Consultation de graphes RDF : SPARQL

I.Mougenot

LIRMM

Mastère informatique 2014

Tirer parti d'un modèle RDF

Comment tirer parti au mieux d'un graphe RDF ?

- Exploiter des patrons sur les triplets (exemples avec l'API RDF de Jena)
- Recours à différents langages de requêtage :
  - RQL (RDF Query Language) - syntaxe proche de OQL
  - RDQL (W3C) - SeRQL syntaxe proche de SQL
  - Utiliser XML: XSLT, XPath, XQuery
  - SPARQL - Recommandation du W3C depuis 2008

Exemple Selector dans Jena

Poser des filtres sur les déclarations du modèle

```
StmtIterator stmtI = m.listStatements(
    new SimpleSelector(null,p, (RDFNode) null));
while (stmtI.hasNext())
{Statement sti = stmtI.nextStatement();
Resource rssi = (Resource) sti.getSubject();
Resource rsoi = (Resource) sti.getObject();
System.out.println (rssi.getLocalName()+" "
+rsoi.getLocalName());}
```

Listing 1: Notion de filtre avec selector

Exemple Naviguer dans les classes

Poser des filtres sur les déclarations du modèle

```
ExtendedIterator<OntClass> classes = m.listClasses();
while (classes.hasNext())
{OntClass thisClass = (OntClass) classes.next();
if (thisClass.isURIResource())
System.out.println("UriRef "+ thisClass.getLocalName());
else if (thisClass.isAnon())
System.out.println("classe anonyme "+ thisClass.getId());
ExtendedIterator instances = thisClass.listInstances();
while (instances.hasNext())
{Individual thisInstance = (Individual) instances.next();
System.out.println(" i "+ thisInstance.getLocalName()); } }
```

Listing 2: Naviguer dans les triplets

SPARQL, Simple Protocol and RDF Query Language

Standard W3C pour faciliter l'interrogation de sources de données distribuées, dernière évolution SPARQL 1.1

- langage de requête pour RDF (calcul d'une requête par appariement de graphes)
- protocole : spécification pour émettre et envoyer des requêtes SPARQL (services Web) vers des serveurs dédiés et en recevoir les résultats
- formats divers dont XML (requêtes de type SELECT et ASK) ou RDF (requêtes de type CONSTRUCT et DESCRIBE) pour l'affichage des résultats obtenus

Illustration

A la fois requêtage et communication

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?website
FROM <http://planetrdf.com/bloggers.rdf>
WHERE { ?person foaf:weblog ?website ;
            foaf:name ?name . }
http://.../qps?
query-lang=http://www.w3.org/TR/rdf-sparql-query/
&graph-id=http://planetrdf.com/bloggers.rdf
&query=PREFIX foaf: <http://xmlns.com/foaf/0.1/...

Extrait de :
http://www.dajobe.org/talks/200603-sparql-stanford/
```

Listing 3: Requête et protocole

SPARQL en tant que langage d'interrogation

Quatre formes de requêtage (LMD)

1. **SELECT** : rechercher des ressources du modèle, qui seront ensuite souvent restituées sous un format tabulaire
2. **ASK** : indique si la requête retourne un résultat non vide
3. **DESCRIBE** : obtenir des informations à propos de ressources présentes dans le modèle (le moins exploité des quatre)
4. **CONSTRUCT** : la requête sert de "template" pour construire de nouveaux graphes RDF en guise de résultats

Prise en charge des autres opérations CRUD depuis SPARQL 1.1

ordre SELECT : exploiter des patterns de graphes

PREFIX indique l'adresse (espace de noms) d'un schéma pouvant être exploité ensuite dans la construction de la requête

SELECT...[FROM]...WHERE retourne les ressources qui sont associées aux variables liées dans la clause WHERE

UNION groupes de patterns de graphes alternatifs (correspond à au au moins un des graphes précisés)

OPTIONAL groupes de patterns de graphes dont un au moins esr requis et un au moins est optionnel (si l'information est présente)

FILTER rajouter des conditions devant être satisfaites notamment sur les littéraux - des fonctions peuvent venir se surajouter

. concaténation de groupes de patterns de graphes

ordre SELECT : graphe requête

Appariement de graphes pour trouver des occurrences dans le graphe source - éléments manipulés : variables, ressource URI ou anonyme, littéraux (constantes)

PREFX Espaces de nommage

SELECT Variables dont on veut les valeurs résultats

WHERE {

patrons de graphe : syntaxe proche de N3

}

LIMIT
ORDER BY — améliorer affichage

Figure: Généralités sur une requête SELECT

Exemples tirés de la documentation ARQ - Jena

Présentation du vocabulaire exemple

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:vCard='http://www.w3.org/2001/vcard-rdf/3.0#'
  xmlns:info='http://somewhere/peopleInfo#'>

<rdf:Description rdf:about="http://somewhere/RebeccaSmith/">
    <vCard:FN>Becky Smith</vCard:FN>
    <info:age>23</info:age>
    <vCard:N rdf:parseType="Resource">
      <vCard:Family>Smith</vCard:Family>
      <vCard:Given>Rebecca</vCard:Given>
    </vCard:N>
  </rdf:Description>
```

