MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

Traitement des connaissances du web sémantique basé sur l'utilisation du modèle MapReduce

Etudiante : « **NAHET Malika** »

Encadrante : « BENHAMED Siham »

Année Universitaire 2016/2017

Résumé

La représentation sémantique des données dans le web à travers le RDF (Resource Description Framework) est en forte augmentation ces dernières années, ce qui a permis aux machines l'exploitation de la sémantique d'une manière formelle. Cependant, le traitement de ces ensembles de données RDF par les solutions classiques des systèmes centralisés engendre différents problèmes au niveau de l'accès à la donnée et sa récupération. Pour ce faire, l'objectif de notre travail est de proposer un mécanisme de traitement de grandes quantités de données sémantiques dans des systèmes distribués, en utilisant le modèle de programmation parallèle MapReduce pour l'interrogation, et le stockage de ces données.

Mots-clés : web sémantique, RDF, SPARQL, Big Data, traitement parallèle, MapReduce.

Sommaire

Sommaire

Liste des abréviations

API: Application Programming Interface.

BGP: Basic Graph Pattern.

CDH: (Cloudera Distribution Including Apache Hadoop

CERN : Centre Européen de Recherche Nucléaire.

DAML: DARPA Agent Markup Language.

DOM: Document Object Model.

Jung: Java Universal Network Graph.

HDFS (Hadoop Distributed File System.

HTML: Hyper Text Markup Language.

NoSQL: Not Only SQL.

OIL: Ontology Inference Layer.

OWL : Web Ontology Language.

P2P: Peer to Peer.

RDF: Resource Description Framework.

RDFS: RDF Schéma.

RDQL: RDF Data Query.

RSS: Really Simple Syndication.

SAX: Simple API for XML.

SGBD: Système de Gestion de Base de Données.

SPARQL: Simple Protocol and Rdf Query Language.

SQL: Structured Query Language.

TP: Triple Pattern.

Turtle: Terse RDF Triple Language.

URI: Uniform Resource Identifier.

URL: Uniform Resource Locator.

W3C: World Wide Web Consortium.

XML: Extensible Markup Language.

Listes des figures

Listes des figures

Liste des tableaux

Introduction générale

Le web est devenu un immense réservoir de ressource d'information et de connaissances à travers un nombre massif de pages web. Ces derniers, contiennent un ensemble important de différents types d'informations qui constitue un ensemble de relations logiques entre ces informations afin qu'elles soient compréhensibles par la machine. Ce nouveau concept est le principe du web sémantique.

Pendant ces dernières décennies, un certain nombre de techniques et de langages ont été mise au point, pour la représentation formelle des connaissances sur le web sémantique tout en préservant leurs sémantiques tels que RDF (Ressource Description Framework) [1], RDFs (Ressource Description Framework schema) [1], et les ontologies [1]. Ces standards sont basés sur l'XML (Extensible Markup Language) [1].

Afin de traiter les données RDF, il existe plusieurs langages d'intégration de données sémantiques dont SPARQL que nous avons choisi pour traiter nos données sémantiques. Le langage SPARQL (Protocol And RDF Query Language) [2] est un langage d'interrogation spécifique aux données RDF. En théorie cela signifie qu'on pourrait accéder à toutes les ressources du web sémantique via ce standard mais en pratique ce n'est pas toujours évident car parfois l'obtention des résultats d'une requête peut atteindre plusieurs heures, voir des jours, puisque la taille des bases de données RDF est souvent d'une taille massive ce qui rend l'accès à la donnée long.

Pour cela, il nous paraît que la solution la plus concrète serait de partitionner la base de données en plusieurs parties ainsi on peut exécuter la requête sur chaque partie pour regrouper ensuite les résultats obtenus peuvent être regroupées pour former la solution finale.

D'une autre part, puisque le web sémantique contient une grande masse de données, une représentation graphique devienne nécessaire, vu la nature de la représentation du framework RDF qui est constitué d'un ensemble de triplets. Pour cela, le RDF est représenté par graphe composé de nœuds et d'arcs dirigés, cependant leurs stockages sont très lourds avec les bases de données traditionnelles.

Pour ce faire, un mode de stockage particulier est adapté pour archiver toute sorte de connaissance de différent source appelé BigData [3]. Ce type de base permet de résoudre le problème du stockage de la grande masse de données qui est sémantique dans notre cas.

Cependant, un autre défi est apparu pour les développeurs qui commencent à se poser la question suivante: comment traiter, gérer et manipuler cette masse d'informations et comment l'exploiter de manière fiable ? C'est-à-dire comment traiter ces connaissances volumineuses est énormément difficile qui prend beaucoup de temps.

Lorsque la taille des données à traiter est grande, il devient nécessaire d'utiliser les outils de la programmation parallèle qui permet de traiter les gros calculs en un temps donné par duplication des éléments de calcul.

Le principe de la programmation parallèle est de réaliser un équilibrage dans la charge de travail, lorsqu'il y a de très grosses demandes. Il est basé sur un ensemble de machines reliées entre eux via

réseau. Parmi les systèmes de programmation parallèles on trouve Hadoop [4] qui fonctionne selon le modèle MapReduce [4].

Ce type de programmation semble parfait pour le traitement des données sémantiques stockées dans une base de données de type BigData. Dans notre travail, nous proposons un modèle de traitement de données sémantique qui utilise MapReduce afin d'améliorer le temps d'exécution des requêtes.

MapReduce est un modèle de programmation popularisé par Google. Il est principalement utilisé pour la manipulation et le traitement d'un nombre important de données au sein d'un cluster de nœuds. L'intérêt de ce modèle de programmation est de simplifier le travail du développeur, qui n'a plus à se soucier du travail de parallélisation et de la distribution de la charge de travail. Ainsi, MapReduce est un modèle de programmation qui permet au développeur de ne s'intéresser qu'à la partie algorithmique. Il transmet alors son programme MapReduce développé dans un langage de programmation au Framework Hadoop pour l'exécution.

Afin d'éclaircir le travail effectué, ce mémoire est articulé autour des quatre chapitres:

Chapitre 1 : le web sémantique et les SGBD appropriés.

Le premier chapitre donne un aperçu général sur le web sémantique en expliquant ses différents formats de représentation des données sémantiques et des langages d'interrogation associés. Ainsi, nous parlerons dans ce chapitre des bases de données adaptées pour le stockage des données du web sémantique.

Chapitre 2: la programmation parallèle et MapReduce.

Dans le deuxième chapitre nous allons définir les différents systèmes existants, par la suite on va discuter sur le principe de la programmation parallèle, tout en expliquons le fonctionnement du modèle de programmation parallèle MapReduce.

Chapitre 3 : Conception d'un modèle de traitement de données RDF par MapReduce

Le troisième chapitre présente notre modèle de traitement de données sémantique basé sur MapReduce qui améliore le traitement des données sémantique en termes de temps d'exécution.

Chapitre 4 : Implémentation d'un système de traitement de données RDF par MapReduce.

Ce chapitre présente la mise en œuvre du modèle proposé dans le chapitre précédant.

Enfin, nous concluons notre étude en donnant un résumé de notre travail, ainsi que, des perspectives futures à nos travaux.

I. Introduction :

Le web contient une masse de connaissances assez importante ce qui a permis aux utilisateurs d'accéder à plusieurs informations appartenant à différents domaines. Cependant, la recherche d'une information particulière à partir d'une connaissance prend un temps très grand puisque c'est l'utilisateur qui doit filtrer les résultats obtenus pour garder l'ensemble des informations adéquates. Pour pallier ce problème, une solution a été proposée qui consiste à étendre le web actuel par le principe du Link, qui permet de relier les données des différentes connaissances entre eux pour obtenir de nouvelles connaissances. Dans cette nouvelle version du web appelé web sémantique, les connaissances sont bien structurées et formulées grâce à l'adoption de différents formats de représentation de données pour que ces dernières deviennent bien compréhensibles par les machines et par conséquent, elles seront bien traitées.

II. Historique du web :

Le web est inventé en 1989 par Tim Berners-Lee, un ingénieur de CERN (Centre Européen de Recherche Nucléaire) [5]. Au début, l'objectif était de reproduire à l'échelle du grand public ce qui fait la force du CERN : la collaboration internationale. Et bien sûr en utilisant un moyen de communication simple, efficace et accessible.

Le web, est donc une application conviviale de consultation à distance de pages d'informations multimédias**.** A partir de l'été 1991, cette invention a été offerte au monde entier.

Le web est passé par trois phases importantes, au début c'était le web 1.0, ensuite le web 2.0 en aboutant au web 3.0.

II.1. Le web 1.0 :

Cette version (Figure 1) est appelée aussi web traditionnel, elle a été inventé en 1991. Le web 1.0 est avant tout un web statique, centré sur la distribution d'informations, considérant que l'internet était un espace où les rares personnes qui y avaient accès mettaient en ligne des documents statiques (du HyperText Markup Language HTML). Il sert alors à donner accès au contenu proposé par un producteur et affiché sur un site Internet consulté par des internautes, il a un fonctionnement très linéaire : C'est un web passif [5] [6] [7].

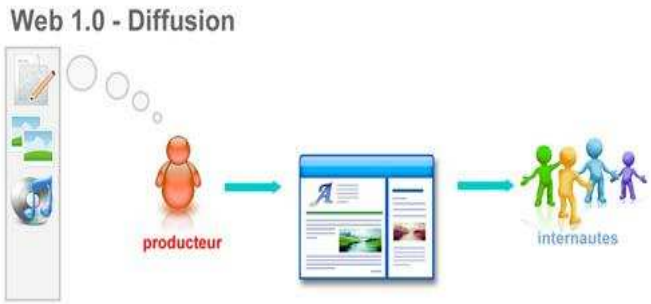


Figure 1 : L'architecture du web 1.0 [5].

Le web 1.0 reste en progrès jusqu'à l'année 1999, il bien évidemment qu'il n'est pas resté totalement statique très longtemps… [6].

II.2. Le web 2.0 :

Par opposition, le web 1.0 qui était essentiellement un web consultatif, le web 2.0 est devenu plus collaboratif, très social, son apparition était en 2000 jusqu'à 2009, dont les internautes ne sont plus seulement consommateurs passifs, mais contribuent activement d'une part à la création de contenus, mais aussi à la validation de leur valeur [5].

En effet il permit de facilement rédiger des documentations, rapports et autres. Il a aussi été utilisé à très grande échelle comme par exemple le très célèbre Wikipédia.

Le web 2.0 (Figure 2) adopte l'aspect de la glocalisation. La glocalisation permet donc une adaptabilité précise des ressources web pour l'utilisateur. L'internaute peut choisir de se syndiquer à un site qui diffuse de l'information via un flux de données dit RSS (Really Simple Syndication) ce qui lui permet d'obtenir les informations en temps réel, l'utilisation de la technologie XML (Extensible Markup Language) permet de transmettre le contenu informationnel dont la « couche présentation » sera gérée par le lecteur de flux RSS [8].

Figure 2 : L'architecture du web 2.0 [5].

A cause de la masse d'informations construites il devient le problème de concevoir et réaliser des systèmes permettant de filtrer les informations et de les délivrer de façon « intelligente ».

Pour ce faire, une troisième génération du web appelé web 3.0 a été proposée. Cette nouvelle version est une extension de la précédente.

II.3. Web 3.0 :

Le web 3.0, aussi nommé web sémantique a eu un effet remarquable dans le cycle de la recherche et le traitement des connaissances puisqu'il vise à organiser la masse d'informations disponibles en fonction du contexte et des besoins de chaque utilisateur, en tenant compte de sa localisation, de ses préférences, etc. C'est un web qui tente de donner un sens aux données. Il sert à intégrer des contenus dynamiques pour la compréhension et l'exploitation des données.

Les premières réflexions sur la mise en place du Web sémantique (Figure 3) ont été entamées par le W3C (World Wide Web Consortium) nouvellement créé, ce qui a abouti à la publication d'un premier " draft "de recommandations sur le web sémantique en octobre 1997 [9].

Le site Internet fournissait alors des applications en ligne qui savent analyser automatiquement les contenus écrits et picturaux, qui peuvent les interpréter, les comprendre, les classer, et les rediffuser vers un nouvel internaute public.

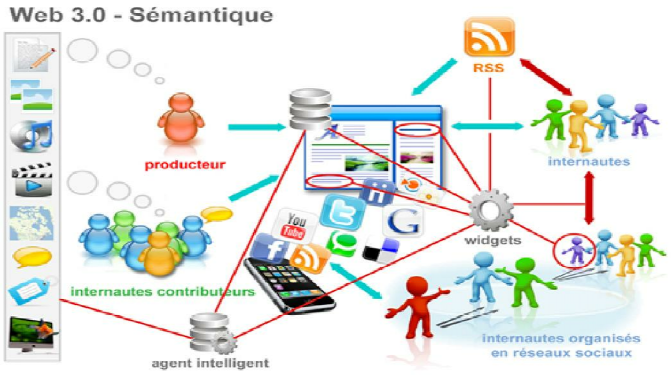


Figure 3 : L'architecture du web 3.0 [5].

Dans le schéma ci-dessous (figure 4), nous proposons un aperçu sur les trois versions du web

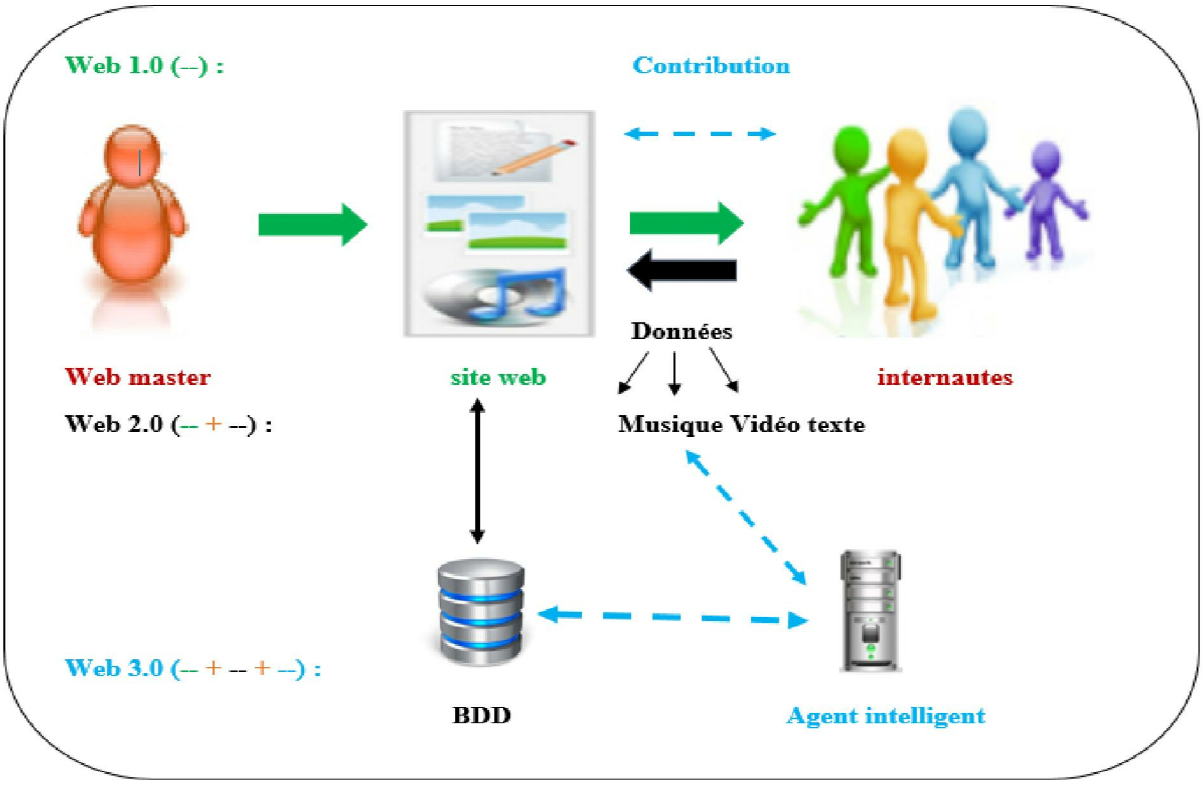


Figure 4 : Schéma global du Web.

III. Le web sémantique :

D'après Tim Berners-Lee, James Handler, et Ora Lassila, « le web sémantique n'est pas un web séparé, mais un prolongement de l'actuel, dans lequel l'information reçoit une signification bien définie, mieux permettre aux ordinateurs et aux gens de travailler en coopération » [10].

Le Web sémantique, plus techniquement appelé « le Web de données » permet aux machines de comprendre la sémantique, la signification de l'information sur le web. Il étend le réseau des hyperliens entre des pages web classiques par un réseau de lien entre données structurées permettant ainsi à l'ordinateur de faire des recherches intelligentes à l'aide des agents automatisés après avoir accédé aux différentes sources de données contenues sur le Web. Ce qui rend le langage humain compréhensible par la machine [9].

Autrement dit le web sémantique désigne la construction des relations logiques entre les données qui pourrait directement ou indirect être traité par des machines pour aider des utilisateurs à créer des connaissances. Tel que les producteurs des connaissances pour aider les machines à reconnaitre et à mettre en relation ces données afin d'améliorer les recherches d'informations et permettre d'agréger des données avec des complémentaires pour aider les machines.

IV. Les Techniques du web sémantique :

Le web sémantique, contient avant tout un ensemble de langages et d'outils permettant de représenter formellement de la connaissance afin que la machine ait accès au sens des documents existant dans le web. On va parler par la suite des déférents formats de représentation des données sémantiques et des langages d'interrogation associés.

IV.1. Les formats de représentation :

IV.1.1. RDF (Resource Description Framework) :

Le premier standard proposé pour la représentation des données sémantiques est le RDF [11]. Les objectifs initiaux de RDF étaient la représentation avec une bonne exploitation des métadonnées. Il permet de voir le web comme un ensemble de ressources reliées par les liens étiquetés « sémantiquement ». L'arrivée de XML, en 1998, a donné un cadre à la structuration des connaissances, permettant de décrire des métadonnées et facilitant leur traitement. C'est dans cet esprit qu'a été créé en 1999 RDF, cette sorte de langage permet de définir des structures ou moyens des triplets.

En effet, la structure fondamentale de toute expression en RDF est une collection de triplets, chacun est composé d'un sujet, un prédicat et un objet où certaines ressources sont identifiées par des URI (Uniform Resource Identifier) dont l'exemple le plus connu est celui des URL (Uniform Resource Locator), qui constituent les « adresses » des pages du web. Les ressources peuvent être identifiées ou rester anonymes, elles peuvent également être typées en utilisant la relation *rdf:type*. Les objets peuvent être des littéraux (comme une chaîne de caractères ou un nombre entier).

