SPARQL
Simple Protocol And RDF Query Language

Cours 1ere Master SIC (2019-2020)

Introduction

- Un langage et protocole de requêtes pour l'interrogation de méta données RDF.
- Recommandation W3C depuis le 15 janvier 2008.
- Un langage de requête: http://www.w3.org/TR/rdfsparqlquery/
- Un protocole: http://www.w3.org/TR/rdfsparqlprotocol/
- Version : SPARQL 1.1.

Syntaxe: requête SPARQL

- **Forme générale :**

```
PREFIX NamSpace
 SELECT ?Variable
 FROM RDF-Dataset
 WHERE {modèle de requête (motif)}
```

- **Exemple**:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?person foaf:name ?name . }
```

Syntaxe: forme générale

- Modèle de requête: Critères sous forme de triplet: {Sujet Prédicat Objet }=> motif de graphe
- Variable: ?x , ?name ou $x, $name.
  - Une variable peut remplacer: un sujet un prédicat ou un objet, un littéral ou une ressource.
- QName: rdf:type foaf:name
- Littéral: “cours SPARQL” , 5.6, true, “12”^^xsd:integer
- nœud anonyme: _:b1, _:b2 ou []

Exemple: Requête simple

- Données:

```
<http://example.org/book/book1>
  <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

- Requête :

```
SELECT ?title
WHERE {
<http://example.org/book/book1>
  <http://purl.org/dc/elements/1.1/title> ?title . }
```

- Résultat:

Title
"SPARQL Tutorial"

Exemple: Requête simple

- **Données (Turtle):** `@prefix foaf: <http://xmlns.com/foaf/0.1/> .`

```
_:a  foaf:name  "Johnny Lee Outlaw" .
_:a  foaf:mbox  <mailto:jlow@example.com> .
_:b  foaf:name  "Peter Goodguy" .
_:b  foaf:mbox  <mailto:peter@example.org> .
_:c  foaf:mbox  <mailto:carol@example.org> .
```

- **Requête :**

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name . ?x foaf:mbox ?mbox }
```

- **Résultat:**

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Syntaxe équivalente

- { ?x **foaf:name** ?name .
  ?x **foaf:mbox** ?mbox } ⇔

  { ?x **foaf:name** ?name ;
  foaf:mbox ?mbox }
- { ?x **foaf:mbox** " Mohamed@yahoo.fr " .
  ?x **foaf:mbox** " Mohamed@gmail.com" } ⇔

  { ?x **foaf:mbox** " Mohamed@yahoo.fr " ,
  "Mohamed@gmail.com " }

Les Filtres

- Le mot clé FILTER.
- Impose des contraintes sur les variables .
- Filtres à base d'expressions régulières: regex().
- **Exemple:**

```
SELECT ?resource
  WHERE{?resource  ex:age  ?age
   FILTER(?age>=24)}
```

Les Filtres : contraintes sur les variables

- **des contraintes sur les variables**
  - **Opérateurs de Comparaison** : $<$, $>$, $=$, $=<$,$=>$, !=.
  - **Opérations** : +, * ,/, -.
  - **Booléens** : && (and) ,|| (or) ,! (not).
  - **Fonctions** :
    - **De test : retournent des valeurs booliennes** : isBlank(?x), isURI(?x), isLiteral(?x), bound(?x), ... :
    - **De transformation :** datatype(?y), str(?x), xsd:integer(?x), xsd:dateTime(?x),...

Filtres : contraintes sur Les variables (exemples)

- Données : `@prefix foaf: <http://xmlns.com/foaf/0.1/> .`

```
_:a  foaf:name  "Alice".
_:a   foaf:mbox  <mailto:alice@work.example> .
_:b  foaf:name  "Ms A.".
_:b  foaf:mbox  <mailto:alice@work.example> .
```

- Requête:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name1 ?name2
WHERE { ?x foaf:name ?name1 ;  foaf:mbox  ?mbox1 .
        ?y foaf:name ?name2 ; foaf:mbox ?mbox2 .
      FILTER ((?mbox1 = ?mbox2 )&& (?name1 != ?name2)) }
```

- Résultat:

Name 1	Name2
"Alice"	"Ms A."
"Ms A."	"Alice"

