Ontologie et Web Sémantique : Sujet 2, Transit et Transports

Clément Siegrist Ihab Bendidi

Université de Paris
clement.siegrist@sciencespo.fr *ihab.bendidi@etu.u-paris.fr*

1 Table des matières

2 Introduction

Les dernières décennies ont vu apparaître le développement d'ontologies, c'est-à-dire les représentations formelles et explicitent des relations entre les différents termes d'un même domaine ou de plusieurs domaines différents. Grâce au développement du World Wide Web les ontologies se sont répandues. Elles sont utilisées pour créer de larges taxonomies pour répertorier les sites Web comme pour Yahoo ou Google ainsi que pour créer des catalogues de ventes comme pour Amazon. Ces avancées sont permises grâce aux développements de langages appropriés comme ceux créés par Le WWW Consortium (W3C). En effet, cette organisation a développé le Resource Description Framework (RDF, Guha et Brickley 1999) dont l'objectif était de représenter la connaissance sur les pages Webs afin de la rendre compréhensible et analysable par les agents électroniques (ordinateurs).

3 Description de la représentation d'une ontologie

Comme indiqué précédemment, une ontologie est une représentation, une description formelle et explicite des concepts d'un domaine ou d'un champ sémantique. Les concepts peuvent être assimilés à des "classes" dotées de "propriétés" qui décrivent différentes caractéristiques et attributs du concept. Des restrictions, contraintes et conditions peuvent être imposées aux classes ainsi qu'à leurs propriétés afin d'en préciser le sens et de spécifier les conditions de leur utilisation. Enfin, chaque classe et/ou sous-classes peuvent se voir attribuer des instances individuelles (ex: classe:Eleve, instance1:élève 1, instance2:élève 2). Une représentation générale du processus décrit est visible dans le **Schéma 1.**

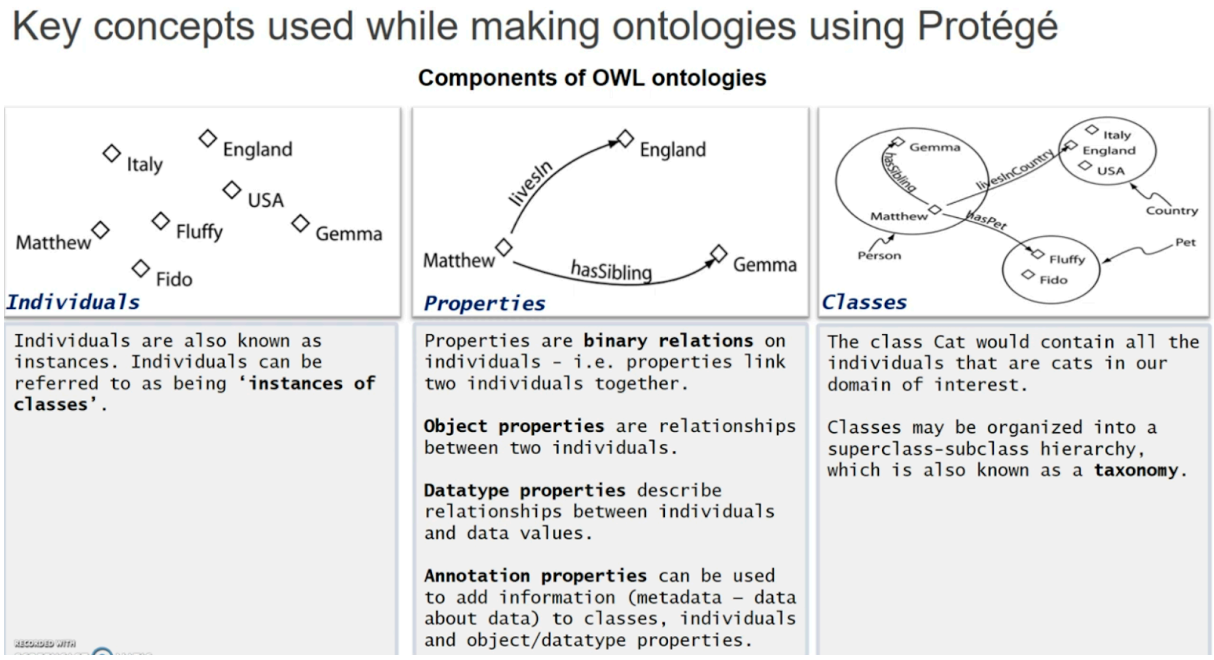

Schéma 1. Représentation succincte des principaux éléments permettant de créer une ontologie avec Protégé.