Dans la figure ci-dessous, nous montrons un exemple simple qui définit une structure de triplet RDF (statment).

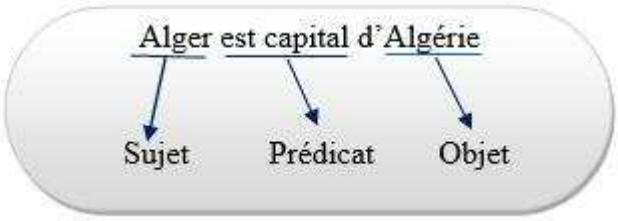


Figure 5 : Exemple d'un triplet (statement) RDF.

Par la suite on va détailler les trois modes de représentations basés sur un langage XML, un graphe et sur un langage Turtle.

- **XML (Extensible Markup Language) :**

La représentation RDF est basée sur le langage XML qui est utilisé pour encoder les langages du web sémantique. Il peut être vu comme la couche de transport syntaxique.

La représentation physique d'un document XML est sous la forme d'un fichier texte structuré en éléments où chaque élément doit commencer par une balise ouvrante et se termine par une balise fermante. En en-tête du document doit figurer un « prologue », une déclaration qui identifie le document comme un document XML.

- **Exemple :**

<? XML version = "1.1" encoding= " ISO-8859-1" ?>

<Cours>

<Intervenant> abc </ Intervenant>

<Chapitre> formation XML

<Para> un paragraphe </ Para >

< Para> un autre paragraphe </ Para >

</Chapitre>

</Cours>

Tel que les nœuds de cet exemple sont : cours, Intervenant, chapitre et para. Le contenu du cours c'est un ensemble de nœuds et le contenue du para est un texte.

- **Graphe :**

Un ensemble de triplets est appelé un graphe RDF. Ceci peut être illustré par un diagramme composé de nœuds et d'arcs dirigés, dans lequel chaque triplet est représenté par un lien nœud-arc-nœud (d'où le terme de "graphe"), voici figure comme un exemple :

Figure 6 : Exemple d'une représentation graphique du RDF.

- **Turtle (Terse RDF Triple Language):**

Turtle est, depuis février 2014, un standard approuvé par le W3C. Cette notation pratique pour les triplets est devenue le standard de la représentation RDF.

Voici un petit exemple de Turtle :

@prefix dbpedia-owl: <http://dbpedia.org/ontology/>.

@prefix dbpprop: <http://dbpedia.org/property/>.

@prefix dbpedia: <http://dbpedia.org/resource/>.

@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

dbpedia:Eiffel_Tower dbpedia-owl:location dbpedia:Paris

dbpedia:Eiffel_Tower dbpprop:startDate "1887"^^xsd:integer

dbpedia:Eiffel_Tower dbpprop:name "Tour Eiffel"@fr

dbpedia:Eiffel_Tower dbpprop:name "Eiffel Tower"@en

Dans ce qui suit, on présente l'explication de l'exemple :

- Il y a trois types de *terme*s :
  - ✓ URI écrit entre < et >.
  - ✓ Littéral entre guillemets ou accolades, possiblement suivie d'une définition de type Indiquée par ^^ suivi d'un nom de type souvent de la forme par exemple ici on a xsd : integer.
  - ✓ Nœud vide indiqué par _ : nodeId ou un groupe entre crochets.

Un préfixe est défini par @prefix p : URI, par exemple on a @prefix dbpprop : <http://dbpedia.org/property/>.

La nécessité de déclarer l'espace de nom employé nous amène au Schéma RDF (RDFS), Présenté dans la section suivante.

IV.1.2. RDFS (RDF Schéma) :

RDFS est un langage extensible de représentation des connaissances. Il appartient à la famille des langages du web sémantique publiés par le W3C. RDFS fournit des éléments de base pour la définition d'ontologies ou de vocabulaires destinés à structurer des ressources RDF.

La première version de RDFS a été proposée en mars 1999, et la recommandation finale publiée par le W3C en février 2004. Les composants principaux de RDFS sont intégrés dans un langage d'ontologie plus expressif, OWL (Web Ontology Language).

RDFS ajoute à RDF la possibilité de définir des hiérarchies de classes et de propriétés dont l'applicabilité et le domaine de valeurs, ces dernières peuvent être contraintes à l'aide des attributs rdfs:domain et rdfs:range.

RDFS a pour but d'étendre le langage en décrivant plus précisément les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme permettant de spécifier les classes dont les ressources seront des instances, comme les propriétés. RDFS s'écrit toujours à l'aide de triplets RDF, en définissant la sémantique de nouveaux mots-clés comme dans l'exemple ci-dessous.

```
<?xml version = "1.0 " ?>
<rdf : RDF xmlns : rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#
        Xmlns : rdfs = " http://www.w3.org/2000/01/-rdf-shema#">
<rdfs : class rdf: ID = "personne">
<rdfs : comment> la classe personne </rdfs : comment>
 </rdfs : class>
<rdfs : class rdf : ID ="véhicule">
<rdfs : comment> la classe véhicule </ rdfs : comment>
</rdfs : class>
<rdfs : class rdf : ID ="voiture">
<rdfs : comment> la classe voiture </ rdfs : comment>
<rdfs : subClassOf rdf : ressource ="# véhicule"/>
</rdfs : class>
<rdf : Property rdf : ID ="conducteur">
<rdfs : domain rdf : ressource ="# personne"/>
<rdfs : range rdf : ressource ="# voiture"/>
</rdf : Property>
</rdf : RDF>
```

Tel que : -rdfs : subClassOf permet de définir des hiérarchies de classes.

- rdf : Property permet de donner un type ou une classe ou sujet et à l'objet pour cela

RDFS ajoute les notions : domain et range, tel que rdfs : domain définit la classe des sujets liés aux propriétés et rdfs : range définit le type des données des valeurs de la propriété. Ce besoin de spécifier davantage les classes est à l'origine du langage dédié aux définitions de classes : OWL.

Le RDF est un langage dédié à l'expression d'assertions sur les relations entre objets, pour définir les propriétés des classes dont ces objets sont des instances. Cependant, l'extension à RDFS ne fournit que des mécanismes primitifs pour spécifier ces classes. Pour résumer,

RDF4 comme un langage relationnel de base et RDFS offre des primitives de représentation de structures ou primitives ontologiques.

IV.1.3. Ontologie :

Une ontologie est une description formelle et explicite des concepts dans un domaine particulier. Selon Gruber, une ontologie doit être :

- **Formelle** se réfère au fait que la spécification doit être lisible par une machine.
- **Explicite** signifie que les types des concepts et les contraintes sur leur utilisation sont explicitement définis.
- **Conceptuelle** se réfère à un modèle abstrait d'un certain phénomène du monde reposant sur l'identification des concepts pertinents de ce phénomène, cela signifie qu'une ontologie capture la connaissance consensuelle, qui n'est pas propre à un individu mais validée par un groupe.

En plus, des outils de comparaison des propriétés et des classes telles que : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc. Il y a le langage OWL (Web Ontology Language) qui est dédié aux définitions de classes et de types de propriétés pour la définition d'ontologies.

L'OWL est inspiré des logiques de descriptions (successeur de DAML+OIL), il fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les propriétés des classes définies [12]. Ce langage a été fractionné en trois langages distincts :

- OWL LITE ne contient qu'un sous-ensemble réduit des constructeurs disponibles, mais son utilisation assure que la comparaison de types pourra être calculée (un problème de NP, donc « simple » en représentation de connaissances);

- OWL DL contient l'ensemble des constructeurs, mais avec des contraintes particulières sur leur utilisation qui assurent la décidabilité de la comparaison de types. Par contre, la grande complexité de ce langage (un de ses fragments est P-SPACE-complet) semble rendre nécessaire une approche heuristique;

- OWL FULL, sans aucune contrainte, pour lequel le problème de comparaison de types est vraisemblablement indécidable.

Par exemple, pour décrire les concepts entrant en jeu dans la conception de cartes électroniques, on pourrait définir l'ontologie (simplifiée ici) suivante :

- Une carte électronique est un ensemble de composants ;
- Un composant peut être soit un condensateur, soit une résistance, soit une puce ;
- Une puce peut être soit une unité de mémoire, soit une unité de calcul.

IV.2. Les langages d'interrogation des web sémantique :

L'interrogation des bases de connaissances est une exigence centrale du web sémantique. De plus en plus on est forcé à reconnaître l'importance de fournir un accès simple à ces dépôts de connaissances. Pour cela, il existe des langages d'interrogation utilisés pour accéder aux données d'une base de données et de les obtenir en vérifiant certaines conditions. Ces langages doivent répondre aux attentes de l'utilisateur afin de fournir les détails de l'exécution, parmi lesquels le RDQL, SeRQL, RQL et SPARQL [9]. Ce dernier, est le plus important dans notre travail. Nous avons opté pour SPARQL parce qu'il permet d'interroger n'importe quel composant d'un triplet qui a la forme Sujet-Prédicat-Objet.

IV.2.1. SPARQL « Simple Protocol and Rdf Query Language »:

SPARQL est considéré comme une amélioration de RDQL (RDF Data Query), il permit de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponibles à travers le net.

Une requête SPARQL se compose de plusieurs BGP (Basic Graph Pattern), ce dernier se compose d'un ensemble de morceaux de requête élémentaire, appelés TP (Triple Pattern).

Un TP s'exprime sous la forme d'un triplet RDF, qui contient une ou plusieurs variables de type sujet, prédicat, ou objet. Une variable est caractérisée par le fait qu'elle commence par un point d'interrogation dans le TP.

Dans l'exemple ci-dessous, on présente une requête qui va nous retourner le nom du capital d'Algérie :

Donnée :

```
<Http://exemple.org/pays/Algérie><http://pur1.org/dc/elements/1.1/capitale>
« Alger »
```

Interrogation :

```
SELECT  ?capitale
WHERE
{
<Http://exemple.org/pays/Algérie><http://pur1.org/dc/elements/1.1/capitale>
 ?capitale
}
```

recherche plutôt que sur la technologie de base de données ou le format sur lesquels repose le stockage des données.

SPARQL cible donc l'interrogation de meta-données RDF, structure de base du web sémantique.

Exemple de la syntaxe en triplets simplifiée avec des points d'interrogation pour marquer les variables :x rdf : type ex : Personne

Langage de patterns à matcher : select ? sujet ? propriete ? valeur where

{? sujet ? propriete ? valeur}

Le pattern est par défaut une conjonction de triplets : {? x rdf : type ex : Personne ? x ex : nom ? nom}

✓ **Avantages du SPARQL :**

Grace SPARQL, nous n'avons pas besoin de connaître a priori la structure et le contenu des données pour pouvoir les interroger. En effet, SPARQL permet d'interroger n'importe quel composant d'un triplet qui a la forme Sujet-Prédicat-Objet [9].

V. Le stockage des données sémantiques :

Vu que la masse de données a augmenté, il est nécessaire de le stocker l'ensemble des données web dans des bases très grandes pour faciliter leurs accès et leurs traitements par la suite. Pour ce faire, on va étudier deux types de bases de stockage : Bigdata et NoSQL.

V.1. Bigdata :

Le terme Bigdata signifie une masse de données, ces données sont produites en permanence avec une vitesse croissante, elles proviennent de tous les secteurs (les sites web, les réseaux sociaux, mes mails, les conversations téléphoniques, les GPS, et les téléviseurs connectés), vus l'émergence du web et des nouvelles technologies de l'information et de la communication. Cette masse de données est utilisée pour créer des applications à but analytique, c'est-à-dire qui traitent des données pour en tirer du sens ainsi pour faciliter la vie des utilisateurs [3].

Le Bigdata rassemble tous les types de base de données tels que les données RDF qui sont volumineuses, ce qui nécessite un mode de stockage particulier. Pour ce nouveau mode de stockage, il faut respecter la règle des 3v :

- Volume : il faut stocker énormément d'information.
- Variété : il faut stocker beaucoup de données de toutes sortes.
- Vélocité : il faut pouvoir avoir accédé rapidement à toutes ces données.

Une représentation d'un schéma Bigdata 3V est présente dans la figure ci-dessous :

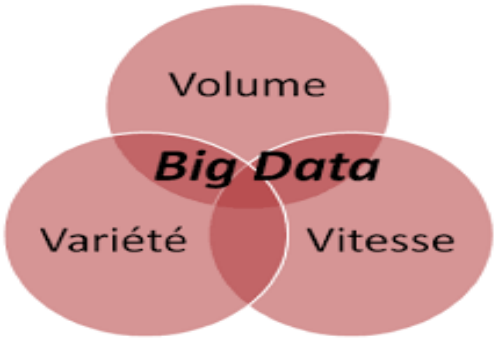


Figure 7: Schéma Bigdata 3V.

V.2.Le NoSQL:

Le terme « NoSQL » signifie « Not Only SQL » c'est-à-dire pas seulement SQL de plus il désigne l'ensemble des bases de données qui s'opposent à la notion relationnelle des SGBDR. Le NoSQL ne remplace pas les bases de données relationnelles mais il les complète en termes de performance, en effet avec le BigData, il est primordial de gérer cette masse de données qui augmente de jour en jour.

V.2.1. Types de base de données NoSQL :

Il existe quatre catégories de base de données NoSQL :

- **Base de données clé-valeur :** Dans ce type de base de données, les tables sont partitionnées. la partition est choisie en fonction d'un hach de la clé qui donnera l'instance qui se chargera du stockage (figure 6). Donc la validité de stockage dépond de l'algorithme de la fonction de hachage qui doit garantir la répartition la plus égale possible entre les partitions.

Figure 8 : Représentation d'une BDD de type clé-valeur.

- **Base de données orientée colonnes :** Dans une base de données relationnelle les colonnes sont statiques par contre dans une base de données orientée colonnes sont dynamiques (figure 7), il est alors possible d'ajouter des colonnes dynamiquement sachant qu'il y'a pas de cout de stockage pour les valeurs nulles.

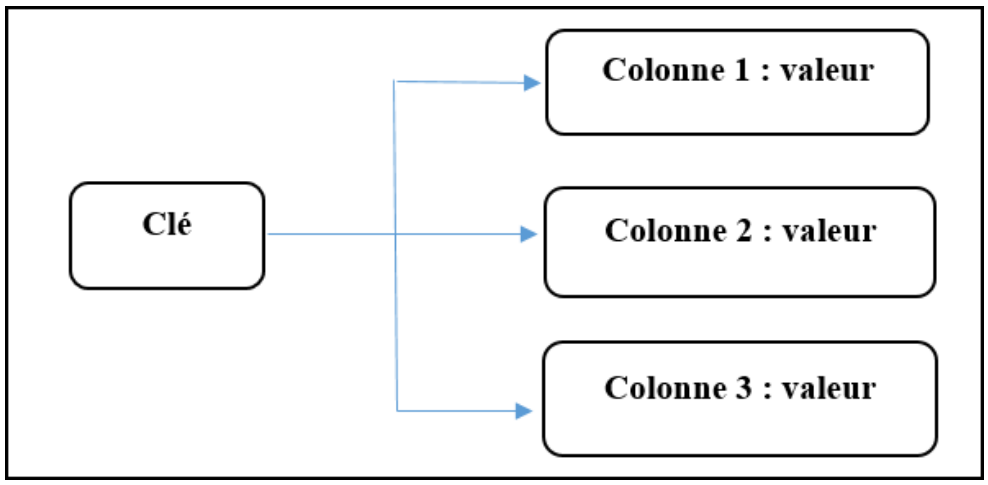


Figure 9 : Représentation d'une BDD de type orientée colonne.

- **Base de données orientée document :** Elles représentent une évolution des bases de données clé-valeur, où chaque clé est associée à un document dont la structure est libre (figure 8).

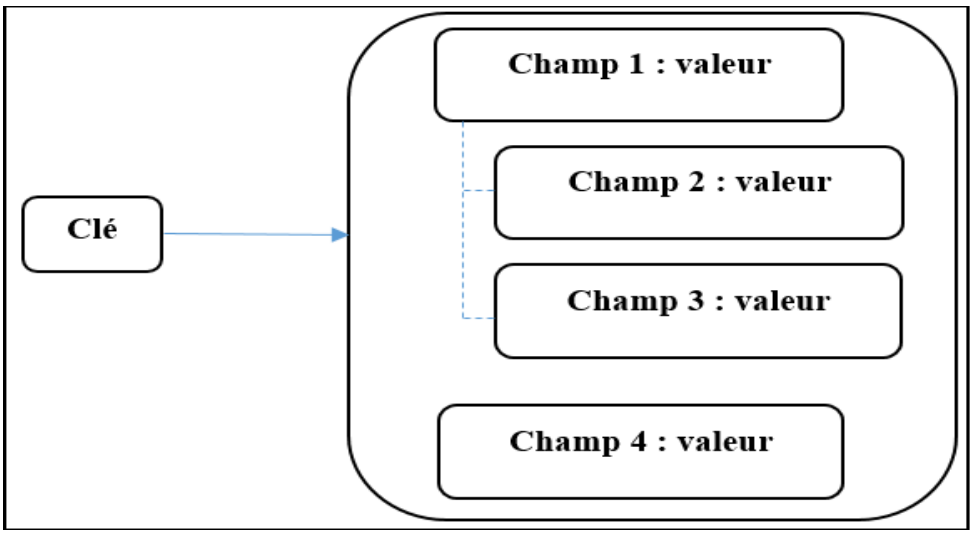


Figure 10: Représentation d'une BDD de type orientée document.

- **Base de données orientée graphe :** Les bases de données orientées graphe ont pour but de pallier les problèmes impossibles à résoudre avec les bases de données relationnelles, par exemple les réseaux sociaux sont un cas typique d'utilisation de la base de données orientée graphe (figure 9).

