu précédemment.

### 4.1.3 Mapping RML

Pour chaque jeu de données, après avoir récupéré toutes les propriétés et les classes à utiliser, il fallait relier nos informations contenues dans celui-ci et nos ontologies au moyen d'un *mapping* RML.

#### Le *mapping*

Un "*mapping*" au sens informatique du terme veut tout simplement dire "mise en correspondance d'éléments".

---

14. <http://rdfs.co/juso/latest/html>

## Le langage RML

Le langage RML [6] (*RDF mapping language*) est utilisé, dans notre cas, pour récupérer les données d'un fichier Json et les transformer dans la syntaxe Turtle (*Terse RDF Triple Language RDF*). Cette syntaxe permet d'exprimer les données selon le modèle RDF (*Resource Description Framework*). Rappelons que RDF représente, quant à lui, les informations en utilisant des "triplets", dont chacun est constitué d'un sujet, d'un prédicat et d'un objet.

### Démarche

Nous allons définir une suite de ressources (ou règles) dans notre fichier de mapping et pour chacune d'entre elle, nous allons procéder comme suit :

1. Définir comment récupérer les données dans le fichier Json.
2. Définir la classe de la ressource.
3. Définir l'URI de la ressource.
4. Définir les propriétés pour faire le lien vers les données.
5. Définir les propriétés pour faire référence aux autres ressources.

### Exemple

Voici un extrait de code, dans la Liste 7, permettant de définir la ressource Annonce présente dans le jeu de données 2 :

Liste 7 – Exemple de code RML pour definir la ressource Annonce

```
1  <#Annonce>  rml:logicalSource [ rml:iterator "$.[*].fields" ;
2      # --- On utilise JSONPath pour lire un fichier json
3      rml:referenceFormulation ql:JSONPath ;
4      # --- On indique le fichier source qui contient les données
5      rml:source "modifications-et-radiations-detablissements-au-registre-du-
6      commerce-et-des-socie.json" ] ;
7
8      # --- Définition de la classe de la ressource
9      rr:subjectMap [ rr:class <http://data.europa.eu/eli/ontology#LegalResource> ;
10         # --- Definition de l URI de la ressource
11         rr:template "http://ods.dataset.com/Annonce/{numeroannonce}" ] ;
12
13      # --- Définition des propriétés de la ressource
14      rr:predicateObjectMap
15         [ rr:objectMap [ rml:reference "$.numeroannonce" ] ;
16             rr:predicate <http://data.europa.eu/eli/ontology#number> ,
17             [ rr:objectMap [ rml:reference "$.typeannonce" ] ;
18                 rr:predicate <http://data.europa.eu/eli/ontology#type_document> ] ,
19             [ rr:objectMap [ rml:reference "$.nojo" ] ;
20                 rr:predicate <http://data.europa.eu/eli/ontology#number> ] ,
21
22             # --- Référence à Tribunal
23             [ rr:objectMap [ rr:parentTriplesMap <#Tribunal> ] ;
24                 rr:predicate <http://data.europa.eu/eli/ontology#publisher_agent> ] .
```

1. Dans l'en-tête (lignes 1 à 5), nous indiquons comment récupérer les données dans notre fichier Json et quel est ce fichier.
2. Ligne 8, nous définissons la classe de la ressource. Ici, comme vu plus haut, il s'agit de la classe LegalResource de l'ontologie eli. On va utiliser l'URI de la classe après le prédicat **rr:class**.
3. Ligne 10 : Chaque annonce présente dans le jeu de données doit avoir son propre et unique URI, ainsi, on va utiliser un *template* défini comme suit : <http://ods.dataset.com/Annonce/{numeroannonce}>. On y retrouve une référence à OpenDataSoft, à Annonce et à l'id de la ressource (sa clef primaire). Nous aurions pu la définir d'une autre manière, il n'y a pas de standard. Si une même ressource existe déjà, il peut être intéressant de renvoyer vers l'URI existante.

4. À partir de la ligne 13, nous définissons les propriétés qui vont permettre de faire le lien vers les données qui sont dans le fichier Json. Pour cela, on indique le nom de la colonne après `rml:reference` et l'URI de la propriété associée après `rr:predicat`. On fait cela pour toutes nos propriétés.
5. Pour faire référence à une autre ressource (ce qui correspond aux associations sur le modèle EAP), on va simplement utiliser `rr:parentTriplesMap` à la place de `rml:reference` et on devra ensuite définir une nouvelle ressource avec le même nom dans notre *mapping*, ici, `<#Tribunal>`.

Liste 8 – Exemple de résultat au format RDF

```

1 <http://ods.dataset.com/Annonce/471> a <http://data.europa.eu/eli/ontology#LegalResource>
2 ;
3 <http://data.europa.eu/eli/ontology#number> "BXB08050004578K" , "471" ;
4 <http://data.europa.eu/eli/ontology#type_document> "Annonce" ;
5 <http://data.europa.eu/eli/ontology#publisher_agent> <http://ods.dataset.com/Tribunal
/GREFFE%20DU%20TRIBUNAL%20DE%20COMMERCE%20DE%20PERIGUEUX> .

```

Une fois le *mapping* terminé et le fichier Json téléchargé, le code est testé à l'aide de RML-Mapper. Celui-ci permet d'appliquer le *mapping* sur les données du fichier Json et renvoie un fichier RDF constitué de triples. On peut voir un exemple dans la Liste 8, nous retrouvons l'URI de notre Annonce suivis de l'URI de son type (sa classe). Ensuite, nous avons les URI des propriétés suivis par les éléments que nous avions dans nos colonnes. À la ligne 4, on voit bien la référence vers le tribunal concerné par cette annonce. Dans ce fichier résultat, tous les éléments du jeu de données sont au format RDF.

