الجمـــهوريـــة الجـــزائريـــة الديـمقراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي والـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Mémoire de MASTER

Domain : Mathématiques et Informatique
Filière : Informatiques
Option : Systèmes d'Information et de Décision

Présenté Par :
Mohamed Ayoub Kachna & **Youcef Radouane Benchekal**

THEME

Mise en œuvre d'un entrepôt de données avec des techniques big data

Soutenu publiquement le 07-07-2021 devant le jury composé de :

Mr Laaradj Chellama	*M.A.A*	*Président*
Mr Mohamed Maicha	*M.A.A*	*Examinateur*
Mr Ziani Bennameur	*M.C.A*	*Encadreur*

Année Universitaire 2020/2021

Résumé

Le Big Data est un terme qui décrit le grand volume de données géré au quotidien par les organisations. Ces dernières ont besoin d'analyser leurs données afin d'en extraire des informations utiles à la prise de décisions stratégiques. Cependant, les outils traditionnels d'analyse sont inadéquats pour traiter les grands volumes de données.

Le travail présenté dans ce mémoire a pour objectif d'utiliser des techniques Big Data pour surmonter ce problème d'analyse. Plus précisément, nous avons installé et configuré l'écosystème Hadoop et l'outil Apache Hive dans un cluster multi-nœuds pour pouvoir interroger efficacement les entrepôts de données relationnels avec le langage SQL.

Nous avons réalisé un ensemble de tests pour valider nos propositions.

Mots clés :

Systèmes décisionnels, entrepôt de données, schéma en étoile, requêtes OLAP, Big Data, Hadoop, Hive, MapReduce.

Abstract

Big Data is a term that describes the large volume of data managed by organizations on a daily basis. They need to analyze their data in order to extract useful information for strategic decision-making. However, traditional analysis tools are incapable to handle large volumes of data.

The work presented in this thesis aims to use Big Data techniques to overcome this analysis problem. Specifically, we installed and configured the Hadoop ecosystem and the Apache Hive tool in a multi-node cluster to be able to efficiently query relational data warehouses with SQL language.

We carried out a set of tests to validate our proposals.

Key Words:

Decision systems, data warehouse, star schema, OLAP queries, big data, Hadoop, Hive, MapReduce.

ملخص

البيانات الضخمة هو مصطلح يصف الحجم الكبير من البيانات التي تديرها المؤسسات على أساس يومي. يحتاجون إلى تحليل بياناتهم من أجل استخراج معلومات مفيدة لاتخاذ القرارات الاستراتيجية. ومع ذلك، فإن أدوات التحليل التقليدية غير قادرة على التعامل مع كميات كبيرة من البيانات.

يهدف العمل المقدم في هذه الأطروحة إلى استخدام تقنيات البيانات الضخمة للتغلب على مشكلة التحليل هذه. على وجه التحديد، قمنا بتثبيت وتهيئة نظام Hadoop البيئي وأداة Apache Hiveفي مجموعة متعددة العقد حتى نتمكن من الاستعلام بكفاءة عن مستودعات البيانات العلائقية باستخدام لغةSQL.

لقد أجرينا مجموعة من الاختبارات للتحقق من صحة مقترحاتنا.

Remerciements

Nous remercions le Dieu Tout-Puissant de nous avoir donné la santé et la volonté de commencer et de terminer ce mémoire. Tout d'abord, nous tenons à remercier M. Ziani pour son aide et son encadrement et sans lui, la réalisation de ce travail ne serait pas possible.

Nous tenons également à remercier les professeurs avec lesquels nous avons passé des années de notre vie à l'université, les professeurs qui nous ont donné le meilleur d'eux-mêmes.

A cette occasion, nous avons tenu à remercier les membres du jury pour leur évaluation et l'enrichissement de ce travail. Nous sommes honorés d'être devant vous aujourd'hui.

Nous tenons à adresser nos remerciements les plus chaleureux et les plus sincères à toutes les personnes qui nous ont contribué et aidé à la réalisation de ce travail.

Dédicace

Je dédie ce travail

Aux deux personnes les plus importantes de ma vie, mes parents, qui n'ont pas hésité un instant à m'apporter aide et soutien, qui ont éclairé et décoré mon chemin avec leurs conseils, et m'ont motivé pour continuer mes études.

A mes sœurs Rofaida, Bouthaina, et mes frères Ilias et Akram.

A tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Merci d'être toujours là pour moi.

Mohamed Ayoub Kachna

Je dédie ce travail

A mon père, qui sans son soutien continu, je n'aurais pas atteint cet endroit.

Et à ma mère, qui ne m'a pas privé de ses prières et a cru en moi.

A mes frères qui me demandent toujours et qui me manquent, et à mon frère Mohamad Achraf, à qui je souhaite qu'il se remette de sa maladie

Et à tous mes amis de près ou de loin. Merci.

Youcef Radouane Benchekal

Table des Matières

Liste des Tableaux

Table de figures

Introduction Générale

Avant l'apparition du terme big data il y a environ 15 ans, l'analyse et le traitement de téraoctets de données étaient une tâche difficile et très coûteuse.

Le grand volume de données a entraîné la nécessité de changements importants dans l'architecture des systèmes de stockage et de traitement, de nos jours, par exemple, les données sont constamment générées chaque fois que nous ouvrons une application, que nous recherchons Google ou que nous nous déplaçons simplement d'un endroit à l'autre avec nos appareils mobiles.

Les outils de données traditionnels ne sont pas équipés pour gérer ces énormes collections d'informations précieuses dont les entreprises et les organisations ont besoin pour gérer, stocker, visualiser et analyser.

À partir de 2003, Google a développé une nouvelle technologie comportant deux composants principaux : Google File System (GFS) et MapReduce.

Google a créé ce système de fichiers distribué évolutif pour les grandes applications distribuées gourmandes en données, et la même année, un concepteur de logiciels chez Yahoo, Doug Cutting, a commencé à travailler sur un cadre logiciel open source pour prendre en charge la distribution du projet de moteur de recherche Nutch, basé sur les mêmes idées suggérées par Google, donc Hadoop est né.

Hadoop est un Framework qui permet le traitement distribué de grands ensembles de données sur des clusters d'ordinateurs, il fournit un stockage massif pour tout type de données, une énorme puissance de traitement et la capacité de gérer des tâches illimitées.

Hadoop est basé sur le système de fichiers distribués Hadoop (HDFS) qui permet aux données d'être distribuées sur plusieurs nœuds [1].

« Écrire une fois et lire plusieurs fois » Hadoop est basé sur cette technologie, les données sont lues en parallèle avec un temps considérablement réduit même le processus d'écriture prend plus de temps.

Il y avait un problème avec l'utilisation d'Hadoop car il est écrit en langage Java et les programmeurs SQL n'étaient pas habitués à le gérer, alors Facebook a développé un outil pour traiter les données structurées stockées dans Hadoop et l’a nommé Hive.

Hive est un outil d'infrastructure d'entrepôt de données pour traiter des données structurées dans Hadoop, il a été initialement développé par Facebook, plus tard, Apache Software Foundation l'a repris et l'a développé en tant qu'open source sous le nom d'Apache Hive. Il est utilisé par différentes entreprises. Par exemple, Amazon l'utilise dans Amazon Elastic MapReduce [1][5][18].

Dans notre projet, nous allons essayer de simuler le travail de Hadoop avec Hive en utilisant nos propres PC pour comprendre comment cela fonctionne vraiment.

Nous allons commencer par l'exemple simple et populaire "WordCount".

L'exemple WordCount lit les fichiers texte et compte la fréquence à laquelle les mots apparaissent. L'entrée et la sortie sont constituées de fichiers texte.

Ensuite, nous allons tester Hive avec des données volumineuses et des requêtes complexes pour évaluer le système Hadoop/Hive et quand cela vaut la peine d'être utilisé.

Les travaux présentés dans cette thèse abordent les problèmes liés à ce domaine d'application. L'objectif principal est d'exploiter les techniques d'analyse Big Data pour améliorer les performances d'un entrepôt de données. Le travail est centré sur la mise en œuvre de la conception existante en exploitant deux technologies big data l'écosystème Hadoop et l'outil Apache Hive.

Ce manuscrit est articulé autour de trois chapitres :

- **Une introduction** à notre travail de mémoire. L'objectif de ce chapitre est de présenter la problématique abordée et la solution que nous allons utiliser.
- **Le premier Chapitre :** est consacré à la présentation des bases de l'entrepôt de données. En particulier, nous décrivons l'architecture, modélisation multidimensionnelle d'entrepôt de données et sa alimentation grâce à un processus ETL (Extract Transform Load) pour l'extraction, la transformation et chargement des données.
- **Le deuxième Chapitre :** est consacré à la première phase de notre projet. Il décrit les deux technologies Big Data (Hadoop et Apache Hive) que nous utilisons pour améliorer les performances de l'entrepôt de données, ainsi que la description, l'architecture, les principaux composants et leur fonctionnement réel.
- **Le troisième Chapitre :** est dédié à la deuxième phase de notre projet. Il s'agit d'une phase technique et concerne l'installation d'Hadoop et d'Apache Hive et leur configuration dans un matériel limité comme celui dont nous disposons, aux côtés du test populaire WordCount.

  Dans ce chapitre, nous avons également parlé un peu de la base de données que nous avons utilisée, du test que nous avons fait pour voir l'efficacité de ces deux outils de Big Data et de la discussion des résultats.
- **Une Conclusion** consacrée à la dernière partie de notre manuscrit qui est la conclusion, quelques perspectives envisagées et avec la volonté de dieu un travail futur.

Chapitre 1

Notions De Base sur les entrepôts de données

1.1 - Introduction :

Dans ce chapitre, nous présentons les bases d'un entrepôt de données et quelques détails techniques sur ses fonctions ainsi les termes et les concepts clés liées à l'architecture et la modélisation d'un entrepôt de données.

1.2 - Un peu d'histoire :

Le concept d'entrepôt de données est apparu pour la première fois et introduit par IBM en 1988 par ses chercheurs Barry Devlin et Paul Murphy.

Le concept a tenté de répondre aux différents problèmes associés au flux de données des environnements opérationnels vers les environnements d'aide à la décision et les coûts élevés qui y sont associés [6].

- **1988 -** Barry Devlin et Paul Murphy publient l'article dans IBM Systems Journal où ils introduisent le terme « entrepôt de données d'entreprise ».
- **1990 -** Red Brick Systems, fondée par Ralph Kimball, lance Red Brick Warehouse, un système de gestion de base de données spécifiquement pour l'entreposage de données.
- **1991 -** Prism Soullusions, fondée par Bill Inmon présente Prism Warehouse Manager, un logiciel de développement d'un entrepôt de données.

1.3 - Définition :

Un entrepôt de données est un système informatique de rapporte et d'analyse qui rassemble les données d'une entreprise provenant de différentes sources, puis les utilise pour faciliter la prise de décision [10].

1.4 - Architecture d'un entrepôt de données :

Un entrepôt de données traditionnel s'appuie sur une architecture trois tiers composée des tiers suivants : [10][11].

1. **Niveaux inférieurs (Accès aux données) :** Contient le serveur de base de données, Il s'agit généralement d'un système de base de données relationnelle**,** les données sont extraites, transformées et chargées dans l'entrepôt de donneés.

