Rapport de Projet GMIN332 : Gestion de données complexes

Par : ALIJATE Mehdi - COUSOT Kevin - NEGROS Hadrien

15 Janvier 2014

Table des matières

Résumé

Accès et consultation de données provenant de différentes solutions de persistance (gros volumes de données distribuées et hétérogènes) au travers d'un démonstrateur.

1 Introduction

Les systèmes NoSQL et les technologies du Web Sémantique sont une altérnative aux SGBD classiques. Cependant, elles sont encore très loin d'être couramment utilisées, comme celles du monde relationnel "classique".
Néanmoins, de grands acteurs d'internet (*comme Facebook (Cassandra Project puis HBase), Google (BigTable), Ubuntu One (CouchDB)... etc.*) commencent à exploiter des bases de données de type NoSQL. L'avantage de celà, c'est que la majorité de ces projets est open source et sous licence libre.
Dans notre projet, nous avons voulu explorer les différentes technologies qui puissent gérer des données RDF. Nous avons donc étudié une solution basée sur le mapping de bases relationnelles (**D2RQ**), et des solutions basées sur des bases NoSQL (**TDB, Hbase et Neo4j**).
Le but de ce projet est d'exploiter différentes sources de données, gérées au travers de plusieurs systèmes de gestion de données, afin de permettre un accès et consultation de ces derniers. Ceci, via une application permettant d'interconnecter ces sources de données, basée sur l'API de Jena.

2 Jeu de données

2.1 Description

Le jeu de données utilisé représente des données de vocabulaire de l'**INSEE** et des données **Geonames**. L'INSEE (Institut National de la Statistique et des Etudes Economiques) collecte, produit, analyse et diffuse des informations sur l'économie et la société françaises. A ce titre, il conduit des recensements et des enquêtes, il gère des bases de données et exploite aussi des sources administratives. Geonames est une base de données géographique gratuite et librement accessible sur Internet (www.geonames.org). La base regroupe plus de 8 millions de noms de lieux géographiques, et beaucoup d'informations autours de ces lieux (par exemple la population, la subdivision administrative, le code postal, la latitude, la longitude, l'altitude, etc.).

2.2 Schéma

//A finir avec une description plus précise de chaque données avec un schéma d'interconnexion

3 Analyse du système

3.1 Différents modèles de représentation des données

3.1.1 Modèle relationnel

Le modèle relationnel est le modèle de représentation le plus utilisé dans les SGBD actuellement. Il est basé sur l'algèbre relationnelle. Une BDD relationnelle est un ensemble de *tuples* groupés ensemble dans des *tables*, et un ensemble de contraintes qui, entre autre, permettent de définir des liens entre les tables. Pour pouvoir exploiter des données stockées dans une BDD relationelle en meme temps que des données stockée dans des BDD NoSQL, nous avons besoin d'une interface qui nous permettra de requêter dans un langage commun, différent donc de SQL.
Le SGBD que nous avons choisi est **MySQL**, et l'interface **D2RQ**.

3.1.2 Triple Store

Un Triple Store est une base de donnée NoSQL conçue pour stocker des triplets RDFs. La consultation des données se fait avec le langage de requête **Sparql**. La création d'un TDB store à pour effet de creer des fichiers persistant dans un dossier précisé en entrée. Les fichiers de données RDF sont donc chargés une

fois, puis l'accès aux données se fait en chargeant ce dossier dans notre application. Le seul type de donnée qu'un triple store peut stocker est le *triplet*. Contrairement au relationnel, ce modèle ne dépend donc pas d'un schéma.
Le triple store que nous avons décidé d'intégrer est **TDB**.

3.1.3 Base de données orientée graphe

Une base de données orientée graphe est une BDD NoSQL utilisant la théorie des graphes. Les données sont donc représentées sous forme de noeuds et d'arcs représentant ces noeuds. Ce modèle est pratique pour la représentation de données, car dans beaucoup de cas les données sont facilement représentables sous forme de graphe.
La base de donnée orientée graphe que nous avons décidé d'intégré est **Neo4j**

3.1.4 Base de données orientée colonnes

Une base de données orientée colonne est une BDD qui stocke les données sous forme de colonnes, contrairement aux BDD relationnelles qui les stockent par lignes (tuples).
L'intérêt est de sérialisé les colonnes les unes après les autres, alors que l'orientation "ligne" écrit chaque entité les unes après les autres.
Les lignes des tables sont partitionnées en plusieurs regions, qui sont stockées dans des region servers. Celles-ci se créent automatiquement au fûr et à mesur de l'augmentation de la taille (un seuil limite est défini). Ainsi, le contenu des region servers est indexé dans un serveur master, qui dirge les clients vers le nœud adéquat.
Enfin, ZooKeeper est un logiciel de gestion de configuration pour systèmes distribués. Celui-ci est utilisé pour définir les instances Hadoop faisant partie du cluster HBase. Le fonctionnement de Hbase est expliqué dans le schéma suivant :

