RI et graphe de données - TP SPARQL

L'objectif de ce TP est de continuer le développement d'une application de Quizz qui interroge Wikidata pour générer des questions de culture générale.

1 Description de l'application

1.1 IHM

Voici un exemple d'interaction :

```
$ python main.py
Prise en compte du module question_reponses.addition
Prise en compte du module question_reponses.capitales

Domaine: Géographie
Question: Quelle est la capitale du (de la) Fidji?
  Réponse 1: Suva
  Réponse 2: Kinshasa
  Réponse 3: Rabat
  Réponse 4: Séoul
Entrez le numéro de la réponse (-1 pour sortir): 1
Bonne réponse!

Domaine: Mathématiques
Question: Combien font 94 + 98 ?
  Réponse 1: 194
  Réponse 2: 192
  Réponse 3: 191
  Réponse 4: 193
Entrez le numéro de la réponse (-1 pour sortir):
```

1.2 Description du code

Le programme est composé des modules suivants :

— `question_reponses.question_reponses` : contient la classe abstraite `QuestionReponses` et le protocole (interface) `QuestionReponsesFactory` :

  — `QuestionReponses` est composée d'une question d'un domaine avec plusieurs réponses possibles, dont l'une est la bonne réponse. Les sous-classes concrètes devront appeler le constructeur de cette classe en lui donnant la liste des réponses possibles ainsi que l'indice de la bonne réponse. Elles devront, de plus, concrétiser les propriétés `domaine` et `question`.

  - `QuestionReponsesFactory` définit la méthode `question_reponses`, qui est capable de retourner une instance de type `QuestionReponses`. Les classes implémentant ce protocole retourneront des instances de `QuestionReponses` sur un domaine donné.
- `question_reponses.addition` : contient la classe `QuestionReponsesAddition`, qui propose des questions sur les additions, et la classe `QuestionReponsesFactory`, dont la méthode `question_reponses` retourne une instance de `QuestionReponsesAddition` avec des opérandes comprises aléatoirement entre 1 et 100 ;
- `question_reponses.sparql` : contient la classe abstraite de fabrique `QuestionReponseFactorySparql`, qui définit quelques méthodes (préfixes des requêtes SPARQL, point d'entrée, transformation de `LangString` RDF en `str`) facilitant la création de fabriques `QuestionReponses` utilisant des données externes interrogeables en SPARQL ;
- `question_reponses.wikidata` : contient la classe abstraite de fabrique `QuestionReponsesFactoryWikidata`, sous-classe de `QuestionReponseFactorySparql`, qui précise les préfixes SPARQL pour Wikidata et son point d'entrée ;
- `question_reponses.capitales` : contient la classe `QuestionReponsesCapitales` et la classe `QuestionReponsesFactory`, sous-classe de la classe `QuestionReponsesFactoryWikidata`. La méthode `question_reponses` de cette dernière classe retourne une instance de `QuestionReponsesCapitales`. À l'instanciation de la fabrique, tous les pays ayant une capitale déclarée dans Wikidata sont récupérés à l'aide d'une requête SPARQL (on récupère le nom du pays en français, le nom de la capitale et ses coordonnées GPS). Des instances de la `dataclass` `PaysCapitale` sont alors créées. Lorsque la méthode `question_reponses` est exécutée, une instance de la classe `QuestionReponsesCapitales` est créée avec cette liste de pays. Cette instance choisit un pays au hasard et sélectionne des capitales les plus proches comme mauvaises réponses.

Enfin, le script `main` contient :
- la classe `QuestionReponsesAleatoireFactory`, dont la méthode `question_reponses` choisit aléatoirement une fabrique de questions et invoque sur cette fabrique la méthode `question_reponses`. Ces fabriques de questions sont recherchées dans les modules du package `question_reponses`, les modules qui possèdent une classe nommée `QuestionReponsesFactory` ;
- la fonction `poser_question`, qui affiche une question à partir d'une instance de la classe `QuestionReponses` ;
- la fonction `verifier_reponse`, qui demande à l'utilisateur de choisir une réponse et qui vérifie si c'est la bonne. En choisissant la réponse `-1`, l'utilisateur peut sortir du programme ;
- la fonction `main`, qui est le programme principal. Elle instancie la fabrique `QuestionReponsesAleatoireFactory`, et une boucle infinie récupère une question, invoque la fonction `poser_question` et la fonction `verifier_reponse`.