## Contributions

Toutes ces étapes ont été réalisées sur les quatre jeux de données du BODACC. Tous les mappings RML réalisés dans le cadre de ce stage sont disponibles sur github à l'adresse suivante : <https://github.com/MarjoriePetit/Mapping-RML>.

## 4.2 Combinaison de jeux de données et requêtes fédérées

La deuxième mission de ce stage a été de travailler sur la recherche de jeux de données qui puissent être recoupés entre eux en effectuant des requêtes fédérées.

Une requête est une commande qui va permettre de récupérer les éléments voulus dans une base de données. Une requête fédérée (FIGURE 6) est une requête qui va interroger non pas une source de données, mais plusieurs. L'utilisateur va envoyer sa requête à un moteur de requêtes fédérées et c'est celui-ci qui va interroger les différentes bases de données séparées. Mon travail consistait à trouver un exemple de requête fédérée SPARQL compréhensible par tous et si possible international.

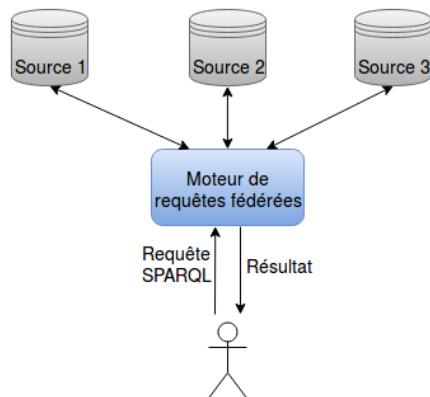


FIGURE 6 – Schéma du fonctionnement d'un moteur de requêtes fédérées

#### 4.2.1 Jeux de données compatibles

La première étape est donc de trouver plusieurs sources de données qui puissent être mises en lien. Pour cela, la démarche a été la suivante :

1. Parcourir le catalogue des jeux de données d'OpenDataSoft.

Regarder en détails ceux concernant des thèmes qui parlent à tout le monde (l'environnement, les transports...) et en mettre de côté.

2. Consulter les sites d'*opendata* de différents pays<sup>15</sup>.

Regarder les données concernant les mêmes thèmes que cités plus haut et voir, pour chaque jeu de données, s'il est possible de les recouper avec ceux préalablement trouvés.

#### Déficulté

Malheureusement, les quelques jeux de données trouvés concernant exactement les mêmes sujets étaient rares et pas pertinents (beaucoup de coordonnées GPS pour l'environnement par exemple). Dans l'impossibilité de les mettre en lien, l'idée de chercher les données sur ces sites-là et d'avoir un exemple international a été abandonnée.

#### Jeux de données trouvés

Voici la première proposition d'exemple qui a été trouvée où l'on peut recouper :

1. Un jeu de données d'OpenDataSoft : il s'agit d'une liste des sites inscrits au patrimoine mondial de l'UNESCO.
2. La base de données des ressources de Dbpedia [3] (Wikipédia du web sémantique).

Il existe une page pour presque chaque site sur Dbpedia, nous avons donc accès à beaucoup d'informations sur un site directement en sémantique (résumé, pays, années...).

Voici la deuxième proposition d'exemple qui a été trouvée où l'on peut recouper :

1. Un premier jeu de données d'OpenDataSoft (ODS1) : "Espèces protégées et réglementées - Référentiel". Il contient une liste d'espèces protégées et réglementées qui provient de l'INPN<sup>16</sup> (Inventaire National Du Patrimoine Naturel). Chaque espèce est associée à un niveau de protection.
2. Un deuxième jeu de données d'OpenDataSoft (ODS2) : "Espèces protégées et réglementées". Il regroupe toutes les informations autour d'un niveau de protection (de quel arrêté il est tiré, en quelle année, une URL vers l'arrêté en question, une URL vers le site de l'INPN...).
3. La base de données des ressources de Dbpedia (Wikipédia du web sémantique). En tapant le nom de l'espèce sur Dbpedia<sup>17</sup>, on peut récupérer des informations intéressantes comme le résumé de celle-ci, le type (fleur, animal...), le genre, la famille, l'ordre, la photo...

La deuxième proposition a été choisie car elle est plus visuelle et offre plus de possibilités de requêtes comme elle concerne trois sources de données.

#### 4.2.2 Sémantisation des jeux de données trouvés

Dans le but de pouvoir appliquer une requête interrogeant les 3 sources de données trouvées dans la dernière proposition, il fallait sémantiser les deux jeux de données ODS1 et ODS2.

#### Démarche

1. Le début du procédé a été le même que précédemment :

étudier le sens des colonnes et rechercher dans les ontologies quelles propriétés et classes correspondaient le mieux.

15. France : <https://www.data.gouv.fr>, Espagne : <http://datos.gob.es>, Belgique : <https://opendata.bruxelles.be>, Suisse : <https://www.europeandataportal.eu>

16. <https://inpn.mnhn.fr>

17. <http://dbpedia.org/page/Dianthus> par exemple

2. Étudier comment les jeux de données vont être reliés.

Le seul moyen de faire le lien entre ces 2 jeux de données est d'utiliser la colonne qu'ils ont en commun, ici, c'est le cd\_protection (c'est le sigle de la protection).

Il faut donc prêter attention à ce que la propriété de cd\_protection soit la même dans les deux *mappings* pour que l'on puisse bien faire le lien lors de la requête.