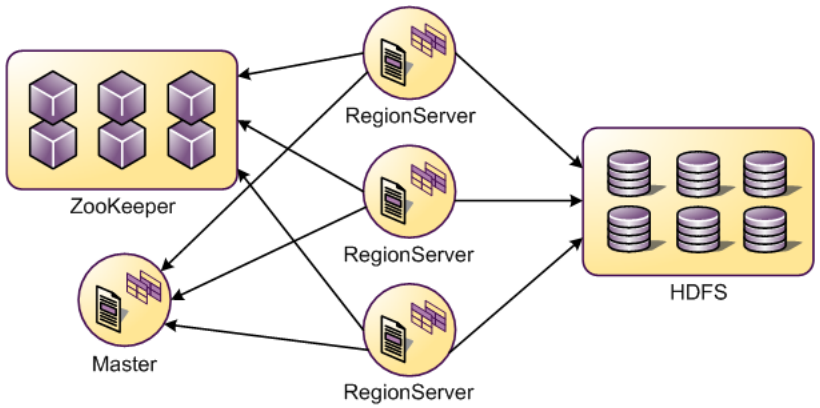

FIGURE 1 – Vue Générale du fonctionnement de Hbase

Source : Cf. Hbase dans 7.

3.2 Une API pour les gouverner tous : Jena

Jena est une API Java qui peut être utilisée pour créer et manipuler des données RDF . Jena possède des classes pour représenter des graphes, des ressources, des propriétés et des littéraux.

Ce projet est uni autour de l'API de Jena (Cf. Section 3.2) et de ses modèles.

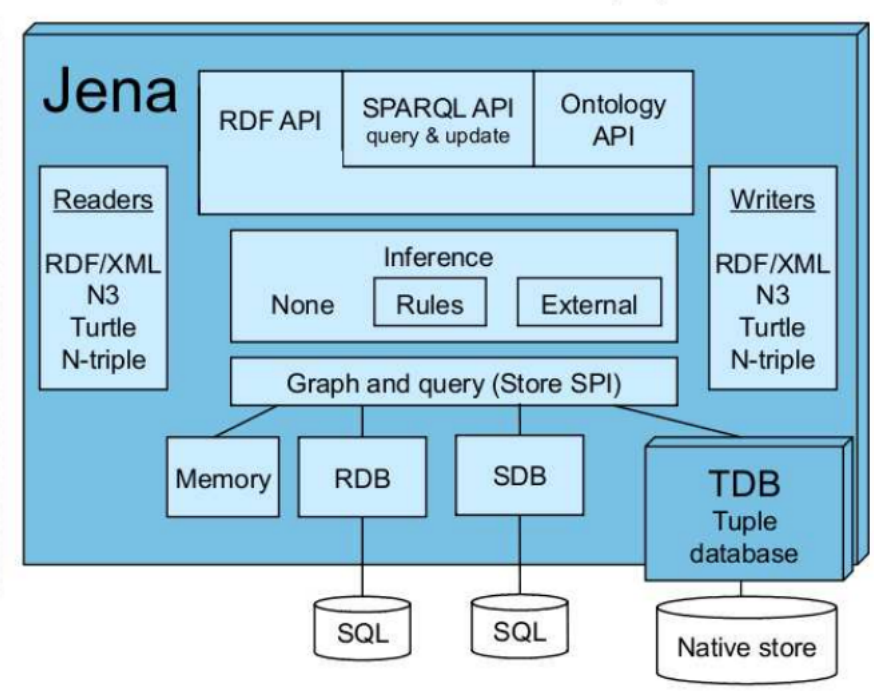

FIGURE 2 – Vue Générale de l'architecture JENA

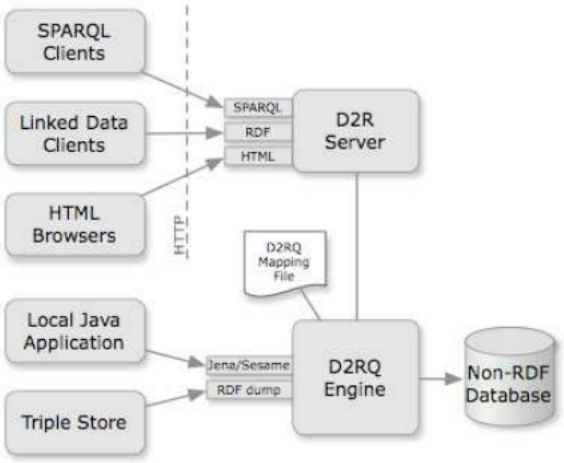

FIGURE 3 – Vue Générale de l'architecture D2RQ

3.3 Le langage de requete SPARQL

La réprésentation Model spécifique à Jena a permis l'union obtenue à partir de nos sources de données. Les requêtes SPARQL sont effectuées sur le modèle ainsi créé.
Chaque triple store définit ses propres fonctions à côté de celles de SPARQL (ARQ pour Jena). Pour gérer les requêtes SPARQL, Jena utilise sa propre librairie *ARQ*. Cette librairie définit également sa propre syntaxe, compatible notre mode de requêtage SPARQL, cela y ajoute des fonctionnalités semblables à ce qu'on peut trouver dans MySql par exemple, comme COUNT, MAX, etc, car ces fonctions ne sont pas gérées par défaut par SPARQL.

