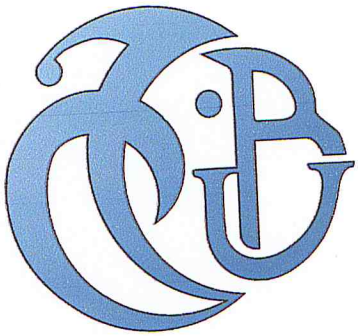
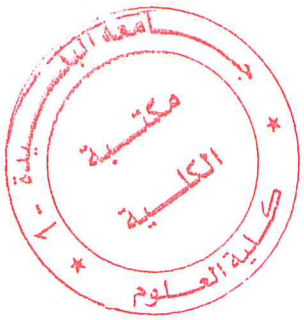
**République Algérienne démocratique et populaire**

**Ministère de l'enseignement supérieur et de la recherche scientifique**

**Université Saad Dahleb Blida**






**Faculté des sciences**

Département d'Informatique

**Mémoire de fin d'études**

Pour l'obtention du diplôme MASTER en informatique

**Option** : Génie Systèmes Informatique.

# Analyse en ligne de cubes de textes basée sur un modèle NoSql

**Encadré par :**

promotrice : Mme L.Oukid
Co-promotrice : Mme N.Benblidia

**Elaboré par :**

Hamoudi Abderraouf

**Présenté devant le jury composé de :**

Mr AH.Kameche

Mme FZ.Zahra

2015/2016

**Résumé**

Les données textuelles dans les entreprises et sur le web représentent une grande masse de données de plus en plus volumineuse. Dans ce cadre, nous présentons dans ce mémoire, la mise au point d ?un système d ?analyse en ligne de données textuelles (Text OLAP) dans un cadre de grand volumes de données. Cela, en se basant sur le modèle de cube de textes CXT-Cube [16] et un modèle NoSql (No Only SQL). A cet effet, nous allons exposer à travers ce travail les différentes étapes suivies pour le développement du système ; partant du nettoyage des données textuelles, la représentation des documents et le chargement vers une base de données NoSQL mise en ?uvre au sommet de trois technologies co-habitantes au niveau d ?un cluster Hadoop permettant ainsi l'interrogation des millions de documents.

**Mots clés :**

OLAP texte, cube de textes, recherche d'information , Base de données NoSql, Big Data.

**Abstract**

Text data in business and on the web represent a large mass of data more and more voluminous. In this context, we present in this docuement, the development of an online analysis system for text data (OLAP Text) in a context of large data volumes. This is based on the CXT-Cube[16] which is a text cube model and a NoSql (No Only SQL) model. To this end, we will expose through this work the different steps followed for the development of the system ; From the cleaning of textual data, the representation of documents and the loading to a NoSQL database implemented on top of three technologies running at a Hadoop cluster thus allowing the interrogation of millions of documents.

**Keywords :**

OLAP Text, text cube, information retreival , NoSql database, Big Data.

**Remerciment**

nous tenons à remercier en premier lieu Mme L.Oukid et Mme N.Benblidia leur encadrement ainsi que pour les nombreuses discussions que nous avons pu avoir. Nous tenons à exprimer nos sincères remerciements à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études. On n ?oublie pas nos parents pour leur contribution, leur soutien et leur patience. Enfin, on remercie tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

# introduction

La mondialisation économique a imposé aux organisations un nouveau rythme de croissance accéléré, ainsi que la survie de l'entreprise est liée fortement à sa performance et sa tolérance d'adaptation de changements. Une organisation moderne est celle qui recherche toujours l'innovation qui lui permettra de devancer la concurrence ; s'outiller d'éléments requis aux nouvelles exigences est la préoccupation des décideurs qui utilisent impérativement une nouvelle vague d'application métier dédiés aux supports d'aide à la décision. Un système d'aide à la décision est un système interactif qui aide les décideurs à dégager des informations pertinentes d'un ensemble de données brutes stockées dans l'entrepôt de données de l'entreprise, des traitements effectués par ces outils permettent d'analyser les tendances afin de prédire l'avenir et prévenir l'organisation des changements[1]. Les systèmes d'entreposage de données et d'analyse en ligne (OLAP) classiques ont fait leurs preuves pour analyser des données numériques qui restent moins complexes ; mais ne satisfait pas le rythme de croissance des différents systèmes liés à l'activité de l'entreprise qui tourne vers l'analyse de données textuelles exigeant de grandes ressources de calculs et stockage. Une nouvelle ère , le BigData ; des données structurées ou non, constituées essentiellement de données textuelles dont le très grand volume requiert des outils d'analyse multidisciplinaires, multi-physique, multicritère et multi-échelle, qu'ils doivent se baser sur des approches rigoureuses qui passe d'une simple amélioration à une révolution.
Afin de répondre à ce besoin, l'adaptation au concept émergeant l'informatique de nuage '*Cloud computing*' est impérativement nécessaire : un concept permettant le bénéfice de la diversité de services d'un système réparti sur le réseau mondial et qui assure l'efficacité opérationnelle. C'est dans ce cadre que s'inscrit notre projet qui consiste à développer une plate-forme d'analyse en ligne de données textuelles. Cette dernière doit se baser sur un modèle de cube de textes facilitant le bénéfice d'avantages offert par les navigations multidimensionnelles des systèmes OLAP et les méthodes de la recherche d'information en utilisant une base de données NoSQL.

**Plan de mémoire :**

Les trois premiers chapitres représentent l'état de l'art sur les systèmes d'entreposage et OLAP clasiques, le modèle adopté et les bases de données NoSQL. le chapitre suivant regroupe la conception et l'architecture générale du système réalisé et qui sera suivit par la phase des implémentations. Les chapitres sont de l'ordre suivant :

- Les entrepôts de données et les systèmes OLAP.
- La recherche d'information et le modèle CXT-Cube.
- Les bases de données NoSQL et le framework Hadoop.
- Conception et architecture générale.
- Réalisation d'un cluster R/Hadoop/Kylin et implémentation de l'ETL utilisant R avec une connexion JDBC/ODBC pour les requêtes où par l'usage des UDF avec RHive.

# Table des matières

# Table des figures

Chapitre I

# Les entrepôts de données et les systèmes OLAP

# 1 Les entrepôts de données et les systèmes OLAP

Les systèmes d'aide à la décision rendent l'information interprétable par les décideurs et facilitent le chemin vers les choix stratégiques. Un système d'aide à la décision a besoin d'être alimenté par une énorme masse de données hétérogènes variante dans le temps et non-volatile. Des critères qui ne sont pas offerts par les bases de données traditionnelles, celles qui subissent des mises à jour fréquentes et qui ne permettent pas d'effectuer une analyse d'un point de vue historique. Ce besoin technologique a forcé la naissance des entrepôts de données. Une infrastructure solide qui a le pouvoir de transformer les données en perpétuelle croissance vers une sources de vision stratégique, et qui offre des mécanismes pour faciliter le développement des systèmes décisionnels.

## 1.1 les entrepôts de données

### 1.1.1 Définition

Un entrepôt de données (Data Warehouse) est un espace centralisé qui regroupe des données pertinentes pour le décisionnel, où elles sont stockées de manière homogène, historiées et agrégées suivant un modèle qui assure une gestion efficace sur les plans de cohérence , validité et fraicheur de données. les données consolidées aide les exécuteurs de l'organisation à analyser ses activités d'un points de vue historique et facilite les intégrations dans les systèmes d'application[2].
la figure suivante schématise et montre les éléments d'un système d'aide à la décision.

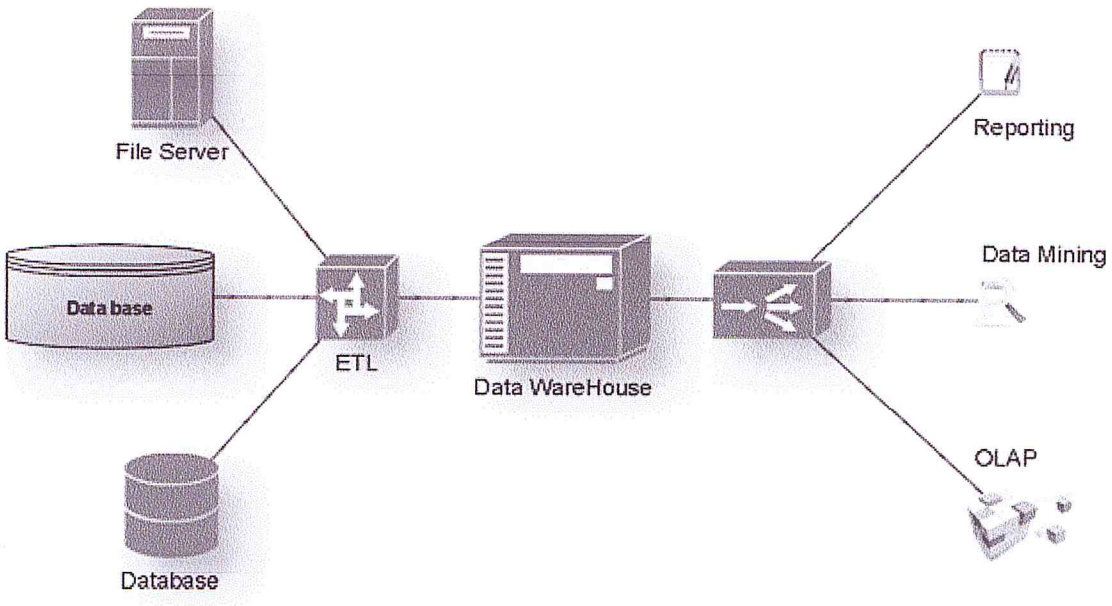



FIGURE 1 – Architecture d'un système décisionnel.

La figure-1 donne une architecture générale d'un système décisionnel; comme nous pouvons le voir l'entrepôt de données (*Data WareHouse*) est alimenté par les différentes sources de données hétérogènes via le processus d'alimentation ('E.T.L' qui sera détaillé prochainement) facilitant l'application des traitements analytiques et l'usage des outils de visualisation.
la conception d'un entrepôts de données ou d'un système décisionnel sera confronté a deux méthodes celle de Inmon Bill connu sous le nom du père de data warehouse et la deuxième de Ralph Kimball qui est la plus populaire[37].

### 1.1.2 Mesures de séparation

Les entrepôts de données sont maintenus séparés de la base de données opérationnelle car cette dernière est construite pour des traitements simultanés de plusieurs opérations nécessitant l'emploi de mécanismes de contrôle et de récupération pour assurer la robustesse et la cohérence au niveau opérationnel. Les requêtes d'une base de données opérationnelle s'effectuent en mode lecture/écriture et ne conservent que les données actives. En revanche les requêtes d'un entrepôt de données s'effectuent en mode lecture seule, sont souvent complexes et présentent une forme générale de données archivées. le tableau suivant met en vue les caractéristiques essentielles qui différencient l'entrepôt de données d'une base de données relationnelle.

| Les entrepôts de données | Les bases de données opérationnelles |
|---|---|
| Traitement de l'information historique | Traitement de jour en jour |
| utilisées par les travailleurs du savoir tels que les cadres, les gestionnaires et les analystes. | utilisés par les secrétaires, les DBA, ou des professionnels de base de données. |
| sont utilisés pour analyser l'activité. | sont utilisé pour gérer l'entreprise. |
| basé sur plusieurs modèles. | sont basés sur le modèle entité-relation. |
| contiennent des données historiques. | contiennent des données actuelles. |
| fournissent des résumés et des données consolidées. | fournissent des données primitives et très détaillées. |
| fournissent des synthèses d'une vue multidimensionnelle des données. | fournissent une vue relationnel détaillée de données. |
| le nombre d'utilisateurs est en centaines. | le nombre d'utilisateurs est en milliers. |
| leurs volumes de données est de 100 Go à 100 To. | sont d'une taille de 100 Mo à 100 Go. |
| sont très flexibles. | fournissent de hautes performances. |

FIGURE 2 – comparaison entre la base de données opérationnelle et l'entrepot de données.[2]

### 1.1.3 Propriétés d'un Data Warehouse

les entrepôts de données sont caractérisés par les propriétés suivantes (selon Inmon B)[37] :

- Orienté sujet : Un entrepôt de données est orienté sujet car il fournit des informations autours d'un sujet plutôt que les opérations en cours de l'organisation ; Ces sujets peuvent êtres des produits, clients, fournisseurs, ventes, revenues et d'autres sujets de traitement.
- Intégré : Un entrepôt se concentre sur la modélisation et l'analyse de données intégrées provenant de sources hétérogènes (telles que les bases de données relationnelles et autres ressources.) ces intégrations renforces l'analyse et la rendent efficace pour la prise de décisions.
- Variant dans le temps : Les données recueillies dans un entrepôt de

données sont identifiées avec une période de temps et fournissent des informations à partir d'un point de vue historique.
— Non volatil : Signifie que les données ne sont pas effacées lors d'un processus d'alimentation.

### 1.1.4 Les processus d'alimentation(E.T.L)

le processus ETL (*Extract, Transform, Load*) est composé des phases suivantes [2] :
— L'extraction de données : c'est la collecte de données provenant de multiples sources hétérogènes.
— Le nettoyage de données : implique de trouver et corriger les erreurs dans les données.
— Le chargement de données : consiste à trier, résumer, consolider, vérifier l'intégrité et mettre des indices de construction pour les parties chargées et un rafraichissement périodique qui implique la mise à jour à partir de sources de données à l'entrepôt.

### 1.1.5 Types de traitements dans un entrepôt de données

Le traitement statistique, analytique, et l'exploration de données sont les trois utilités majeurs d'un entrepôt de données[2].
— Traitement statistique : Les données peuvent être traitées au moyen : d'interrogation, d'une analyse statistique de base et/où par des rapports à l'aide des tableaux croisés et des graphiques de visualisation.
— Traitement analytique :Un entrepôt de données prend en charge le traitement analytique de l'information stockée où les données peuvent être analysées au moyen d'opérations OLAP.
— Data Mining : L'exploration de données soutient la découverte et l'extraction de connaissances en trouvant des motifs cachés et des associations, constructions de modèles d'analyse, classification et la prédiction de scènes ; les résultats d'exploration peuvent être présentés à l'aide d'outils de visualisation.

### 1.1.6 Data Mart(magasin de données)

Les Data Marts sont des groupements (dépôts) de données, ou chacun contient un sous-ensemble des données de l'organisation spécifiques à des groupes/personnes, En d'autres termes un magasin de données ne contient que les données spécifiques à un groupe particulier et qui sont organisées suivant un modèle facilitant l'intégration et l'analyse. Par exemple le data Mart marketing peut contenir que des données relatives aux articles, clients

et ventes ; ce dépôt est confiné seulement à ce sujet. Ces dépôts sont de petite taille, personnalisés, structurés et entreposés par département ; le data Mart est aussi flexible et sa mise en œuvre au niveau d'un serveur n'est pas couteuse, où cette implémentation est mesuré dans un cycle court de temps (quelques semaines au lieu des mois et des années). Un magasin de données ayant une organisation multidimensionnelle est nommé 'BDM' base de données multidimensionnelle[37].

## 1.2 Les schémas

Les entrepôts de données reposent sur des modèles de représentation des données stockées, ces modèles correspondent à la représentation multidimensionnelle et qui incluent les dimensions et les faits, les types de données ainsi que les agrégations. Cette structure essentiel peut être établie selon les modèles suivants[2] :

### 1.2.1 Schéma en étoile(*Star Schema*)

la structure en étoile est représentée par le fait central est les dimensions. chaque dimension est représentée par une table où chaque table contient l'ensemble d'attribut de tous les niveaux d'agrégation. Dans ce modèle des redondances peuvent apparaitre durant les traitements.

### 1.2.2 Flocon de neige(*Snowflake Schema*)

contrairement au premier modèle les tables de dimension dans la modélisation en flocon, peuvent être normalisées suivant la hiérarchie. La normalisation permet d'éclater une table afin de représenter chaque niveau hiérarchique, ce qui élimine la redondance. Ce modèle est facile à maintenir et il offre un accès rapide avec un lent traitement.

### 1.2.3 Schéma en Constellation(*Constellation Schema*)

Ce modèle est basé sur la décentralisation.Il est composé de plusieurs tables de fait qui peuvent partager les mêmes tables de dimension.

## 1.3 Le dépôt de Méta-données

Les Métadonnées sont Les données utilisées pour représenter d'autres données. Par exemple, l'index d'un livre sert de métadonnées pour le contenu du livre. En d'autres termes, nous pouvons dire que les métadonnées sont le groupe de données résumées qui nous conduit aux données détaillées(ex : star

schemas, snowflake ...). Les métadonnées sont enregistrées dans un référentiel qui est très important contenant les Métadonnées suivantes :

- Métadonnées du business : contient les informations de la propriété, la définition de l'entreprise et les évolutions des politiques.
- Métadonnées opérationnelles : comprend les indices de données courantes, la traçabilité des données. Donc il se réfère aux données actives, archivés, ou purgés. La traçabilité des données signifie l'histoire des données migrées et la transformation appliquées sur elles.
- Métadonnées cartographiques : référence les bases de données source et leur contenu, l'extraction de données, la partition de données, le nettoyage, les règles de transformation, l'actualisation des données et des règles de purifications.
- des algorithmes pour résumer : il inclut des algorithmes de dimension, les données sur la granularité, l'agrégation et d'autres opérations.[2]

## 1.4 Les Systèmes OLAP

OLAP (OnLine Analytical Processing) est un système qui permet aux utilisateurs d'extraire facilement d'une manière sélective des informations. Les données manipulées par les processus OLAP sont stockées dans une base de données multidimensionnelle suivant le modèle de cube de données. Cette modélisation permet d'obtenir un aperçu de l'information grâce à un accès rapide aux grands volumes de données (par rapport à une requête sur OLTP) ; il offre également une cohérence et une interactivité à l'information stockée dans l'entrepôt[37].

### 1.4.1 Le Cube de données

Un cube de données représente un ensemble de mesures organisées selon un ensemble de dimensions qui représentent des axes d'analyse. Le cube de données est une structure de données qui surmonte les limitations opérationnelles d'une base de données relationnelle. Un espace multidimensionnel qui peut afficher et résumer les grands volumes de données tout en offrant aux utilisateurs un accès consultable à tous les points de données, ces opérations sont orchestrées par les systèmes OLAP[37].

### 1.4.2 La représentation de données

Le cube de données permet de représenter les données par plusieurs dimensions, il est définit par des dimensions et des faits, où les dimensions sont les entités sur lesquelles l'entreprise préserve les données archivées. Suppo-

sons, une firme veut analyser les données concernant les ventes, elle dispose de tables bidimensionnel les (temps, produits), contenant les données sur chacune de ses filiales. comme le montre le tableau suivant :

| | localisation = 'Blida' | | |
|---|---|---|---|
| Le temps (trimestre) | Produits | | |
| | accessoires | Vitres | Bureautiques |
| 1T | 600890 | 250000 | 807000 |
| 2T | 525200 | 210500 | 321600 |
| 3T | 250000 | 370500 | 967000 |
| 4T | 680440 | 400300 | 360000 |

FIGURE 3 – Exemple d'une analyse en ligne des ventes.

Pour observer la totalité, une table tridimensionnelle permet de créer une nouvelle dimension qui agrège les données particulières de chaque région par une nouvelle dimension(localisation), comme l'illustre le tableau suivant :

| Le temps | localisation = 'Chlef' | | | localisation = 'Blida' | | | localisation = 'Adrar' | | |
|---|---|---|---|---|---|---|---|---|---|
| | Produits | | | Produits | | | Produits | | |
| | Accesso-ires | Vitres | Burea-utiques | Accesso-ires | Vitres | Burea-utiques | Accesso-ires | Vitres | Burea-utiques |
| 1T | 754000 | 310000 | 700000 | 600890 | 250000 | 807000 | 398000 | 210000 | 307000 |
| 2T | 455900 | 810000 | 421600 | 525200 | 210500 | 600830 | 447000 | 246000 | 209800 |
| 3T | 157600 | 170000 | 600000 | 250000 | 370500 | 600830 | 580000 | 286500 | 257000 |
| 4T | 1000280 | 286600 | 360860 | 680440 | 400300 | 600830 | 640000 | 400300 | 400000 |

FIGURE 4 – Exemple d'une analyse en ligne des ventes sur trois dimensions.

Cette table tridimensionnelle peut être représentée par un cube de données, voir la figure 5.

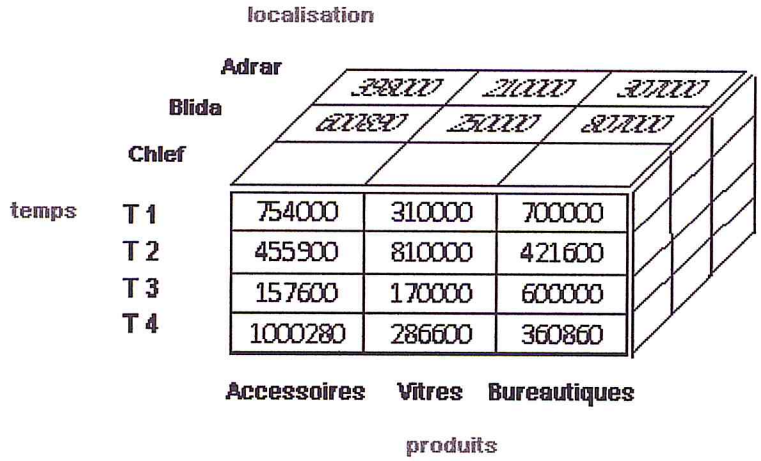



FIGURE 5 – Exemple d'un cube de données pour l'analyse des ventes.

Ce cube de données et composé de trois dimensions (axes d'analyse) et qui sont localisation , produit par rapport au temps.

### 1.4.3 Les opérateurs de navigation d'OLAP

Dans la modélisation multidimensionnelle, les données sont organisées en plusieurs dimensions, et chaque dimension contient plusieurs niveaux d'abstraction qui sont définis pas des nivaux hiérarchiques. Cette organisation flexible permet à l'utilisateur de faire des traitements selon les différentes perspectives. Dans ce qui suit nous présentons les opérateurs de navigation OLAP.

#### forage vers le haut 'Roll-Up'

Le rôle de cette opération consiste à faire une analyse avec moins de précision et en fonction d'un niveau de granularité moins détaillé. Le forage vers le haut effectue l'agrégation sur un cube de données dans l'une des deux façons suivantes :

— En grimpant dans la hiérarchie de concept pour une dimension.
— Par la réduction d'une dimension[2].

La figure 6, expose un exemple d'un forage vers le haut sur la dimension localisation pour une analyse en ligne de vente.

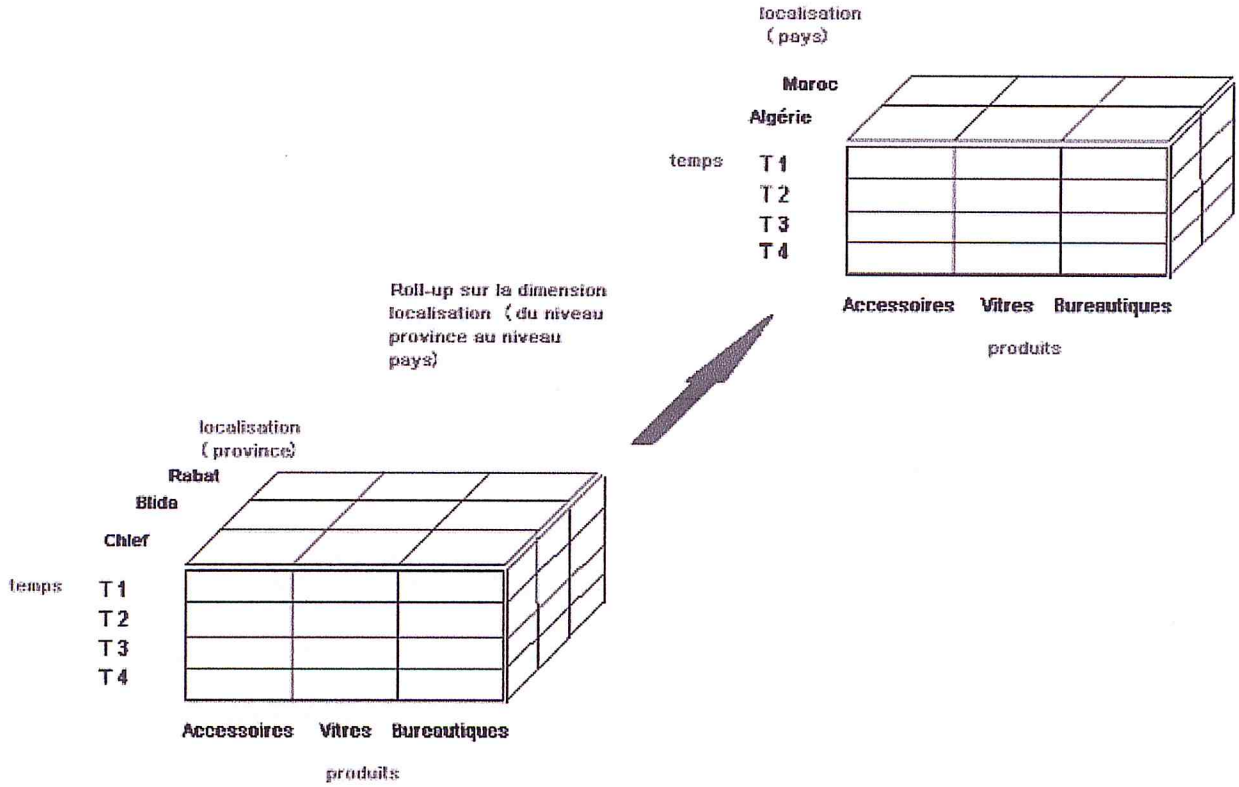



FIGURE 6 – Éxample de l'opération 'forage vers le haut'.

Dans cet exemple l'opération Roll-Up a permis d'observer les données selon un niveau hiérarchique supérieur dans la dimension localisation, cette dernière comprend les niveaux Ville, Province, Pays ; Dans l'exemple les données sont agrégées, en remontant du niveau 'province' au niveau 'pays', dans l'exemple les provinces 'Blida','Chlef' sont regroupées pour formé une nouvelle cellule qui représente le pays Algérie.

### forage vers le bas 'Drill-Down'

Drill-Down permet d'analyser les données avec un niveau granulaire plus fin. Le forage vers le bas c'est l'opération inverse du forage vers le haut. Elle se réalise en descendant vers l'un des niveau de granularité d'une dimension[2]. la figure 7 illustre l'opération à l'aide de l'exemple de vente : un exemple de l'opération dans la figure 7.

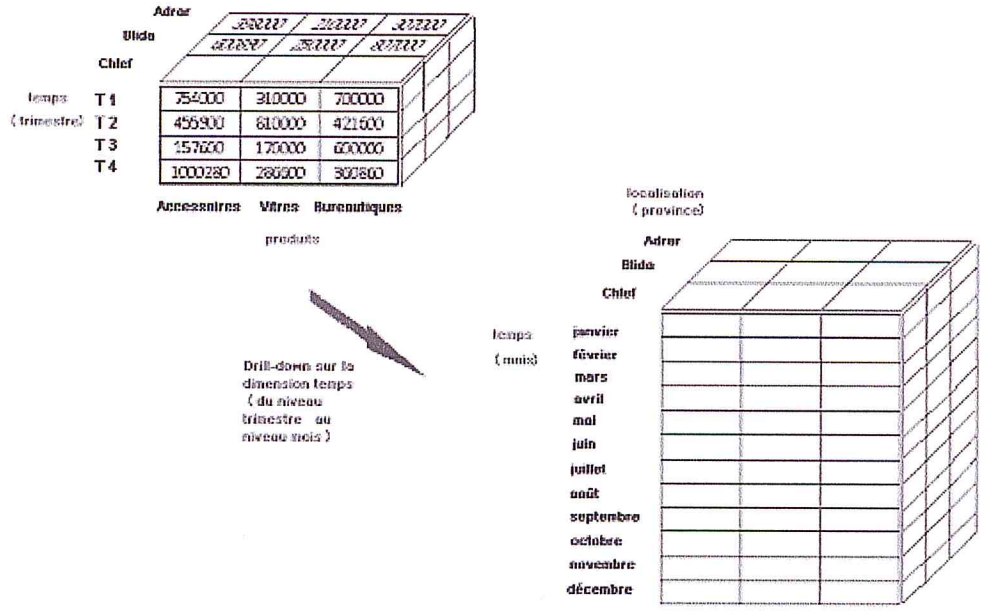



FIGURE 7 – Éxample de l'opération 'forage vers le bas'.

ici, l'opération Drill-Down a été appliqué sur la dimension temps où cette dernière a descendue du niveau hiérarchique 'trimestre' vers le niveau 'mois'. Appliquer ce mouvement vas permettre d'avoir une nouvelle vue de données d'un niveau de granularité plus fin.

### La tranche 'Slice'

L'opération de la tranche est une opération de restriction ; par la sélection d'une valeur pour l'une des dimensions, *Slice* fournit le sous-cube de l'intersection avec les autres dimensions. Considérons le schéma ci-dessous qui montre le fonctionnement de l'opération[2].