## Résultats

Voici deux schémas en FIGURE 7 et 8 qui représentent le *mapping* de ces deux jeux de données :

**Espèces protégées et réglementées - Référentiel**

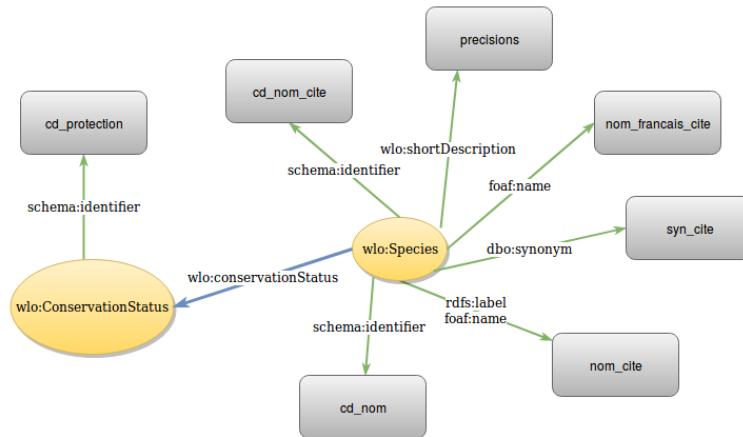


FIGURE 7 – Schéma représentant le *mapping* du jeu de données ODS1

**Espèces protégées et réglementées**

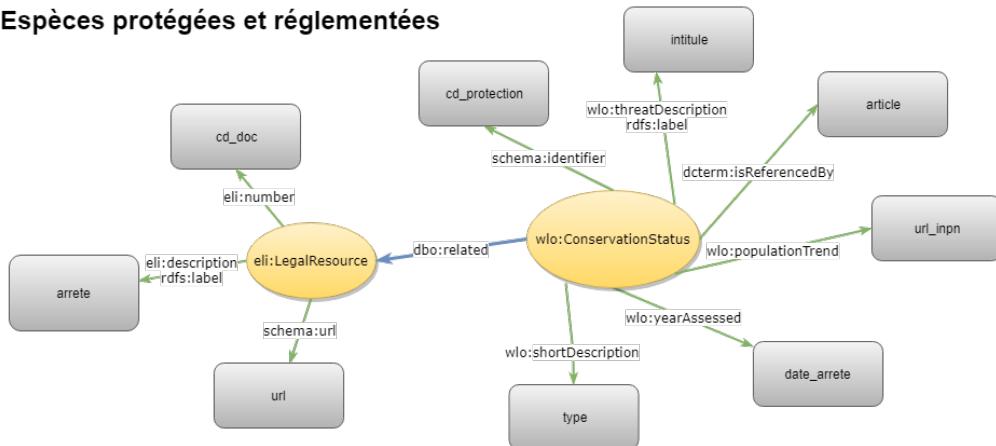


FIGURE 8 – Schéma représentant le *mapping* du jeu de données ODS2

Légende :

- En jaune sont indiquées les ressources avec les classes utilisées.
- Les rectangles gris représentent les données des colonnes du même nom.
- Les flèches vertes sont les propriétés utilisées pour décrire la ressource à laquelle elles sont reliées.
- Les flèches bleues quant à elles vont indiquer les propriétés qui permettent de relier deux ressources entre elles.

Ainsi pour le jeu de données ODS1 de la FIGURE 7, on retrouve deux ressources, **Species** qui représente les espèces et **ConservationStatus** pour le statut de protection de cette espèce. Elles sont liées par la propriété **conservationStatus**.

Pour le jeu de données ODS2 de la FIGURE 8, On a deux ressources pour représenter un article de loi (**LegalRessource**) et un statut de protection. Ce dernier possède des propriétés et des informations différentes du précédent jeu de données.

#### 4.2.3 Requêtes SPARQL fédérées

La dernière étape de cette mission a été de travailler sur les requêtes SPARQL afin de tester la combinaison des jeux de données précédents et de Dbpedia.

##### Démarche

1. Débuter en faisant des requêtes "star" sur un seul jeu de données.  
On les appelle ainsi car elles vont aller chercher toutes les informations autour d'un seul sujet.
2. Faire des requêtes fédérées en croisant deux jeux de données.
3. Tester une requête fédérée questionnant les trois sources de données.

Pour tester les requêtes, un site de requêteur<sup>18</sup> qui envoie des requêtes SPARQL sur plusieurs serveurs a été utilisé : Linked Data Fragments.

Une requête SPARQL va fonctionner en raisonnant avec un ensemble de "*Triple Pattern*" TP (?sujet ?propriété ?objet) :

- Un premier TP va être envoyé et un ensemble de réponses possibles va être retourné.
- Grâce à un deuxième TP, la liste des réponses candidates va être réduite pour correspondre à ce TP.
- L'opération va être répétée autant de fois qu'il y a de TP.

##### Exemples

Liste 9 – Requête SPARQL permettant de récupérer les photos des plantes de Dbpedia

```

1  SELECT ?s ?photo WHERE {
2      ?s dbo:kingdom dbr:Plant .
3      ?s foaf:depiction ?photo
4 }
```

La première requête effectuée, dans la Liste 9, va nous donner l'ensemble des URI des plantes recensées sur Dbpedia ainsi que leur photo :

- Ligne 1, on veut récupérer le sujet et sa photo (le nom des variables est précédé d'un point d'interrogation).
- Ici, la ligne 2 se lit : la variable s a pour rang (kingdom) Plante.
- La ligne suivante permet de récupérer l'objet (photo) de la propriété **depiction** qui possède comme sujet la variable s.

La requête va donc permettre de filtrer les plantes et de nous donner leur photo.