2. **Niveau intermédiaire (Le traitement) :** Ce niveau, nous trouvons le serveur OLAP qui peut fonctionner de deux manières :
    a) ROLAP : Relational Online Analytical Processing, également appelées tables relationnelles, stocke les données dans des colonnes et des lignes et récupère les informations à la demande via des requêtes.
    b) MOLAP : Multidimensional Online Analytical Processing, qui implémente directement les données et opérations multidimensionnelles.
3. **Haut niveau (La présentation) :** Couche Client, Contient les outils d'analyse et les outils d'exploration de données.

1.4.1 - Extraction des données (Extract) :

Avant que les données puissent être déplacées vers une nouvelle destination, elles doivent d'abord être extraites de leur source, les données peuvent provenir de pratiquement n'importe quelle source structurée ou non structurée : serveurs SQL ou NoSQL, systèmes CRM et ERP, fichiers texte et documents, e-mails, pages Web, etc [12].

1.4.2 - Transformation des données (Transform) :

Dans cette phase, nous allons appliquer des règles et réglementations pour garantir la qualité et l'accessibilité des données et pour s'adapter au schéma de l'entrepôt de données cible éventuel. Cela peut concerner les éléments suivants :

- ❖ Nettoyage — les incohérences et les valeurs manquantes dans les données sont résolues.
- ❖ Normalisation — les règles de formatage sont appliquées à l'ensemble de données.
- ❖ Déduplication — les données redondantes sont exclues ou supprimées.
- ❖ Vérification — les données inutilisables sont supprimées et les anomalies sont signalées.
- ❖ Tri — les données sont organisées par type.

1.4.3 - Chargement des données (Load) :

La dernière étape du processus ETL consiste à charger les données transformées dans un entrepôt de données cible.

Un volume énorme de données doit être chargé dans une période relativement courte (nuits), il existe 3 types de chargement :

1. **Chargement initial :** remplissage de toutes les tables de l'entrepôt de données.
2. **Charge incrémentielle :** appliquant les changements en cours au besoin périodiquement.
3. **Rafraîchissement complet :** efface le contenu d'une ou plusieurs tables et recharge avec de nouvelles données.

1.5 - Fonctions des outils et utilitaires de l'entrepôt de données :

Voici les fonctions des outils et utilitaires de l'entrepôt de données : [11][12].

- **Extraction de données** - Implique la collecte de données à partir de plusieurs sources hétérogènes.
- **Nettoyage des données** - Implique la recherche et la correction des erreurs dans les données.
- **Transformation des données** - Implique la conversion des données du format hérité au format d'entrepôt.
- **Chargement des données** - Implique le tri, la synthèse, la consolidation, la vérification de l'intégrité et la création d'indices et de partitions.
- **Actualisation** - Implique la mise à jour des sources de données vers l'entrepôt.

1.6 – Concept de fait et dimension :

1.6.1 – Concept de fait :

Le fait est l'objet d'une analyse et le résultat d'une opération d’agrégation des données.

Les mesures sont stockées dans la table des faits, et généralement sont souvent des valeurs numériques permettant des réaliser des opérations d’agrégation : addition, minimum, maximum, moyenne...etc.

Exemple :

Dans le schéma ci-dessous, nous avons une table de faits Ventes qui a un grain qui nous donne un nombre d'unités vendues par date, par magasin et par produit.

Toutes les autres tables telles que DATE, Magasin et Produit sont des tables de dimensions. Ce schéma est connu sous le nom de schéma en étoile.

La figure [1.1] illustre les faits et les dimensions.

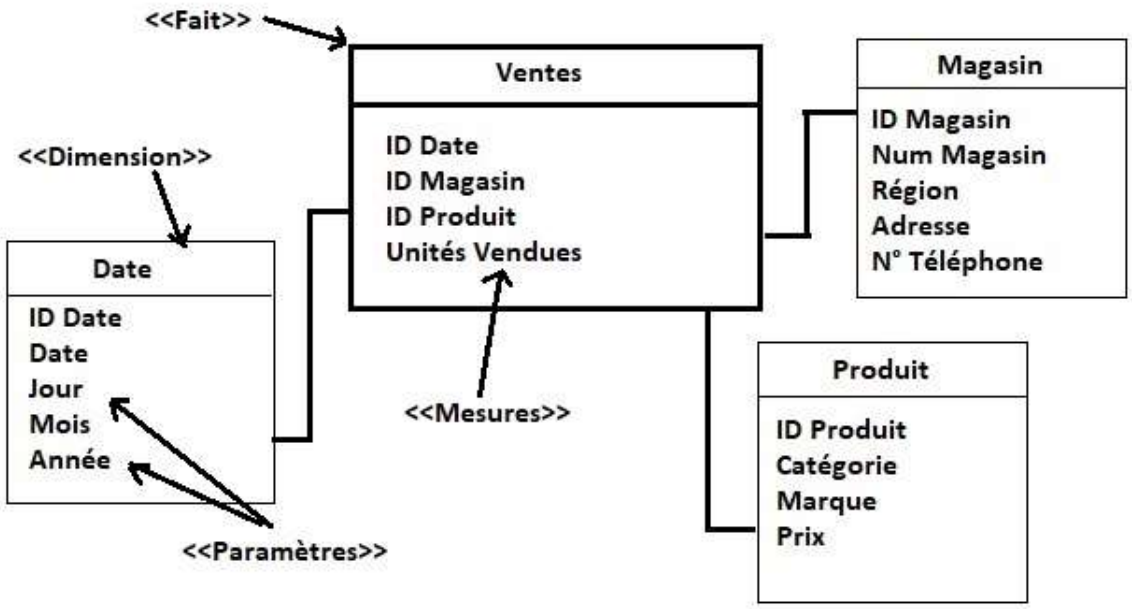


Figure [1.1] – Concept de Fait et Dimension

1.6.2 – Concept de dimension :

Une dimension est un axe d'analyse d'un fait, et est un ensemble d'informations de référence sur un événement mesurable.

Une table de dimension contient des attributs, appelés paramètres, qui représentent différents niveaux de détail auxquels les mesures d'un fait sont visualisées.

1.7 – Modélisation multidimensionnelle :

Un modèle multidimensionnel visualise les données sous la forme d'un cube de données. Un cube de données permet de modéliser et d'afficher des données dans plusieurs dimensions. Il est défini par des dimensions et des faits [19].

Les dimensions sont les perspectives ou les entités concernant lesquelles une organisation conserve des enregistrements.

Un modèle de données multidimensionnel est organisé autour d'un thème central. Ce thème est représenté par une table de faits. Les faits sont des mesures numériques les tables des faits contient les attributs clés étrangères qui la relient vers chacune des tables de dimension associée à un axe d'analyse. Chaque table de dimension est dotée d'une clé primaire permettant de réaliser des jointures avec la table des faits. Ces relations sont organisées selon trois types de schéma : schéma en étoile, schéma en flocon et schéma en constellation.

- **Schéma en étoile :** Un schéma en étoile est le schéma fondamental parmi les schémas du data mart et c'est le plus simple.

  Un schéma en étoile utilise une seule grande table de faits pour stocker des données transactionnelles ou mesurées, et une ou plusieurs tables dimensionnelles qui stockent des attributs sur les données.

  La figure [1.2] illustre un exemple d'un schéma en étoile.

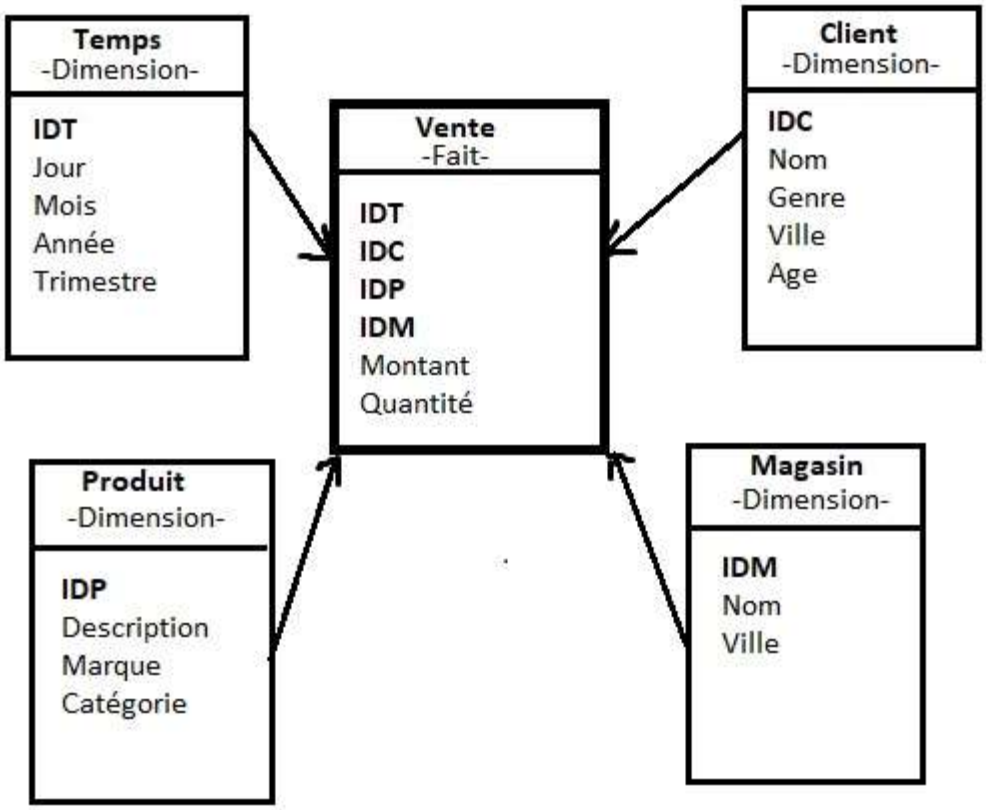


Figure [1.2] – Schéma en étoile.

- **Schéma en flocon :** Un schéma en flocon est une extension d'un schéma en étoile et ajoute des dimensions supplémentaires. Les tables de dimension sont normalisées, ce qui divise les données en tables supplémentaires.

  La figure [1.3] illustre un exemple d'un schéma en flocon.