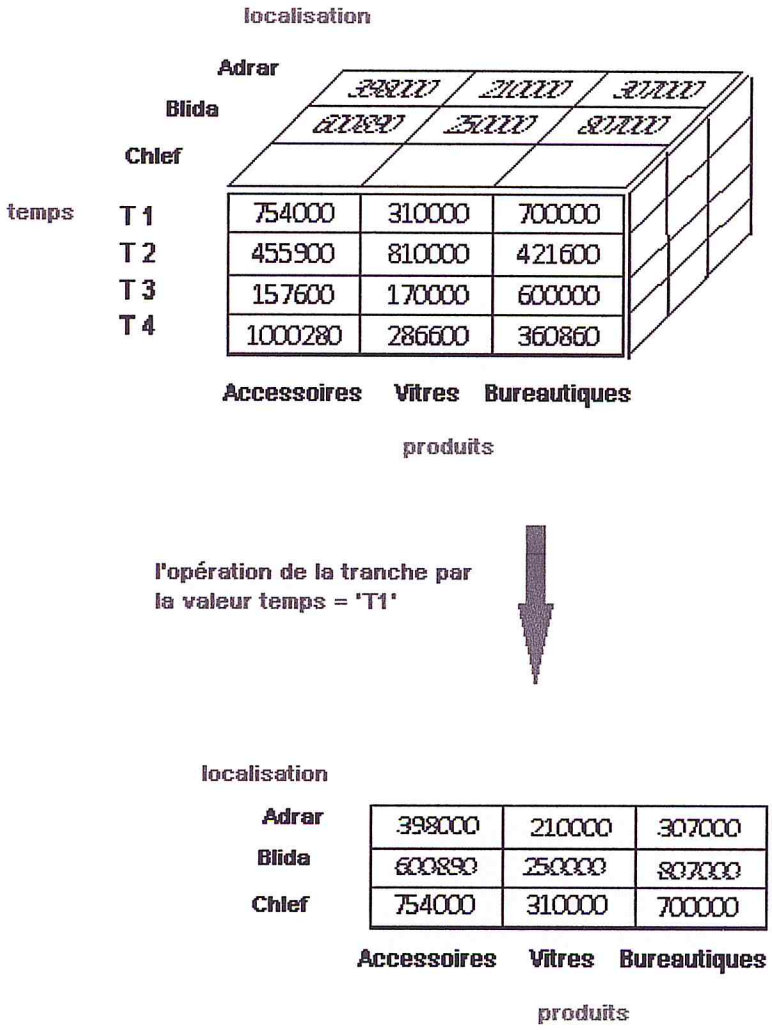



FIGURE 8 – Example de l'opération de la tranche.

la tranche a été appliquée sur la dimension temps où l'opération a eu comme paramètre 'temps = T1', ce qui a donné le sous-cube d'éléments de la tranche Q1.

### Dice

Dice est une opération de restriction basée sur la sélection de deux ou plusieurs dimensions d'un cube donné pour fournir un nouveau sous-cube, Considérons la figure 9, qui montre l'utilité de l'opération[2].

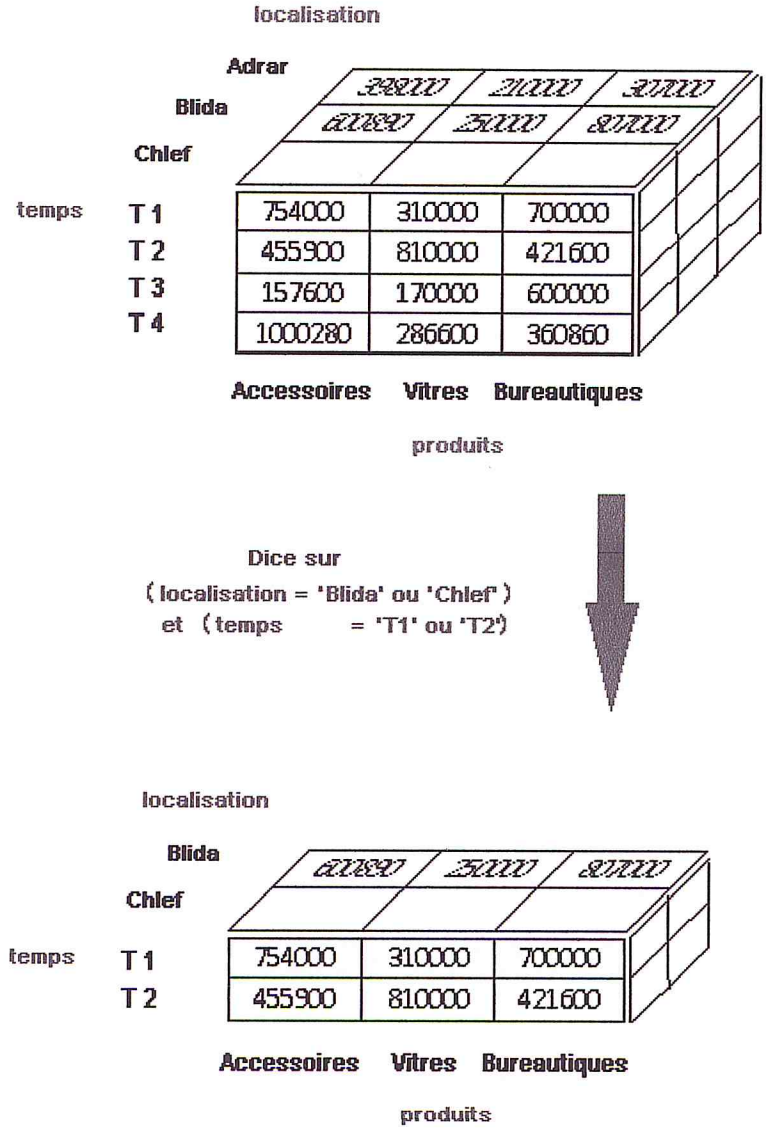



FIGURE 9 – Example de l'opération Dice.

La requête a été effectuée sur deux dimensions avec les paramètres (Localisation = 'Blida' ou 'Chlef'), (Temps = 'T1' ou 'T2') ; ce qui a permis d'avoir le sous-cube montré dans la figure.

### Pivot

C'est une rotation qui fait tourner les axes d'un cube en vue afin de fournir de nouvelles alternatives de présentation de données[2], comme l'illustre l'exemple dans la figure 10 :

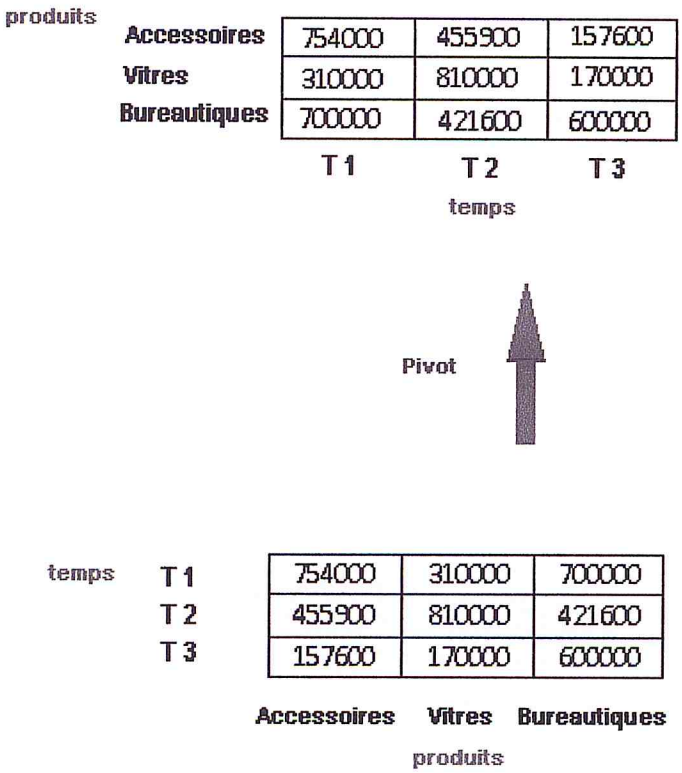



FIGURE 10 – Example de Pivot.

L'opération a permuté les dimensions 'localisation' et 'article'.

### Autres opérations

1. Opération de rotation :
2. Drill-Across : est une rotation de faits qui consiste à changer le sujet de l'analyse dans le contexte de constellation, cette opération requière une forte compatibilité entre les dimensions des faits sur lesquels s'applique l'opération.
3. Opérations de transformation :
4. Push : l'ajout d'attributs (de dimension) en tant qu'indicateur d'analyse (mesure d'analyse).

5. Pull : converti un indicateur d'analyse en attribut.
6. Opération d'ordonnancement :
7. Nest : permet d'insérer un nouveau attribut dans une hiérarchie.
8. Order : permet de changer l'ordre des valeurs des attributs d'une dimension.[3]

### 1.4.4 Les architectures des serveurs OLAP

Dans la section précédente, on a vu l'utilité des opérations de navigation OLAP. Un ensemble qui rends OLAP une solution efficace pour l'analyse de données. les implémentation de ces serveurs est suivant plusieurs types d'architecture ; c'est ce que nous allons introduire dans cette section.

#### Relational OLAP(ROLAP)

C'est l'approche la plus répandue qui se serve d'un SGBD relationnel pour stocker et gérer des données multidimensionnelles dans un environnement relationnel ce qui permet à ce serveur d'êtres évolutif[2]. Les serveurs relationnelles OLAP analysent de grands volumes de données à travers de multiples dimensions et montrent une haute performance quand il s'agit de données volatiles et changeantes, ainsi cette approche procure de nombreux avantages concernant les mécanismes de gestion des volumes de données très importants[1].
ROLAP présente une harmonie et une facilité d'interaction avec une base de données relationnelle, une efficacité de stockage car un serveur ROLAP n'utilise pas des cube de données pré-calculées.
La puissance de traitements des données structurées par ROLAP est indiscutable, mais ce dernier se montre limité avec une mauvaise qualité de requêtes sur les données non-structurées[1].

#### Multidimensional OLAP (MOLAP)

Cette approche est basée sur des moteurs de stockage vectoriel multidimensionnel de données sous la vision d'offrir des niveaux élevés de performance ; Ces moteurs stockent des données natives physiquement sous une forme multidimensionnelle (le concept de cube de données). MOLAP traite l'information avec un temps de réponse cohérent indépendamment du niveau de calculs ou d'abstractions sélectionnées, il adopte aussi deux niveaux de représentation et de stockage pour gérer les ensembles de données denses ou clairsemées. Lors de surcharge ou de densité d'un sous-cube de données ce dernier sera indexé, stocké dans un vecteur unidimensionnel ; en revanche les sous-cubes

légers seront compresser[1].
MOLAP est recommandé pour les utilisateurs inexpérimentés car il facilite l'indexation des données volumineuses résumées et pré-calculées.

### Hybrid OLAP (HOLAP)

Cette approche est préconisée par les grandes firmes industrielles de logiciel qui se base sur la création d'une cohabitation entre les deux approches précédentes au sein du même système afin de bénéficier d'avantages offerts par les deux et minimiser leurs faiblesses ; globalement dit ROLAP pour des données détaillées et MOLAP pour des données agrégées[1].
la figure 11 compare les critères sur les fonctionnalité essentielles et uniques de ROLAP et MOLAP.

| MOLAP | ROLAP |
|---|---|
| La recherche de l'informations est rapide. | La recherche de l'information est relativement lente. |
| Utilise des tableaux aux ensembles de données du magasin. | Utilise la table relationnelle. |
| le mieux adapté pour les utilisateurs inexpérimentés. | le mieux adapté pour les utilisateurs expérimentés. |
| Maintient une base de données distincte pour les cubes de données. | Il ne peut pas exiger un autre espace de stockage que celui disponible dans l'entrepôt de données. |

FIGURE 11 – Comparaison entre ROLAP et MOLAP.[2]

## 1.5 Conclusion

l'entrepôt de données est une structure flexible qui permet à l'entreprise d'élaborer la stratégie d'un développement durable. Dans ce premier chapitre nous avons parlé des aspects caractéristiques des entrepôts de données et des systèmes OLAP, Leurs efficacités quand il s'agit de données numérique et ses opérations dans un environnement multidimensionnel interactive. Les systèmes OLAP offrent des mécanismes de manipulation facilitant la visualisation des données par le biais des attributs catégoriques. Le processus d'aide à la prise de décision et la gestion de la connaissance, fait usage de ces technologies pour construire des modèles d'analyse, de classification et d'exploration de données.

Chapitre II

# La recherche d'information et le cube de texte

# 2 la recherche d'information et le Cube de texte

De nos jours les données disponibles représentent une masse considérablement énorme. Extraire et rechercher l'information à partir de ces textes devient de plus en plus important. Avoir un accès aux contenus des données textuelles, nécessite l'emploi d'un ensemble variant de technique selon le cadre de l'analyse dans un processus complexe. Le profile de l'analyste influence sur sa créativité durant le processus, par sa discipline d'origine, son référentiel théorique, ces compétences personnelles et son entourage. Malgré l'attention offerte à ce domaine de recherche, il n'existe ni standards ni consensus dans la communauté scientifique à part des émergences académiques popularisées par les praticiens qui divise l'analyse de données textuelles (L'ADT)en sept domaines de pratique, cette catégorisation est basée sur les caractéristiques uniques de chacun, les sept catégories sont : La recherche d'information (information retrieval, RI), le regroupement de documents (document clustering), la classification de documents (document classification), l'exploration du web (web mining), l'extraction de l'information (information extraction), l'extraction de concept (concept extraction) et le traitement du langage naturel (natural langage processing) [10].
L'analyse des données textuelles allie des outils et des méthodes multidisciplinaires durant le processus. Ce dernier commence par des opérations de prétraitement et d'extraction d'aspects caractéristiques représentatifs, permettant de faire une transformation de données non structurées stockées dans les collections de documents vers un format intermédiaire plus explicitement structuré. Réussir dans cette phase facilite l'analyse des grandes bases de données documentaires. La procédure de l'analyse traverse une multitude de passages et niveaux de traitement, un déplacement à partir des documents d'entrée brutes vers le texte entièrement codé. Le Choix de la méthode approprié est lié fortement à plusieurs facteurs qui influencent sur la réussite de l'analyse. Définir le niveau de granularité souhaité selon le corpus soumit à l'analyse et les objectifs de cette dernière. Le corpus doit être d'une bonne qualité et doit apporter suffisamment d'éléments pour pouvoir repérer des comportements significatifs sémantiques et/ou syntaxiques d'un point de vue statistique. Savoir s'orienter durant la procédure permet de bien choisir la sous-discipline qui donne les bons résultats.
Comme évoqué dans le chapitre précédent, les systèmes OLAP sont incapables de répondre aux besoins analytiques qui sont orienté vers l'extraction de l'information contenues dans les masses de données textuelles ; ces dernières offrent un potentiel analytique variant pour les systèmes décision-

nels. à-partir de ce constat nous allons introduire un modèle qui permet de transformer le text en nombres et un opérateur pour OLAP-text. L'approche proposée dans le cadre d'une recherche académique suggère un modèle de cube de données CXT-Cube et un opérateur pour l'agrégation des documents (Contextual Text Cube Model and Aggregation Operator for Text OLAP [16]). Le modèle et l'opérateur sont fondés sur des techniques de la RI en calculant des mesures reflétant la pertinence d'un document par rapport à une requête utilisateur où le contexte sera pris en considération et dans un environnement multidimensionnel.
Dans ce chapitre, nous nous focalisions sur les concepts de base de la Recherche d'information, ses techniques de traitements et le peu de son apport à l'analyse de données textuelles, et nous allons introduire le modèle CXT-cube.

## 2.1 La recherche d'information

Actuellement, les masses d'informations numériques ne cessent d'augmenter (web, bibliothèque numérique, encyclopédie en ligne...), d'où la nécessité d'avoir des systèmes permettant aux utilisateurs de chercher rapidement et d'exploiter ces masses d'information ; des systèmes à qui on fournit des descriptions appropriés, agissant avec robustesse dans ces référentiels de documents[4]. La recherche d'information identifie adéquatement le contenu de l'information des données textuelles et traite l'incertitude et l'imprécision par ces modèles[5]. une discipline autonome et ces techniques regroupent la représentation, le stockage, l'organisation et la récupération des données non structurées généralement, et des documents textuels en particulier. Un domaine de collecte et découverte.

### 2.1.1 Définition

Un système de recherche d'information est un système informatique qui inclut un ensemble d'opérations et traitements d'informations dans un processus qui permet la gestion, le stockage, la représentation, la recherche, la sélection et l'interrogation d'une collection de documents. Un système de Recherche d'information permet de retourner à partir d'un corpus de documents, ceux dont le contenu correspond le mieux à une requête exprimée par l'utilisateur[6].

### 2.1.2 Le processus de la recherche d'information

Un processus formel de recherche d'information se compose essentiellement de deux phases, une pour la représentation des documents et de la requête

et l'étape de l'analyse qui compare la requête avec les documents indexés pour renvoyer les résultats.

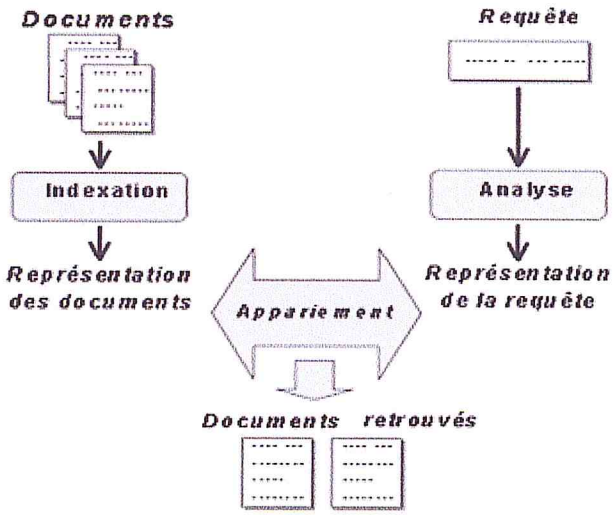



FIGURE 12 – Un processus en U de la recherche d'information. [6]

comme l'illustre la figure 12,le système procède par le traitement de la collection documentaire durant le processus de l'indexation afin de faciliter la représentation de l'information. d'autre part le système analyse et représente les requêtes pour l'étape d'appariement qui compare la requête et les documents afin de retrouver ceux qui sont pertinents pour la requête.

## 2.2 Les prétraitements des collections de documents

Une collection de documents est tout groupement de documents à base de texte, qui peut aller de plusieurs milliers à plusieurs dizaines de millions et la qualité de la collection est un élément clé de la réussite des traitements analytiques. Les collections de documents peuvent être statiques ou dynamiques. Une collection de documents dynamique' est un terme appliqué à des collections de documents caractérisés par leur inclusion des documents nouveaux ou misent à jour au fil du temps; relativement à plusieurs facteurs, les collections extrêmement grandes peuvent causer des problèmes de performances pour les divers composants du système d'analyse.

### 2.2.1 Le document

Le document est l'élément de base en recherche d'information, et qui peut être défini en tant que l'unité de données textuelles discrètes dans une collection de documents. Rapport d'activité, note juridique, e-mail, document de recherche, manuscrit, article, communiqué de presse, ou histoire de nouvelles...etc. Bien qu'il ne soit pas typique, un document peut être défini un peu moins arbitraire dans le cadre d'une collection de documents notamment en décrivant un document prototype basé sur la représentation d'une catégorie similaire d'entités au sein d'une collection. Un document textuelle peut être vu de différents angles; un objet structuré linguistiquement qui peut également montrer une riche quantité sémantique et syntaxique, ses structurations sont d'avantage les moyens permettant l'interprétation du contenu. D'un autre point de vue, les éléments typographiques tels que les signes de ponctuation et la capitalisation lorsqu'ils sont couplés avec les éléments de la mise en page (les tabulations, espacements, retours chariot, soulignements, tableaux ...etc) peuvent être interprétés comme une sorte de langage de balisage (ex : HTML) fournissant les indices qui facilite la découverte de métadonnées et identifier les champs importants dans un document[11].

### 2.2.2 Caractéristiques de documents

Les documents doivent passer par des prétraitements avant tout traitement analytique. Les algorithmes de ce dernier opèrent sur des modalités de représentation de base, ils identifient les caractéristiques efficaces à employées parmi de nombreuses, les plus utilisées sont les caractères, termes, formes linguistiques, concepts.

### 2.2.3 L'indexation

C'est le processus qui a pour objet de produire une présentation réduite et formalisée des documents en y retenant l'ensemble des éléments essentiels . Trouver les tendances afin de créer une représentation du document par l'acquisition des formes linguistiques et les richesses sémantiques et syntaxique qui doivent être représentés exhaustivement[11]. L'indexation facilite la description de requêtes et documents par le biais de descripteurs qui contiennent des mots et/ou des suites de mots qui sont rangé dans un dictionnaire d'indexation. L'indexation peut être automatique (effectuée par un système informatique et sans l'intervention de l'expert), manuelle (par un expert de domaine) ou semi-automatique assisté par l'expert qui intervient pour définir certains paramètres (ex : relations sémantiques). Le processus formel de l'indexation automatique contient un ensemble de traitements : la

Tokenisation, la suppression des mots vide (stop words removal), la racinisation et/ou la lemmatisation pour normaliser les termes et finalement des calculs statistique pour trouver les poids de termes. Ces prétraitements sont très importants pour faciliter la représentation de documents et améliorer la performance du système.

### 2.2.4 Tokenisation

C'est la phase de récupération de termes ; la Tokenisation découpe un flux de texte ou une séquence de caractères en unités linguistiques nommées *Tokens*(des mots généralement)[11],et qui seront regroupés pour les opérations analytiques. durant la procédure de découpage, les espaces, les sauts de ligne et la ponctuation sont considérés séparateurs, mais ceci n'affirme pas que les résultats ne peuvent contenir de la ponctuation ou des espaces blancs, dans certains cas particuliers(les noms propres composés).

### 2.2.5 Elimination des mots vides

Les mots vides (stop words) sont les mots très communs qui sont inutiles pour être indexés et réutilisés pour retrouver les documents correspondants aux besoins d'utilisateur. dans cette phase l'ensemble de mots vides est complètement exclu du vocabulaire (ex : de, des, le, la, les, sur, dans, ...etc).

### 2.2.6 techniques de Normalisation

Pour avoir la productivité estimé de l'analyse, l'étape de la normalisation peut être réalisée par la racinisation et/ou lemmatisation.
La lemmatisation désigne l'analyse lexicale du contenu d'un texte regroupant les mots par la réduction en leurs lemme (forme canonique). cette technique regroupe les différentes formes que peut revêtir un mot, soit : le nom, le pluriel, le verbe à l'infinitif, etc.[14] En linguistique, la racinisation ou la désuffixation (anglais : stemming) est un procédé de transformation des flexions en leur radical ou racine (anglais : stem). Les techniques utilisées pour ce faire reposent généralement sur une liste d'affixes (suffixes, préfixes, postfixes, antéfixes) de la langue considérée et sur un ensemble de règles construites a priori qui permettent, étant donné un mot de trouver sa racine.[15] la normalisation permet de faire l'unification de termes ayant la même racine et/ou le même lemme afin qu'il soient considérés comme un seul ; ainsi facilitant la représentation des unités textuelles (requêtes/documents).

## 2.3 les modèles de représentation de documents

Pour voir la pertinence dans un document, c'est très important de transformer la version textuelle complète du document vers une forme qui permet d'accéder rapidement à l'information. le système de recherche d'information est censé calculer la similarité entre la requête indexée et les descripteurs de documents ; seul les documents dont la similarité dépasse un seuil prédéfini seront jugées pertinents. L'appariement peut être approché ou exact ; dans le premier, les résultats sont triés selon un ordre de mesure, cet ordre reflète le degré de pertinence. Dans le deuxième mode, le système retourne des documents qui répondent exactement à la requête spécifiée et qui ne sont pas triés. afin de réussir le processus de représentation, les modèles existants font usage d'une méthode de pondération qui range les termes d'après leur pertinence.
la RI compte trois modèles de représentation de document ; modèle booléen, probabiliste et vectoriel, dans cette partie nous allons focaliser sur le dernier vu sa relation avec le modèle CXT-Cube.

### 2.3.1 La méthode de pondération TF-IDF

La méthode statistique de pondération $TF - IDF$ (*Terme Frequency - Inverse Document Frenquency*) permet d'évaluer l'importance d'un terme dans une collection documentaire. la fréquence de termes $TF$ est le nombre d'occurrences d'un terme dans l'unité de texte. La fréquence inverse de document $IDF$ est une mesure qui permet de discriminer les termes les moins fréquents. la méthode $TF$ est Utilisé dans Le vecteur de termes qui regroupe les fréquences avec lesquelles les termes apparaissent dans l'unité textuelle codée (document/requête) ; les documents sont considérés similaires si leurs vecteurs de termes se trouvent rapprochés dans l'espace vectoriel. Avant de déterminer la distance dans le vecteur de terme, les dimensions de l'espace devraient être normalisées d'une manière qui reflète l'importance des différents mots. cette dernière est généralement mesurée simplement sur la base de la fréquence des mots, des mots rares étant plus pertinents que ceux qui sont courants [7].

$$TF.IDF(M_i, D) = TF(M_i, D).IDF(M_i)$$

Où $TF(M_i, D)$ est le nombre d'occurrences du terme $M_i$, dans le document D.

$$IDF(M_i) = log(\frac{(1+N)}{1+DF(M_i)})$$

Où $IDF(M_i)$ est le nombre de documents contenant $M_i$ et $N$ est la taille du corpus(nombre de documents). Cette méthode donne de très bons résultats et c'est une référence pour les pratiques analytiques de la recherche d'information[7].

### 2.3.2 Le modèle vectoriel de termes

Ce modèle repose sur la base mathématique des espaces vectoriels, qui est interprété dans l'indexation des requêtes et documents, ces derniers sont représentés dans un espace vectoriel engendré par l'ensemble de termes d'indexation :$t_1, t_2, t_3, , , , , , t_N$ où $N$ est le nombre totale de termes issus de l'indexation d'une collection de documents [6].
Le document $D_J$ est représenté par le vecteur :$D_J = (d_{1J}, d_{2J}, , d_{iJ}, , , d_{NJ})$, où $d_{iJ}$ est le poids du terme $t_i$ dans le document $D_J$ .
La requête R est représenté par le vecteur :$R = (r_1, r_2, , r_i, , , r_N)$, où $r_i$ est le poids du terme $t_i$ dans la requête $R$.
Dans cette approche Les termes absents ont les poids nuls et les valeurs sont calculées à partir du nombre d'occurrences de chaque terme dans un document. Toutes les autres informations implicites dans le texte, tel que le sens et l'ordre des mots, seront perdues une fois la représentation vectorielle a été calculée.
L'appariement dans ce modèle est basé sur une fonction (RSV : *Retreival Status Value*), qui calcul le coefficient de similarité entre chaque document $D_J$ et la requête .
Exemples de fonctions RSV :[6]

Produits scalaire : $RSV(R, D_J) = \sum_{i=1}^{N} r_i * d_{iJ}$

Mesure de Jaccard : $RSV(R, D_J) = \frac{\sum_{i=1}^{N} r_i * d_{iJ}}{\sum_{i=1}^{N} r_i^2 + \sum_{i=1}^{N} d_{iJ}^2 - \sum_{i=1}^{N} r_i * d_{iJ}}$

Mesure de cosinus : $RSV(R, D_J) = \frac{\sum_{i=1}^{N} r_i * d_{iJ}}{(\sum_{i=1}^{N} r_i^2)^{\frac{1}{2}} * (\sum_{i=1}^{N} d_{iJ}^2)^{\frac{1}{2}}}$

Ainsi La similarité dépend des poids de termes qui coïncident dans les textes requête/documents ce qui permet de faire un classement de pertinence par ordre décroissant.
L'avantage du modèle vectoriel réside particulièrement dans l'ordonnancement des documents sélectionnés selon leurs pertinences. Cependant,

### 2.4.1 Définition du modèle

Le modèle de cube de texte contextuel CXT-Cube et un cube de données qui modélise un sujet d'analyse appelé fait $F$ défini par plusieurs dimensions $Dim_r, r \in [1, *]$. Une dimension comprend les attributs $A =< a_1, a_2, ..., a_* >$ qui peuvent être organisés en plusieurs niveaux hiérarchiques $< n_1, n_2, ...., n_* >$. F est associé à une ou plusieurs mesures. Une mesure d'une table de faits $F$ est notée $M$. un fait $F$ avec ses dimensions $Dim_r$ forment un cube par un schéma en étoile $Cube = (F, Dim_1, Dim_2, ..., Dim_*, M_1, M_2, ..., M_*)$. Ainsi, CXT-Cube soutient le contexte du fait qui est représenté en utilisant les dimensions contextuelles.

### 2.4.2 les dimensions contextuelles

Les dimensions dans ce modèle sont contextuelles. Chaque dimension est liée à un facteur contextuel correspondant à la définition de contexte dans 4.1 . Par exemple, considérons les sujets du document comme un facteur contextuel du contexte du document. Ainsi, la dimension d'un sujet qui correspond à ce facteur contextuel doit être créer . dans le modèle CXT-Cube, les dimensions contextuelles peuvent être classées en deux types :

**Les dimension sémantiques :**

elle est extraite d'une ontologie de domaine liée à la zone de dimension comme source de connaissance externe. Cette hiérarchie spécifie les niveaux sémantiques et les relations entre les termes de texte dans le cube de texte. Chaque niveau $n_i =< c_1, c_2, ....c_n >$ comprend un ensemble de concepts $c_j (j \in [1, n])$ extraits d'une ontologie de domaine.

