Consultation des données : langage SPARQL et endpoints associés

I.Mougenot

UM

$HMIN218$ 2021

UNIVERSITÉ DE MONTPELLIER

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)
- 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

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: SimpleSelector

Simple Protocol And RDF Query Language

SPARQL : standard W3C pour faciliter l'interrogation de sources de données distribuées

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

Architecture du web

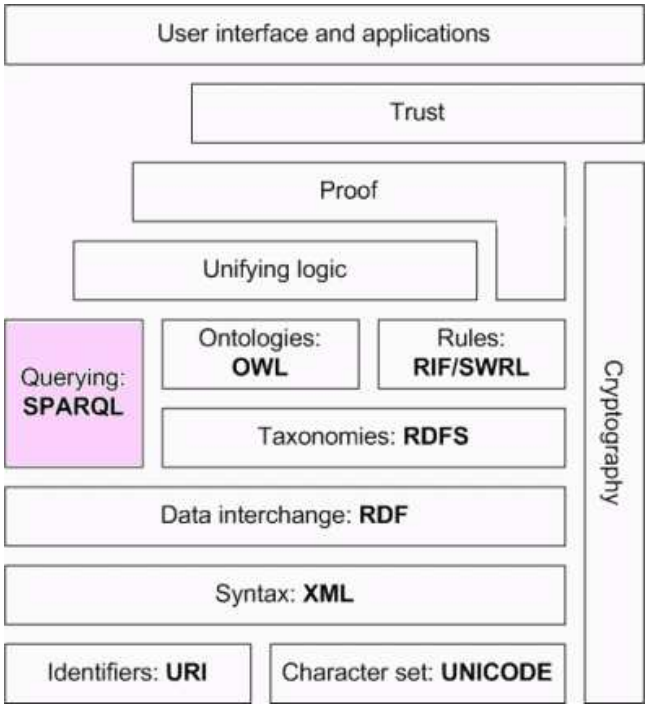


Figure: Empilement de couches

Illustration protocole

Requêtage mais aussi communication

```
curl -X POST https://sparql.uniprot.org/sparql/
--data-urlencode 'query=SELECT * {
    <http://purl.uniprot.org/uniprot/P61221> ?p ?o }
    LIMIT 2'
--data-urlencode 'format=json'
```

Listing 2: Avec le client curl

Exemples de points d'accès SPARQL

Divers domaines cibles (non exhaustif) (voir `https://www.wikidata.org/wiki/Wikidata:Lists/SPARQL_endpoints`)

1. Wikidata `https://query.wikidata.org/`
2. UniProtKb `https://sparql.uniprot.org/`
3. INSEE `https://rdf.insee.fr/sparql`
4. BNF `https://data.bnf.fr/sparql/`
5. Korean Intellectual Property Office `http://lod.kipo.kr/data/sparql`
6. Getty Museum `http://vocab.getty.edu/sparql`

SPARQL en tant que langage d'interrogation

Quatre formes de requêtage et dans les dernières évolutions du langage, également LDD

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

Ordre SELECT : graphe requête

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

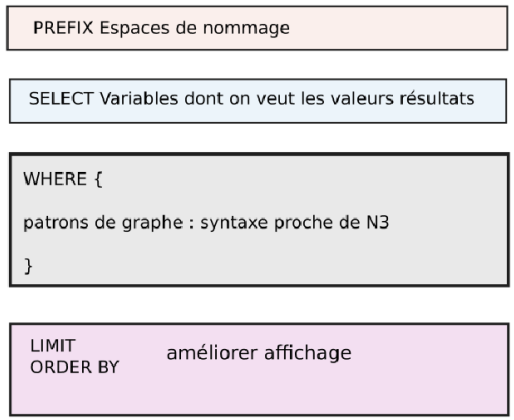


Figure: Généralités sur une requête SELECT

Ordre SELECT : exploiter des patterns de graphes

PREFIX espace de noms d'un schéma exploitable 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

Exemple de base

Exemple d'interrogation simple, les variables libres sont préfixées par **?** et peuvent correspondre à tout noeud (ressource comme littéral) présent dans le modèle RDF

```
SELECT ?subject ?property ?object
WHERE { ?subject ?property ?object}
```