Listing 4: Jeu de données RDF

Exemples tirés de la documentation ARQ - Jena

Exemple d'interrogation simple, les variables à lier sont préfixées par **?** ou **$** et peuvent correspondre à tout type noeud présent dans le modèle RDF

```
SELECT ?person
WHERE
 { ?person <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John
     Smith" }

PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?person
WHERE {
   ?person vCard:FN "John Smith" }
```

Listing 5: Notion de variable

Exemples tirés de la documentation ARQ - Jena

Plusieurs variables

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?person ?fname
WHERE {?person vCard:FN ?fname}
```

Listing 6: Illustration variables

Exemples tirés de la documentation ARQ - Jena

BGP - Basic Graph Pattern - Ensemble collection de triplets - Le . : conjonction entre patterns

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?givenName
WHERE
  { ?y vCard:Family "Smith" .
    ?y vCard:Given ?givenName .
  }
```

Listing 7: Illustration conjonction de patterns

Exemples tirés de la documentation ARQ - Jena

Nœud anonyme consultable comme toute autre ressource

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?y ?givenName
WHERE
 { ?y vcard:Family "Smith" .
   ?y vcard:Given ?givenName .
 }
```

Listing 8: Illustration noeud anonyme

Exemple Primitives RDF/RDFS

Exprimer des requêtes sur les représentations sous-jacentes RDF, RDFS

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object }

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?subject ?object
WHERE { ?subject rdf:type ?object }
```

Listing 9: Consulter les modèles sous-jacents

Accès au jeu de données

Accéder à l'ensemble du jeu de données

```
SELECT *
{ ?s ?p ?o }
```

Listing 10: Requête ouverte

Exemple Union

Clause UNION : réaliser une union de graphes alternatifs

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX famille: <http://www.lirmm.fr/famille#>
SELECT ?individu WHERE {{?individu
rdf:type family:Femme }
 UNION {?individu rdf:type family:Homme }}
```

Listing 11: énoncé UNION

Exemple OPTIONAL

Clause OPTIONAL : restituer l'information potentiellement associée à des sous-graphes

```
PREFIX info: <http://somewhere/peopleInfo#>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?name ?age
WHERE
{
   ?person vCard:FN ?name .
   OPTIONAL { ?person info:age ?age }
}
```

Listing 12: énoncé OPTIONAL

Poser des filtres

syntaxe accompagnant les filtres (FILTER) - conditions facilitant le rejet de résultats non souhaités

- Opérateurs logiques (OR, AND, NOT) : &&, ||, ! pour construire des expressions à évaluer
- Opérateurs arithmétiques : +, -, *, /
- Opérateurs de comparaison =, !=, <, >, <=, >=
- Tests SPARQL : isURI, isBlank, isLiteral, bound
- Accesseurs SPARQL : str, lang, datatype
- Autres opérateurs : sameTerm, langMatches, regex

Exemple Filtre et fonction regex

Exploiter les expressions régulières

```
FILTER regex(?x, "pattern" [, "flags"])

PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?g
WHERE
{ ?y vCard:Given ?g .
  FILTER regex(?g, "r", "i") }
```

Listing 13: Illustration FILTER

Exemple Filtre et comparaison

Personnes d'au moins 24 ans

