

Construction d’un extracteur d'information géographique

avec R et SPARQL

Juliette Delannoy

Projet de développement informatique de troisième année du cycle ingénieur

Encadré par Hadrien Commenges et Thomas Louail

# Résumé

Depuis plusieurs années, le web sémantique a fait son apparition avec l’idée de structurer la donnée pour pouvoir automatiser son utilisation. Par exemple, le projet DBpédia a été créé en marge de Wikipédia pour stocké l’information brute disponible sur cette plateforme dans une base de données. Bien que libre, ces données sont difficiles d’accès car il faut s’être familiarisé au web sémantique, à DBpédia et au langage SPARQL permettant de faire des requêtes sur la base de données. L’objectif de mon stage est donc de fournir un outil simple d’utilisation avec une prise en main rapide, qui permette d’interroger la base de données sans être familier des contextes cités plus haut.

# Détails du projet

Le projet a été mené par Juliette Delannoy, étudiante en 3ème année d’école d’ingénieur à l’ENSG, filière Carthagéo, dans le cadre du projet développement.

Ce projet a été encadré par Hadrien Commenges et Thomas Louail, dans les locaux du laboratoire de recherche Géographie-cités, 13 rue du Four, 75006 Paris.

Ce projet a duré 5 semaines, du 8 janvier 2018 au 9 février 2018.

# Plan

[Résumé 2](#_Toc505866178)

[Détails du projet 2](#_Toc505866179)

[Plan 3](#_Toc505866180)

[Introduction générale 4](#_Toc505866181)

[Du web sémantique à DBpédia 5](#_Toc505866182)

[Le Web Sémantique c'est quoi ? 5](#_Toc505866183)

[Normaliser le format des données 5](#_Toc505866184)

[Interroger une base de données 6](#_Toc505866185)

[Vocabulaire standard 6](#_Toc505866186)

[Où la donnée est-elle stockée ? 7](#_Toc505866187)

[La base de données DBpédia 8](#_Toc505866188)

[Fonctionnement de SPARQL 9](#_Toc505866189)

[Le langage d’interrogation de la base de données 9](#_Toc505866190)

[Choix des délimitations du projet 10](#_Toc505866191)

[Interface 12](#_Toc505866192)

[Guide utilisateur 12](#_Toc505866193)

[Requête 12](#_Toc505866194)

[Carte 12](#_Toc505866195)

[Informations supplémentaires 12](#_Toc505866196)

[Potentiel et limites 14](#_Toc505866197)

[Problèmes liés à DBpédia 14](#_Toc505866198)

[Choix d’extraction des ontologies 15](#_Toc505866199)

[Potentiel de l’application 16](#_Toc505866200)

[Conclusion 17](#_Toc505866201)

# Introduction générale

Le développement en flèche d’internet ces dernières années et l’apparition de nouvelles sources de données massives (big data) permet à chacun d’avoir à disposition une quantité quasi-illimitée de données, la difficulté majeure étant de savoir où chercher l’information qui nous intéresse ainsi que comment. Depuis une dizaine d’années est apparu le concept du web sémantique, dont l’idée est de structurer au mieux les données pour permettre l’accès à la donnée brute. Cela rendrait possible l’ interrogation automatique et pourrait notamment permettre aux machines de devenir plus autonomes en leur permettant de faire des recherches standardisées sur internet.

Une des bases de données structurées les plus connues et celle de DBpédia, qui contient une grande partie des informations disponibles sur Wikipédia. Afin de pouvoir utiliser cette base de données structurée du web, ainsi que d’autres, un langage permettant de les interroger est nécessaire. Le langage SPARQL permet d’interroger les bases de données, mais, comme tout langage, il nécessite un apprentissage préalable pour obtenir les résultats souhaités.

Mais quel est l’intérêt de DBpédia ? Lorsqu’on a besoin d’une information simple telle que les noms des albums de notre chanteur préféré, on peut les trouver très facilement sur Wikipédia. Mais quand on cherche quelque chose comme « les athlètes nés en France depuis 1950 », cela devient plus compliqué.

Il serait donc utile d’avoir un outil d’extraction d’information géographique qui permette d’avoir une interface utilisable par n’importe qui et permettant en quelques clics d’avoir n’importe quel type d’information. Le projet que j’ai mené pendant ces 5 semaines en était le but.

Pour construire cet extracteur d’information géographique, il a fallu que je me familiarise avec les concepts du web sémantique, que j’apprenne le langage de requêtage SPARQL, que je délimite le projet, puis que je construise l’interface. Ces étapes sont les parties principales de ce rapport. Le temps nécessaire à chacune des étapes, ainsi que le détail de chacune d’entre elles sont présentés en annexe sous forme d’un GANTT.

Je commencerai donc par expliquer ce qu’est le web sémantique, puis je parlerai du langage SPARQL permettant d’interroger DBpedia et enfin je présenterai l’interface que j’ai mise en place.

# Du web sémantique à DBpédia

De plus en plus de données étant disponibles sur internet, il y a une réelle nécessité de traiter et d’interpréter ces données. Or ces données ne sont pas toujours structurées et donc difficilement lisibles par les ordinateurs. Un exemple de cela est celui des sites de comparaison des prix des vols selon les compagnies aériennes. Dans ces compagnies, un développeur est employé pour étudier la structure de chacun des sites de vols d’avion et ainsi de permettre l’extraction de son contenu. Avec un web uniformément structuré, ce travail ne serait pas nécessaire puisque l’architecture de toutes les pages serait similaire.

On s’oriente alors vers une transition d’un web dit « sémantique ».

## Le Web Sémantique c'est quoi ?

Le web sémantique, ou ***Web 3.0*** est souvent appelé le web des données.

Le but est de pouvoir utiliser le web comme une immense base de données, que l'on peut enrichir (les projets gouvernementaux fournissent des jeux de données utilisables par tous), ou utiliser (recherche d'information pour enrichir sa propre base de données).

L'idée est de parvenir à un Web intelligent, où les pages des sites seraient gérées par une base de données intelligente ce qui permettrait de mettre les objets au service des personnes.

Pour cela il faut que les données soient interopérables et soient accompagnées de leur sémantique (***ontologie***). L’ontologie constitue un modèle de données qui représente les concepts d’un domaine ainsi que les relations entre ces concepts. Les ontologies permettent de représenter la donnée pour qu’elle soit compréhensible par un ordinateur. Il s’agit d’une sorte d’étiquette : Usain Bolt est une personne et un athlète, il s’agit là de deux ontologies.

En plus de données interopérables, il est nécessaire d’utiliser un dispositif langagier normalisé afin qu'il soit utilisable par tous de la même manière, ce qui permettra l’accès universel et l’utilisation intelligente de la donnée par les ordinateurs et par les personnes.

## Normaliser le format des données

Les données doivent être formalisées sous un format standard.

Pour cela on utilise un graphe appelé ***RDF*** (= Resource Description Framework) qui est un modèle de données composé du triplet suivant :

- Le ***sujet*** (l'élément à décrire)

- Le ***prédicat*** (une propriété de cet élément)

- L'***objet*** (la valeur de cette propriété, qui est un autre élément)

On peut les voir comme une phrase reliant 2 éléments, comme par exemple "U2 a écrit l'album 'Songs of Innocence' ".

U2 est le sujet, 'a écrit l'album' est le prédicat et 'Songs of Innocence' est l'objet. Cet objet peut-être le sujet d’un nouveau triplet, par exemple la date d’écriture : « Songs of Innocence est sorti en 2014 ».

Le but du web sémantique étant de relier des données du monde entier, ces 3 éléments doivent être uniques. Un RDF est donc un triplet d’***URI*** (Uniform Resource Identifier). Le but principal d’une URI est justement de fournir un nom universellement unique à une ressource afin qu’il soit possible de lier des données de différentes sources partout dans le monde. Cela permet également d’ajouter une sémantique à l’information, c’est-à-dire que l’information est décrite dans l’URI.

RDF est un langage qui permet de stocker de l’information sous une certaine forme. Il existe plusieurs formats afin de sauvegarder les triplets RDF en un flux d’octets (***sérialisation***), les trois principaux étant ***RDF/XML*** (.rdf), qui est la syntaxe originale mais plus verbeuse que les autres, ***N3*** (.n3) et ***Turtle***(.tll), ce dernier étant un standard W3C. La plupart des outils savent jongler entre ces trois formats, notamment en reconnaissant l’extension.

## Interroger une base de données

Lorsqu’une base de données est créée, il faut aussi savoir l’interroger pour récupérer des informations. Pour cela il faut interroger le serveur à l'aide d'un langage de requête.

Chaque requête permet de chercher des informations dans les ontologies. Le langage le plus connu et le plus utilisé est SPARQL.

***SPARQL*** permet principalement de chercher de l'information dans des bases de données structurées, d’où l’intérêt de structurer l’information.