- Données :

```
_:a  foaf:name "Alice".
_:a ex:shoeSize "9.5"^^xsd:float .
_:b foaf:name "Bob".
 _:b ex:shoeSize "42"^^xsd:integer .
```

- Requête :

```
SELECT ?name ?shoeSize
WHERE { ?x foaf:name ?name ; ex:shoeSize ?shoeSize .
FILTER ( datatype(?shoeSize) = xsd:integer ) }
```

- Résultat:

Name	shoeSize
"Bob"	42

- **Données** :

```
_:a foaf:name "Alice".
_:a foaf:mbox <mailto:alice@work.example> .
 _:b foaf:name "Bob" .
 _:b foaf:mbox "bob@work.example" .
```

- **Requête :**

```
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name ; foaf:mbox ?mbox .
FILTER isURI(?mbox) }
```

- **Résultat:**

Name	mbox
« Alice"	<mailto:alice@work.example>

Filtres à base d'expressions régulières

- Opérations pour tester les chaines de caractères à base d'expressions régulières.
- Similaire à « LIKE » du SQL.
- FILTER **regex(?x,"pattern", ["flags"]).**
  - **?x** : variable ;
  - **« patterne »:** modèle d'expression régulière.
  - **« flags »**: optionnel , fournit des contraintes supplémentaires

Filtres à base d'expressions régulières

- **Exemple**:

```
_:a foaf:name "Alice".
 _:b foaf:name "Bob" .
```

- **Requête** :

```
SELECT ?name
WHERE { ?x foaf:name ?name
        FILTER regex(?name, "^ali", "i") }
```

(?name commance par “ali” (^), sans respecter la casse (flag “i”))

- **Résultat:**

Name
"Alice"

Filtres à base d'expressions régulières

- **Exemple d'expressions régulières :**
  - « ^ch »: les chaines qui commencent par « ch ».
  - « ch »: regex(?x, "ch"): les chaines qui contiennent « ch ».
  - « ch$ »: les chaines qui se terminent par « ch ».
- **Exemple de flags:**
  - « i »: faire le matching sans considération de la casse.
  - « x »: supprimer les espaces avant de faire le matching.

Filtres à base d'expressions régulières

- Opérations pour tester les chaines de caractères à base d'expressions régulières.
- Similaire à « LIKE » du SQL.
- FILTER **regex(?x,"pattern", ["flags"]).**
  - **?x** : variable ;
  - **« patterne »:** modèle d'expression régulière.
  - **« flags »**: optionnel , fournit des contraintes supplémentaires

FILTER : EXISTS / NOT EXISTS

```
 _:a  rdf:type  foaf:Person .
_:a  foaf:name  "Alice" .
_:b  rdf:type  foaf:Person .
```

Requête :

```
SELECT ?person
WHERE { ?person rdf:type foaf:Person .
 FILTER NOT EXISTS {?person foaf:name ?name}
    }
```

- Résultat:

person
_ :b

FILTER : MINUS

```
 _:a  rdf:type  foaf:Person .
_:a  foaf:name  "Alice" .
_:b  rdf:type  foaf:Person .
_:b  foaf:name  "Bob" .
_:c  rdf:type  foaf:Person .
_:c  foaf:name  "Celin" .
```

Requête :

```
SELECT DISTINCT ?s
WHERE { ?s ?p ?o .
MINUS { ?s foaf:name "Celine" . }}
```

- Résultat:

S
_ :a
_ :b

Le pattern OPTIONAL

- Permet de dire qu'un *pattern peut-être optionnel* .

```
_:a  rdf:type  foaf:Person .
 _:a  foaf:name  "Alice" .
 _:a  foaf:mbox  <mailto:alice@example.com> .
_:a  foaf:mbox  <mailto:alice@work.example> .
_:b  rdf:type  foaf:Person .
_:b  foaf:name  "Bob" .
```

- **Requête :**

```
SELECT ?name ?mbox
    WHERE { ?x  foaf:name  ?name .
    OPTIONAL { ?x foaf:mbox  ?mbox } }
```

- Résultat:

Name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example> .
"Bob"	

Le pattern OPTIONAL

- **Exemple 2:**

```
ex:book1 dc:title "SPARQL Tutorial" .
ex:book1 ns:price  42 .
ex:book2 dc:title "The Semantic Web" .
ex:book2 ns:price 23 .
```