**Dimension des métadonnées :**

les métadonnées sont des informations externes sur les documents, telles que : date, titre, auteur, etc. Chacune de ces informations représente un facteur contextuel tel qu'il est expliqué ci-dessus. Ainsi, dans ce modèle contextuel, les dimensions de métadonnées sont crées pour représenter chaque information de métadonnées.
Revenons à l'exemple utilisé dans les testes pour voir l'intérêt de l'approche qui est le cas d'une analyse OLAP sur une collection de CV. Le fait observé, dans cet exemple, sont les documents (CVs). Un document est défini par deux dimensions sémantiques : sujet et localisation ; Chacun contient une hiérarchie conceptuelle. La dimension thématique comprend une hiérarchie

d'actualité représentant des compétences de domaine. La dimension de localisation comporte trois attributs hiérarchiques qui sont 'pays', 'région' et 'ville'. Une autre dimension contextuelle est le temps et qui représente la date d'envoi du CV comme une métadonnée des documents.

### 2.4.3 Mesure pour l'analyse des données textuelles

Afin d'analyser les données textuelles en tenant compte des facteurs contextuels définis dans la section précédente, l'approche propose une mesure d'analyse basée sur le modèle d'espace vectoriel qui est habituellement utilisé dans la RI. Cependant, la représentation d'un document par le modèle d'espace vectoriel dans le contexte d'une analyse OLAP nécessite une adaptation, en particulier pour prendre en compte l'information sémantique dans les dimensions du cube. Ainsi, la représentation du sujet de l'analyse (documents) en fonction des différentes dimensions du cube OLAP. l'approche emploie une technique de propagation de pertinence sur les dimensions sémantiques représentée par des hiérarchies de concepts dans l'ordre de mieux soutenir la sémantique des données textuelles. La propagation de la pertinence est une technique largement utilisée dans la recherche d'information. Le modèle vectoriel spatial ou le modèle vectoriel à terme est un modèle algébrique pour représenter des documents texte où chaque document d'une collection est représenté par un vecteur de termes dans un espace multidimensionnel.
Soit R l'espace vectoriel défini par l'ensemble des termes : $R < t_1, t_2, ..., t_n >$.
Un document d est représenté par un vecteur de poids comme suit : $d < w_{t1}, w_{t2}, ..., w_{tn} >$.

**Définition :**

Une mesure d'analyse de données textuelles représente chaque document d par plusieurs vecteurs de concepts pondérés, Un vecteur pour chaque dimension $dim_r$ du cube.
$M =< \overrightarrow{d_{Dim_1}}, \overrightarrow{d_{Dim_2}}, ..., \overrightarrow{d_{Dim_r}} >$ où $\overrightarrow{d_{Dim_r}} =< w_{c_1}, w_{c_2}, ..., w_{c_n} >$
est le vecteur des concepts pondérés d'un document d dans une dimension d'espace vectoriel spécifique à $Dim_r$ et $w_{c_i}$ est le poids attribué au concept $c_i$.
Dans l'exemple de CVs la mesure d'analyse est :
$M =< \overrightarrow{d_{Dim_z}}, \overrightarrow{d_{Dim_t}}, \overrightarrow{d_{Dim_l}} >$ où $\overrightarrow{d_{Dim_z}}, \overrightarrow{d_{Dim_t}}, \overrightarrow{d_{Dim_l}}$ sont les vecteur de dimensions : SUJET($dim_z$),LOCALISATION($dim_l$) et TEMPS($dim_t$).

### 2.4.4 La propagation de pertinence pour la mesure d'analyse

l'approche propose de considérer la sémantique du texte représentée par les dimensions sémantiques lors de la pondérations de termes dans un document. Le poids de terme est calculé à la fois par sa fréquence d'occurrence et par sa propagation de pertinence dans la hiérarchie conceptuelle. La technique permet de réaffecter les scores dans la hiérarchie des concepts. Par exemple, si les concepts Java, PHP, UML et Oracle existent dans un document d, le poids du concept informatique sera augmenté en utilisant la technique de propagation de pertinence car il s'agit d'un nœud ancêtre sur les concepts précédents dans la hiérarchie. En outre, la propagation de la pertinence permet de prendre en considération de nouveaux concepts. Par exemple, si nous avons un document représenté par un vecteur qui comprend les concepts Java et PHP, le concept programmation sera inclus dans le vecteur, même s'il n'appartiens pas au document. La technique de propagation de pertinence s'effectue par les étapes suivantes :

* Tout d'abord, on calcule les poids des termes du document qui existent dans la hiérarchie conceptuelle en utilisant la fréquence d'apparition des termes Tf qui est calculée selon la formule suivante :

$$Tf_{t,d} = \frac{n_{t,d}}{N_d}$$

Où $n_{t,d}$ est la fréquence d'occurrence du terme t dans le document d et $N_d$ est le nombre total de termes dans le document d.

* Ensuite, on applique la technique de propagation de pertinence sur la hiérarchie des concepts. Cela permet d'étendre le vecteur conceptuel d'un document par de nouveaux concepts qui sont les nœuds ancêtres des concepts pondérés dans le document. De plus, il permet de donner de nouveaux poids aux concepts. Le principe de cette méthode de propagation de pertinence est de propager les scores attribués aux nœuds foliaires à travers une structure arborescente. Pour chaque nœud de feuille qui a un poids non nul, les poids de ses ancêtres sont calculés selon la formule ci-dessous :

$$Poid(n_k, n_{fi}) = Poid(n_k) + Poid(n_{fi})^{distance(n_k, n_{fi})+1}$$

Où $n_{fi}$ est le nœud feuille à partir duquel la propagation de la pertinence est effectuée. $n_k$ est le nœud ancêtre auquel le nouveau poids est calculé. $distance(n_k, n_{fi})$ est la distance sémantique entre les nœuds $n_k$ et $n_{fi}$ représentés par le nombre d'arêtes entre eux. Ainsi, la propagation de la pertinence dépend de deux paramètres :

— Le nœud feuille $n_{fi}$ à partir duquel la propagation de la pertinence

est effectuée ; et si le poids du nœud de la feuille est important, la propagation de la pertinence augmente.
— La distance entre le nœud feuille $n_fi$ et son ancêtre $n_k$ ; Lorsque la distance entre les deux nœuds augmente, la propagation de la pertinence diminue.

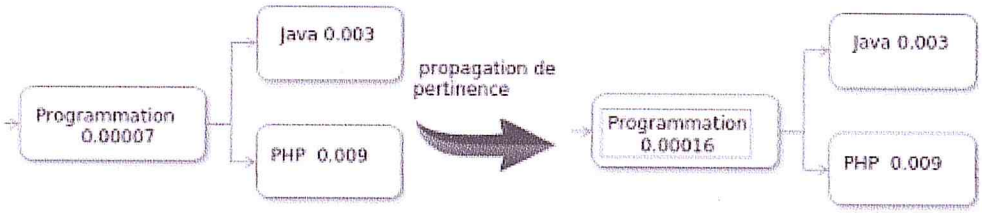



FIGURE 13 – Exemple de la propagation de pertinence.

### 2.4.5 la mesure de similarité pour l'agrégation des données

Les opérations de navigation à travers les cubes OLAP utilise massivement des agrégations telles que les opérations de forage et de déploiement. l'approche repose sur le calcule de la mesure de similarité pour juger sur la pertinence des documents. Le modèle d'espace vectoriel représente un document par un vecteur de termes dans un espace multidimensionnel contenant les termes résultant de l'indexation de la collection. Dans les systèmes de la RI, une requête est également représentée par un vecteur de termes.
Soit $R$ un espace vectoriel défini par l'ensemble des termes :
$R < t_1, t_2, ..., t_n >$.
Un document $d$ et une requête $q$ sont représentés par des vecteurs de poids comme suit :
$d < w_{t1}, w_{t2}, ..., w_{tn} >$ , $q < w_{qt1}, w_{qt2}, ..., w_{qtn} >$.
Où : $w_{ti}$ et $w_{q_{ti}}$ correspondent au poids du terme $ti$ dans le document $d$ et à la requête $q$, $n$ est le nombre de termes dans l'espace. En général, la similarité entre un document $d$ et une requête $q$ peut être calculée par le cosinus de l'angle entre les deux vecteurs $d$ et $q$ comme suit :

$$Sim(d,q) = cosa = \frac{\sum_i {w_{ti}}^2 * w_{qti}^2}{\sqrt{\sum_i w_{ti}^2 * \sum_i w_{qti}^2}}.$$

## 2.5 Conclusion

Dans ce chapitre nous avons donné un aperçu de la Ri, de ses techniques d'indexation, du modèle vectoriel pour la représentation et la formulation de

requêtes basé sur les mots clés. Dans la plupart du temps, les requêtes dans un système de recherche d'information prennent la forme d'un texte brut réduit en mots clé. le calcul de pertinence ou l'appariement dans la RI est basé sur la mesures de similarité RSV. la recherche d'information est un domaine quasi autonome, qui détient l'attention de chercheurs académiques ou industriels voulant concevoir des systèmes d'analyse performants et efficaces.
Nous avons vu également l'approche CXT-Cube qui permet d'adapter une collection de documents textuels non structurés à une analyse OLAP avec la prise du contexte.
La réalisation d'une système d'ADT basé sur un modèle pareil, requiert des systèmes performants ayant les caractéristiques nécessaires pour avoir une haute scalabilité et évolution. Dans le chapitre suivant nous allons parlé de ces dépendances et exigences.

Chapitre III

# Les bases de données NoSQL et le framework Hadoop

# 3 Les bases de données NoSQL et le framework Hadoop

Les systèmes de gestion de base de données relationnelle (SGBDR) prennent en charge l'atomicité, la cohérence, l'isolation et la durabilité. Un ensemble de propriétés définies par l'acronyme ACID où les données doivent se conformer au schéma de la base de données sous une gestion stricte des différentes opérations. Les SGBDRs imposent que les transactions simultanées s'exécutent séparément et garantissent la capacité à revenir au dernier état connu après les pannes imprévues du système. Obtenir une performance élevées dans un SGBDR est lié à l'optimisation de la structure de tables et la bonne modélisation de requêtes, d'autre part dans les modèles relationnels la conception précède le stockage ce qui devient chaotique lors d'une mauvaise conception, et qui pousse vers une modification en profondeur. L'évolutivité des SGBDRs est lente ainsi que les surcharges de serveurs demandent des extensions et le déploiement de l'infrastructure (serveurs propriétaires) devient couteux. Pour faire face à ces problèmes de scalabilité, plusieurs techniques étaient développées pour étendre les possibilités des SGBDRs comme le SHARDING, la DENORMALIZATION et le DISTRIBUTED CATCHING. Malgré les alternatives conçus pour pallier aux différents problèmes, les industriels du software et les consommateurs n'ont aucun intérêt à investir dans des solutions couteuses. Les développeurs ont dû prendre en main le fardeau et ils ont inventé leurs propres technologies NoSQL principalement et majoritairement autour d'un système nommé Hadoop.

## 3.1 Histoire

Le terme NoSQL (Not only SQL) a marqué son apparition pour la première fois en 1998 pour les bases de données relationnelles qui manque d'une interface SQL. Le terme a été utilisé pour une deuxième fois dans l'an 2009 durant une conférence ( Advocates of Non-Relational Databases) qui a été organisé pour parler de nouvelles alternatives qui surpassent les problèmes des SGBDRs. Parmi les leaders de marché informatique Google et Amazon ont inventé et développé leurs propres technologies NoSQL et ultérieurement d'autres vendeurs et communautés Open-source ont pris leurs part de cette émergence de solutions NoSQL[12].

## 3.2 Pourquoi choisir le NoSQL

Les modèles NoSQL partagent des propriétés qui doivent être prisent en considération avant tous choix de basculer vers ces technologies. L'une de ces propriétés est le CATCHING intégré qui permet de stocké et retrouvé les données sans impliquer l'utilisateur dans cette opération complexe. Les bases NoSQL peuvent également effectuer un SHARDING automatique qui permet à une base d'être répartie sur plusieurs serveurs qui sont créés, ajoutés ou supprimés à la volée et d'une manière automatique sans affecter la performance. L'élasticité est le concept qui permet aux bases NoSQL de pallier les vertiges du BigData ; une caractéristique qui permet aussi à la représentation de données d'être changé sans impacter l'application. Les bases NoSQL sont conçues pour minimiser les problèmes des SGBDRs, mais un certain nombre de points sont à considérer avant de foncer dans le choix d'une solution de stockage non relationnelle, comme le montre le théorème de CAP.

## 3.3 Théorème de CAP

C'est un théorème sur les caractéristiques vitales d'une base de données ; c'est l'abréviation de (Consistency, Availability, Partition tolerence). Ce théorème initié par Eric Brewer, prouvé par Seth Gilbert et Nancy Lynch ; énonce qu'il est impossible pour un système distribué de fournir les trois propriété précédentes à la fois[12].
La consistance : tous les nœuds du système voient les même données au même moment.
La disponibilité de données : les requêtes d'écriture et de lecture sont toujours satisfaites.
La tolérance au partitionnement : la seule raison qui pousse un système à l'arrêt est la coupure totale du réseau. ce qui veut dire le système reste opérationnel lors d'une panne partielle.
la conception d'une architecture distribuée est limité à choisir une paire de ses propriétés ce qui permet d'avoir l'une des trois paires suivantes :
- CP : dans ce type de systèmes, les données sont consistantes entre tous les nœuds ainsi que le système possède une tolérance au pannes, mais des problèmes de disponibilité et de latence doivent être considérés[13].
- AP : la conservation de ces deux propriété permet d'avoir ; un système tolérant aux pannes et qui garantit un temps de réponse fiable, cependant la consistance des données entre nœuds est le point faible de cette architecture[13].
- CA : est une architecture qui offre la consistance entre tous les nœuds du système avec une satisfaction sur les requêtes lecture/écriture, sauf dans

du système peut changé dû à la prochaine propriété.
- Éventuellement consistant 'Eventually consistent' : veut dire que le système deviendra consistent à mesure que le temps passe, à condition qu'il ne sera pas interrompu par les action d'utilisateurs entre temps[12].

## 3.5 Les critères de choix entre ACID et BASE

afin de choisir la bonne technologie pour un cas d'utilisation bien défini, il faut bien peser certain points, ce qui facilite l'adoption de la solution :
les notions ACID sont recommandées si :

- une données est partagée par un groupe de processus ou d'utilisateurs au même moment.
- si les transactions sont en temps réel et l'échec d'une opération devient chaotique.
- si l'ordre des transaction est d'une importance majeure. les bases BASE sont recommandées si :
- l'abandon de la transaction n'est pas d'impact important.
- si l'ordre des opérations n'est pas fondamental.
- si le partage d'une même données au même moment est relative à une mise-à-jours.

la détermination d'un choix ACID ou BASE sera relative à ces conditions .

## 3.6 Les types de bases de données NoSQL

Le théorème de CAP insinue que chaque architecture à une faiblesse liée à l'une des trois caractéristiques, pour réduire les faiblesses au niveau opérationnel plusieurs types de base de données sont apparus.

### 3.6.1 Orientées clé-valeur

Une base orientée clé-valeur est nommée aussi associative. Elle est basée sur le principe le plus simple des bases NoSQL ; un système qui stocke des valeurs indexées par des clé uniques sans être limité à un modèle de données prédéfini ; Garantissant des requêtes simples, seulement par le biais des clauses (PUT, GET et DELETE). les requêtes de restriction comme en SQL avec la clause (WHERE) sont liées à la bonne gestion de la paire clé/valeur. Par conséquent une opération de filtrage nécessite un traitement intensif si les clé de valeurs voulues sont inconnues[12]. L'architecture de ce type est tolérante aux pannes dans un mode distribué. elle offre également la meilleure évolutivité (scalability) que celle des autres types, pourtant elle ne garantit guère

le support des propriété ACID car elle ne dispose d'aucune API SQL pour la gestion.

### 3.6.2 Orientées colonnes

C'est une évolution du premier modèle et c'est l'ensemble le plus adopté des solutions émergentes du mouvement NoSQL, car elles sont proches du concept relationnel par leur structure qui organise les données en colonnes et pas en lignes. un avantage qui élimine le problème de latence dû à la lecture de toutes les lignes avant de faire la sélection de colonnes nécessaire. Un modèle qui permet de retrouvé l'information, d'effectuer des agrégation et des tris sans encombrer le processus avec des colonnes inutiles pour le traitement. Ce type montre une performance remarquable pour des opérations sur des grandes masses de données[12]. Il existe des solutions issus d'une fusion entre ce modèle et les bases relationnelles classiques, qui permettent de faire la gestion de processus métiers dans une organisation en lignes, ainsi que les traitement analytique en colonnes, ce qui facilite l'accès rapide à l'information, une fiabilité du premier rang et assurant un grand potentiel de scalabilité[13].

### 3.6.3 Orientées documents

les bases de données orientées documents fonctionnent par le principe de l'associativité (clé-document), C'est l'une des implémentations alternatives de bases orientées colonnes. Les documents dans ces bases sont représentés par des champs indexés où chacun possède sa propre structure et types, une modélisation qui optimise le temps d'accès à l'information sans faire appel seulement à la clé ; les fonctionnalités offertes par ce modèle le rapproche de son ancêtre relationnel et elles comprennent l'ajout, la modification, la lecture ou la suppression de certains champs du document, y compris la possibilité d'inclure des prédicats sur les champs. les API de gestion de ces bases fournissent des mécanismes pour cartographier la base suivant le modèle (Document/Objet) afin de requêter de manière objet[12].

### 3.6.4 Orientées objets

les bases de données sont liée fortement aux langages de programmation, C'est une approche ancienne appuyée par le mouvement NoSQL ; en voyant la solution dans le couplage fort entre l'application et la base de données. L'ensemble de ces systèmes permettent d'opérer sous l'optique de la programmation orientée objet et majoritairement par le langage OQL (Object

Querry Langage).elles supporte la création, manipulation et le stockage d'objets instanciés (par l'application) dans la base de données[13]. un gain énorme issu de relations entre objets où les opérations sont basées sur les modèles des objets plutôt que sur les volumes de données. cette architecture est la moins utilisée parmi les solutions NoSQL.

### 3.6.5 Orientées graphes

Des systèmes qui basent sur "les théories des graphes" pour représenter la complexité des données, ces derniers sont l'axe le plus important quand il s'agit de l'évaluation de la scalabilité. "les théories des graphes" est une discipline riche d'algorithme et de modèles permettant d'assouplir les contraintes de conception dans les bases relationnelles classiques. le principe dans ces systèmes et contrairement aux ancêtres relationnels, est de définir les relations durant les enregistrements où la structure peut évoluer librement. dans les bases orientées graphes il faut examiner les enregistrements pour définir les relations. ce paradigme est recommandé pour représenter une structure chaotiquement complexe (ex :les relations d'un réseau social) et il offre des potentiels analytiques énormes via les techniques de 'Graph mining'[13]. l'inconvénient majeur de ces système est dans le manque de support pour la gestion et l'implémentation.

## 3.7 Hadoop, "une solution pour le BigData"

Le modèle cxt-cube offre la possibilité d'adapter une analyse de données textuelles non structurées à des systèmes OLAP ; permettant de faire des agrégations sur les grandes masses documentaires. Dans cette section nous allons introduire une famille de système très répandue et qui se montre idéale parmi les solutions prometteuses pour la réalisation d'un système d'ADT basée sur les system OLAP et d'une haute scalabilité.
De nombreux projets l'Apache Software Foundation sont dédiés à fournir les services nécessaires au déploiement, à l'intégration et à la manipulation d'importants volumes de données structurées, semi-structurées ou non structurées. Parmi ces projets, le framework open source Apache Hadoop s'est montré révolutionnaire et il permet de traiter des grands volumes de données et de façon distribuée sur du matériel standard regroupés en une seule unité de traitement(un cluster). Hadoop permet aux entreprises d'exploiter rapidement les masses d'information et de réaliser les différentes applications d'aide à la décision ; le diagramme suivant simplifie une architecture des première versions de Hadoop :

La traçabilité des fragments est protégée par le couple clé/valeur où la fonction Map associe un ensemble de nouveaux couples clé/valeur aux nouveaux fragments.
l'étape Reduce s'occupe de la collecte des résultats, où les plus bas des nœuds font remonter leurs résultats aux nœud parents qui les avait sollicités . Celui-ci calcule un résultat partiel à l'aide de la fonction Reduce (réduction) qui retrouve le chemin de répartition par l'association de toutes les valeurs correspondantes via leurs paire clé/valeur unique (l'opération shuffle). Les nœuds qui ont assemblé les résultats de leurs sous-arbres, remonte l'information à leurs tours.
le schéma suivant illustre un example simple d'une tâche MapReduce :

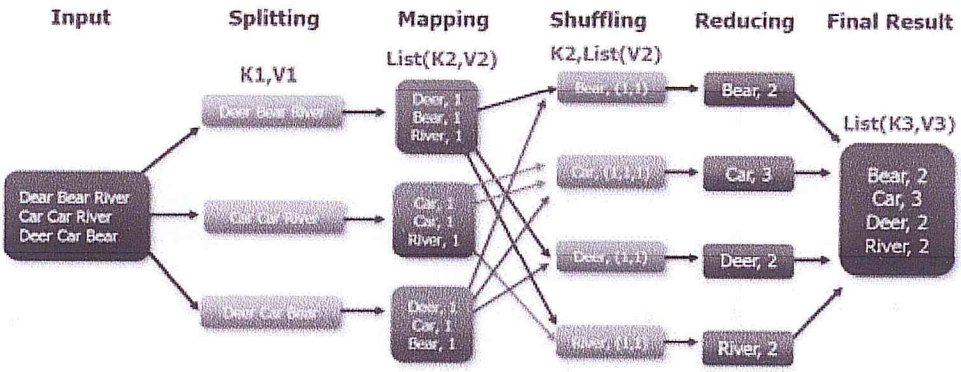



FIGURE 16 – L'algorithme Wordcount avec MR.[44]

La figure 15 donne un example classique du MapReduce wordcount(le 'hello world !' du MapReduce) qui est un algorithme pour compter les mots à partir d'une chaine de caractères. dans la phase pré-Map la tâche split découpe les données de l'entrée en trois fragments, où chaque ligne représente un bloc de données et qui reçoit son couple clef = (K le texte de la ligne)/valeur = (V,NULL) et elle sera attribué à un Mapper ; par la suite au niveau de chaque Mapper la fonction Map fragmente a nouveau les entrées et donne à chaque mot sa paire K = (le mot)/V = (1) ; la phase prochaine (pré-Reduce) shuffle regroupe les blocs par leurs paires identiques et donne à chaque Reducer l'un d'eux en entrée (pour cet exemple seulement ; dans d'autres cas, c'est plus compliqué et même les paire sont cryptées). finalement la fonction Reduce garde la clef unique et calcule la valeur = (la somme des valeurs) et renvoie le couple K,V.
Le MapReduce fournit une tolérance aux fautes à grain fin grâce à laquelle il peut redémarrer les nœuds ayant rencontré une erreur ou affecter la tâche à un autre nœud dans le cas où les fautes sont au niveau des nœuds esclaves ; et

garde périodiquement des checkpoints pour lancer à partir d'eux de nouveaux calculs si la faute est au niveau du nœud maître.
le framework MapReduce communique avec Java mais l'ajout de service Hadoop streaming permet de connecter plusieurs langages de programmation : Python, R, Scala, Ruby ...etc.

### les versions de MapReduce

Comme tout projet répandu, Le framework MapReduce a eu sa part de l'évolution et cette dernière a affecté son architecture et son mode de fonctionnement se qui a permit de réduire la complexité et d'améliorer l'environnement. Avant de parler des versions de MapReduce, nous tenons à préciser que les informations présentes dans cette section à-propos l'architecture d'un système Hadoop ne sont pas standards. les projets relatives à Hadoop et qui sont adoptés par la fondation Apache ou par d'autres organisations dépasses la centaine ; et sa implique la possibilité d'implémenter une infinité de combinaisons pour établir un cluster Hadoop selon l'utilité du dernier et les moyens du bord.

### MapReduceV1

la première version du MapReduce est composé de trois éléments :

- Resource Manager : est l'infrastructure qui gère l'allocation des ressource et l'ordonnancement des Jobs via le JobTracker.
- Framework : garantit l'exécution des différentes taches MapReduce (shuffle, Map, Sort ,Reduce) via le TaskTracker.
- API : est l'interface qui permet de développer une Application en MapReduce.

### Fonctionnement

Les Jobs MapReduce divisent le processus sur plusieurs taches nommées Mappers et Reducers où chacune est exécutée sur un nœud du cluster qui a un nombre limité de slots (des unités d'entrées) et qui sont divisés en deux catégorie Map slots et Reduce slots. le nombre des slots est défini par la capacité du TaskTracker à exécuter une tâche individuellement, la figure suivante simplifie le principe :

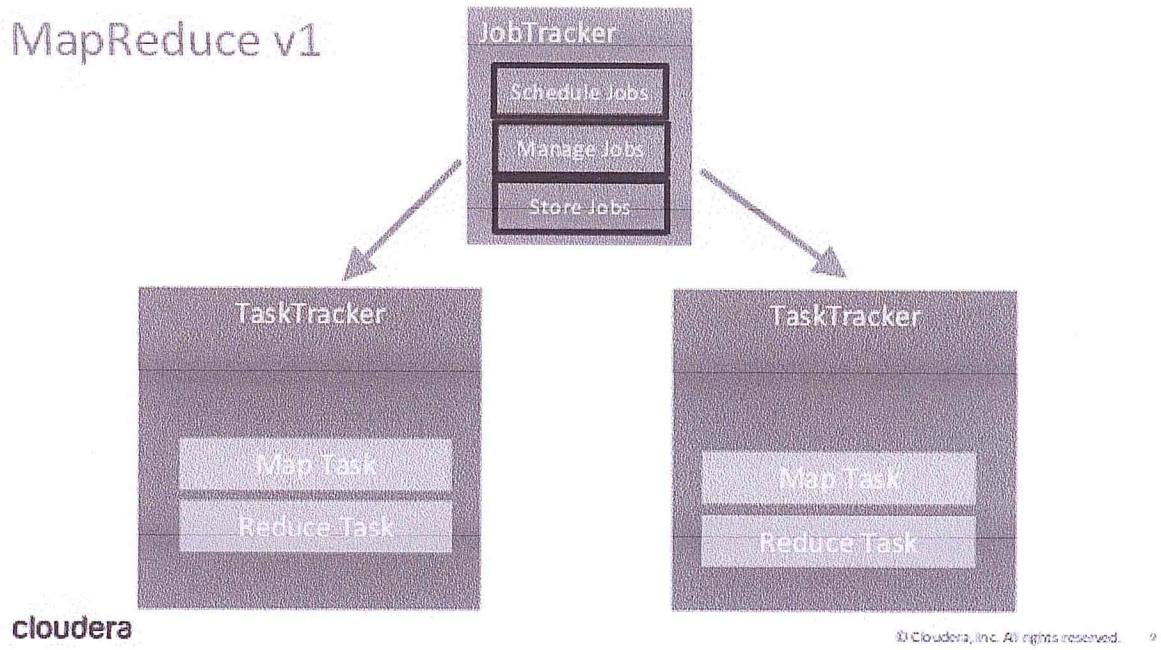



FIGURE 17 – Mode de fonctionnement en MRv1.[38]

Le JobTracker s'occupe d'allouer les ressources (réallouer s'il détecte une faute), coordonner l'exécution des Jobs, réserver et ordonnancer les slots. Par sa présence dans une seule machine et l'ensemble de fonctionnalité gérées par le JobTracker ; plusieurs points doivent êtres considérés :

- le SPOF est l'acronyme de Single Point Of Failure et qui veut dire si le JobTracker tombe en panne le Job en cours doit redémarrer.
- l'usage du HDFS est réservé que pour des Jobs MapReduce car le JobTracker est fortement intégré à toutes les unités de commandement.
- Problème de scalabilité, les DataNodes ne sont pas tous impliqués et exploités, car si tous les slots de Map sont occupés à exécuter plusieurs Jobs les Reduce slots restent inexploitables et vice-versa.