Dans le cas de corpus de texte non structurés, il est également possible d’extraire de l’information, en utilisant le « machine learning ». On appelle cela le ***text mining*** ou text analytics. La traduction française moins communément utilisée est l’extraction de connaissances ou fouille de textes.

Le text mining permet de transformer un texte non structuré en texte structuré par l’analyse d’une collection de ressources écrites. Il permet d’identifier les faits, les relations entre ces faits ainsi que les assertions.

L’immense avantage de ce concept par rapport à la recherche par mots clefs (keyword search) est qu’il permet de reconnaître des concepts similaires même quand ils sont exprimés différemment.

## Vocabulaire standard

On a dit que pour construire une base de données, celle-ci doit respecter le schéma RDF recommandé par W3C afin de construire un Web normalisé, compréhensible de tous. Ce même, il est également nécessaire d’avoir un langage standard, construit sur le modèle de données RDF. Ce langage, appelé ***Web Ontology Language*** (**OWL**) permet de définir des ontologies [web](https://fr.wikipedia.org/wiki/Web) structurées.

Toujours dans la perspective d’une utilisation universelle des données, des ontologies web écrites avec le standard OWL ont été créées pour être utilisées par tous. Cela permet d’éviter à chacun de créer tout un ensemble d’ontologies lors de la création d’une base de données, mais également cela permet à tout le monde d’avoir les mêmes définitions de concepts.

Voici une liste non exhaustive des principales ontologies,  accompagnées du préfixe communément utilisé. A noter qu’il est possible de choisir n’importe quel préfixe étant donné que c’est une abréviation mais qu’utiliser les préfixes standards permet une compréhension plus universelle de son code.

* Le ***FOAF***, littéralement Friend Of A Friend, permet de décrire des entités. Elle fournit des informations diverses sur des personnes ou choses (quelles qu’elles soient) ainsi que leurs relations.

foaf: <http://xmlns.com/foaf/0.1/>

* Le ***Dublin Core*** permet de décrire des ressources numériques ou physiques (titre, créateur, éditeur, sujet, langue, description, format, date ..)

dc: <http://purl.org/dc/elements/1.1/>

dct : <http://purl.org/dc/terms/>

Par exemple <http://purl.org/dc/elements/1.1/title> est le titre d’un document, d’un livre.

* Le vCard de W3 concerne le monde des affaires.

v :<http://www.w3.org/2006/vcard/>

Par exemple <http://www.w3.org/2006/vcard/title> est l’intitulé du poste d’une personne.

* Les relations entre objets liés au schéma RDF ainsi que la syntaxe des éléments sont décrites par les ontologies suivantes :
* rdf: [http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns)

rdfs: [http://www.w3.org/2000/01/rdf-schema#](http://www.w3.org/2000/01/rdf-schema)

Exemple rdfs :label pour le nom d’une URI quelle qu’elle soit.

* Pour le format d’un element.

xsd: [http://www.w3.org/2001/XMLSchema#](http://www.w3.org/2001/XMLSchema)

Par exemple "4"^^xsd:integer

## Où la donnée est-elle stockée ?

En théorie, on pourrait accéder à toutes les données du [Web](https://fr.wikipedia.org/wiki/Web) avec le langage standard SparQL. Cependant ce n’est pas le cas pour le moment puisque seule une partie des fichiers du Web est standardisée au format RDF.

Néanmoins certaines personnes ou institutions, ont créé des fichiers ayant pour vocation d’être utilisés universellement. Il s’agit donc en quelques sortes de bases de données de ressources et de propriétés libres. Ces fichiers respectent le schéma RDF ainsi que les standards OWL.

Ces système de bases de données (SGBD) relationnelles (RDBMS-Relational Database Management System- en anglais) spécifiques au format des triplets sont appelées Triplestores et sont optimisés pour le stockage et l’extraction de triplets.

Quelques exemples de triplestores, dans un ordre quelconque: OpenLink Virtuoso, AllegroGraph, Stardog, Neo4J, MarkLogic.

Ces Triplestores sont stockées sur internet pour être accessibles, à l’aide de ***SPARQL endpoint***, ou point de terminaison. Cela fonctionne comme pour les moteurs de recherche classiques : on a une URL racine (« base URL ») et on lui ajoute les paramètres de la requête.

Si elles ne sont pas mises à jour, les bases de données deviennent obsolètes. Or mettre à jour une base de données prend du temps et ne peut pas être fait toutes les minutes. Sur chaque point de terminaison est stockée une base de données à une version v, à un instant t. Il est donc important de choisir un point de terminaison récent lorsqu’on souhaite interroger une base de données.

DBpedia possède différents SPARQL endpoints dont l’un est mis à jour régulièrement: <http://live.dbpedia.org/sparql> . DBpedia live est le seul point de terminaison qui mette à jour régulièrement la base en ne mettant à jour que les données modifiées.

Voici d’autres SPARQL endpoints mais dont la mise à jour est en général bien antérieure à celle de DBpedia Live :

<http://dbpedia.org/snorql/>

<http://demo.openlinksw.com/sparql>

<http://librdf.org/query/>

Il est normalement impossible d’interroger une base de données d’un autre point de terminaison que celui que vous utilisez sauf si vous incluez des informations à ce propos par exemple à l’aide de « SparQL federation ». Ce dernier permet d’interroger plusieurs points de terminaison SparQL en même temps pour obtenir un résultat combiné.

## La base de données DBpédia

Le but de DBpédia est d’avoir une version structurée des données disponibles dans Wikipédia. L’équipe de développeur a mis au point un algorithme permettant de parcourir les pages de Wikipédia pour retrouver les informations brutes et les enregistrer dans la base.

Elle a été créée dans le but d’être utilisé par tous librement et automatiquement, notamment pour applications Web.

C’est cette base de données que nous allons interroger avec l’interface, puisque c’est l’une des plus riches existantes.

# Organisation de l’interface

Pour la réalisation de mon interface comme pour la rédaction de mon rapport, j’ai choisi mon vocabulaire. Ce choix est évidemment discutable, notamment car j’ai appris de nouvelles choses au fur et à mesure du projet, mais je m’y suis tenue pour la compréhensibilité de mon rendu.

* J’ai appelé **élément** toute chose qui a une URI et donc une unicité. Il aurait peut-être été plus clair de choisir le mot objet mais celui-ci est déjà réservé pour le triplet RDF.
* J’ai appelé **sujet**, le sujet du triplet, donc la première partie du triplet.
* J’ai appelé **prédicat** le verbe du triplet. Celui qui permet en général d’obtenir l’objet.
* J’ai appelé **objet** la dernière partie du triplet. Celui-ci peut-être une URI, donc un élément, ou un texte brut comme cela sera expliqué par la suite.

## Guide utilisateur

L’interface comprend une page d’accueil qui explique rapidement et simplement à l’utilisateur comment utiliser l’application et quelle est son utilité. Cette explication est accompagnée d’exemples pour aider l’utilisateur à transformer correctement sa question.

IMAGE USER GUIDE

## Requête

Ensuite l’onglet Query permet d’interroger la base de données. Au départ l’interface est au format le plus simple pour aider la compréhension. Ensuite des menus déroulants apparaissent lorsque la sélection s’y prête. Il est également possible de choisir d’ajouter un prédicat.

Le fonctionnement de cet onglet est détaillé dans la partie suivante.

Exemple de requete et ce qui se passe derriere en sparql

## Carte

En cas de requête avec résultats spatialisés vous pouvez les consulter sur une carte. Lorsque les 2 objets présentes une spatialisation , par exemple dans le cas du lieu de naissance et du lieu de mort, les 2 informations apparaissent, le deuxième objet étant affiché par un marqueur orange.

En survolant le marqueur, vous avez le nom du sujet. En cliquant vous avez son nom ainsi que le nom du lieu. A cela est ajouté le lieu de l’objet 2 si celui-ci est spatiailisé.

IMaGE CARTE

IMAGE CARTE 2 objets spatailisés

Expliquer choix latitude latitude\_of-\_the\_place

## Informations supplémentaires

L’onglet « Further information » permet de répondre aux problèmes qui sont souvent rencontrés, qui sont principalement dû à la base de données DBpédia elle-même. Je me suis pour cela basée sur ma propre expérience de l’interface puisque j’ai fait des centaines de tests pendant le développement, mais également sur les retours des personnes a qui j’ai fait testé l’interface.

De plus, si des questions demeurent, une partie « Contact » est prévue à cet effet.

De même des explications plus poussées sur le web sémantique ainsi que le code de l’application sont disponibles sur Github, via un lien dans le même onglet.

# Fonctionnement de l’interface

## Choix par rapport à DBpédia

## SPARQL et l’interface

Après avoir bien compris les concepts du web sémantique ainsi que le potentiel de DBpédia, j’ai pu apprendre à en interroger la base de données. Pour cela, j’ai appris à utiliser le langage SPARQL.

Dans cette partie je vais vous introduire le langage SPARQL en vous faisant découvrir ce qui se cache derrière l’interface.