Liste 10 – Requête SPARQL permettant de récupérer toutes les informations du jeu de données ODS1

```

1  SELECT ?cd_nom ?nom_cite ?cd_protection ?syn_cite ?precisions ?cd_nom_cite WHERE {
2      ?s schema:identifier ?cd_nom .
3      ?s foaf:name ?nom_cite .
4      ?s wlo:conservationStatus ?cs .
5      ?cs schema:identifier ?cd_protection .
6      OPTIONAL { ?s dbo:synonym ?syn_cite } .
7      OPTIONAL { ?s wlo:shortDescription ?precisions } .
8      ?s schema:identifier ?cd_nom_cite
9 }
```

18. <http://client.linkeddatafragments.org/>

Ensuite, dans la Liste 10, une requête a été effectuée sur le jeu de données ODS1 "Espèces protégées et réglementées - Référentiel" afin de récupérer toutes les informations de toutes les colonnes :

- On va créer des variables pour chaque colonne (qui vont avoir le même nom que celles-ci pour simplifier) et on va les lui demander ligne 1.
- Les TP des lignes 2, 3, 6, 7 et 8 vont permettre de récupérer le `cd_nom`, `nom_cite`, `syn_cite`, `precisions` et `cd_nom_cite` d'un même sujet (de classe `wlo:Species`) en allant chercher les informations qu'il y a au bout des propriétés indiquées sur ces mêmes lignes.
- Les OPTIONAL présents lignes 6 et 7 permettent de renvoyer la valeur `null` si ces éléments n'existent pas pour un sujet. Si l'on ne l'indique pas, il ne nous renverra aucune autre valeur concernant ce sujet.
- Pour récupérer le `cd_protection` qui se situe, selon le *mapping* effectué, dans une autre ressource que `Species`, il faut utiliser le lien qui se trouve entre ces deux ressources, c'est-à-dire `wlo:conservationStatus`.

C'est ce qui est fait à la ligne 4 et l'URI de l'objet résultat va être stocké dans la variable "`cs`". Ligne suivante, on va chercher les objets qui ont pour sujet celui stocké dans `cs` et qui ont pour propriété `schema:identifier`.

Une autre façon d'expliquer la ligne 4 serait : "le sujet s a pour statut de conservation quelque chose" que l'on stocke dans la variable `cs`. Ligne 5 : "Ce quelque chose est identifié par une valeur" que l'on va pouvoir récupérer dans la variable `cd_protection`. Ainsi, au fur et à mesure des TP, on va égrainer les résultats et lier les sujets entre eux en fonction de nos demandes.

Une requête qui va récupérer toutes les informations concernant les sujets du jeu de données ODS2 "Espèces protégées et réglementées" a également été effectuée en suivant le même principe : chaque ligne et donc chaque TP va recouper les informations pour un même sujet s en fonction des propriétés de celui-ci.

Liste 11 – Requête fédérée SPARQL permettant de récupérer les plantes protégées et réglementées

```

1  SELECT ?cd_nom ?nom_cite ?cd_protection ?syn_cite ?precisions ?cd_nom_cite ?photo
2    WHERE {
3      ?s schema:identifier ?cd_nom .
4      ?s foaf:name ?nom_cite .
5      ?s wlo:conservationStatus ?cs .
6      ?cs schema:identifier ?cd_protection .
7      OPTIONAL { ?s dbo:synonym ?syn_cite } .
8      OPTIONAL { ?s wlo:shortDescription ?precisions } .
9      ?s schema:identifier ?cd_nom_cite .
10     ?s foaf:depiction ?photo .
11     ?s dbo:kingdom dbr:Plant
12   }
```

Voici, dans la Liste 11, une requête fédérée qui est finalement la concaténation des requêtes de la Liste 9 et de la Liste 10. Elle permet de récupérer toutes les plantes et leurs informations ainsi que leur photo.

À la ligne deux, la requête va finalement effectuer une jointure naturelle avec `foaf:name`. C'est-à-dire qu'elle va récupérer les sujets qui sont communs dans le jeu de données ODS1 et dans Dbpedia selon leur nom. En effet, `foaf:name` est une propriété qui est utilisée à la fois dans le *mapping* et sur la fiche de la plante sur Dbpedia.

## Difficultés rencontrées

Le consommateur de données s'appelle le client et le producteur de données le serveur. Ils sont indépendants l'un de l'autre et ne se connaissent pas. Le serveur ne peut pas connaître les données et les clients auxquels il va avoir à faire.

Aussi, nous avons essayé de faire une requête fédérée en utilisant nos trois sources de données : ODS1, ODS2 et Dbpedia. Malheureusement, le client avait beaucoup de mal à exécuter notre demande et faisait énormément de requêtes HTTP pour aller rechercher les informations. Cela se terminait en *timeout* au bout de quelques minutes, c'est-à-dire que les demandes HTTP expiraient sans avoir eu de retour et donc avant que l'on ait le moindre résultat. Il est probable que cette erreur soit due au fait que l'on demande des données sur trois sources différentes et que le client n'arrive pas à suivre. Il faudrait peut-être trouver ou créer un autre moteur de requêtes fédérées SPARQL et y tester notre requête.

## Contribution

Après avoir testé la requête de la Liste 11, les résultats obtenus par la requête fédérée se sont montrés cohérents et pourront être exploités par OpenDataSoft.

L'intérêt de cette requête fédérée est le suivant : en une seule requête, les résultats obtenus vont permettre de créer un nouveau jeu de données en enrichissant les données d'OpenDataSoft avec celles provenant de Dbpedia. Par exemple, on va pouvoir créer le jeu de données "Plantes protégées et réglementées" qui n'existe pas et il sera même possible de rajouter la photo pour chaque plante.

### 4.3 Planning des tâches réalisées par semaine

Tâches / semaines	semaine 1	semaine 2	semaine 3	semaine 4 (2jours)	semaine 5	semaine 6	semaine 7	semaine 8
Compréhension et apprentissage du sujet (web sémantique, RDF, RDFS, OWL, SPARQL, Linked data)								
Prise en main des sites (LOV, old.datahub.io, datahub.io, laundromat)								
Schématisation des datasets avec JMerise								
Recherche d'ontologies (de classes et de propriétés)								
Sémantisation, Mapping RML des 4 datasets du BODACC								
Recherche de datasets pour un exemple de requête fédéré								
Sémantisation des datasets Espèces protégées								
Requêtes SPARQL + requête fédéré + test								
Mapping RML de linkedin pour aider Chanez								
Rapport de stage								
Préparation de la soutenance								

FIGURE 9 – Planning des tâches réalisées par semaine

Voici un tableau représentatif de la répartition des tâches effectuées durant ce stage en FIGURE 9.

