

Endpoint SPARQL & Triplestore

I.Mougenot

UM

*HMIN*₂₁₈ 2021

Tendances du Web de données (Web Of Data)

Se rapprocher d'un système de connaissances intégré et ouvert à tous

- formats de représentation et protocoles de communication standards W3C
- sources de données ouvertes et liées (Linked Open Data ou LOD)
- gestion et accès des LOD

LOD

Sources de données aux formats du Web sémantique, de plus en plus nombreuses

- Privilégier la mutualisation de données structurées et interconnectées sur le Web
- Une Donnée Ouverte est une Donnée Liée, qui est publiée sous une licence ouverte (réutilisation gratuite) (T. Berners-Lee).
- Dans la réalité, les données liées ne sont pas toujours ouvertes, et les données ouvertes ne sont pas toujours liées

Les catégories considérées

Au nombre de 7 :

- ① media (media)
- ② géographique (geographic)
- ③ gouvernemental (government)
- ④ réseaux sociaux (user-generated content)
- ⑤ sciences de la vie (life sciences)
- ⑥ publications (publications)
- ⑦ généraliste (cross-domaine)

Sources de données ouvertes et liées (LOD)

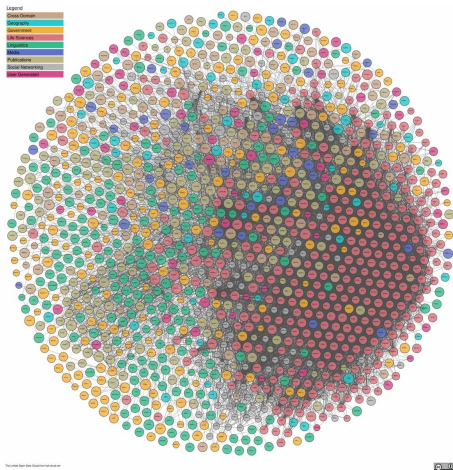


Figure: Un nuage de sources

Exemples de sources de données ouvertes et liées (LOD)

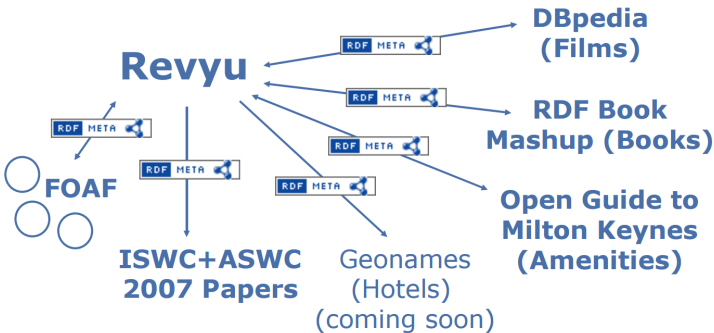


Figure: Liens entre un site d'avis critiques et des sites de diverses ressources

Définition Endpoint SPARQL

Endpoint SPARQL : nœud qui fait office d'extrémité à un canal de communication SPARQL

- recevoir et traiter des requêtes émises au travers du protocole SPARQL

Synthèse visuelle

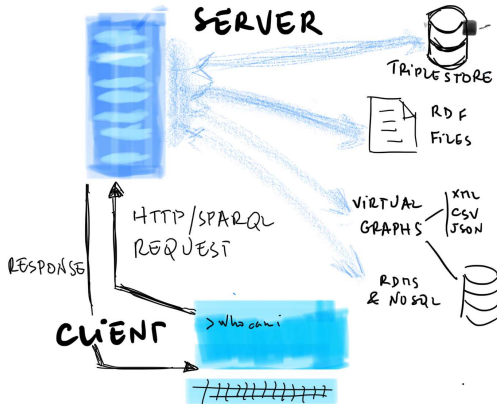


Figure: Schéma général Endpoint

Exemple EndPoint SPARQL Wikidata

The screenshot shows the Wikidata Query Service interface. The top navigation bar includes the Wikidata logo, the text "Wikidata Query Service", and a language selector set to "français". Below the navigation bar, there are tabs for "Exemples" and "Aide", and a dropdown menu for "Davantage d'outils".