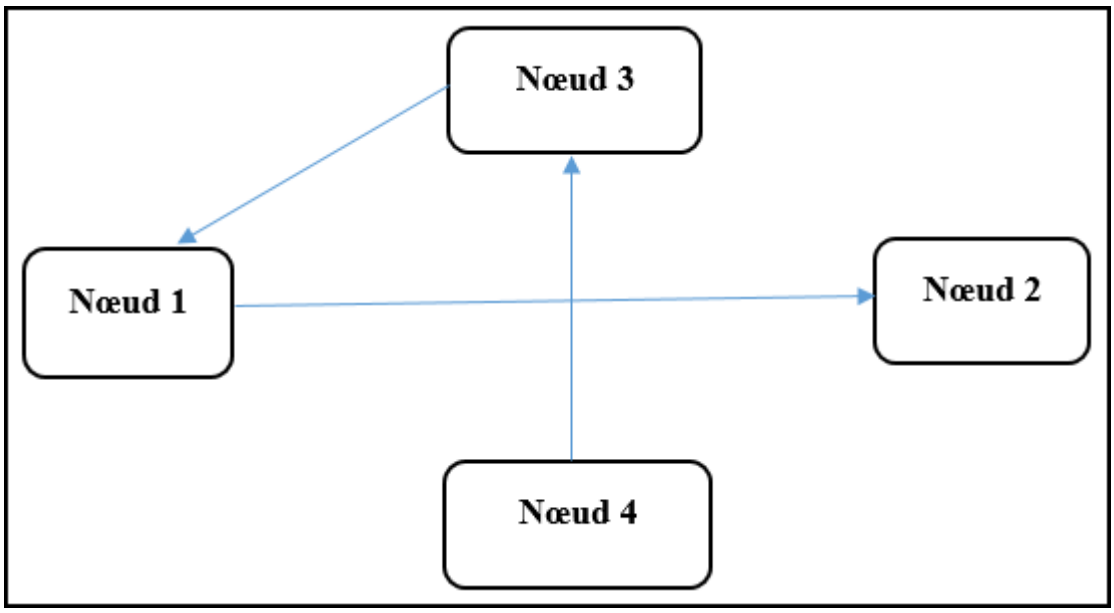


Figure 11 : Représentation d'une BDD de type orientée graphe.

VI. Conclusion :

Le web contient une très grande quantité de connaissances qui nécessite un traitement suivant les besoins des utilisateurs. Cependant, la masse de données et de connaissances rend leurs stockages très lourds ce qui requière une représentation à l'aide de différents formats plus simple et compréhensible par les machines afin de bien gérer toutes sortes de traitements.

L'un des majeurs standards proposés pour la représentation des données sémantiques est la représentation graphique. Cette dernière, permet de rendre l'accès aux données stockées dans des bases de données telles que BigData simple et rapide à atteindre par la machine.

Cependant les machines ont rencontré des difficultés, puisqu'ils sont face à traiter les limites des performances et de la taille des problèmes traités par un système centralisé suivant le principe d'une programmation ordinaire. A cet effet, il devient alors nécessaire de dupliquer les éléments de calcul pour obtenir un flux d'opérations ou de traitements afin de le lancer simultanément et en parallèle sur l'ensemble des éléments de calcul.

Ce type de programmation est dénoté par le terme calcul parallèle. Il apporte une solution pour le traitement pour ce genre de base de données. Il existe plusieurs types dont le MapReduce. Nous allons présenter le principe de ce type de programmation, et plus précisément nous allons ainsi expliquer le modèle MapReduce dans le chapitre suivant.

I. Introduction :

La densité de connaissances qui se trouve dans web conduit à une difficulté de traitement d'accès et de traitement.

Pour cela, auparavant, les systèmes centralisés ont été utilisés pour gérer cette quantité de données. Ces systèmes ont trouvé des problèmes à cause de la dépendance des données, et puissance de calcul insuffisante, ainsi que le risque de panne puisqu'ils disposent d'un seul serveur qui organise tout le mécanisme et le fonctionnement du système.

En opposition aux systèmes centralisés, on trouve les systèmes distribués qui règlent ce type de problème puisqu'ils visent à réaliser un équilibrage de charge de travail. Dans le cas de très grosse demandes, telles que pour les recherches sur les moteurs de recherche, puisque ce genre de systèmes est constitué d'un ensemble de machines reliées entre eux par un réseau constituant un ensemble de systèmes centralisés collaboratifs.

Dans notre travail, on suppose qu'un cluster d'ordinateurs est dédié pour le stockage de notre base de données massive, afin d'appliquer un traitement parallèle.

II. Les Types des Systèmes :

Dans le monde de l'informatique, il existe plusieurs systèmes parmi lesquels les plus connus sont les systèmes : centralisé, distribué, et P2P (Peer to Peer) [13]. Ces systèmes sont adaptés pour le stockage et le traitement des données sémantique vue la nature du web.

II.1. Les Systèmes Centralisés :

C'est un système qui est tout à fait localisé sur la même machine et accessible par le programme. Dans ces systèmes, un système logiciel s'exécutant sur une seule machine qui représente le serveur du système, en accédant localement aux ressources nécessaires (données, code, périphériques, mémoire …).

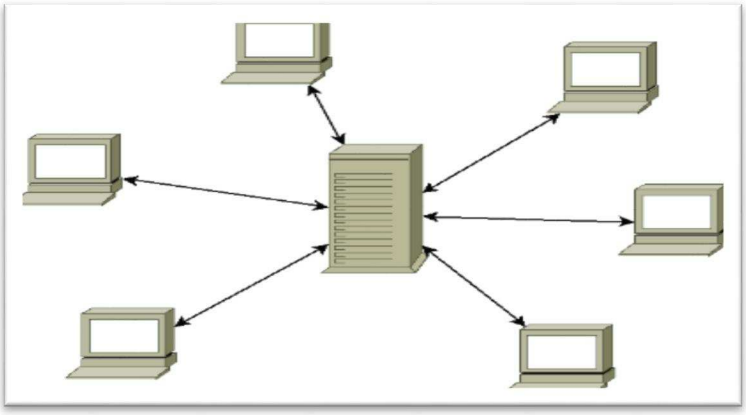


Figure 12 : L'architecture des systèmes centralisés.

II.2. Les Systèmes Distribués :

Ces systèmes comportent un ensemble d'ordinateurs indépendants connectés en réseau et communiquant via ce réseau. Cet ensemble apparaît du point de vue de l'utilisateur comme une unique entité. C'est la réalisation d'une certaine tâche globale par un ensemble d'entités logicielles distribuées.

Autrement dit, ces machines sont constituées d'un ensemble de nœuds. Chaque nœud possède un processeur et une mémoire appelée mémoire locale. Les communications entre nœuds se font soit par échange de messages entre les processeurs ou bien directement par l'accès à une mémoire distante.

Parmi les exemples de systèmes distribués existants on peut citer : le web auquel un internaute se connecte à un nombre quelconque de navigateurs web (clients) permettant l'accès à distance à un ensemble d'informations.

Dans ces systèmes, ce type de machine appelée serveur est facilement extensible à la fois grasse au nombre de processeurs et à la capacité mémoire qu'il dispose. Dans de tell situation, la mise en œuvre de la programmation parallèle sur ces machines est donc plus adaptée puisque le programmeur doit distribuer et gérer les données sur les mémoires locales et distantes.

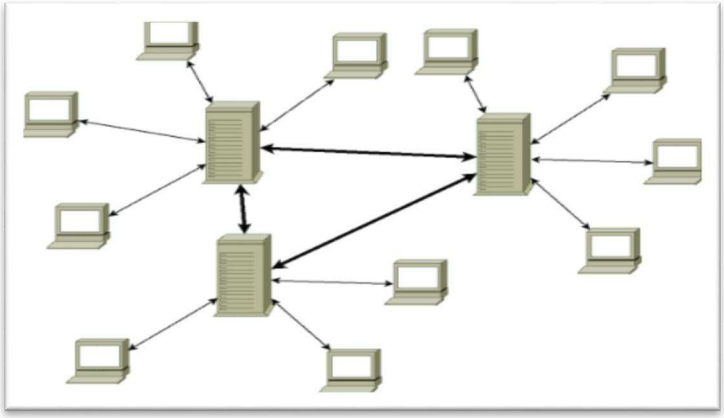


Figure 13 : L'architecture des systèmes distribués.

II.3. Les Système P2P :

L'acronyme P2P signifie Peer to Peer en anglais et Pair à Pair en français, est une technique utilisée pour le partage de ressources parfaitement légales comme des logiciels open sources (Linux, EDI, outils, etc.) ou des versions d'évaluation par exemple.

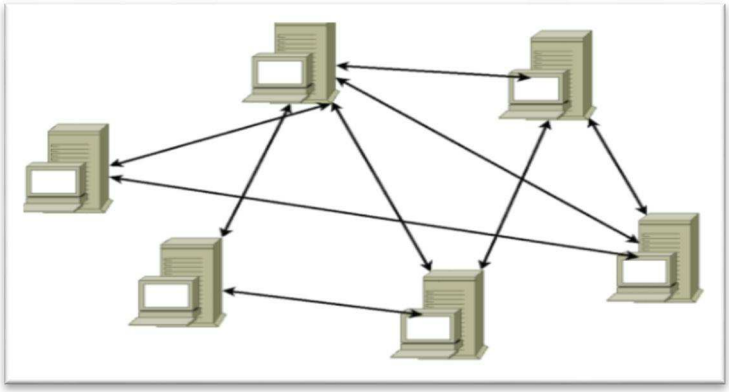


Figure 14 : L'architecture des systèmes P2P.

III. La programmation parallèle :

Lorsque la taille des données à traiter est grande, voire immense, il devient nécessaire de lancer plusieurs calculs simultanés. Ce principe est dénoté par le terme calcul parallèle qui permet de traiter certains gros calculs en un temps donné par duplication des éléments de calcul. Ce type de programmation semble parfait pour le traitement des données 'une base de données de type BigData. Le calcul parallèle apporte une solution pour des problèmes nécessitant une haute tolérance aux fautes et nous offre un coût performant et un accroissement de la puissance d'un élément de calcul.

Il existe deux grandes approches du parallélisme la première le parallélisme de la tâche (task ou contrôle parallelism) tel que chacune va être décomposée en sous tâches exécutant simultanément à l'aide de plusieurs processeurs, et la deuxième est le parallélisme de données (data parallelism) qui consiste à traiter les données en même temps par des processeurs différents. Dans notre travail qui consiste à utiliser MapReduce, nous allons adapter la première approche.

IV. Le Modèle MapReduce :

MapReduce est un modèle qui a été introduit par Google en 2004, il est aujourd'hui l'un des modèles de programmation parallèle le plus utilisé. Dans laquelle une succession d'ensemble de tâches indépendantes s'exécute, elle permet le traitement parallèle de grandes masses de données. MapReduce est fondé sur un schéma qui permet la parallélisation automatique, l'équilibrage de charge, des optimisations sur les transferts disques et réseaux et de la tolérance aux pannes.

Ce modèle est plus adapté au système distribué, puisqu'il consiste à exécuter un problème de manière distribuée, en le découpant en un ensemble de sous problèmes et en traitant chacune d'eux sur une machine différente.

IV.1. Fonctionnement de MapReduce :

Le principe de MapReduce d'une manière générale est comme suit (figure 15) [14] :

- Un problème est soumis par l'utilisateur à un nœud maître c'est-à-dire le serveur, ce dernier, commence à découper les données en entrée et choisit des nœuds-esclaves libres, appelés aussi Job-Tracker pour leur attribuer des tâches Map ou Reduce à exécuter en parallèle. Si un nœud est suspecté d'être défaillant, le maître relance ses tâches sur une nouvelle machine. D'autre part, si un nœud exécutant des tâches est peu performant, le maître peut lancer de façon spéculative une copie de ses tâches sur une autre machine pour obtenir un résultat plus rapidement. Le mécanisme de spéculation consiste donc à ordonnancer une réplique supplémentaire si le calcul de la tâche n'a pas été effectué dans un délai prédéfini.

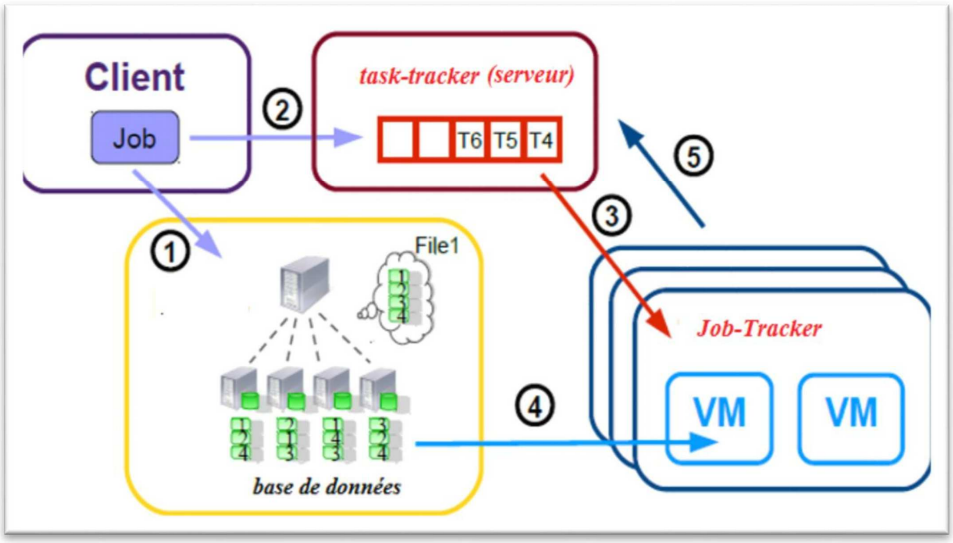


Figure 15 : Architecture de Map-Reduce.

- Ensuite, dans chaque machine des tâches, des fonctions Map et Reduce sont exécutées. La fonction Map consiste à former des couples <clé, valeur> à partir des fragments dénotés aussi splits envoyés par le serveur. Une opération de shuffle est effectuée à cet ensemble de couples qui vont être groupés selon la clé. Par la suite la fonction Reduce consiste à réduire les résultats intermédiaires c'est-à-dire, les groupes indexés par clé en une forme finale, avec une valeur pour chacune des clés distinctes.

La figure ci-dessous représente une architecture générale du fonctionnement MapReduce.

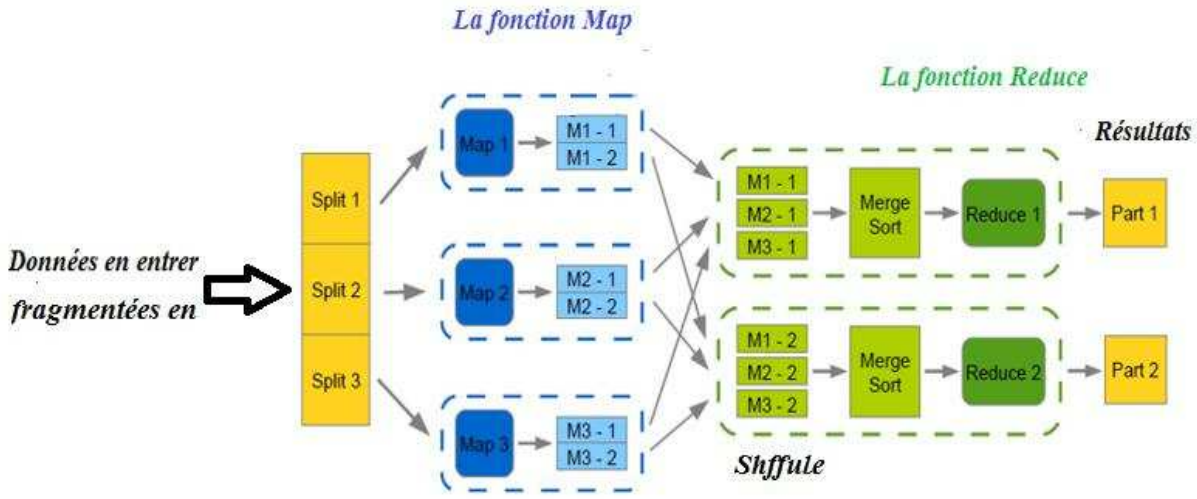


Figure 16 : Flot de données du MapReduce [14].

Pour résumer en peut déduire qu'il y a quatres étapes distinctes dans un traitement Map/Reduce expliquées dans un paragraphe précédent :

- ✓ ***Découper*** (split) : les données d'entrée sont subdivisées en plusieurs fragments.
- ✓ ***Mapper :*** chacun de ces fragments sont réorganiser en sous-groupes dans lequel MapReduce affecte une clé pour obtenir un ensemble de couples (clé ; valeur).
- ✓ ***Grouper*** (shuffle) : ces couples (clé ; valeur) par clé.
- ✓ ***Réduire*** (reduce) : cette étape consiste à réduire les résultats intermédiaires c'est-à-dire, les groupes indexés par clé en une forme finale, avec une valeur pour chacune des clés distinctes.

➢ **Exemple :**

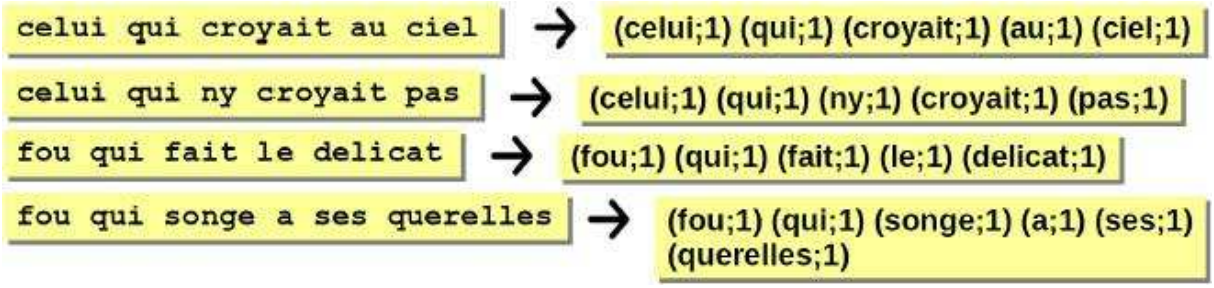
Pour bien éclaircir le principe de ce mécanisme on va étudier cet exemple classique : le comptage de mots [15].

- On a un fichier textuel en entrée, et on veut connaître le nombre d'occurrences de chacun des mots dans ce fichier.