Une requête SPARQL comprend, dans l’ordre :

* Une déclaration des préfixes qui permettent d’abréger les URI qui sont utilisées pour la définition des ontologies
* Une clause de résultat (SELECT) qui identifie les informations qui doivent être renvoyées
* La requête elle-même (WHERE) qui spécifie les données exactes qu’il faut interroger. Cette requête est formée de plusieurs clauses, organisées selon le triplet *sujet prédicat objet*.
* Des modificateurs de requêtes qui permettent d’ordonner ou de ne prendre qu’une partie des résultats.

Les préfixes que j’ai utilisé pour mon interface sont les suivants :

PREFIX db: <http://dbpedia.org/resource/>

PREFIX dbp: <http://dbpedia.org/property/>

PREFIX dbo: <http://dbpedia.org/ontology/>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX dct: <http://purl.org/dc/terms/>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

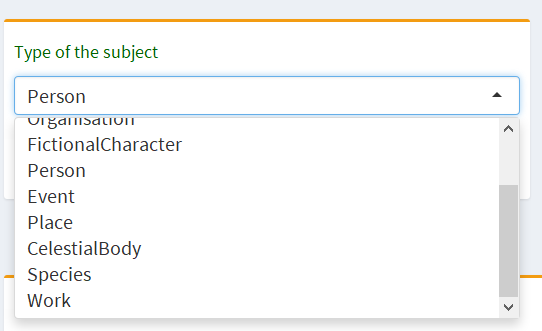
Par la suite les différentes possibilités de l’interface seront détaillées à l’aide d’exemples.

Sélection du sujet qui nous intéresse, à savoir les objets Personne, et de leur label (c’est à dire leur etiquette).

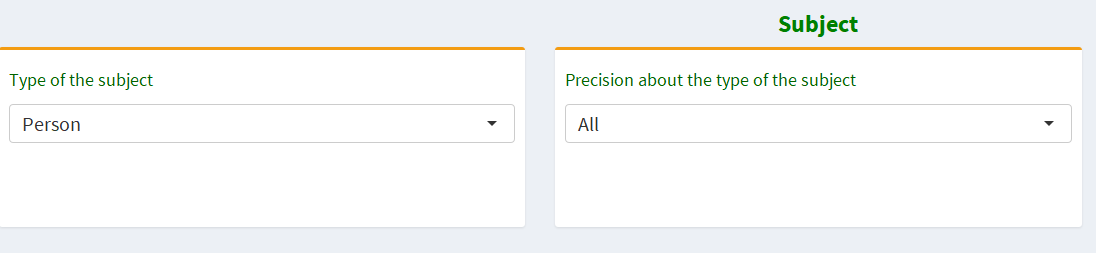
## Sujet

### Type principal

On choisit le type d’ontologie qui nous intéresse.



Par exemple on choisit de chercher des personnes, c’est la catégorie la plus fournie et la mieux remplie de DBpédia.



La requête correspondante est :

SELECT \*

WHERE {

**?x a dbo:Person .**

?x rdfs:label ?y .

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

LIMIT 100

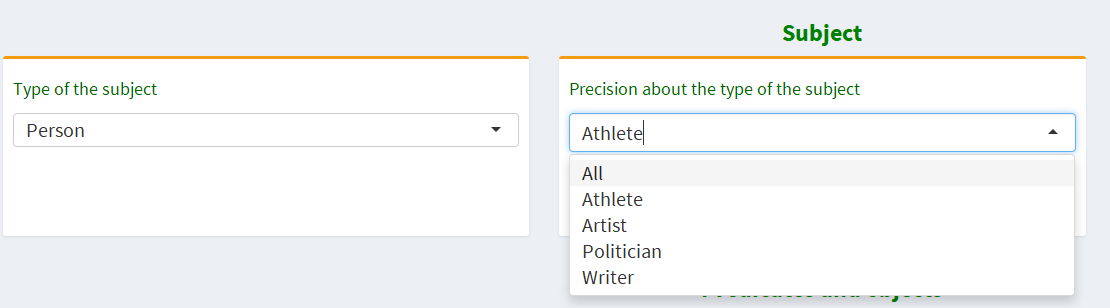
La ligne correspondant particulièrement à ce qu’on est en train d’expliquer est en gras.

Remarques :

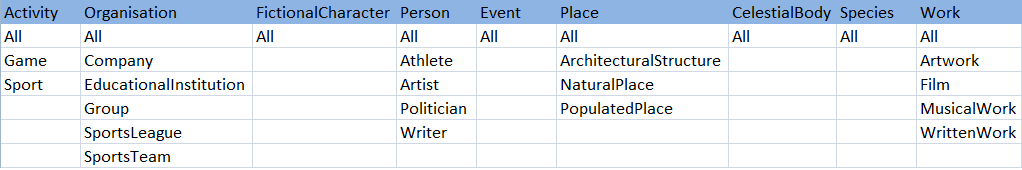
* *?x a dbo:Person* est equivalent à ?x rdf:type dbo :Person. Il s’agit d’un simple raccourci très pratique puisque ce prédicat est très souvent utilisé.
* *?x rdfs:label ?y .* est la ligne permettant de créer un triplet même si on n’a que le sujet. En effet on part du principe qu’on a toujours besoin du nom de l’élément, donc le triplet est introduit automatiquement.
* *BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .* permet l’ajout d’une colonne contenant le lien vers la page wikipédia de l’élément.

### Type secondaire

On peut également vouloir préciser le type de sujet :



Voici les sous-catégories existantes :



La requête correspondante est :

SELECT \*

WHERE {

?x a dbo:Person .

**?x a dbo:Athlete .**

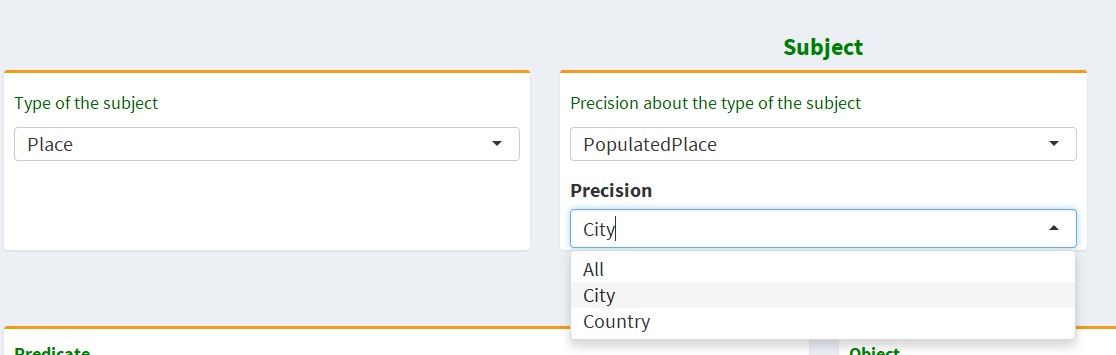
?x rdfs:label ?y .

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

LIMIT 100

Dans le cas des lieux peuplés, c'est-à-dire les pays, régions, villes, villages, hameaux et autres, on peut ajouter une troisième précision : si on veut une ville ou un pays, les autres cas se trouvant dans « All ».



### Précision du nom

On peut également vouloir préciser le nom de l’élément qui nous intéresse.

Par exemple, on veut l’athlète Teddy Riner :

SELECT \*

WHERE {

?x a dbo:Person .

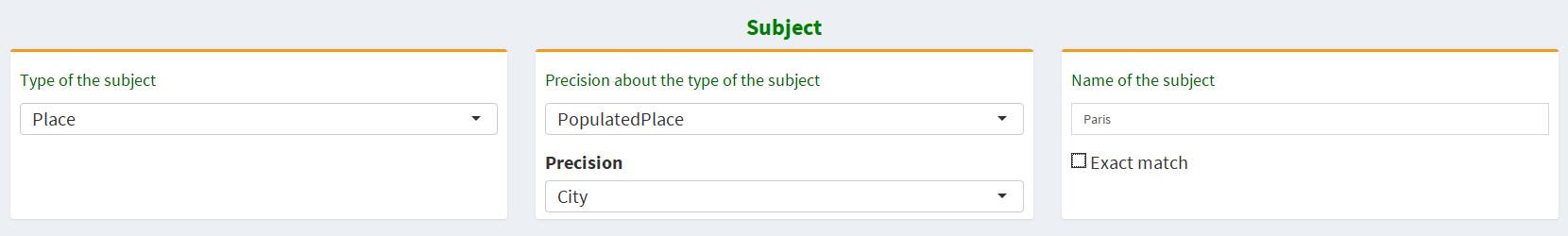
?x a dbo:Athlete .

?x rdfs:label "Teddy Riner"@en .

}

LIMIT 100

Autre exemple, toutes les villes contenant le mot Paris.



SELECT distinct \*

WHERE {

?x a dbo:Place .

?x a dbo:PopulatedPlace .

?x rdfs:label ?name .

BIND(STR(?name) as ?Subject) .

**FILTER(CONTAINS(?name,"Paris")) .**

OPTIONAL{?x georss:point ?coordinates} .