Cette façon de représenter l'information se base sur le modèle RDF dont l'expression est la forme sujet-prédicat-objet plus connue sous le nom de triplet. Par exemple : dans la phrase "La couleur du ciel est bleue" le sujet peut être représenté par "Le ciel", le prédicat par "est de couleur" et l'objet par "bleue". Le sens d'une ontologie OWL est déterminé par son graphe RDF bien que l'on puisse utiliser des formes syntaxiques comme le RDF/XLM tant que le résultat final représente l'information sous la forme de triplets. De surcroît, selon la logique des bases de données SQL, un triplet RDF peut être considéré comme un tableau avec trois

colonnes: la colonne correspondant au sujet, celle correspondant au prédicat et celle correspondant à l'objet. Ainsi on utilise le langage SPARQL pour effectuer des requêtes sur les ontologies d'une manière similaire à celles que l'on peut effectuer sur des bases de données SQL.

4 L'ontologie des moyens de transport publics

4.1 Travaux similaires

Des ontologies très complètes telle que la *Public Transit Ontology* développée par l'université de Toronto dont un aperçu de la représentation en graphe et des métadonnées principales sont représentées dans la **Figure 1.** sont disponibles et pourrait être utilisée dans un travail futur pour compléter notre ontologie.

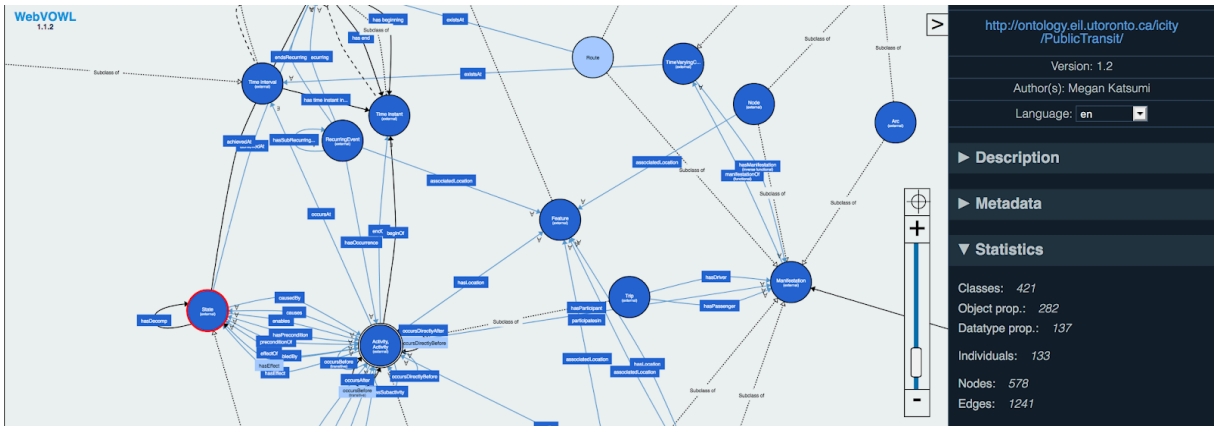

Figure 1. Partie du graphe et statistiques de l'ontologie *Public Transit Ontology* développée par Megan Katsumi.

Des précisions complémentaires sur l'ensemble de cette ontologie comme sa structure, les motivations de ce travail (utilisations dans le cadre de projet urbain smart city), requêtes et développement de l'API sont disponibles dans la thèse récemment publiée par Megan Katsumi [2].

4.2 Notre Solution

Nous avons développé, sur la base des consignes édictées dans le descriptif du projet, une ontologie transit et transports avec le logiciel Protégé et illustrée grâce à des technologies de visualisation et une API web.

4.2.1 Construction de l'ontologie avec Protégé

Notre travail consiste à créer une ontologie des transports publics et de leur utilisation par trois individus : Alexis, Marie et Olaf.
L'ontologie a été créée grâce au logiciel Protégé développé par l'Université de Standford et plus précisément par le groupe de Mark Munsen du laboratoire d'informatique médical [1]. Protégé permet de créer les classes de l'ontologie, de leur attribuer des propriétés, de vérifier la cohérence et la consistance de l'ontologie grâce à un débogueur, de visualiser l'ontologie ou certaines et ces parties ainsi que les relations qu'elles entretiennent entre elles

grâce à la fonctionnalité OntoGraf et d'effectuer des requêtes pour vérifier la fonctionnalité de l'ontologie grâce à SPARQL Query.

Classes : Nous avons défini plusieurs classes comportant elles-mêmes plusieurs sous-classes comme indiqué ci-dessous dans le **Schéma 1.** La légende des propriétés (prédicats) est disponible en dessous du **Schéma 7.** quelques pages plus bas.