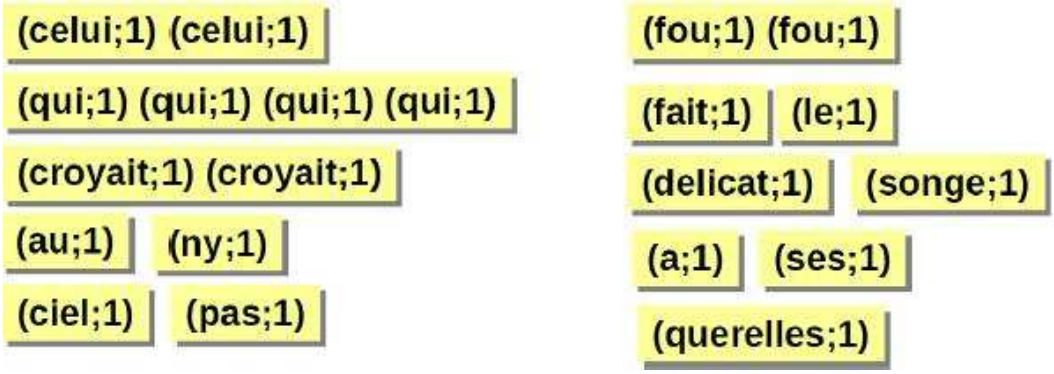
```
Celui qui croyait au ciel
Celui qui n'y croyait pas
[…]
Fou qui fait le délicat
Fou qui songe à ses querelles
```

- Le texte va être découpé en 4 unités de traitement, en le fragmenter par exemple par ligne.

- Le système applique une séparation de l'unité en mots et une construction des couples <clé, valeur> appliquée par Map à chaque morceau de texte, tel que la clé ici représente le mot séparé.



- Un traitement de regroupement (ou shuffle) des clés communes sont opérées par Reduce, ça veut dire les couples qui contiennent les mêmes clés par exemple « celui » seront dans un seul ensemble. Comme il est montré ci-dessous :



- Enfin, la fonction reduce fait la sommation des valeurs de toutes les paires de clé commune va être écrite dans un (ou des) fichier(s) résultats, ici on a trouvé que le nombre d'occurrences du mot « qui » est égal à quatre ainsi de suite.

```
qui: 4
celui: 2
croyait: 2
fou: 2
au: 1
ciel: 1
ny: 1
pas: 1
fait: 1
[…]
```

V. Conclusion :

La programmation parallèle devienne très importante lorsqu'il y a plusieurs traitements et surtout quand ils sont effectuées sur de base de données contenant un grand nombre de données parce qu'elle est basée sur un mécanisme très approprié à ce genre de traitement.

Un modèle de programmation a été étudié dans ce chapitre appelé MapReduce, ce paradigme est le plus utilisé, ceci est principalement en raison de sa capacité à stocker ainsi que de distribuer de grands ensembles de données à travers de nombreux serveurs. Ces serveurs peuvent être peu coûteux et ils opèrent également en parallèle. Ainsi il sert à augmenter la puissance de traitement.

I. Introduction :

Dans ce chapitre, Nous allons vous présenter un modèle qui permet de traiter de grandes quantités de données sémantique en parallèle sur des clusters de serveurs d'une manière fiable et insensible aux défaillances. Pour ce faire, nous allons adopter le paradigme MapReduce pour le traitement de données à grande échelle dans un environnement informatique distribué en utilisant Hadoop [4]. Ce dernier, permet aux applications de travailler avec des milliers de nœuds de données, sans que l'utilisateur se préoccupe des détails sur la répartition et la distribution des données et des calculs.

Notre principal objectif est de faciliter l'accès aux bases de données sémantiques afin d'atteindre un temps de réponse de plus en plus réduit pendant l'interrogation de ces données.

Dans ce qui suit, nous allons décrire l'ensemble des étapes nécessaires pour l'interrogation d'une base de données RDF suivant notre modèle. Tous d'abord, nous allons donner l'architecture générale de notre modèle contenant les différents composants de notre modèle, ensuite nous expliquerons le fonctionnement de chaque composant et enfin nous allons terminer ce chapitre par une conclusion.

II. Modèle de traitement de connaissances :

Le modèle qu'on propose envisage à tirer parti des avantages de l'utilisation du modèle MapReduce pour le traitement des connaissances volumineuses qui sont un ensemble de données sous forme des documents RDF vont être transformés en graphes.

Une requête est partitionnée et partagée sur plusieurs nœuds des calculs pour alléger la charge de travail et de réduire le temps du traitement, et dans le but d'avoir des résultats dans un temps de réponse raisonnable.

Afin de montrer le fonctionnement de notre modèle, nous proposons dans le schéma ci-dessous l'architecture générale de notre modèle.

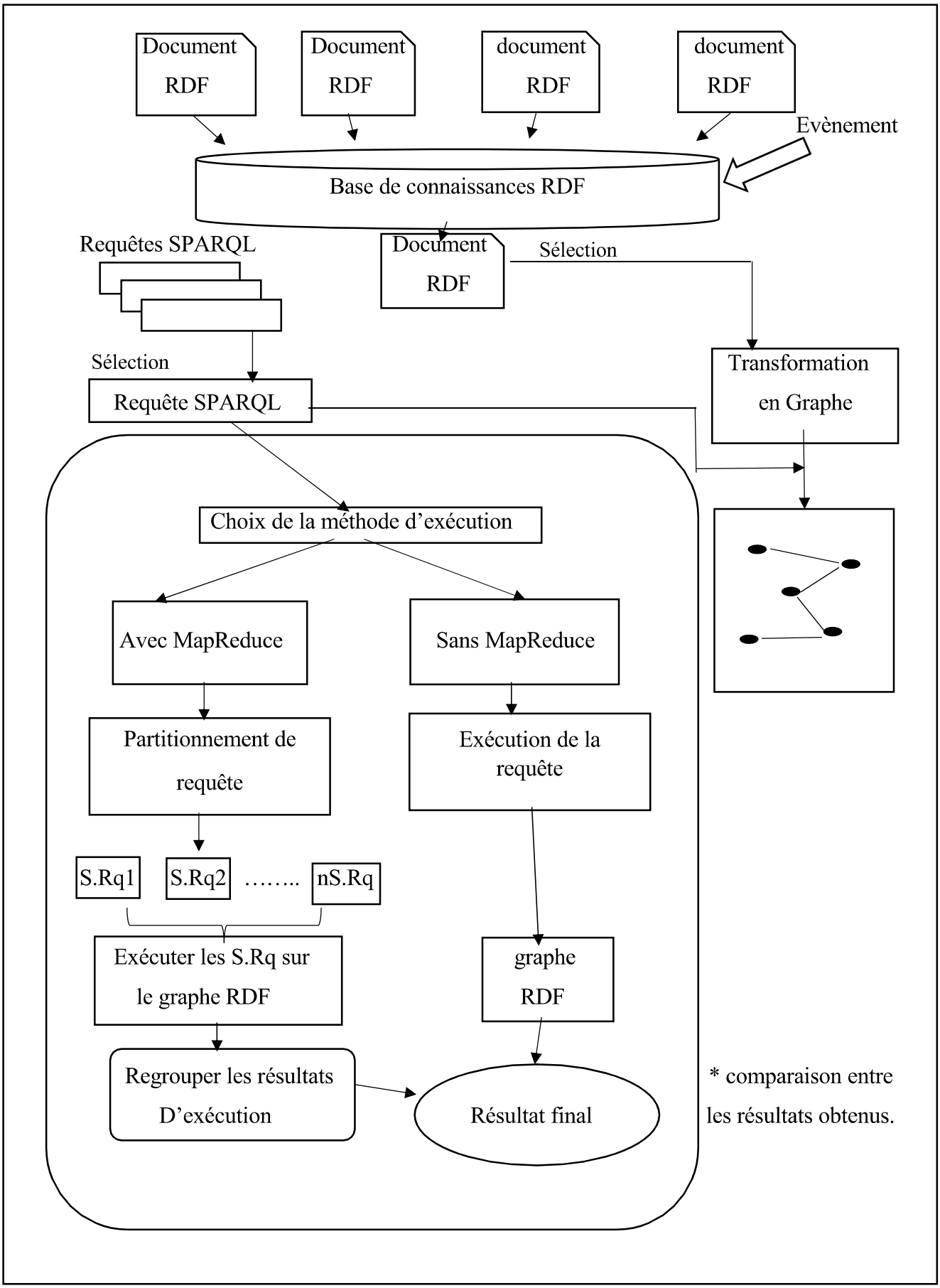


Figure 17 : Architecture du modèle de traitement des connaissances.

Le processus d'exécution de notre modèle passe par les étapes suivantes :

- Suite à un événement survenu, un document RDF est sélectionné parmi l'ensemble des documents sémantiques qui sont stockés dans une base de données. Ces documents représentent la connaissance à traité.
- L'étape suivante consiste à transformer le document sélectionné en graphe.
- Ensuite une requête adéquate est sélectionnée. Pour l'exécution de la requête SPARQL, l'utilisateur doit choisir le type d'exécution parmi les deux modes d'exécutions suivants:
  - ✓ soit avec le partitionnement de requête en utilisant le modèle de programmation parallèle : MapReduce.
  - ✓ soit sans partitionnement de requête c'est-à-dire que la requête va être exécutée tel qu'elle est sur le graphe.
- Enfin, le modèle affiche affichage de l'exécution de la requête ainsi que le temps d'exécution associé. Le modèle permet de comparer le temps d'exécution de la requête des deux modes (avec et sans MapReduce). Ceci permettra aux utilisateurs de comparer et de marquer la différence entre les deux modes.

Dans ce qui suit nous allons expliquer le fonctionnement de notre modèle en détaille.

Le modèle reçoit en entrée des données qui parviennent de plusieurs sources sous différentes formes. Ces derniers sont transformés en document RDF pour mettre en évidence la sémantique des données, ensuite elles seront stockées dans une base de données RDF sur des bases de données de type TripleStores [16].

Ce modèle donne la possibilité d'accéder à la base des documents sémantiques pour en choisir un document RDF parmi un ensemble de document RDF qu'il contient, ensuite, le document sera représenté sous forme graphique.

Suite à un évènement déclenché dans le système, une requête parmi un ensemble de requêtes sera exécutée sur ce document avec ou sans l'utilisation du MapReduce c'est-à-dire soit avec le partitionnement de la requête où sans partitionnement, Par la suite un résultat sera soumis sous une représentation graphique. À la fin une comparaison sera effectuer sur les deux graphes obtenus de l'exécution de la requête avec et sans MapReduce, pour déterminer laquelle de ces deux méthodes offre un temps d'exécution plus court, en visualisant le résultat obtenu.

II.1. Base de données RDF :

II.1.1. Sources de données RDF :

Les données RDF peuvent provenir de différentes sources qui ne sont pas forcément stockées dans une mémoire commune (figure 18). Ainsi, ce type de donnée ne suit pas généralement un modèle particulier de stockage, puisque ces données sont distribuées. Pour cela, il existe plusieurs supports de stockage dont la base de données RDF TripleStores qui contient uniquement des triplets RDF dans une grande et unique table ayant trois colonnes, une pour le sujet, une pour le prédicat, et la dernière est réservée pour l'objet. La récupération

de ces données se fait via des requêtes exprimées suivant le langage SPARQL à partir de base données nommée BigData.

Figure 18 : Sources de données dans les BigData

II.2. Les TripleStores :

Les TripleStores sont divisées en trois catégories, les TripleStores autochtones qui sont mises en œuvres à partir de zéro, les TripletStores SGBDR soutenues qui sont construits en ajoutant une couche RDF à un SGBDR et enfin les TripleStores NoSQL sont étudiés récemment en tant que gestionnaires de stockage possible pour RDF.

Afin de montrer le principe des TripleStores nous proposons dans la figure ci-dessous l'architecture de ce type de base de données.

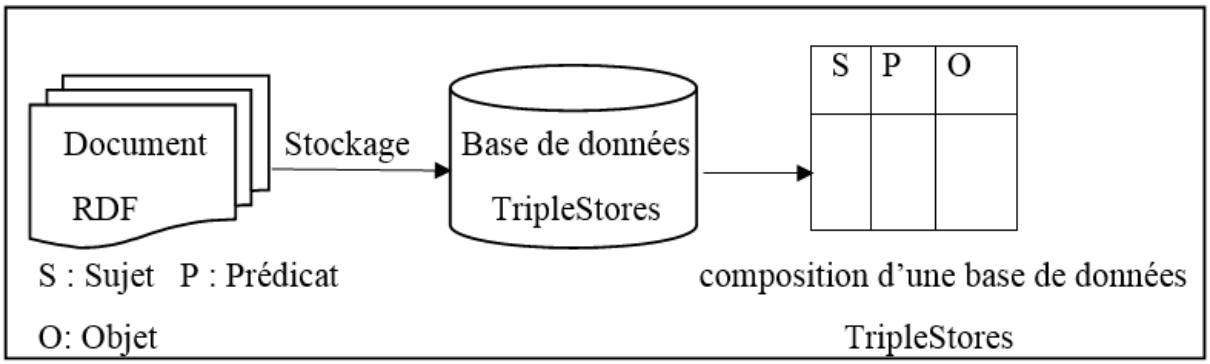


Figure 19: Architecture d'un TripleStore.

Dans le modèle, on opte pour l'utilisation d'une base de données TripleStores de type NoSQL pour stocker les données RDF. Notre choix se justifie par la grande taille des documents RDF que nous utilisons dans ce modèle.

II.3. Document RDF :

Le document RDF contient un ensemble de données de type RDF qui constitue les composants d'une ontologie, d'un schème RDF etc. Ce document est sélectionné lorsqu'un évènement survient pour ajouter, supprimer, modifier ou récupérer une donnée ou un ensemble de données. Le document RDF peut être représenté sous différents formes telles que la syntaxe XML (Extensible Markup Language), TURTLE (Terse RDF Triple Langage).