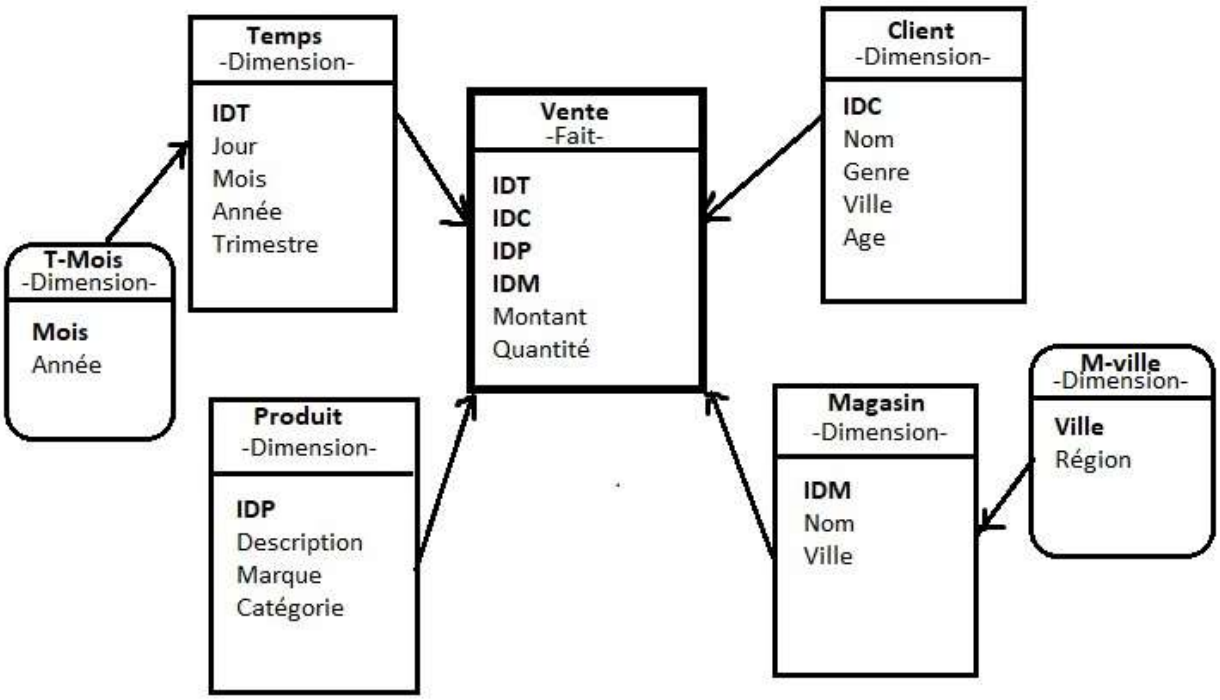


Figure [1.3] – Schéma en flocon.

- **Schéma en constellation :** Un schéma en constellation représente plusieurs relations de faits qui partagent des dimensions communes. Ces différentes relations de faits composent une famille qui partage les dimensions mais où chaque relation de faits a ses propres dimensions.

  La figure [1.4] illustre un exemple d'un schéma en constellation.

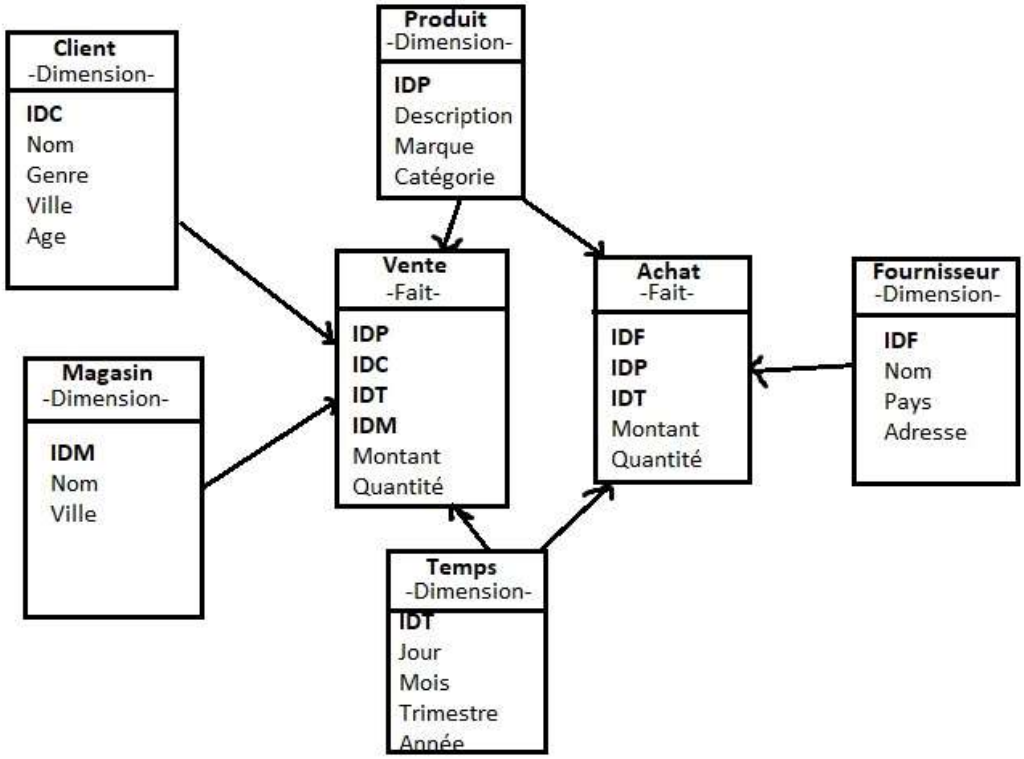


Figure [1.4] – Schéma en constellation.

1.8 – Implémentation du modèle multidimensionnel :

OLAP est une technologie qui permet aux analystes d'extraire et d'afficher des données commerciales de différents points de vue [2][13].

Il existe trois types de systèmes OLAP largement utilisés : MOLAP, ROLAP et HOLAP.

1.8.1 – MOALP : MOLAP utilise des moteurs de stockage multidimensionnels basés sur des baies pour des vues multidimensionnelles des données. Avec les magasins de données multidimensionnels, l'utilisation du stockage peut être faible si l'ensemble de données est clairsemé. Par conséquent, de nombreux serveurs MOLAP utilisent deux niveaux de représentation de stockage de données pour gérer des ensembles de données denses et éparses.

1.8.2 – ROLAP : Les serveurs ROLAP sont placés entre le serveur principal relationnel et les outils frontaux client. Pour stocker et gérer les données d'entrepôt, ROLAP utilise un SGBD relationnel ou relationnel étendu.

ROLAP comprend les éléments suivants :

- Implémentation de la logique de navigation d'agrégation.
- Optimisation pour chaque back-end SGBD.
- Outils et services supplémentaires.

1.8.3 – HOLAP : HOLAP est une combinaison de ROLAP et MOLAP. Il offre une plus grande évolutivité de ROLAP et un calcul plus rapide de MOLAP. Les serveurs HOLAP permettent de stocker les gros volumes de données d'informations détaillées. Les agrégations sont stockées séparément dans le magasin MOLAP.

Chapitre 2

Solution BIG DATA pour le traitement de données

2.1 – Introduction :

Depuis plusieurs années jusqu'à maintenant le volume de données numériques ne cesse d'augmenter due à la numérisation grandissante de tous les domaines. Cette multiplication des données a rendu difficile leur gestion et leur analyse pour aider les entreprises à améliorer les services et les processus pour les clients et l'identification précoce des risques à affronter. Pour cette raison, les analystes avaient besoin d'une solution qui réduise les coûts de gestion et de traitement de cette énorme quantité de données et réduit le temps nécessaire pour prendre les décisions nécessaires. C'est là que le terme BIG DATA est apparu [3][14].

Dans notre étude, nous utiliserons l'un des outils de BIG DATA qui est Hadoop, qui nous permet de stocker et de traiter ces données dans plusieurs appareils simultanés à grande vitesse.

2.2 – Un peu d'histoire :

L'application du BIG DATA et la quête de compréhension et d'analyse des données disponibles existent depuis longtemps.

Cependant, le terme Big Data a été inventé en 2005 lorsque l'analyste de l'industrie Doug Laney a articulé la définition désormais commune du BIG DATA comme les trois V : Volume, Vitesse et Variété [11].

- **Volume :** Le stockage de grandes quantités de données était un gros problème dans les organisations commerciales, mais des technologies comme Hadoop ont considérablement allégé le fardeau.
- **Vitesse :** Les conclusions, la généralisation et l'actualisation des entreprises dépendent également de la vitesse des données, il est donc important pour elles de ne pas sauter le pas, car toutes les plates-formes ne reçoivent pas les données entrantes à la même vitesse.
- **Variété :** Les données que nous recueillons sont présentées sous des formes non traditionnelles, telles que vidéo, texte, PDF et graphiques sur les réseaux sociaux. Bien que ces données nous soient extrêmement utiles, elles créent plus de travail et nécessitent plus de compétences analytiques pour déchiffrer ces données entrantes, les rendre gérables et leur permettre de fonctionner.

L'année 2005 a également vu la création de Hadoop - le framework open source qui a été spécialement créé pour stocker et analyser des ensembles de BIG DATA.

2.3 – Définition du terme BIG DATA :

Le BIG DATA implique les ensembles vastes, diverse et complexes d'informations qui croissent à un rythme exponentiel, leur stockage, leur récupération et enfin leur analyse.

Malheureusement, le BIG DATA est très volumineux et aucun des outils de gestion de données traditionnels ne peut le stocker ou le traiter efficacement [15].

2.4 – Les types de BIG DATA :

Les données se présentent sous des formes diverses et différentes, voici les 3 principales : [11][15].

1. **Données structurées :** Ce sont les données qui se présentent dans un format similaire et peuvent être stockées, consultées et traitées, il s'agit généralement de données tabulaires représentées par des colonnes et des lignes dans une base de données.
2. **Données non structurées :** Ce sont les données qui ne sont pas organisées de manière prédéfinie ou qui n'ont pas de modèle de données prédéfini, et leur traitement et leur analyse sont une tâche difficile.
3. **Données semi-structurées :** Ce sont les données qui contiennent à la fois des données structurées et non structurées. Nous pouvons voir les données semi-structurées sous forme structurée, mais elles ne sont en réalité pas définies avec la définition de table dans le SGBD relationnel.

2.5 – Les outils BIG DATA :

Comme nous l'avons dit dans la définition, analyser et traiter le BIG DATA n'est pas une tâche facile. C'est pourquoi nous avons besoin d'un ensemble d'excellents outils BIG DATA pour faire le travail.

Celui que nous avons choisi pour faire notre étude est Hadoop.

Il existe de nombreux outils disponibles tels qu'Apache Storm, Apache Spark, Flink et MangoDB et la liste continue.

2.6 - MapReduce :

2.6.1 - Définition :

MapReduce est le cœur d'Apache Hadoop, c'est un paradigme de programmation qui permet une évolutivité massive sur des centaines ou des milliers de serveurs dans un cluster Hadoop, il exécute diverses tâches en parallèle et fournit une redondance et une tolérance aux pannes [3][4].

Il s'inspire des fonctions Map et Reduce couramment utilisées en programmation fonctionnelle [8][16].

2.6.2 – MapReduce au travail :

- Le processus Master coordonne tous les travaux exécutés sur le système en planifiant les tâches.
- Les processus Worker exécutent des tâches et envoient des rapports d'avancement au processus Master.
- Si une tâche échoue, le processus Master peut la replanifier sur un autre Worker.

2.6.3 – Étapes MAP et Reduce :

- Etape Map : Le travail de MAP ou du Mapper consiste à traiter les données d'entrée. Généralement, les données d'entrée se présentent sous la forme d'un fichier ou d'un répertoire et sont stockées dans le système de fichiers Hadoop (HDFS). Le fichier d'entrée est passé à la fonction de mappeur ligne par ligne. Mapper traite les données et crée plusieurs petits morceaux de données.
- Etape Reduce : Cette étape est la combinaison de l'étape Shuffle et de l'étape Reduce. Le travail du Reducer est de traiter les données qui proviennent du Mapper. Après le traitement, il produit un nouvel ensemble de sorties, qui sera stocké dans le HDFS [1].