4 Application

4.1 Généralités

Notre application se base sur un modèle JENA (Cf. 3.2) construit, avec des données réparties, comme vu au dessus, au sein de différents systèmes de stockage
Le modèle JENA sert donc d'union ou interconnexion de ces données, pour qu'on puisse à la fin intérroger ce dernier via des requêtes SPARQL(Cf. 3.3), sur n'importe quel type de données sans devoir dépendre d'un type de stockage précis.

4.2 Architecture

Le but est de pouvoir respecter l'architecture ou le schéma suivant :

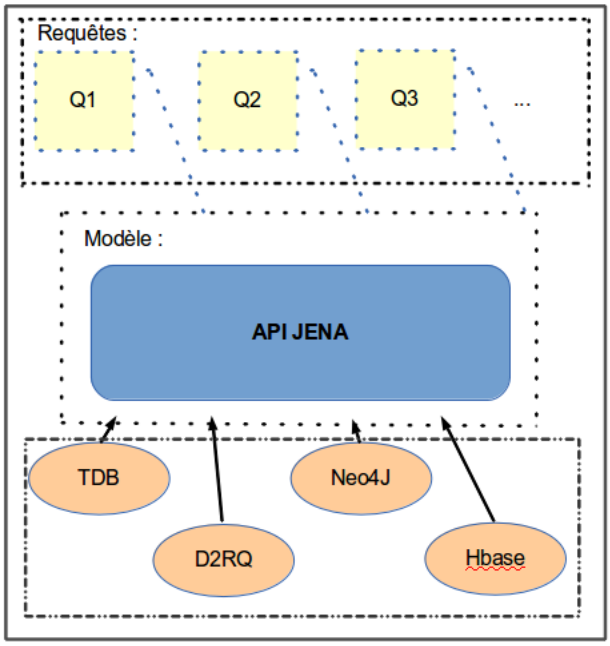

FIGURE 4 – Architecture ou schéma de fonctionnement de l'application

La génération du modèle se fait dans les classes correspondantes à nos systèmes, puisqu'une fonction pour chacun d'entre eux génère le mapping :

- Pour D2RQ : c'est la fonction //TODOOO avec une desc et screenshot de la fction
- Pour TDB : //TODO Pareil
- Pour Ne4j : la fonction //TODO construit unmodèle jena à partir d'une bdd embarquée neo4j. //Manque screenshot

Une fois le modèle JENA construit, l'union y est faite, facilitant l'interrogation de ce modèle.

5 Démonstration

5.1 Principe

//TODO

5.2 Requêtes

Donner l'ensemble des résidences de tourismes avec leurs codes postales dans le Calvados (14) :

```
float cosinus  (float x, int maxIteration)
{
   /* encore du code ici */
}
```

6 Discussion et Conclusion

Durant la réalisation de ce projet, il a été constaté qu'il serait plus intéréssant de distribuer les donéees sur des sites différents et notamment, pour Neo4j, ne pas utiliser une bdd embarquée mais l'api REST. C'est l'une des perspectives pour la complétion de ce travail dans le futur.

Mais l'une des difficultés majeures rencontrer c'est celle concernant l'installation et l'exploitation de la base de données orientée colonnes : Hbase, Cf. 3.1.4.
En effet, le choix du type de base de donnée NoSQL (en l'occurence dans ce cas là, celle orientée colonnes) et la manière de stocker les données en structurant les triples pour celle-ci, afin de la lier avec les autres via l'API JENA ne relève pas de l'impossible, mais reste très difficile, ceci est dû principalement au manque de documentations.
Malgré les difficultés rencontrées pour la mise en place de ce système d'interconnexion des sources de données, ce projet nous a avant tout permis de se familiariser et d'expérimenter l'API Jena, en complément au processus théorique et expérimental vu en Cours et TP durant le semestre.
Nous avons en fin de compte pu intégrer 3 solutions sur les 4 initiales (D2RQ, TDB et Neo4j), et nous avons pu intérroger le modèle faisant office de l'union de ces derniers.
On peut dire qu'on a retenu que l'interconnexion des données simplifie énormement l'exploitation de celle-ci, puisque le système voulant les exploiter devient plus performant ainsi.

7 Sources

— API Jena et TDB : http://jena.apache.org/

— D2RQ : http://d2rq.org/

— Neo4J : http://neo4j.org/

— Hbase :
  — http://blog.xebia.fr/2009/11/18/devoxx-jour-1-nosql-avec-hbase/
  — http://hbase.apache.org/
— SPARQL : http://www.sparql.org/

— Geonames : http://www.geonames.org/

— INSEE : http://rdf.insee.fr/

— Données sur le tourisme : http://data.gouv.fr/fr/dataset/residences-de-tourisme-classees-en-france/