The main content area is divided into two panels. The left panel, titled "Assistant de requête", contains a search bar with the text "http://schema.org/name", a "Limiter à 10" option, and a "Filtrer" button. The right panel displays a SPARQL query:

```
1 SELECT ?s ?name WHERE { ?s rdf:type schema:Article.
2   OPTIONAL {?s schema:name ?name }
3 LIMIT 10
```

Below the query, there is a table with 10 results. The table has two columns: "s" (URI) and "name" (Label). The results are as follows:

s	name
https://ab.wikipedia.org/wiki/D0%90%D0%B1%D2%B5%D0%B0%D1%80%D0%B0%D0%B7%D0%B0	Абдарапка
https://ab.wikipedia.org/wiki/D0%90%D0%BA%D0%BE%D0%BC%D0%B8%D1%83%D1%82%D0%B5%D1%80	Акомплутер
https://ab.wikipedia.org/wiki/D0%90%D0%BB%D0%B0	Ала
https://ab.wikipedia.org/wiki/D0%90%D0%BB%D1%8B%D0%BC	Алья
https://ab.wikipedia.org/wiki/D0%90%D0%BD%D1%82%D0%B0%D1%80%D0%BA%D1%82%D0%B8%D0%B4%D0%B0	Антарктида
https://ab.wikipedia.org/wiki/D0%91%D0%B5%D0%BB%D1%8C%D0%B3%D0%B8%D0%B0	Бельгия

At the bottom of the interface, there is a status bar showing "10 résultats en 603 ms", a "Code" button, a "Télécharger" button, and a "Lien" button.

Sources de données ouvertes et liées (LOD)

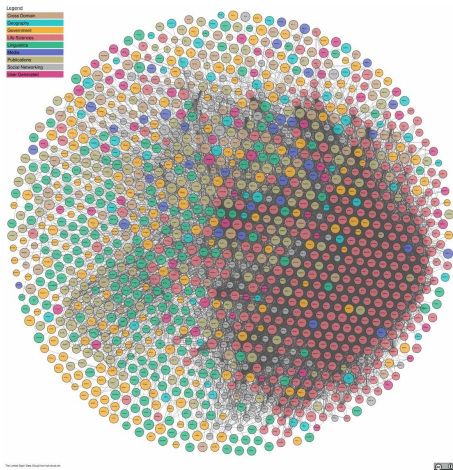
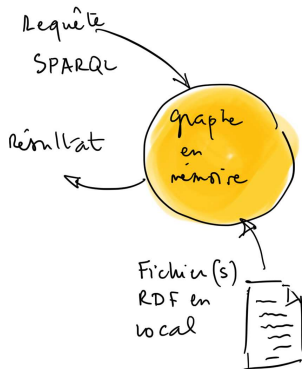


Figure: Autant d'interfaces de communication potentielles

Vu jusqu'à présent : local



Dans cette séance : externalisation

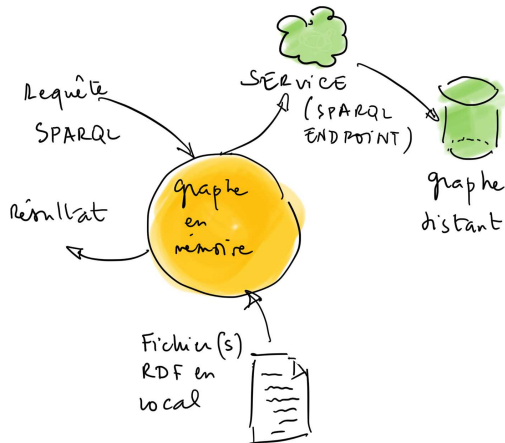


Figure: Accès au contenu d'une source de données externe

Wikidata

Item [Discussion](#)

Tom and Jerry (Q131144)

series of theatrical animaed cartoon films [edit](#)

[+ In more languages](#)

Language	Label	Description	Also known as
English	Tom and Jerry	series of theatrical animaed cartoon films	
French	Tom et Jerry	série américaine de dessins animés de courte durée	Tom et Jerry Kids Show Tom et Jerry Tom and Jerry Tom et Jerry show The New Tom & Jerry Show
Spanish	Tom y Jerry	serie de dibujos animados de Hanna Barbera	Tom & Jerry Tom and Jerry Jerry Mouse Tom Cat
German	Tom und Jerry	Serie von 161 kurzen Zeichentrickfilmen	Kater Tom Tom und Jerry