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

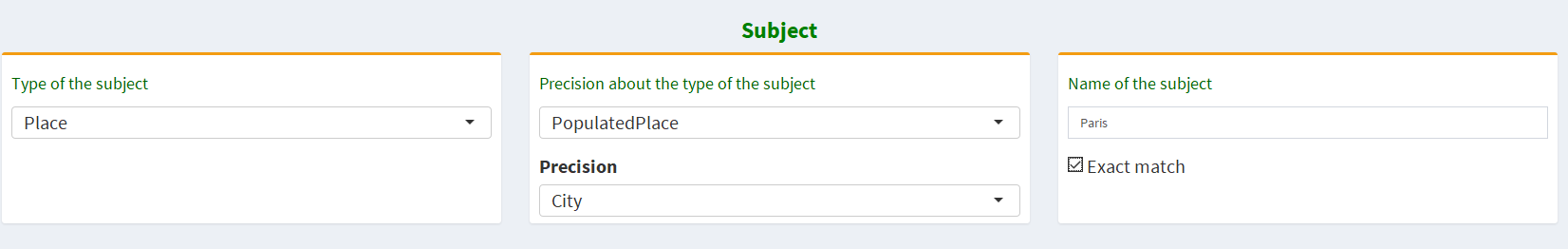
LIMIT 100

Remarque :

* *BIND(STR(?name) as ?Subject) .* permet de transformer l’URI en chaine de caratères pour pouvoir l’étudier.
* Dans le cas de lieu, on ajoute les coordonnées GPS lorsqu’ils existent : *OPTIONAL{?x georss:point ?coordinates} .*

Pour Paris, comme pour d’autres éléments, il existe un grand nombre de résultats. Par exemple « Parish » signifie Paroisse en anglais, et en Estonie beaucoup de ville s’appelle « Paroisse de … ».

Il peut donc être intéressant de chercher le résultat exact dans ce cas là.



SELECT distinct \*

WHERE {

?x a dbo:Place .

?x a dbo:PopulatedPlace .

?x rdfs:label ?name .

BIND(STR(?name) as ?Subject) .

**FILTER(?Subject = STR("Paris"))**

OPTIONAL{?x georss:point ?coordinates} .

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

LIMIT 100

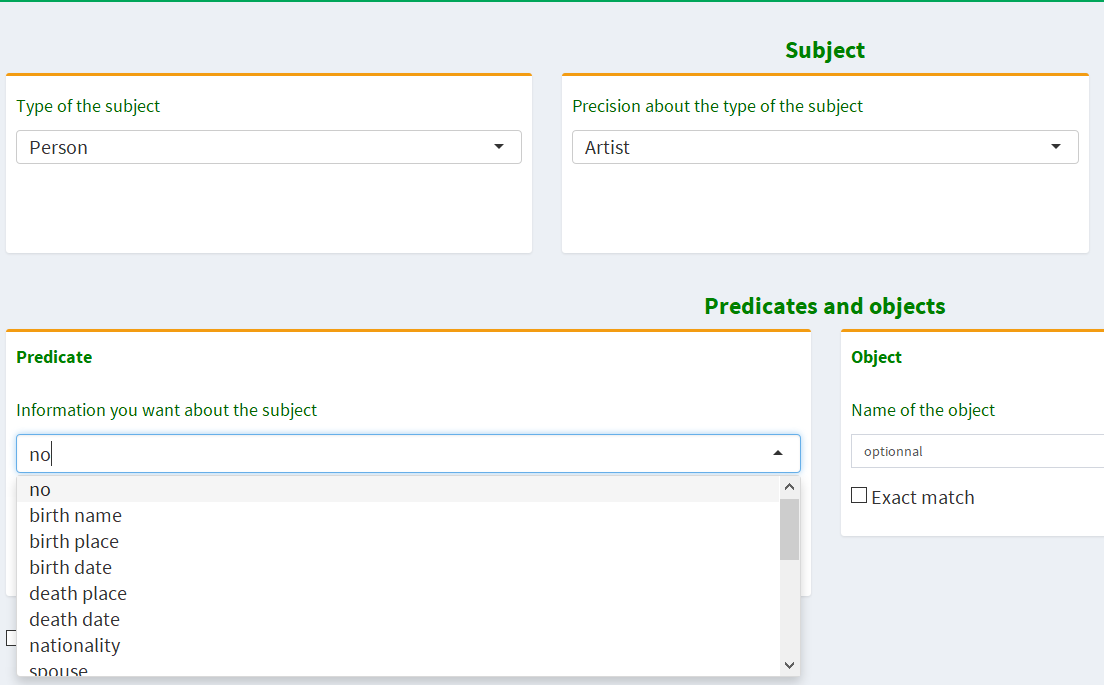
De manière général il vaut mieux éviter ce cocher directement la case car il y a beaucoup d’écritures différentes qui peuvent être perdues. Par exemple « Paris, France » serait perdu. America, en égalité stricte perdrait « United states of America », « American » ou toutes les variantes que les utilisateurs de Wikipédia on peut entrer.

Le cas du sujet simple sert en géneral si on cherche une liste de personnes, ou une liste de lieux ... Dans la plupart des cas, on cherche des informations sur ces éléments, c’est l’intérêt des prédicats.

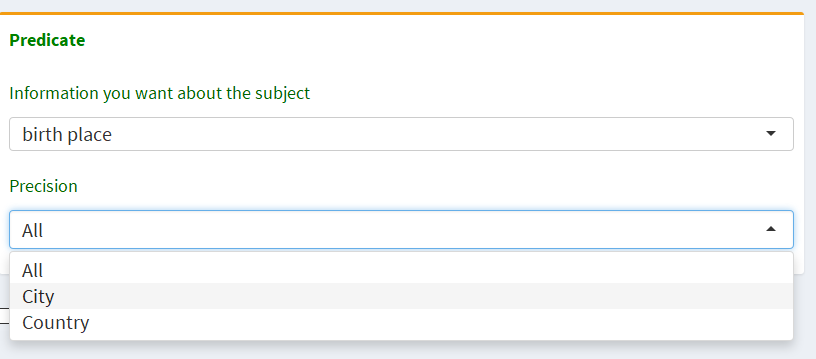
## Prédicat et objet

### Choix du prédicat

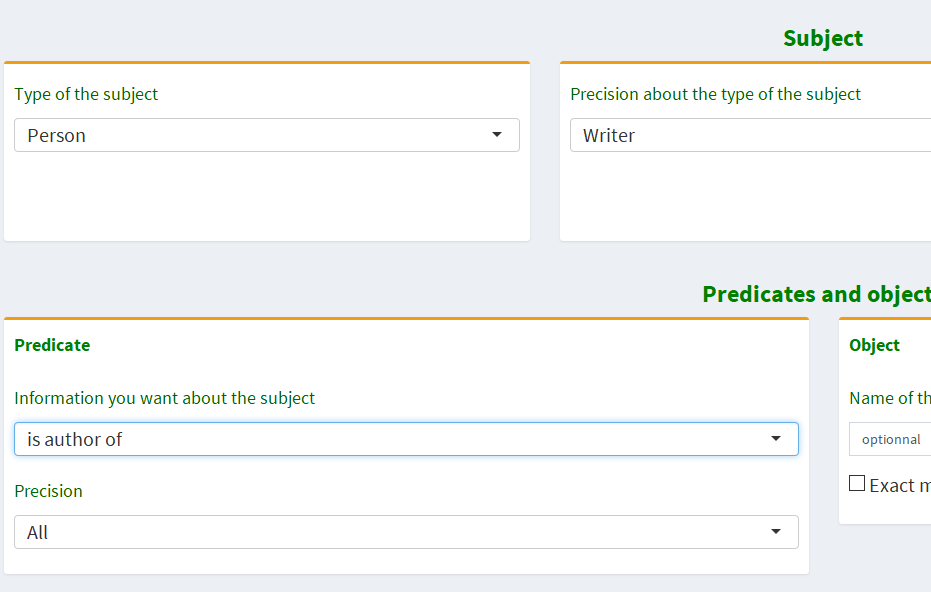
On choisit dans la liste qui nous est proposée. Cette liste varie en fonction du type de sujet qui a été selectionné. Pour voir tous les prédicats par types de sujet, voir l’annexe « Prédicats ».



Dans le cas de prédicats qui renvoient des objets spatialisés on peut, comme pour le sujet, préciser le type de lieu qui nous intéresse.



Voici un exemple avec un prédicat.



SELECT distinct \*

WHERE {

?x a dbo:Person .

?x a dbo:Writer .

?x rdfs:label ?name .

BIND(STR(?name) as ?Subject) .

**?z dbo:author ?x .**

OPTIONAL{ ?z rdfs:label ?nameobjectURI .}

BIND (COALESCE(STR(?nameobjectURI),str(?z)) as ?Object) .