### MapReduceV2

La deuxième version a apporté une nouvelle politique par la séparation de la gestion des ressources et celle des taches MapReduce. Autrement nommée YARN (Yet Another Ressource Negotiator) n'emploie plus la notion de slot et il alloue les ressources à la demande. la nouvelle unité nommé Ressource Manager s'encharge de la plupart des taches effectuées dans la première version par le JobTracker et le cluster peut avoir plusieurs RessourceManagers(un seul actif, il tourne sur le nœud maître) et plusieurs ApplicationMasters.

cette nouvelle architecture supporte aussi les applications non-MapReduce. la figure suivante permet de voir les acteurs de la nouvelle version :

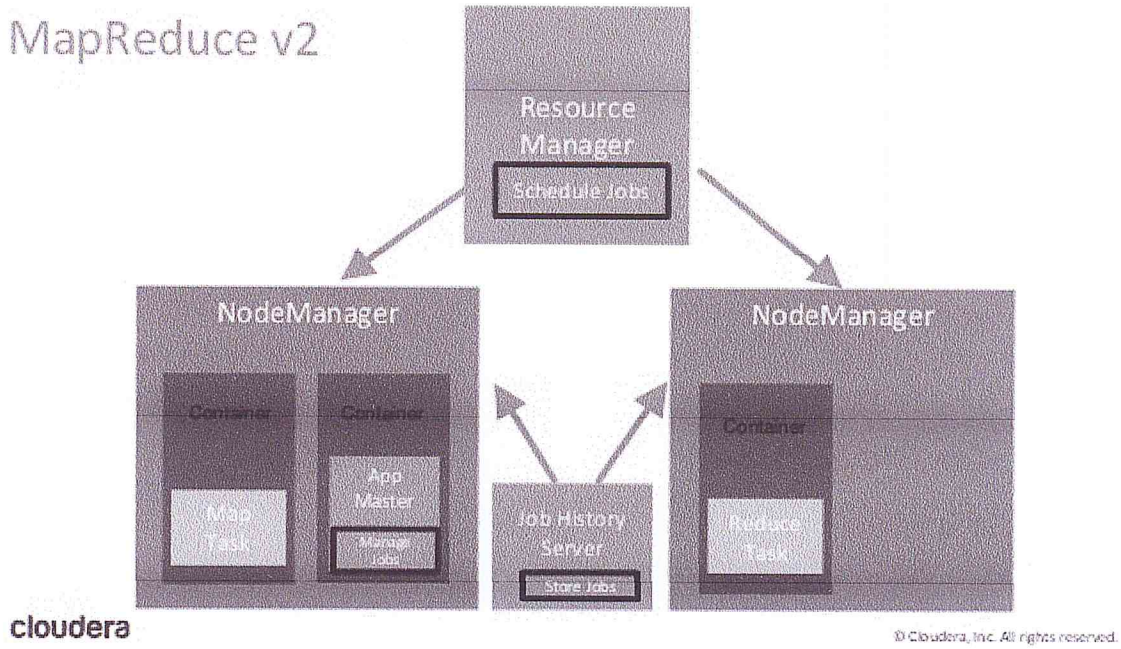



FIGURE 18 – Mode de fonctionnement en MRv2.[39]

### Fonctionnement

la menace du SPOF dans la première génération du Framework n'est plus la préoccupation dans celle la ; le RessourceManager ordonnance les ressources globalement et fait le partage entre les différentes applications. les NodeManagers communiquent avec le RessourceManager et reçoivent des requêtes pour créer des container qui préoccupent des ressources à la demande. la deuxième génération emploie un Application Master (AM) pour chaque application. un AM s'exécute sur un container et occupe d'autre pour exécuter l'application assigné.
la deuxième version a résolu les problèmes majeurs de la première, le tableau suivant compare entre les deux :

| Mesures de comparaison | **MapReduceV1** | **MapReduceV2** |
|---|---|---|
| Scalabilité | Le JobTracker veille sur toutes les tâches, 4000 nœuds au max et se limite à 40000 tâches | partage la gestion des tâches entre le ResourceManager et l'ApplicationMaster, supporte 10000 nœuds et 100000 tâches |
| Disponibilité | Haute Disponibilité du JobTracker (HA) | Haute Disponibilité du RessourceManager |
| Taux d'utilisation | taille fixe des slots Map / Reduce donne un faible taux | L'allocation est faite selon les besoins de l'application donnant un taux $> 70$ % |



FIGURE 19 – Comparaison entre MRv1 et MRv2.

### 3.7.2 HDFS

HDFS est l'acronyme de (Hadoop Distributed File System) qui est un système de gestion de fichiers distribué, extensible et portable, écrit en java comme la majorité des composants de l'écosystème Hadoop.Le HDFS regroupe les volumes de stockage des différents nœuds et qui seront reconnus comme une seule unité de stockage. Le stockage dans le HDFS est par bloc où chacun aura un nom unique, une taille paramétrable et permet de répartir le même fichier sur plusieurs nœuds. Un système de fichiers fonctionnel possède plus d'un DataNode (généralement), avec des données répliquées à travers eux. les DataNodes sont chargés du stockage. Au démarrage, un DataNode lance un processus démon et reste en attente pour se connecter au NameNode, l'établissement de la connexion le permet de répondre aux demandes du NameNode pour les opérations du système de fichiers.
Les applications clientes peuvent dialoguer directement avec un DataNode, une fois que NameNode a fourni les emplacements des données. De même, les opérations MapReduce seront exploitable dans des instances TaskTracker/RessourceManager près des DataNodes,et qui parlent directement aux DataNodes pour accéder aux fichiers. Les instances TT/RM peuvent, en effet, être déployées sur les mêmes serveurs qui hébergent des instances DataNode, de sorte que les opérations MapReduce sont effectuées à proximité des données.Le diagramme suivant montre ces dépendances :

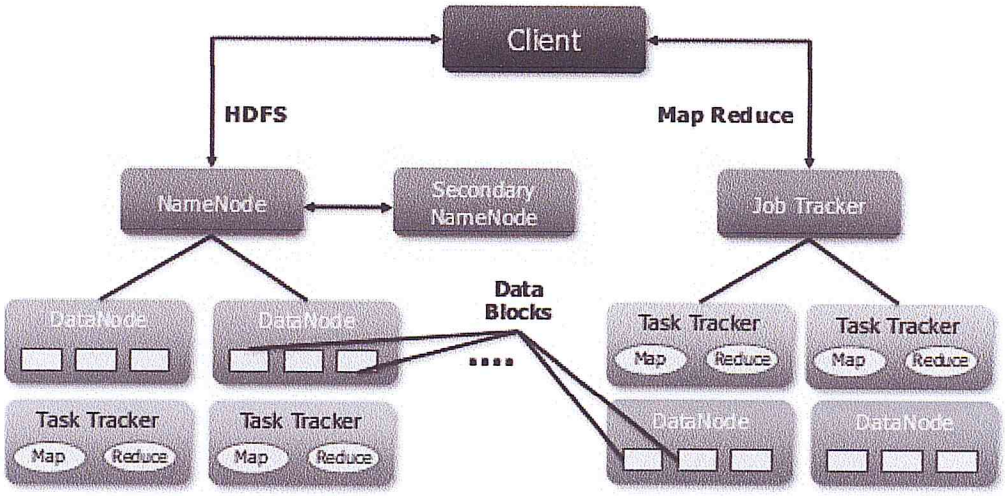



FIGURE 20 – Les relations entre les couches MapReduce et HDFS.[40]

si l'un des NameNode causera une panne , les données sont perdues, la gestion de fichiers est neutralisé ce qui affecte les opérations en cours. L'alerte sera transmise et déclenchera le processus de diagnostique et de récupération. le système retrouve la réplique du bloc, ensuite il choisit au hasard un autre nœuds pour créer la nouvelle réplique afin de garder trois copies ('NameNode High Availability'). Dans une autre situation où le NameNode est défaillant(la conclusion d'une longue attente d'un sœuil configurable, généralement 1800 secondes), le système procède à dupliquer les informations du NameNode et lance le NameNode de réserve.
Loin des composants de base de Hadoop d'autres outils existent pour permettre la simplification de l'interaction avec le système de stockage HDFS (extraction/chargement de données), des SGBD pour la gestion de données (Hive, Pig, Cassandra ...etc), des connecteurs pour les différents langages de programmation (RHadoop project), des outils de monitoring (Ambari, cloudera-manager) et des alternatives pour le MapReduce (Spark, Tez). Dans le reste de la section nous allons parlé d'un outil qui répond a certains besoins de stockage et gestion de données.

### 3.7.3 Hive

Le modèle de programmation MapReduce n'existe que pour lever les complexités autour des données. la nouvelle problématique réside dans les exigences liées à l'interaction avec ce patron de conception ; qui requiert une programmation personnalisée du bas niveau et qui deviendra difficile à maintenir et réutiliser. d'autre part les langages du haut niveau étaient négligés

par le framework (ex : SQL). Ces conditions ont permit à plusieurs projets d'administration et traitement de données de voir le jour et en compte parmi eux : Pig Latin, Hive, Spark SQL, JAQL ...etc.
Hive est un environnement qui permet de définir une structure pour les données non structurées ; un critère qui simplifie le processus d'analyse. par son langage déclaratif du haut niveau HQL qui est proche du SQL, il facilite le traitement des données à grande échelle. les requêtes sous Hive sont interprétées en Jobs MapReduce et exécutées sur un cluster Hadoop.[18]

### Concepts

Hive offre un accès aux données qui se fait via des tables structurées. Il se matérialise par la création d'un plan d'exécution de la requête HQL qui se traduit par la création et l'exécution d'un job MapReduce ; il offre aussi la possibilité aux familiers du modèle parallèle MapReduce de pouvoir effectuer des tâches de type MapReduce dédiés aux traitements spécifiques aux données non supportés par HQL.[18] L'usage de Hive via un pilot JDBC ou ODBC permet de définir des UDF(User Defined Function), UDAF(User Defined Aggregate Function) et UDTF(User Defined Tabular function), qui sont faciles à maintenir et réutilisable.

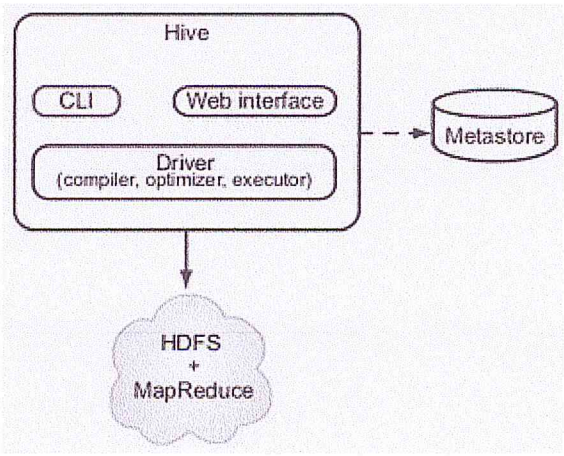



FIGURE 21 – Architecture simplifiée de Hive.[41]

### Modèles de données

— DATABASE|SCHEMA, pour la création d'une base de données / un schéma (actuellement elles représentent la même entité , je crois que SCHEMA est en cours de développement et bientôt elle sera différenciée).

- TABLE, des tables qui sont stockées dans : le HDFS, une Bucket(stockage en objet ex : Amazon Web Service S3 bucket) ou bien dans le système de fichier local où elle est référencées par des métadonnées.
- Types primitif :(les classiques) INT, FLOAT, DOUBLE, BOOLEAN, STRING, TIMESTAMP, ...etc.
- Type complexe : ARRAY, MAP, STRUCT.

### Les Requêtes HQL

Le Hive Query langage est proche du langage SQL et il supporte les requêtes de : sélection, jointure, agrégation, union et les sous-clauses (ex :where, case). Hive permet à travers le langage de manipulation de données (DML) le chargement des données dans les tables. Cette manipulation par les commandes load et insert. il supporte aussi le DDL (Data Definition Language)qui permet de créer, réparer, modifié et visualiser les différentes structures complexes de Hive. le DDL s'occupe également de la sérialisation des données pour le stockage ou l'extraction et il supporte les modes suivants :

- TEXTFILE : fichier texte CSV ou TSV COMMA/TAB Seperated Values.
- SEQUENCEFILE : fichier de séquences compressé.
- ORC(Optimized Row Columnar) : via ce format de fichier Hive garanti les propriétés ACID, un stockage en colonnes optimisées et offre un optimiseur de coût CBO (Cost-Based Optimizer).
- PARQUET : un stockage efficace créé à partir de structures complexes permettant la compression de données en colonnes.
- Avro : est un framework de stockage Objet et de communication ; il utilise le format JSON pour définir et sérialiser les données.
- RCFILE : stockage en colonnes pour les bases relationnelles qui est conçu pour définit l'emplacement de données dans un cluster utilisant le MapReduce. La structure de ce format inclut le stockage des données compressées et des des techniques d'optimisation pour la lecture.
- Stockage et création des tables non natives au niveau de HBase, Druid ou Accumulo.[19]

### Architecture de Hive

L'ensemble de services offerts par Hive, repose sur une architecture complexe ; le diagramme suivant montre ses composants :

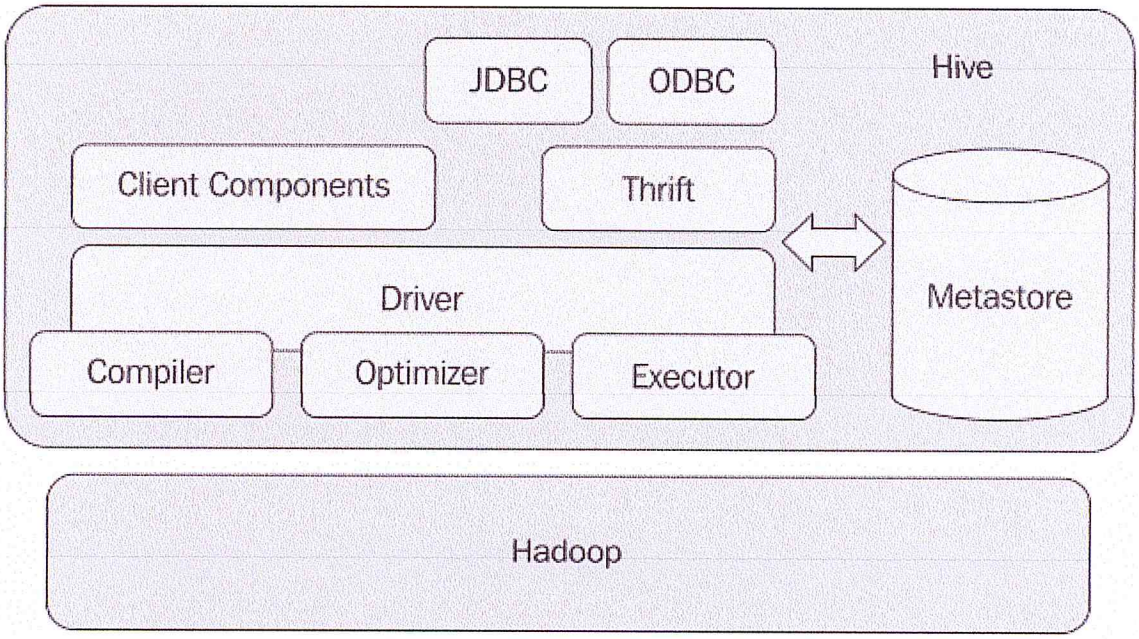



FIGURE 22 – Architecture de Hive.[42]

— Thrift server : facilite le développement de services multi-langage évolutif, combine une pile de logiciels avec un moteur de génération de code pour créer des services qui fonctionnent de manière efficace et transparente entre C/C++, Java, Python, PHP, Ruby, R, JavaScript, Node.js et d'autres langages de programmation. En soumettant une requête HQL via un client Thrift, JDBC ou ODBC elle sera traduite et transmise par le serveur Thrift.
— Thrift client : permet d'écrire en différents langages : Ruby, Scala, R ...etc.
— JDBC (Java DataBase Connectivity) : comme la majorité des base de données relationnelles/NoSQL, Hive fournit un pilote (JDBC driver). La connexion au serveur Thrift se fait via un processus séparé, en faisant appelle à une interface implémentée par le client Thrift Hive.
— ODBC (Open DataBase Connectivity) : Une API présente dans tous les langages et avec le pilote (ODBC Driver) les applications peuvent connecter au serveur Thrift.
— Hive Metastore : nommé aussi HCatalog, est le référentiel centrale de hive ; une base de données relationnelle (PostgresSQL par défaut) qui contient les métadonnées sur les objets crées.
— Driver : composé de trois éléments : Compilator, Executor et Optimizer.

* Compilateur : en analysant la requête, il fait une analyse sémantique avant de créer un plan d'exécution DAG (Directed Acyclic Graph).
* Exécuteur : exécute le plan du compilateur.
* Optimiseur : optimise de plan d'exécution.

Grâce à son langage de requête HiveQL, Hive apporte des concepts de bases de données SQL et relationnelles à Hadoop. Le cas d'utilisation principal pour Hive est l'entreposage de données et l'analyse de requêtes pour des applications telles que Business Intelligence. Les composants de support de Hive sont conçus pour aider ce cas d'utilisation. Dans la section suivante nous allons voir les privilèges et les limites spécifiques à la réalisation d'un entrepôt de données avec Hive.

### 3.7.4 OLAP et Hadoop

la construction d'un entrepôt de données est un projet massif, il y a de différentes application, méthodes et théories. La façon la plus simple de décrire un entrepôt de données est de se rendre compte qu'il s'agit de schémas, de faits et de dimensions en étoile. La façon de créer ces éléments dépend vraiment d'un bon plan de route qui comprend les sources de données, les processus d'extraction, de transformation et de chargement. Hive est apparu comme une technologie de base pour répondre aux besoins, il supporte les données non structurées. Il est beaucoup plus un outil d'intégration, de transformation et de recherche rapide par rapport à un entrepôt de données légitime de l'époque avant Hadoop, mais avec des limite. le bon outil pour le bon travail n'est pas toujours disponible ; dans un autre cas l'outil est disponible mais les coûts sont en jeu. Sa arrive souvent de construire, concevoir ou développer un projet en utilisant un outil qui pourrait ne pas être le meilleur pour le travail, mais on peut arriver à le pousser à ses limites. Dans cette section nous allons parler d'une solutions OLAP coexistante avec Hadoop.

#### Apache Kylin

Apache Kylin est un moteur à analyse distribuée conçu pour fournir une interface SQL et une analyse multidimensionnelle (MOLAP) sur Hadoop prenant en charge des ensembles de données extrêmement volumineux. le projet apporté par eBay est open source.
Apache Kylin permet d'interroger des données massives à une latence sous-seconde et en 3 étapes :

— Identification d'un schéma en étoile sur Hadoop.
— Création du Cube à partir des tables identifiées (via une interface qui permet de voir les tables de hive).

— Résultats en sous-seconde d'une Requête avec ANSI-SQL via ODBC, JDBC ou RESTful API(l'API de kylin qui utilise le serveur tomcat7).

### C'est quoi Kylin

Kylin est un Moteur OLAP extrêmement rapide à l'échelle ; conçu pour réduire la latence des requêtes sur Hadoop pour plus de mille milliards de lignes de données. Agissant sur Hadoop, Kylin supporte la majorité des fonctions SQL-ANSI. Il est plus rapide que Hive pour les mêmes masses de données et facilite l'interaction avec les données de Hadoop avec une latence inférieur à la seconde. Sous Kylin l'utilisateur peut définir un modèle de données pour construire un cube MOLAP. Il offre aussi des interfaces d'intégration transparente avec les différents outils de BI comme Excel, powerBI, Tableau et bientôt avec Microstrategy.

### Architecture

les composants de Kylin sont regroupés en cinq catégories :

— Kylin Core : la partie fondamentale du moteur Kylin OLAP et qui comprend les métadonnées, le moteur de requête, le moteur Job (routage/exécution) pour exécuter l'ensemble de la pile et faire le stockage . Il comprend également un serveur REST pour répondre aux demandes des clients (via l'API REST).

— Extensions : plugins pour prendre en charge les fonctionnalités supplémentaires

— Intégration : Prise en charge de la gestion du cycle de vie pour l'intégration avec Job Scheduler, ETL, Monitoring and Alerting Systems.

— Interface utilisateur : permet aux utilisateurs tiers de créer une interface utilisateur personnalisée au sommet de Kylin.

— Pilotes : les Driver ODBC et JDBC supportent de différents outils et produits, tels que Tableau, R, Python.

le schéma suivant montre l'Architecture de kylin :

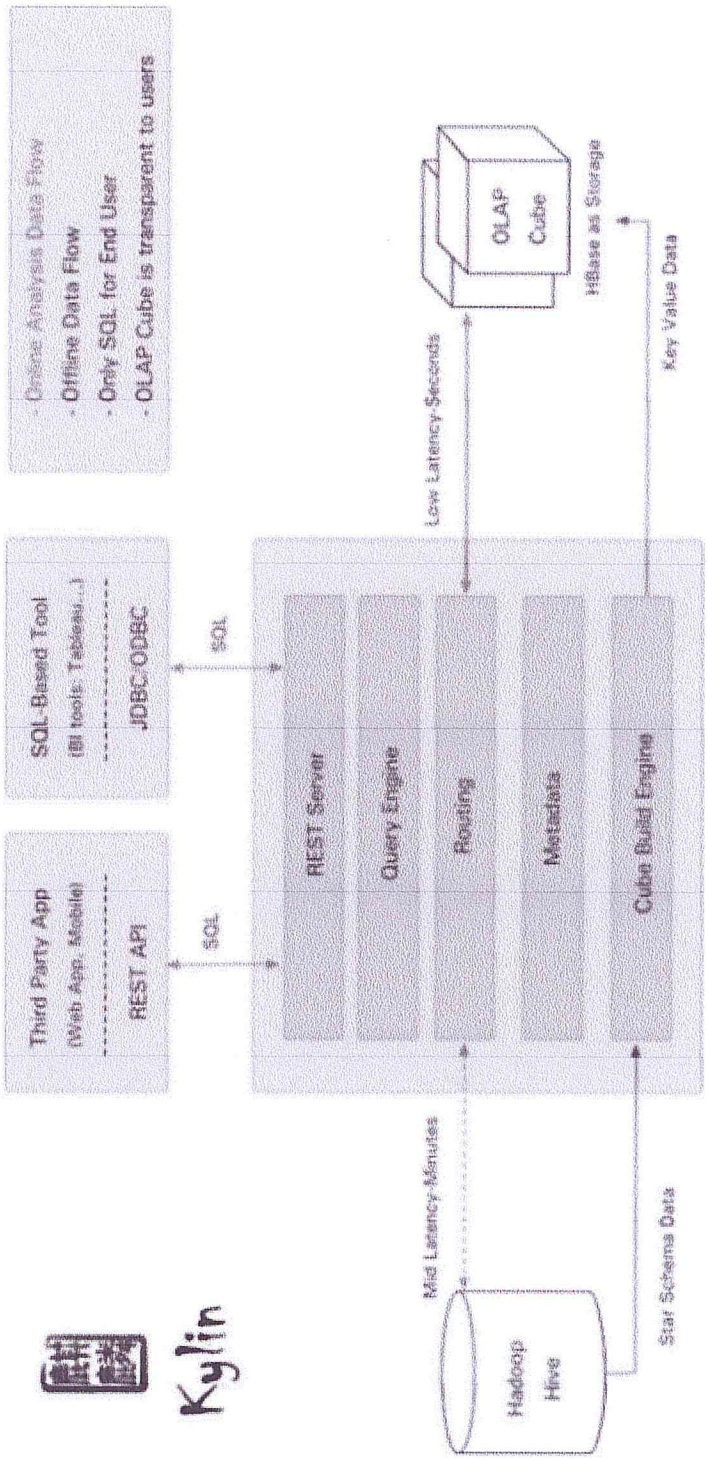



FIGURE 23 – Architecture de Kylin.[43]

**Caractéristiques et avantages :**

- Gestion et suivi des tâches (self-monitoring).
- Support de compression et de codage.
- Rafraîchissement incrémental des cubes.
- Utiliser le coprocesseur HBase pour la latence des requêtes.
- Capacités de requêtes approximatives et précises pour un nombre distinctif
- Capacité de requête top-N approximative.
- Interface Web simple pour gérer, construire, surveiller et interroger des cubes.
- Capacité de sécurité pour définir ACL au niveau Cube / Project.
- Prise en charge de l'intégration LDAP (Lightweight Directory Access Protocol ) et SAML (Security Assertion Markup Language).

Apache Kylin est un outil puissant et prometteur qui facilite la création des cube MOLAP avec une haute scalabilité et une latence optimale pour les solutions BI de l'avenir.

## 3.8 Conclusion

Dans ce chapitre, nous avons mi le point sur les généralités des bases de données NoSQL ; un mouvement technologique qui a su faire naitre des solutions pour un avenir prometteur de l'entreprise. les propriétés ACID d'une base relationnelle ne permettent pas d'avoir la scalabilité requise vu les masses et la qualité de données disponibles actuellement. Le théorème de CAP parle des caractéristiques vitales permettant de déployé l'architecture nécessaire d'un SGBD, selon l'utilité de ce dernier et la complexité des données à gérer. de ce constat plusieurs modèles sont apparus en adoptant les propriétés BASE, celles qui tolèrent que les propriétés ACID soient gérées avec résilience ; ainsi que pleins de solutions hybrides qui conservent les propriétés ACID partiellement.

Après, on a vu les technologies de pointes nécessaires pour notre entrepôt de données ; l'innovation de Hadoop et ces œuvres qui ont apporté la solution aux applications sensibles à tous problèmes de latence élevée. l'éléphant constitué essentiellement du framework MapReduce et le Stockage distribué avec le système de gestion de fichier HDFS , le data warehouse prometteur Hive et le projet Kylin qui le succède pour surpasser ses limitations.

Chapitre IV

# Conception et architecture générale

# 4 Conception et architecture générale

Dans l'état de l'art, nous avons recueilli les informations nécessaires sur les systèmes d'analyse en ligne classiques, l'approche CXT-Cube que nous allons utiliser pour la réalisation de notre système, et finalement le chapitre qui donne une vision sur le NoSQL et les différents concepts relatifs à l'écosystème Hadoop.
Pour la réalisation du système nous allons prendre un exemple issu des pratiques courantes des entreprises modernes qui est le recrutement en ligne (*E-Recruiting*) ; ce dernier et par sa nature, nous permet d'implémenter un modèle assez représentatif pour l'approche, et avec une collection documentaire testable et ajustable.
la fonctionnalité principale est de faire une analyse en ligne en utilisant un système OLAP pour faire le classement (*ranking*) d'une collection documentaire de CVs spécifique à une requête formée par le recruteur. Cette dernière représente le profil idéal et elle sera composée de trois parties : le sujet qui regroupe les maîtrises et les compétences demandées par le recruteur, l'adresse du candidat (localisation) et finalement la date d'envoi de son CV. ex("base de données, programmation","Janvier 2017","Paris").
Ce chapitre regroupe la conception qui détaille le cheminement des taches et d'une autre partie pour l'architecture de l'infrastructure d'accueil qui sera un cluster R/Hadoop/Kylin effectuant des calculs et assurant un stockage en distribuer.

## 4.1 Conception de la solution

Dans cette section, nous allons définir la conception du système et les tâches permettant de réaliser les opérations formelles du processus d'entreposage d'une collection (de documents et ainsi que la formulation et l'envoie d'une requête pour interroger le cube.

### 4.1.1 E.T.L

Pour le processus de l'extraction, la transformation et le chargement, nous allons implémenter les trois opérations au niveau d'un environnement R, et qui permet de faire les prétraitements, le calcule des fréquences d'apparition (avec la propagation de pertinence), la récupération des métadonnées ainsi que le chargement via l'un des pilotes JDBC, ODBC, ou par une connexion directe à Hive avec RHive. Nous allons introduire les outils utilisés dans le prochain chapitre. les étapes du processus E.T.L utilisées sont cartographiées dans le diagramme suivant :

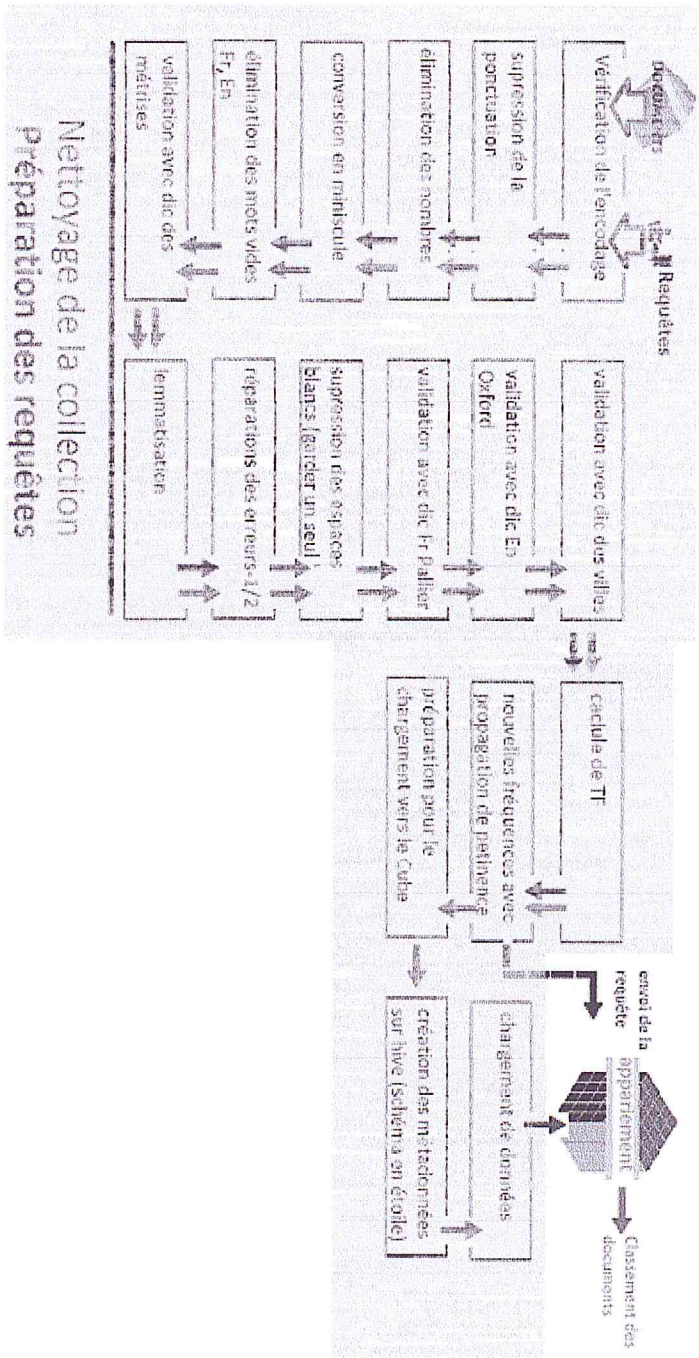



FIGURE 24 – E.T.L/requêtage .

la figure illustre le cheminement des opérations qui sont regroupées en trois blocs ; le premier destiné aux opérations de nettoyage qui sont similaires pour

les documents et les requêtes, le deuxième contient les étapes de l'indexation et la représentation et dans le troisième bloc les chemins seront séparés, celui des documents et l'autre pour les requêtes. L'entreposage des documents se fait après l'indexation et la collecte des métadonnées, la validation d'une requête sera effectué par la formulation d'une requête en HQL ou par l'usage d'une UDF(3.7.3).