```
PREFIX info: <http://somewhere/peopleInfo#>

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

Listing 14: Illustration FILTER

Filtre + facultatif

Personnes et eventuellement leur âge quand plus de 24 ans

```
PREFIX info: <http://somewhere/peopleInfo#>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?name ?age
WHERE
{
?person vCard:FN ?name .
OPTIONAL { ?person info:age ?age . FILTER ( ?age > 24 ) }
}
```

Listing 15: Illustration FILTER et OPTIONAL

Filtre + facultatif + Variable non liée

Personnes dont l'âge est inconnu ou qui ont plus de 24 ans

```
PREFIX info: <http://somewhere/peopleInfo#>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?name ?age
WHERE
{
   ?person vCard:FN ?name .
   OPTIONAL { ?person info:age ?age . }
   FILTER ( !bound(?age) || ?age > 24 )
}
```

Listing 16: FILTER

Exemple Filtre

Exploiter les filtres avec opérateurs logiques

```
SELECT DISTINCT ?b ?start ?end
WHERE
{
?b a b:EmployeeBooking ;
  b:startDate ?start ;
  b:endDate ?end .
  FILTER (
    (?start > "2008-03-05T09:00:00"^^xsdt:dateTime
 && ?start < "2008-03-07T09:00:00"^^xsdt:dateTime)||
    (?start < "2008-03-05T09:00:00"^^xsdt:dateTime
 && ?end > "2008-03-05T09:00:00"^^xsdt:dateTime)
  )
}
```

Listing 17: Opérateurs logiques

Négation par l'échec avec SPARQL 1.0

Prédicats OPTIONAL et BOUND

```
# Personnes avec age non renseigne
SELECT ?person WHERE { ?person vCard:FN ?fname .
  OPTIONAL { ?person info:age ?age } .
  FILTER ( !bound(?age)) }
```

Listing 18: Négation par l'échec

Not Exists avec SPARQL 1.1

Prédicats OPTIONAL et BOUND

```
# Personnes avec age non renseigne
"SELECT ?person WHERE { ?person vCard:FN ?fname .
FILTER NOT EXISTS { ?person info:age ?age } }
```

Listing 19: NOT EXISTS

Test de vacuité

Prédicats Exists et Not Exists avec filter

```
# Personnes qui connaissent d'autres personnes au temps t
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:givenName ?name .
FILTER (EXISTS { ?x foaf:knows ?who })
    }
```

Listing 20: EXISTS

Partitionnement

Syntaxe Group By avec opérations count, sum

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?person (COUNT(?tel) AS ?nbreTel) " +
 WHERE { ?person vCard:FN ?fname .
  ?person vCard:TEL ?tel . } " +
 GROUP BY ?person
```

Listing 21: Illustration Partitionnement

Partitionnement

Syntaxe Group By avec condition having

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?person (COUNT(?tel) AS ?nbreTel) " +
 WHERE { ?person vCard:FN ?fname .
  ?person vCard:TEL ?tel . } " +
 GROUP BY ?person HAVING(COUNT(?tel) > 3)
```

Listing 22: Illustration Partitionnement avec condition

SPARQL avec Jena : exemple partiel

```
String prolog2 = "PREFIX rdf: <"+RDF.getURI()+">" ;
String queryString = prolog1 + NL + prolog2 + NL +
"SELECT ?ind WHERE {?ind rdf:type family:Femme }" ;
Query query = QueryFactory.create(queryString) ;
QueryExecution qexec =
QueryExecutionFactory.create(query, m) ;
 try {
 ResultSet rs = qexec.execSelect() ;
 for ( ; rs.hasNext() ; )
 {
 QuerySolution rb = rs.nextSolution() ;
   RDFNode y = rb.get("ind");
...}
```

Listing 23: Illustration RDF/XML

SPARQL avec Jena : exemple partiel

```
String prolog2 = "PREFIX rdf: <"+RDF.getURI()+">" ;
String queryString = prolog1 + NL + prolog2 + NL +
"SELECT ?ind WHERE {?ind rdf:type family:Femme }" ;
Query query = QueryFactory.create(queryString) ;
QueryExecution qexec =
QueryExecutionFactory.create(query, m) ;
query.serialize(new IndentedWriter(System.out,true)) ;
      try { ResultSet rs = qexec.execSelect() ;
 ResultSetFormatter.out(System.out,rs,query); }
      finally
      {  qexec.close() ; }
```