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

LIMIT 100

Deux remarques :

* Le prédicat donné est « is author of », le lien est donc indirect. Ici le sujet est le livre et l’auteur est l’objet. Dans un cas classique, le sujet de l’interface est sujet du triplet. Mais ceci se voit uniquement dans la requête et pas dans l’interface :

?z *dbo:author ?x .*

* Les objets renvoyés par DBpédia peuvent être des URI ou pas (les URI pouvant allant servir de sujet pour un triplet, alors qu’une chaine de caractère par exemple ne le peut pas). Donc on transforme tout en chaine de caractère pour un affichage uniforme :

*OPTIONAL{ ?z rdfs:label ?nameobjectURI .}*

*BIND (COALESCE(STR(?nameobjectURI),str(?z)) as ?Object) .*

Autre exemple, le lieu de naissance de Teddy Riner. Teddy Riner est le sujet, « est né à » est le prédicat et le lieu de naissance est l’objet.

SELECT \*

WHERE {

?x a dbo:Person .

?x a dbo:Athlete .

?x rdfs:label 'Teddy Riner'@en .

**?x dbo:birthPlace ?z .**

OPTIONAL{ ?z rdfs:label ?nameobjectURI .}

BIND (COALESCE(STR(?nameobjectURI),str(?z)) as ?Object) .

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

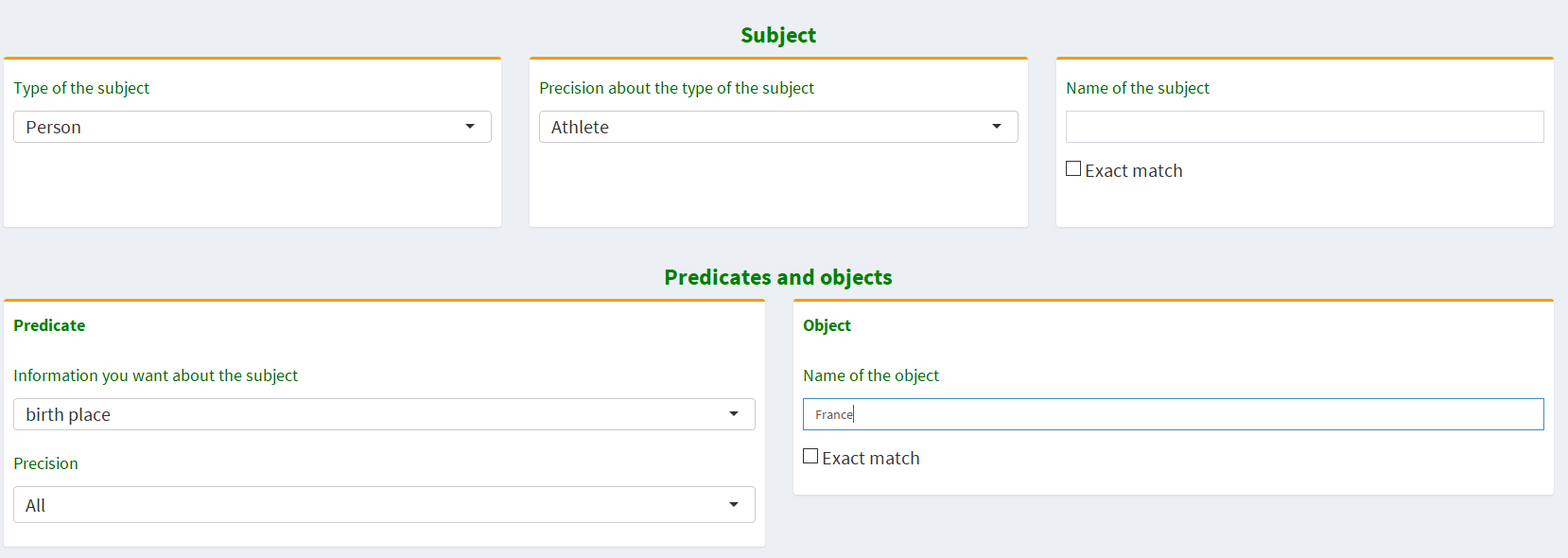
}

LIMIT 100

### Prédicat nommé- choix de l’objet

Dans les cas précédant, ce qui nous intéresse c’est de connaitre l’objet. On connaissait le sujet (Teddy Riner) et on voulait apprendre quelque chose sur lui à l’aide d’un prédicat (lieu de naissance).

Dans d’autres cas, on connait l’objet et on cherche le sujet. Par exemple, on veut tous les athlètes nés en France.



SELECT distinct \*

WHERE {

?x a dbo:Person .

?x a dbo:Athlete .

?x rdfs:label ?name .

BIND(STR(?name) as ?Subject) .

?x dbo:birthPlace ?z .

OPTIONAL{ ?z rdfs:label ?nameobjectURI .}

BIND (COALESCE(STR(?nameobjectURI),str(?z)) as ?Object) .

OPTIONAL{?z georss:point ?place} .

OPTIONAL{?px a dbo:Place .

?px rdfs:label ?z .

?px georss:point ?place} .

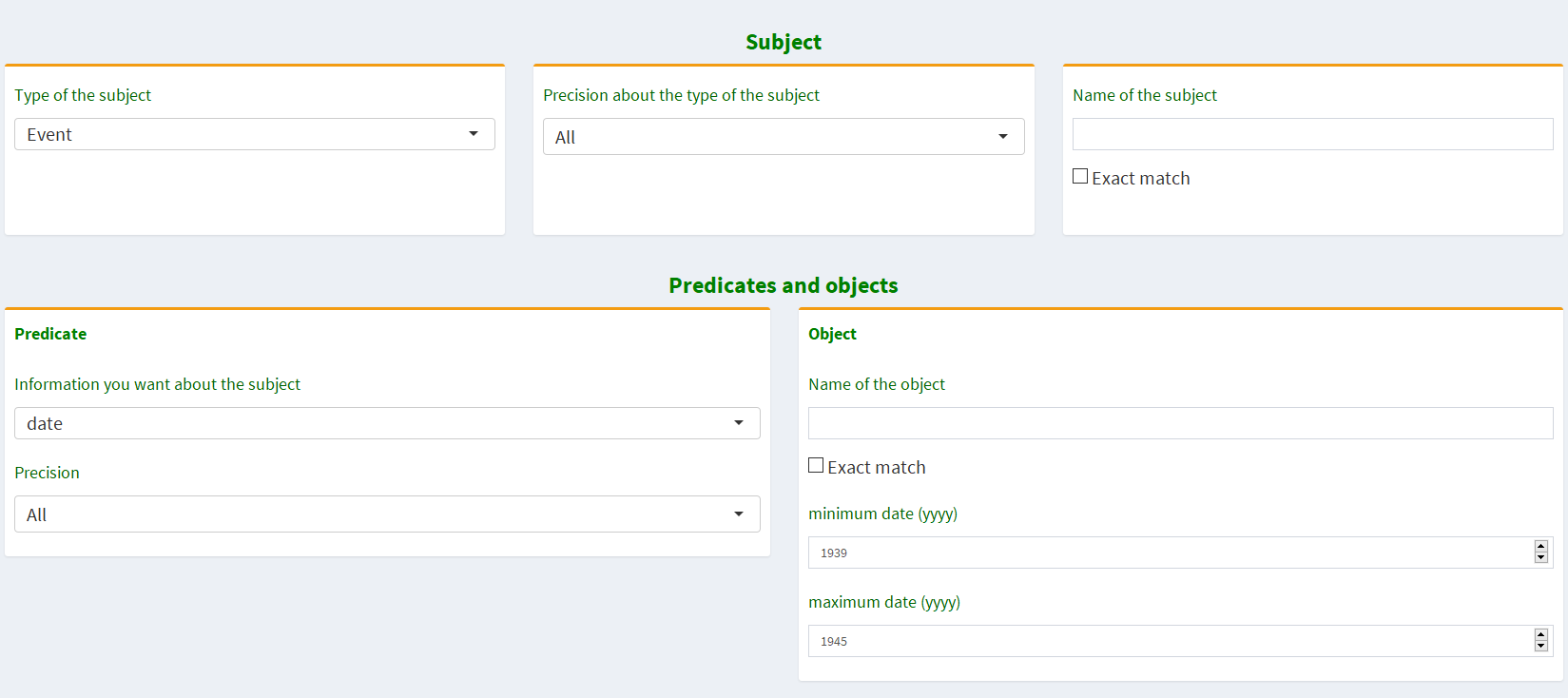
**FILTER(CONTAINS(?Object,"France")) .**

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

LIMIT 100

On peut le faire également avec des dates. Lorsqu’un prédicat permet d’obtenir des objets de type date, une fenêtre apparait pour choisir la période si l’utilisateur le souhaite. On peut entrer l’année minimale, maximale, les deux ou aucun. On peut également ne vouloir qu’une année, dans ce cas on entre l’année dans le champ « Name of the object » ou on l’écrit en minimum et maximum.



SELECT distinct \*

WHERE {

?x a dbo:Event .