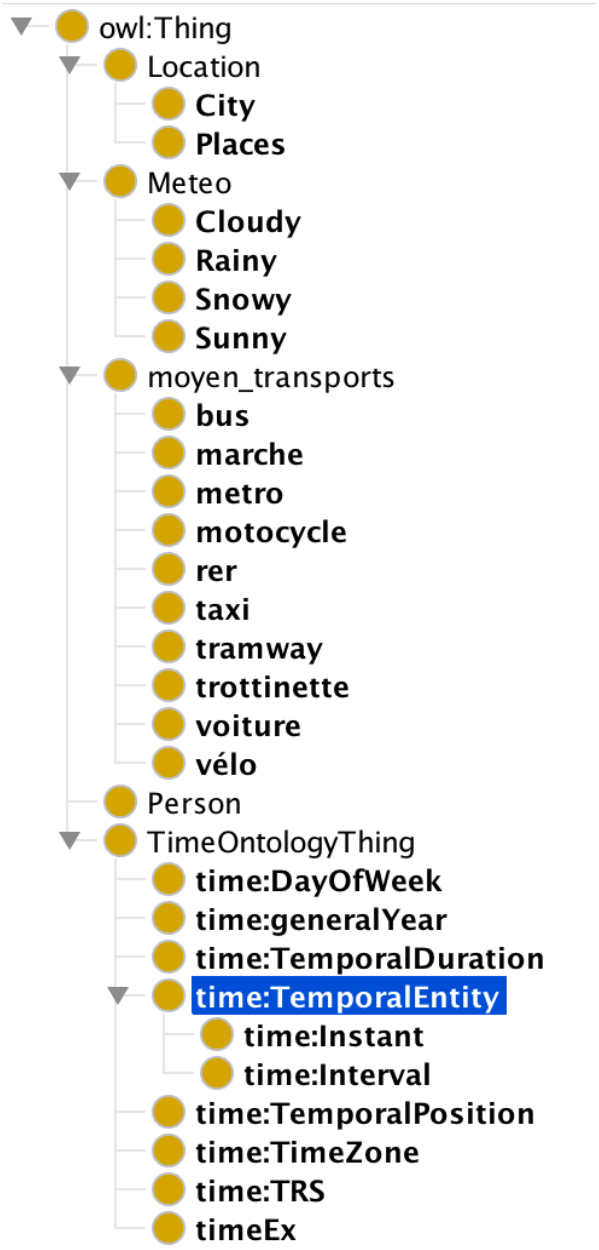

Schéma 1. Liste des classes et de sous-classes définies pour notre ontologie

Comme indiqué dans le **Schéma 1.** les principales classes sont les classes : "Location", qui correspond aux différents lieux, les classes "moyens_transports" qui correspond aux différents moyens de déplacements utilisés, la classe "Person" qui représente les individus, la classe "TimeOntologyThing" que nous souhaitions rendre plus granulaire, mais par manque de temps la seule sous-classe pertinente et qui sera utilisée sera "timeEx" qui comporte les instances de temps nécessaires pour représenter les informations aux pertinentes. Enfin, nous souhaitions représenter fidèlement les contraintes de déplacement

pour les individus concernés en fonction de la météo, ce qui n'est pas encore achevé mais le sera dans un travail prochain.

La classe la plus importante est la classe "Person" décrite dans **Schéma 2.** à partir de laquelle les principales propriétés découlent. Le **Schéma 7** donne une description plus globale de l'ontologie.

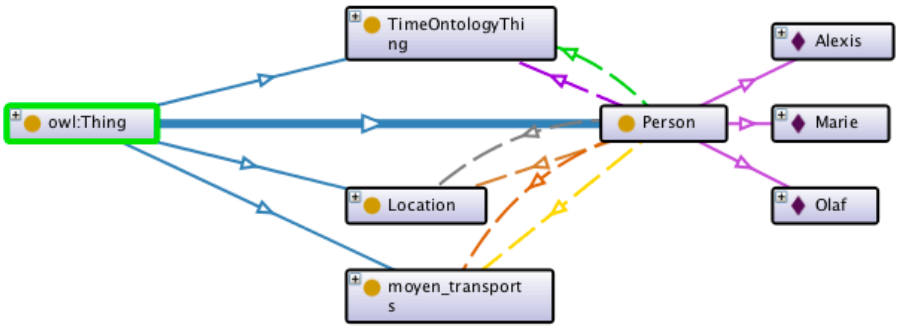

Schéma 2. Description de la classe "Person", de ces instances et des propriétés avec les autres classes.

Les principales classes utilisées pour caractériser les lieux sont "Location" et ses sous-classes "City" et "Places" comme indiqué dans le **Schéma 3.**

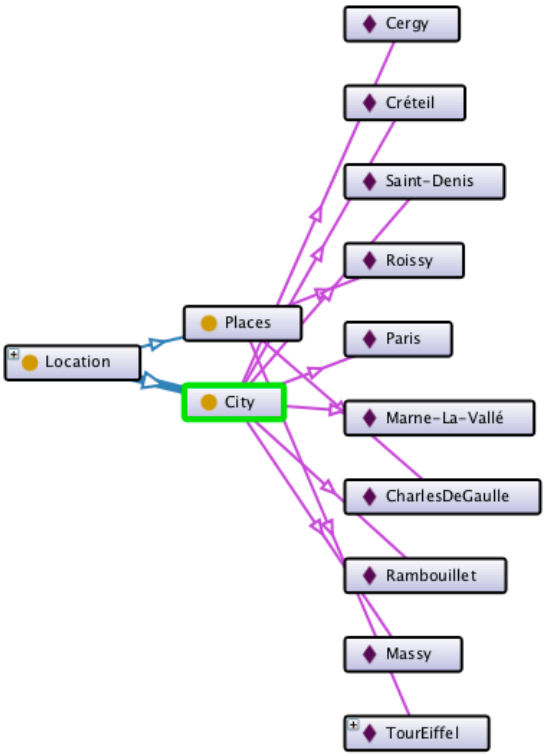

Schéma 3. Classe Principale Location, ses sous-classes City (ville) et Places (endroits, monuments) et les instances : Paris, Créteil, Roissy pour la classe "City" et TourEiffel, CharlesDeGaulle, Roissy pour la classe "Places".