- **Requête :**

```
SELECT ?title ?price
  WHERE { ?x dc:title ?title .
  OPTIONAL { ?x  ns:price  ?price .
  FILTER (?price < 30) } }
```

- **Résultat :**

Title	price
"SPARQL Tutorial"	
"The Semantic Web"	23

Le pattern OPTIONAL

- **Exemple 3:**

```
_:a   foaf:givenName   "Alice".
_:b   foaf:givenName   "Bob" .
_:b   dc:date "2005-04-04T04:04:04Z"^^xsd:dateTime .
```

- **Requête** : filtre les personnes avec un nom (givenName) mais *sans* date exprimée.

```
SELECT ?name
WHERE { ?x  foaf:givenName ?name .
OPTIONAL { ?x dc:date ?date } .
FILTER (!bound(?date)) }
```

- **Résultat :**

name
"Alice".

Les alternatives

- Un moyen de combiner les motifs de graphe.
- Si un des motifs de graphe alternatifs correspondre : un ou logique .
- Si plusieurs alternatives correspondent, toutes les solutions de motifs possibles sont trouvées.
- Les alternatives de motifs sont exprimées par le mot-clé **UNION**.

Les alternatives: Exemple

```
_:a  foaf:name "Alice" .
_:a  foaf:homepage <http://work.example.org/alice/> .
_:b  foaf:name "Bob" .
_:b  foaf:mbox <mailto:bob@work.example> .
_:c  foaf:name "Charly" .
_:c  foaf:mbox <mailto:charly@work.example> .
_:c  foaf:homepage <http://work.example.org/charly/> .
```

- Requête :

```
SELECT ?name  ?mbox  ?hpage
WHERE { {?x  foaf:name  ?name ; foaf:mbox  ?mbox}
        UNION
        { ?x  foaf:name  ?name ;  foaf:homepage  ?hpage} }
```

- Résultat

Name	mbox	hpage
”Bob”	<mailto:bob@work.example>	
"Charly"	<mailto:charly@work.example>	
"Alice"		<http://work.example.org/alice/>
"Charly"		<http://work.example.org/charly/>

Modificateurs des séquences de solutions

- les motifs génèrent une collection non-ordonnée de solutions, chaque solution étant une valuation des variables présentes dans les motifs.
- Ces solutions sont ensuite traitées comme une séquence, sur laquelle on peut appliquer un opérateur (***modificateur de séquence).***
- **Exemple de modificateur:**
  - **ORDER BY**: pour ordonner les solution.
  - **DISTINCT:** éliminer les doublons parmi les solutions.
  - **Offset:** indiquer à partir de quelle position dans la séquence on démarre.
  - **Limit** : borner la taille de la séquence des solutions.

Modificateurs: ORDER BY

```
ex:p1      dc:author   ex:doc2
ex:p2      dc:author   ex:doc1
ex:p2      dc:author   ex:doc3
ex:doc2    ex:date     20011-01-01
ex:doc1    ex:date     2009-12-31
ex:doc3    ex:date     2009-12-31
```

- Requête : `SELECT ?doc ?date`
  `WHERE { ?pers dc:author ?doc`
  `?doc ex:date ?date }`
  `ORDER BY ?date desc(?doc)  //`
  `(ASC() :ascendant par défaut )`
- Résultat:

Doc	date
Ex:doc3	2009-12-31
Ex: doc1	2009-12-31
Ex:doc2	20011-01-01

Modificateurs: Distinct

```
_:x  foaf:name  "Alice" .
_:x  foaf:mbox  <mailto:alice@example.com> .
_:y  foaf:name  "Alice" .
_:y  foaf:mbox  <mailto:asmith@example.com> .
_:z  foaf:name  "Alice" .
 _:z foaf:mbox  <mailto:alice.smith@example.com> .
```

- Requête 1: `SELECT ?name WHERE { ?x foaf:name ?name }`
- Résultat 1:

name
"Alice"
"Alice"
"Alice"

- Requête 2: `SELECT DISTINCT ?name WHERE { ?x foaf:name ?name }`
- Résultat 2:

name
"Alice"

Modificateurs: Limit/Offset

- **OFFSET n**: signifie qu'on "passe" les *n premières solutions. Un offset* de 0 n'a pas d'effet.
- **LIMIT n** :signifie qu'on donne au maximum *n solutions.*
- **Exemple:**

```
SELECT ?name
WHERE { ?x foaf:name ?name }
ORDER BY ?name
LIMIT 5
OFFSET 10
```

- Cette requête donne au maximum 5 solutions, à partir de la 11eme dans la séquence des solutions.