Listing 24: Illustration Jena et ARQ

SPARQL avec Jena : exemple partiel

```
import com.hp.hpl.jena.rdf.model.AnonId;
...
import com.hp.hpl.jena.rdf.model.RDFVisitor;

public class Skos_UnVisiteur implements RDFVisitor {
public Object visitBlank(Resource r, AnonId id) {
System.out.println("anon: " + id); return null;}

public Object visitURI(Resource r, String uri) {
System.out.println("qualified name : " + r.getLocalName());
return null;}

public Object visitLiteral(Literal l) {
System.out.println(l); return null;}}
```

Listing 25: Illustration Jena et ARQ

Exemple CONSTRUCT

Restituer une portion de modèle au format RDF

```
CONSTRUCT { ?person info:nomComplet ?fname }
WHERE {?person vCard:FN ?fname }
```

Listing 26: Illustration CONSTRUCT

Exemple CONSTRUCT

Exemple de résultat

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix vCard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix info: <http://somewhere/peopleInfo#> .

info:JohnSmith info:nomComplet "John Smith" .
info:MattJones info:nomComplet "Matt Jones" .
info:RebeccaSmith info:nomComplet "Becky Smith" .
info:SarahJones info:nomComplet "Sarah Jones" .
```

Listing 27: Illustration CONSTRUCT

Exemple CONSTRUCT

Implémentation Jena

```
 String rdq = prolog1 + NL + prolog2 + NL + prolog3 + NL +
"CONSTRUCT { ?person info:nomComplet ?fname } WHERE {?person
   vCard:FN ?fname }" ;

 Query query = QueryFactory.create(rdq);
 QueryExecution qexec = QueryExecutionFactory.create(query, m);

Model results = qexec.execConstruct();
results.write(System.out, "N3");
qexec.close();
```

Listing 28: Illustration CONSTRUCT

Exemple ASK

Evaluer la validité d'une requête sur un modèle

```
ASK {?person vCard:FN ?fname }
```

Listing 29: Illustration ASK

avec pour résultats : Ask result = TRUE

Exemple ASK

Implémentation Jena

```
 String rdq = prolog1 + NL + prolog2 + NL + prolog3 + NL +
 "ASK {?person vCard:FN ?fname }" ;

Query query = QueryFactory.create(rdq);
QueryExecution qexec = QueryExecutionFactory.create(query, m);

boolean b = qexec.execAsk();
System.out.println("Ask result = " + ((b)?"TRUE":"FALSE"));
qexec.close();
```

Listing 30: Illustration ASK

Exemple DESCRIBE

Restituer la description d'une ressource

```
DESCRIBE ?person WHERE {?person vCard:FN ?fname }
```

Listing 31: Illustration DESCRIBE

Exemple DESCRIBE

Exemple de résultat

```
@prefix rdfs: ....

info:JohnSmith info:age "\"25\"^^xsd:integer" ;
      vCard:FN "John Smith" ;
      vCard:N [ vCard:Family "Smith" ;
                vCard:Given "John"
              ] ;
      vCard:TEL [ a     vCard:work , vCard:voice ;
                rdf:value "+44 117 555 5555"
              ] ;
      vCard:TEL [ a     vCard:home , vCard:voice ;
                rdf:value "+44 117 555 7777"
              ] .
```

Listing 32: Illustration DESCRIBE

Exemple DESCRIBE

Implémentation Jena

```
// Query string.
 String rdq = prolog1 + NL + prolog2 + NL + prolog3 + NL +
 "DESCRIBE ?person WHERE {?person vCard:FN ?fname }" ;
 Query query = QueryFactory.create(rdq);
 QueryExecution qexec = QueryExecutionFactory.create(query,
     m);

 Model results = qexec.execDescribe();
 results.write(System.out, "N3");
 qexec.close();
```

Listing 33: Illustration DESCRIBE