Enfin, la classe décrivant les différents moyens de transports ainsi que leurs instances est la classe "moyen_transport" décrite dans le **Schéma 4.**

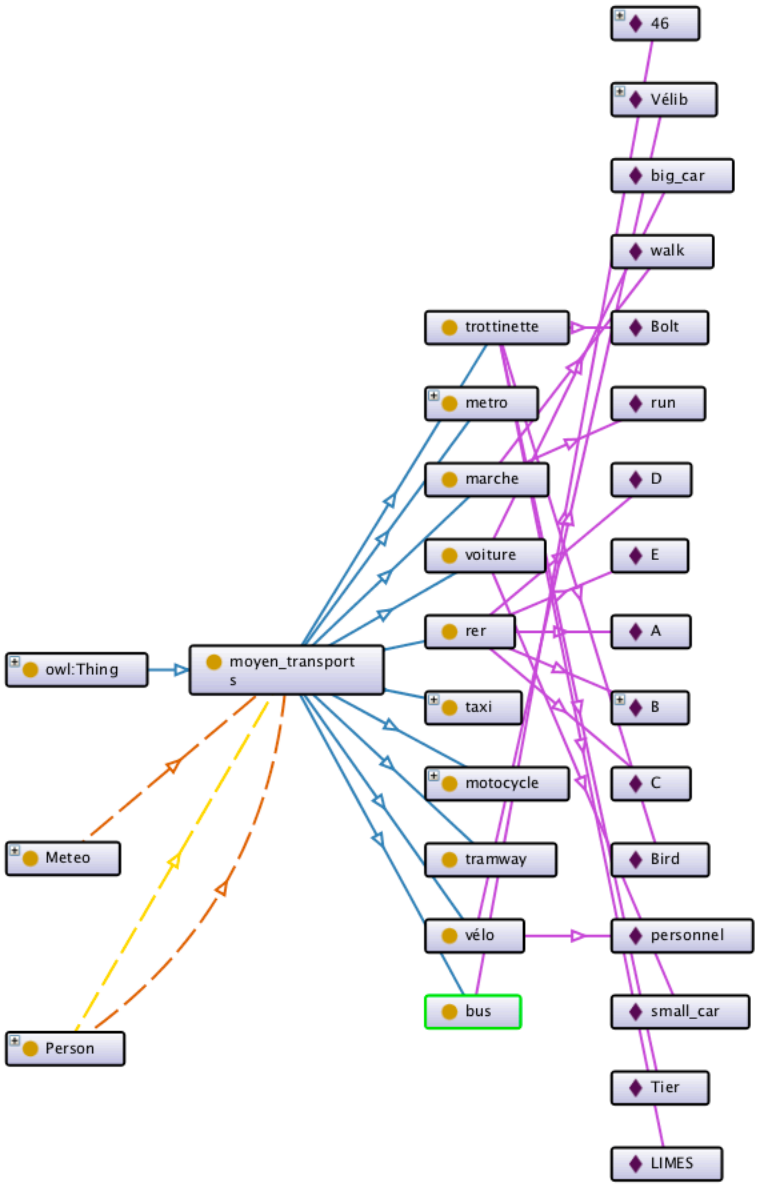

Schéma 4. Classe principale “moyen_transport“ comportant dix-sous classes comportant elles-mêmes plusieurs instances

Propriétés : Nous avons défini plusieurs propriétés/relations reliant les classes et individus des classes entre eux. Les propriétés que nous avons définies sont celles visibles dans le **Schéma 5**. La légende des propriétés est disponible en dessous du **Schéma 7**.

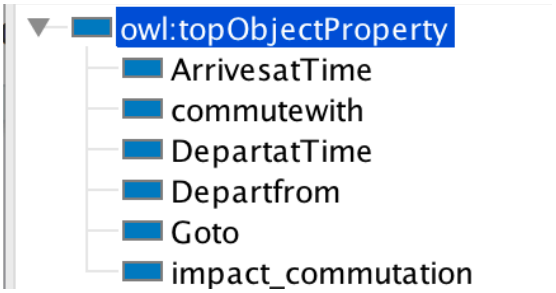

Schéma 5. Principales propriétés (prédicats) définies pour mettre en relation les classes et instances de différentes classes de notre ontologie.