L'exemple cité ci-dessous, permet de visualiser une représentation d'un document RDF avec la syntaxe XML.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:admin="http://webns.net/mvcb/">
<foaf:PersonalProfileDocument rdf:about="">
 <foaf:maker rdf:resource="#me"/>
 <foaf:primaryTopic rdf:resource="#me"/>
 <admin:generatorAgent rdf:resource="http://www.ldodds.com/foaf/foaf-a-matic"/>
 <admin:errorReportsTo rdf:resource="mailto:leigh@ldodds.com"/>
</foaf:PersonalProfileDocument>
<foaf:Person rdf:ID="me">
<foaf:name>Nahet Malika</foaf:name>
<foaf:title>Mme</foaf:title>
<foaf:givenname>Malika</foaf:givenname>
<foaf:family_name>Nahet</foaf:family_name>
<foaf:nick>Fatima</foaf:nick>
<foaf:mbox_sha1sum>5d6686d9c37c299719c13a418738c6166f824356</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="facebook.com"/>
<foaf:phone rdf:resource="tel:0770830000"/>
<foaf:workplaceHomepage rdf:resource="work"/>
<foaf:workInfoHomepage rdf:resource="devlopment"/>
<foaf:schoolHomepage rdf:resource="university"/>
<foaf:knows>
<foaf:Person>
<foaf:name>Fatiha</foaf:name>
```

<foaf:mbox_sha1sum>5df9c3f34af6608f9e270dadbfa2df06b199513c</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="AMS"/></foaf:Person></foaf:knows>
<foaf:knows>
<foaf:Person>
<foaf:name>Amel</foaf:name>
<foaf:mbox_sha1sum>103c1dffbc9ea7965ea36f3234a522fa529c42d3</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="Kunstler"/></foaf:Person></foaf:knows>
<foaf:knows>
<foaf:Person>
<foaf:name>Rahma</foaf:name>
<foaf:mbox_sha1sum>6775fc9170c253f425d3e7f66dbe181dabb4d48c</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="crt"/></foaf:Person></foaf:knows></foaf:Person>
</rdf:RDF>

II.4. Transformation des données RDF en graphe :

Le processus d'une transformation en graphe passe par les trois étapes suivantes :

1. Lire le document RDF à transformer.
2. Créer des listes des sujets, objets et prédicats.
3. Créer des sommets et des arcs pour construire notre graphe.

Nous avons résumé ces étapes dans la figure ci-dessous.

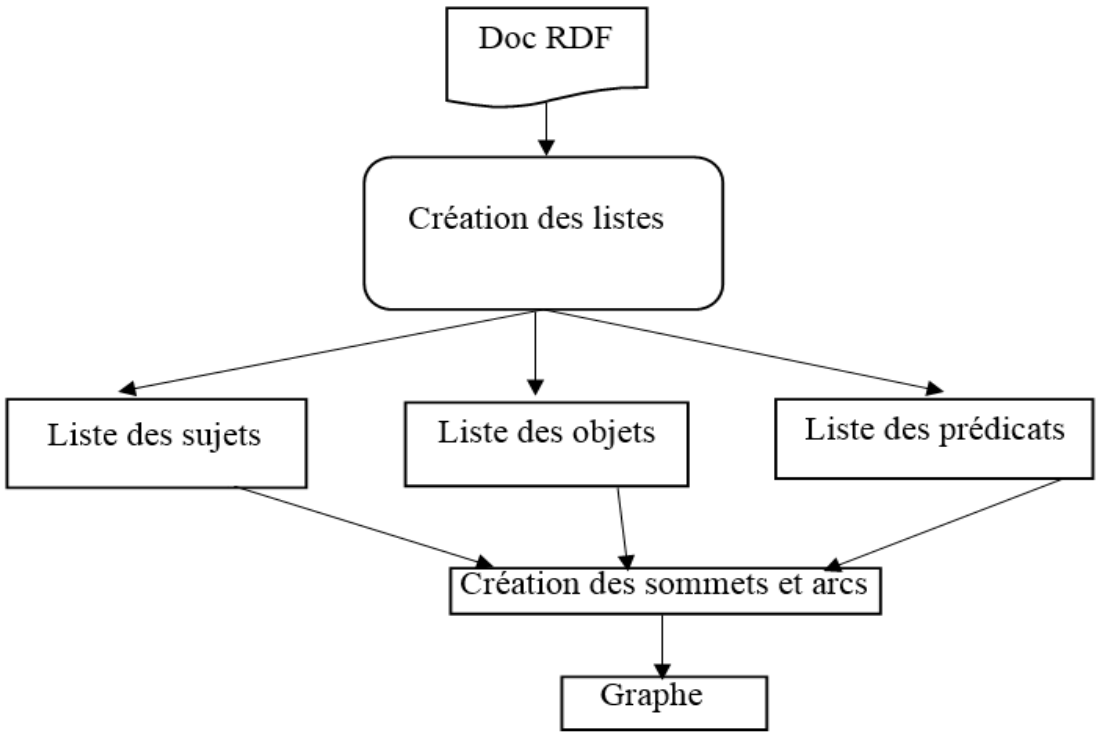


Figure 20: étapes de la transformation des documents RDF en graphe.

Pour la création des listes, une sélection va être effectuée sur une ressource en identifiant leur type, si c'est un sujet le système va tester si la liste existe il va placer ce sujet dans cette liste sinon il va créer une liste et placer un sujet après autre jusqu'à terminer tous les sujets pour passer à la liste des objets, pour cela le système va tester si le nœud est un objet il va vérifier si la liste existe sinon il va la créer et placer cet objet dans cette liste. Le système suivra le même paradigme pour la création de la liste prédicats.

La troisième étape dans ce processus c'est la création des sommets et des arcs pour la construction du graphe, elle permet de créer des sommets pour les sujets qui existent ainsi pour les objets de tel sorte le système va produire leur sommet s'il n'existe pas. A la fin il va tracer les arcs.

Le schéma ci-dessous explique la création des listes et la création des sommets et des arcs pour former à la fin un graphe RDF.

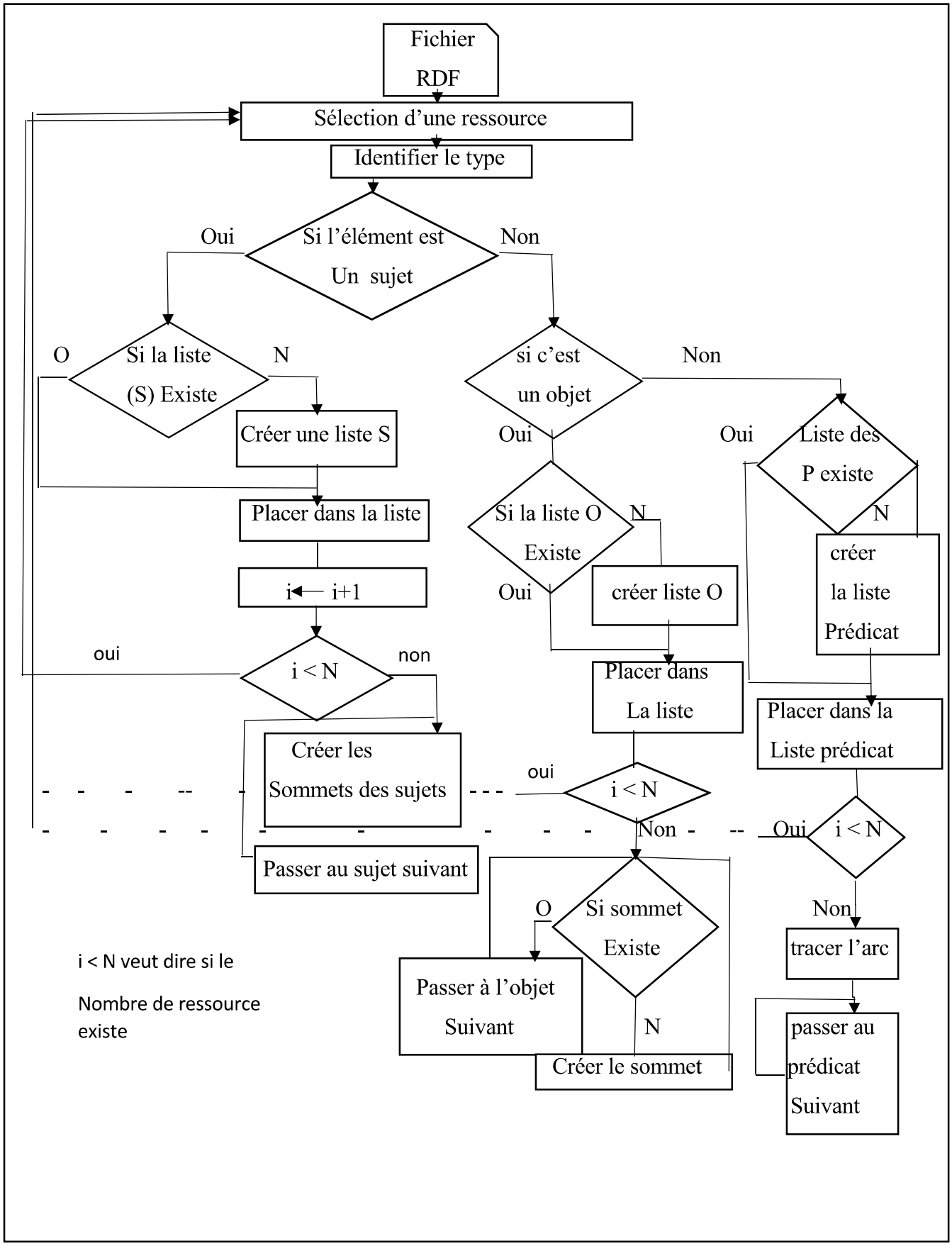

Fichier RDF
Sélection d'une ressource
Identifier le type
Si l'élément est Un sujet
Oui
Non
Si la liste (S) Existe
O
N
Créer une liste S
Placer dans la liste
i ← i+1
i < N
oui
non
Créer les Sommets des sujets
Passer au sujet suivant
si c'est un objet
Oui
Non
Si la liste O Existe
N
Oui
créer liste O
Placer dans La liste
oui
i < N
Non
Si sommet Existe
O
N
Passer à l'objet Suivant
Créer le sommet
Liste des P existe
Oui
N
créer la liste Prédicat
Placer dans la Liste prédicat
Oui
i < N
Non
tracer l'arc
passer au prédicat Suivant
i < N veut dire si le
Nombre de ressource existe

Figure 21 : Diagramme de la création des listes et la création des sommets et des arcs.

II.5. Les requêtes SPARQL :

Le composant des requêtes SPARQL consiste à interroger la base de données RDF. Ce composant contient plusieurs choix, chaque choix représente une requête particulière. Le choix d'une requête déclenche un appel de la classe « query ».

- ✓ Exécution de la requête SPARQL sans MapReduce. Par exemple :

```
"SELECT * WHERE { " +
 " ?Person foaf : name ?x . " +
 "  ?Person foaf : knows ?Person2 . " +
 "    ?Person2 foaf : name ?y ." +
 "    FILTER (?y = /*rahma/*)  " +
"} ";
```

Cet exemple sert à trouver toutes les personnes qui savent la personne « rahma », de telles sortes ici elle ne va pas être partitionnée.

- ✓ Exécution de la requête SPARQL avec MapReduce, on peut utiliser la même requête mais dans ce cas chaque nœud reçoit une partie de la requête.

```
 SELECT * WHERE { " +
 " ?Person foaf : name ?x . " +          ——————→ partition 1
 "  ?Person foaf : knows ?Person2 . " +  ——————→ partition 2
 "    ?Person2 foaf : name ?y ." +       ——————→ partition 3
 "    FILTER (?y = /*rahma/*)  " +       ——————→ partition 4
"} ";
```

III. Le paradigme MapReduce :

Le problème avec le parallélisme jusqu'ici était «Avoir un Framework disponible, déployable facilement, et permettant l'exécution et le suivi des différentes tâches parallélisables de maniéré rapide et simple à mettre en oeuvre ». L'idée étant d'avoir un outil «off the shelf » qui puisse être installé et configuré rapidement au sein d'une entreprise ou d'une université et qui permet à des développeurs d'exécuter des tâches distribuées avec un minimum de formation requise. L'outil en question devant être facile à déployer, simple à supporter, et pouvant permettre la création de clusters de taille variables extensibles à tout moment. Un nouvel algorithme a été mis au point par Google afin de distribuer des traitements sur un ensemble de machines avec son systeme GFS Google File System, le

premier article a été publié en 2004 par : Jeffrey Dean and Sanjay Ghemawat sous le nom de : «MapReduce : Simplified Data Processing on Large Clusters».

a. Principe et fonctionnement :

MapReduce est un Framework, open source de la communauté Appache, dans lequel sont effectués des calculs parallèles, et souvent distribués, de données potentiellement très volumineuses. MapReduce est le premier modèle à combiner distribution massive et reprise sur panne dans le contexte d'un cloud de serveurs à bas coûts.

Une fois le Framework installé (avec un minimum de configuration) il a la charge de repartir automatiquement la ou les tâches sur un cluster important de machines. Les machines ne sont pas nécessairement puissantes.

Parmi les calculs simples et intéressants que pourrait implémenter MapReduce nous avons :

- **Distribution en grappe (Cluster) :** Trouver les patterns dans un certain nombre de fichiers.
- **Calcul des fréquences d'accès aux URLs.**
- **Graphe inversé du web :** étant donnée une liste (cible, source) entre une paire d'URLs, le problème se résume à trouver (cible, liste (sources)) i.e : trouver toutes les URLs qui pointent vers une cible donnée.
- **Construction des indices inversés à partir des pages web explorées**,
- **Tris distribués.**

Apache software fondation a très rapidement repris l'idée. Il a développé le concept en mettant ainsi en oeuvre un Framework à part entière appelé Hadoop. Hadoop est gratuit et bénéficie d'une très grande communauté de programmeurs et d'entreprises essentiellement du web 2.0 comme Yahoo, Amazon, etc. Nous aborderons plus en détail le Framework dans les sections suivantes.

Pour la création d'un job (tâche) MapReduce, le programmeur spécifie une fonction map et une fonction reduce : deux primitives présentes dans la programmation fonctionnelle. Le Framework exécute plusieurs instances de ces fonctions en parallèle. La fonction map traite une paire (Cl; Valeur) pour en générer une autre paire (Clé; Valeur) en sortie.

- Map (key in; value in)! ((key out; value out)
- Reduce (key out; value out)! Traitement

Dans notre cas d'exécution d'une requête SPARQL, le MapReduce consiste à :

- Découper cette requête en multiples sous-requête (le Map de MapReduce).chaque des sous-requêtes pouvant être traités par différents serveurs d'une ferme (style hadoop).

- Récupérer les multiples résultats des sous-requêtes et les réduire en un résultat global (le Reduce du MapReduce).

Je trouve un simple exemple pour montrer ce fonctionnement :

Exemple : soit un ensemble de fichiers contenant des mots. Etablir une liste des mots et pour chaque mot une liste des fichiers ou il apparait.

F1 : Stransboufg Nantes Paris, F2 : Mulhouse Colmar Stransboufg.

- Colmar : F2.
- Nantes : F1.
- Mulhouse : F2.
- Paris : F1.
- Stransboufg : F1, F2.

- Un code index :

```
Map (String filename, String line):
    foreach word w in line:
    EmitIntermediate (w, filename);
Reduce (String key, Iterator intermediate_values):
// key=word, intermediate_values=filenames
foreach F in intermediate_values:
result + = F + ‘  ‘;
Emit (key, result);
```

Un certain nombre de fonctions map peut être exécuté en parallèle sur des données partagées dans le cluster. Produisant ainsi un ensemble de paires (Clé; Valeur) intermédiaires. La fonction de réduction est chargée de fusionner toutes les valeurs intermédiaires ayant la même clé et/ou effectuer un traitement local.

Comme mentionné précédemment, le système d'exécution prend soin de la répartition des données, des maps et des reduces, tout en prenant compte des problèmes de parallélisation pour mieux utiliser les ressources des grands systèmes distribués.

b. Répartition du Map/Reduce :

Le modèle d'exécution MapReduce utilise les termes suivants :

- ✓ **Machine :** Un ordinateur physique et réel, qui fait partie d'un cluster distribué.
- ✓ **Tâche (task) ou worker :** Un processus en cours d'exécution sur une machine.
- ✓ **nœud (Node) :** on le considère comme étant un gestionnaire de processus où un nœud est fondamentalement un Java Daemon Node (cela deviendra plus clair en abordant Hadoop).

Nous décrivons un "Job MapReduce" complet en les étapes suivantes :

- Le Framework découpe d'abord les fichiers de données d'entrée en M fragments de taille fixe : en générale 16 à 64 mégaoctets par pièce. Les fragments obtenus sont ensuite distribués sur les machines du cluster. Habituellement, il existe 3 copies (modifiables par l'utilisateur) de chaque fragment à des fins de tolérances aux pannes. A partir du programme "utilisateurs", ses copies sont lancées dans les nœuds du cluster.

  Un nœud spécial du cluster est le nœud maitre, les restes sont des workers affectés au maitre. Il y a donc M tâches map et R tâches reduce à affecter. Le maitre sélectionne les workers inactifs et attribue à chacun un map.

  Une fois que les maps génèrent leurs sorties intermédiaires respectives, le maitre assigne alors les Reduce de la même manière que pour les maps.

- Un worker affecté pour une tâche map correspondant à un fragment en entrée et transmet chaque pair à une instance map définie par l'utilisateur. Les couples (clé ; valeur) intermédiaires générés par les map sont stockés dans les mémoires tampons de la machine associée.

- Les paires sont mises en mémoire sont périodiquement écrites sur le disque local et distribuées selon la fonction de partitionnement dans les R machines correspondantes. Le Framework fournit une fonction de partitionnement par défaut mais qui peut, aussi, être modifiée par le programmeur selon les besoins. L'emplacement des paires en mémoire tampon dans le disque local sont passés au maitre qui les transmet, à son tour, aux workers reduce.

- Quand un worker reduce est avisé, par le maitre, de son emplacement il utilise un appel de procédure distante. Le worker doit lire les données à partir des buffers des mapworker. Quand un reducer a terminé sa lecture de données intermédiaires, il trie les clés intermédiaires (key out) de sorte que toutes les occurrences de la même clé sont regroupées. Le tri est une étape nécessaire, plusieurs clés peuvent correspondre à la même tâche de réduction. Si par ailleurs la taille des données intermédiaires est trop grande pour tenir en mémoire, un tri externe est utilisé.

- Ensuite le reducer itère sur les données intermédiaires triées et pour chaque clé intermédiaire (unique) rencontrée, il passe la clé ainsi que l'ensemble des valeurs correspondantes aux utilisateurs de la fonction reduce. La sortie de la fonction de réduction est ajoutée à un fichier de sortie finale pour chaque partition réduite.

- Lorsque toutes les tâches Map et reduce ont été accomplies, le nœud maitre réveille le programme utilisateur (main). A ce stade les appels des MapReduce ont été exécutés et le programme peut alors continuer son déroulement.

- A la fin, le processus doit générer les résultats en tant que fichiers de sorties ainsi qu'un fichier sucés qui nous garantit le bon déroulement des jobs.

IV. Conclusion :

La taille des connaissances trouve ses racines dans un domaine en pleine croissance qui est le BigData. Par conséquent, cela pose un problème pendant leur traitement. Pour ce faire, nous avons présenté dans ce chapitre une solution qui peut générer ce problème suivant un modèle qui est constitué de plusieurs étapes pour réussir à obtenir un bon traitement pour ces connaissances qui sont représentées sous forme de documents RDF.

Notre modèle consiste à extraire un document RDF stocké dans une grande base de connaissances pour transformer en graphe, car le traitement sur les graphes sera facile à la machine en comparant avec celui qui est appliqué sur un document RDF textuel et ça pour éviter la complexité exponentielle. Par la suite, l'utilisateur doit choisir une requête à partir d'un ensemble de requêtes pour être exécutée sur un graphe. Pour l'exécution de la requête, il y a deux modes selon le choix de l'utilisateur, entre l'application d'une requête sans MapReduce c'est-à-dire sans partitionnement de requête ou avec MapReduce ici la requête va être partitionnée en sous-requêtes (S.Rq). Ces dernières sont exécutées sur le graphe en parallèle, grâce au modèle MapReduce qui regroupe les résultats obtenus en un seul résultat. A la fin l'utilisateur peut faire la comparaison entre l'exécution de requête sans et avec MapReduce.

I. Introduction :

Afin de mettre en place le modèle proposé dans le chapitre précédent, ainsi que ses différents composants, nous allons présenter dans ce chapitre les différents outils de développements et les logiciels utilisés pour l'implémentation de notre modèle.

Dans ce chapitre, nous allons présenter notre application en indiquant en détail les étapes nécessaires pour son utilisation, et enfin on va terminer par une conclusion.

II. Environnement de travail :

Pour l'implémentation de notre modèle nous avant travailler dans une machine ayant les caractéristiques suivantes (les ressources matérielles) :

- PC portable : hp86.
- Processeur : Intel Core i3-3110M CPU@2.40GHz x 2.
- Mémoire vive d'une capacité de 4Go.
- Type du système : système d'exploitation 64 bits, processeur 64 bits.

Et comme ressource logicielle, nous avons utilisé :

- Système d'exploitation : Windows 8.1.
- Langage de programmation : JAVA.
- L'EDI : NetBeans de version 8.2.

Il faut noter que les résultats obtenus dépondent des ressources matérielles et logicielles utilisées.

II.1.Langage de programmation :

Pour l'implémentation de notre modèle, nous avons utilisé un langage de programmation qui constitue une interprétation conventionnelle d'un algorithme afin de produire un programme informatique. Les langages de programmation représentent d'une manière générale un moyen de communication entre le programmeur et l'outil informatique et les programmeurs entre eux.

Ainsi pour la concrétisation d'un modèle algorithmique, il l'interprète sous un langage de programmation précis, notre choix pour implémenter notre modèle s'est orienté vers le langage java.

Par rapport à la plupart des autres langages de programmation existants, java met à la disposition du développeur une API très riche en lui permettant de faire de très nombreuses choses [17]. Contrairement à des langages comme le C, où il fallait que le programmeur code lui-même des fonctionnalités basiques comme le chargement d'images, java nous propose à peu près tout ce dont nous avons besoin directement dans le JDK. Ceci est un énorme avantage, qui augmente la productivité du développement.

II.1.1. Java :

Java est un langage de programmation développé par Sun Microsystems [18], qui permet de créer différents types de programmes informatiques, une de ces grandes force c'est sa portabilité, c'est-à-dire une fois qu'un programme a été créer il fonctionne automatiquement sous Windows, Lunix, Mac, OS ...

Pour développer des applications Java il existe une large gamme d'outils payants ou libres, parmi eux on distingue les IDE (Integrated Developpent environment) [19], qui représente une application logicielle permettant aux utilisateurs d'écrire plus facilement. Il existe beaucoup d'IDEs qui offrent des fonctionnalités comme la coloration syntaxique et la compilation de code, qui aident l'utilisateur à coder plus facilement.

NetBeans est un environnement de développement intégré (EDI), open source et multi-langue, créé par Sun et racheté par Oracle [18]. Il permet aux programmeurs d'écrire, compiler, déboguer, et déployer leurs programmes, il est compatible avec Windows, MacOs, Linux et Solaris, aussi il peut être utilisé comme support pour d'autres langages C/C++, PHP, HTML5… Il permet d'intégrer une interface graphique en utilisant la syntaxe du langage JAVA.

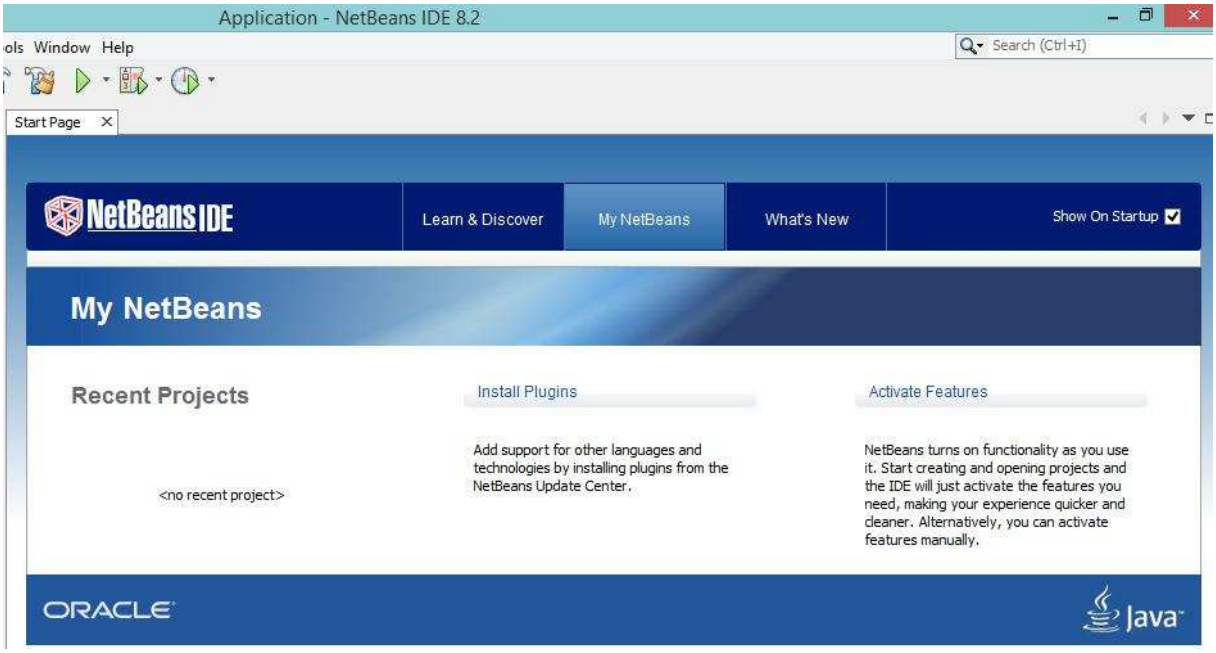


Figure 22 : NetBeans IDE 8.1.

III. Les API utilisées :

Une API « Application Programming Interface » est une interface de programmation qui permet de définir la manière dont un composant informatique peut communiquer avec un autre [17]. C'est donc une interface de code source fournie par un système informatique ou une bibliothèque logicielle, en vue de répondre aux requêtes pour des services qu'un programme informatique pourrait lui faire. La connaissance des API est indispensable à l'interopérabilité entre les composants logiciels.

De plus, il existe énormément d'API tierces de très bonnes qualités, pour des fonctionnalités concernant les graphes RDF tel que pour la création, la modification et la visualisation dans une interface graphique de ces graphes, parmi ces API on a l'API Jena qui permet de créer, stocker et interroger en langage SPARQL des graphes RDF, il y a l'API Jung qui offre la possibilité de visualiser une variété de graphes (pas forcément des graphes RDF) ou un réseau, il y a ainsi l'API JenaJung qui est une liaison entre l'API Jena et l'API Jung dans le but de permettre la création des graphes Jung à partir de modèle Jena.

III.1. DOM et SAX :

Les analyseurs XML sont également divisés selon l'approche qu'ils utilisent pour traiter le document. On distingue actuellement deux types d'approches :

- DOM (Document Object Model) : est un API indépendant de tous langages qui permet de définir la structure d'un document sous forme d'un arbre [17]. Elle permet de modéliser, parcourir et modifier n'importe quel document.
- SAX (Simple API for XML) : Cette API a été développée par David Megginson [17]. Ce type de parseur utilise des événements pour piloter le traitement d'un fichier XML. Un objet (nommé handler en anglais) doit implémenter des méthodes particulières définies dans une interface de l'API pour fournir les traitements à réaliser : selon les événements, le parseur appelle ces méthodes. Les classes de l'API SAX sont regroupées dans le package org.xml.sax.

III.2. JDOM :

JDOM est une API java qui facilite la manipulation de document XML [17]. Cette API encapsule un document XML dans un arbre d'objets, ainsi il est disponible d'ajouter ou d'enlever des nœuds en respectant la norme XML. De même, il est possible de créer des documents XML en collaboration avec SAX et DOM.

Nous avons regroupé les classes utilisées dans notre application avec leurs significations dans le tableau ci-dessous :

Classes	Son utilité
org.jdom.Document	Cette classe encapsule un document XML.
org.jdom.input.SAXBuilder	Cette méthode établi un document JDom a partir d'un fichier (flux, lecteurs, URL, ou un SAX InputSource) à l'aide d'un analyseur SAX.
org.jdom.output.Format	Classe pour encapsuler un document XML Outputter.
org.jdom.output.XMLOutputter	Classe qui génère un document JDom comme un flux d'entrées.
org.jdom.JDOMException	Représente les exceptions que les classes JDom peuvent avoir.

Tableau 1 : Les classes utilisées de JDom.

On montre dans le code ci-dessous l'utilisation de la classe qui permet de parser et parcourir un fichier XML tel que le paramètre « path » est le chemin du fichier à parser.