All entered languages

Statements

instance of [animated film series](#) [edit](#)

[+ 1 reference](#)

[+ add value](#)

logo image  [edit](#)

Wikipedia (96 entries) [edit](#)

- am [توم و جیری](#)
- ar [توم وجیری](#)
- ast [Tom y Jerry](#)
- azb [تام و جیری](#)
- az [Tom va Cerri](#)
- ban [Tom and Jerry](#)
- bar [Tom und Jerry](#)
- be [Том і Джэры](#)
- bg [Том и Джери](#)
- bh [ટમ \(ટો\)](#)
- bm [Tom and Jerry](#)
- bn [টম জেরি](#)
- ca [Tom i Jerry](#)
- ce [Тӕмӕт Джӕрри](#)
- ckb [تۆم و جیری](#)
- cs [Tom a Jerry](#)
- da [Tom og Jerry](#)
- de [Tom und Jerry](#)
- dv [ޏޮމް ޖަރީ](#)
- el [Τομ και Τζέρι](#)
- en [Tom and Jerry](#)
- eo [Tom kaj Jerry](#)
- es [Tom y Jerry](#)
- et [Tom ja Jerry](#)
- eu [Tom eta Jerry](#)
- fa [تام و جیری](#)

Figure: Schéma Page Tom and Jerry

Une ressource cible d'une 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/Q131144.ttl");
    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 } }" ;
    ...
}
```

Endpoint : notion de service

```
public static void main(String[] args)
{
    String sparqlService = "https://query.wikidata.org/sparql";
    String prolog2 = "PREFIX rdf: <" + RDF.getURI() + ">" ;
    String prolog1 = "PREFIX schema: <http://schema.org/>";
    String rdq = prolog1 + NL + prolog2 + NL +
        "SELECT ?s ?name WHERE { ?s rdf:type schema:Article. "
        + "  OPTIONAL {?s schema:name ?name } FILTER (lang(?name) = 'en') } "
        + " LIMIT 10 " ;
    Query query = QueryFactory.create(rdq);
    QueryExecution qexec = QueryExecutionFactory.sparqlService(sparqlService,
        query);
    try {
        ResultSet rs = qexec.execSelect() ;
        ResultSetFormatter.out(System.out, rs, query);
    }
    finally {qexec.close();}
}
```

Listing 1: Avec Wikidata

Endpoint DBPedia

```
String sparqlService = "http://dbpedia.org/sparql";
String prolog2 = "PREFIX rdf: <"+RDF.getURI()+">" ;
String prolog1 = "PREFIX db-onto: <http://dbpedia.org/ontology/>";
String prolog5 = "PREFIX db-resource: <http://fr.dbpedia.org/resource/>";

String rdq = prolog1 + NL + prolog2 + NL + prolog5 + NL +
    "SELECT ?ville ?region WHERE { "
    + " ?ville rdf:type db-onto:Settlement ."
    + " ?ville db-onto:region ?region ."
    + "} LIMIT 100";

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

Listing 2: Avec DbPedia (extrait)

Ressource Tom and Jerry

Ressource Q131144 dans Wikidata

```
PREFIX wd: <http://www.wikidata.org/entity/>  
PREFIX wdt: <http://www.wikidata.org/prop/direct/>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
  
SELECT ?s ?p ?o WHERE { bind(wd:Q131144 as ?s) ?s ?p ?o }
```

Listing 3: requête SPARQL

Ressource Tom and Jerry

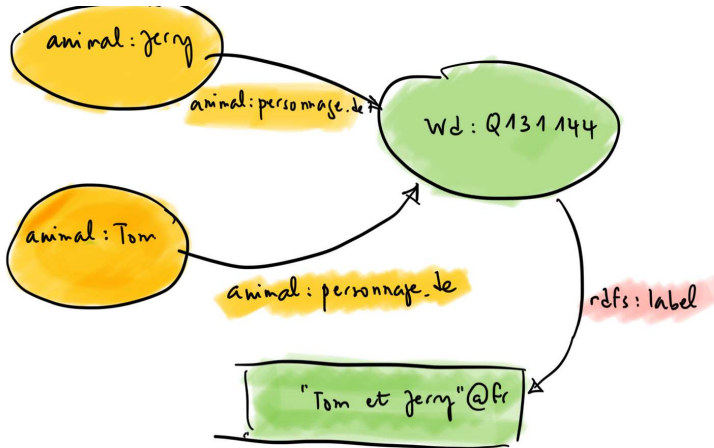
Ressource Q131144 instance de "série de films d'animation"