2.6.4 – Implémentation de MapReduce :

- Normalement, un Worker gère soit les tâches Map (un Worker Map) soit les tâches Reduce (un Worker Reduce), mais pas les deux [6].
- Le Maitre (Master) :

- Crée un certain nombre de tâches de mappage et un certain nombre de tâches de réduction, telles que sélectionnées par le programme utilisateur.
- Attribue des tâches aux travailleurs (Workers).
- Assure le suivi de l'état de chaque tâche de mappage et de réduction (inactif, en cours d'exécution sur un travailleur particulier, terminé).

❖ La tâche Map crée un fichier pour chaque tâche Reduce sur le disque local du Worker qui exécute la tâche Map et informe le Master de l'emplacement et de la taille de chacun de ces fichiers.

❖ Lorsqu'une tâche de réduction est affectée par le maître à un ouvrier, cette tâche reçoit tous les fichiers qui constituent son entrée.

❖ La tâche de réduction écrit sa sortie dans un fichier du système de fichiers distribué.

2.7 – L'outil Hadoop :

2.7.1 - Définition :

Hadoop est un logiciel open source écrit en langage Java, il est conçu pour préformer le traitement et l'analyse de gros volumes de données [1][8][16].

Il est utilisé par des géants du web comme Twitter, LinkedIn et Amazon.

Hadoop à des avantages, tels que :

1- Toutes les données sont accessibles, il n'est donc pas nécessaire d'archiver / nettoyer les données avant le stockage car Hadoop permet d'augmenter l'espace de stockage sans limites.

2- Est une architecture évolutive qui permet l'ajout de nouveaux clusters / serveurs à l'architecture d'origine sans limites.

3- Est une architecture robuste et facile à utiliser qui peut être facilement configurée en modifiant le fichier de configuration.

2.7.2 - Composants de Hadoop :

Hadoop basé sur le principe de grilles de calcul, le framework se compose principalement des modules suivants : [1][8][16].

1.**Hadoop Distributed File System (HDFS) :** un système de fichiers distribués qui permet de stocker de très gros volumes de données sur un grand nombre de machines équipées de matériel de base.

2.Hadoop Common : il contient les bibliothèques et les utilitaires nécessaires aux autres modules Hadoop.

3.Hadoop YARN (Yet Another Resource Negociator) : cette plateforme permet de gérer les ressources informatiques du cluster et les utilise pour la planification des applications des utilisateurs.

4.Hadoop MapReduce : il permet un traitement de la donnée à grande échelle et ainsi créer facilement des applications qui traitent de grandes quantités de données en parallèle sur de gros clusters de manière fiable et tolérante aux pannes.

2.7.3 - Les clés de fonctionnement de Hadoop :

Par défaut, les données chargées dans HDFS (système de fichiers d'Hadoop) sont stockées en trois exemplaires, sur des nœuds différents. Cette réplication répond à deux objectifs : [1][2].

1. Disponibilité des données en cas de panne.
2. Profiter de la localité des données lors de l'exécution d'une tâche d'un job MapReduce.

En effet, le principe de fonctionnement de Hadoop est assez simple, il consiste à répartir l'exécution d'un traitement sur plusieurs nœuds.

2.7.4 - Architecture de Hadoop :

La figure [2.1] montre l'architecture de Hadoop.

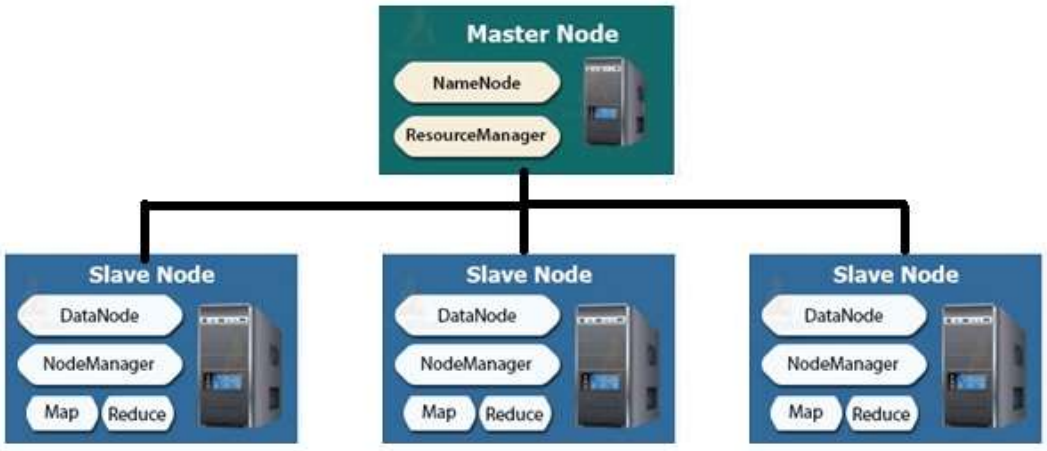


Figure [2.1] – Architecture de Hadoop.

2.7.5 - Hadoop File System (HDFS) :

HDFS est un système de fichiers distribué qui gère de grands ensembles de données fonctionnant sur du matériel standard. Il est utilisé pour mettre à l'échelle un seul cluster Apache Hadoop à des centaines (voire des milliers) de nœuds [17].

HDFS n'est pas fait pour les utilisateurs ci-dessous :

1- Accès aux données avec faible latence :

- HDFS optimisé pour traiter un grand volume de données.

2- Grand nombre de petits fichiers :

- Métadonnées des fichiers chargés en mémoire centrale.

(Jusqu'à plusieurs dizaines de millions de fichiers par nœud)

3- Ecritures arbitraires, écrivains multiples :

- HDFS conçu pour écrivains uniques, et écriture en fin de fichiers.

Il a pour objectifs :

. Passage à l'échelle (gestion plusieurs milliers de nœuds)

. Tolérance aux pannes (hardware et software).

. Gestion de fichiers de grande taille.

. Accès donnés : Write once/Read Many times.

2.7.6 - Placement des blocs (HDFS) :

- Placement nœuds différents.

- Réplication de chaque bloc sur 3 nœuds différents.

La figure [2.2] montre un exemple de 5 nœuds avec un facteur de réplication dfs égal à 3 [17].

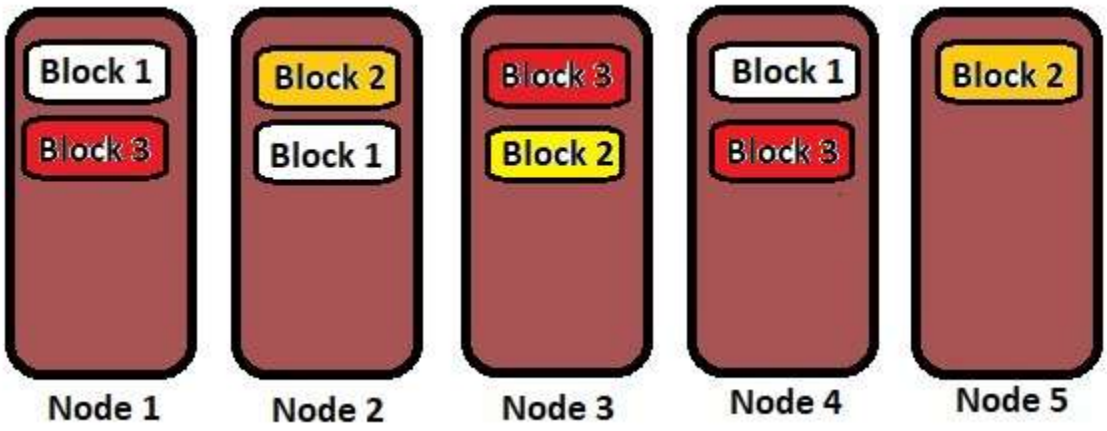


Figure [2.2] – Placement des blocs.

Dans ce cas, même si un nœud est tombé en panne, le travail continue toujours.

2.7.7 - Comment Hadoop distribue les données :

Lors de l'écriture de données dans un fichier HDFS, les données sont d'abord écrites dans le cache local du client. Lorsque le cache atteint un certain seuil (taille de bloc, 128 Mo par défaut), le client demande et récupère une liste de DataNodes à partir du NameNode (qui conserve les métadonnées). Cette liste contient les DataNodes qui ont de l'espace et peuvent avoir une réplique de ce bloc. Le nombre de DataNodes pouvant contenir les données de réplica est basé sur le facteur de réplication. Le client crée ensuite un pipeline entre les DataNodes pour vider les données. Le premier DataNode

commence à recevoir les données (le sous-jacent io.file.buffer.size est de 4 Ko , Hadoop utilise pour les opérations d'E/S), écrit les données mises en mémoire tampon dans le répertoire local du nœud et transfère les mêmes données mises en mémoire tampon vers le deuxième DataNode de la liste . Le deuxième DataNode, à son tour, commence à recevoir les données mises en mémoire tampon du bloc de données, écrit dans son répertoire local, puis vide les mêmes données vers le troisième DataNode. Enfin, le troisième DataNode écrit les données dans son répertoire local.

Lorsque le premier bloc est rempli, le client demande que de nouveaux DataNodes soient choisis parmi NameNode pour héberger les répliques du bloc suivant. Ce flux se poursuit jusqu'au dernier bloc du fichier. Le choix des DataNodes pour chaque bloc peut être différent [8][17].

2.7.8 - Types des nœuds (HDFS) :

Il y a 2 catégories de nœuds dans HDFS :

1. **Namenode** (maître) / Backup Node.
2. **Datanodes** (esclaves).

Namenode : Gère système de fichiers dont il a la charge

– 1 seul Namenode par cluster.

– Maintient arborescence et meta-données sur Système de Fichiers :

. 2 types de fichiers : Namespace et Logs.

– Stocke liste des datanodes sous son contrôle.

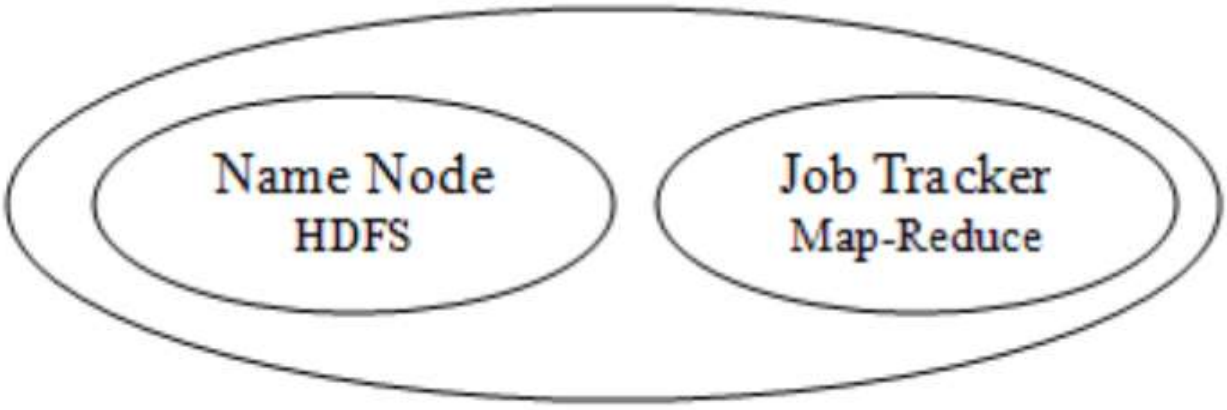


Figure [2.3] – Master Node.

Backup Node : Sauvegarde de l'état du Namenode.