Un certain nombre de sous-propriétés/caractéristiques/contraintes peuvent être attribuées aux Propriétés. Nous avons utilisé les fonctionnalités suivantes pour définir et préciser les domaines d'applicabilité des propriétés : *Domains*, *Ranges* ainsi que les caractéristiques qui imposent des contraintes aux propriétés comme indiqué dans le **Schéma 6.**

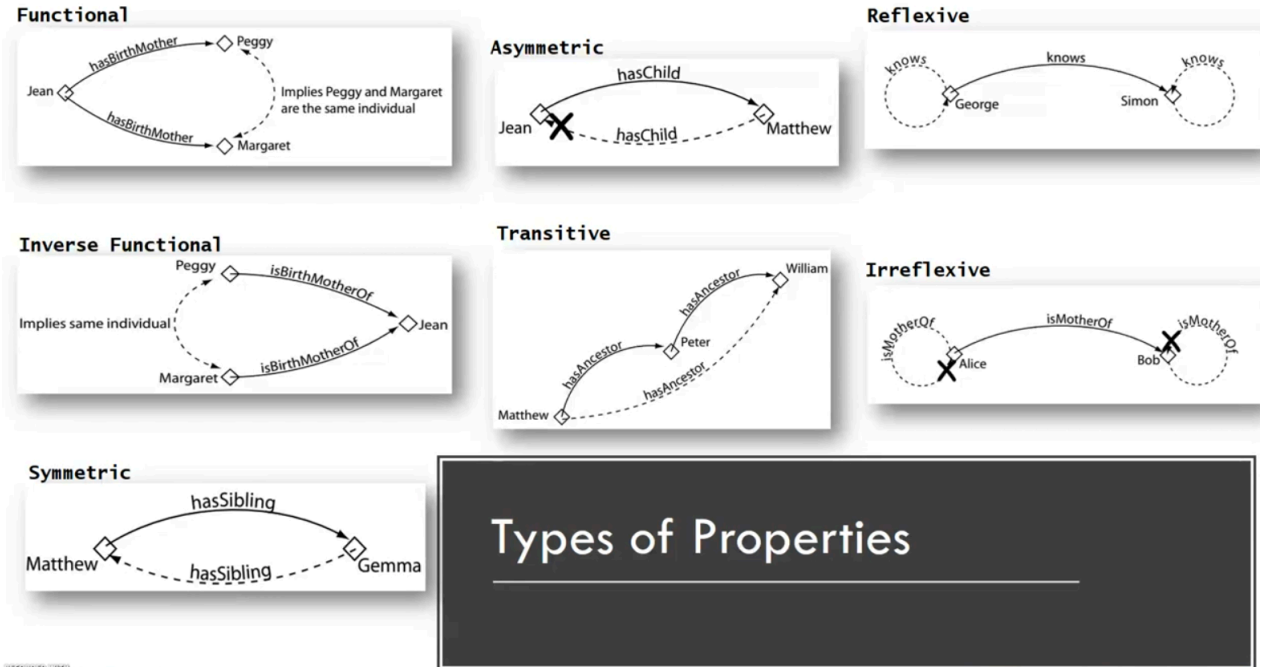

Schéma 6. Description schématique des cas d'usage pour chaque type de fonctionnalités.

Disjoints Classes / Individuals : On rend chaque instance de chaque classes disjointes afin d'éviter l'héritage multiple.

Data Properties : Définit des propriétés/relations entre les instances de classes et les valeurs des données d'une instance d'une classe. Par exemple : ID d'une ligne de métro, d'un module d'un cours pour un professeur etc. Nous avons défini comme "Data property" le niveau de pollution pour chaque instance des moyens de transports comme indiqué sur le **Schéma 6**.

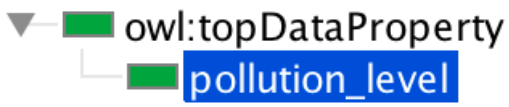

Schéma 6. "Data Property" qui assigne un niveau de pollution pour chaque moyen de transport dont l'unité est donnée en g/CO2/km.

Schéma global de notre ontologie :