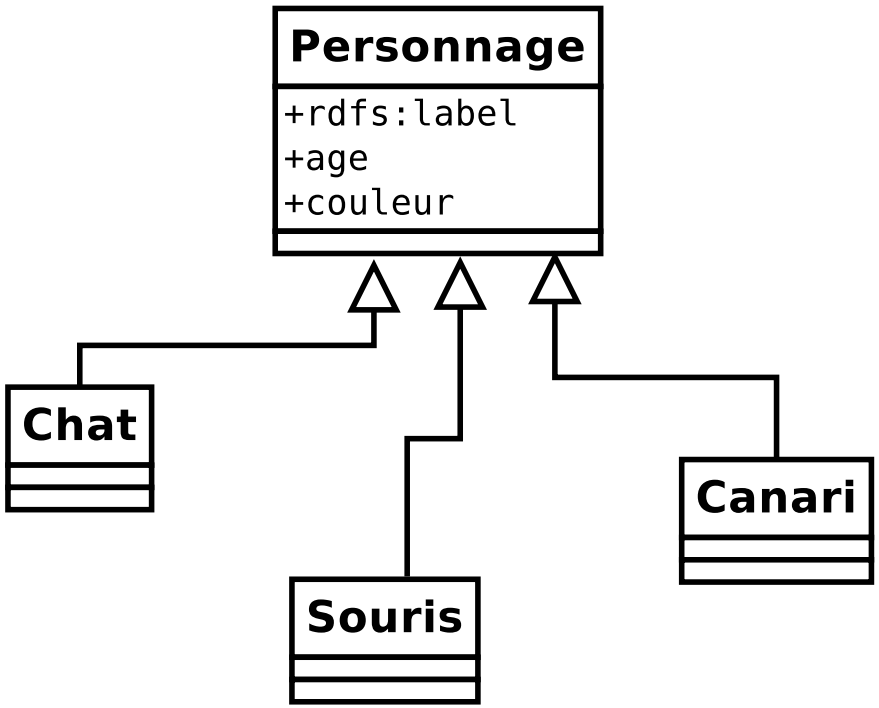
Exemple Primitives RDF/RDFS

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

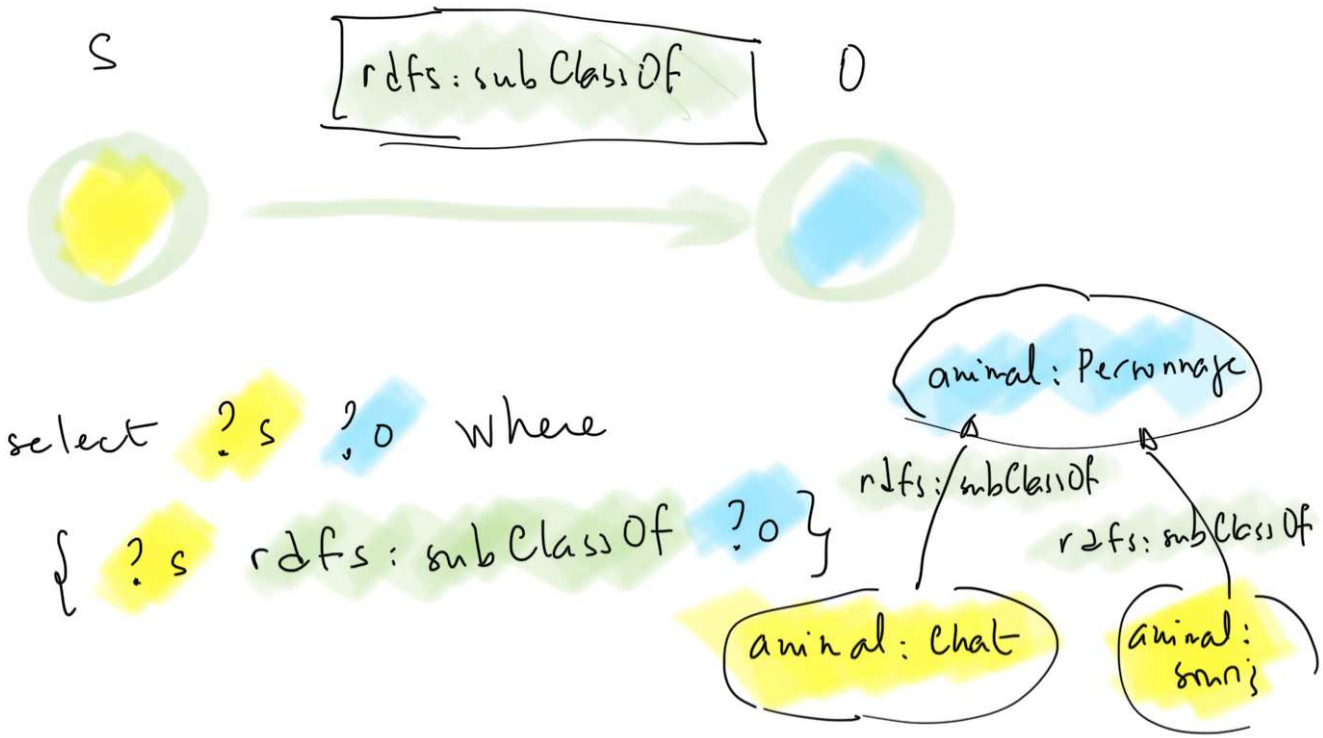
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object }