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?s ?label ?o ?label_s WHERE { bind(wd:Q131144 as ?s)
?s rdfs:label ?label ; wdt:P31 ?o .
?o rdfs:label ?label_s
filter(lang(?label) ='fr' && lang(?label_s) ='fr') }
```

Listing 4: nouvelle requête

Interconnecter le graphe construit en local avec la ressource wikidata



Ressource Tom and Jerry

Interconnecter le graphe construit en local avec la ressource wikidata

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wdt: <http://www.wikidata.org/prop/direct/> .
@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#> .
@prefix wd: <http://www.wikidata.org/entity/> .

wd:Q56884562 rdfs:label "serie de films d'animation"@fr .

animal:Personnage rdfs:subClassOf rdfs:Class .
animal:Tom a          animal:Chat ;
    rdfs:label        "Tom" ;
    animal:age         "4"^^xsd:int ;
    animal:couleur     "gris" ;
    animal:personnage_de wd:Q131144 .

wd:Q131144 a      wd:Q56884562 ;
    rdfs:label    "Tom et Jerry"@fr .

animal:Chat rdfs:subClassOf animal:Personnage .
```

Listing 5: connexion avec l'existant

Code associé 1/3 (à améliorer)

```
public class Wikidata_3
{
    public static final String NL = System.getProperty("line.separator");
    public static final String rdf_file = "tom.n3";

    public static void main(String[] args) {
        try {
            Model m = ModelFactory.createDefaultModel();
            m.read(rdf_file);
            String a_ns = m.getNsPrefixURI("animal");
            Resource tom = m.getResource(a_ns + "Tom");
            Resource jerry = m.getResource(a_ns + "Jerry");
            Property personnage_de = m.createProperty(a_ns + "personnage_de");
            m.setNsPrefix("wd", "http://www.wikidata.org/entity/");
            m.setNsPrefix("wdt", "http://www.wikidata.org/prop/direct/");
            String sparqlService = "https://query.wikidata.org/sparql";
```

Listing 6: partie 1

Code associé 2/3 (à améliorer)

```
String prolog2 = "PREFIX rdfs: <" + RDFS.getURI() + ">";
String prolog1 = "PREFIX schema: <http://schema.org/>";
String prolog3 = "PREFIX wd: <http://www.wikidata.org/entity/>";
String prolog4 = "PREFIX wdt: <http://www.wikidata.org/prop/direct/>";

String rdq = prolog1 + NL + prolog2 + NL + prolog3 + NL + prolog4 + NL
    + "SELECT ?s ?label ?o ?label_s WHERE { bind(wd:Q131144 as ?s) "
    + " ?s rdfs:label ?label ; wdt:P31 ?o . "
    + " ?o rdfs:label ?label_s filter(lang(?label) ='fr' &&
        lang(?label_s) ='fr') }";

Query query = QueryFactory.create(rdq);
QueryExecution qexec = QueryExecutionFactory.sparqlService(sparqlService,
    query);
Resource cartoon = null;
Resource cartoon_type = null;
ResultSet results = qexec.execSelect();
```

Listing 7: partie 2

Code associé 3/3 (à améliorer)

```
for (; results.hasNext();) {
    QuerySolution sol = results.next();
    cartoon = (Resource) sol.get("?s");
    cartoon_type = (Resource) sol.get("?o");
    Literal label = (Literal) sol.get("?label");
    Literal label_s = (Literal) sol.get("?label_s");
    m.add(tom, personnage_de, cartoon);
    m.add(jerry, personnage_de, cartoon);
    m.add(cartoon, RDF.type, cartoon_type);
    m.add(cartoon, RDFS.label, label);
    m.add(cartoon_type, RDFS.label, label_s);
}
gexec.close();
m.write(System.out, "N3");
FileOutputStream outputStream = new FileOutputStream("tomAndWikidata.n3"); //
m.write(outputStream, "N3");
outputStream.close();
} catch (Exception e) {
    System.out.println("failure" + e);
} finally {
}
}
```

Listing 8: partie 3

Rendre persistant un jeu de données RDF (triplestores)

