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 un système basé sur le triple store (**TDB**), mais aussi une solution basée sur le mapping de bases relationnelles (**D2RQ**), ou bien encore une solution basée sur des bases NoSQL, avec des bases orientées "graphes" (**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

/**TODOO à finir**

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 API de 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 4.5) et de ses modèles.

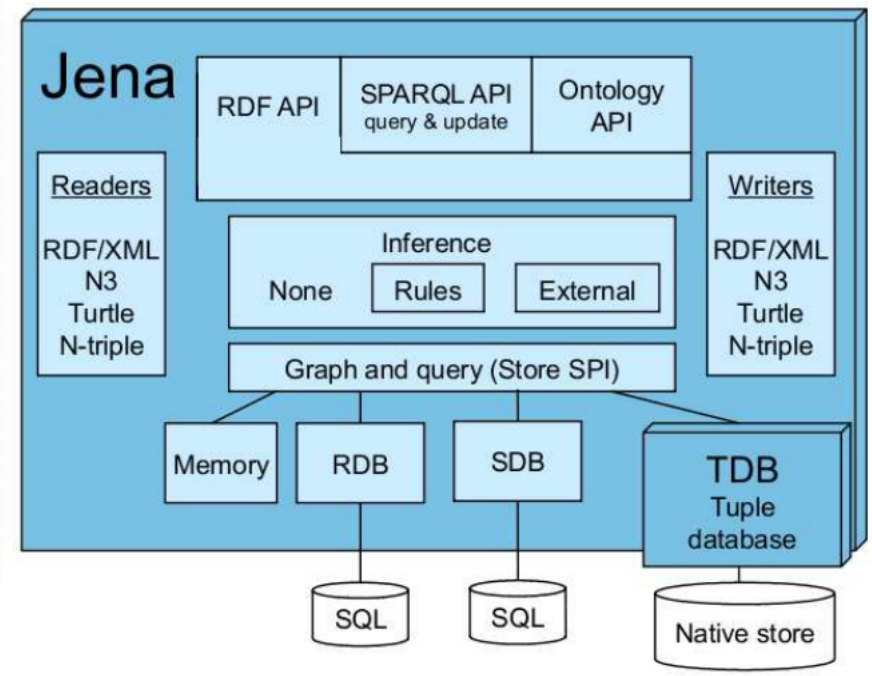

FIGURE 1 – Vue Générale de l'architecture JENA

— **TDB :** c'est un triple store avec une solution souple sans schéma prédéfini.
— **D2RQ :** ce système est adapté au stockage des données de fichiers csv par exemple, dans une relation d'un schéma relationnel. Avec Direct Mapping, on peut facilement traduire un modèle relationnel en RDF.

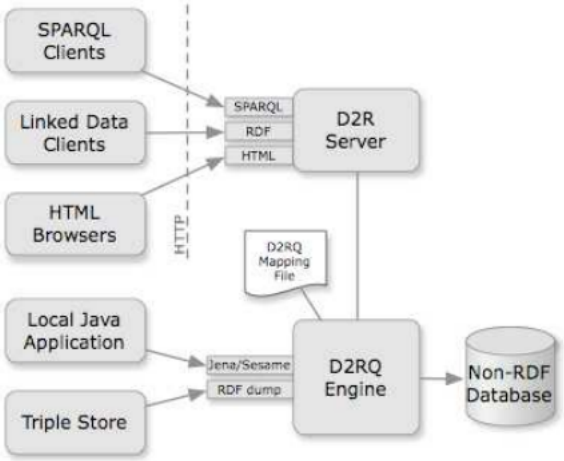

FIGURE 2 – Vue Générale de l'architecture D2RQ

— **Neo4j :** est une base de données graphique, et il est un bon modèle de stockage de données pour une ontologie. Neo4j est robuste (entièrement ACID).
— **HBase :** Jena-HBase fournit un stockage évolutif et une solution d'interrogation qui prend en charge toutes les fonctionnalités de la spécification RDF.

3.2 Intérrogation SPARQL

La réprésentation Model spécifique à Jena a permi l'union obtenus 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 Réalisation

4.1 Triple store : TDB

//TODO

4.2 D2RQ

//TODO

4.3 Neo4j

//TODO

4.4 HBase

//TODO

4.5 Modèle JENA : Intégration

//TODO

5 Démonstrateur

5.1 Principe

//TODO

5.2 Requêtes

//TODO

6 Discussion et Conclusion

//TODO

7 Sources

//TODO