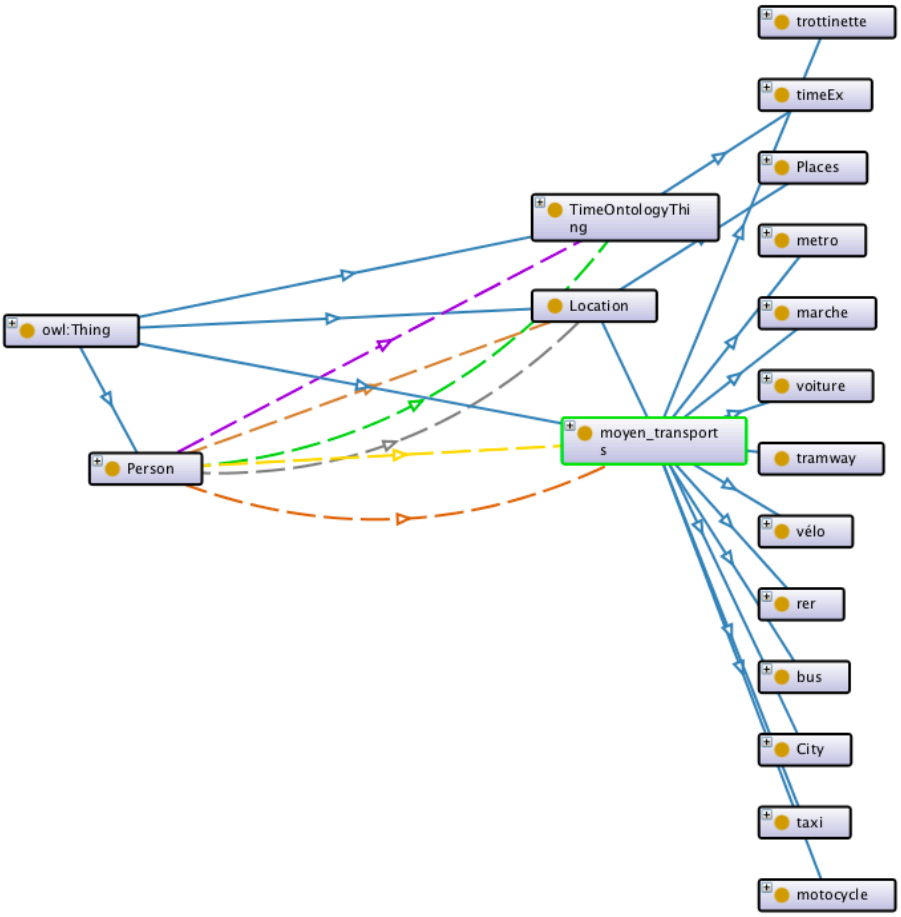

Schéma 7. Schéma global de l'ontologie.

Légende des Propriétés et traduction en français :

------ : DepartatTime (Heure de départ)
------ : DepartFrom (Lieu de départ)
------ : ArrivesatTime (Heure d'arrivée)
------ : Goto (Va vers)
------ : Commutewith (Se déplace avec)
------ : ImpactCommutation (A une influence sur le mode de transport choisi)

Requêtes SPARQL

Nous avons effectué les requêtes SPARQL suivantes permettant d'afficher les résultats demandés dans les questions. Les requêtes ont été effectuées grâce à la fonctionnalité SPARQL Query de Protégé et par la suite avec RDFlib depuis notre application.

1. La requête suivante permet d'afficher toutes les classes ainsi que leurs sous-classes :

```
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#>
```

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

2. La requête suivante permet d'afficher tous les moyens de transport utilisés par Marie :

```
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
xd:      <http://www.semanticweb.org/clementsiegrist/ontologies/2021/0/pub-transportation#>

SELECT ?object
   WHERE { xd:Marie xd:commutewith ?object}
```

3. La requête suivante permet d'obtenir la liste des personnes ayant pris un Uber, à quelle heure et où :

```
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 xd:
<http://www.semanticweb.org/clementsiegrist/ontologies/2021/0/pub-transportation#>

SELECT ?personne ?lieu_date ?heure_depart
   WHERE {
        ?personne rdf:type xd:Person.
        ?personne xd:commutewith xd:Uber.
        ?personne xd:Departfrom ?lieu_date.
        ?personne xd:DepartatTime ?heure_depart. }
```

4. La requête suivante (avec RDFLib) permet d'obtenir le nombre de personnes prenant un Uber :

```
qres = g.query("""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 xd:
<http://www.semanticweb.org/clementsiegrist/ontologies/2021/0/pub-transportation#>
                  SELECT ?personne
                  WHERE {
                        ?personne rdf:type xd:Person.
                        ?personne xd:commutewith xd:""" + moyen + """.}""")
```

5. La requête suivante permet d'obtenir le nombre de personnes prenant un Uber par heure et par ville :

```
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 xd: http://www.semanticweb.org/clementsiegrist/ontologies/2021/0/pub-transportation#

SELECT ?lieu_date ?heure_depart (count(?personne) as ?num)
        WHERE {
         ?personne rdf:type xd:Person.
         ?personne xd:commutewith xd:Uber.
         ?personne xd:Departfrom ?lieu_date.
         ?personne xd:DepartatTime ?heure_depart. }

group by ?lieu_date ?heure_depart
```

6. La requête suivante (avec RDFlib) permet d'obtenir des informations sur la quantité de pollution en gramme de CO2 par kilomètres pour chaque moyen de transport. Les mesures se basent sur plusieurs études, lesquelles ne parviennent pas toujours aux mêmes résultats. Les mesures indiquées doivent donc être considérées comme indicatives d'un possible ordre de grandeur et non pas comme une vérité absolue :

```
qres = g.query("""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 ex:
<http://www.semanticweb.org/clementsiegrist/ontologies/2021/0/pub-transportation#>
                    SELECT ?pollution_level ?type
                       WHERE { ?type rdf:type ex:""" + moyen + """.
                       ?type ex:pollution_level ?pollution_level.}""")
```

4.2.2 Application Web et rendu final

Après avoir conceptualisé notre ontologie, notre objectif suivant est de créer une interface qui nous permettra de pouvoir observer des résultats extraits de notre ontologie. Bien que le volume de données injecté soit faible, notre application fonctionne relativement bien et pourrait s'avérer performante avec un volume de données plus important.