### E.T.L Prétraitement de la collection

La représentation des documents est la clé du succès pour la validité des résultats ; il existe de différents protocoles pour l'effectuer. Essentiellement les étapes d'une indexation automatique sont : Tokenization, éliminations des mots vides, espaces blancs et nombres et de faire la normalisation (racinisation/stemming) avant de pouvoir transformer le document. Durant l'implémentation de cette phase, nous avons remarqué qu'une indexation automatique, endommage le corpus et ne permet pas d'avoir une représentation juste des documents ; donc nous allons faire une indexation semi-automatique. les étapes de prétraitement sont les suivantes :

1 résolution des problèmes d'encodage.
2 suppressions de la ponctuation.
3 éliminations des nombres.
4 conversions de texte en minuscule.
5 éliminations des mots vides.
6 suppressions des espaces blancs.

Durant l'opération et en parcourant la nouvelle liste des termes ; nous avons remarqué qu'il y a des termes potentiellement pertinents, écrits faux et avec des modèles répétitifs ; donc nous avons ajouté d'autres étapes.

7 validations de termes avec un dictionnaire français(Pallier).
8 éliminations de mots vides anglais.
9 validation de termes avec un dictionnaire anglais('projet sur github' https ://github.com/dwyl/english-words).
10 validations de termes représentant des localisations(liste des villes, comptant plus de 5000 habitants 'service de l'organisation geonames').
11 validation des termes présents dans la hiérarchie de concepts (construite pour la dimension sujet et elle compte 198 termes).
12 récupérations des termes potentiellement faux.
13 réparations du premier modèle d'erreur (les termes avec des lettres accentuées).
13.1 création d'un nouveau dictionnaire français qui contient les mots avec des lettres accentuées.
13.2 transformation des lettres accentués vers leurs racines (ex : é,ê,è ->

e) au niveau du nouveau dictionnaire.

13.3 transformation de la liste (12) comme dans l'étape précédente.

13.4 transformation des termes suivant les formes originales du dictionnaire Fr (ça fonctionne parfaitement avec des manipulations simples des chaînes de caractères ex : algébre -> algebre -> algèbre).

14 réparation du deuxième modèle d'erreur (les mots collés ex : htmlcssjava).

14.1 récupération des combines de termes possibles et qui doivent être valides par rappot aux dictionnaires utilisés et suivant l'ordre (dictionnaire français, anglais, localisations, hiérarchie de concepts et finalement le dictionnaire complet qui est compsé des dictionnaires précédents), et si la chaîne ne contient pas des fragments non significatifs. si on trouve une chaine de mots valide par rapport au premier dictionnaire le processus s'arrete. Pour l'exemple de la chaine 'htmlcssjava' le processus s'arrête au niveau du quatrième dictionnaire.

15 lemmatisation.

à présent la collection/requêtes sont prêtes pour la deuxième grande étape qui consiste à calculer les poids de termes ($TF$ : Fréquence d'apparition d'un terme dans une unité textuelle) puis avec la propagation de pertinence ; et seulement pour les termes figurant dans la hiérarchie de concepts et seulement pour la dimension sujet (*Topic*).

Pour la deuxième dimension localisation et qui sera une dimension de métadonnées ; un choix justifié par l'impossibilité de concevoir une dimension avec des dizaines de milliers d'attributs qui engendre un temps de calcule énorme. la dimension sera composée donc de six niveaux hiérarchiques : pays et 5 niveaux administratifs (suivant le système des divisions administratif Français). la recherche des niveaux administratifs pour une localisation, s'effectue à l'aide du service geonames (service de localisation offert par l'organisation Américaine GeoNames et accessible via une API).

Il reste de récupérer la date d'envoi avant de procéder au chargement. la figure suivante regroupe les informations extraites d'un document :


FIGURE 25 – Les informations extraites des documents.

après avoir récolté les informations présentes dans la figure, la collection sera représentée en utilisant l'approche CXT-Cube.

### 4.1.2 E.T.L L'entrepôt Hôte et le Schéma en étoile

Comme le modèle CXT-Cube l'indique, notre entrepôt sera fondé autour d'un schéma en étoile composé d'une table de faits centrale et des tables de dimensions sémantiques et celles des métadonnées.
Dans un schéma en étoile, les tables de fait sont composées généralement des clés étrangères et d'enregistrements pour des faits mesurables. Dans notre cas de CVs nous avons opté pour une table de fait contenant des clés étrangères reliant les tables de dimensions à la table de fait. cette dernière contient une mesure nommée pow qui ne change rien à-propos l'approche, mais elle permet de réduire le temps de calcul. Notre mesure n'est pas liée à un événement bien précis ; de plus le calcul des classements, utilise tous les niveaux de granularité de la dimension sémantique(Topic). notre système OLAP sera fondé autour du schéma en étoile suivant :

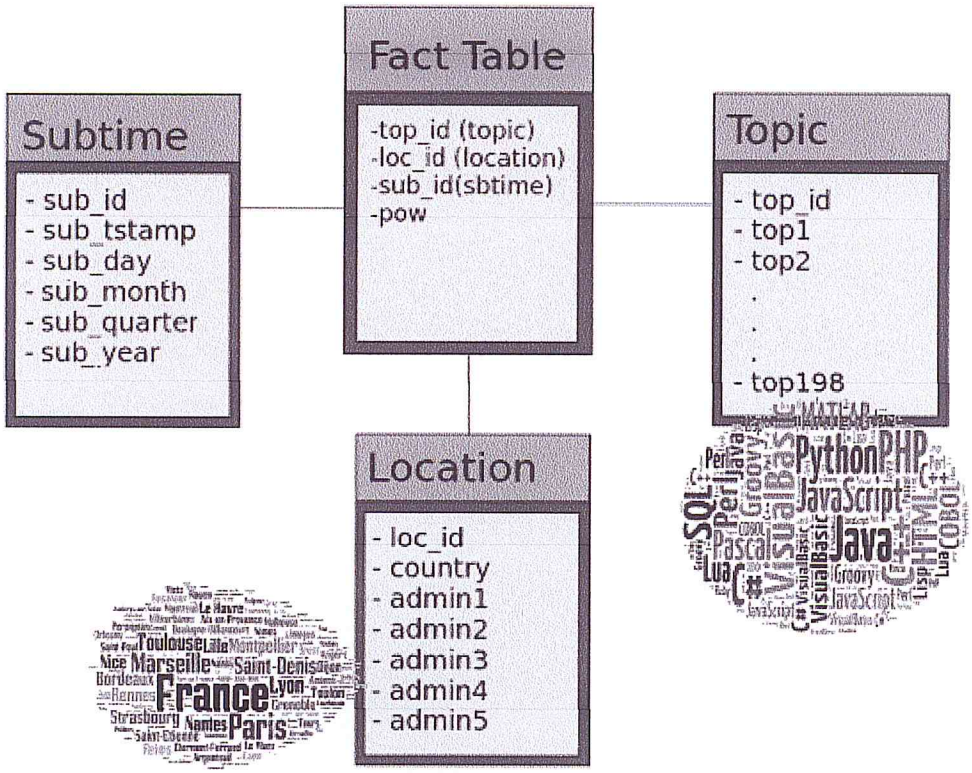



FIGURE 26 – Le schéma en étoile.

nous commençons par l'explication de la mesure pow qui est présente dans la table de fait. Le calcule de similarité requête/document s'effectue par le cosinus suivant (2.4.5) :

$$Sim(d,q) = cosa = \frac{\sum_i {w_{ti}}^2 * w_{q_{ti}}^2}{\sqrt{\sum_i w_{ti}^2 * \sum_i w_{q_{ti}}^2}}.$$

ce calcul fait usage des poids de termes présents dans la requête et ceux présents dans le document ; et si on veut optimiser le temps de calculs on doit voir en profondeur comment l'opération sera effectuée. pour notre projet nous allons utiliser une base de données Hive au niveau d'un cluster Hadoop et durant les calculs, les tables seront chargées au niveau de la mémoire ; où le calcul de la valeur $\sum_i w_{ti}^2$ sera répété à chaque fois(requêtage) en sachant que la mesure est statique pour chaque document et ne nécessite pas les valeurs issues de la requête indexée ; Donc nous allons stocker la mesure (nommée pow) au niveau de la table de fait pour ne pas épuiser les ressources. $pow = \sum_i w_{ti}^2$,où$w_{ti}$ est le poids du terme i dans un document $d$. Notre table

de fait sera définie et composé de *pow* et de clés étrangères vers les trois tables de dimensions.
pour ces trois dernières, la première dimension sujet(topic) est sémantique, ell est composé d'une clé (l'identifiant du document) et des poids de termes appartenant à une hiérarchie de concepts composée de 198 termes. les noms de colonnes sont : top_id(clé), top1,top2,...,top198.
la deuxième dimension sera donc conçue en tant que dimension de métadonnées et elle contient une clé et six niveaux hiérarchiques, commençant par le niveau pays jusqu'au dernier niveau de granularité qui est le 5 éme niveau administratif.
la dernière dimension des métadonnées est représentée par une table contenant la date d'envoi du CV. elle est constitué d'une clé primaire (identifiant du document) et d'autre colonnes du type TIMESTAMP, jour, mois, trimestre et année.
la figure suivante montre, le cube de données :

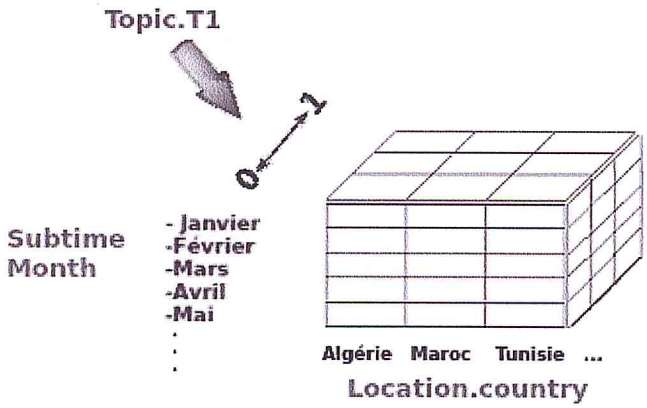



FIGURE 27 – Le Cube de données.

Un aperçu de la vue multidimensionnelle du cube de données, qui est composé de trois dimensions, Topic(sujet), Subtime(data d'envoie) et Location(localisation) avec les niveau de granularité : T1, country(pays) et month(mois) respectivement. pour la première dimension les valeurs seront toujours des nombres réels compris entre 0 et 1.

### 4.1.3 Gestion des requêtes

La requête sera divisée en trois parties séparées, où chaque partie utilise l'une des dimension. Cette mesure de séparation nous permet de former les requêtes rapidement, avec précision et d'exclure les dimensions qui n'ont pas d'impact sur le résultat final. par exemple une requête de 'classement des

candidat ayant des compétences en apprentissage automatique en France' en ignorant l'importance de la date d'envoi ; les valeurs de cette dernière n'auront pas d'impact sur le résultat et la dimension sera exclue des calculs.
La première entrée est pour les compétences ; permettant de calculer la similarité document/requête . les dimensions de métadonnées seront prêtes à être utiliser pour faire les restrictions.
une restriction suivant une localisation n'est pas pratique dans certaines conditions et pour certains profiles à recruter ; pour cette raison, nous proposons qu'elle soient ouverte à une nouvelle perspectives. Notre vision est de prendre en charge la distance administrative entre le profile demandé et le document, où les requêtes seront conçues comme suit($loc(d, q)$ pour le document $d$ et la requête $q$) :

- définition des niveaux de granularité à considérer (ex : pays,niveau admin1, niveau admin 2).
- définition de la localisation suivant le niveau de grain le plus fin (niveau admin 2, ex : Blida).
- comparaison descendante vers les niveaux hiérarchiques inférieurs, où le calcule s'arrête quand les deux valeurs ne seront plus identiques (valeurs booléennes 1/0).
- la valeur finale pour la localisation : la division de nombre de niveaux identiques par le nombre des niveaux hiérarchiques (comprise entre 0 et 1).
  ex : pour une requête de 3 niveaux administratifs : 'pays = algérie' 'na1 == blida' 'na2 == blida', et un document 'pays = algérie' 'na1 == blida' 'na2 == cheffa' le résultat est : 2/3

pour la dernière dimension la restriction sera stricte et les documents qui ne sont pas conformes seront exclus du classement finale.
les requêtes possibles initialement sont :

- le calcul de classement avec la dimension sujet :
  $Rank(d, q) = \alpha sim(d, q) + \beta sim(d, p)$ pour une localisation $l$ et durant une phase temporelle $t$ qui peut être représentée en SQL par une valeur unique (ex : durant : Janvier / 17 Janvier - 02 Février).
- le calcul classement avec la dimension sujet :
  $Rank(d, q) = \alpha sim(d, q) + \beta sim(d, p) + loc(d, q)$ durant une phase temporelle $t$.

la requête finale sera générer par une fonction suivant le paramétrage voulu, envoyé utilisant l'un des connecteurs classiques JDBC, ODBC ou avec RHive qui est un system d'application pour R et Hive, et que nous allons introduire sa mise en œuvre dans le chapitre suivant.

## 4.2 Composition du système

Dans la section précédente nous avons parlé de la chaîne de traitement document/requête et dans celle-là nous allons décrie la composition de notre système et les unités chargées de chaque taches.
Notre système est un cluster informatique composé de quatre machines (nœuds) 'Ubuntu14.04'. Cette structure héberge un cluster Hadoop (HDR) qui regroupe les services suivants :

1 **HDFS** : (veuillez voir dans ce document[3.7.2])

1.1 **DataNode** : (veuillez voir dans ce document[3.7.2])

1.2 **NameNode** : (veuillez voir dans ce document[3.7.2])

1.3 **NFSGateway** : Cette passerelle permet à HDFS d'être monté dans le système de fichiers local.

1.4 **SNameNode** : Le NameNode stocke les modifications apportées au système de fichiers en tant que journal annexé à un document de gestion de fichiers natif et contient les modifications. Quand un NameNode démarre, il lit l'état HDFS à partir d'un fichier image, fsimage, puis applique des modifications à partir du fichier journal des modifications. Il écrit alors un nouvel état HDFS sur la fsimage et commence l'opération normale avec un fichier d'édition vide. Puisque NameNode fusionne fsimage et édite les fichiers uniquement pendant le démarrage, le fichier journal des modifications pourrait être très important dans un cluster occupé. l'effet secondaire de la taille immense d'un fichier d'édition est la longue durée du prochain redémarrage de NameNode.
Le SNameNode fusionne périodiquement la fsimage et les fichiers journaux d'édition et conserve les modifications de la taille du journal dans une limite. Il est habituellement exécuté sur une autre machine que le NameNode primaire puisque ses exigences de mémoire sont sur le même ordre que le NameNode principal.

2 **YARN + MapReduce2** [3.7.1]

2.1 **App Timeline Server** : Le stockage et la récupération des informations actuelles et historiques des applications de manière générique sont résolus dans YARN via le serveur Timeline (anciennement appelé Generic Application History Server). Cela comporte deux responsabilités :
- Informations génériques sur les applications complétées
- Informations par image relatives aux applications en cours d'exécution et complétées

2.2 **NodeManager** : [3.7.1]

2.3 **ResourceManager** : [3.7.1]

3 **Tez** : Le projet Apache Tez est un framework d'application qui permet de diriger des tâches par un graphe complexe acyclique de traitement des données. Il est actuellement construit au sommet de YARN. Tez garanti une API de définition des : flux de données expressives, modèles d'exécution "entrée->processus->sortie" flexibles, types de données agnostiques et la Simplification du déploiement. Tez amélioré la performance d'exécution par : un gains supérieur à MapReduce, la gestion optimale des ressources et planification de la reconfiguration au moment de l'exécution.

4 **Hive** [3.7.3]

4.1 **Metastore** : [3.7.3]

4.2 **HiveServer2** : [3.7.3]

4.3 **WebHCat Server** : est le serveur de l'API WebHCat, cette dernière offre une interface de monitoring permettant d'exécuter et surveiller des Job MapReduce, Hive, Pig, ...etc.

5 **HBase** : est une base de données NoSQL orientée colonnes d'un accès aléatoire, en temps réel dédiée aux Big Data. Ce projet vise à héberger de très grandes tables avec des millions de colonnes sur des clusters de matériel de base. HBase est une base de données non-relationnelle distribuée et basée sur Google Bigtable : un système de stockage distribué pour les données structurées.

5.1 **Master** : dans un mode opérationnel maître/esclave, HBase Master gère les requêtes de HBase, fait la coordination et distribue les tables automatiquement sur les différents nœuds quand elles deviennent très grandes (AUTO-SHARDING).

5.2 **RegionServer** : une région est l'unité de scalabilité horizontale qui est un sous-ensemble des données de la table; contenant des lignes contiguës triées qui sont stockées ensemble. Au départ, il n'y a qu'une seule région pour une table. Lorsque les régions deviennent trop grandes après avoir ajouté plus de lignes, la région est divisée en deux sur la touche centrale, créant deux moitiés à peu près égales. En regardant en arrière sur l'architecture HBase, les esclaves sont appelés serveurs régionaux. Chaque serveur régional est responsable de servir un ensemble de régions.

5.3 **Phoenix Query Server** : fournit un autre moyen d'interaction avec HBase. Cela permettra l'accès à partir d'autres environnements que la JVM. Il offre un pilote JDBC pour connecter via un 'Phoenix query Client', ouvert à exécuter des opérations à partir de chaque nœuds du cluster et offre une interface SQL pour HBase permettant de réaliser des systèmes OLTP et OLAP.

6 **Pig** : Apache Pig est une plateforme pour l'analyse des Big Data

offrant un langage interprété de haut niveau qui permet de faire des traitements parallèles pour résoudre des problèmes complexes autour des grands volumes de données.

7 **ZooKeeper** : est un service centralisé pour le maintien des informations de configuration, la nomination(ex : HDFS est le super utilisateur de l'HDFS), et la fourniture de synchronisation distribuée et la gestion de services par les groupes d'utilisateurs et pour tous les services du cluster.

7.1 **Server** : garde des captures sur l'état du système (logs, données) depuis les Znodes (nœuds de ZooKeeper) permettant la récupération dans une situation chaotique. on peut avoir plusieurs ZK server dans un cluster garantissant un meilleur service (si un cluster a un ZK server il ne sera pas attribué un client ZK depuis un autre ZK server).

8 **Ambari Infra** : Service d'indexation de données.

8.1 **Infra Solr Instance** : l'unique service de Ambari Infra ; l'installation de multiples instances Infra Solr, garanti l'indexation distribuée pour le service LogSearch qui est nécessaire pour le débogage.

9 **Ambari Metrics** : un service de monitoring. l'interface web d'Ambari est un tableau de bord du cluster ; elle contient des widgets de métriques qui fournissent des informations sur l'état de chaque service, ainsi que l'état du cluster (ex :opérationnel/en veille).

9.1 **Metrics Collector** : via les différent processus démons au niveau de chaque nœud du cluster, ce service récolte des informations sur l'état de chaque service.

9.2 **Grafana** : est un outil de visualisation et de monitoring.

10 **SmartSense** : un service de support et de recommandation proactive.

10.1 **Activity Analyzer** : Activity Analyzer communique avec les applications YARN, Timeline Server , et avec l'HDFS faisant la récolte des données historiques de MapReduce. Il agrège et transforme ces données et les stocke dans le Collecteur de métriques Ambari.

10.2 **Activity Explorer** : Activity Explorer inclut une instance intégrée d'Apache Zeppelin, qui héberge des cahiers journal portables pré-construits qui visualisent les données d'utilisation du cluster pour YARN, Apache Hive ou Apache Tez, et les charges de travail MapReduce. Plus précisément, Activity Explorer inclut des données relatives à l'utilisateur, la file d'attente, la durée du travail et la consommation de ressources par Job.

10.3 **HST Server** : un outil de diagnostique situé dans un nœud dans le cluster, le serveur HST peut consolider efficacement les données collectées par tous les agents HST dans un seul fichier téléchargeable (appelé bundle).

11 **Slider** est un service pour déployer des applications distribuées existantes sur YARN et les surveiller en garantissant la possibilité de changer la configuration même si l'application est en cours. en rappelant les emplacements précédents des données ; Les applications peuvent présenter des temps de démarrage rapides à partir de cette fonctionnalité. il permet également de contrôler tout les nœuds/services du cluster (démarrage/arrêt, configuration spécifique à chaque application).

Par les différents moyens et technologies du cluster réaliser, nous somme capable de crées un entrepôt de données sur hive, stocker les documents sur le HDFS, et de faire le monitoring du culster.
Sur les mêmes machines nous avons réalisé un cluster R qui permet de faire les opérations d'E.T.L. la connexion à la base de données s'effectue avec un pilote JDBC/ODBC ou par l'usage du package RHive permettant à R d'être connecté directement à l'interface HQL de hive.
Notre cluster accueilli également Kylin en mode cluster où les quatre serveur Kylin peuvent opérer en mode requête avec un seul maître qui assigne les Jobs a chacun (query/job = all mode). R permet aussi de connecter à Kylin avec des pilotes JDBC/ODBC.
le tableau suivant montre le positionnement de chaque élément du cluster :

| —Nœud Service— | HDmaster | HDslave1 | HDslave2 | HDslave3 |
|---|---|---|---|---|
| HDFS | DataNode<br>NameNode<br>NFSG<br>SNNode | DataNode | DataNode | DataNode |
| YARN + MR2 | NM RM | ATServer<br>NM | NM | NM |
| TEZ | client | client | client | client |
| Hive | | Metastore<br>HServer2<br>WebHCS | | |

| | | | | |
|---|---|---|---|---|
| HBase | Master RegionServer Phoenix-QueryServer | | | |
| Pig | Client | Client | Client | Client |
| Zookeeper | Server | Server | Server | Server |
| Ambari Infra | | | Infra Solr Instance | |
| Ambari Metrics | | | Grafana | Metrics Collector |
| SmartSense | | | | HSTServer Activity Analyzer Activity Explorer |
| Slider | Client | Client | Client | Client |
| R | RServe RHive RJDBC RODBC | RServe | RServe | RServe |
| Kylin | Kylin Rest Server | Kylin Rest Server | Kylin Rest Server | Kylin Rest Server |

FIGURE 28 – Composition du cluster Hadoop.

Dans le tableau précédent, nous avons montré le positionnement de chaque composant en faisant une répartition que nous jugeons équitable entre les quatre machines selon les capacité de chacune.

certains composant sont présents dans le tableau avec les abréviations suivantes :

- NFSG : NFSGateway.
- SNNode : Secondary NameNode.
- NM : Node Manager.
- RM : Ressources Manager.
- WebHCS : WebHCat Server.
- Hserver2 : HiveServer2.
- ATServer : App TimeLine Server.

## 4.3 Conclusion

Dans la première section du chapitre, nous avons définit l'architecture du système en donnant la chaîne des prétraitements d'un corpus, l'indexation qui est conforme au modèle CXT-Cube et puis la procédure du chargement vers la base de données ; et aussi l'indexation des requêtes qui est similaire à celle d'un document. D'autre part dans la section nous avons conçu le schéma en étoile de la base de données d'accueil et expliqué la nature des requêtes. Dans le reste du chapitre, nous avons couvert essentiellement tout les éléments du cluster HDR sa composition par services et par les entités chargées du fonctionnement de chaque service client ou serveur. Une solution pareille requiert la précision et nécessite une bonne lecture des objectifs.
La prochaine et la dernière partie du mémoire sera dédié à la réalisation du système et l'implémentation du Cluster HDR.

Chapitre V

# Réalisation et éxpérimentation

# 5 Réalisation et expérimentaions

Dans ce chapitre nous implémentons le système suivant l'architecture donnée dans le chapitre précédent. Premièrement l'installation de tous les éléments du cluster, Hadoop et ses applications cohabitantes, l'environnement R et les librairies requises et finalement Apache Kylin. le bon déploiement de ces éléments joue un rôle important sur la performance du système.
La première phase sera suivie par L'E.T.L et le requêtage via R utilisant RJDBC pour Hive et Kylin ; ainsi que le mode distribué R-Hadoop offert par RHive.

## 5.1 Implémentation et réalisation du Cluster

Avant de passer aux détailles de l'installation du cluster, notre cluster d'expérimentation est multi-nœuds de 4 machines virtuelles lancées à partir d'une seul machine physique.

### 5.1.1 Caractéristiques de la machine

Le choix d'une telle machine est une démarche très importante, car la virtualisation des systèmes d'exploitation est un processus complexe, et si on prend on considération que l'utilité de ces machines est pour un système distribué/parallèle. Un seul noeud maître ou esclave requiert au moins deux cœurs processeurs, 8/4Gb de ram (au minimum), et d'un espace disque suffisant pour le stockage HDFS. Donc il faut choisir un cpu d'une architecture multinœuds et qui supporte la virtualisation, et d'une grande capacité en mémoire et de stockage.
nous avons choisi le processeur multi cœurs Intel Core i5-6600K de la sixième génération skylake, 4 cœurs/threads, fréquence de 3.5 à 3.9 GHz, avec une mémoire cache de 6 mb ; pour la ram nous avons choisi 32Gb DDR4/2400 ; et pour le stockage deux disques (SSD : 240 GB / HDD : 1TB).

### 5.1.2 Plan de route

Avant de foncer sur les étapes nous allons introduire l'application de virtualisation. VirtualBox est une application de virtualisation multiplate-forme. elle s'installe sur les ordinateurs Intel ou AMD existants, qu'ils fonctionnent avec des systèmes d'exploitation Windows, Mac, Linux ou Solaris. Elle étend les capacités de l'ordinateur existant afin qu'il puisse exécuter simultanément plusieurs systèmes d'exploitation (à l'intérieur de plusieurs machines virtuelles) ; les seules limites pratiques sont l'espace disque et la mémoire.

VirtualBox est puissante. elle peut fonctionner partout, sur les petits systèmes embarqués comme sur des machines de bureau jusqu'aux déploiements de centres des calculs et même des environnements Cloud.
Pour l'installation d'un cluster Hadoop, la configuration des machines requiert pratiquement les mêmes procédures : exportation des variables globales, changement de certaines valeurs, installation des services nécessaires pour le bon fonctionnement. Il existe plusieurs techniques pour l'installation par exemple Hortonworks SandBox, Cloudera-manager CDH, Ambari-server...etc. Avant de travailler avec Ambari nous avons essayé de le faire avec Couldera-manager express édition qui est une application d'installation et de monitoring pour les cluster Hadoop et qui est fonctionnelle sur le système d'exploitation CentOS 6/7. la version expresse est accessible via une image de disk (vdi, vmdk) qui contient un cluster pseudo-distribué d'un seul nœud installé et configuré. la problématique de Cloudera manager c'est qu'il se sert d'un script (cloudera-manager-init) de démarrage pour paramétrer les variables globales, et configurer les différents services du système. nous avons essayé d'implémenter un cluster multi-nœuds avec CDH mais le script réinitialise les changements effectués au redémarrage et le cluster ne sera plus fonctionnel, et si on désactive ce script le processus d'ajout de nœuds ne sera plus accessible.
Nous avons abandonné la solution précédente pour travailler sur des machines Ubuntu server 14.04 LTS avec Apache Ambari Server qui est plus simple et plus efficace ; et afin de gagner le temps nous avons fait le travail sur une seule VM (machine virtuelle), ensuite nous l'avons cloné avant d'effectuer de petits changements. L'utilité de chaque procédure sera justifié durant l'installation.

### 5.1.3 Préparation de l'environement

une petite remarque avant de commencer ; nous voulons affirmer que dire nœud maître est relative au positionnement des serveur/clients pour un service (le nœud maître HBase est celui qui héberge HBase master). HDmaster contient la plupart de managers/servers d'un cluster Hadoop et pour cette raison nous l'appelons maître.

- Création d'une VM ubuntu 14.04 LTS 64-bit : nous avons choisi cette version pour des raisons de stabilité (généralement les bugs de nouvelles versions prennent du temps avant que la communauté réagisse),de plus les versions récentes (15,16,17) ne sont pas supportées par les différentes distributions de Hadoop, Ambari, Kylin et d'autres services (récemment les versions 16.xx.xx LTS sont ajoutées à la prise en charge su support).
- Installation et activation du serveur ntp Network Time Protocol(apt-

get install ntpd / service ntpd start). ntp est nécessaire pour l'installation et le fonctionnement des services du cluster.

— Désactivation de pare-feu avec (service ufw disable)

— Désactivation des ACLs iptables (service iptables stop). Les différentes applications et services de Hadoop utilisent des ports réseaux ; et si l'une des deux applications est active nous somme obligé de crées des ACLs pour toutes les communications inter-nœuds (permettre la transmission via les ports nécessaires).