Datanodes : Stockage des fichiers et exécution des tâches

– Une instance sur chaque nœud du cluster (1 à 4000 nœuds)

– Traitement des requêtes des clients

– Informent périodiquement le namenode :

– Blocs stockés.

– Avancement des tâches.

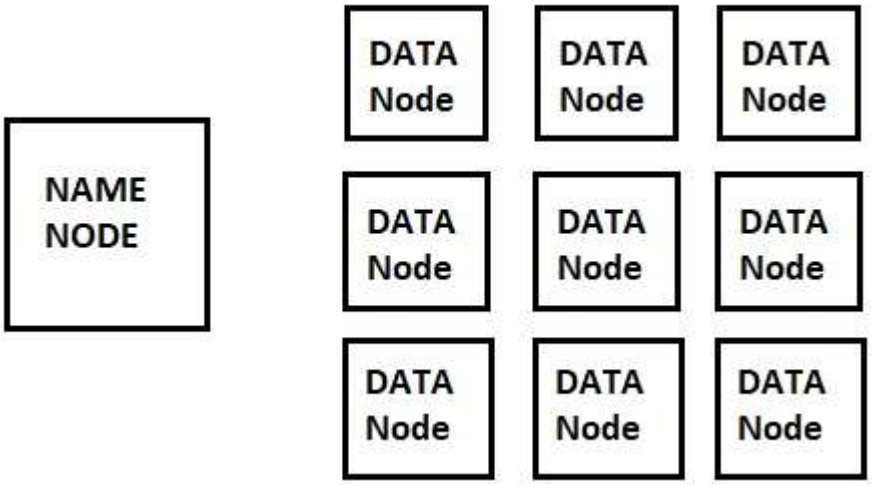


Figure [2.4] – Exemple d'un maitre et des esclaves.

2.7.9- Comment Hadoop lit un fichier :

La figure [2.5] montre la lecture d'un fichier par Hadoop.

[17]

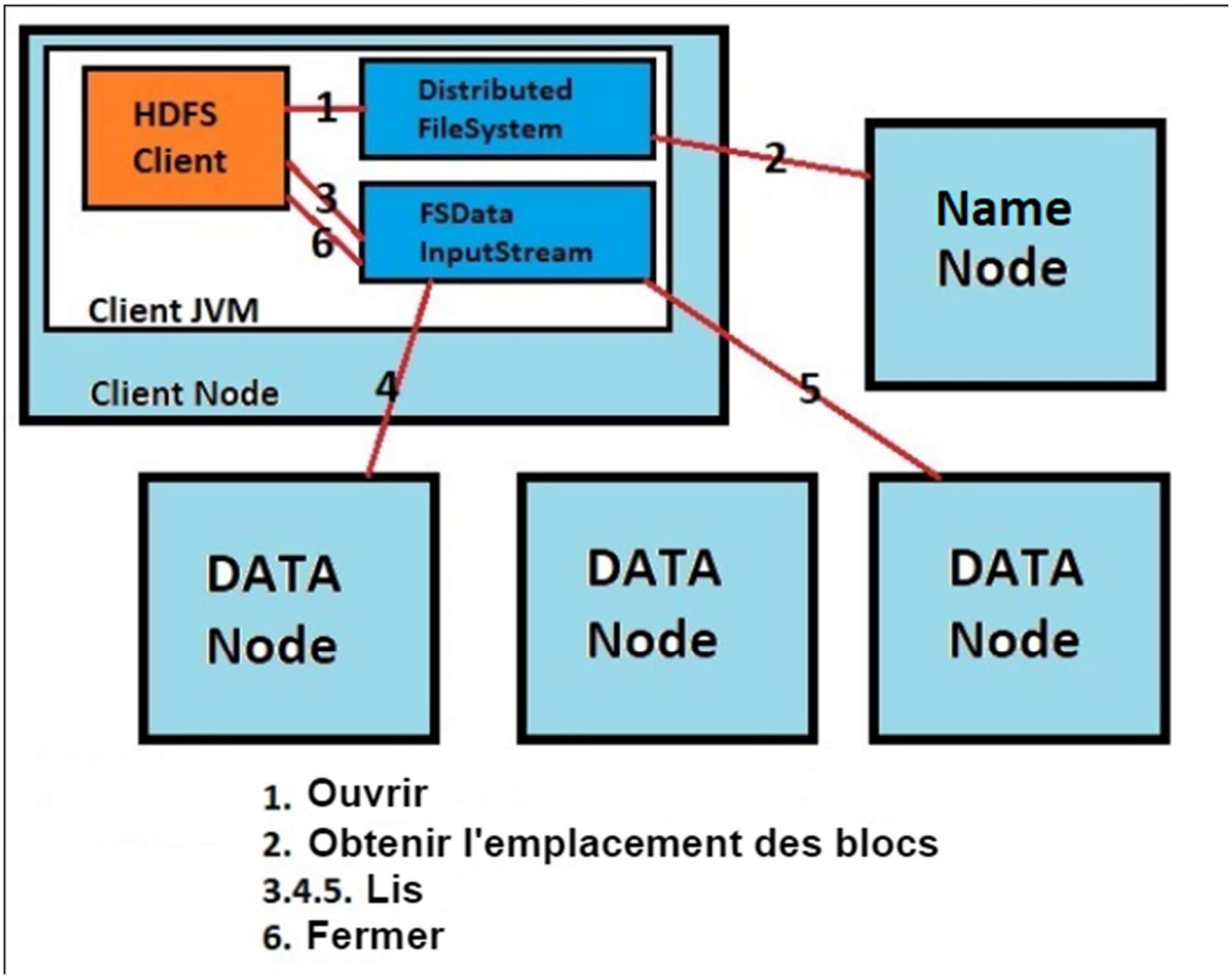


Figure [2.5] – Lecture d'un fichier par Hadoop.

2.7.10 - Comment Hadoop écrit dans un fichier :

La figure [2.6] montre l'écriture dans un fichier par Hadoop.

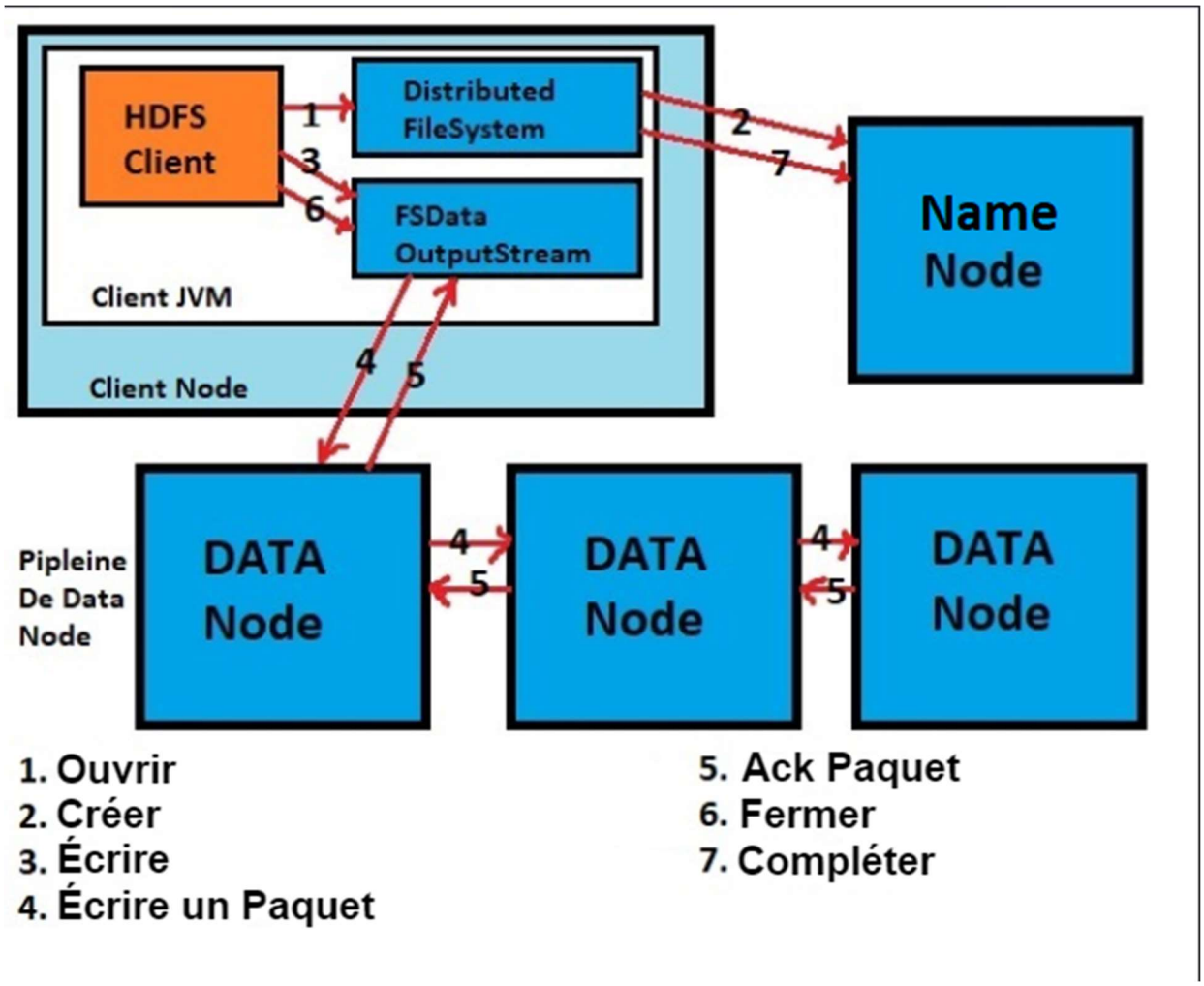


Figure [2.6] – Ecriture dans un fichier par Hadoop.

2.8 – L'outil Hive :

2.8.1 - Définition :

Hive est construit sur Hadoop, il s'agit d'un système d'entrepôt de données distribué et tolérant aux pannes qui permet des analyses à grande échelle.

Hive permet aux développeurs SQL d'écrire des instructions Hive Query Language (HQL) similaires aux instructions SQL standard pour la requête et l'analyse de données.

Il a été initialement développé par Facebook, plus tard, Apache Software Foundation l'a repris et l'a développé en tant qu'open source sous le nom d'Apache Hive [5][18].

2.8.2 - Architecture :

L'architecture Hive se compose principalement de trois parties principales : [5][18].

1. Hive Clients.
2. Hive Services.
3. Hive Stockage et Calcul.

1. **Hive Clients :** Hive fournit différents pilotes pour la communication avec différents types d'applications. Pour les applications basées sur les économies, il fournira une communication client Thrift. Pour les applications liées à Java, il fournit des pilotes JDBC. Autre que tout type d'applications fournies pilotes ODBC. Ces clients et pilotes communiquent à leur tour avec le serveur Hive dans les services Hive
2. **Hive Services :** Les interactions des clients avec Hive peuvent être effectuées via les services Hive. Si le client souhaite effectuer des opérations liées aux requêtes dans Hive, il doit communiquer via Hive Services.CLI est l'interface de ligne de commande qui agit comme Hive Service pour les opérations DDL.
3. **Hive Stockage et Calcul** : Les services Hive tels que Meta store, File system et Job Client communiquent à leur tour avec le stockage Hive et effectuent les actions suivantes :

    - Les informations de métadonnées des tables créées dans Hive sont stockées dans la « base de données de stockage des métadonnées » de Hive.
    - Les résultats de la requête et les données chargées dans les tables vont être stockées dans le cluster Hadoop sur HDFS.