2 Ajout d'un type de question

L'objectif est d'ajouter un type de question. Choisissez un thème :
- Géographie (préfecture des départements, longueur des fleuves, dans quelle mer se jette un fleuve, etc.)
- Histoire (siècle de la date naissance ou de décès d'un Roi Français, pays impliqués dans une bataille, etc.)

— Sport (champion olympique, gagnant de la league 1, tournoi des 6 nations, formule 1, etc.)
— ...

2.1 Requête SPARQL

En utilisant l'interface Web d'interrogation SPARQL de Wikidata
(`https://query.wikidata.org/`), trouvez la ou les requêtes SPARQL permettant de gérérer des données servant à créer vos futures questions.
Les URI de Wikidata suivent la nomenclature suivante :
— `http://www.wikidata.org/entity/` (préfixe `wd`) pour les ressources ;
— `http://www.wikidata.org/prop/direct/` (préfixe `wdt`) pour les propriétés.
Il est à noter que Wikidata représente la notion d'instance par la propriété `wdt:P31` (et non `rdf:type`).

2.2 Un thème en plus

Développez un module python permettant d'ajouter des questions sur le thème choisi. Ce module devra contenir :
— une classe `QuestionReponsesXXX` sous classe de `QuestionReponses` qui implémente les méthodes abstraites de la classe mère ;
— une classe `QuestionReponsesFactory` sous classe de `QuestionReponsesFactoryWikidata` qui retournera une instance de `QuestionReponsesXXX`. C'est cette fabrique qui interrogera WIkidata pour récupérer les données (utilisation possible d'une dataclasse) utilisées par `QuestionReponsesXXX`.

2.3 Plusieurs thèmes

Échanger vos modules de façon à avoir un Quizz avec plusieurs thèmes.

Annexe : cas particulier de Wikidata

L'objectif de cette annexe est présenter les particularités du modèle de Wikidata à partir d'un exemple.

Le modèle

Wikidata permet d'avoir plusieurs valeurs pour une propriété. Il est de plus possible de contextualiser, de justifier les déclarations (*statement*). Par exemple, comme le montre la figure 1, il est possible d'avoir plusieurs fois la même information, ici l'altitude du sommet du Mont Blanc, dans plusieurs contextes, ici la date, avec quelques fois des précisions concernant cette mesure.