— Désactivation de SELINUX. Changement de la valeur existante (enforcing/enabled) par disabled au niveau du fichier /etc/selinux/config.

— Changement de la valeur UMASK vers 022 dans le fichier /etc/login.defs. car le cluster Hadoop à un super-utilisateurs (superuser) pour chaque service et les droits de lecture, écriture et exécution doivent être garantis seulement pour l'utilisateur propriétaire du service.

— Changement du nom de la machine vers HDmaster dans le fichier /etc/hostname (mode root / sudoer).

— Configuration du FQDN (Fully Qualified Domain Name) le nom de la machine avec le préfixe pour le noms de domaine. La tâche sera accomplie par le changement de fichier /etc/hosts en mode root ou avec les privilèges de sudoer. vi /etc/hosts afin d'effectuer les changements figurants dans l'image suivante :

```
127.0.0.1       localhost
127.0.1.1       BAIT

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
192.168.1.101 HDmaster.dotes.mat

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters


192.168.1.100 HDmaster.dotes.mat
192.168.1.101 HDslave1.dotes.mat
192.168.1.102 HDslave2.dotes.mat
192.168.1.103 HDslave3.dotes.mat
```

FIGURE 29 – Configuration de FQDN.

Changer le localhost vers HDmaster.dotes.mat pour le nœud maître (Master node), seulement cette ligne sera changé après le clonage selon la machine(ex :192.168.1.101 HDslave1.dotes.mat pour le premier nœud esclave). il existe un autre moyen pour accomplir cette tâche avec la commande hostname (ex :hostname 192.168.1.100)
Donner les adresses des machines restantes et leurs noms de domaine pour permettre à chacune d'elle de retrouver les autres machines :
192.168.1.100 HDmaster.dotes.mat (maître).
192.168.1.101 HDslave1.dotes.mat (esclave1).
192.168.1.102 HDslave2.dotes.mat (esclave2).
192.168.1.103 HDslave3.dotes.mat (esclave3).

## Installation de R

R est un logiciel de traitement statistique des données. Il fonctionne sous la forme d'un interpréteur de commandes. Il dispose d'une bibliothèque très large de fonctions statistiques, d'intelligence artificiel et de traitement de données. Un langage interprété (de script) et dérivé de S ; R intègre les caractéristiques suivantes : données simples et structurées, il permet également la création des types complexes et la programmation orientée objet (S4,S3,R6 methods) ...etc. R est parmi les langages les plus populaires pour l'ingénierie des données.
installation de R repository (dépôt) :
sudo sh -c 'echo "deb http ://cran.rstudio.com/bin/linux/ubuntu trusty/" » /etc/apt/sources.list'
Ajout de la clé public :
gpg –keyserver keyserver.ubuntu.com –recv-key E084DAB9
Ajout de la clé au manager de package apt :
gpg -a –export E084DAB9 | sudo apt-key add -
m-à-j et installation de R : sudo apt-get update && sudo apt-get install r-base

— Installation de Java : sudo apt-get install default-jdk
— configuration de Java pour R : (as root) R CMD javareconf
— installation des packages nécessaires :
Commencer une session R.
installation des packages base64enc, rJava, NLP, Rcpp, slam, BH, tm, SnowballC,corups, stringr, stringi, geonames, plyr, reshape2, RJSONIO, Rserve, parallel, RUnit.
— installation de rstudio-server :
wget https ://download2.rstudio.org/rstudio-server-1.0.153-

amd64.deb
Installation de debian package installer :
sudo apt-get install gdebi-core
Téléchargement du paquet rstudio.deb :
wget https ://download2.rstudio.org/rstudio-server-1.0.153-amd64.deb
Installation du paquet :
sudo gdebi rstudio-server-1.0.153-amd64.deb

## Installation de Ambari et Hadoop

Cette outil nous permet de déployer un cluster Hadoop, faire le monitoring de cluster, configurer tout les services et supprimer/ajouter de nouveaux nœuds. Lors de l'installation des services un bug ne cesse d'apparaitre où les composant nécessaires aux services de monitoring (Ambari-metrics) ne peuvent être installé donc nous allons le faire manuellement dans cette machine échantillon. Un autre point à citer c'est le moyen de retrouver les autres machines (assurer la connectivité entre les différents services) il faut procéder par une installation manuelle des Agents ambari ou bien par l'installation de passwordless ssh et nous avons choisi le premier moyen.
avant d'installer les deux paquets (ambari-metrics, ambari-agent) il faut ajouté le repository de Ambari :
wget -nv http ://public-repo-1.hortonworks.com/ambari/ubuntu14/2.x/updates/2.4.0.1/ambari.list -O /etc/apt/sources.list.d/ambari.list
apt-key adv –recv-keys –keyserver keyserver.ubuntu.com B9733A7A07513CAD
apt-get update
apt-cache showpkg ambari-server (vérification)
apt-cache showpkg ambari-agent (vérification)
apt-cache showpkg ambari-metrics-assembly (vérification)

— installation de ambari-agent : sudo apt-get install ambari-agent
vi /etc/ambari-agent/conf/ambari-agent.ini pour donner le FQDN du hôte qui héberge ambari-server hostname= HDmaster.dotes.mat et la valeur hostname_ script=/var/lib/ambari-agent/hostname.sh du fichier que nous allons l'éditer.
vi /var/lib/ambari-agent/hostname.sh afin d'écrire le script suivant qui permet d'utiliser un nom d'hôte personnalisé :
#!/bin/sh
echo 'HDmaster.dotes.mat'

— installation de ambari-metrics :
apt-get install ambari-metrics-assembly
— clonage de la machine : nous allons cloner la machine 3 fois, ajouter une carte réseau pour chaque machine('internal network' Réseau interne et aussi rafraîchir l'adresse mac), et donner les ressources nécessaires à chaque machine :
HDmaster : 3coeurs/30%cpu, 12288mo ram
HDslave1 : 2coeurs/49%cpu, 6144mo ram
HDslave2 : 2coeurs/49%cpu, 6144mo ram
HDslave3 : 2coeurs/49%cpu, 6144mo ram
— configuration des machines : il faut modifier les fichiers contenant le nom du hôte pour chaque machine cloné afin de donner les trois nom(HDslave+ 1,2,3) vi /etc/hotname vi /etc/hosts vi /var/lib/ambari-agent/hostname.sh
— installation d'un réseau local qui inclut les quatre machines et avec les adresses 192.168.1.[100-103].
— installation et connexion à ambari-server : au niveau du hôte HDmaster,
apt-get install ambari-server
ambari-server setup
'ambari-server setup' permet d'installer la base de données pour un nouvel usage de l'application et de choisir la version de java qui sera utilisé par le cluster (changeable).
la connexion a l'interface web d'ambari-server s'effectue avec le port 8080 : HDmaster.dotes.mat :8080
— Installation du cluster suivant l'architecture donné dans le chapitre précédent.
— connexion à ambari server avec l'utilisateur/mot de passe : admin admin.
— l'interface d'ambari pour une première installation n'inclut que peu de fonctionnalités qui restent accessibles après l'installation d'un cluster Hadoop. elle inclut des lien vers : la gestion d'accès utilisateurs/groupes et le service qui permet d'identifier des systèmes gérables par ambari et qui sont existants (ex : un cluster Hadoop installé manuellement).

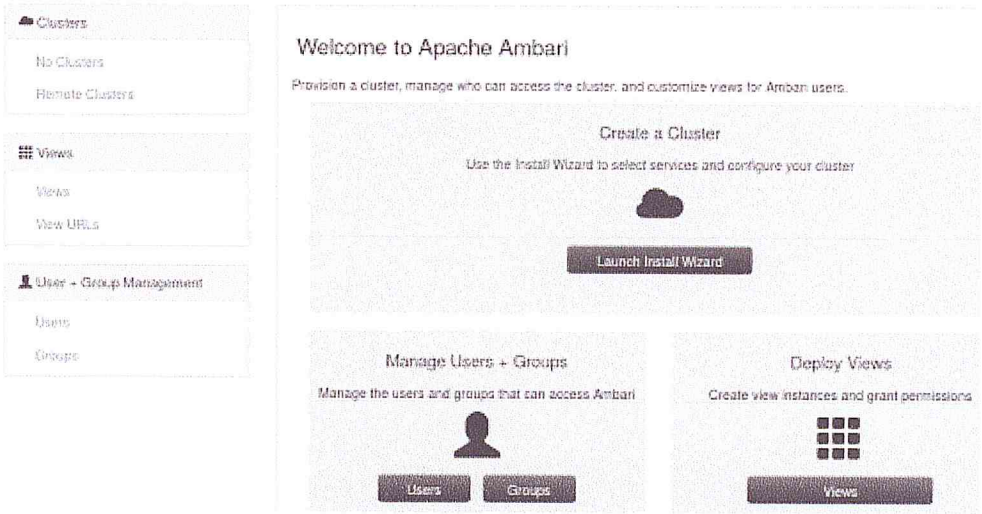



FIGURE 30 – interface d'ambari server.

— après la sélection de 'launch install wizard' nous allons nommé notre cluster 'HDR', l'interface contient un menu glissant pour les prochaines étapes.

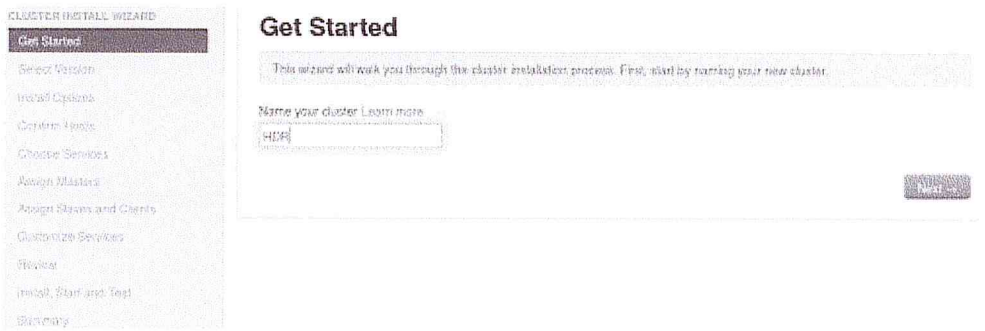



FIGURE 31 – étape 1 de l'installation de HDR.

— la prochaine étape de l'installation sera la sélection de la distribution Hadoop. pour cette dernière nous avons choisi la version 2.4 par défaut malgré, ambari server supporte jusqu'à la version 2.5.3 pour la raison de stabilité.

Une remarque à faire, l'installation des composants se fait par le téléchargement de chaque service depuis le serveur (repository) et au niveau de chaque client du cluster. Les services que nous avons sélectionnés [4.2] occupent 80 Gb et avec une connexion faible à Internet le seuil d'attente (1800sec) s'épuise sans avoir accompli la tâche et toute l'installation sera un échec.

pour cette raison nous avons sélectionné la deuxième option (Use Local Repository) pour déployer un serveur local.

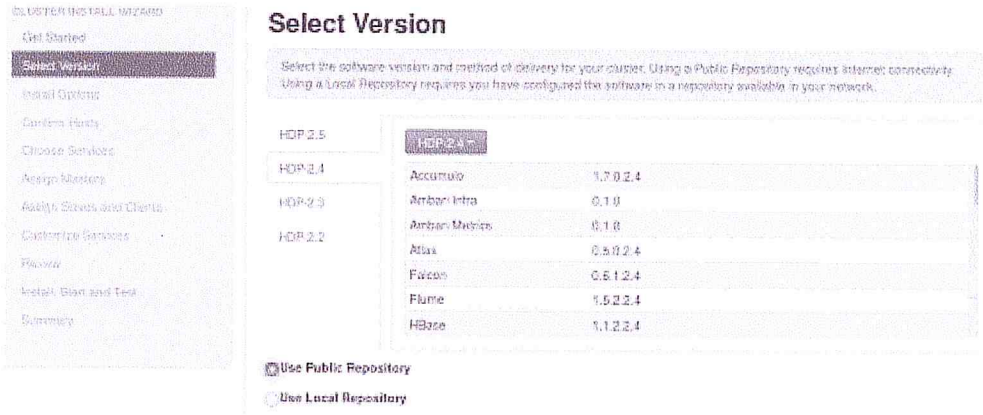



FIGURE 32 – étape 2a de l'installation de HDR.

— Installation du repository local se fait par l'installation du serveur web apache2
sudo apt-get install apache2
service apache2 start.
après on télécharge le fichier deb de la distribution 2.4 ainsi que le paquet des utilitaires Hadoop 1.1.0.20 pour ubuntu 14.04. et les mettre dans le dossier /var/www/html.

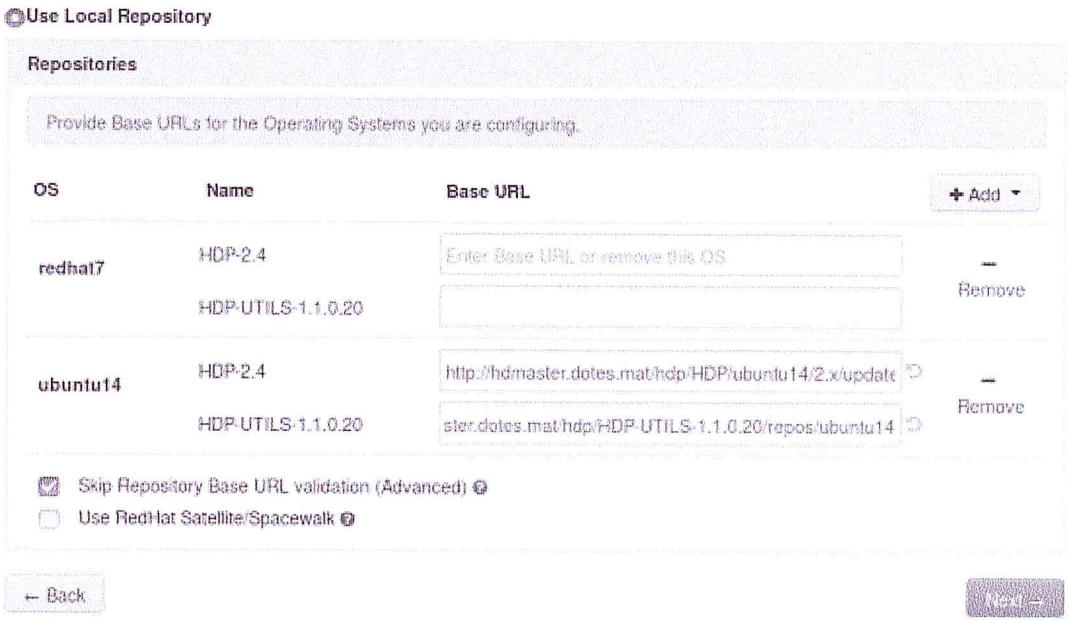


FIGURE 33 – étape 2b de l'installation de HDR.
après la copie des dossier vers :

/var/www/html/hdp/HDP/ubuntu14/2.x/updates/2.4.0.0
/var/www/html/hdp/HDP-UTILS-1.1.0.20/repos/ubuntu14 les nouveaux liens du repository sont :
http ://hdmaster.dotes.mat/hdp/HDP/ubuntu14/2.x/updates/2.4.0.0.
http ://hdmaster.dotes.mat/hdp/HDP-UTILS-1.1.0.20/repos/ubuntu14.
on sélectionne le système d'exploitation ubuntu 14.04, on met les liens dans les champs appropriés et on supprime le reste des systèmes (debian, redhat/centos, suse).

— Sélection des machines hôtes et le choix d'une installation manuelle pour les agents ambari-agent (nous avons montré la procédure précédemment).

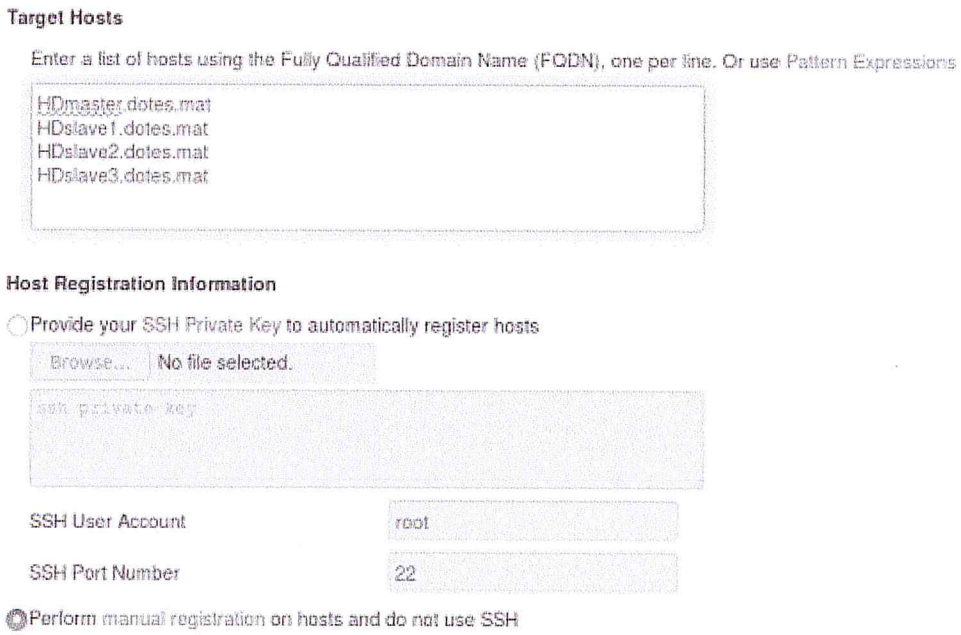



FIGURE 34 – étape 3 de l'installation de HDR.

— Prochainement ambari-server retrouve les hôtes ciblés via les agents ambari de chacun et il s'assure que le serveur NTP est opérationnel ; ainsi qu'il prend le contrôle de sudoer principale pour installer prochainement les composants et la gestion des super-utilisateurs créer par ambari-server.

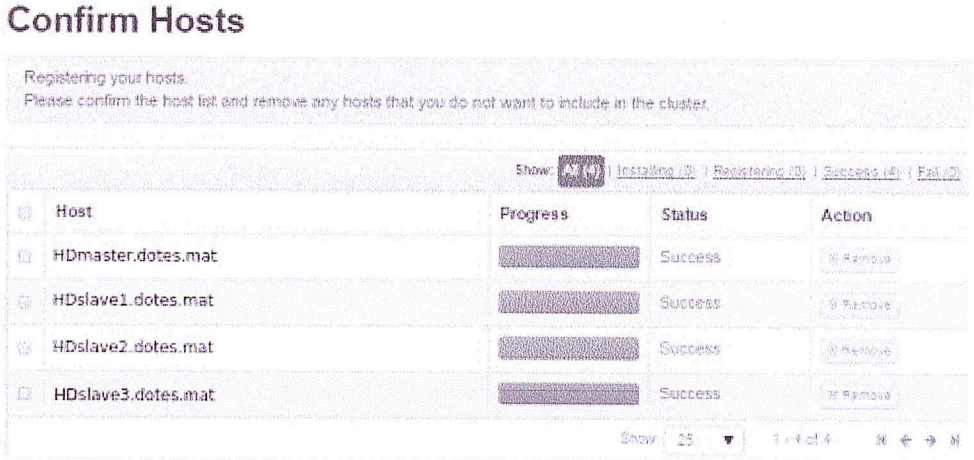



FIGURE 35 – étape 4 de l'installation de HDR.

— après la validation des nœuds on procède à la sélection des applications et services nécessaire pour l'architecture proposée dans [4.2] et qui sont : HDFS, YARN+MR2, TEZ, HIVE, HBase, Pig(nous n'avons pa travaillé sur Pig mais il est nécessaire pour le fonctionnement de Hive en mode arrière plan), Zookeeper, Ambari infra, Ambari metrics, Smart Sense et Slider.

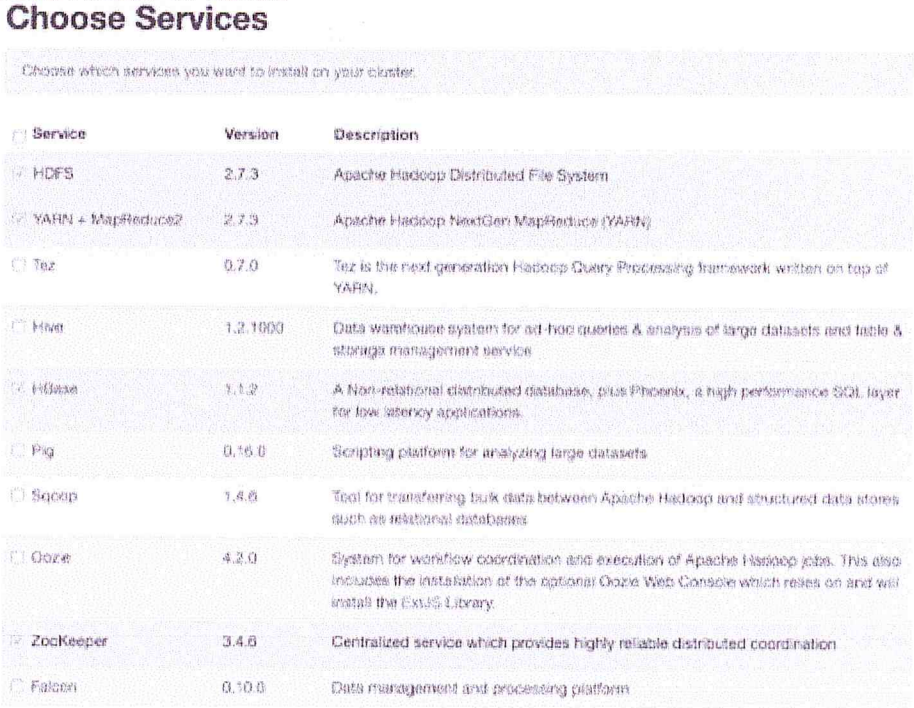



FIGURE 36 – étape 5 de l'installation de HDR.

— Affectation des serveurs suivant l'architecture proposée[4.2].

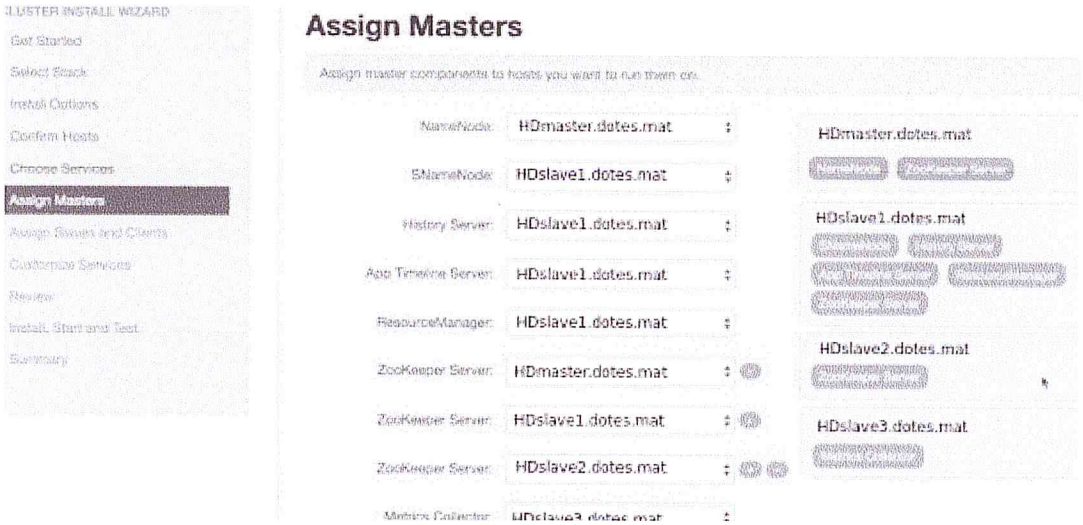



FIGURE 37 – étape 6 de l'installation de HDR.

— Affectation des Clients / esclaves.

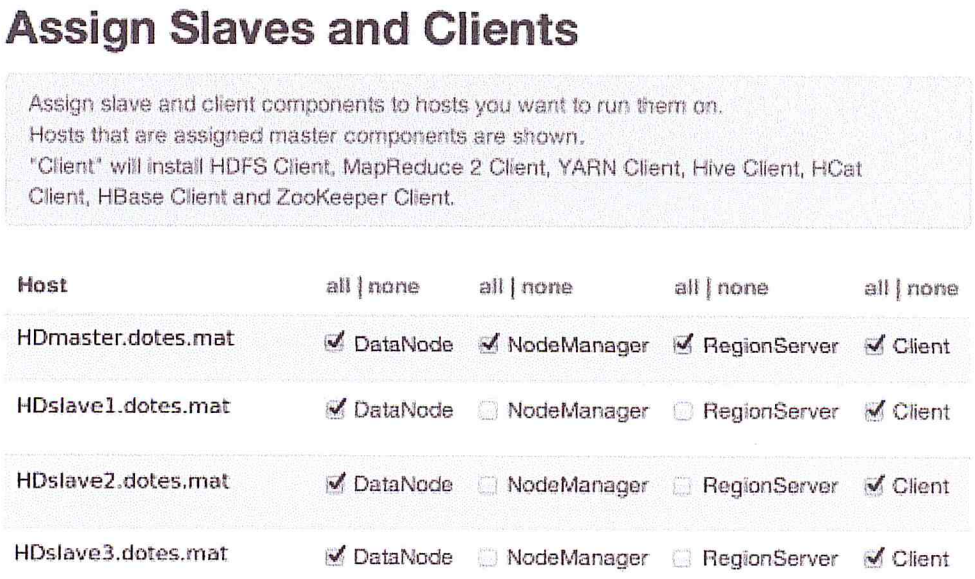



FIGURE 38 – étape 7 de l'installation de HDR.

— Personnalisation des services : dans cette étape on configure chaque service; ambari propose une configuration par défaut pour chacun (ex : port d'accès de serveur Zookeeper, la capacité mémoire des application_ containers, droit d'accès utilisateur/service).

dans cette étape on configure également les comptes { utilisateur/mot de passe } pour les bases de données disponibles (Hive, HBase, new SQL ...).

— La phases précédentes seront accomplies par la génération du rapport contenant une vue globale sur le cluster ; permet de prendre la décision de continuer (deploy button) ou de reconfigurer le cluster (back).

— Ambari fait le déploiement (installation des composants, lancement et testes).

— à la fin de l'installation, ambari-server génère un rapport finale contenant le nom d'utilisateur, nom du cluster, nombre d'hôtes, serveurs (référentiels utilisés pour l'installation des composants) et les services avec leurs positionnements (le document sera attaché a ce mémoire). le cluster HDR nous permet de réaliser un entrepôt de données utilisant Hive,la prochaine étape sera l'installation de Kylin server afin de comparer les résultat des deux infrastructures.

## Installation de Kylin(prérequis)

l'exportation des variables globales contenant les chemins des fichier suivant(tout les nœuds) :

```
vi /.bashrc
export Hadoop_HOME=/usr/hdp/2.4.0.0-169/Hadoop
export Hadoop_LIB=/usr/hdp/2.4.0.0-169/Hadoop/bin
export Hadoop_CMD=/usr/hdp/2.4.0.0-169/Hadoop/lib
export HIVE_HOME=/usr/hdp/2.4.0.0-169/hive
export HIVE_LIB=/usr/hdp/2.4.0.0-169/hive
export HCAT_HOME=/usr/hdp/2.4.0.0-169/hcatalog
export HCAT_LIB=/usr/hdp/2.4.0.0-169/hcatalog/lib
source /.bashrc
```

— Téléchargement du fichier tar.gz pour la version 1.6 (plus stable que les versions 2.*), la copie vers /usr/Kylin (après l'extraction) et l'exportation de KYLIN_HOME au niveau du fichier .bashrc (pour tout les nœuds).

— Configuration des rôles : la tâche s'effectue par la modification de la valeur kylin.server.mode en donnant la valeur all pour le nœud HDmaster et query pour les autres. la valeur se trouve dans le fichier /usr/Kylin/conf/kylin.properties.

— Lancement des serveurs kylin au niveau de tout les nœuds avec /usr/Kylin/bin/kylin.sh start. et le cluster sera fonctionnel.

— Installation de RHive :
premièrement on modifie le fichier /etc/Rserv.conf dans tout les

nœuds.
'remote enable' pour autoriser la connexion a distance entre les instance de Rserve.
lancement de Rserve avec la commande : R CMD Rserve

— Téléchargement du code source RHive de Github :
git clone https ://github.com/nexr/RHive.git
— Construction du packet RHive pour R :
cd RHive
ant build
R CMD build RHive
— installation de RHive avec R CMD install.packages("./RHive.tar.gz").
La mise en œuvre est terminé avec succès et notre système est prêt à accueillir les données et faire les traitements.

## 5.2 ETL

après avoir déployer le cluster et installer les composants nécessaires, nous allons procéder au nettoyage du corpus, transformer le texte en nombres et effectuer le chargement des données vers Hive.