?x rdfs:label ?name .

BIND(STR(?name) as ?Subject) .

OPTIONAL{?x georss:point ?coordinates} .

?x dbo:date|dbp:date ?z .

OPTIONAL{ ?z rdfs:label ?nameobjectURI .}

BIND (COALESCE(STR(?nameobjectURI),str(?z)) as ?Object) .

BIND(strdt(?Object,xsd:date) AS ?date).

**FILTER(?date >= "1939-01-01"^^xsd:date).**

**FILTER(?date <= "1945-12-31"^^xsd:date).**

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

}

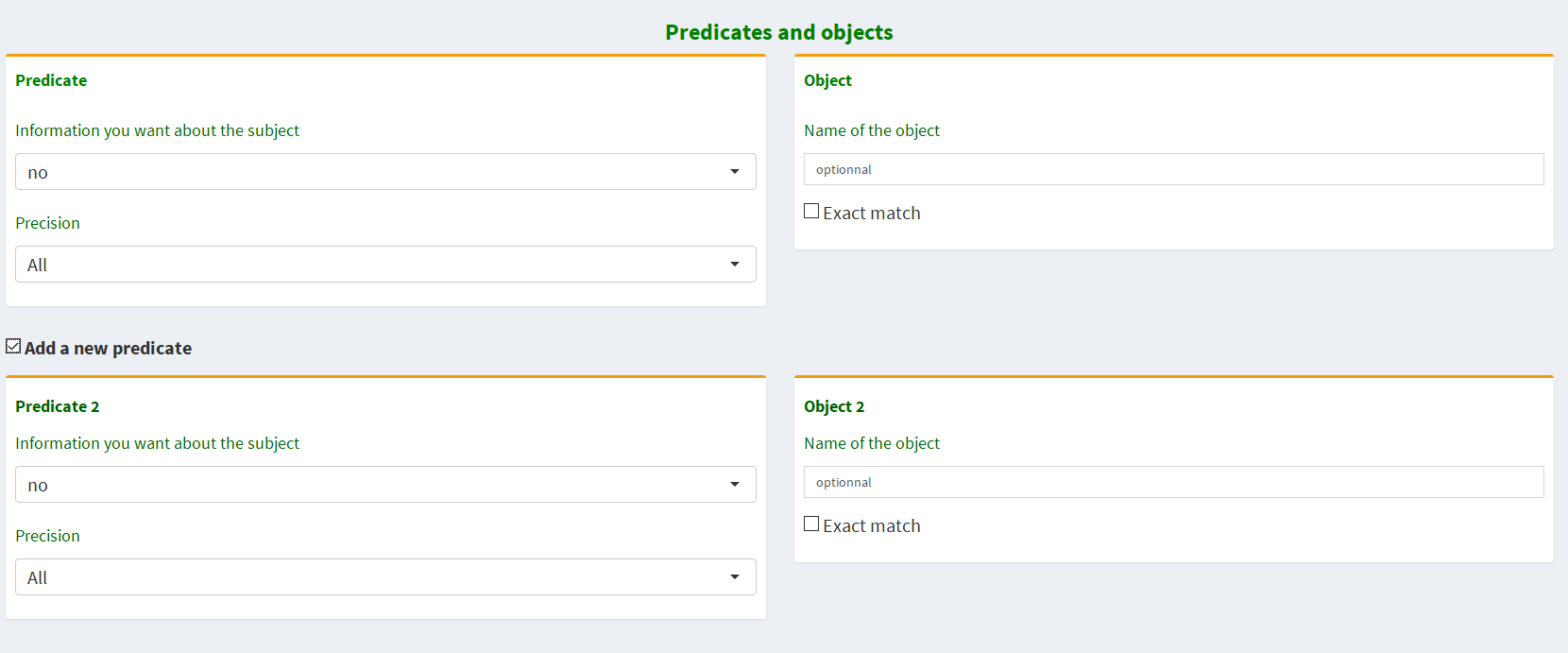
LIMIT 100

### Deuxième prédicat

On peut parfois chercher à connaitre plus qu’une information. C’est ce que permet la case suivante :

predicat_ajout.PNG

Lorsqu’elle est cochée, il est possible de choisir un deuxième prédicat et donc un deuxième objet.



Par exemple, on peut s’amuser un peu et chercher les dates de naissance et de mort de toutes les personnes s’appelant Smith.

SELECT distinct \*

WHERE {

?x a dbo:Person .

?x rdfs:label ?name .

BIND(STR(?name) as ?Subject) .

FILTER(CONTAINS(?name,"Smith")) .

?x dbo:birthPlace ?z .

OPTIONAL{ ?z rdfs:label ?nameobjectURI .}

BIND (COALESCE(STR(?nameobjectURI),str(?z)) as ?Object) .

OPTIONAL{?z georss:point ?place} .

OPTIONAL{?px a dbo:Place .

?px rdfs:label ?z .

?px georss:point ?place} .

?z a dbo:City .

**?x dbo:deathPlace ?zbis .**

**OPTIONAL{ ?zbis rdfs:label ?nameobjectURIbis .}**

**BIND (COALESCE(STR(?nameobjectURIbis),str(?zbis)) as ?Object2) .**

**OPTIONAL{?zbis georss:point ?place2} .**

**OPTIONAL{?pxbis a dbo:Place .**

**?pxbis rdfs:label ?zbis .**

**?pxbis georss:point ?place2} .**

**?zbis a dbo:City .**

BIND (concat("http://wikipedia.org/wiki/",replace(?name," ","\_")) as ?wikilink) .

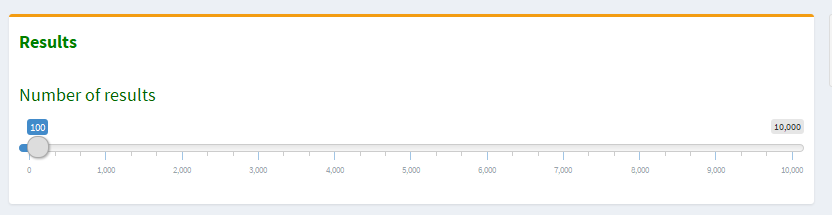
}

LIMIT 100

## Résultats

### Nombre de résultats

On peut bien entendu pour chaque requête, choisir le nombre de résultats souhaités. Cela correspond au « LIMIT n » que l’on trouve à la fin de chaque requête.

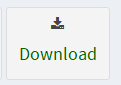


Par défaut, le nombre est de 100 car l’affichage est ainsi plus rapide et que 100 résultats suffisent dans les cas exploratoires.

Il est possible de choisir jusqu’à 10 000 résultats, qui est le maximum donné par DBpédia. Si l’on voulait les résultats suivant ces 10 000 premiers résultats, un « OFFSET » dans la requête serait nécessaire. Je n’ai pas proposé cette fonctionnalité pour le moment car il est rare d’avoir un intérêt à avoir un si grand nombre de résultats.

### Export

Il est possible d’exporter les résultats au format CSV pour pouvoir les réutiliser.



# Spatialisation des résultats

Dans certains cas, les résultats peuvent être spatialisés. J’ai donc ajouter un item « Map » pour permettre de visualiser la localisation des résultats.

Il existe 3 cas :

* Le sujet est localisé. C’est le cas pour les lieux et les événements. Dans ce cas, même si l’objet est spatialisé on gardera uniquement les coordonnées du sujet car ce sont les plus précis.
* Un objet est localisé. C’est le cas lorsque le prédicat choisi est un lieu.
* Deux objets sont localisés. C’est le cas lorsque les deux prédicats choisis sont des lieux.