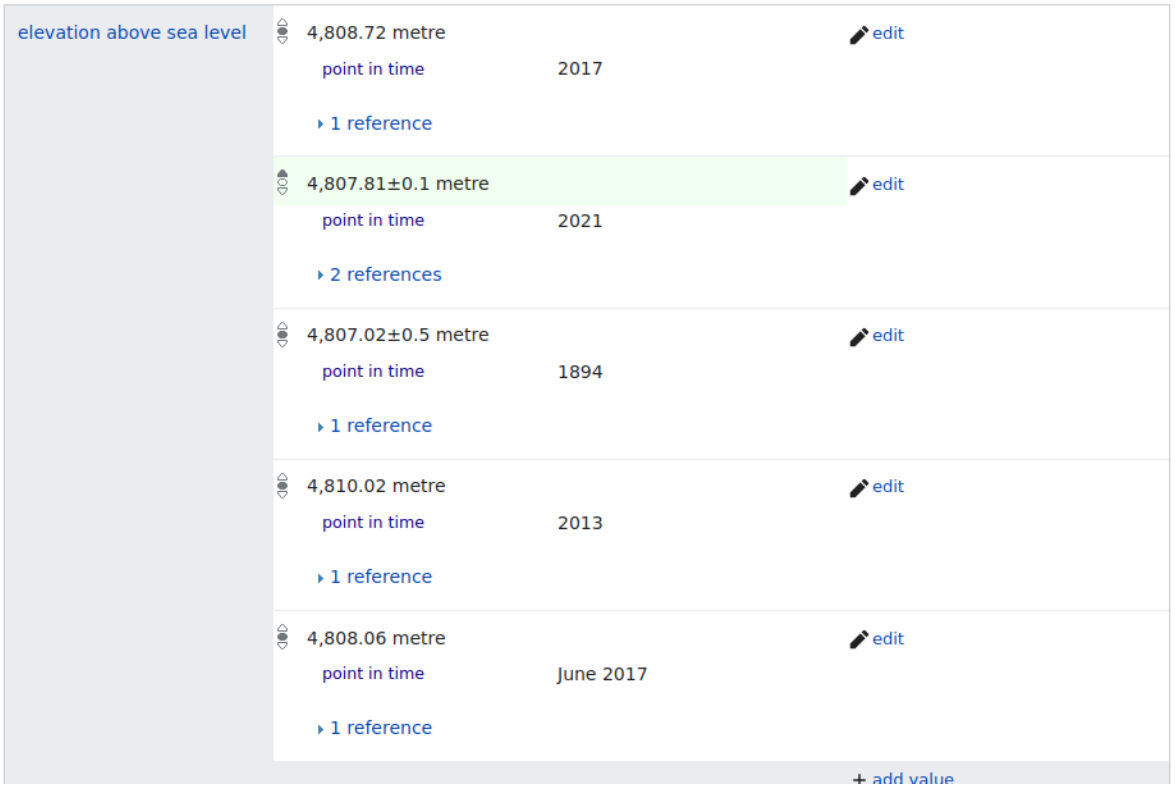

FIGURE 1 – Extrait de la fiche Wikidata du Mont Blanc

Pour récupérer la hauteur du Mont Blanc, la requete SPARQL standard est la suivante :

```
select ?altitude where {
  wd:Q583 wdt:P2044 ?altitude
}
```

Mais dans ce cas on n'obtient qu'une seul résultat, 4807.81, qui correspond à la valeur la plus récente. Comment obtenir toutes les valeurs, et comment obtenir la date d'obtention de ces valeurs et si possibles la précision des mesures ?
En fait il faut savoir que le modèle interne de Wikidata n'est pas du RDF [1]. Mais Wikidata expose bien ses données au format RDF. Il y a donc un mapping qui permet d'exprimer le modèle interne

1. `https://www.mediawiki.org/wiki/Wikibase/DataModel` présente le modèle de Wikidata

Wikidata en RDF. Ce mapping est décrit par le figure 2 [2].

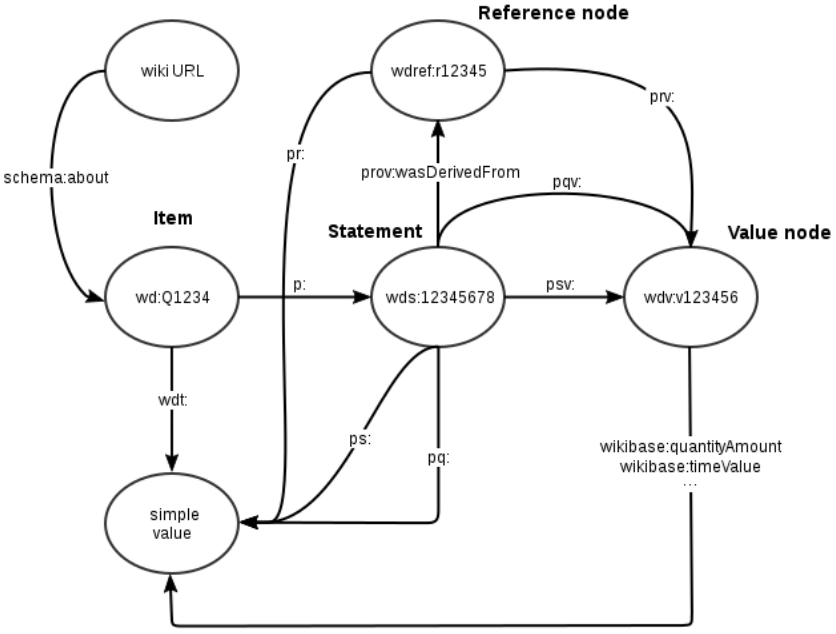

FIGURE 2 – Le mapping RDF du modèle de Wikidata

Une propriété Wikidata d'une entité du modèle interne de Wikidata est décrite à l'aide de n propriétés RDF :