- Les deux premières semaines ont vraiment été axées sur la compréhension du sujet et la découverte des ressources utiles pour la suite, notamment sur la prise en main de différents sites d'annuaires de jeux de données et d'ontologies.
- La schématisation des jeux de données a pris un peu moins de deux semaines et c'était environ le temps que je m'étais fixé, car il y avait beaucoup de détails à parcourir pour pouvoir bien les comprendre et les représenter.
- La tâche dont le temps de réalisation a réellement été sous-estimé est la sémantisation des quatre jeux de données du BODACC. Je ne pensais pas que cela allait prendre autant de temps. En fait, écrire le *mapping* RML est très rapide, une journée environ. Le plus long est de trouver les classes et les propriétés que l'on va utiliser. Je pouvais passer des journées à chercher dans les ontologies pour finalement ne trouver qu'une classe et quelques propriétés.  
De plus, le site qui permettait de chercher de manière plus efficace des ontologies a commencé à ne plus être accessible vers le milieu du stage et cela a contribué à l'augmentation de la durée de cette tâche.
- À partir de la semaine 5, le début de la nouvelle mission m'a permis de chercher des jeux de données pour la requête fédérée et cela a été agréable de varier les tâches. La sémantisation des deux jeux de données sur les espèces protégées a été rapide et l'écriture ainsi que les tests des requêtes SPARQL ont été réalisés les semaines suivantes.

## 5 Bilan

La formation suivie à la faculté de Nantes m'a préparée à ce stage. Lors de la deuxième année de licence informatique, nous avons étudié les modèles EAP dans le cadre de l'UE "Introduction aux systèmes d'information". Connaître ainsi les bases du logiciel JMerise et de la modélisation a permis d'effectuer les modèles EAP plus rapidement. C'est également grâce à ces modèles que la recherche d'ontologies a été plus efficace.

Les tâches qui ont été définies ont été très intéressantes à effectuer et j'ai tout mis en oeuvre pour les réaliser le mieux possible. À la fin de ce stage, les *mappings* effectués seront disponibles sur github<sup>19</sup> et seront ajoutés aux meta-informations des datasets correspondants sur le site d'OpenDataSoft. Ainsi, ils pourront être utilisés pour filtrer les recherches sur le site quand cette option sera mise en place.

Avant de réaliser ce stage, je ne connaissais le web sémantique que par son nom et ne savais pas du tout de quoi il s'agissait. J'ai eu la chance de pouvoir travailler sur ce sujet et surtout d'aller plus loin que la théorie. Peu de personnes ont accès au web sémantique de nos jours, cela pour plusieurs raisons : ce n'est pas un sujet compliqué, mais il faut connaître les bases et prendre le temps de les apprendre pour pouvoir s'en servir. De plus, cela reste encore une discipline utilisée dans le monde de la recherche mais pas en entreprise, le grand public n'y est pas encore confronté directement. C'est une discipline qui est néanmoins en plein essor et qui se développe très vite.

## 6 Conclusion et perspectives

Afin de répondre à la problématique de départ qui était : comment mettre en oeuvre la sémantisation à la volée d'un ou plusieurs jeux de données pour les intégrer dans le Web des données, ce stage a été séparé en deux missions :

1. Comprendre les enjeux et les problématiques lors de la sémantisation de jeux de données.
2. Comprendre les enjeux et les problématiques de la combinaison de jeux de données à travers des requêtes fédérées.

### Mission 1

Pour mener à bien la première mission, la sémantisation de jeux de données a été effectuée et pour ce faire, il a fallu comprendre le sens des noms des colonnes des jeux de données, trouver les bonnes ontologies et réaliser un *mapping*.

Ce *mapping* permet d'accéder à la version sémantique de ces données et ainsi de contribuer à l'ajout de connaissances au réseau du web sémantique.

Le travail de sémantisation de jeux de données est très long à mettre en place et fastidieux. De même, réaliser un *mapping* RML pour tous les jeux de données qui existent n'est pas concevable sur le long terme.

Il est vrai que si l'on veut faire en sorte que plus de personnes accèdent au web sémantique et puissent contribuer à l'ajout de données, il faudrait trouver un moyen de simplifier et d'automatiser la tâche. En effet, si une personne ne connaît pas le web sémantique ou n'est pas un spécialiste en ingénierie des connaissances, ce travail est impensable. D'autant plus que l'on est vite confronté à un manque d'ontologies et de standardisation de celles-ci. À part LOV, qui a des problèmes de fonctionnement, il n'existe pas beaucoup de matériel pour faciliter la recherche et il y a un réel manque de catalogues d'ontologies.

On peut notamment citer le travail de Benjamin qui développe actuellement un chatbot pour essayer de palier un de ces problèmes. C'est un outil mis en place pour faciliter le *mapping* de données et permettre à des personnes de faire un premier *mapping* sans forcément connaître le web sémantique. Il va étudier le jeu de données et proposer des classes et des propriétés pour certaines colonnes. L'utilisateur va devoir répondre par oui, non, peut-être en fonction du sens proposé et du sens réel. Le chatbot peut trouver des classes qui ne correspondent pas aux données : c'est pour cela qu'il faut l'intervention d'un humain derrière, pour bien vérifier les réponses. J'ai eu l'occasion de tester celui-ci, mais il n'est pas encore au point et les résultats dépendent énormément du jeu de données proposé. Pour l'instant, il permet quand même d'obtenir une base de *mapping* à compléter par la suite.

### Mission 2

Lors de la deuxième mission, le travail réalisé sur les requêtes fédérées a permis de tester le potentiel du web sémantique en terme de recherche. Nous avons pu voir la possibilité de créer de nouvelles connaissances en groupant des informations provenant de différentes sources de données.