De la même manière que pour l’onglet query

Un objet spatialisé – carte des Paris

2 objets spatialisés – carte des smith et carte de la 2nde guerre mondiale

# Posts traitements

**Parler des traitements R, post Sparql**

**A supprimer ????**

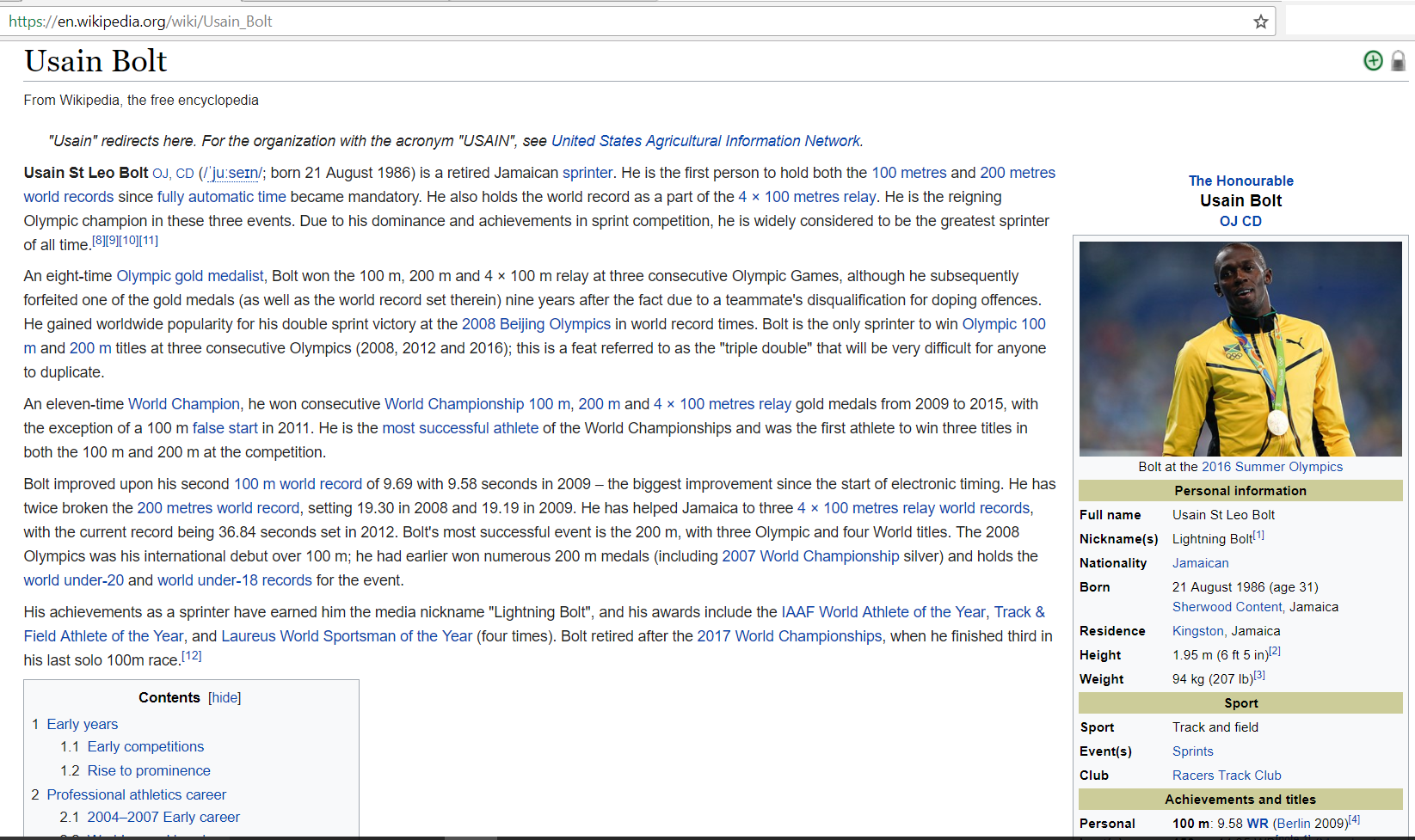
# Potentiel et limites

## Problèmes liés à DBpédia

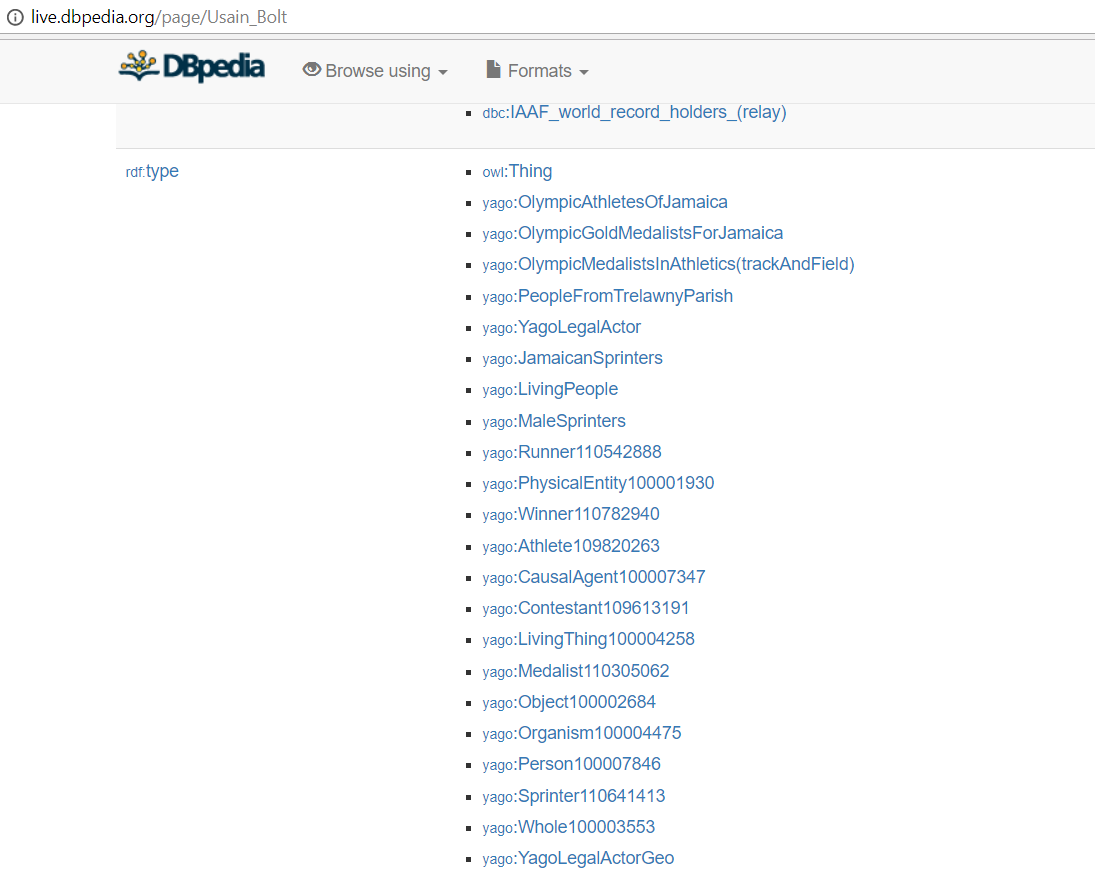
La base de données de DBpédia est remplie grâce à l’extraction automatique de données sur Wikipédia. C'est-à-dire que l’équipe en charge de son remplissage a mis au point un algorithme qui tente de récupérer la donnée brute dans le texte. Cela fonctionne généralement très bien mais il y a néanmoins de nombreux problèmes que mon interface ne peut pas gérer puisqu’ils viennent de la source.

**Problème d’extraction de la nature**

Par exemple voici les pages d’Usain Bolt sur Wikipédia et sur live.dbpedia.org, consultées le 5 février 2018.



On peut voir dès la première ligne la mention de « Jamaican sprinter », qui n’a pas été détectée dans Live DBpédia. Comme expliqué précédemment, on se base sur live DBpédia car il est mis à jour plus régulièrement. Le champs rdf :type est celui qu’on utilise pour savoir la nature d’un élément. On devrait y voir apparaitre « dbo :Person » et « dbo :Athlete ».



**Mauvais classement de la donnée**

De même, on peut avoir des éléments mal reconnus et qui n’apparaissent donc pas dans la bonne colonne de l’interface. Par exemple, la date de naissance d’Émile Rigaud apparait dans la colonne du nom de naissance.

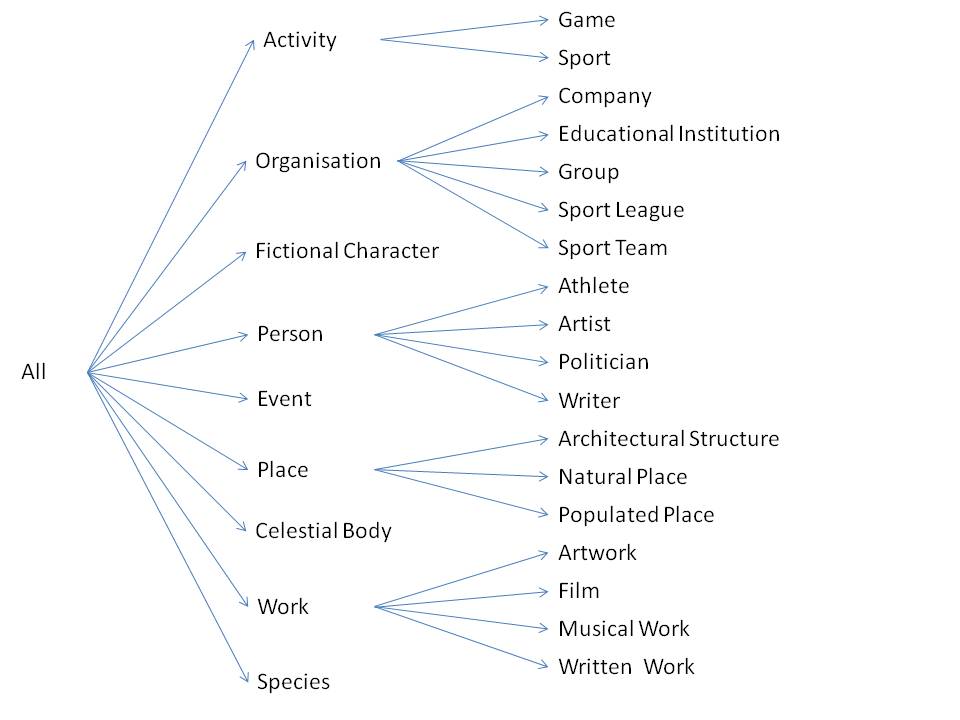
**Pas de formatage de la donnée**

Il n’y a pas de règle de formatage pour les éléments de DBpédia. Par conséquent on peut trouver des Uri, tout comme des chaines de caractères. On peut également trouver différentes écritures pour le même élément : Les États-Unis peuvent être appelé « United States of America », with a o or a O, « America ». De même on pourrait trouver « Paris », « Paris,France » .. C’est pour cela que la case « Exact match » n’est pas cochée automatiquement dans l’interface. Elle permet juste d’aider l’utilisateur dans des cas où le nombre de résultats comprenant sa recherche est trop nombreux.