```
199     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
200         JFileChooser fc = new JFileChooser();
201
202         if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
203             //Recupiration le chemin de fichier.
204             String path = fc.getSelectedFile().getAbsolutePath();
205             //charger le fichier selectionné dans le f.
206             File f = fc.getSelectedFile();
207             path2 = f.getAbsolutePath();
208             try {//transformation d'un fichier RDF en XML.
209                 SAXBuilder builder = new SAXBuilder();
210                 File fichierXML = new File(path);
211                 try {
212                     documentList.add(builder.build(fichierXML));
213
214                 } catch (IOException e) {
215                 }
216                 //Collection de l'arbre XML.
217                 XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
218                 //Affichage de nom de fichier selestionné dans le jComboBox.
219                 jComboBox1.addItem(fc.getSelectedFile().getName());
220                 //Affichage de XML forme.
221                 jTextArea1.setText(sortie.outputString(documentList.get(documentList.size() - 1))); //la ve
222
223             } catch (JDOMException ex) {
224                 Logger.getLogger(home.class.getName()).log(Level.SEVERE, null, ex);
225             }
226         }
227
228     }
```

Figure 23 : Méthode qui permet de parsser un fichier XML

Dans l'exemple suivant, nous allons le fichier RDF :

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:admin="http://webns.net/mvcb/">
<foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
  <admin:generatorAgent rdf:resource="http://www.ldodds.com/foaf/foaf-a-matic"/>
  <admin:errorReportsTo rdf:resource="mailto:leigh@ldodds.com"/>
</foaf:PersonalProfileDocument>
<foaf:Person rdf:ID="me">
<foaf:name>Nahet Malika</foaf:name>
<foaf:title>Mme</foaf:title>
<foaf:givenname>Malika</foaf:givenname>
<foaf:family_name>Nahet</foaf:family_name>
<foaf:nick>Fatima</foaf:nick>
<foaf:mbox_sha1sum>5d6686d9c37c299719c13a418738c6166f824356</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="facebook.com"/>
```

<foaf:phone rdf:resource="tel:0770830000"/>
<foaf:workplaceHomepage rdf:resource="work"/>
<foaf:workInfoHomepage rdf:resource="devlopment"/>
<foaf:schoolHomepage rdf:resource="university"/>
<foaf:knows>
<foaf:Person>
<foaf:name>Fatiha</foaf:name>
<foaf:mbox_sha1sum>5df9c3f34af6608f9e270dadbfa2df06b199513c</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="AMS"/></foaf:Person></foaf:knows>
<foaf:knows>
<foaf:Person>
<foaf:name>Amel</foaf:name>
<foaf:mbox_sha1sum>103c1dffbc9ea7965ea36f3234a522fa529c42d3</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="Kunstler"/></foaf:Person></foaf:knows>
<foaf:knows>
<foaf:Person>
<foaf:name>Rahma</foaf:name>
<foaf:mbox_sha1sum>6775fc9170c253f425d3e7f66dbe181dabb4d48c</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="crt"/></foaf:Person></foaf:knows></foaf:Person>
</rdf:RDF>

Dans notre application, le fichier RDF est transformé en fichier XML (figure). Pour confirmer que ce fichier RDF est valide et lui donner un espace de nom. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche…) entre systèmes d'informations hétérogènes (interopérabilité).

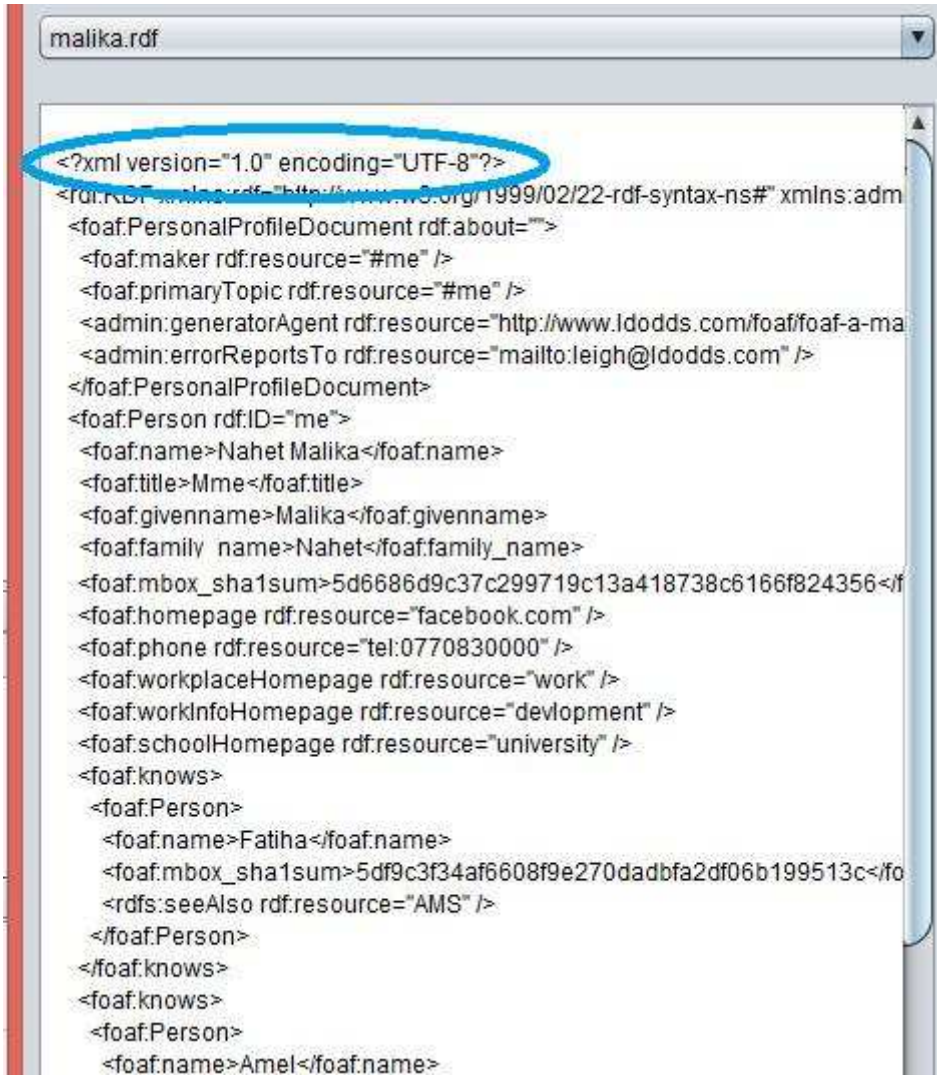


Figure 24 : Fichier RDF transformé en XML.

III.3. Jena-3.0.1:

Jena est une API java spécifique pour le développement d'applications dédiées au web sémantique [17]. Elle permet de manipuler plusieurs langages et modèles de représentation de données sémantiques tels que OWL, RDF/RDFS, SPARQL, ainsi elle fournit des classes Java qui permettent la création, l'interrogation et le maintien de données RDF en mémoire ou sur disque.

Nous avons regroupé toutes les classes utilisées dans notre application avec leur signification dans notre application dans le tableau ci-dessous :

Classes	Utilisation
org.apache.jena.query.Query	Cette classe est utilisée pour la création d'une requête SPARQL.
org.apache.jena.query.QueryExecution	Cette classe est utilisée pour l'exécution de la requête.
org.apache.jena.query.QueryExecutionFactory	Définit une variable qui contient

	QueryFacory.
org.apache.jena.query.QueryFactory	Méthode qui permet d'accéder au différents parseurs.
org.apache.jena.util.FileManager	Donne l'accès aux ressources d'un fichier par l'URL.
org.apache.jena.rdf.model.Model	Cette classe fournit un ensemble de méthodes pour créer un modèle à partir d'un fichier RDF.
org.apache.jena.rdf.model.ModelFactory	Cette classe fournit des méthodes pour créer des modèles RDF.
org.apache.jena.rdf.model.RDFNode	Interface couvrant les ressources RDF.
org.apache.jena.rdf.model.Statement	Classe qui représente une déclaration RDF.
org.apache.jena.rdf.model.RDFWriter	Est une interface pour sérialiser un fichier RDF

Tableau 2: Les classes utilisées sur l'API Jena.

Afin d'exploiter ces classes, nous avons importé les APIs ci-dessous :

```
34  import net.rootdev.jenajung.examples.ToImage;
35  import org.apache.jena.query.QueryExecution;
36  import org.apache.jena.query.QueryExecutionFactory;
37  import org.apache.jena.query.QueryFactory;
38  import org.apache.jena.query.QuerySolution;
39  import org.apache.jena.query.ResultSet;
40  import org.apache.jena.rdf.model.Literal;
41  import org.apache.jena.rdf.model.Model;
42  import org.apache.jena.rdf.model.RDFNode;
43  import org.apache.jena.rdf.model.Statement;
44  import org.apache.jena.util.FileManager;
```

Figure 25 : les importes utilisés pour l'API Jena.

L'utilisation de la classe query est illustrée dans la figure ci-dessous :

```
326     org.apache.jena.query.Query query = QueryFactory.create(queryString);
        QueryExecution qescec = QueryExecutionFactory.create(query,Model);
328
```

Figure 26 : Utilisation de la classe query.

III.4. Jung :

Jung-2.0.2 (Java Universal Network Graph) : est une API java qui permet la modélisation, l'analyse, et la visualisation de données RDF [17]. Ces dernières peuvent être représentées sous forme de graphe ou de réseaux, et l'architecture globale de Jung permet de supporter une variété de représentations telles que, les graphes dirigés, ou non, ou bien des multi-modales, avec des bords parallèles, de même, il offre la possibilité d'annotées des graphes, des entités des relations avec des métadonnées, enfin, il fournit ainsi, des mécanismes pour

une exploitation interactive des données c'est-à-dire que l'utilisateur peut créer sa propre présentation personnalisée.

Nous avons regroupé ainsi les classes utilisées dans notre application avec leur signification dans notre application dans le tableau ci –dessous.

Classe	Utilisation
edu.uci.ics.jung.algorithms.layout.FRLayout	Algorithme pour assigner des coordonnées en deux dimensions aux sommets plus Précisément l'algorithme Fruchterman-Reingold.
edu.uci.ics.jung.graph.Graph	Interface pour les types graphe de Jung.
edu.uci.ics.jung.visualization.RenderContext	Mécanismes permettant de visualiser des graphes Jung en utilisant SWING/AWT.
edu.uci.ics.jung.visualization.VisualizationImageServer	Une classe qui pourra être utilisé sur le côté serveur.

Tableau 3 : Les classes utilisées dans l'API Jung.

La figure ci-dessous illustre les différents imports utilisés pour l'API Jung :

```
8  import edu.uci.ics.jung.algorithms.layout.FRLayout;
9  import edu.uci.ics.jung.algorithms.layout.Layout;
10 import edu.uci.ics.jung.graph.Graph;
11 import edu.uci.ics.jung.visualization.RenderContext;
12 import edu.uci.ics.jung.visualization.VisualizationImageServer;
```

Figure 27 : Liste des API Jung utilisés.

III.5. JenaJung :

JenaJung est une API java utilisées pour créer des graphes RDF avec l'API Jung en incorporant un modèle qui est la représentation d'un document RDF avec l'API Jena.

Le tableau ci- dessous, illustre la classe utilisée dans notre application avec sa signification.

Classe	Fonction
JenaJungGraph	Classe qui permet de créer un graphe Jung à partir d'un modèle Jena.

Tableau 4 : Les classes utilisées sur l'API JenaJung.

