A la découverte d'un Triple Store

Nicolas Larrousse (TGIR Huma-Num - http://huma-num.fr)

Qu'est-ce qu'un Triple Store ?

Un stock de triplets au sens du modèle de données RDF

Comment on y accède (SPARQL End-Point ou autre ressource) et comment on l'interroge ?

Un langage d'interrogation, SPARQL
http://www.w3.org/TR/rdf-sparql-query/

Quelques essais en utilisant un « requêteur » générique en ligne
http://sparql.org/sparql.html

On interroge sur le contenu d'un fichier XML en ligne qui contient quelques triplets RDF (http://www.nakala.fr/jdev2015/tim_berners_lee.xml)

Tous les triplets

```
SELECT *
FROM <http://www.nakala.fr/jdev2015/tim_berners_lee.xml>
WHERE {
    ?sujet ?verbe ?objet .
}
```

Une sélection de triplets

```
SELECT *
FROM <http://www.nakala.fr/jdev2015/tim_berners_lee.xml>
WHERE {
    ?personne <http://xmlns.com/foaf/0.1/nick> ?nom.
}
```

On interroge sur le contenu d'une ressource RDF en ligne qui correspond aux données de Bordeaux dans la base GeoNames

Si l'on recherche « Bordeaux » dans la base GeoNames, on trouve que le fichier où se trouve les données concernant Bordeaux en RDF est celui ci :
http://sws.geonames.org/3031582/about.rdf

Par le même principe que précédemment, on peut donc interroger ce fichier en ligne.
Tous les triplets

```
SELECT *
FROM <http://sws.geonames.org/3031582/about.rdf>
WHERE {
    ?sujet ?verbe ?objet .
}
```

Une sélection de triplets

```
SELECT *
FROM <http://sws.geonames.org/3031582/about.rdf>
WHERE {
    ?sujet <http://www.w3.org/2003/01/geo/wgs84_pos#lat>
?objet .
}

SELECT *
FROM <http://sws.geonames.org/3031582/about.rdf>
WHERE {
    ?sujet
<http://www.geonames.org/ontology#wikipediaArticle> ?objet
.
}

SELECT *
FROM <http://sws.geonames.org/3031582/about.rdf>
WHERE {
    ?sujet <http://www.w3.org/2000/01/rdf-schema#seeAlso>
?objet .
}
```

A noter que GeoNames met en œuvre la négociation de contenu et que l'on aurait pu utiliser l'URI générique pour Bordeaux (Les humains voient une carte et les machines … du RDF).
http://sws.geonames.org/3031582

```
SELECT *
FROM <http://sws.geonames.org/3031582>
WHERE {
    ?sujet <http://www.w3.org/2000/01/rdf-schema#seeAlso>
?objet .
}
```

Interrogation du Triple Store de NAKALA (http://nakala.fr/sparql)

NAKALA a un modèle très simple, donc c'est un point de départ assez lisible.

En général, la requête par défaut consiste à afficher les « concepts »

```
SELECT DISTINCT ?Concept
WHERE {
?X rdf:type ?Concept .
}
LIMIT 100
```

On trouve 4 concepts, on peut afficher les « choses » d'un type donné, par exemple un « foaf :agent »

```
SELECT DISTINCT ?choses
WHERE {
?choses rdf:type <http://xmlns.com/foaf/0.1/Agent> .
}
LIMIT 100
```

Avec prudence (mettre impérativement un « LIMIT »), on peut demander d'afficher les triplets contenus dans le Triple Store sans effectuer de sélection.

```
SELECT *
WHERE {
?sujet ?verbe ?objet .
}
LIMIT 300
```

On note qu'il y a plusieurs types de ressources (5 de 4 types différents)
- http://www.nakala.fr/collection
- http://www.nakala.fr/scheme
- http://www.nakala.fr/resource
- http://www.nakala.fr/data
- http://www.nakala.fr/account

On peut regarder les choses liées à une collection donnée, identifiée ici par son URI http://www.nakala.fr/collection/11280/5e3dd209 et par quels verbes

```
SELECT *
WHERE {
<http://www.nakala.fr/collection/11280/5e3dd209> ?verbe
?chose .
}
LIMIT 100
```

De même, on peut regarder les choses liées à une compte donné, identifié ici par son URI http://www.nakala.fr/account/11280/19606173 et par quels verbes

```
SELECT *
WHERE {
<http://www.nakala.fr/account/11280/19606173> ?verbe
?chose .
}
```

A partir du résultat on voit que

- les comptes sont de type foaf :Agent

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
     http://xmlns.com/foaf/0.1/Agent
```

- la chaîne de caractères de leur nom est associé à leur identifiant via un rdfs :label

```
http://www.w3.org/2000/01/rdf-schema#label   "CFEETK "
```

On peut donc construire une requête pour afficher tous les noms des comptes