2.8.3 - Comment fonctionne Hive :

L'interface de HIVE telle que la ligne de commande ou l'interface utilisateur Web transmet la requête au pilote à exécuter, puis le pilote conçoit un descripteur de session pour la requête et transfère la requête au compilateur pour établir un plan d'exécution.

Après cela, le pilote enverra le plan d'exécution au moteur d'exécution pour l'exécuter.

Une fois le travail d'exécution terminé, le résultat est récupéré des DataNodes vers le pilote vers l'interface utilisateur [17][18].

Chapitre 3

Installation et Configuration

3.1 - Introduction :

Ce chapitre a pour objectif de présenter les phases de mise en œuvre et de validation de notre conception.

Nous avons constaté que le meilleur système d'exploitation pour installer Hadoop et Hive est « CentOS », mais nous avons déjà commencé avec Ubuntu et Windows.

Nous avons rencontré des problèmes lors de la phase d'installation en raison de nos limites matérielles. Nous avons résolu ces problèmes en modifiant certains des fichiers de configuration pendant la phase d'installation.

Outre l'exemple explicatif "WordCount" que nous allons aborder dans la section suivante, nous avons téléchargé une base de données benchmark pour tester certaines requêtes et déterminer la puissance de Hive/Hadoop lorsqu'il s'agit de traiter et d'analyser des données volumineuses.

3.2 - Installation de Hadoop dans Windows et Linux :

Tout en travaillant sur notre PFE, nous avons dû installer Apache Hadoop sur Windows 10 et Linux UBUNTU 20.04. De nombreux guides ont été trouvés en ligne, mais malheureusement, ils n'ont pas fonctionné pour nous. Et nous avons presque pris chaque configuration d'une source différente.

3.2.1 - Préparation :

Après avoir essayé de nombreuses versions de HADOOP, certaines d'entre elles ne sont pas compatibles avec Java et ont causé des problèmes et n'ont pas pu terminer le travail, nous avons donc fait le choix d'utiliser la version 3.2.1.

Ces logiciels doivent être préparés pour installer Hadoop 3.2.1.

1. Hadoop 3.2.1.
2. Java JDK 8.

3.2.2 - Installation :

1. Pour vérifier que Java 1.8.0 est déjà installé, nous tapons la commande "Javac -version".
2. Nous devons définir la variable d'environnement HADOOP_HOME.
3. Nous devons définir le chemin de la variable d'environnement JAVA_HOME.

3.2.3 - Configuration :

1. Modification du fichier **« Hadoop-3.2.1/etc/hadoop/core-site.xml »** pour spécifier le système de fichiers par défaut et le port que les nœuds de données utiliseront pour contacter le nœud maître [16].
2. Édition du fichier **« Hadoop-3.2.1/etc/hadoop/mapred-site.xml »** pour spécifier la réplication de bloc par défaut et la vérification des autorisations sur HDFS.
3. Création du dossier « data » sous « Hadoop-3.2.1 » et éditez le fichier **« Hadoop-3.2.1/etc/hadoop/hdfs-site.xml »,** pour spécifier le chemin des dossiers hadoop DataNode et NameNode.
   - Création des dossiers « datanode » et « namenode » sous « Hadoop-3.2.1\data ».
4. Modification du fichier **« Hadoop-3.2.1/etc/hadoop/yarn site.xml »**, pour spécifier où stocker les données et les fichiers log de Node Manager et Resource Manager.
5. Modification du fichier Hadoop-3.2.1/etc/hadoop/hadoop-env.cmd en fermant la ligne de commande **« JAVA_HOME =% JAVA_HOME% »** au lieu de définir **«JAVA_HOME = C:\Java »**, cette étape est uniquement dans Windows car Linux nous permet de définir JAVA_HOME à partir du fichier bashrc.
6. Téléchargement du fichier de configuration Hadoop qui contient le dossier bin, cette étape est uniquement lorsque vous téléchargez la dernière version de Hadoop, nous téléchargeons donc le dossier bin d'une version précédente car il est plus stable.
7. Ouvrir cmd ou le terminal linux et tapez la commande « hdfs namenode –format ».

3.2.4 – Configuration de HADOOP dans un cluster à plusieurs nœuds :

Nous avons au moins deux machines (Master et Slave) avec des adresses IP [16].

Étape 1 : Vérification de l'adresse IP de toutes les machines.

Étape 2 : Ouvrir le fichier "Workers" dans " hadoop\etc\hadoop" et ajoutez-y :

– hadoop-slave1

– hadoop-slave2

La figure [3.1] montre comment nous avons ajouté ces 2 esclaves.

Figure [3.1] – Le fichier Workers.

Si nous avions plus de machines, nous les ajouterions simplement à la liste.

Étape 3 : Nous avons ouvert le fichier hosts pour ajouter les adresses IP de nos machines.

La figure [3.2] montre comment nous avons ajouté les adresses IP de notre machine.

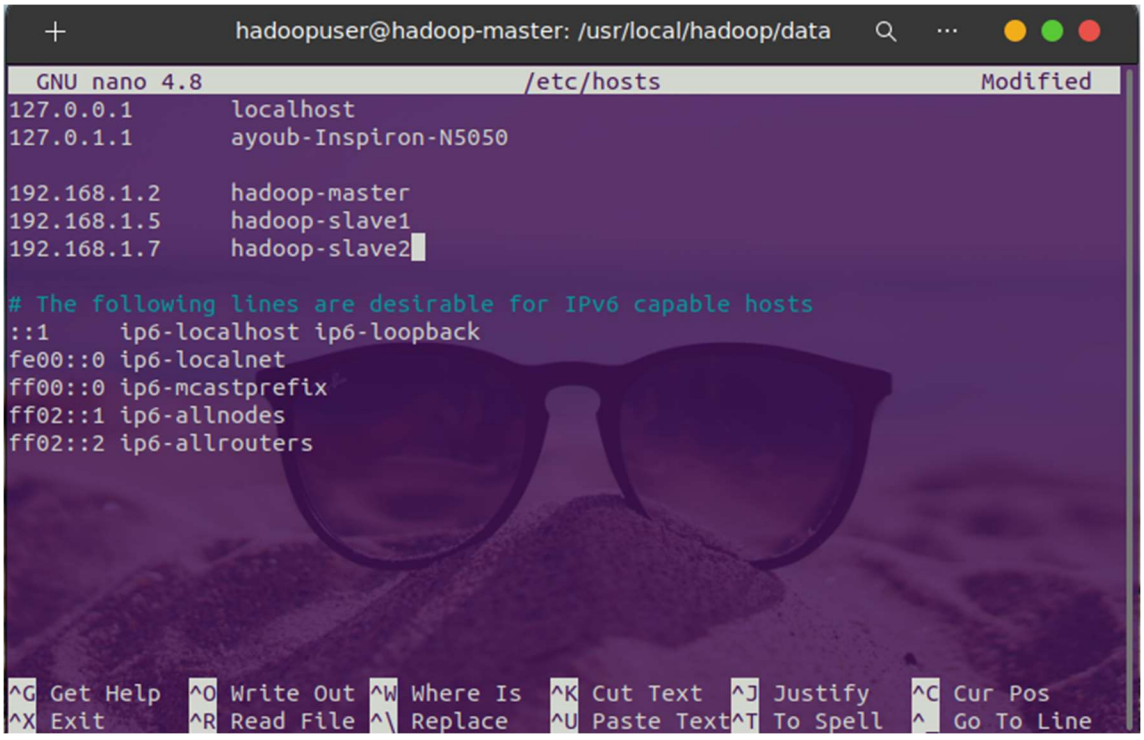


Figure [3.1] – Le fichier Hosts.

Étape 4 : Nous avons généré une clé SSH dans le nœud maître.

Étape 5 : Après avoir généré la clé SSH, nous l'avons transmise aux esclaves et établi la connexion.

La figure suivante [3.3] montre une capture d'écran de l'interface utilisateur Web de Hadoop et contient les deux nœuds esclaves que nous utilisons avec 1,5 Go de RAM allouée dont nous allons discuter de la façon de l'allouer dans la section suivante.

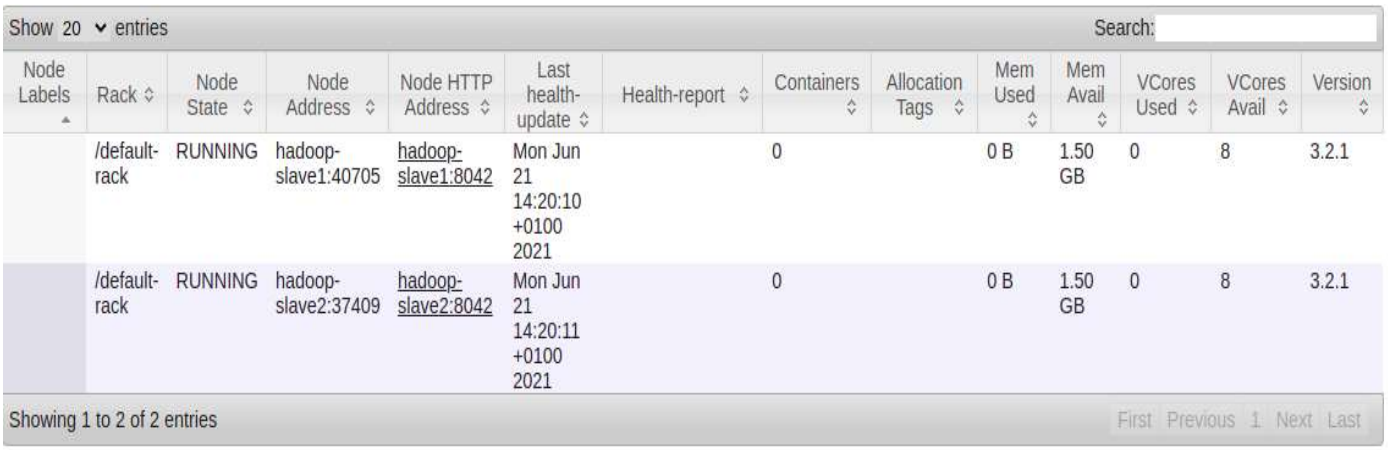


Figure [3.3] – L'interface web de Hadoop.

3.2.5 - L'allocation de Mémoire :

Nous avons rencontré beaucoup de problèmes en ce qui concerne l'allocation de mémoire car cela peut être difficile sur les nœuds à faible RAM car les valeurs par défaut ne conviennent pas aux nœuds avec moins de 8 Go de RAM. Dans cette section, nous mettrons en évidence le fonctionnement de la mémoire allouée pour les tâches MapReduce sur nos PC et fournirons un exemple de configuration pour les nœuds de 4 Go de RAM.

Nous définissons la quantité de mémoire qu'un seul conteneur peut consommer et l'allocation de mémoire minimale autorisée, selon nos PC.

Un conteneur ne sera jamais plus grand que le maximum, sinon l'allocation échouera et sera toujours allouée comme un multiple de la quantité minimale de RAM.