On montre dans le code ci-dessous l'utilisation d'une classe qui utilise les classes de l'API Jung pour représenter un document RDF sous forme de graphe, comme le montre la figure ci-

dessous :

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    String resource = path2;
    //Création Jena ressources
    Model model = FileManager.get().loadModel(resource);
    //Envoyer les ressources a la class de création de graphe
    Graph<RDFNode, Statement> g = new JenaJungGraph(model);

    Layout<RDFNode, Statement> layout = new FRLayout(g);

    int width = 1600;//dimension de l'image de graph
    int height = 800;
    //initialisation du graph dans l'image
    layout.setSize(new Dimension(width, height));
    VisualizationImageServer<RDFNode, Statement> viz =
            new VisualizationImageServer(layout, new Dimension(width, height));
    RenderContext context = viz.getRenderContext();
    //Envoyer les ressources a la class de identification des ressources
    context.setEdgeLabelTransformer(Transformers.EDGE);
    context.setVertexLabelTransformer(Transformers.NODE);

    //Création de arriére plan de l'image
    viz.setPreferredSize(new Dimension(width, height));
    Image img = viz.getImage(new Point(width/2, height/2), new Dimension(width, height));
    //placement du graphe dans l'arriére plan
    BufferedImage bi = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2d = bi.createGraphics();

