

Tutoriel complet : SPARQL et son utilisation avec Python

1. Pourquoi utiliser SPARQL ?

- Introduction à SPARQL
- Cas d'usage de SPARQL

2. Les données RDF et RDFS

- RDF (Resource Description Framework)
 - Triplet RDF : Sujet - Prédicat - Objet
 - Exemple RDF
- Les principales notions de RDFS :
 - Classe (rdfs:Class)
 - Sous-classe (rdfs:subClassOf)
 - Propriété (rdf:Property)
 - Sous-propriété (rdfs:subPropertyOf)
 - Domaine (rdfs:domain)
 - Portée (rdfs:range)
 - Nœud anonyme (Blank Node)

3. Les types de requêtes SPARQL

- **SELECT** : Récupération de résultats sous forme de tableau
- **ASK** : Vérification de l'existence d'un motif
- **CONSTRUCT** : Génération d'un sous-graphe RDF
- **DESCRIBE** : Extraction des informations détaillées d'une ressource

4. Les principales notions SPARQL

- **PREFIX** : Définition des préfixes
- **FILTER** : Filtrage des résultats
- **OPTIONAL** : Données facultatives
- **UNION** : Alternance de motifs
- **GROUP BY / ORDER BY** : Agrégation et tri des résultats
- **LIMIT / OFFSET** : Pagination
- **BIND / VALUES** : Affectation de valeurs
- **MINUS / NOT EXISTS** : Exclusion de motifs
- **SERVICE** : Interrogation d'autres endpoints SPARQL distants

5. Exécuter des requêtes SPARQL avec Python

- Utilisation de **SPARQLWrapper** pour accéder à des endpoints publics (ex : DBpedia, Wikidata)
- Utilisation de **RDFLib** pour manipuler des données RDF locales

1. Pourquoi utiliser SPARQL ?

SPARQL (SPARQL Protocol and RDF Query Language) est un langage standardisé par le W3C pour interroger des données RDF (Resource Description Framework).

Il est utilisé pour :

- Interroger des graphes de connaissances.
- Manipuler et extraire des données liées.
- Intégrer des données provenant de plusieurs sources.
- Accéder à des bases de données ouvertes (DBpedia, Wikidata...).

Cas d'usage :

- Trouver des informations précises dans un graphe sémantique.
- Extraire des relations complexes entre entités.
- Réaliser des analyses sur des graphes RDF.

2. Les données RDF et RDFS

RDF (Resource Description Framework)

Le RDF décrit des données sous forme de triplets :

Sujet - Prédicat - Objet

Exemple RDF :

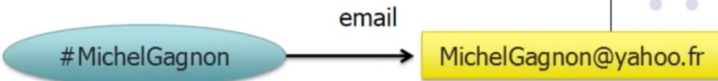
```
turtle
CopierModifier
@prefix ex: <http://example.org/> .

ex:Personne1 ex:nom "Alice" .
ex:Personne1 ex:age "30"^^xsd:integer .
```

Syntaxe RDF XML

1. Description avec une seule propriété

Exemple



```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:local=http://www.MonDomaine.dz/Vocabulary#
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#

  <rdf:Description
    rdf:about="http://www.polymtl.ca/Profs#MichelGagnon">
    <local:email>MichelGagnon@yahoo.fr</local:email>
  </rdf:Description>

</rdf:RDF>
```

Notions de RDFS :

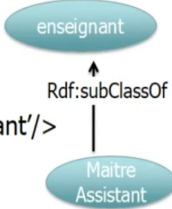
- **Classe (rdfs:Class)** : définit un type d'entité.
Exemple : `ex:Personne a rdfs:Class` .
- **Sous-classe (rdfs:subClassOf)** : relation hiérarchique entre classes.
Exemple : `ex:Employe rdfs:subClassOf ex:Personne` .

RDFS sous-classe

- Syntaxe :

```
<rdfs:class rdf:about=[classefille]>
  <rdfs:subclassof rdf:resource [classe mère]/>
</rdfs:class>
```
- Exemple:

```
<rdfs:class rdf:about='#maitreAssistant'>
  <rdfs:subclassof rdf:resource '#enseignant'/>
</rdfs:class>
```



- **Propriété (rdf:Property)** : relation entre deux ressources.
Exemple : `ex:nom a rdf:Property` .

RDFS propriété

- exemple:

```
<rdfs:property rdf:about='#responsable'>
  <Rdfs:domain rdf:resource='#enseignant'/>
  <Rdfs:range rdf:resource='#département'/>
</rdfs:property>

<rdfs:property rdf:about='#créateur'>
  <Rdf:domain rdf:resource='#personne'/>
  <Rdf:range rdf:resource='#site'/>
</rdfs:property>
```

- **Littéraux (rdf:Literal)** : Propriété à valeur constante (String, Integer ...)

RDFS les littéraux



- `<rdfs:Literal>` : permet de déclarer une propriété à valeur constante ex:String; integer ...

