Compte rendu du projet

Web sémantique - Web des données

Romain BARBIER,
Elsa BERNET, Lucas SUBE

Sommaire

1 Introduction

GitHub du projet : https://github.com/Romb38/ERL_WEBSEM

Nous avons choisi la musique comme thématique de nos données. Les données viennent de deux sources :

- L'API de Deezer
- L'API de MusicBrainz

Ces deux sources envoient des informations dans un format JSON. Notre premier travail a consisté à définir un modèle de donnée (Partie 2) puis d'implémenter un script python pour transformer les données JSON dans le format Turtle en suivant notre modèle de données (Partie 3).

Ensuite, nous avons importé le fichier Turtle dans une instance de GraphDB installée en local pour visualiser les données.

Enfin, nous avons interrogé nos données avec quelques requêtes SPARQL (Partie 4).

2 Modèle de données

2.1 Ontologies

Nous avons choisi les ontologies suivantes :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix owl: <http://www.w3.org/2002/07/owl#>/ .
```

```
@prefix erl: <http://www.websem.csv/resource/> .
@prefix erlo: <http://www.websem.csv/ontology/> .
```

Le préfixe **erl** correspond aux ressources de notre système. Le namespace associé n'existe pas vraiment et agit ici de remplacement.

2.1.1 Notre ontologie : ERLO

Durant ce projet, nous nous sommes rendu compte que le namespace `purl.org` n'était plus disponible. Nous avons donc décidé d'utiliser la nôtre à la place. Notre ontologie contient les définitions suivantes

- `erlo :compose` : Lie un dbr:Artist à une dbr:Song
- `erlo :produce` : Lie un dbr:Artist à un dbr:Album

2.2 Les ressources

Depuis l'API de Deezer, nous avons décidé de récupérer les titres des 100 premiers artistes dans l'ordre des identifiants donnés par Deezer. À cela, nous avons couplé l'identité légale de chaque artiste que nous retrouvons sur MusicBrainz

2.2.1 Un artiste

La déclaration d'un artiste ressemble à ceci :

```
erl:{ID_Artist_Name} a foaf:Person
    dbo:artist dbr:{ID_Artist_Name} ;
    foaf:givenName "{artist_legal_firstname}" ;
    foaf:familyName "{artist_legal_lastname}" ;
    owl:sameAs dbr:{ID_Artist_Name} ;
    erlo:produce erl:{UUID_ALBUM} ;
    erlo:compose erl:{UUID_SONG}.
```

Ou **ID_Artist_Name** est le nom de scène de l'artiste avec les espaces remplacés par des underscores. Cet ID nous permet de relier nos données à la base de DBpédia

2.2.2 Un album

La déclaration d'un album ressemble à ceci

```
erl:{UUID_ALBUM} a dbo:Album
    rdfs:label "{album name}" ;
    foaf:Person erl:{ID_Artist_Name} ;
    dbo:genre erl:{KIND_LABEL} ;
    erlo:contains erl:{UUID_SONG} .
```

Ou l'**UUID_ALBUM** est un nombre généré aléatoirement par notre algorithme

2.2.3 Un titre

La déclaration d'un titre ressemble à ceci :

```
erl:{UUID_SONG} a dbo:Song
    foaf:Person erl:{ID_Artist_Name} ;
    rdfs:label "{song_name}" ;
    dbo:featuredArtist erl:{ID_Featured_Artist} .
```

Ou l'**UUID_SONG** est un nombre généré aléatoirement par notre algorithme

Romain BARBIER,
Elsa BERNET, Lucas SUBE

2.2.4 Un genre

La déclaration d'un genre ressemble à ceci

```
erl:{KIND_LABEL} a dbo:genre
    rdfs:label "{Kind label}" ;
    owl:sameAs "{Kind label}" .
```

3 La récupération des informations

Pour ce projet, nous avons décidé de créer un algorithme en python pour créer notre fichier Turtle. Cet algorithme fonctionne en 4 étapes :

1. On récupère les informations des 100 premiers chanteurs (par ordre des ID) via l'API de Deezer. Cela nous fournit un fichier JSON avec toutes les informations sauf l'identité légale des chanteurs. Cette étape est assez longue, car Deezer accepte au maximum 50 requête par seconde, nous avons donc dû mettre un petit temps de latence entre chaque requête pour bien tout récupérer.
2. On récupère l'identité légale, si elle existe, des chanteurs grâce à l'API de MusicBrainz
3. On fusionne nos données et on les transforme pour créer les objets Turtle qu'on a défini dans la partie 2.2
4. On écrit les objets dans un fichier en respectant la réflexivité des liens

4 Les requêtes SPARQL

Nous avons testé ces différentes requêtes SPARQL via l'endpoint proposé par notre instance de GraphDB :

4.1 Requête 1

Cette requête récupère toutes les musiques, donne l'artiste et son vrai nom et prénom.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX erl: <http://www.websem.csv/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?song ?songName ?artist ?givenName ?familyName
WHERE {
  ?song a dbo:Song ;
        rdfs:label ?songName ;
        foaf:Person ?artist .
  ?artist foaf:givenName ?givenName ;
    foaf:familyName ?familyName .
}
```

Le résultat est disponible ici.

4.2 Requête 2

Cette requête répertorie tous les artistes ayant composé au moins un album du genre "Rap/Hip Hop".

Romain BARBIER,
Elsa BERNET, Lucas SUBE

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX erl: <http://www.websem.csv/resource/>
PREFIX erlo: <http://www.websem.csv/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?artist
WHERE {
  ?artist a foaf:Person ;
          erlo:produce ?album .
  ?album a dbo:Album ;
         dbo:genre ?genre .
  ?genre rdfs:label "Rap/Hip Hop" .
}
```

Le résultat est disponible ici.

4.3 Requête 3

Cette requête compte le nombre d'albums par genre, trié par ordre décroissant.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX erl: <http://www.websem.csv/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?genreLabel (COUNT(?album) AS ?albumCount)
WHERE {
  ?album a dbo:Album ;
         dbo:genre ?genre .
  ?genre rdfs:label ?genreLabel .
}
GROUP BY ?genreLabel
ORDER BY DESC(?albumCount)
```

Le résultat est disponible ici.

Romain BARBIER,
Elsa BERNET, Lucas SUBE