### 5.2.1 Prétraitement

Dans cette partie nous allons montrer les techniques utilisées pour chaque étape :
la fonction Corpus du package tm nous permet de créer des objets S3 contenant les attributs figurant dans la capture d'écran suivante :

```
1
2  docs
3  docs$`character(0)`
4  docs[[1]]$content
5  docs[[1]]$meta$author
6  docs[[1]]$meta$datetimestamp
7  docs[[1]]$meta$description
8  docs[[1]]$meta$heading
9  docs[[1]]$meta$id
10 docs[[1]]$meta$language
11 docs[[1]]$meta$origin
12
```

FIGURE 39 – attributs de Corpus.

lors de la création d'un corpus, la fonction retrouve le contenu des documents et les métadonnées de chacun. la fonction utilise comme paramètres :

DirSource(le chemin vers le dossier contenant la collection) et le choix de la langue. le chargement de document se montre récalcitrant souvent vu les problèmes d'encodage qui apparaissent.
Avec le script suivant, nous avons fait les prétraitements initiales automatiquement :

```
#chargement de package
library(tm)
library(SnowballC)
library(stringi)
library(stringr)
#Création du corpus:
docs <- Corpus( DirSource("/home/machine/Documents/Cropus"),
                readerControl = list(reader=readPlain, language="fr"))
#Supression de la ponctuation
docs <- tm_map(docs , content_transformer(function(x)
  {return (str_replace_all(x,"[[:punct:]]"," "))}))
docs <- tm_map(docs , removePunctuation)
#Transformation en miniscule
docs <- tm_map(docs , tolower)
#Élimination des nombres
docs <- tm_map(docs , removeNumbers)
docs <- tm_map(docs , PlainTextDocument)
#Élimination de mots vides
docs <- tm_map(docs , removeWords, stopwords('fr'))
docs <- tm_map(docs , removeWords, stopwords('en'))
#Enlèvement des espaces blancs
docs <- tm_map(docs , stripWhitespace )
```

FIGURE 40 – Nettoyage initiale.

la fonction tm_ map permet de faire le mapping d'un objet S3, donc nous l'avons utilisée pour faire les transformations nécessaires.
Création des dictionnaires et validation des termes :

- Dictionnaire Français.
- Dictionnaire Anglais.
- Dictionnaire Français contenant les mots avec des lettres accentuées (Fr_acc).
- dictionnaire contenant l'origine des mots du dictionnaire précédent Fr_aco.

```
1  library(stringr)
2  FR_dic_acc_finder = function(dic,acc)
3  {
4    A <- length(dic)
5    P <- NULL
6
7    for(i in 1:A)
8    {
9      B <- str_length(dic[i])
10     C <- dic[i]
11     for(j in 1:B)
12     {
13       if(is.na(match(substring(C,j,j),acc)) == FALSE)
14         P <- c(P, dic[i])
15       j <- B
16     }
17   }
18   return(P)
19 }
20 FR_dic_acc_creator = function(dic,acc,rac)
21 {
22   for(i in 1:length(acc))
23     {
24     dic <- str_replace_all(dic,acc[i],rac[i])
25     }
26   return(dic)
27 }
28 Fr_dic  <- c('le','président','algérien','a','réussi','à','être','légendaire')
29 FR_acc <- FR_dic_acc_finder(Fr_dic,c('é','è','ê','à'))
30 FR_aco <- FR_dic_acc_creator(FR_acc,c('é','è','ê','à'),c('e','e','e','a'))
```

FIGURE 41 – exemple : script de création de Fr_acc, Fr_aco.

— Dictionnaire contenant les noms des villes qui comptent plus de 5000 habitant avec ces données de géolocalisation. les noms de villes en 7 à 38 langues où la première dans la ligne est la langue française et à l'aide du séparateur(tab \t)on a isolé les noms ciblés.(http ://download.geonames.org/export/dump/cities5000.zip)

```
Console ~/
> WCities <- readLines('/home/machine/Downloads/cities5000.txt',encoding = 'UTF-8')
> WCities[[11140]]
[1] "2481700\t5étif\tSetif\tGorad Setyf,QSF,Sathif,Satif,Setif,Setifas,Sitifis,Stif,Sé
tif,sai ti fu,setifu,setipeu,stf,styf,styp,wlayt styf,Горад Сетыф,Сетиф,Сетиф қаласы,C
етіф,سطيف,ولاية سطيف,سطف,ס'טיף,Ṣṭif,セティフ,塞提夫,세티프\t36.19112\t5.41373\tP\tPPLA\t
DZ\t\t12\t\t\t\t288461\t\t1096\tAfrica/Algiers\t2013-08-04"
>
```

FIGURE 42 – cities5000.txt.

— Création de la hiérarchie de concepts contenant les compétences/métiers ainsi que son dictionnaire.
— validation des termes avec les dictionnaire et récupération des termes susceptibles d'être faux.
— correction des termes composés (ex : htmlcss) ; qui sont des modèles

courants dans les CVs, en (utilisant des opérations des bas niveaux de chaine de caractère 'string').
Avant les traitements le nombre de termes uniques était 24010, et après 17183 où 12429 sont les termes qui appartient aux dictionnaires (inclut les termes corrigé), les 4754 termes restants représentent les noms d'individus, entreprises, écoles et les maitrises qui n'appartiennent pas à la hiérarchie de concepts...etc. nous savons que l'approche de CXT-Cube s'intéresse seulement aux termes contenus dans la hiérarchie de concept donc une indexation automatique est plus rapide pour achever le processus quand il s'agit d'une collection énorme.
notre tentative de validation pour les termes avec d'autres dictionnaires est justifiée par (dans un cadre expérimentale où professionnel) :

- Avoir une idée profonde sur la composition de la collection.
- La collection traité ainsi, sera conforme et prête pour d'autres traitements analytiques.
- Avoir une meilleur qualité des résultat (331 termes uniques réparés(632 avec toutes les occurrences) où 103 font partie de la hiérarchie utilisée.

— Finalement la lemmatisation : les fonctions existantes dans les différentes bibliothèque que nous avons essayé, n'effectue pas la tâche parfaitement donc nous avons créer un dictionnaire qui contient des vecteurs , chacun des derniers regroupe un ensemble de termes (variant de 1 à 40) partageant une racine commune où le premier représente le lemme (pas forcément dans le vocabulaire). notre choix et justifié par : l'indexation (dans la RI) requiert la lemmatisation qui affect l'analyse d'un point de vue purement statistique/quantitatif. la figure suivante montre un exemple du dictionnaire crée :

```
> FRstemm[[947]]
 [1] "acculer"         "acculera"        "acculerai"       "acculeraient"
 [5] "acculerais"      "acculerait"      "acculeras"       "acculerez"
 [9] "acculeriez"      "acculerions"     "acculera"        "acculerai"
[13] "acculeraient"    "acculerais"      "acculerait"      "acculeras"
[17] "acculerai"       "acculeraient"    "acculerais"      "acculerait"
[21] "accultura"       "acculturai"      "acculturaient"   "acculturais"
[25] "acculturait"     "acculturant"     "acculturas"      "acculturasse"
[29] "acculturassent"  "acculturasses"   "acculturai"      "acculturaient"
[33] "acculturais"     "acculturait"     "acculturas"      "acculturasse"
[37] "acculturassent"  "acculturasses"   "acculturassiez"  "acculturassions"
```

FIGURE 43 – dictionnaire de lemmatisation.
la fonction de lemmatisation effectue la tâche par la transformation des

termes appartenant à un vecteur, vers la première valeur. par example le terme 'acculerais' sera transformé vers 'acculer'. la lemmatisation n'a pas affecté notre analyse due à la composition de la collection.

### 5.2.2 Transformation et chargement des données

Après avoir nettoyer la collection, les procédures faites dans cette section sont :

- Calcul des fréquences : pour cette tâche nous avons crée une fonction qui fait le travail sur 2 étapes : la première est de définir le nouveau nombre de termes, car la hiérarchie de concepts contient des termes composés de plusieurs mots ; et deuxièmement le calcul des fréquences. Par exemple, un document composé de 3000 mots où le terme "base données" est présent 3 fois ; le nombre de termes total doit être 3000-(2-1)*3 = 2997.
- la propagation de pertinence : nous avons utilisé une fonction qui effectue le calcul commençant par les dernières feuilles en remontant vers la racine.
- Récupérer les métadonnées de la collection : pour la date d'envoi nous étions obligés d'utiliser un générateur qui donne des valeurs aléatoires en format "DD-MM-YY HH :MM :SS" car les dates présentes comme métadonnées sont celles de la date de transfert des documents vers le disque dur.
  pour la localisation, les documents contient des noms de villes Africaines, Françaises et parfois ne contenaient pas des localisations ; pour retrouver les niveau hiérarchiques supérieur il fallait chercher tout les termes d'un document avec l'API geonames pour savoir s'il s'agit des données de localisation, autrement le processus sera extrêmement couteux. donc pour cette raison nous avons opté pour récupérer tous les données accessibles via l'API et faire un traitement local ; malheureusement les requêtes utilisant l'API sont limitées par utilisateur (2000 crédit/heurs,30000/jour où une requête de type GNchildren coute 4 pts de crédit). Cette solution nous à permit de récupérer les données de 28 pays durant un mois et il fallait environ 8 pour le reste des pays.
  la figure suivante contient le code des fonctions (écrit en R) permettant de retrouver les données de géolocalisation :

```
library(geonames)
library(stringi)
library(stringr)

options(geonamesUsername = '[redacted]')

db_xml_countries2 = function(fipsCD)
{
  HHH <- ' '
  A <- length(fipsCD)
  H <- '<world_countries_fr>'
  for(i in 1:A)
  {
    B <- GNcountryInfo(fipsCD[i],lang='fr')
    H <- paste0(H,'<country')
    P <- '<country><info'
    C <- variable.names(B)
    D <- length(C)
    for(j in 1:D)
    {
      E <-  paste0(C[j],'=',"'",B[j],"'")
      H <- paste(H,E)
      P <- paste(P,E)
    }
    H <- paste0(H,'/>',B$countryName)
    P <- paste0(P,'/>', B$countryName)
    try(HHH <- db_xml_child2(B$geonameId),FALSE)
    H <- paste(H,HHH)
    P <- paste(P,HHH)
    H <- paste0(H,'</country>')
    P <- paste0(P,'</country>')
    writeLines(P,paste0('Downloads/Application/dictionnaires/geo/country-',
                        B$fipsCode,'-',B$countryname,'.xml'))
  }
  H <- paste0(H,'</world_countries_fr>')
  return(H)
}

      db_xml_child2 = function(gnID)
      {
        A <- 1

        A <- GNchildren(gnID,lang='fr')
        # condition for empty childs left
        B <- variable.names(A)
        C <- length(B)
        D <- A$name
        D <- length(D)
        H <- '<children>'
        for(i in 1:D)
        {
          H <- paste0(H,'<child')
          Sys.sleep(D*0.2)
          for(j in 1:C)
          {
            E <- paste0(B[j],'=',"'",A[[j]][i],"'")
            H <- paste(H,E)

          }
          H <- paste(H,'>',A$name[i])
          try(H <- paste(H,db_xml_child2(A$geonameId[i])),FALSE)
          H <- paste0(H,'</child>')

        }
        H <- paste0(H,'</children>')
        return(H)
      }
```

FIGURE 44 – Fonctions R : Générateur de fichiers XML pour les géo-infos.

la première fonction " db_xml_countries2(fipsCD)" prend l'entrée qui représenté un vecteur de type string contenant le code fips d'un pays (ou liste de plusieurs ex : "AG","AJ","AL" pour Algérie, Azerbaijan, Albanie), premièrement et via la fonction GNcountryInfo elle retrouve les informations (Country, Capital, Area(in sq km), Population, Continent, tld, CurrencyCode, CurrencyName, Phone, Postal Code Format Postal Code Regex, Languages, geonameid, neighbours, EquivalentFipsCode). Dans l'étape suivante, la fonction fait appel à "db_xml_child2(gnID)" qui utilise le géo-identifiant pour retrouver des informations sur les provinces du premier niveau administratif et récursivement elle retrouve les information sur les enfants de cette province jusqu' a l'atteinte du dernier niveau administratif. les fonctions sont écrite pour fournir mutuellement un document XML qui contenant toutes les informations disponibles sur les pays données en paramètre. la fonction fait une pause après chaque appel récursif pour éviter l'arrêt des calcules causé par le manque des points de crédit (la pause est optimisée pour consommé 30000 pts/jour).
Malheureusement le temps ne jouait pas à notre faveur; donc nous avons récupéré les géo-infos de la France, et via un générateur nous avons attribué une localisation de 6 niveaux administratifs pour chaque document. Le but c'est d'avoir une collection de testes parfaite pour l'analyse et pas réussir à avoir les métadonnées de la collection de teste.

— Création de la base de données au niveau de Hive.
— Exportation des fichiers en format CSV où chaque fichier contient les données d'une table.
— Chargement des données via l'interface_web/cli de Hive et vers chaque table en faisant le paramétrage des tables. le tableau suivant montre mes types de données utilisés pour chaque colonne :

| Tables | colonnes | types de données |
|---|---|---|
| fact_table | loc_id, top_id,<br>sub_id<br>pow | <br>INTEGER<br>DOUBLE |
| subtime | sub_id<br>sub_tstamp<br>sub_day<br>sub_month<br>sub_quarter<br>sub_year | INTEGER<br>TIMESTAMP<br>DATE<br><br>TINYINT<br>SMALLINT |

| | | |
|---|---|---|
| location | loc_id<br>country, admin1-5 | INTEGER<br>STRING |
| topic | top_id<br>top1-198 | INTEGER<br>DOUBLE |

FIGURE 45 – Les types de données de la base Hive.

Les types de données utilisés sont choisis avec précision pour améliorer les ressources de calcul et stockage ; par ex : pour la colonne month nous avons choisit le type TINYINT car il prend une valeur de 0-255 et le stockage en 1 byte en sachant que les valeurs d'un mois sont comprises entre 1-12, dans ce cas l'usage de SMALLINT/INT/BIGINT (2,4,8 bytes en stockage) rend la taille de la base immense. ne pas utiliser STRING dans les valeurs year/month est justifié par : les opérateurs utilisés prochainement ne requiert pas des opérations des dates "ex : cast(date as string)" sauf le cas d'usage des intervalles et qui requiet day/subtime qui sont stockés dans les formats requises.

— Duplication des documents pour atteindre les 500 millions ( 2.4Tb). durant le calcul des rankings, les duplicatas d'un document auront le même ranking que l'original, se qui facilite la vérification des résultats.
— Chargement des nouveaux documents vers la base de données utilisant RHive.

## 5.3 Expérimentation

après l'ETL les données sont prêtes pour la construction du cube de données ; dans cette partie nous allons créer le cube avec Hive, kylin et introduire comment faire les requêtes :

### 5.3.1 Concept de Cube avec Hive

la fonctionnalité CUBE a été ajoutée récemment à HQL, elle crée un sous-total de toutes les combinaisons possibles de l'ensemble des colonnes sélectionnées. Une fois que nous avons calculé un CUBE sur un ensemble de dimensions et par la définition des jointures, nous pouvons obtenir une réponse à toutes les questions d'agrégation possibles sur ces dimensions.
GROUP BY a, b, c avec CUBE est équivalent à
GROUP BY a, b, c GROUPING SETS ((a, b, c), (a, b), (b, c), (a, c), (a), (b), (c), () ).
Dans notre cas avec le modèle CXT_cube pour les modes discutés au niveau de la partie conception nous allons utilisé toutes tables et colonnes appartenant à notre base multidimensionnelle, donc la création d'un Cube ne sera pas bénéfique vis-à-vis l'optimisation du temps de réponse.

### 5.3.2 Construction du Cube de données avec Kylin

Pour Kylin le mode d'opération est différent car il crée des cuboïdes (Delta Cube,$2^3 = 8$ pour un cube de 3 dimension) optimisés pour les opérateurs définis en utilisant les opérateur existants .
la création d'un cube avec Kylin s'effectue au niveau de l'interface web du serveur par la création d'un nouveau modèle de cube et ses opérateurs en suivant les étapes :

- Création du projet "e_recruite".
- Synchronisation avec les tables de Hive.
- Création du modèle de Cube "CXT_Cube".
- Sélection de la table de fait.
- Définition des jointures internes (INNER JOIN) entre les tables de dimension et la table de fait utilisant les clés étrangères.
- Sélection des colonnes utilisables dans le Cube (toutes).
- Sélection de la mesure à utilisé; l'une des limitations de Kylin c'est qu'il exige une mesure de la table de fait (l'utilité de la mesure pow [4.1.2]).
- Sélection de la taille de cube,la valeur MEDIUM est la valeur par défaut. cette valeur représente la taille des données stockées dans le Cube au niveau de HBase (10Gb SMALL, 20 Gb MEDIUM, 100Gb LARGE).
- Filtrage des données (l'étape n'est pas nécessaire car les données sont déjà nettoyées).

Après la Création d modèle de Cube de données nous allons procéder à la création du Cube et suivant les étapes :

- Sélection du modèle crée CXT_Cube.
- Définition des dimension normales (topic, subtime, location) et dérivées (les niveaux hiérarchiques).
- Enregistrement du cube.
- Exploration des cube existant dans une autre interface, qui représente un tableau de bord, permettant de lancer la construction du Cube (l'étape à pris 25 minutes).

### 5.3.3 Connexion à Hive et requêtage

Dans cette partie nous élaborons les connexion depuis R à Hive, formulation des requêtes propres au langages utilisés par Hive(HQL) et avec les UDFs. Commençons d'abord par les connexions :

**connexion à Hive via JDBC :**

La Bibliothèque RJDBC permet à l'utilisateur de brancher un pilote JDBC pour créer une connectivité à une base de données, le code de source suivant permet d'effectuer l'opération avec notre base de données Hive :

```
Sys.setenv('HADOOP_CMD'='/usr/hdp/2.4.0.0-169/hadoop/bin/hadoop')
Sys.setenv('HADOOP_HOME'='/usr/hdp/2.4.0.0-169/hadoop')
Sys.setenv('HADOOP_LIB'='/usr/hdp/2.4.0.0-169/hadoop/lib/')
Library(RJDBC)

for(l in list.files('/usr/hdp/2.4.0.0-169/hive/lib/'))
  { .jaddClassPath(paste("/usr/hdp/2.4.0.0-169/hive/lib/",l,sep=""))}
for(l in list.files('/usr/hdp/2.4.0.0-169/hadoop/'))
  { .jaddClassPath(paste("/usr/hdp/2.4.0.0-169/hadoop/",l,sep=""))}
for(l in list.files('/usr/hdp/2.4.0.0-169/tez/'))
  { .jaddClassPath(paste("/usr/hdp/2.4.0.0-169/tez/",l,sep=""))}

options( java.parameters = "-Xmx8g" )
.jinit()
.jclassPath()
drv <- JDBC("org.apache.hive.jdbc.HiveDriver","/usr/hdp/2.4.0.0-169
            /hive/lib/hive-jdbc-1.2.1000.2.4.0.0-169.jar")
conn <- dbConnect(drv,"jdbc:hive2://hdmaster.dotes.mat:2181/;
            serviceDiscoveryMode=zooKeeper;
            zooKeeperNamespace=hiveserver2")
```

FIGURE 46 – connexion JDBC vers Hive avec un script R.

Nous avons défini les variables globales nécessaires,
les ClassPaths vers les applications requises pour la connectivité (dans ce cas c'est des fichiers JAR),
modification de la valeur maximal de la mémoire utilisable par Java (java heapsize par défaut = 256m) vers 8g.
définition du JDBC driver de Hive et finalement la connexion sera établie.

**requêtes avec JDBC**

Comme évoqué dans le chapitre précédent [4.1.3], nous allons généré des requêtes pour les différents modèles abordés :

```
HQL_jdbc_ = function(qry_top,qry_loc,loc_lvl,loc_mode,sbt
                     ,sbt_mode,sbt_lvl,imp,loc_imp,top_imp)
{ A <- 0
  B <- 0
  C <- NULL
  E <- NULL

  for(i in 1:length(qry_top))
  {   if(qry_top[i] != 0)
      A <- paste0(A,"+ topic.top","*",qry_top[i]^2)
      B <- B + qry_top[i]^2
  }
  if(loc_mode == 0)
  {
      C <- paste0('WHERE location.',
          c("country","admin1","admin2","admin3","admin4","admin5")
              [loc_lvl]," LIKE  %", qry_loc ,"%")
  }
  else
  {
    if(loc_lvl == 1)
    {
      C <- paste0(" + CASE WHEN location.country   LIKE   '%",
                  qry_loc,"%' THEN 1 ELSE 0 END  ")
    }
    if(loc_lvl == 2)
    {
      C <- paste0(" + ((CASE WHEN location.country  LIKE  '%",
                  qry_loc[1],"%' THEN 1 ELSE 0 END) +
                  (CASE WHEN location.admin1  LIKE  '%",qry_loc[2],
                  "%' THEN 1 ELSE 0 END ))/2  ")
    }
    if(loc_lvl == 3)
    {
      C <- paste0(" + ((CASE WHEN location.country  LIKE  '%",
                  qry_loc[1],"%' THEN 1 ELSE 0 END) +
                  (CASE WHEN location.admin1  LIKE  '%",qry_loc[2],
                  "%' THEN 1 ELSE 0 END )+
                  (CASE WHEN location.admin2  LIKE  '%",qry_loc[3],
                  "%' THEN 1 ELSE 0 END ))/3  ")
    }
    if(loc_lvl == 4)
    {
      C <- paste0(" + ((CASE WHEN location.country  LIKE  '%",
                 qry_loc[1],"%' THEN 1 ELSE 0 END) +
                (CASE WHEN location.admin1  LIKE  '%",qry_loc[2],
                  "%' THEN 1 ELSE 0 END )+
                (CASE WHEN location.admin2  LIKE  '%",qry_loc[3],
                  "%' THEN 1 ELSE 0 END )+
                (CASE WHEN location.admin3  LIKE  '%",qry_loc[4],
                  "%' THEN 1 ELSE 0 END ))/4  ")
    }
```

```
    if(loc_lvl == 5)
    {
      C <- paste0(" + ((CASE WHEN location.country  LIKE  '%",
                  qry_loc[1],"%' THEN 1 ELSE 0 END) +
                 (CASE WHEN location.admin1  LIKE  '%",qry_loc[2],
                   "%' THEN 1 ELSE 0 END )+
                 (CASE WHEN location.admin2  LIKE  '%",qry_loc[3],
                   "%' THEN 1 ELSE 0 END )+
                 (CASE WHEN location.admin3  LIKE  '%",qry_loc[4],
                   "%' THEN 1 ELSE 0 END )+
                 (CASE WHEN location.admin4  LIKE  '%",qry_loc[5],
                   "%' THEN 1 ELSE 0 END ))/5  ")
    }
    if(loc_lvl == 6)
    {
      C <- paste0(" + ((CASE WHEN location.country  LIKE  '%",
                  qry_loc[1],"%' THEN 1 ELSE 0 END)+
                  (CASE WHEN location.admin1  LIKE  '%",qry_loc[2],
                  "%' THEN 1 ELSE 0 END )+
                  (CASE WHEN location.admin2  LIKE  '%",qry_loc[3],
                  "%' THEN 1 ELSE 0 END )+
                  (CASE WHEN location.admin3  LIKE  '%",qry_loc[4],
                  "%' THEN 1 ELSE 0 END )+
                  (CASE WHEN location.admin4  LIKE  '%",qry_loc[5],
                  "%' THEN 1 ELSE 0 END )+
                  (CASE WHEN location.admin5  LIKE  '%",qry_loc[6],
                  "%' THEN 1 ELSE 0 END))/6  ")
  }


  }
  if(sbt_mode == 0 && loc_mode == 0 && typeof(sbt) == "double")
  {
    E <- paste0(" AND  subtime.",
                c("","","sub_month","sub_quarter","sub_year")
                [sbt_lvl],"== ",sbt)

  }
  if(sbt_mode == 1 && loc_mode == 0)
    {
    if(typeof(sbt) == "double")
      {E <- paste0(" AND ")
       for(i in 1:length(sbt))
       {
                  E <- paste0(E, "subtime.",
                  c("","","sub_month","sub_quarter","sub_year")
                                          [sbt_lvl],"== ",sbt[i])
       if(i < length(sbt)) E <- paste0(E," OR ")
      }}
    else
      {
```

```
104         if(str_length(sbt[1]) > 6)
105           E <- paste0(" AND  sub_time.sub_tstamp BETWEEN  '",
106                       sbt[1],"' AND '",sbt[2],"'")
107         else
108         {
109           E <- paste0(" AND  sub_time.sub_day BETWEEN  '",
110                       sbt[1],"' AND '",sbt[2],"'")
111         }
112
113       }
114     }
115   if(sbt_mode == 0 && loc_mode == 1 && typeof(sbt) == "double")
116   {
117     E <- paste0("WHERE subtime.",
118                 c("","","sub_month","sub_quarter","sub_year")
119                 [sbt_lvl],"== ",sbt)
120
121   }
122   if(sbt_mode == 1 && loc_mode == 1)
123   {
124     if(typeof(sbt) == "double")
125     {E <- paste0("WHERE")
126     for(i in 1:length(sbt))
127       E <- paste0(E, "subtime.",
128             c("","","sub_month","sub_quarter","sub_year")
129                 [sbt_lvl],"== ",sbt[i])
130     if(i < length(sbt)) E <- paste0(E," OR")
131     }
132     else
133     {
134       if(str_length(sbt[1]) > 6)
135         E <- paste0("WHERE sub_time.sub_tstamp BETWEEN  '",
136                     sbt[1],"' AND ",sbt[2],"'")
137       else
138       {
139         E <- paste0("WHERE sub_time.sub_day BETWEEN  '",
140                     sbt[1],"' AND '" ,sbt[2],"'")
141       }
142   } }
143   if(imp == 0){top_imp <- 0.5
144                loc_imp <- 0.5}
145     QRY <-
146
147
148 paste0("SELECT /*+ MAPJOIN(topic,fact_table,
149 location,subtime)*/
150 top_id,((",A,")/(SQRT(",B,"*pow))*",top_imp,")"," (",C,")*",loc_imp,E,")
151 AS RANK FROM fact_table
152 JOIN topic ON topic.top_id == fact_table.top_id
153 JOIN location ON location.loc_id == fact_table.loc_id
154 JOIN subtime ON subtime.sub_id == fact_table.sub_id
155        ORDER BY RANK;")
```

```
156     QRY <- str_replace_all(QRY,'\n','')
157     QRY <- str_replace_all(QRY,'        ','')
158     QRY <- gsub('\\','',QRY, fixed=T)
159     QRY_s <- dbSendStatement(conn,QRY)
160     QRY_s <- dbFetch(QRY_s)
161
162     return(QRY_s)
163   }
```

FIGURE 47 – Requêtes HQL avec RJDBC, source R .

Cette fonction écrite en R, permet de formuler, paramétrer une requête et la soumettre avec le connecteur JDBC qui utilise les application de Hadoop pour créer un Job MapReduce depuis Hive. les paramètre de la fonction sont et suivant l'ordre dans la fonction :

- Le vecteur de la requête représentant la dimension sujet.
- Un vecteur pour la localisation qui peut prendre une seule valeur.
- Le niveau de granularité choisi pour la dimension localisation.
- Le mode de requêtage avec la dimension localisation (strict/distance administrative).
- La date d'envoi de CV.
- Le mode de calcule avec la dimension date d'envoi (intervalle/mois,semestre,année).
- Le niveau de granularité utilisé pour la dimension date d'envoi.
- L'usage de la préférence des décideurs, possible seulement avec le calcul de la distance administrative.
- L'importance de la dimension sujet.
- L'importance de la dimension localisation (la somme des deux valeurs == 1).

les modes qui peuvent être générés sont :

| | qry loc | loc lvl | loc mode | sbt | sbt mode | sbt lvl | imp | loc imp | top imp |
|---|---|---|---|---|---|---|---|---|---|
| a | 1val | 1-6 | 0 | 1val | 0 | 3-5 | 0 | 0.5 | 0.5 |
| b | 1val | 1-6 | 0 | +2val | 1 | 1-5 | 0 | 0.5 | 0.5 |
| c | 1-6 | 1-6 | 1 | 1val | 0 | 3-5 | 0 | 0.5 | 0.5 |
| d | 1-6 | 1-6 | 1 | 1val | 0 | 3-5 | 1 | ]0,1[ | ]0,1[ |
| e | 1-6 | 1-6 | 1 | +2val | 1 | 1-5 | 0 | 0.5 | 0.5 |
| f | 1-6 | 1-6 | 1 | +2val | 1 | 1-5 | 1 | ]0,1[ | ]0,1[ |

FIGURE 48 – modes des requêtes.

qry_loc prend une valeur unique si on considère la localisation comme une contrainte de restriction (pour les habitant de qry_loc = "Blida") ; ainsi qu'elle peut prendre entre 1-6 valeurs si on applique le calcule de la distance

administrative.
loc_lvl permet de définir le niveau de granularité choisi dans les deux modes pour la dimension localisation.
loc_mode 0 pour une contrainte stricte et 1 pour effectuer le calcule de la distance administrative pour la dimension localisation.
sbt la date d'envoi qui peut prendre une valeur pour les niveaux de grain mois, semestre, année; ou plus de 2 valeurs sur les niveaux hiérarchique mois, trimestre, année et seulement deux valeur d'intervalle pour timestamp et date.(ex : val1=10/11/2012 et val2=15/12/2012).
sbt_mode définit le mode pour la dimension date d'envoi; 0 pour une valeur unique et 1 pour l'intervalle.
sbt_lvl pour définir le niveau de granularité.
imp pour choisir l'importance d'une dimension en tant que préférence utilisateur; 0 pour ne pas utiliser le mode et 1 pour le faire.
loc_imp , top_imp représentent les préférences utilisateurs pour les dimensions sujet et localisation respectivement.