Que l'on soit sous Linux ou Windows, on s'aperçoit que le système d'exploitation n'utilise jamais moins de 1Go de RAM et parce que nous comptions toujours sur Internet pour trouver des réponses aux erreurs qui nous rencontraient, la plupart du temps nous avions ouvert le navigateur internet et il jamais

consomme moins de 600 Mo, sans oublier les applications qui s'exécutent en arrière-plan nous laissant avec moins de 2 Go de RAM.

Nous avons donc alloué 1,5 Go pour le conteneur yarn, cette valeur est configurée dans yarn-site.xml avec yarn.scheduler.maximum-allocation-mb [20].

3.2.6 - L'essai :

1. Pour démarrer les daemons de Hadoop nous devons tapez la commande « start-all.cmd » dans Windows et la commande « start-all.sh ».

   La figure suivante [3.4] montre ce qui va se passer.

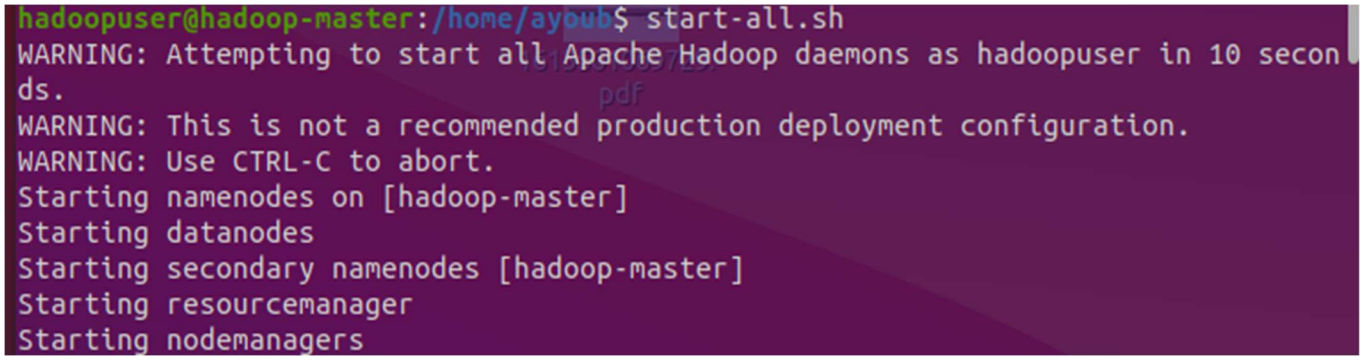


Figure [3.4] – Le démarrage des daemons de Hadoop.

2. Nous nous assurons que ces applications en cours d'exécution en exécutant la commande « jps » :

   La figure suivante [3.5] montre ce qui va se passer.

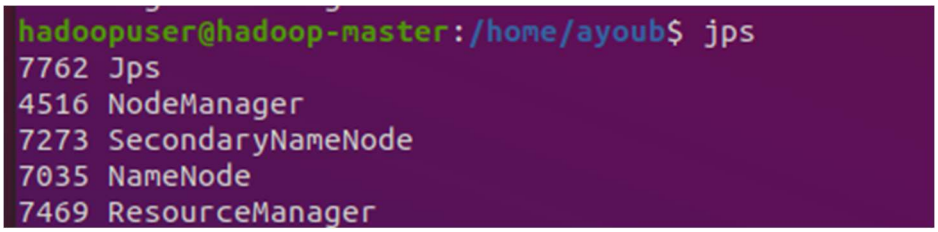


Figure [3.5] – La commande « jps ».

Dans les nœuds esclaves, nous ne devrions voir que Datanode car ce sont eux qui font le travail.

Les deux figures [3.6] et [3.7] montre le résultat de la commande « jps ».

Figure [3.6] – Commande « jps » dans esclave 1.

Figure [3.7] – Commande « jps » dans esclave 2.

3.3 - Installation de Hive :

Apache Hive nécessite une base de données relationnelle pour créer son Metastore (où toutes les métadonnées seront stockées). Hive utilise derby comme métastore par défaut. Au début, nous l'avons utilisé, mais avec les divers problèmes et erreurs auxquels nous avons été confrontés, nous avons décidé de passer à MySQL car il existait de nombreuses solutions en ligne et un peu de connaissances préalables à ce sujet.

3.3.1 - Installation :

Puisque Java 8 est installé, nous devrons installer la version Apache Hive 3.1.2.

Après avoir téléchargé et extraite l'archive de Hive, nous devons configurer les variables d'environnement.

Nous devons maintenant ajouter les variables utilisateurs suivants :

- HIVE_HOME: "\ hadoop-env \ apache-hive-3.1.2 \"
- HIVE_LIB: «% HIVE_HOME% \ lib»
- HIVE_BIN: "% HIVE_HOME% \ bin"
- HADOOP_USER_CLASSPATH_FIRST: "true"

Nous avons installé MySQL version 8.0.25.

La figure [3.8] montre une capture du lancement de MYSQL.

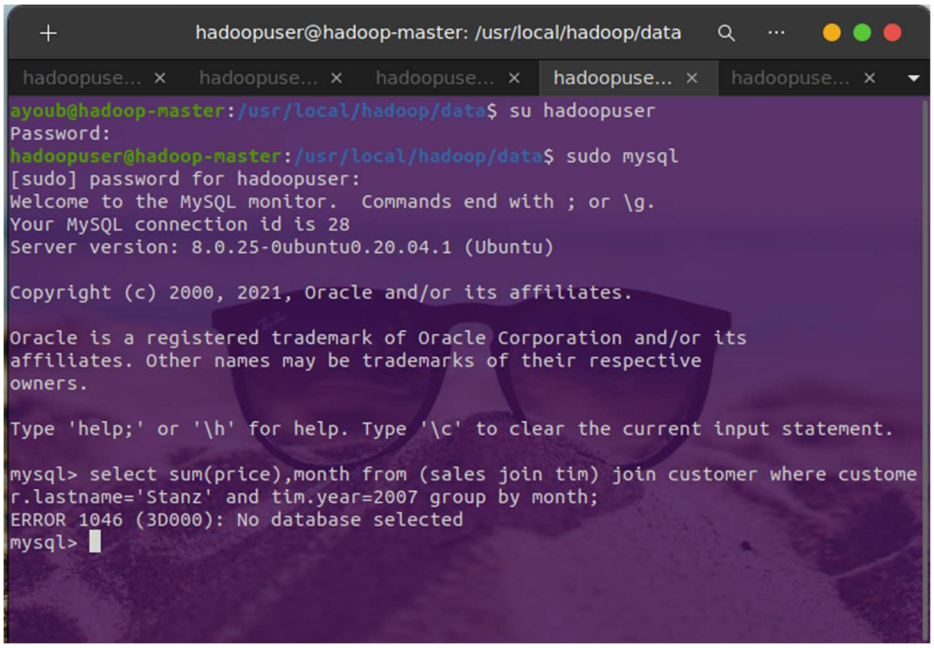


Figure [3.8] – Le Lancement de MYSQL.

3.3.2 - Configuration :

Avant de passer aux étapes suivantes, la première chose à faire est de télécharger le pilote jdbc exact pour connecter la version MySQL spécifiée avec Hive.

Après cela, nous devons définir le nom du pilote et le lien pour se connecter au serveur MySQL dans le fichier hive_site.xml.

Par défaut, Hive utilise une base de données nommée Hive_metastore prédéfinie dans les fichiers de configuration et comme nous n'utilisons pas le métastore par défaut Derby, nous devons créer cette base de données nous-mêmes dans MySQL et lui donner les autorisations nécessaires.

Avant de pouvoir exécuter Hive, nous devons lancer son métastore et nous connecter à son serveur qui est actuellement à la version 2 "Hiveserver2" (Hiveserver2 nous permet d'exécuter des requêtes sur Hive).

Cette figure [3.9] montre l'initiation du métastore.

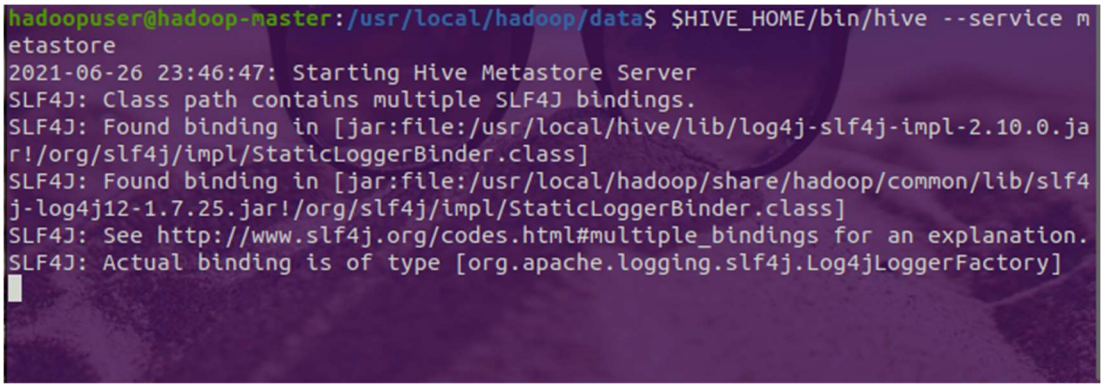


Figure [3.9] – L'initialisation du service metastore.

Et celui-ci [3.10] montre la connexion à Hiveserver2.

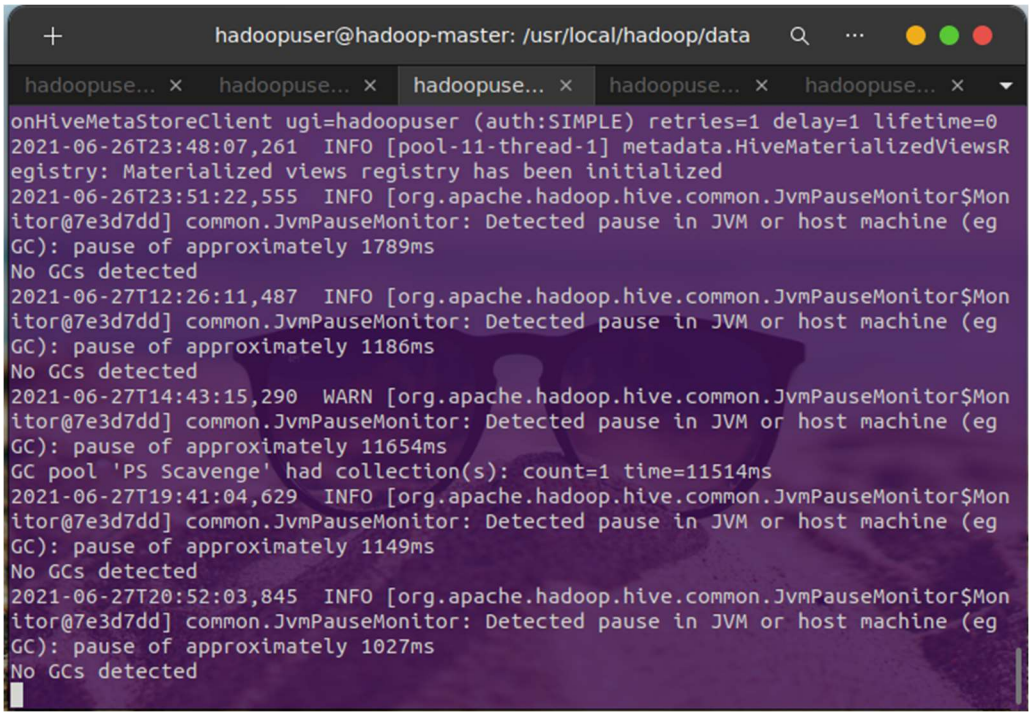


Figure [3.10] – La connexion à Hiveserver2.