19. <https://github.com/MarjoriePetit/Mapping-RML>

Plus le web sémantique aura de données, plus les possibilités de faire des recherches pertinentes, précises et intéressantes augmenteront.

Dans notre cas, il fallait savoir comment le *mapping* a été réalisé pour pouvoir faire une requête correcte et obtenir ce que l'on veut. Du côté usage, il serait plus accessible de pouvoir indiquer la demande dans une phrase et d'obtenir les résultats correspondants.

De plus, les résultats prennent quelques minutes à arriver. Aujourd'hui, les utilisateurs d'internet veulent obtenir ce qu'ils recherchent le plus rapidement possible, en quelques secondes maximum. C'est une des raisons pour laquelle le web sémantique n'est pas encore utilisé à grande échelle. Les résultats sont certes plus cohérents mais beaucoup plus longs à obtenir.

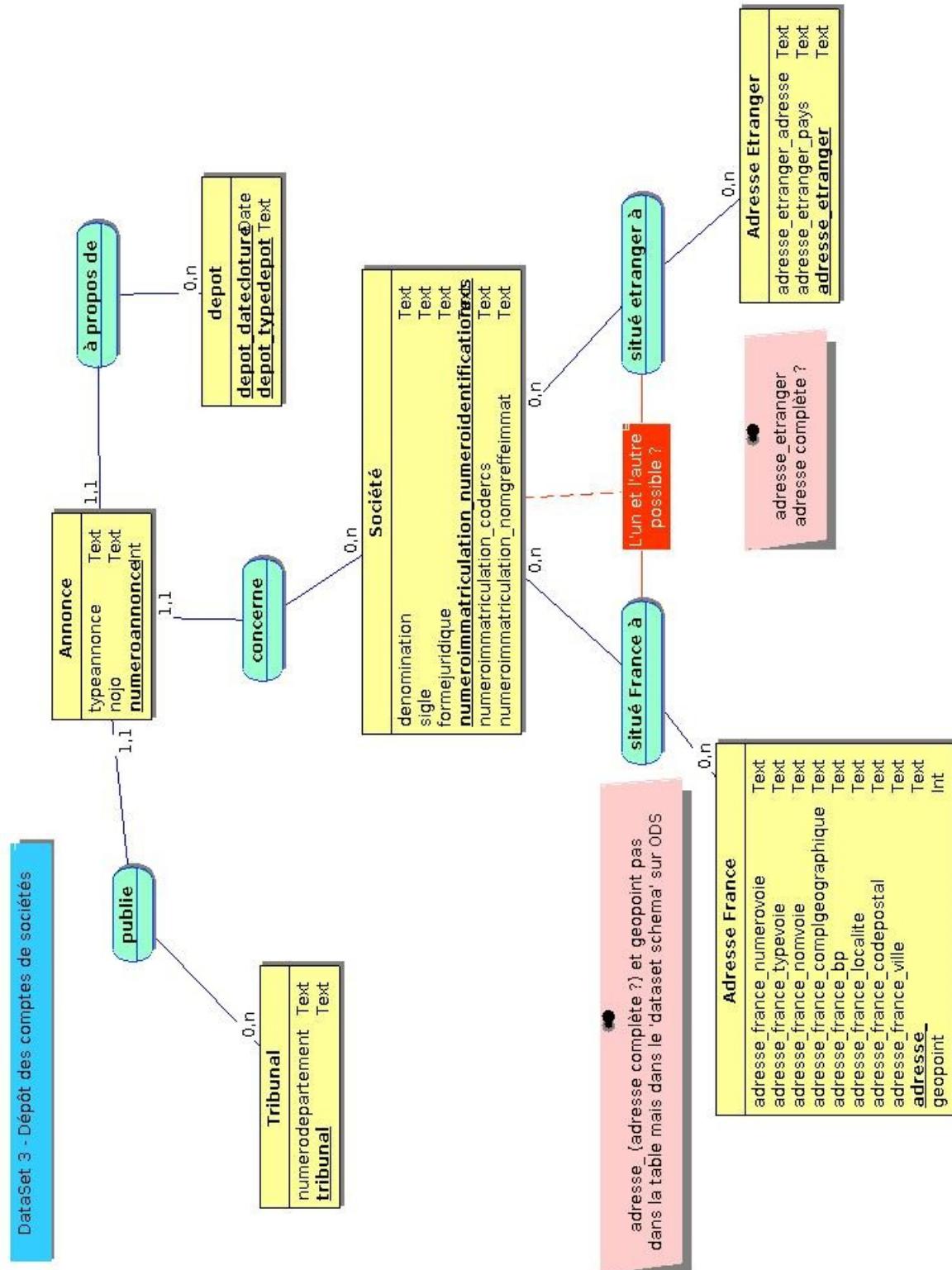
On utilise des outils qui sont actuellement en recherche et l'on sait qu'il existe des moyens de les améliorer. Notamment, on pourrait explorer les différentes techniques de caches pour la rapidité et revoir l'exécution des requêtes fédérées.