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

Diagramme de classes exemple : Cartoons



Principes SPARQL

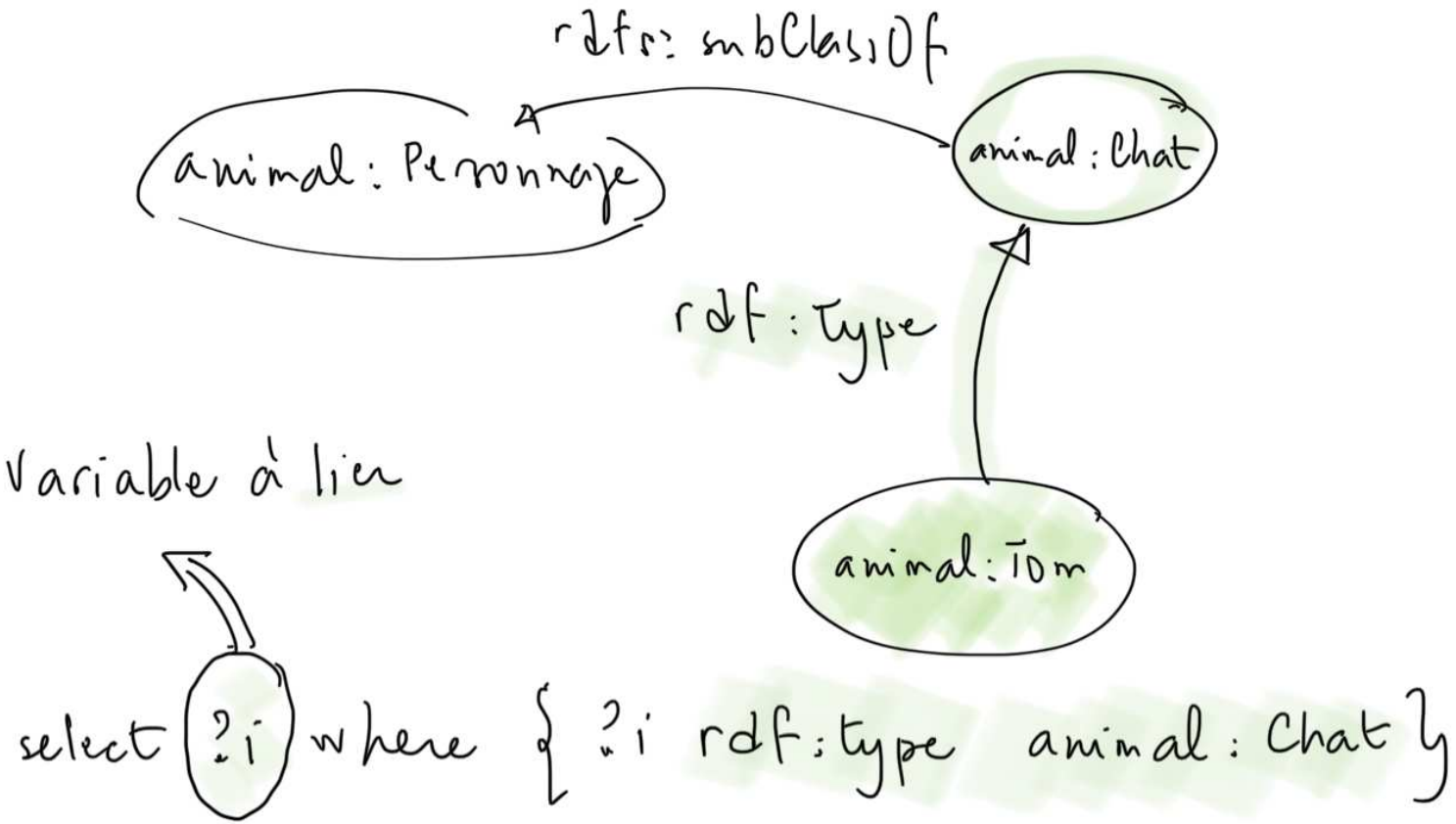


Individus d'une classe Chat depuis le graphe

```
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX animal: <http://www.ex.fr/animal#>

SELECT ?s
WHERE
  { ?s rdf:type animal:Chat }
```

Régularité sur le graphe



Exemple Conjonction

Patterns de graphes incorporant différentes constructions. Le . nous permet d'exprimer une concaténation de liaisons de variables

```
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX animal: <http://www.ex.fr/animal#>

 SELECT ?s ?age
 WHERE
   { ?s rdf:type animal:Chat .
     ?s animal:age ?age
   }
```

Exemple d'information facultative

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

```
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX animal: <http://www.ex.fr/animal#>

SELECT ?s ?age
 WHERE
   { ?s rdf:type animal:Chat
     OPTIONAL
       { ?s animal:age ?age }
   }
```

Exemple Filtre et fonction regex

Filtre sur le nom du label qui doit commencer par T

```
1 PREFIX animal: <http://www.ex.fr/animal#>
 2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
 3 PREFIX rdf:
     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 4
 5 SELECT ?s ?label
 6 WHERE
 7   { ?s rdfs:label ?label
 8     FILTER regex(?label, "^T")
 9   }
```

Exemple Filtre et clause LIMIT

Filtre sur le nom du label qui finit pas i dans la limite des six premiers résultats

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

 SELECT ?s ?label
 WHERE
   { ?s rdfs:label ?label
    FILTER regex(?label, "i$")
   }
LIMIT 6
```

Exemple Filtre et variable liée

Test de liaison : individus qui ont un âge

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?s ?age
WHERE
  { ?s animal:age ?age
    FILTER bound(?age)
  }
```

Exemple Filtre

Test de liaison : individus de plus de 2 ans

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?s ?age
WHERE
   { ?s animal:age ?age
     FILTER ( ?age > 2 )
   }
```

Clause NOT EXISTS