Exemple :

```
<rdfs:property rdf:about='#email'>
  <Rdfs :domain rdf:resource='#personne'/>
  <Rdfs :range rdf:resource='rdfs:Literal'/>
</rdfs:property>
```

- **Sous-propriété (rdfs:subPropertyOf)** : spécialisation de propriétés.
Exemple : ex:prenom rdfs:subPropertyOf ex:nom .

RDFS souspropriété



- `<rdfs:SubProperty>` : Permet de déclarer qu'une propriété est une sous propriété d'une autre

- Syntaxe :

```
<rdfs:property rdf:about=[propriété]>
  <Rdfs :SubProperty rdf:resource=[proprité mère]/>
</rdfs:property>
```

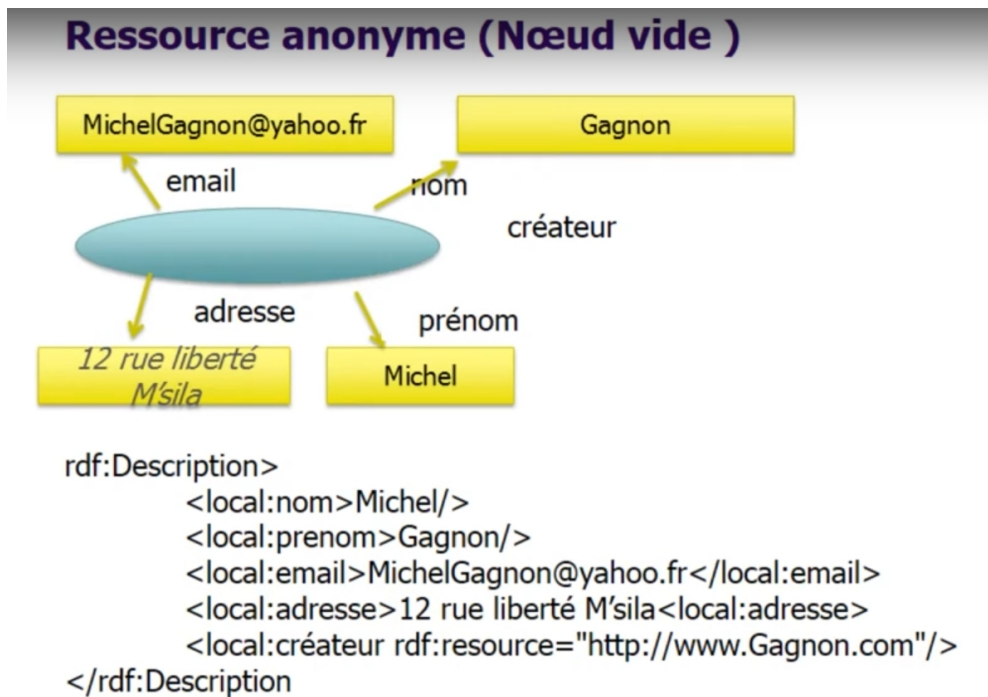
- Exemple :

```
<rdfs:property rdf:about='#parentDe'>
  <Rdfs :SubProperty rdf:resource='#père'/>
</rdfs:property>
```

- **Domaine (rdfs:domain)** : classe d'application d'une propriété.
Exemple : ex:nom rdfs:domain ex:Personne .
- **Portée (rdfs:range)** : type des valeurs possibles.
Exemple : ex:nom rdfs:range xsd:string .

Nœud anonyme :

Utilisé lorsqu'une ressource n'a pas d'URI propre.



3. Les types de requêtes SPARQL

SPARQL permet de faire différents types de requêtes pour interroger les données RDF. Voici les principaux types :

- **SELECT : Extraction de données tabulaires**

Cette requête permet de récupérer des résultats sous forme de tableau.

Exemple :

```

sparql
CopierModifier
PREFIX ex: <http://example.org/>

```

```

SELECT ?personne ?nom WHERE {
  ?personne a ex:Personne .
  ?personne ex:nom ?nom .
}

```

Cette requête sélectionne toutes les personnes avec leur nom.

- ASK : Vérifie l'existence d'un motif

Une requête ASK retourne **true** ou **false** selon si le motif existe ou non dans les données.

Exemple :

```
sparql
CopierModifier
ASK {
  ex:Personne1 ex:nom "Alice" .
}
```

Cette requête vérifie si `ex:Personne1` a le nom "Alice".

- CONSTRUCT : Génère un sous-graphe RDF

La requête CONSTRUCT permet de créer un sous-graphe RDF basé sur un motif.

Exemple :

```
sparql
CopierModifier
CONSTRUCT {
  ?personne ex:aPourNom ?nom .
} WHERE {
  ?personne ex:nom ?nom .
}
```

Cette requête génère un sous-graphe où chaque personne est associée à son nom.

- DESCRIBE : Récupère des informations détaillées sur une ressource

Cette requête fournit un ensemble de triplets concernant la ressource spécifiée.