## Références

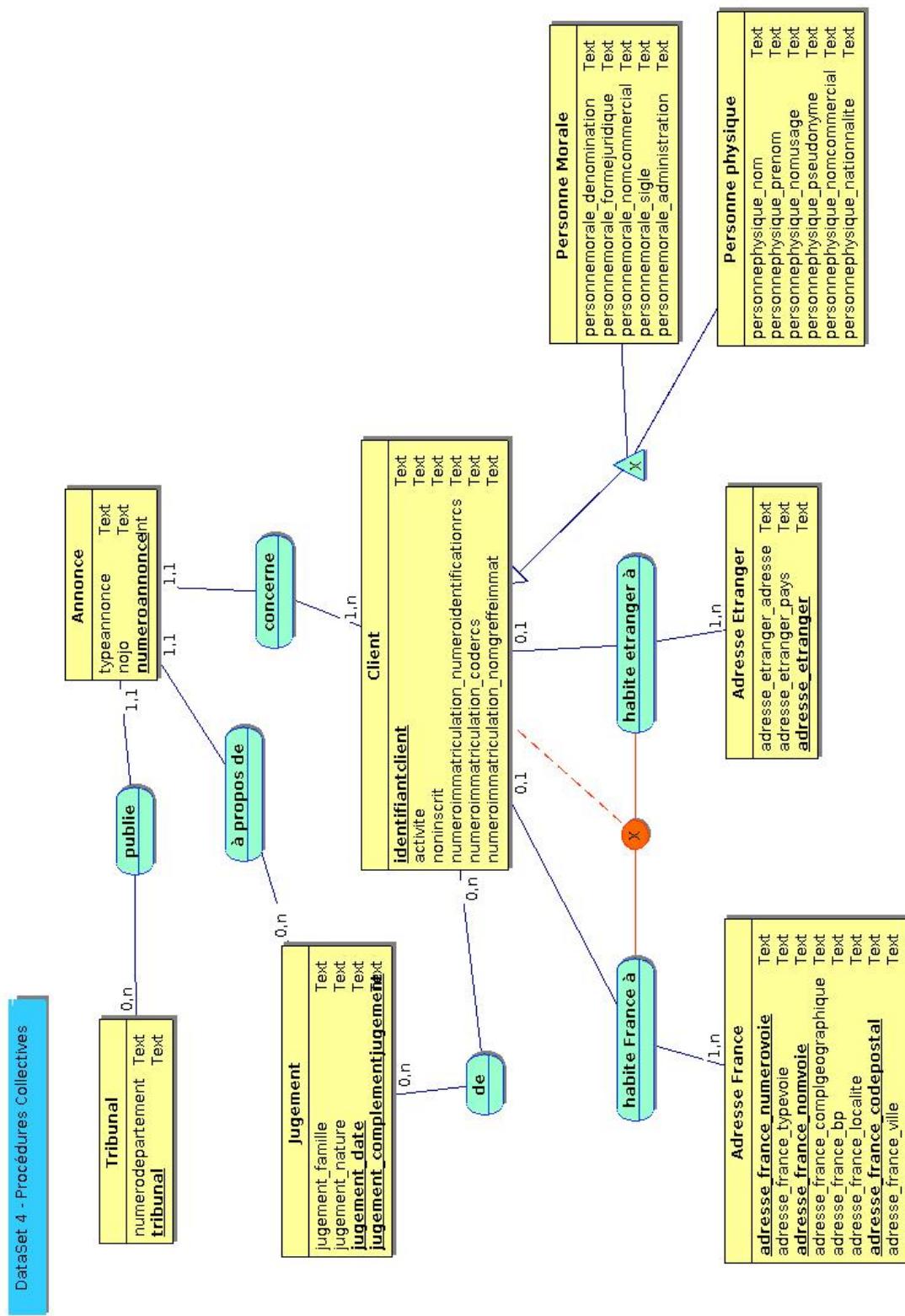
- [1] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific american*, 284(5) :34–43, 2001.
- [2] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - the story so far. *International journal on Semantic Web and information systems*, 5(3) :1–22, 2009.
- [3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia : A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer, 2007.
- [4] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C Recommendation 2008.
- [5] Benjamin Moreau, Patricia Serrano-Alvarado, Emmanuel Desmontils, and David Thoumas. Querying non-RDF Datasets using Triple Patterns. *International Semantic Web Conference (ISWC)*, Vienna, Austria, Oct 2017.
- [6] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. RML : A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Linked Data on the Web*, 2014.

## 7 Annexes

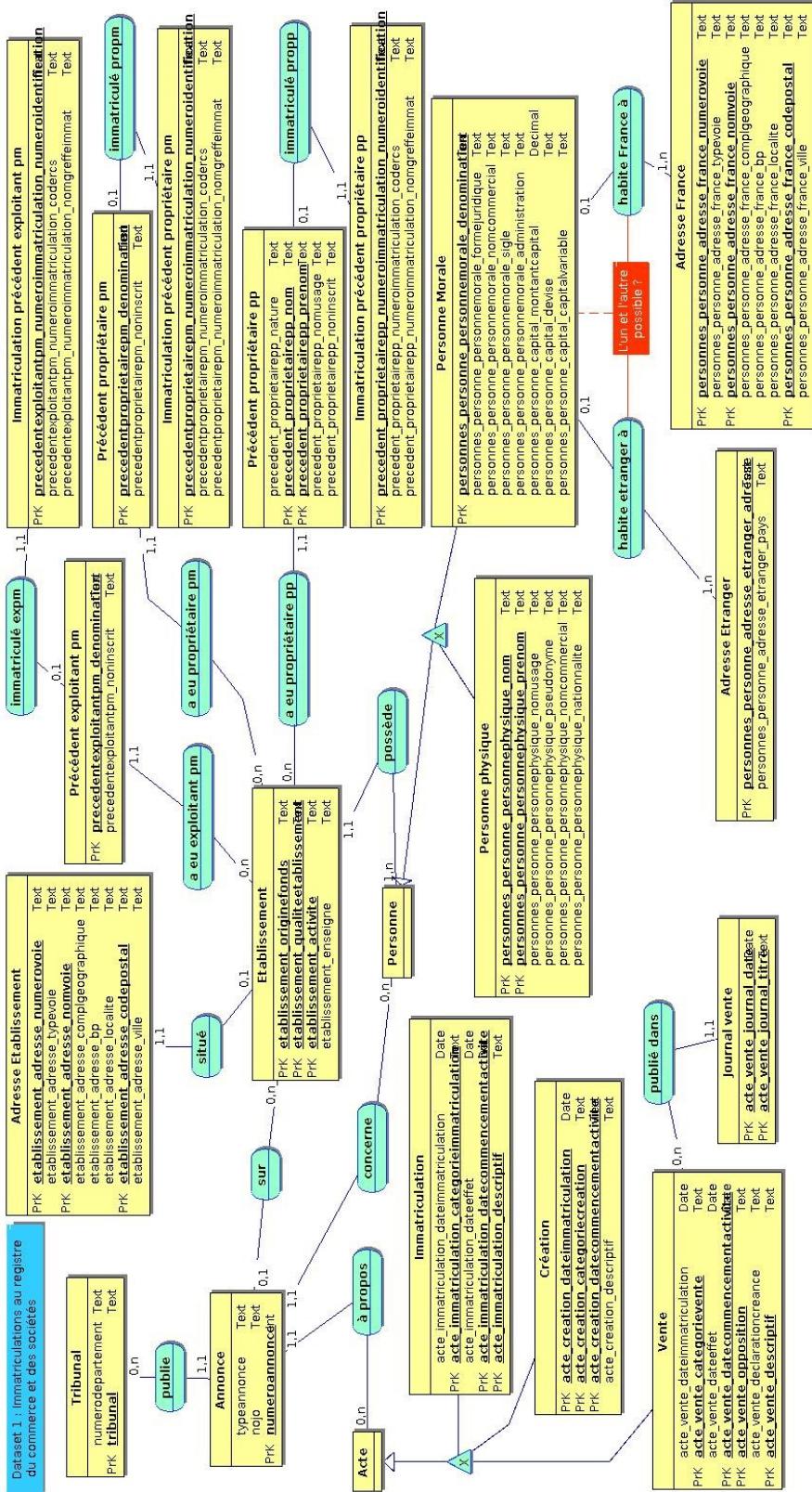
### A Modèle EAP du jeu de données 3 : Dépôt des comptes de sociétés