## Choix d’extraction des ontologies

En raison des nombreux problèmes de « rdf :type », j’ai choisi de ne pas travailler avec l’ensemble des ontologies de DBpédia (listées ici : <http://mappings.dbpedia.org/server/ontology/classes/>).

Cela aurait pour majeur inconvénient de perdre l’utilisateur parmi toutes les natures que peuvent prendre un élément. J’ai donc choisi de choisir les natures suivantes :



Ce réseau est donc bien plus clair et ne prend pas en compte des natures en général très peu utilisées. Ces natures sont néanmoins accessibles en sélectionnant l’ensemble des types de sujet.

Cependant l’inconvénient de cette extraction est que le schéma choisi est statique. Si de nouvelles classes sont crées dans DBpédia (ce qui arrive rarement car nécessite un nouveau type de choses), elles ne seront pas prises en compte dans mon schéma. Cela nécessite donc un maintien à jour futur de ce document (format CSV).

De même j’ai choisi des prédicats pour chaque type d’élément, il n’est pas possible d’en récupérer d’autres. J’ai fait ce choix car il n’y a pas de liste exhaustive des prédicats possibles puisqu’on peut toujours en ajouter, cependant seuls quelques uns sont communs à la plupart des éléments d’une catégorie. Cela s’élève à 80 prédicats environ.

Malgré les ontologies de référence permettant de limiter le problème, il existe toujours plusieurs façon de décrire un élément. Par exemple, on trouve les ontologies dbp:nationality et dbo:citizenship pour décrire la nationalité d’une personne. Dans tous les cas similaires, c'est-à-dire ceux où il n’y a pas un prédicat majoritaire, j’ai choisi de mettre les deux, à l’aide de l’opérateur « OU » logique. Ainsi, le résultat sera rempli avec le prédicat non vide.

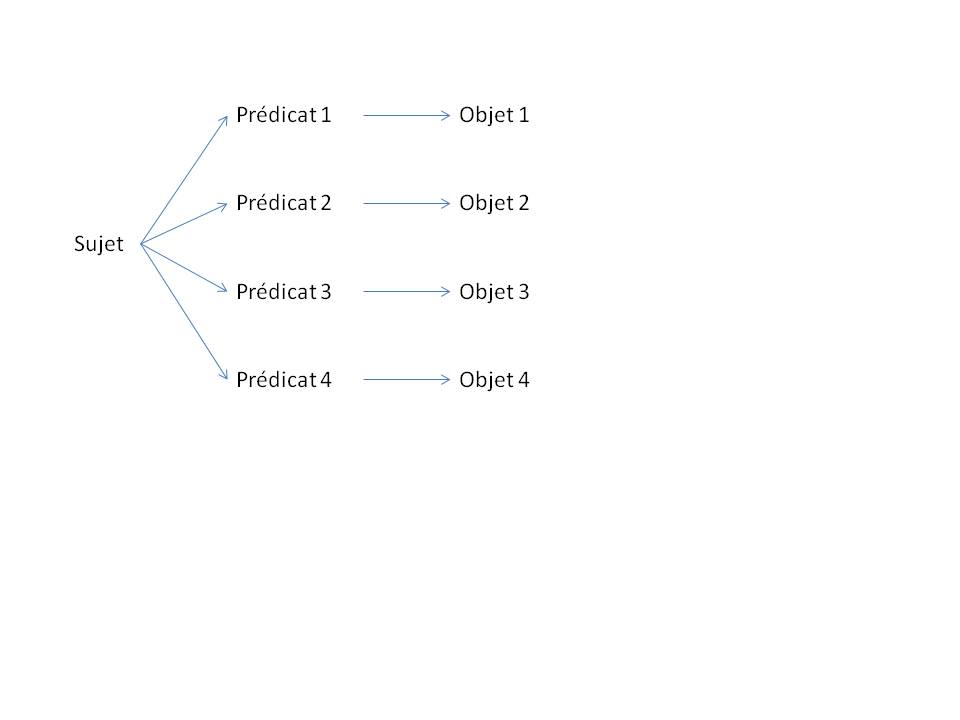
Tous les fichiers décrivant les ontologies et les prédicats choisis, ainsi que le dictionnaire créé pour une meilleure compréhension des prédicats dans l’interface sont disponibles en annexe au format csv.

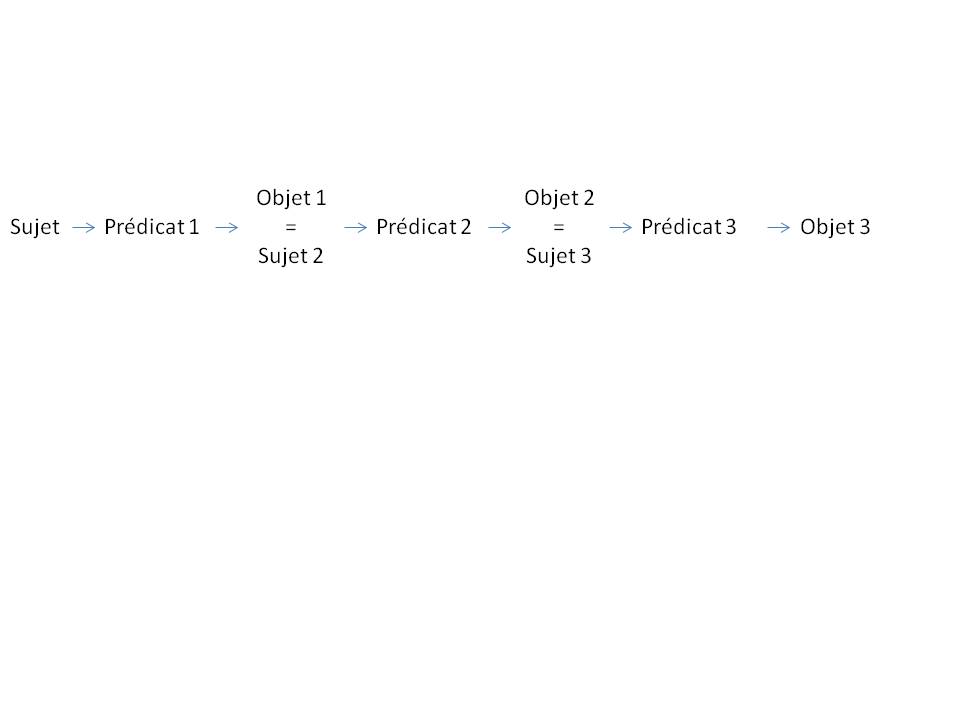
## Choix des délimitations du projet - Perspectives

Les combinaisons possibles de triplets n’ayant de limite que l’imagination, il a été nécessaire de délimiter le projet.

Il existe principalement deux cas de figures d’enchainement de triplets :

* Le premier consiste à construire les requêtes en arbre :



* Le second consiste à rendre l’objet sujet d’un nouveau triplet :

Ces deux cas de figures peuvent évidemment être tous les deux utilisés pour une même requête.

Etant donné que le format de la requête doit être automatisé, et que l’interface doit être organisée clairement, il n’est pas facile de laisser libre choix à l’utilisateur. De plus, entre autres problèmes techniques, un sujet nécessite d’être une URI alors qu’un objet ne l’ai pas forcément. Par conséquent, le deuxième cas de figure peut se révéler difficile à traiter.

Enfin, nous avons estimé qu’avoir la possibilité d’utiliser deux prédicats est en général suffisante pour l’utilisateur et plus propice à donner des résultats (les prédicats ne sont pas toujours remplis, voir « Potentiels et limites »).

Néanmoins, il est possible de spécifier le nom de l’objet. Il s’agit donc d’un objet sujet d’un autre triplet, mais cela fonctionne uniquement pour ce prédicat.

Il serait donc intéressant d’ajouter ces fonctionnalités de prédicats multiples et d’objet-sujet par la suite.

## Potentiel de l’application

# Conclusion

Perspectives :

Ajouter le cas de figure objet sujet

# Annexes

## Annexe : Prédicats

## Annexe : GANTT

Annexe : tous les csv