— la première est de préfixe `wdt`. Cette propriété RDF, dite directe, référence une valeur qui est très souvent la plus récente, ou celle qui fait consensus. Dans le cas du Mont Blanc, c'est cette valeur que l'on a obtenu précédemment;

— les autres propriétés RDF permettent d'accéder à toutes les valeurs contextualisées de la propriété Wikidata (pour le Mont Blanc les autres altitudes avec la précision, quand elle est disponible, et la date de mesure). Ces propriétés RDF ont comme préfixe `p` et la même partie droite que l'URI en préfixe `wdt` (si le première propriété est d'URI `wdt:XX`, ces propriétés ont comme URI `p:XX`). Elles permettent d'accéder à une partie du graphe qui va contextualiser la valeur (littéral ou URI) de la propriété RDF `wdt`. Chacune donne accès à un noeud de préfixe `wds` (le *s* signifie le *statement*). Plusieurs propriétés sont issus de ce nœud, leurs préfixes indiquent leur rôle:

  ps permet d'accéder à la valeur de la donnée sans autre information (tel que l'unité par exemple). La partie droite de cet URI est identique à celle de l'URI en préfixe `wdt` (`ps:XX`):

  pq permet d'accéder à la valeur qui contextualise la donnée sans autre information. L'identifiant de l'URI de cette propriété dépend la nature du contexte;

  psv permet d'accéder à un nœud qui précise le type de la valeur de la donnée (par exemple pour une longueur le mètre) et des valeurs associées (comme par exemple son imprécision);

  pqv permet d'accéder à un noeud qui précise le type de la valeur qui contextualise la donnée.

Les nœuds accessibles à l'aide des propriétés en préfixe `psv` et `pqv` ont comme préfixe `wdv` (*v* comme *value*). Depuis ces nœuds, des propriétés permettent d'obtenir des informations sur les valeurs de la propriété Wikidata et les valeurs qui contextualise la propriété Wikidata.

2. Source : `https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format`

Exemple d'utilisation

Par exemple si on veut récupérer toutes les altitudes du sommet du Mont Blanc avec les dates des relevés, on doit utiliser la requête suivante:

```
select ?altitude ?date where {
  wd:Q583 p:P2044 ?statement.
  ?statement ps:P2044 ?altitude.
  ?statement pq:P585 ?date.
}
```

On obtient:

```
4808.06 1 juin 2017
4808.72 1 janvier 2017
4810.02 1 janvier 2013
4807.81 1 janvier 2021
4807.02 1 janvier 1894
```

Si on veut connaitre en plus les unités des altitudes:

```
select ?altitude ?unite ?date where {
  wd:Q583 p:P2044 ?statement.
  ?statement ps:P2044 ?altitude.
  ?statement pq:P585 ?date.
  ?statement psn:P2044 ?statementAltitude.
  ?statementAltitude wikibase:quantityUnit ?unite
}
```

On obtient alors:

```
4808.06 wd:Q11573 1 juin 2017
4808.72 wd:Q11573 1 janvier 2017
4810.02 wd:Q11573 1 janvier 2013
4807.81 wd:Q11573 1 janvier 2021
4807.02 wd:Q11573 1 janvier 1894
```

...sachant que `wdt:Q11573` est l'URI du mètre.
Si enfin on veut connaitre la précision de la mesure de l'altitude, lorsque elle est disponible, on peut obtenir la borne min et la borne max:

```
select ?altitude ?borneMinAltitude ?borneMaxAltitude ?unite ?date where {
  wd:Q583 p:P2044 ?statement.
  ?statement ps:P2044 ?altitude.
  ?statement pq:P585 ?date.
  ?statement psn:P2044 ?statementAltitude.
  ?statementAltitude wikibase:quantityUnit ?unite.
  OPTIONAL {
    ?statementAltitude wikibase:quantityLowerBound ?borneMinAltitude.
    ?statementAltitude wikibase:quantityUpperBound ?borneMaxAltitude
 }
}
```

On obtient alors:

```
4807.02 4806.52 4807.52 wd:Q11573 1 janvier 1894
4807.81 4807.71 4807.91 wd:Q11573 1 janvier 2021
```

```
4808.72   wd:Q11573 1 janvier 2017
4808.06   wd:Q11573 1 juin 2017
4810.02   wd:Q11573 1 janvier 2013
```