    //Dessinez l'image
    g2d.setColor(Color.white);
    g2d.drawImage(img, 0, 0, width, height, Color.blue, null);
    try {
        //Enregistrez l'image dans un fichier
        ImageIO.write(bi, "PNG", new File("graph.png"));
        System.out.println("Image saved");
    } catch (IOException ex) {
        Logger.getLogger(ToImage.class.getName()).log(Level.SEVERE, null, ex);
    }
    //Impression de l'image dans JLabel
    ImageIcon imagee = new ImageIcon(bi);
    Image imgg = imagee.getImage();
    Image newimg = imgg.getScaledInstance(Graph.getWidth(),Graph.getHeight(),java.awt.Image.SCALE_SMOO
    ImageIcon finalImg = new ImageIcon(newimg);
    Graph.setIcon(finalImg);
}
```

Figure 28 : Fonctionnement du JenaJung (la transformation en graphe).

IV. Implémentation de MapReduce « Hadoop »

Hadoop est un système distribué qui répond aux problématiques de stockage et l'analyse des grandes quantités de données, il propose un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File System) et ce dernier offre la possibilité de stocker la donnée en la dupliquant. D'autre part, Hadoop fournit un système d'analyse des données appelé MapReduce [4]. Ce dernier s'exécute sur le système de fichiers HDFS pour réaliser des traitements sur des gros volumes de données

Pour des raisons pratiques nous avons choisi la distribution Hadoop de Cloudera CDH, et comme OS nous allons travailler avec linux Centos 6.2 [19] :

Les différentes étapes d'installation sont :

- Installer la machine virtuelle (VMware).
- Télécharger le Hadoop de distribution de CDH (Cloudera Distri-bution Hadoop). Cette distribution contient les principaux composants de Hadoop (HDFS, MapReduce, Hbase, Pig, Hive, Zookeper, Mahout, etc.), Centos 6.2, JDK, Eclipse.
- Télécharger le fichier de configuration de l'entrepôt de Cloudera.
- Copier le fichier cloudera-cdh4.repo dans les deux machines virtuelles Linux dans le chemin : /etc/yum.repos.d.
- Exécuter la commande : yum –nogpgchecklocalinstall cloudera-cdh-5-0.x86_64.rpm.
- Configuration du Cluster Hadoop, on va utiliser un Cluster de deux machines virtuelles, un Master (Namenode, Datanode, JobTracker) et une machine Slave (Datanodes, Tasktrackers).

IV.1. Variables de configuration de Hadoop :

Ces variables concernent les objets manipulés par HDFS et le MapReduce, Ces variables sont stockées dans des fichiers d'extension XML généralement dans le chemin linux : /etc/hadoop/conf/, les principaux fichiers de configuration de l'environnement Hadoop sont :

core-site.xml : contient les paramètres de configuration de base pour Hadoop, tels que le HDFS URL, le Hadoop répertoire temporaire, les paramètres d'E/S qui sont commun à HDFS est MapReduce.

hdfs-site.xml : contient les paramètres de configuration pour les processus HDFS : le nom du Namenode, le secondaire Namenode, et les noms des Datanodes, aussi le nombre de réplication et la taille d'un bloc qui détermine généralement le nombre de Datanodes, ainsi que d'autres paramètres.

mapred-site.xml : contient les paramètres de configuration pour les jobs MapReduce, l'adresse du Jobtracker et l'adresse du Tasktrackers.

- Ces paramètres de configuration sont lus par les Framework java de HDFS et MapReduce. Les paramètres du fichier mapred-site.xml :

Paramètre	Valeur	Exemple
mapred.job.tracker	Host ou IP et port du JobTracker.	master : 2598
mapred.system.dir	Chemin sur le HDFS où Mapreduce stocke les fichiers système.	Doit être accessible depuis le serveur et les machines clientes.
mapred.local.dir	Liste de chemins séparés par des virgules, sur le système de fichiers local où les données de MapReduce seront temporairement écrites.	
mapred.tasktracker.map |reduce.tasks.maximum	Le nombre maximum de tâches MapReduce, qui est exécutées simultanément sur un Tasktracker donné.	Par défaut, 2 (2 cartes et 2 réduit), mais il varie en fonction de votre matériel.
dfs.hosts/dfs.hosts.exclude	Liste des Datanodes admis et exclus.	
mapred.hosts/mapred .hosts.exclude	Liste des Tasktrackers admis et exclus.	

Tableau 5 : Les paramètres du fichier mapred-site.xml

- Paramètres du fichier core-site.xml sont :

Paramètre	Valeur	Exemple
fs.default.name	URI du Namenode	hdfs : //master/

Tableau 6 : Paramètres du fichier core-site.xml.

- Paramètres du fichier Hdfs-site.xml :

Paramètre	Valeur	Exemple
dfs.name.dir	Chemin sur lequel le Namenode stocke les journaux de transactions.	Liste de répertoires qui peuvent être séparés par des virgules pour la redondance. rep1, rep2, rep3.
dfs.data.dir	Chemin local sur lequel des Datanodes stockent leur bloc.	Liste de répertoires. Ces derniers peuvent être séparés par des virgules pour la redondance (par exemple : rep1, rep2, rep3,. . .).
dfs.replication	Nombre de réplications du bloc sur les Datanodes.	

Tableau 7 : Paramètres du fichier Hdfs-site.xml.

IV.2. Configuration d'un Cluster Hadoop :

- Pour vérifier la version java, nous utilisons les commandes java –version.

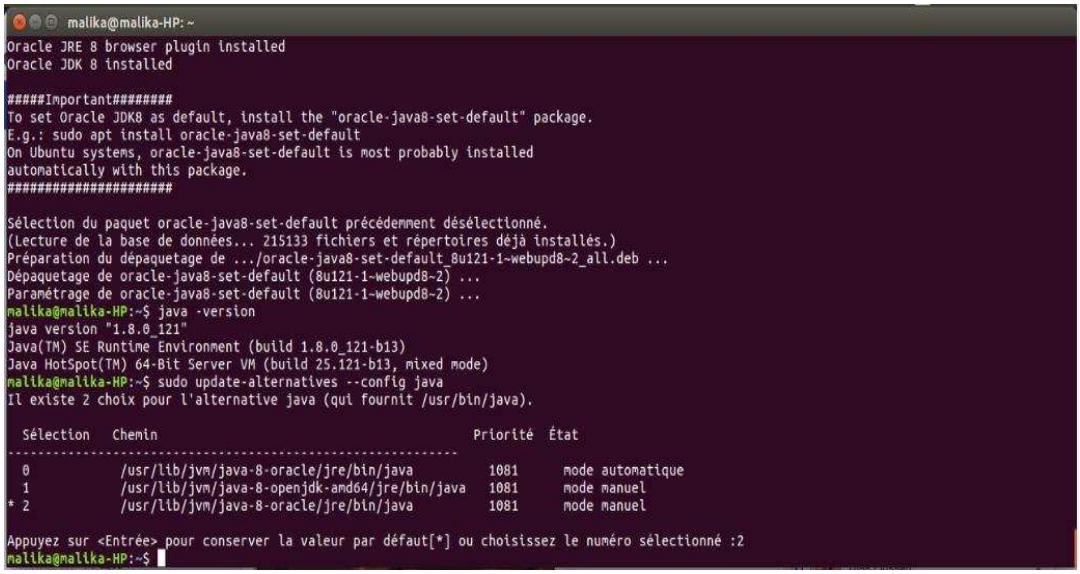


Figure 29 : Vérifier de la version java.

- Hadoop nécessite un accès SSH pour gérer les différents nœuds. Bien que nous soyons dans une configuration simple nœud, nous avons besoin de configurer l'accès vers localhost pour l'utilisateur crée.

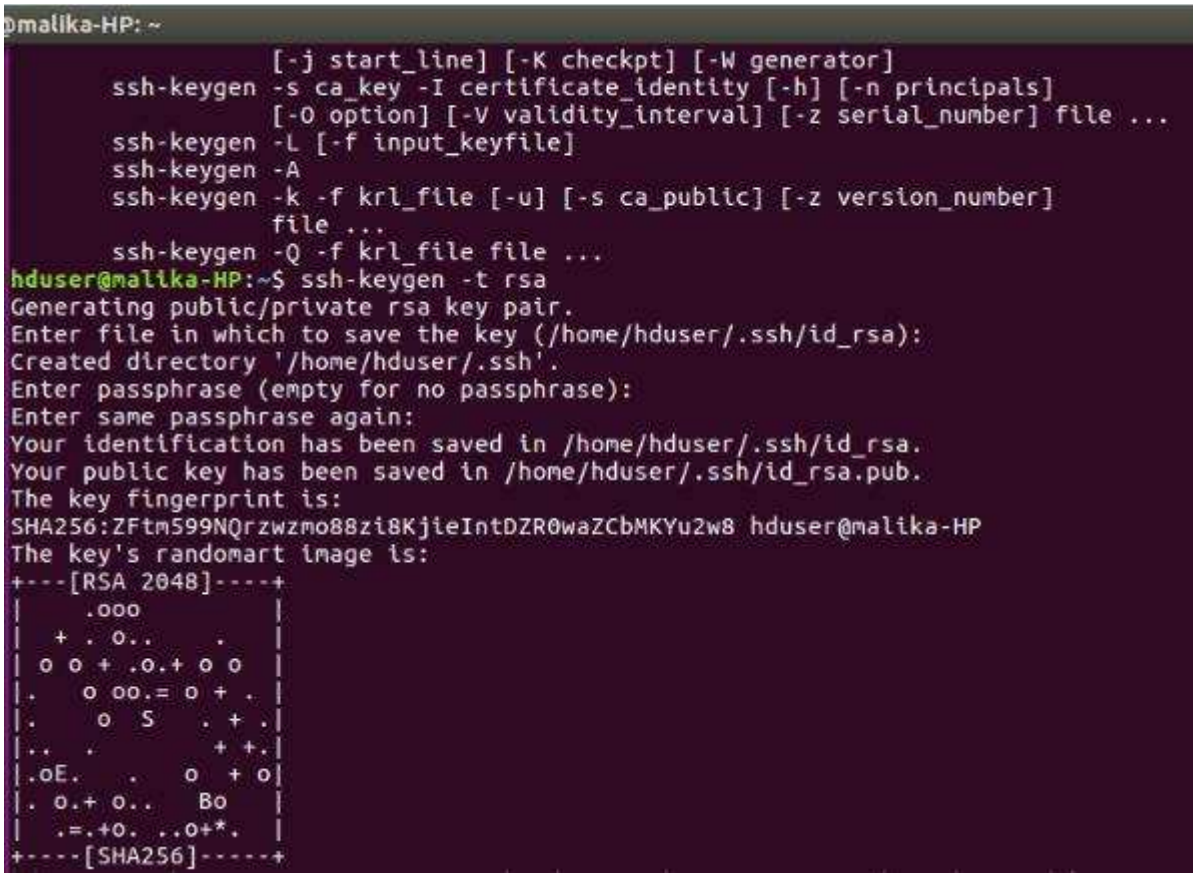


Figure 30 : Génération de la clé publique RSA.

Dans la Figure la commande : ssh-keygen -t rsa va créer une clé RSA avec un mot de passe vide. En cas de virtualisation, l'absence de mot de passe n'a pas d'importance, le résultat est sauvegardé dans le répertoire de l'HDFS, qui est montré par un rectangle dans la Figure (30).

❖ Pour démarrer les services d'Hadoop nous utilisons les commandes suivantes :

- ./hadoop-daemon.sh start datanode.
- ./hadoop-daemon.sh start namenode.
- ./hadoop-daemon.sh start tasktracker.
- ./hadoop-daemon.sh start mapred.

V. Développement des programmes MapReduce :

Pour développement des jobs MapReduce, nous avons notamment les packages plus important : package org.apache.hadoop.io, Package org.apache.hadoop.fs, et Package org.apache.mapred définit. Nous allons faire une brève description des principales classes de ces packages.

- **Package org.apache.hadoop.io:**

Ce package est utilisé lors de la lecture et de l'écriture des données sur le réseau, bases de données, et les fichiers, les principales classes de ce package sont [20] :

Classe	Usage de la classe
IntWritable	Pour la lecture et écriture d'un entier.
LongWritable	Pour la lecture et écriture d'un long donnée.

Tableau 8 : Classes org.apache.hadoop.io.

- **Package org.apache.hadoop.fs:**

Ce package gère les systèmes de fichiers (notamment HDFS) parmi ses principales classes [20] :

Classe	Usage de la classe
Path	Spécifie les noms des fichiers ou de répertoires dans un système de fichiers.
FSDataInputStream	Utiliser en général pour la lecture, il encapsule les données, et fait le lien entre le Namenode et les Datanodes.
FSDataOutputStream	Utiliser lors de l'écriture d'un fichier, il se charge de la coordination entre le Namenode et les Datanodes lors de l'écriture.

Tableau 9 : Classes org.apache.hadoop.fs.

- **Package org.apache.mapred :**

Il est utilisé pour l'implémentation du MapReduce, parmi ses principales classes [20] :

Classe	Usage de la classe
InputFormat<K,V>	Input Format décrit l'entrée d'un job MapReduce (keys, value)
Mapper<K1,V1,K2,V2>	Implémente la fonction Map (), reçoit une liste de clé value en input et génère une autre liste de clé value.
Reducer<K2,V2,K2,V3>	Implémente la fonction Reduce (), reçoit une liste de clé value en input (k2, v2) et génère une liste réduite (k2, v3).

Tableau 10 : Classes Package org.apache.mapred.

VI. Description du travail réalisé :

Cette partie est consacrée à la description de la phase de la réalisation et d'implémentation de ce projet. On va commencer par la présentation de notre application par l'interface d'accueil.

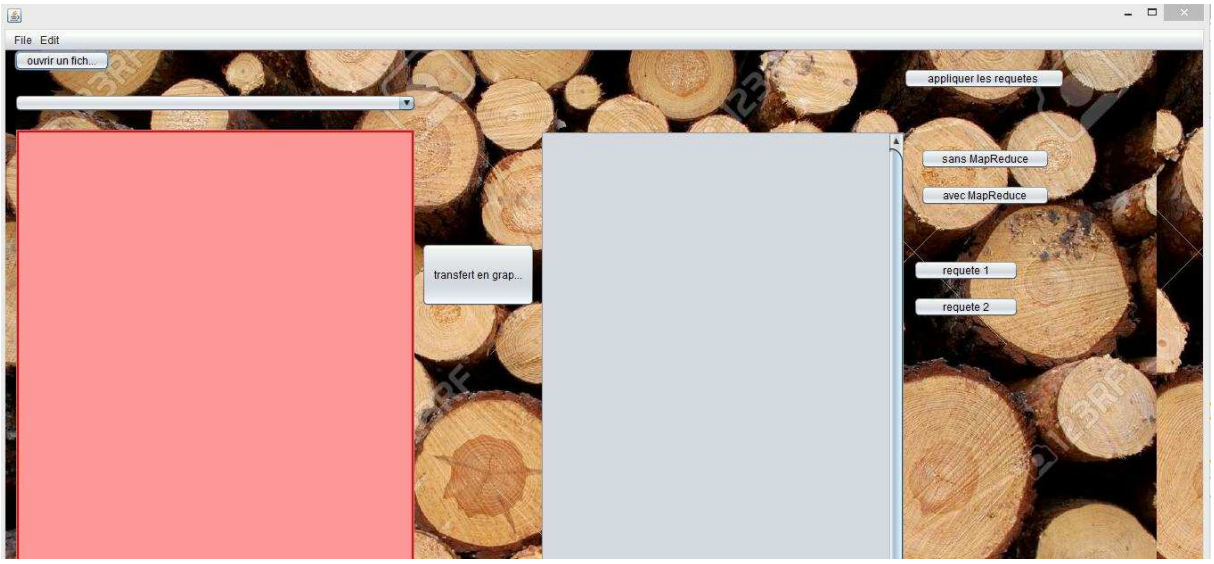


Figure 31 : Fenêtre principale.

- Tous d'abord il faux ouvrirune image en cliquant sur le boutons « ouvrer ».

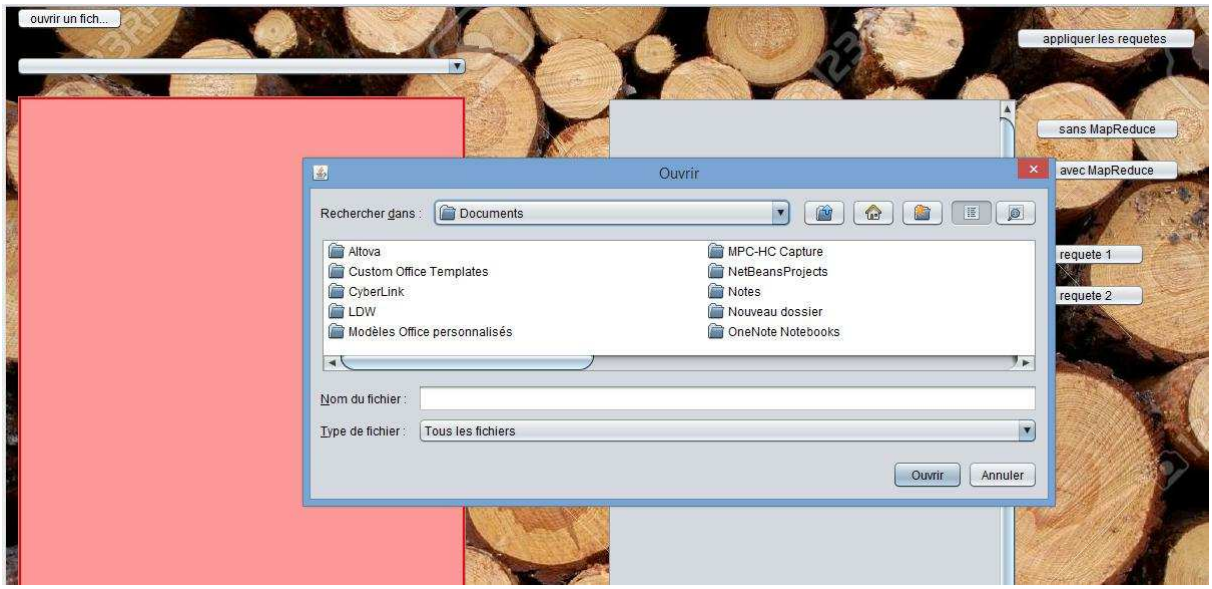


Figure 32 : L'exécution du bouton ouvrir.

- On va sélectionner un fichier RDF pour avoir son passage au fichier XML comme il est illustré dans la figure ci-dessous :

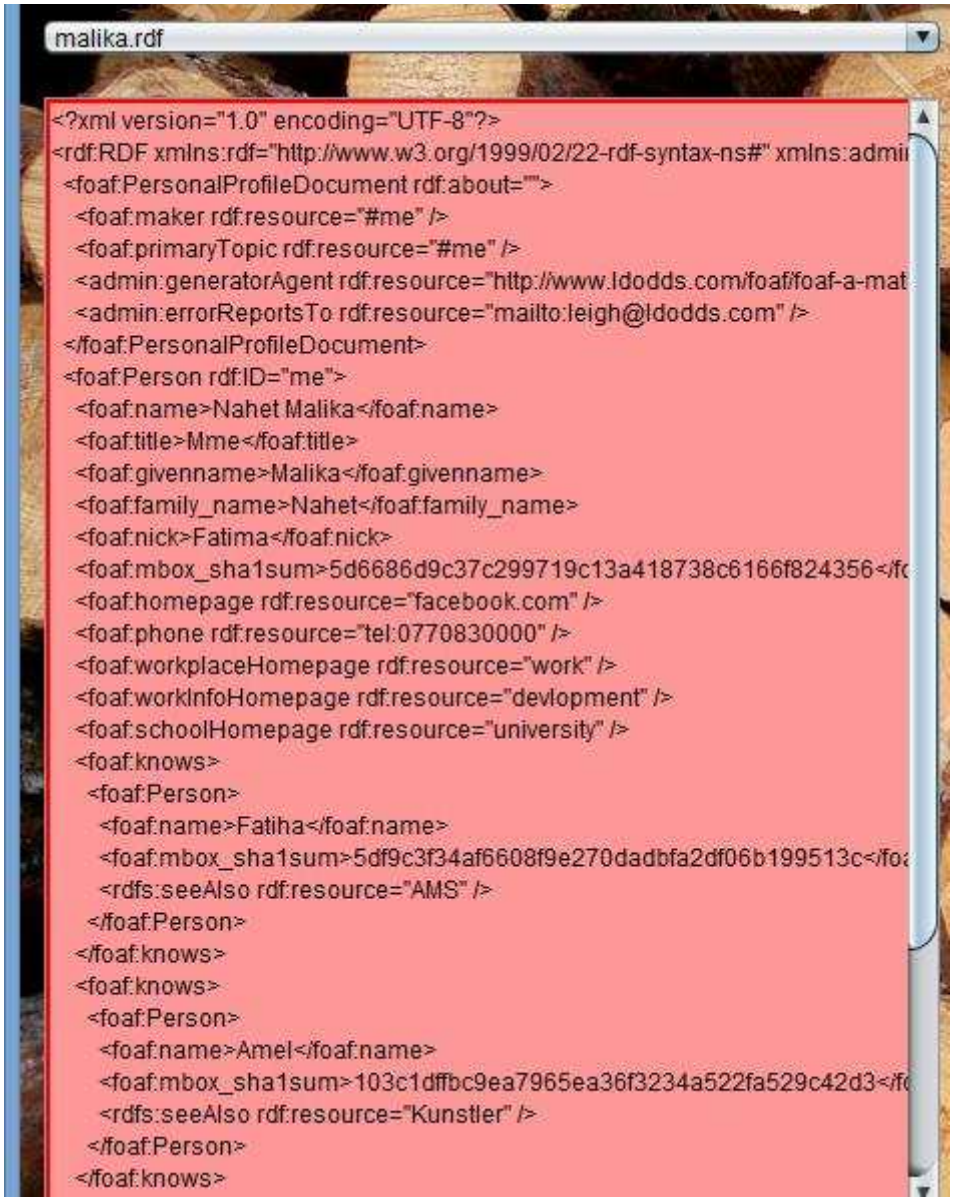


Figure 33 : Ouverture du fichier RDF.

- Après avoir choisir un fichier, on va cliquer sur le bouton transfert en graphe pour le transformer en graphe.

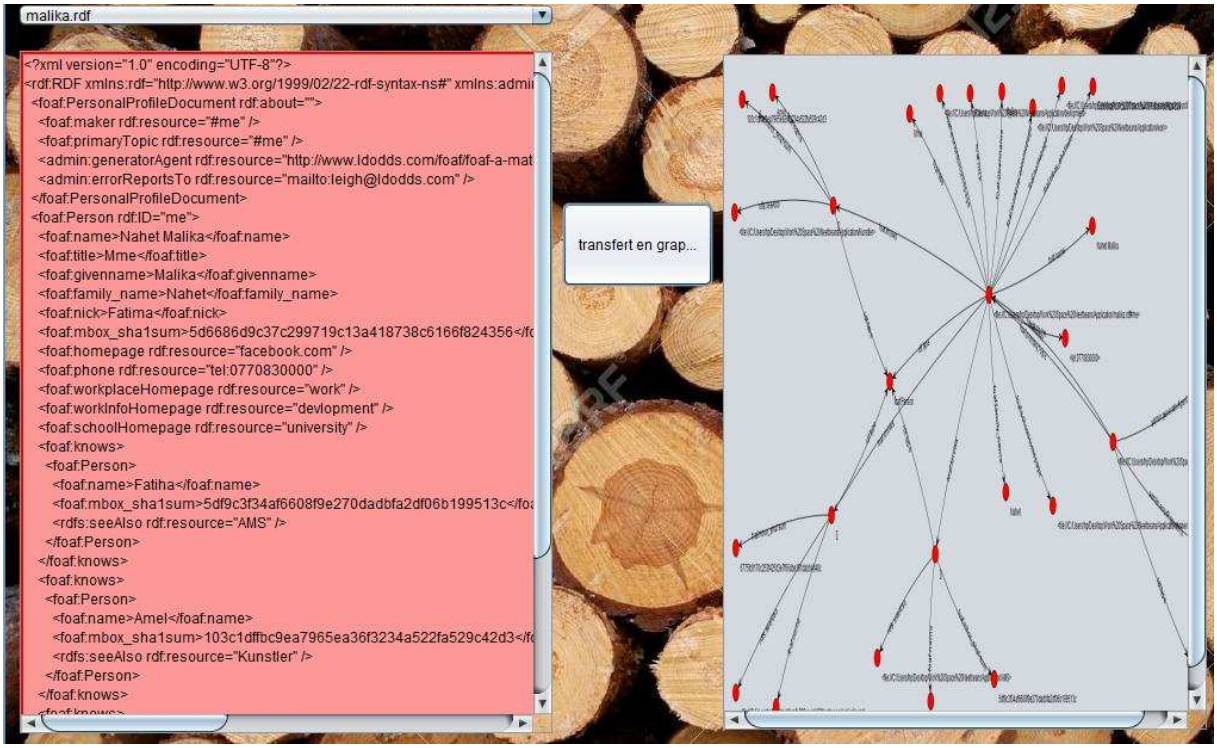


Figure 34 : La transformation du fichier en graphe.

- La figure ci-dessous illustre l'exécution de la requête 1 sans MapReduce en appuyant sur le bouton « sans MapReduce » puis sur le bouton « requête 1 » tell que le temps d'exécution ici c'est 0.141s.

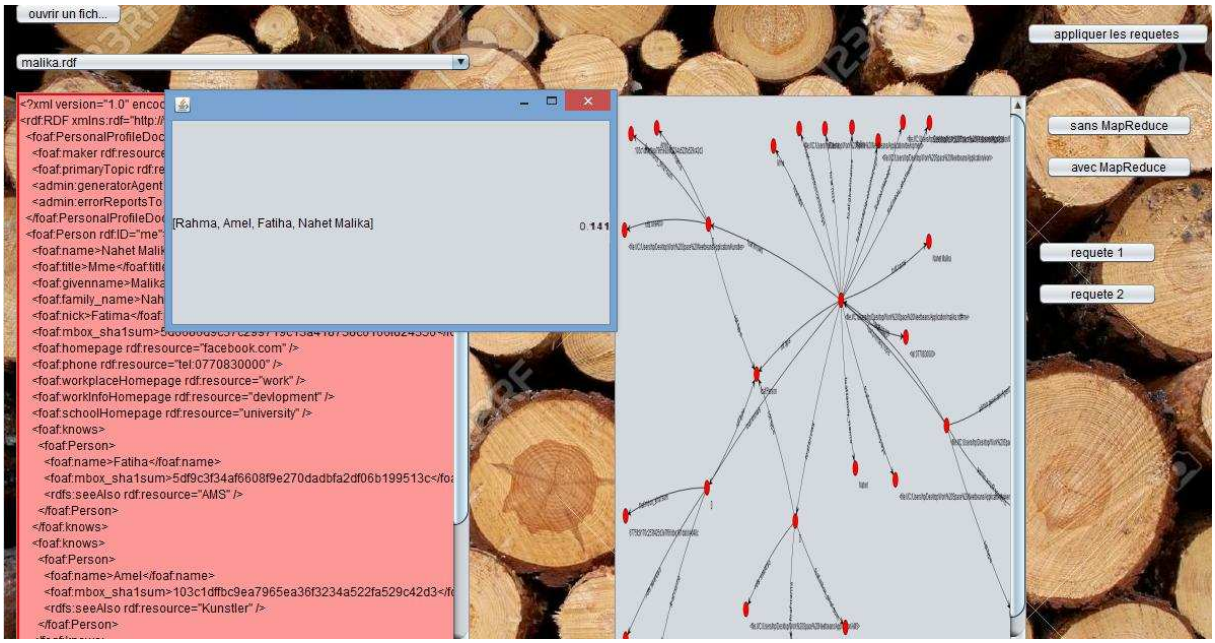


Figure 35 : L'exécution de la requête sans MapReduce.

- La figure ci-dessous illustre l'exécution de la requête 1 avec MapReduce en appuyant sur le bouton « avec MapReduce » puis sur le bouton « requête 1 » tell que le temps d'exécution ici c'est 0.003s.

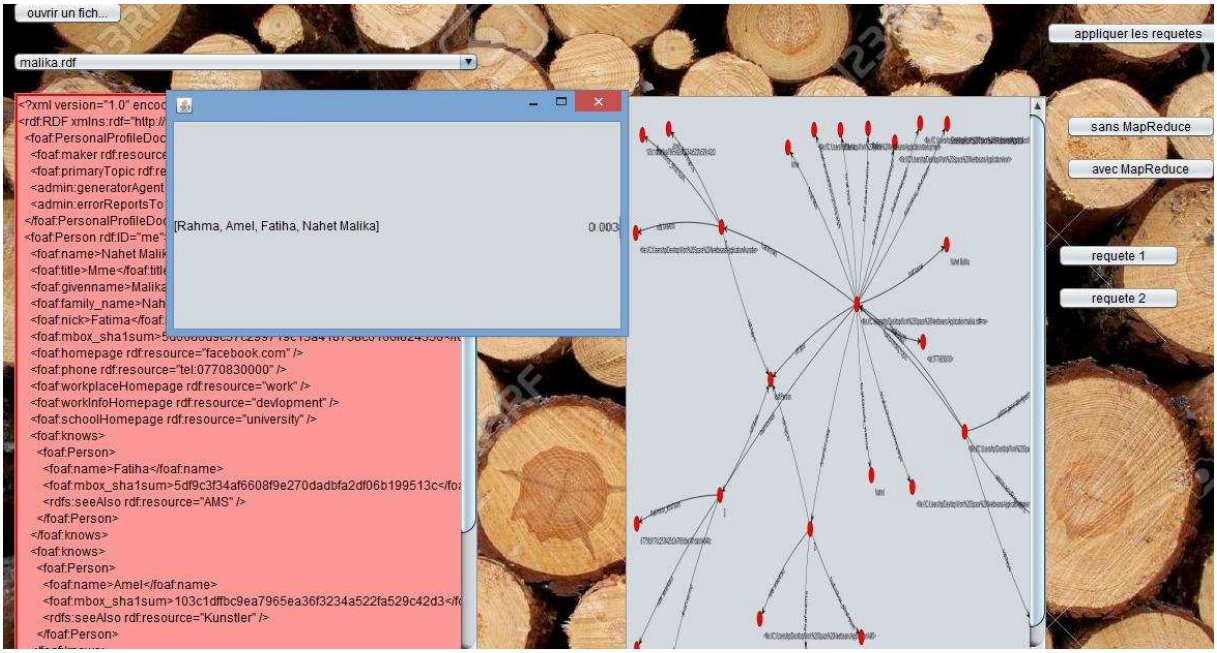


Figure 36: L'exécution de la requête avec MapReduce.

VII. Discussion :

- Les résultats obtenus nous permettons de déduire que le paradigme MapReduce nous aide de faire des traitements sur des données RDF avec un temps très réduit par rapport à l'exécution des traitements sans l'utilisation du MapReduce.
- Nous pouvons dire que le modèle MapReduce est un paradigme de programmation très fiable.

VIII. Conclusion :

Nous avons présenté dans ce chapitre, les outils nécessaires pour l'implémentation de notre modèle qui permet le traitement des données sémantique.

On se basant sur les résultats des tests donnés par l'application on peut dire que la méthode de partitionnement de requête avec le modèle MapReduce donne des résultats fiables par rapport à l'exécution d'une requête sans partitionnement, c'est-à-dire, sans l'utilisation du MapReduce et cette conclusion est basée sur le temps d'exécution obtenu pendant la réalisation de notre application.

De nos jours, le web contient une immense quantité de données, cependant, les utilisateurs ont constamment besoin de récupérer ces données afin de les utiliser et les traiter. Pour ce faire, les chercheurs et les développeurs ont proposé le web sémantique qui a pour but de rendre les machines intelligentes afin d'aide l'utilisateur à trouver ces données et de faire toutes sortes de traitement sans difficulté. Le principe de cette solution est d'intégrer le concept du RDF « Ressource Description Frame work » pour rendre les données plus compréhensibles par la machine c'est-à-dire obtenir des données sur d'autres données (qui forme le concept de connaissance, cet ajout du sens aux données augmente la taille de ces derniers et la rendre énormément massive.

Parmi les différentes représentations formelles des données sémantiques RDF (Ressource Description Framework), RDFs (Ressource Description Framework schema), et les ontologies. Ces standards sont basés sur l'XML (Extensible Markup Language). Comme elles peuvent être représenté sous forme graphique.

Afin de stocker les données du web sémantique, les développeurs ont opté pour les bases NoSQL et les BigData car elles sont plus appropriées aux formats et au volume de ce nouveau type de donnée. Ces bases de données supporte la taille volumineuse des données et optimise l'accès aux données sémantiques.

Dans le but de traiter les données sémantiques, le langage d'interrogation SPARQL (SPARQL Protocol and RDF Query Language). Ce dernier est utilisé en particulièrement pour les documents RDF. Ainsi, l'évolution et le changement de l'outil de stockage facilite le traitement des connaissances avec SPARQL. Cependant, il n'est pas facile de faire le traitement sur une grande quantité de données dans un temps raisonnable et d'une manière fiable. Pour cela, il est préférable de faire la transformation des données stockées sous forme RDF vers des données sous forme RDF graphe pour faciliter la recherche et le traitement.

Ainsi, pour améliorer le traitement volumineux des données, le célèbre Google a inventé le modèle de programmation parallèle MapReduce. Ce dernier, est un paradigme qui s'appuie sur le partage de charge et il sert à réaliser l'équilibrage pendant le traitement de données à fin de réduire le temps d'exécution.

Dans notre travail, nous avons appliqué MapReduce sur des données sémantiques représentées par les graphes RDF afin de traiter les requêtes SPARQL. Le principe est de subdiviser la requête en sous-requêtes pour traiter en parallèle dans le but d'améliorer le temps d'exécution de la requête SPARQL.

Suite à ce travail, nous avons trouvé que l'utilisation de MapReduce pour le traitement des données et la forme graphique pour la représentation des données permet de réduire le temps d'exécution.

Perspectives

Les perspectives liées à ce travail consistent à améliorer les résultats en termes de temps d'exécution de la requête en adoptant des techniques de partitionnement de graphe. Il est possible de partitionner le graphe en sous-graphes en plus de la subdivision de la requête en sous-requêtes. Ensuite, il est possible de lancer les sous-requête au niveau de tous les sous-graphes, afin d'obtenir un temps d'exécution plus réduit.

Bibliographie

[1] W3C, https://www.w3.org/2004/01/sws-pressrelease.html.fr, 10/02/2004, consulté le 22/12/2016.

[2] W3C, https://www.w3.org/2007/03/VLDB/, 20/04/2007, consulté le 22/12/2016.

[3] Tutoriel d'introduction à Apache Hadoop, http://mbaron.developpez.com/tutoriels/bigdata/hadoop/introduction-hdfs-map-reduce/, consulté le 23/12/2016.

[4] haboop, https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#Sample+Runs, le 08/04/2013, consulté le 29/03/2017.

[5] C.Ferrand, Identité Numérique, Stage de Familiarisation Ent Bureau d'Aide à l'Insertion Professionnelle, http://www.univ bpclermont.fr/Ressources_Num/Les_reseaux_sociaux_web_web/co/Les_reseaux_sociaux_ web.html, consulté le : 23/12/2016.

[6] C. Roux, L'histoire et l'avenir du web, Document web, 2008.

[7] Agence Marketing Digital des PME, https://c-marketing.eu/du-web-1-0-au-web-4-0/, consulté le: 22/12/2016.

[8] L. Quoniam et C.V. Boutet, Web 2.0, La révolution connectique, rapport, Université de Toulon, institut Ingémedia, 2008.

[9] Introduction au web sémantique, club des développeurs et IT pro, http://jplu.developpez.com/tutoriels/web-semantique/introduction/ consulté le:24/12/2016.

[10] S. Antipolis, Introduction au web sémantique, Projet Endelweiss, consulté le : 24/12/2016.

[11] P. Laublet, Ch. Reynaud, J. Charlet, Sur quelques aspects du Web sémantique, Mission de recherche en sciences et technologies de l'information médicale, Université de Paris-Sorbonne, Université Paris-Sud.

[12] J. Charlet, P. Laublet, Ch. Reynaud, Web sémantique, Rapport final, université Laval, Québec, 2003.

Bibliographie

[16] W3C, https://www.w3.org/2001/sw/wiki/RDFStore, consulté le 01/03/2017.

[17] Cyrille Herby, Apprenez à programmer en Java [livre], www.siteduzero.com, 03/2010.

[18] Oracle, https://www.oracle.com/sun/index.html?PHPSESSID=8fd76773d26875481e097a4a27c7a6a1 Internet, consulté le 14/04/2017.

[19] M.R. Khatir, Mapmening, Mémoire Master, Université d'Oran 1, 2015.

[20] Apache Software Foundation, https://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/mapreduce/package-summary.html, consulté le 20/04/2017.