individus ayant au moins un couleur mais sans âge

```
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

 SELECT ?s ?couleur
 WHERE
   { ?s animal:couleur ?couleur
     FILTER NOT EXISTS { ?s animal:age ?age }
   }
```

Variable non liée, négation et optional

individus ayant au moins un couleur mais sans âge

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

 SELECT ?s ?couleur
 WHERE
   { ?s animal:couleur ?couleur
     OPTIONAL
       { ?s animal:age ?age }
     FILTER ( ! bound(?age) )
   }
```

Exemple Disjonction : Opérateur Union

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?p ?label
WHERE
  { ?p rdfs:label ?label
      { ?p rdf:type animal:Chat }
    UNION
      { ?p rdf:type animal:Souris }
    UNION
      { ?p rdf:type animal:Canari }
 }
```

Remarque : même chose que :

```
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX animal: <http://www.ex.fr/animal#>

SELECT ?s
WHERE
  { ?s rdf:type animal:Personnage }
```

Personnage notion de classe partition et requêtage sur modèle inféré

Agrégat et partitionnement

Nombre d'individus pour les classes du modèle

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?class (COUNT(?individu) AS ?nbreIndividus)
WHERE
  { ?individu rdf:type ?class }
GROUP BY ?class
```

Agrégat et partitionnement avec condition

Condition sur l'agrégat

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?class (COUNT(?individu) AS ?nbreIndividus)
WHERE
  { ?individu rdf:type ?class }
GROUP BY ?class
HAVING ( COUNT(?individu) > 1 )
```

SPARQL avec Jena : exemple partiel

```
String prolog2 = "PREFIX rdf: <"+RDF.getURI()+">" ;
String queryString = prolog1 + NL + prolog2 + NL +
"SELECT ?term WHERE {?term rdf:type go:term }" ;
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("term");
...}
```

Exemple CONSTRUCT

Sous-graphe et/ou transformation

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 CONSTRUCT
   {   ?i rdf:type animal:Chat .
     ?i rdfs:label ?label .
     ?i animal:couleur ?couleur . }
WHERE
  { ?i rdf:type    animal:Chat ;
       rdfs:label  ?label ;
       animal:couleur ?couleur }
```