4.2.2.1 Outils technologiques choisis

Flask :

Flask est un framework web. Cela signifie que flask fournit des outils, des bibliothèques et des technologies qui permettent de construire une application web. Cette application web

peut être une page web, un blog, un wiki ou être aussi grande qu'une application de calendrier sur le web ou un site web commercial, ainsi que des APIs qui permettent d'introduire des fonctionnalités définies par Flask dans d'autres applications.

Flask est un microframework. Ceux-ci sont des frameworks ayant peu ou pas de dépendances avec des bibliothèques externes. Cela présente des avantages et des inconvénients. L'avantage princiapl est que le framework est léger, qu'il y a peu de dépendances à mettre à jour et à surveiller pour les bugs de sécurité, le désavantage est qu'à un moment donné la charge de travail pour le développeur augmente proportionnellementà l'augmentation des dépendances et des plugins nécessaires.

Jinja2 :

Jinja est un langage de template moderne et convivial pour Python, inspiré des modèles de Django. Il est rapide, largement utilisé et sécurisé grâce à l'environnement d'exécution de modèles en sandbox optionnel. Il offre plusieurs caractéristiques, à savoir qu'il dispose d'un puissant système d'échappement automatique du HTML pour la prévention du XSS, un très bon modèle d'héritage, une capacité de compilation de code python optimal juste à temps, et compilation optionnelle de modèles à l'avance. Il est aussi facile à déboguer, en effet les numéros de ligne des exceptions renvoient directement à la ligne correcte du modèle.

RdfLib :

RDFLib est une bibliothèque Python permettant de travailler avec RDF, un langage simple mais puissant pour représenter l'information. Cette bibliothèque contient des analyseurs/sérialiseurs pour presque toutes les sérialisations RDF connues, telles que RDF/XML, Turtle, N-Triples, & JSON-LD, dont beaucoup sont maintenant supportées dans leur forme mise à jour (par exemple Turtle 1.1). La bibliothèque contient également des back-ends Graph en mémoire, persistants pour le stockage des informations RDF et de nombreuses fonctions pratiques pour la déclaration des espaces de noms des graphes et pour l'hébergement des requêtes SPARQL par exemple. Elle est en développement continu, la dernière version stable, rdflib 5.0.0, ayant été publiée le 18 avril 2020. Il a été créé à l'origine par Daniel Krech avec la première version en novembre 2002.

4.2.2.2 Développement des APIs

Un API est l'acronyme d'Application Programming Interface, que l'on traduit en français par interface de programmation applicative ou interface de programmation d'application. L'API peut être résumée à une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données. Il s'agit en réalité d'un ensemble de fonctions qui facilitent, via un langage de programmation, l'accès aux services d'une application.

Pour pouvoir extraire et utiliser les données de l'ontologie dans plusieurs contextes et applications, nous avons décidé de développer des APIs qui permettent de consommer ces informations et d'extraire des données qui visualisent l'utilité de notre ontologie.

- **Lister les moyens de transports existants :**

Cette API permet de lister tous les moyens de transports existants dans l'ontologie.

Appel sur terminal :

```
curl -X GET 'http://127.0.0.1:5000/api/moyen_transports'
```

Retour :


```
{
 "code": 200,
 "response": [
        "metro",
        "v\u00e9lo",
        "trottinette",
        "motocycle",
        "rer",
        "marche",
        "voiture",
        "taxi",
        "bus",
        "tramway" ]}
```


Cette API permet de lister tous les moyens de transport utilisés par un individu spécifique dans notre base de données.

Variable : Nom de l'individu.

Appel sur terminal :

```
curl -X GET 'http://127.0.0.1:5000/api/individu_transports?individu=Alexis'
```

Retour :


```
{
  "code": 200,
  "response": [
        "Uber"
  ]
}
```


- **Calculer les statistiques journalières de l'utilisation d'un moyen de transport:**

Cette API permet de calculer les statistiques de l'utilisation quotidienne d'un moyen de transport donné.

Variable : Moyen de transport.

Appel sur terminal :

```
curl -X GET 'http://127.0.0.1:5000/api/statistics?moyen=Uber'
```

Retour :

```
{
 "code": 200,
 "response": 1
}
```


- **Extraire le degré de contribution à la pollution de chaque moyen de transport:**

Cette API permet de donner le degré de contribution à la pollution de chaque moyen de transport donné.

Variable : Moyen de transport.

Appel sur terminal :

```
curl -X GET 'http://127.0.0.1:5000/api/pollution?moyen=bus'
```

Retour :


```
{
 "code": 200,
 "response": "68"
}
```


4.2.2.3 Développement d'interface Web

Pour pouvoir tester et valoriser notre travail, nous avons développé une interface Web en utilisant nos APIs. L'application est en libre accès et disponible sur github.