Différentes solutions de gestion plus ou moins performantes

- solutions adossées à du relationnel (mapping r2rml, d2rq ...)
- Jena TDB
- Virtuoso
- 4Store
- Sesame
- ...

voir <https://www.w3.org/wiki/LargeTripleStores>

Modules fonctionnels Jena

Modèles en mémoire vive et rendus permanents

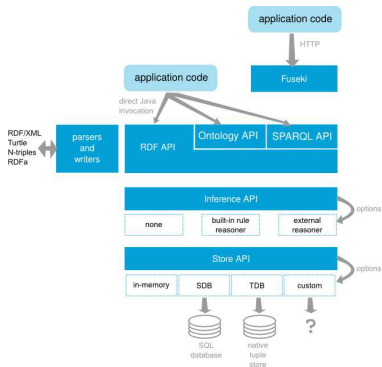


Figure: Composants plateforme Jena Apache

Jena TDB, Dataset, Model, Datasource

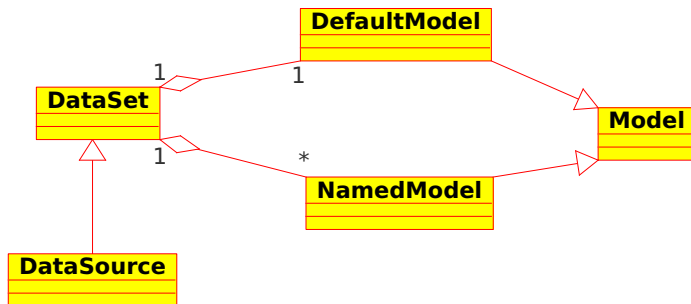


Figure: Diagramme de classes d'illustration

Construire un dataset associé à un modèle par défaut

```
public class TripleStore1
{
    public static final String rdf_file = "tomAndWikidata.n3";
    public static void main(String[] args)
    {
        String directory = "/home/isa/Bureau/TS_Tom" ;
        Dataset ds = TDBFactory.createDataset(directory) ;
        Model model = ds.getDefaultModel() ;
        model.read(rdf_file) ;
        StmtIterator stmt_i = model.listStatements();
        System.out.println("Objets des triplets du modele ");
        System.out.println(" ===== ");
        while (stmt_i.hasNext())
        {
            Statement stmt = stmt_i.nextStatement();
            RDFNode o = stmt.getObject();
            ...
        }
        ds.close();
    }
}
```

Listing 9: Création dataset

Enrichir un dataset associé à un modèle par défaut

```
public class TripleStore4
{
    public static void main(String[] args)
    {
        String directory = "/home/isa/Bureau/TS_Tom" ;
        Dataset dataset = TDBFactory.createDataset(directory) ;
        Model model = dataset.getDefaultModel() ;
        String a_ns = model.getNsPrefixURI("animal");
        Resource bugs_bunny = model.createResource(a_ns+"Bugs_Bunny");
        Resource lapin = model.createResource(a_ns+"Lapin");
        model.add(bugs_bunny, RDF.type, lapin);
        dataset.close();
    }
}
```

Listing 10: Ajout de triplets

Exploiter le dataset construit

```
public class TripleStore3
{
    public static void main(String[] args)
    {
        String directory = "/home/isa/Bureau/TS_Tom" ;
        Dataset dataset = TDBFactory.createDataset(directory) ;
        Model model = dataset.getDefaultModel() ;
        System.out.println("nombre de triplets : "+model.size());
        model.write(System.out,"N3");
        dataset.close();
    }
}
```

Listing 11: Exploiter le dataset

Consulter avec SPARQL le dataset construit

```
public class TripleStore2
{
    public static void main(String[] args)
    {
        String directory = "/home/isa/Bureau/TS_Tom" ;
        Dataset dataset = TDBFactory.createDataset(directory) ;

        String sparqlQueryString = "SELECT * WHERE " +
            " { ?s ?p <http://www.ex.fr/animal#Souris> } " ;

        Query query = QueryFactory.create(sparqlQueryString) ;
        QueryExecution qexec = QueryExecutionFactory.create(query, dataset) ;
        ResultSet results = qexec.execSelect() ;
        ResultSetFormatter.out(results) ;
        qexec.close() ;

        dataset.close();
    }
}
```

Listing 12: Consultation SPARQL