— les 3 figure suivantes contiennent des exemples sur ces modes :

```
SELECT /*+ MAPJOIN(topic,fact_table,location,subtime)*/
top_id,((0+ topic.top1*1e-04+topic.top2*0.01+ ... +
          topic.top198*0.01)/(SQRT(0.0301*
fact_table.pow))*0.5)  WHERE location.admin2 LIKE  %
blida%)*0.5 AND  subtime.sub_month == 1  AS RANK FROM
fact_table JOIN topic ON topic.top_id ==
fact_table.top_id JOIN location ON location.loc_id ==
fact_table.loc_id JOIN subtime ON subtime.sub_id ==
fact_table.sub_id ORDER BY RANK;
```

---

```
SELECT /*+ MAPJOIN(topic,fact_table,location,subtime)*/
top_id,((0+ topic.top1*1e-04+topic.top2*0.01+ ... +
          topic.top198*0.01)/(SQRT(0.0301*
*0.5) WHERE location.admin2 LIKE  %blida%  AND
subtime.sub_year== 2010 OR subtime.sub_year== 2011 OR
subtime.sub_year== 2012) AS RANK FROM fact_table JOIN
topic ON topic.top_id == fact_table.top_idJOIN
location ON location.loc_id == fact_table.loc_idJOIN
subtime ON subtime.sub_id == fact_table.sub_id ORDER
BY RANK;
```

FIGURE 49 – Requêtes HQL générées avec HQL_jdbc_ (modèles a,b).

— a) pour un profile x pour la dimension sujet , une localisation du deuxième niveau administratif "blida" et d'une date d'envoi durant le mois de janvier.
— b) pour un profile x pour la dimension sujet , une localisation du

deuxième niveau administratif "blida" et d'une date d'envoi durant les années 2010,2011,2012.

```
SELECT /*+ MAPJOIN(topic,fact_table,location,subtime)*/
top_id,((0+ topic.top1*1e-04+topic.top2*0.01+ ... +
          topic.top198*0.01)/(SQRT(0.0301*
fact_table.pow))*0.5)+(   ((CASE WHEN
location.country  LIKE  '%algérie%' THEN 1 ELSE 0 END)
+     (CASE WHEN location.admin1  LIKE  '%blida%' THEN
1 ELSE 0 END )+     (CASE WHEN location.admin2  LIKE
'%blida%' THEN 1 ELSE 0 END ))/3  )*0.5 WHERE
subtime.sub_quarter== 2) AS RANK FROM fact_table JOIN
topic ON topic.top_id == fact_table.top_id JOIN
location ON location.loc_id == fact_table.loc_id JOIN
subtime ON subtime.sub_id == fact_table.sub_id ORDER
BY RANK;
```

---

```
SELECT /*+ MAPJOIN(topic,fact_table,location,subtime)*/
top_id,((0+ topic.top1*1e-04+topic.top2*0.01+ ... +
          topic.top198*0.01)/(SQRT(0.0301*
fact_table.pow))*0.25)+(   ((CASE WHEN
location.country  LIKE  '%algérie%' THEN 1 ELSE 0 END)
+     (CASE WHEN location.admin1  LIKE  '%blida%' THEN
1 ELSE 0 END )+     (CASE WHEN location.admin2  LIKE
'%blida%' THEN 1 ELSE 0 END ))/3  )*0.75 WHERE
subtime.sub_quarter== 1) AS RANK FROM fact_table JOIN
topic ON topic.top_id == fact_table.top_id JOIN
location ON location.loc_id == fact_table.loc_id JOIN
subtime ON subtime.sub_id == fact_table.sub_id ORDER
BY RANK;
```

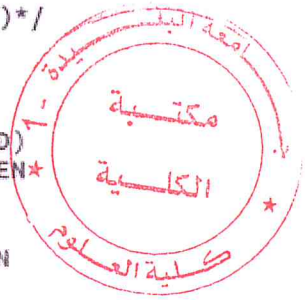


FIGURE 50 – Requêtes HQL générées avec HQL_jdbc_ (modèles c,d).

- c) pour un profile x pour la dimension sujet , une localisation du deuxième niveau administratif "blida" (usage de la distance administrative ) et d'une date d'envoi durant le deuxième trimestre.
- d)pour un profile x pour la dimension sujet , une localisation du deuxième niveau administratif "blida" (usage de la distance administrative) et d'une date d'envoi durant le deuxième trimestre, dans cette requête le décideur a donné une importance de 25% à la dimension sujet et de 75% à la dimension localisation.

```
SELECT /*+ MAPJOIN(topic,fact_table,location,subtime)*/
top_id,((0+topic.top1*1e-04+topic.top2*0.01+ ... +
          topic.top198*0.01)/(SQRT(0.0301*
fact_table.pow))*0.5)+(   ((CASE WHEN
location.country  LIKE  '%algérie%' THEN 1 ELSE 0 END)
+     (CASE WHEN location.admin1  LIKE  '%blida%' THEN
1 ELSE 0 END )+     (CASE WHEN location.admin2  LIKE
'%blida%' THEN 1 ELSE 0 END ))/3  )*0.5 WHERE
sub_time.sub_tstamp BETWEEN  '10/02/2012 00:00:00' AND
'20/03/2012 00:00:00') AS RANK FROM fact_table JOIN
topic ON topic.top_id == fact_table.top_id JOIN
location ON location.loc_id == fact_table.loc_id JOIN
subtime ON subtime.sub_id == fact_table.sub_id ORDER
BY RANK;
```

---

```
SELECT /*+ MAPJOIN(topic,fact table,location,subtime)*/
top_id,((0+topic.top1*1e-04+topic.top2*0.01+ ... +
          topic.top198*0.01)/(SQRT(0.0301*
fact_table.pow))*0.7)+(   ((CASE WHEN
location.country  LIKE  '%algérie%' THEN 1 ELSE 0 END)
+     (CASE WHEN location.admin1  LIKE  '%blida%' THEN
1 ELSE 0 END )+     (CASE WHEN location.admin2  LIKE
'%blida%' THEN 1 ELSE 0 END ))/3  )*0.3 WHERE
sub_time.sub_tstamp BETWEEN  '10/02/2012 00:00:00' AND
'20/03/2012 00:00:00') AS RANK FROM fact_table JOIN
topic ON topic.top_id == fact_table.top_id JOIN
location ON location.loc_id == fact_table.loc_id JOIN
subtime ON subtime.sub_id == fact_table.sub_id ORDER
BY RANK;
```

FIGURE 51 – Requêtes HQL générées avec HQL_jdbc_ (modèles e,f).

— e)pour un profile x pour la dimension sujet , une localisation du deuxième niveau administratif "blida" (usage de la distance administrative) et d'une date d'envoi entre "10/02/2012 00 :00 :00" et "20/03/2012 00 :00 :00".

— f)pour un profile x pour la dimension sujet , une localisation du deuxième niveau administratif "blida" (usage de la distance administrative) et d'une date d'envoi entre "10/02/2012 00 :00 :00" et "20/03/2012 00 :00 :00", dans cette requête le décideur a donné une importance de 70% à la dimension sujet et de 30% à la dimension localisation.

**connexion à Hive avec RHive :**

RHive prend en charge les fonctions de base, HDFS et MapReduce, et les fonctions avancées utilisé pour l'implémentation des UDF, UDAF et UDTF

avec R. Les fonctions avancées sont composées de fonctions UDF / UDAF existantes pour implémenter des UDF et UDAF avec R; l'usage ces fonctions, permet d'appliquer des calculs R aux niveaux inférieurs de complexité et la programmation Hive avec MapReduce. l'usage de R permet d'intégrer des opérateurs contenant des algorithmes complexes ou procédures de traitement exhaustives au niveau de Hive.
Hive fournit des UDF (User Defined Function), UDAF (User Defined Aggregate Function) Function), et UDTF (User Defined Table create Function). ces fonctions font référence aux fonctions contenues dans Hive, elles utilisent une syntaxe SQL telle que count, avg, min et max pour effectuer des calculs pour une colonne ou plusieurs colonnes. à l'origine,les UDF, UDAF et UDTF doivent être écrites en seulement en Java; RHive permet de surpasser cet obstacle et facilite l'écriture de ces fonctions avec le langage R et les rendre utilisables dans Hive, rendant la division d'une colonne complexe en plusieurs colonnes appropriées aux calcules. la figure suivante montre l'architecture de RHive :

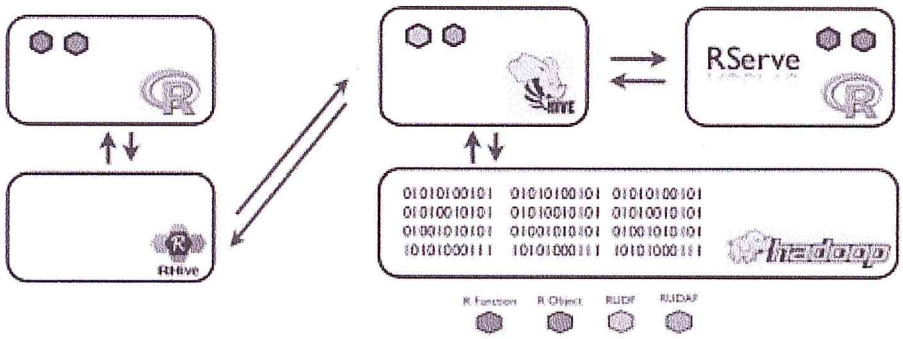



FIGURE 52 – Architecture de RHive.

RHive permet de créer des Objet R contenant des procédures afin de les intégrer en tant que UDF, UDAF ou UDTF. Rserve permet d'établir la connexion entre chaque nœud R et le système Hadoop ainsi qu'entre les instances R.
le code de source suivant permet de lancer RHive :

```
Sys.setenv('HADOOP_CMD'='/usr/hdp/2.4.0.0-169/hadoop/bin/hadoop')
Sys.setenv('HADOOP_HOME'='/usr/hdp/2.4.0.0-169/hadoop')
Sys.setenv('HADOOP_LIB'='/usr/hdp/2.4.0.0-169/hadoop/lib/')
Sys.setenv('HIVE_HOME'='/usr/hdp/2.4.0.0-169/hive')
Sys.setenv('HIVE_LIB'='/usr/hdp/2.4.0.0-169/hive/lib')
Sys.setenv('TEZ_HOME'='/usr/hdp/2.4.0.0-169/tez')
Sys.setenv('HCAT_HOME'='/usr/hdp/2.4.0.0-169/hive-hcatalog')
library(RHive)
rhive.init()
rhive.env()
rhive.connect('192.168.1.102',user = 'hive',password = 'hive')
```

FIGURE 53 – connexion à Hive avec RHive.

Via un Object R le code de source suivant, permet d'intégrer L'UDF calculant la similarité (sujet) dans la formulation de la requête utilisant l'UDF crée :

```
HQL_rhive_= function(qry_top,qry_loc,loc_lvl,loc_mode,sbt,
                     sbt_mode,sbt_lvl,imp,loc_imp,top_imp)
{
  A <- 0
  B <- 0
  C <- NULL
  E <- NULL

  similarity <- function(top1,top2,,,top197,top198,pow)
  {(  top1*qry_top[1] + top2*qry_top[2] + + +
        top197*qry_top[197] + top198*qry_top[198])/
         sqrt(sum(qry_top^2)*pow)}

  A <-"R('similarity', top1, top2, , ,top197 ,top198 ,pow)"
  rhive.asign("similarity", similarity)
  rhive.export("similarity")

  if(loc_mode == 0)
  {...}
  else
  {...}
  if(sbt_mode == 0 && loc_mode == 0 && typeof(sbt) == "double")
  {...}
  if(sbt_mode == 1 && loc_mode == 0)
  {...}
  if(sbt_mode == 0 && loc_mode == 1 && typeof(sbt) == "double")
  {...}
  if(sbt_mode == 1 && loc_mode == 1)
  {
    if(typeof(sbt) == "double")
    {...}
    else
    {...} }
  if(imp == 0){...}
  QRY <- paste0("SELECT /*+ MAPJOIN(topic,fact_table, location,subtime)*/
                top_id,((",A,")/(SQRT(",B,"*pow))*",top_imp,")",
                " (",C,")*",loc_imp,E,")
                AS RANK FROM fact_table
                JOIN topic ON topic.top_id == fact_table.top_id
                JOIN location ON location.loc_id == fact_table.loc_id
                JOIN subtime ON subtime.sub_id == fact_table.sub_id
                ORDER BY RANK;")
  QRY <- str_replace_all(QRY,'\n','')
  QRY <- str_replace_all(QRY,'       ','')
  QRY_Fth <- rhive.query(QRY)
  return(QRY_Fth)
}
```

FIGURE 54 – Génération d'UDF depuis R.

Dans ce code nous avons implémenté une UDF pour les calculs relatifs à la dimension sujet et les opérations sur les deux dimensions restantes sont implémentées directement avec HQL et transmises avec RHive, permettant d'assurer le calcul avec les six modes décrit implémenté avec une connexion RJDBC (paramétrage/paramètres d'entrée sont les mêmes). L'Objet similartiy contient une procédure qui permet de calculer la similarité requête/document ayant en paramètre les arguments qui représente les colonnes requises pour le calcul. la fonction rhive.assign() permet à l'Objet "similarity" d'être référencé par Hive, et la fonction rhive.export() effectue le déploiement de l'UDF en mode distribué sur toutes les instances R par le biais de RServe.

### 5.3.4 Connexion à Kylin et requêtage

Avant de connecter au projet e_recruite nous devons lancer Kylin sur les quatre machines avec le script (/usr/Kylin/bin/kylin.sh start et lancer le script R suivant :

```
1  Sys.setenv('HADOOP_CMD'='/usr/hdp/2.4.0.0-169/hadoop/bin/hadoop')
2  Sys.setenv('HADOOP_HOME'='/usr/hdp/2.4.0.0-169/hadoop')
3  Sys.setenv('HADOOP_LIB'='/usr/hdp/2.4.0.0-169/hadoop/lib/')
4  Sys.setenv('HIVE_HOME'='/usr/hdp/2.4.0.0-169/hive')
5  Sys.setenv('HIVE_LIB'='/usr/hdp/2.4.0.0-169/hive/lib/')
6  Sys.setenv('TEZ_HOME'='/usr/hdp/2.4.0.0-169/tez')
7  Sys.setenv('HCAT_HOME'='/usr/hdp/2.4.0.0-169/hive-hcatalog')
8  Sys.setenv('KYLIN_HOME'='/usr/kylin')
9  library(RJDBC)
10 for(l in list.files('/usr/hdp/2.4.0.0-169/hive/lib/'))
11   { .jaddClassPath(paste("/usr/hdp/2.4.0.0-169/hive/lib/",l,sep=""))}
12 for(l in list.files('/usr/hdp/2.4.0.0-169/hadoop/'))
13   { .jaddClassPath(paste("/usr/hdp/2.4.0.0-169/hadoop/",l,sep=""))}
14 for(l in list.files('/usr/hdp/2.4.0.0-169/tez/'))
15   { .jaddClassPath(paste("/usr/hdp/2.4.0.0-169/tez/",l,sep=""))}
16 for(l in list.files('/usr/kylin/'))
17   { .jaddClassPath(paste("/usr/kylin/",l,sep=""))}
18
19 options( java.parameters = "-Xmx8g" )
20 .jinit()
21 .jclassPath()
22
23 drv <- JDBC("org.apache.kylin.jdbc.Driver")
24 conn <- dbConnect(drv,"jdbc:kylin://hdmaster.dotes.mat:7070/e_recruite")
```

FIGURE 55 – Connexion à Kylin via JDBC driver.

la formulation des requêtes Kylin, pour une requête via RJDBC sera en six modes, sauf pour cette connexion nous n'avons pas à spécifier les jointures entres les trois dimension car nous l'avons fait durant l'implémentation du cube e_recruite . le script suivant permet de générer les requêtes et les soumettre à Hive via RJDBC :

```
Kylin_jdbc_ = function(qry_top,qry_loc,loc_lvl,loc_mode,
                       sbt,sbt_mode,sbt_lvl,imp,loc_imp,top_imp)
{
  A <- 0
  B <- sum(qry_top^2)
  C <- NULL
  E <- NULL

  for(i in 1:length(qry_top))
  {   if(qry_top[i] != 0)
    A <- paste0(A,"+ topic.top",i,"*",qry_top[i])

  }
  if(loc_mode == 0)
  {...}
  else
  {...}
  if(sbt_mode == 0 && loc_mode == 0 && typeof(sbt) == "double")
  {...}
  if(sbt_mode == 1 && loc_mode == 0)
  {...}
  if(sbt_mode == 0 && loc_mode == 1 && typeof(sbt) == "double")
  {...}
  if(sbt_mode == 1 && loc_mode == 1)
  {
    if(typeof(sbt) == "double")
    {...}
    else
    {...} }
  if(imp == 0){...}
  QRY <-

    paste0("SELECT top_id,((",A,")/(SQRT(",B,"*pow))*",top_imp,
           ")"," (",C,")*",loc_imp,E,")
           AS RANK FROM e_recruite ORDER BY RANK;")
  QRY <- str_replace_all(QRY,'\n','')
  QRY <- str_replace_all(QRY,'       ','')
  QRY_s <- dbSendStatement(conn,QRY)
  QRY_s <- dbFetch(QRY_s)
  return(QRY_s)
}
```

FIGURE 56 – Requête Kylin avec RJDBC, source R.

### 5.3.5 évaluation des performances

Afin de faire une conclusion sur le système et le modèle CXT-Cube nous avons opté pour une comparaison entre les requêtes RJDBC et RHive utilisant des UDFs ; Le grand défi et toujours la scalabitlité verticale, pour cette

raison le scénario idéal est d'augmenter le nombre de colonnes pour la dimension sujet et voir les temps des calculs.
pour une requête d'une seule colonne : cherchant un candidat qui maitrise 'Statistiques et informatique décisionnel', qui habite en 'France' et qui a envoyé son CV durant l'an '2012'.
pour une requête de deux colonnes : cherchant un candidat qui maitrise 'Statistiques et informatique décisionnel - programmation ', qui habite en 'France' et qui a envoyé son CV durant l'an '2012'.
pour une requête de trois colonnes : cherchant un candidat qui maitrise 'Statistiques et informatique décisionnel - Programmation - Anglais', qui habite en 'France' et qui a envoyé son CV durant l'an '2012'.
pour une requête de quatre colonnes : cherchant un candidat qui maitrise 'Statistiques et informatique décisionnel - programmation - Anglais - Chinois', qui habite en 'France' et qui a envoyé son CV durant l'an '2012'.
le graph suivant regroupe les résultats :

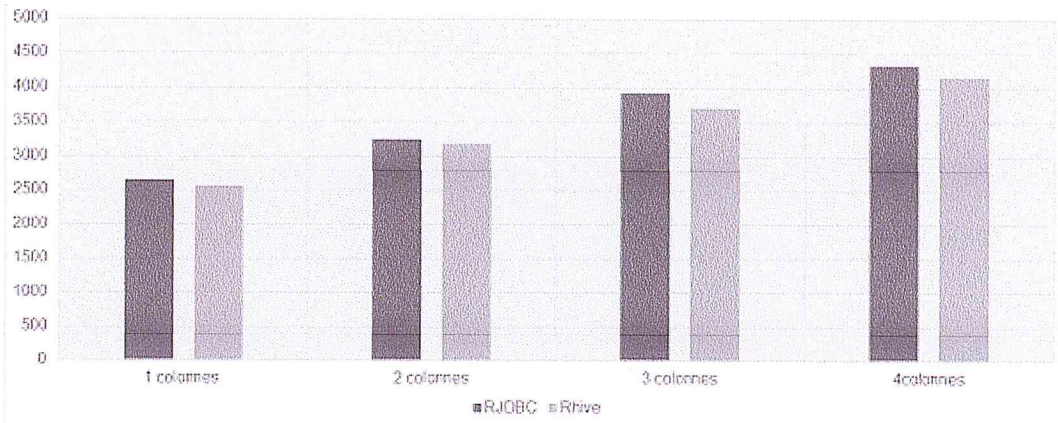



FIGURE 57 – évaluation des performances par comparaison entre requêtes RJDBC/RHive.

l'observation des résultats obtenus montre que l'utilisation des UDFs via RHive a permit d'améliorer le temps de réponse ; On remarque également et pour les quatre requêtes que le temps d'exécution augmente légèrement quand on utilise plus de colonnes ; un critère qui résume la scalabilité verticale de notre système et du modèle utilisé.
Nous voulions comparaitre avec les temps de réponse de Kylin, mais malheureusement nous étions empêché pour des raisons que nous ne pouvons pas les expliquées dans ce document.

### 5.3.6 Conclusion

Dans ce dernier chapitre nous avons vu l'implémentation du cluster HDR en mode distribué sur quatre machines Ubuntu, l'installation de l'environnement R ainsi que le cluster Kylin. Cette infrastructure nous a permit d'effectuer le processus ETL pour déployer un système OLAP au niveau de Hive et Kylin, et l'interrogation via le connecteur JDBC pour Hive/Kylin et avec le mode distribué avec RHive. la génération des requête pour intéroger le modèle CXT-Cube en six modes de requêtes
nous souhaitons dire que les solutions OLAP pour les grandes masses de données sont en cours de développement et manquent de de fonctionnalités, mais n'empêche qu'ils effectuent la tâche parfaitement pour pallier aux problèmes de scalabilité actuels.

# 6 Conclusion Générale

Dans Ce projet, nous avons Conçu un système d'ADT qui suivant le modèle CXT-Cube et en utilisant des modèles de requêtes basés sur la mesure de similarité ; la réalisation est faite pour l'analyse des millions de documents indexés et stockés dans une base de données NoSQL (Hive) et en utilisant une technologie implémentée au sommet de Hive qui est apache Kylin. Notre système s'exécute en mode distribué au niveau d'un cluster RHadoop permettant de bénéficier de la performance exemplaire de R et celle des langages HQL/ ANSI-SQL appuyés par le modèle de programmation MapReduce et le Framework Tez. Ce project a également montré que la phase la plus importante est l'indexation. Cette dernière et très importante pour la valorisation des données, la qualité du processus se reflète sur l'analyse où la mauvaise représentation sera impérativement contre intuitive pour la prise de décision. nous avons acquis une vision sur les solutions d'Apache dédiées aux Big Data et les architectures existantes au sommet de Hadoop ainsi que l'approche CXT-Cube rendant une masses de données textuelles non structurée une source importante pour le décisionnel.
Dans cette conclusion nous souhaitons mettre le point sur le concept plateforme qui est loin d'être une simple implémentation ; la réalisation d'un projet pareil dans un environnement productif sera mis à suivre les pratiques formelles pour le bon fonctionnement des services et leurs rendements. Notre tâche principale était la mise en œuvre des composants essentiels pour le fonctionnement dans un environnement de testes, ce qui représente au niveau de l'entreprise la phase A du projet. Cette dernière sera suivit par une deuxième phase B qui rend le système plus approprié à l'utilisation en simplifiant tous les processus par l'ajout des interfaces de visualisation, des pipeline Machine Learning pour l'intégration et un plan de route pour le développement durable de chaque composant du système ; par exemple, l'usage des techniques d'apprentissage durant l'indexation ; L'ensemble de ces mesures permet d'optimiser les ressources et le code afin de mieux répondre à la scalabilité.

# 7 références


1. An Overview of Business Intelligence Technology By Surajit Chaudhuri, Umeshwar Dayal, Vivek Narasayya Communications of the ACM, Vol. 54 No.8 August 2011
2. www.tutorialspoint.com. (2016). Data Warehousing Tutorial. [online] Available at : https ://www.tutorialspoint.com/dwh/index.htm
3. Tournier, R. (2007). Analyse en ligne (OLAP) de documents. L'UNIVERSITE DE PAUL SABATIER , TOULOUSE III, France.
4. Salton, G. and McGill, M. (1983). Introduction to modern information retrieval. New York : McGraw-Hill.
5. roft, W. and Lafferty, J. (2003). Language modeling for information retrieval. Dordrecht : Kluwer Academic Publishers.
6. Zemirli WN, (2004) Vers le développement d'un système de recherche d'information personnalisé intégrant le profil utilisateur. L'UNIVERSITE DE PAUL SABATIER , TOULOUSE III, France.
7. Manning, C., Raghavan, P. and Schutze, H. (2008). Introduction to information retrieval. New York : Cambridge University Press.
8. Overview of the ninth Text REtrieval Conference (TREC-9) By Ellen M Voorhees, Donna Harman National Institute of Standards and Technology Gaithersburg, MD 20899 November 2000.
9. Robertson, SE. (1997) THE PROBABILITY RANKING PRINCIPLE IN IR. University College, London. journal of documentation 33 1997 294-304 England.
10. Miner, G. (2012). Practical text mining and statistical analysis for non-structured text data applications. Amsterdam : Academic Press.
11. Felman, R. et Sanger, J. (2007). The text mining handbook advanced approaches in analyzing unstructured data. New York, NY, USA : Cambridge university press.
12. Strauch, C Stuttgart Media university(2011). Selected Topics on Software-Technology Ultra-Large Scale Sites lecture(NoSQL Databases).
13. Meyer, L. (2014). L'avenir du NoSQL. 1st ed.
14. Fr.wikipedia.org. (2017). Lemmatisation. [online] Available at : https ://fr.wikipedia.org/wiki/Lemmatisation [Accessed 14 Jan. 2017].
15. Fr.wikipedia.org. (2017). Racinisation. [online] Available at : https ://fr.wikipedia.org/wiki/Racinisation [Accessed 14 Jan. 2017].

16. Oukid, L et al. "CXT-cube : contextual text cube model and aggregation operator for text OLAP." DOLAP (2013).
17. Wiki.apache.org. (2017). FrontPage - Hadoop Wiki. [online] Available at : https ://wiki.apache.org/Hadoop [Accessed 9 Feb. 2017].
18. NOVA, L. (2017). Hive le Data Warehouse de Hadoop | Le SI de demain se façonne aujourd'hui. [online] Linusnova.com. Available at : http ://www.linusnova.com/2013/09/hive-le-data-warehouse-de-Hadoop/ [Accessed 26 Apr. 2017].
19. Cwiki.apache.org. (2017). Home - Apache Hive - Apache Software Foundation. [online] Available at : https ://cwiki.apache.org/confluence/display/Hive/Home [Accessed 26 Apr. 2017].
20. Kylin, A. (2017). Apache Kylin | Home. [online] Kylin.apache.org. Available at : http ://kylin.apache.org/ [Accessed 30 Apr. 2017].
21. Hadoop.apache.org. (2017). Apache Hadoop 2.8.0? HDFS NFS Gateway. [online] Available at : https ://Hadoop.apache.org/docs/r2.8.0/Hadoop-project-dist/Hadoop-hdfs/HdfsNfsGateway.html [Accessed 6 Sep. 2017].
22. Hadoop.apache.org. (2017). Apache Hadoop 2.7.2? HDFS Users Guide. [online] Available at : https ://Hadoop.apache.org/docs/r2.7.2/Hadoop-project-dist/Hadoop-hdfs/HdfsUserGuide.html [Accessed 6 Sep. 2017].
23. Hadoop.apache.org. (2017). Apache Hadoop 2.4.1 - YARN Timeline Server. [online] Available at : https ://Hadoop.apache.org/docs/r2.4.1/Hadoop-yarn/Hadoop-yarn-site/TimelineServer.html [Accessed 6 Sep. 2017].
24. Tez.apache.org. (2017). Apache Tez? Welcome to Apache Tez?. [online] Available at : https ://tez.apache.org/ [Accessed 6 Sep. 2017].
25. Cwiki.apache.org. (2017). WebHCat UsingWebHCat - Apache Hive - Apache Software Foundation. [online] Available at : https ://cwiki.apache.org/confluence/display/Hive/WebHCat+Using WebHCat [Accessed 6 Sep. 2017].
26. Hbase.apache.org. (2017). Apache HBase? Apache HBase? Home. [online] Available at : https ://hbase.apache.org/ [Accessed 6 Sep. 2017].
27. Blogs.apache.org. (2017). HBase - Who needs a Master? : Apache HBase. [online] Available at : https ://blogs.apache.org/hbase/entry/hbase_who_needs_a_master [Accessed 6 Sep. 2017].

39. Cloudera, I. (2017). Yarns About Yarn. [online] Slideshare.net. Available at : https ://www.slideshare.net/cloudera/yarn-sabout-yarnkathleenting112114 [Accessed 5 May 2017].

40. Lebigdata.com. (2017). Mapreduce | Le Big Data. [online] Available at : https ://lebigdata.com/fr/mapreduce/ [Accessed 6 May 2017].

41. Enterprise Just Builder. (2017). Apache Hive Architecture. [online] Available at : http ://www.ejb.cc/archives/4290/hadoop-technical-manuals-a-the-hadoop-ecosystem/hive-architecture [Accessed 10 May 2017].

42. Slideshare.net. (2017). Apche hive. [online] Available at : https ://www.slideshare.net/pavan5780/apche-hive [Accessed 10 May 2017].

43. Kylin, A. (2017). Apache Kylin | Home. [online] Kylin.apache.org. Available at : http ://kylin.apache.org/ [Accessed 10 May 2017].

44. Wikis.nyu.edu. (2017). Big Data Tutorial 1 : MapReduce - High Performance Computing at NYU - NYU Wikis. [online] Available at : https ://wikis.nyu.edu/display/NYUHPC/Big+Data+Tutorial+1%3A + MapReduce [Accessed 4 May 2017].