Résultat CONSTRUCT

```
@prefix rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix animal: <http://www.ex.fr/animal#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

animal:Tom a       animal:Chat ;
      rdfs:label   "Tom" ;
      animal:couleur "gris" .

animal:Sylvester a animal:Chat ;
      rdfs:label   "Sylvester" ;
      animal:couleur "noir" , "blanc" .
```

Exemple DESCRIBE

Exploiter la clause DESCRIBE : focus sur la ressource Tom du graphe

```
PREFIX animal: <http://www.ex.fr/animal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

DESCRIBE ?tom
WHERE
  { ?tom rdf:type animal:Chat ;
         rdfs:label "Tom"
  }
```

Résultat DESCRIBE

```
@prefix rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix animal: <http://www.ex.fr/animal#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

animal:Tom a          animal:Chat ;
      rdfs:label    "Tom" ;
      animal:age    "4"^^xsd:int ;
      animal:couleur "gris" .
```

SPARQL avec Jena : exemple partiel (partie 1)

```
public class Ex_SPARQL_1
{
   public static final String NL =
       System.getProperty("line.separator") ;
   public static void main(String[] args)
   {
Model m = ModelFactory.createDefaultModel();
String fil_tom = "tom.n3";
String prolog1 = "PREFIX rdf: <"+RDF.getURI()+">" ;
m.read(fil_tom);
String rdq = prolog1 + NL +
    "SELECT ?s ?p ?o WHERE { ?s ?p ?o}" ;
Query query = QueryFactory.create(rdq);
QueryExecution qexec =
   QueryExecutionFactory.create(query, m);
```

SPARQL avec Jena : exemple partiel (partie 2)

```
try {
 Iterator<QuerySolution> results = qexec.execSelect();
 RDFVisitor aVisitor = new Un_Visiteur();
 for (;results.hasNext();)
        {
           QuerySolution sol = results.next();
           RDFNode s = sol.get("s");
           RDFNode p = sol.get("p");
           RDFNode o = sol.get("o");
           System.out.print(s.visitWith(aVisitor)+" ");
           System.out.print(p.visitWith(aVisitor)+" ");
           System.out.println(o.visitWith(aVisitor));
        } }
     finally {qexec.close();}
   }  }
```

Mise en pratique du pattern Visiteur

```
import org.apache.jena.rdf.model.AnonId;
import org.apache.jena.rdf.model.Literal;
import org.apache.jena.rdf.model.RDFVisitor;
import org.apache.jena.rdf.model.Resource;
public class Un_Visiteur implements RDFVisitor {
   public Object visitBlank(Resource r, AnonId id) {
   return "anon: " + id;
   }
   public Object visitURI(Resource r, String uri) {
   return r.getLocalName();
   }
   public Object visitLiteral(Literal l) {
   return l.getValue();
   }}
```

Eléments de formatage

```
String rdq = prolog1 + NL + prolog2 + NL + prolog3
    + NL +
   " SELECT .... " ;
   Query query = QueryFactory.create(rdq);
   QueryExecution qexec =
       QueryExecutionFactory.create(query, m);
   query.serialize(new
       IndentedWriter(System.out,true)) ;
   System.out.println();
   try {
       ResultSet rs = qexec.execSelect() ;
       ResultSetFormatter.out(System.out, rs,
           query);
   }
   finally {qexec.close();}
 }
```

Source externe

```
  public static final String NL =
      System.getProperty("line.separator") ;
    public static void main(String[] args)
      {
Model m = ModelFactory.createDefaultModel();
m.read("https://www.wikidata.org/wiki/Special:EntityData/Q1311
String schema_ns = m.getNsPrefixURI("schema");
String wikidata_ns = m.getNsPrefixURI("wd");
String prolog1 = "PREFIX schema: <"+schema_ns+">" ;
String prolog3 = "PREFIX wd: <"+wikidata_ns+">" ;
String rdq = prolog1 + NL + prolog2 + NL + prolog3 + NL
    +
 "SELECT ?s ?name WHERE { ?s rdf:type schema:Article ."
 + " OPTIONAL {?s schema:name ?name } }" ;
        ...
```