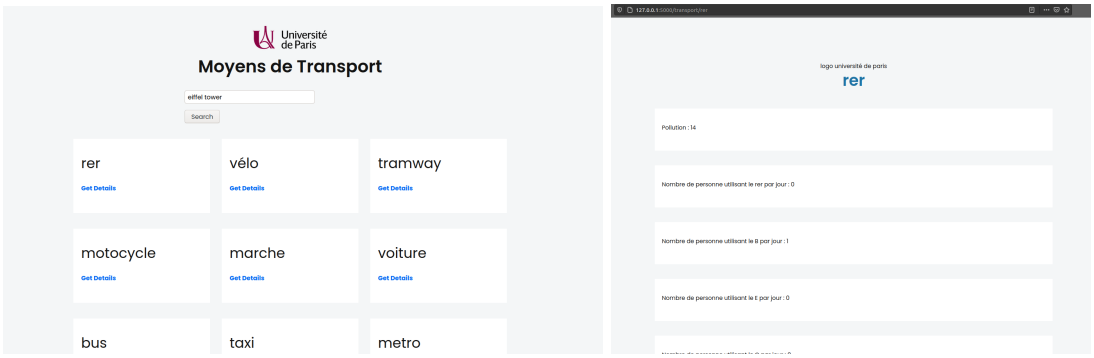

Figure 1 : Présentation des moyens de transports et de leurs détails.

La **Figure 1** montre notre première page d'interface qui liste tous les moyens de transports existant dans notre ontologie. Pour ces moyens de transport, nous pouvons aussi extraire leur niveau de pollution, ainsi que les statistiques de leur utilisation quotidienne.

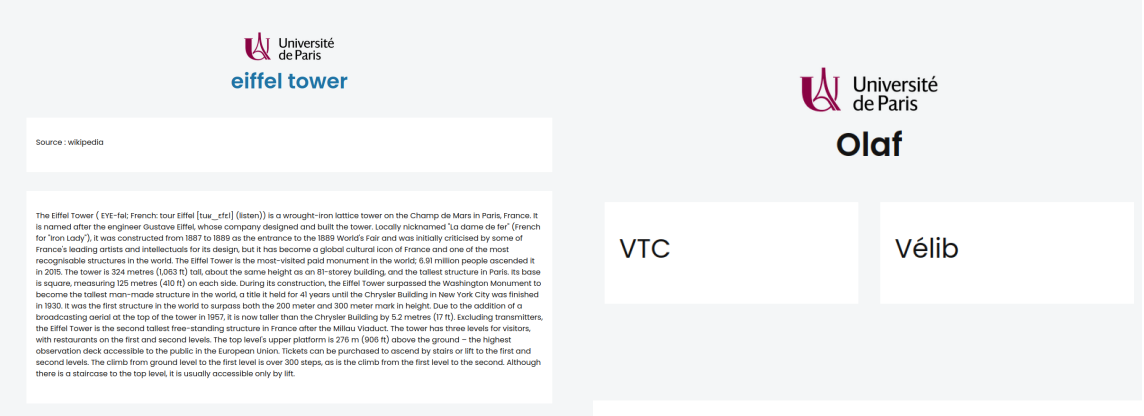

Figure 2 : A gauche, des informations de zones touristiques recherchées, à droite, les utilisations de transport des individus.

La **Figure 2** montre à gauche des informations extraites de la *Linked Data* sur des zones et localisations touristiques, dont nous affichons le contenu et la source. A droite, nous essayons de lister les moyens de transports utilisés par un individu donné.

5 Travaux futurs et améliorations

Une étape supplémentaire aurait été d'importer des ontologies directement depuis Dbpedia http://fr.dbpedia.org/page/Paris pour venir enrichir la classe "Location" et y ajouter en tant qu'objets les lieux principaux de Paris et éventuellement de la région Ile-de-France. Intégrer des fonctions capables d'effectuer des requêtes sur des sites (wikipédia, dbpedia) ou sur des APIs permettant d'obtenir en temps réel des informations sur les villes, lieux monuments, évènements en cours ou encore la qualité de l'air locale représenteraient de vraies valeurs ajoutées.

6 Conclusion

L'application Protégé permet assez aisément de créer des ontologies complètes, de charger et de modifier rapidement des ontologies déjà existantes, de les visualiser, de générer des fichiers décrivant l'ontologie de formats différents et de tester des requêtes SPARQL directement dans l'application.

7 Bibliographie

[1] Rothenfluh, T.R., Gennari, J.H., Eriksson, H., Puerta, A.R., Tu, S.W. and Musen, M.A.(1996) *Reusable ontologies, knowledge-acquisition tools, and performance systems:PROTÉGÉ-II solutions to Sisyphus-2*, International Journal of Human-Computer Studies44:303-332
[2] M. Katsumi et Mark Fox (2020), *iCity Transportation Planning Suite of Ontology*, University of Toronto Faculty of Applied Science & Engineering.