Exemple :

```
sparql
CopierModifier
DESCRIBE ex:Personne1
```

Elle retourne toutes les informations disponibles sur `ex:Personne1`.

4. Les principales notions SPARQL

- PREFIX : Simplification des URIs

Les préfixes permettent de réduire les URIs en les associant à des alias. Cela rend les requêtes plus lisibles et plus courtes.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne WHERE { ?personne ex:nom "Alice" . }
```

- FILTER : Filtrage conditionnel

FILTER permet de restreindre les résultats en fonction de conditions.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne ?age WHERE {
  ?personne ex:age ?age .
  FILTER (?age > 25)
}
```

Cette requête sélectionne les personnes dont l'âge est supérieur à 25.

- OPTIONAL : Données facultatives

OPTIONAL permet d'ajouter des données qui ne sont pas forcément présentes.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne ?nom ?age WHERE {
  ?personne ex:nom ?nom .
  OPTIONAL { ?personne ex:age ?age . }
}
```

Si une personne a un âge, il sera retourné, sinon, l'âge sera absent.

- UNION : Combinaison de motifs alternatifs

UNION permet de combiner plusieurs motifs qui peuvent être vrais indépendamment.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne WHERE {
  { ?personne ex:nom "Alice" . }
  UNION
  { ?personne ex:nom "Bob" . }
}
```

Cette requête retourne les personnes nommées soit "Alice", soit "Bob".

- GROUP BY / ORDER BY : Agrégation et tri

GROUP BY permet de regrouper les résultats, tandis que ORDER BY trie les résultats.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne (COUNT(?propriete) AS ?nb_proprietes) WHERE {
    ?personne ex:aPourPropriete ?propriete .
}
GROUP BY ?personne
ORDER BY DESC(?nb_proprietes)
```

Cela retourne le nombre de propriétés pour chaque personne, trié par nombre décroissant.

- LIMIT / OFFSET : Pagination des résultats

LIMIT permet de restreindre le nombre de résultats retournés, tandis que OFFSET définit à partir de quel résultat commencer.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne WHERE {
    ?personne ex:nom ?nom .
}
LIMIT 10 OFFSET 20
```

Cela retourne les résultats de la 21ème à la 30ème personne.

- BIND / VALUES : Affectation de valeurs

BIND permet de créer des variables basées sur des expressions. VALUES permet de définir une liste de valeurs.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
VALUES ?nom { "Alice" "Bob" }
SELECT ?personne WHERE {
    ?personne ex:nom ?nom .
}
```

Cette requête retourne les personnes ayant pour nom "Alice" ou "Bob".

- MINUS / NOT EXISTS : Exclusion de motifs

MINUS permet d'exclure un motif particulier des résultats.

Exemple :

```
sparql
CopierModifier
PREFIX ex: <http://example.org/>
SELECT ?personne WHERE {
    ?personne ex:nom ?nom .
}
MINUS { ?personne ex:age ?age . }
```

Cette requête exclut les personnes ayant un âge.

- SERVICE : Requêtes sur des endpoints distants

La clause SERVICE permet d'interroger des endpoints SPARQL distants.

Exemple :

```
sparql
CopierModifier
SERVICE <https://dbpedia.org/sparql> {
    SELECT ?personne ?nom WHERE {
        ?personne dbo:birthPlace <http://dbpedia.org/resource/Paris> .
        ?personne foaf:name ?nom .
    }
}
```

Cela permet d'interroger DBpedia pour récupérer des informations sur des personnes nées à Paris.

5. Exécuter des requêtes SPARQL avec Python

- Utilisation de SPARQLWrapper : Accéder à des endpoints publics

SPARQLWrapper est une bibliothèque Python permettant d'interroger des endpoints SPARQL distants.

Exemple :

```
python
CopierModifier
from SPARQLWrapper import SPARQLWrapper, JSON

sparql = SPARQLWrapper("https://dbpedia.org/sparql")
sparql.setQuery("""
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?name WHERE {
    ?person dbo:birthPlace <http://dbpedia.org/resource/Paris> .
    ?person foaf:name ?name .
} LIMIT 10
""")
sparql.setReturnFormat(JSON)
results = sparql.query().convert()

for result in results["results"]["bindings"]:
    print(result["name"]["value"])
```

Ce code récupère les noms de personnes nées à Paris à partir de DBpedia.

- Utilisation de RDFLib : Travailler avec des données locales

RDFLib est une bibliothèque Python permettant de manipuler des données RDF locales.

Exemple :

```
python
CopierModifier
from rdflib import Graph

g = Graph()
g.parse("donnees.ttl", format="turtle")

results = g.query("""
PREFIX ex: <http://example.org/>
SELECT ?nom WHERE {
    ?personne ex:nom ?nom .
}
""")
for row in results:
    print(row.nom)
```

Ce code interroge un fichier Turtle local pour récupérer les noms des personnes.