Autres formes de requêtes

- Il existe d'autres formes de requêtes :
  - **ASK** : retourne un booléen indiquant l'existence d'une solution qui satisfait le motif.
  - **CONSTRUCT** : pour construire un graphe RDF de solution, à partir des valuations des variables.
  - **DESCRIBE** : retourne un graphe RDF décrivant les ressources trouvées. Le résultat dépend du moteur de requête.

Autres formes de requêtes : ASK

```
_:x  foaf:name  "Alice" .
_:x  foaf:mbox  <mailto:alice@example.com> .
```

- **La Requête** :

```
ASK { ?x foaf:name ?name ; foaf:mbox ?mbox }
```

Retourne true.

- **La Requête** :

```
ASK { ?x foaf:name "Bob"}
```

Retourne false.

Autres formes de requêtes : CONSTRUCT

```
ex:Mohamed  ex:sister  ex:Khadidja
```

- **Requête :**

```
CONSTRUCT {  ?girl ex:brother ?boy }
   WHERE { ?boy ex:sister ?girl  }
```

- Retourne le graphe suivant:

```
ex:Khadidja ex:brother ex:Mohamed
```

Autres formes de requêtes : DESCRIBE

- **Exemple :**

```
DESCRIBE ?x
WHERE {  ?x ns:name ?name. ?x ns:age "42"  }
```

- Retourne le graphe suivant:

```
:William rdf:type :Person;
:William ns:hasSpouse :Laura;
:William ns:shoesize "10";
:William ns:age "42";
:William ns:name "William"
```

Graphes par défaut et graphes Nommés

- Interroger une base de plusieurs graphes
- Un graphe par défaut (*background*) *et des graphes nommés avec* des URI.

```
# Default graph (stored at dft.ttl)
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<bob.ttl>    dc:publisher  "Bob Hacker" .
<alice.ttl>  dc:publisher  "Alice Hacker" .

# Named graph: bob.ttl
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .

# Named graph: alice.ttl
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example.org> .
```

Graphes par défaut et graphes Nommés

- Requête

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?who ?g ?mbox
FROM <dft.ttl>
FROM NAMED <alice.ttl>
FROM NAMED <bob.ttl>
WHERE
{
   ?g dc:publisher ?who .
   GRAPH ?g { ?x foaf:mbox ?mbox }
}
```

- Résultat

```
-------------------------------------------------------------------------
| who             | g                                          | mbox     |
=========================================================================
| "Alice Hacker" | <file:///xxxx/alice.ttl> | <mailto:alice@work.example.org>  |
| "Bob Hacker"   | <file:///xxxx/bob.ttl>   | <mailto:bob@oldcorp.example.org> |
-------------------------------------------------------------------------
```

SPARQL Update

- **INSERT DATA et DELETE DATA**

```
INSERT DATA { ex:book1 dc:title "un livre";
                       dc:creator "un auteur".}

DELETE DATA { GRAPH <http://example/bookStore>
              {  ex :book2 dc:title "SPARQL";
                           dc:creator "W3c" .}}
```

SPARQL Update

```
INSERT {
?x ex:uncle ?uncle
   }
WHERE {
{ ?x ex:mother [ ex:brother ?uncle ] }
UNION
{ ?x ex:father [ ex:brother ?uncle ] }
}
```

- Autres:

```
CLEAR GRAPH <uri>
CREATAE GRAPH <uri>
DROP GRAPH <uri>
```

SPARQL Endpoints

- Point de présence sur un réseau HTTP capable de recevoir et de traiter des requêtes de protocole SPARQL.
- **Exemples :**
  - **http://sparql.org/sparql.html**
  - **http://dbpedia.org/sparql**

    http://bio2rdf.org/sparql