```
SELECT ?compte, ?nom
WHERE {
  ?compte <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type>  <http://xmlns.com/foaf/0.1/Agent> .
  ?compte <http://www.w3.org/2000/01/rdf-schema#label>
?nom .
}
```

Ce qui s'écrit de manière plus lisible

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?compte, ?nom
WHERE {
  ?compte rdf:type  foaf:Agent .
  ?compte rdfs:label ?nom .
}
```

A noter que dans le cas de NAKALA, on n'aurait pas besoin d'utiliser les préfixes, car ils sont connus du Triple-Store. Ils sont implicites compte tenu du contenu du graphe. Le préfixe RDF est supposé connu systématiquement.

PREFIX rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX ore: <http://www.openarchives.org/ore/terms/>

La requête suivante aurait donc donné le même résultat.

```
SELECT ?compte, ?nom
WHERE {
  ?compte rdf:type  foaf:Agent .
  ?compte rdfs:label ?nom .
}
```

Si l'on s'intéresse à ce que NAKALA nomme une ressource, par exemple http://www.nakala.fr/resource/11280/00114418

On peut regarder tous les éléments associés à cette ressource

```
SELECT *
WHERE {
  <http://www.nakala.fr/resource/11280/00114418> ?verbe
?objet .
}
```

On s'aperçoit que :
- la ressource est liée à un compte par la relation "dcterms:publisher"
- la ressource est liée à une donnée par la relation "foaf:primaryTopic"

On peut regarder la structure des données associées à cette ressource, dans notre cas http://www.nakala.fr/data/11280/00114418

```
SELECT *
WHERE {
  <http://www.nakala.fr/data/11280/00114418> ?verbe ?objet
.
}
```

La donnée possède un titre exprimé par la propriété « dcterms:title »

La combinaison de ces informations permet de trouver tous les titres des données associées à un compte en parcourant le graphe : ici la « ressource » ou fiche de donnée est le pivot..

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX ore: <http://www.openarchives.org/ore/terms/>

SELECT ?donnee ?titre
WHERE {
  ?fiche dcterms:publisher
<http://www.nakala.fr/account/11280/19606173> .
  ?fiche foaf:primaryTopic ?donnee .
  ?donnee dcterms:title ?titre .
}
```

A noter que NAKALA supporte la négociation de contenu. La requête suivante peut être exécutée depuis le requêteur générique (http://sparql.org/sparql.html).

```
SELECT *
FROM <http://www.nakala.fr/account/11280/19606173>
WHERE {
    ?sujet ?verbe ?objet .
}
```

Vue alternative du contenu d'un Triple Store avec Pubby
(http://wifo5-03.informatik.uni-mannheim.de/pubby/)

Les URIs de NAKALA peuvent être parcourues dans un simple navigateur grâce à l'interface vers le SPARQL EndPoint « Pubby ».

Par exemple avec l'URI
« http://www.nakala.fr/account/11280/19606173 »

On retrouve les éléments du modèle et leurs propriétés. C'est un bon complément à l'approche précédente.

Vers des données liées : un exemple de « Follow Your Nose »

L'idée est de parcourir différents entrepôt pour enrichir ses propres données.
La requête sur les informations d'un compte

```
SELECT *
WHERE {
<http://www.nakala.fr/account/11280/b5927b13> ?verbe ?chose
.
}
```

Donnait une information de type « spatial »
http://purl.org/dc/terms/spatial " http://sws.geonames.org/2968705/"

On peut la récupérer directement par la requête

```
SELECT ?geo
WHERE {
<http://www.nakala.fr/account/11280/b5927b13>
<http://purl.org/dc/terms/spatial> ?geo .
}
```

A partir du résultat, on peut interroger la base « Geonames » pour récupérer un lien vers DbPedia

```
SELECT ?lien
FROM <http://sws.geonames.org/2968705>
```

```
WHERE {
    ?sujet <http://www.w3.org/2000/01/rdf-schema#seeAlso>
?lien .}
```

On obtient le lien
http://dbpedia.org/resource/Villejuif

On peut alors aller rechercher les informations sur cette URI dans le SPARQL End Point de DbPedia (ou en utilisant pubby qui est installé également sur DbPedia)

```
SELECT * WHERE
{ <http://dbpedia.org/resource/Villejuif> ?verbe ?objet.
}
LIMIT 100
```

On repère qu'une présentation est associée à la propriété :
`<http://dbpedia.org/ontology/abstract>`
On peut donc récupérer cette présentation en français.

```
SELECT ?resume WHERE
{ <http://dbpedia.org/resource/Villejuif>
<http://dbpedia.org/ontology/abstract> ?resume.
FILTER langMatches( lang(?resume), 'fr')
.
}
```