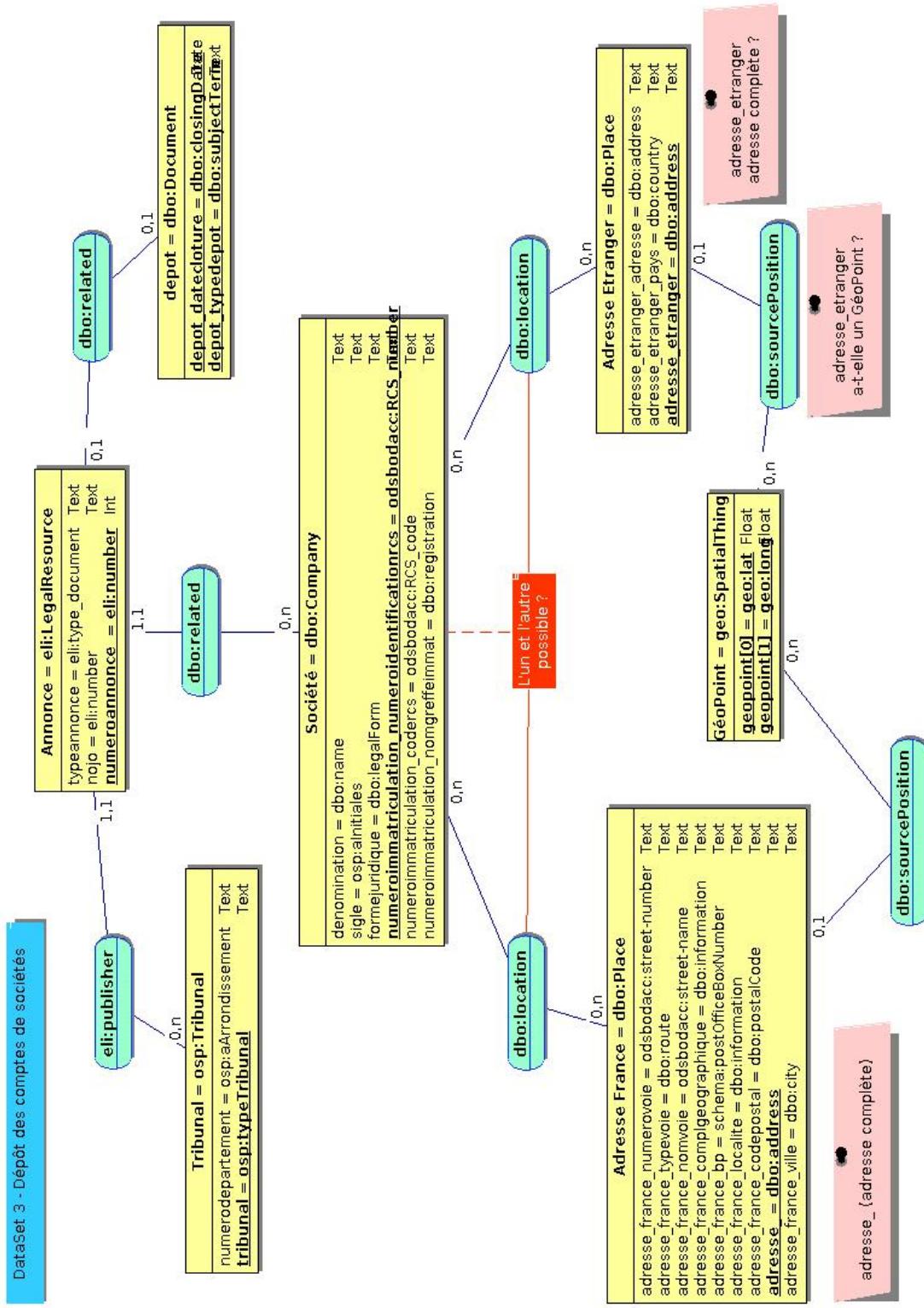
## B Modèle EAP du jeu de données 4 : Procédures Collectives



## C Modèle EAP du jeu de données 1 : Immatriculations au registre du commerce et des sociétés



## D Modèle du jeu de données 3 avec les ontologies



## E Modèle du jeu de données 4 avec les ontologies