Après tout cela, nous pouvons exécuter Hive simplement en exécutant la commande "hive".

La figure suivante [3.11] montre l'exécution de la commande hive.

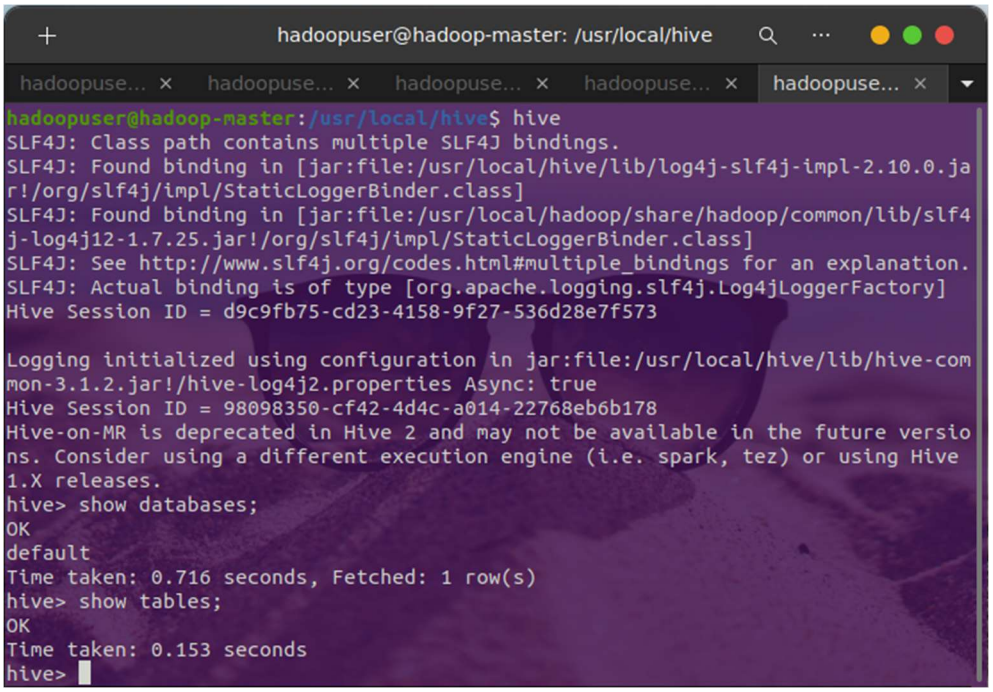


Figure [3.11] – L'exécution de la commande hive.

3.4 – L’exemple WordCount :

WordCount est une simple application de mappage/réduction qui traite le texte d'entrée et compte le nombre d'occurrences de chaque mot.

L'exemple WordCount lit les fichiers texte et compte la fréquence des mots. Chaque mappeur prend une ligne du fichier d'entrée en entrée et la divise en mots. Il émet ensuite une paire clé/valeur du mot (sous la forme de (mot, 1)) et chaque réducteur additionne les comptes pour chaque mot et émet une seule clé/valeur avec le mot et la somme [9].

3.4.1 – Les étapes pour exécuter l’exemple :

Puisque Hadoop est écrit en JAVA, un fichier JAR contenant les fonctions Map et Reduce doit être créé.

Nous n'en avons pas créé car nous l'avons trouvé sur Internet.

Pour démarrer ce test, nous devons fournir un fichier texte contenant des mots (dans le fichier texte que nous avons utilisé, nous avons mis des noms de personnes au hasard).

1. Hadoop doit être démarré ; nous l'avons fait comme précédemment en tapant la commande "start-all.sh".
2. Ensuite, nous créons un répertoire d'entrée dans HDFS en exécutant la commande « hadoop fs -mkdir /input_dir ».
3. Le fichier texte doit être copié dans le répertoire HDFS en tapant la commande « hadoop fs -put C:/input_file.txt /input_dir ».
4. Nous pouvons vérifier sa présence avec cette commande « hadoop fs -ls /input_dir/ ».
5. Exécutons maintenant MapReduceClient.jar « hadoop jar MapReduceClient.jar wordcount /input_dir /output_dir ».
6. Ensuite, nous pouvons voir le résultat du test « hadoop dfs -cat /output_dir/* ».

La figure suivante [3.12] montre le résultat de l'exécution de l'exemple de WordCount Hadoop dans notre cluster.

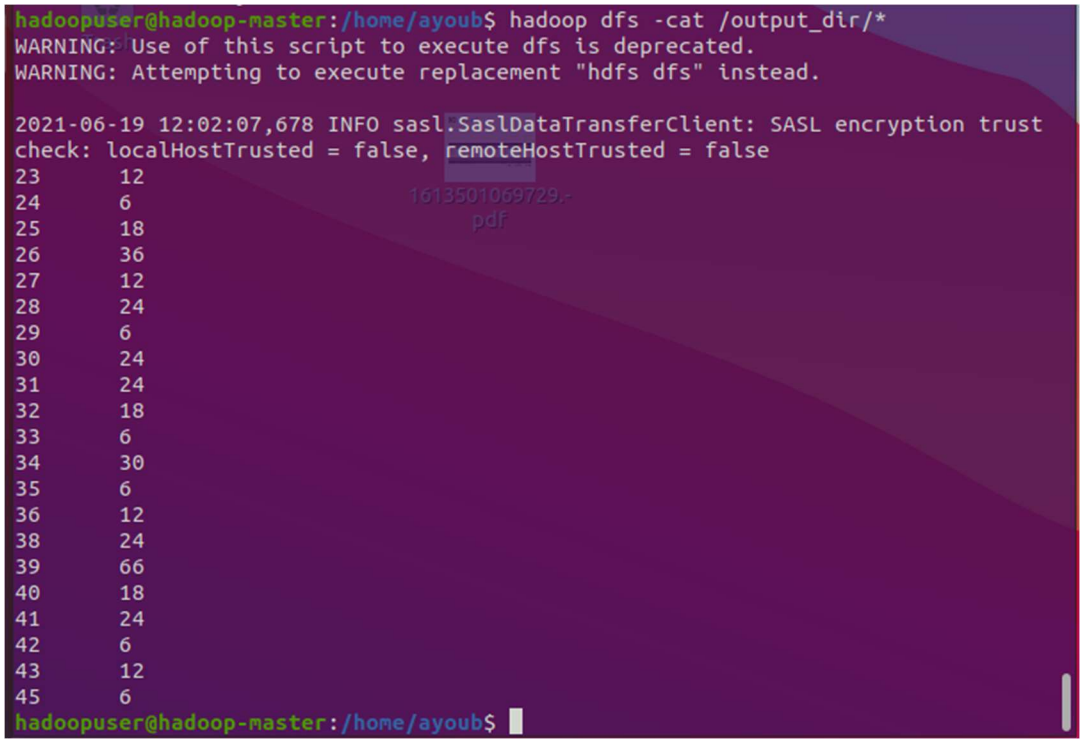


Figure [3.12] – L'exécution de l'exemple WordCount sur Hadoop.

Nous avons également fait ce test avec Hive avec les deux mêmes nœuds esclaves, la figure suivante [3.13] montre le test.

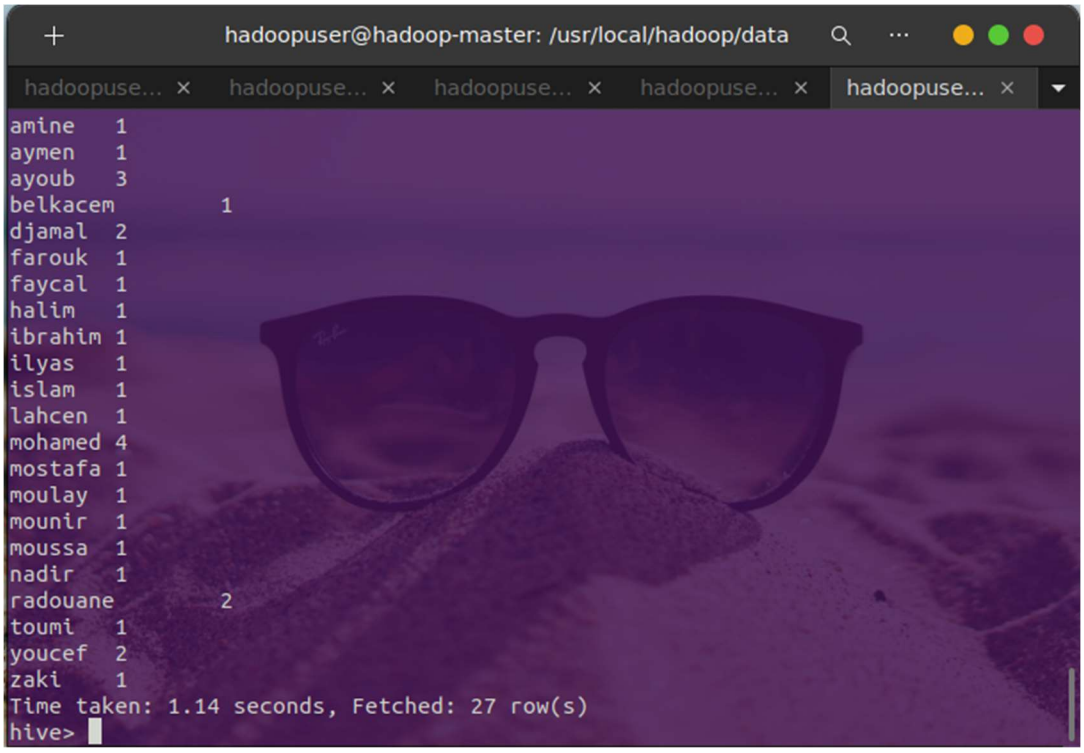


Figure [3.13] – L'exécution de la commande hive.

3.5 – Notre Conception :

Nous avons téléchargé une base de données d'enregistrement des ventes, les données incluent environ 500 000 transactions, y compris des informations sur le client, la région, le produit et le temps.

Voici quelques questions que nous avons posées pour tester Hive.

- Qui sont les principaux acheteurs dans chaque État et combien ont-ils dépensé ?
- Quelle est la dépense moyenne pour les personnes qui ont laissé un avis négatif ? Et les avis positifs ?
- De combien, en pourcentage, les ventes ont augmenté ou diminué au fil des mois en 2007 ?

- Quel est l'article le plus acheté dans une zone spécifiée ?

Pour répondre à ces questions, nous avons utilisé 3 PC, 2 esclaves (un avec un i3 4éme génération et l'autre i5 4éme génération, les 2 PC ont 4Go de ram), 1 maître avec un i5 2éme génération et 4 Go de RAM, les a connectés au réseau sans fil et fournis une adresse IP statique.

Nous avons utilisé la taille de bloc HDFS par défaut de 128 Mo et le facteur de réplication par défaut était défini sur 1 car nous étions dans la même chambre et nous n'avions pas besoin de réplication, si un PC tombait en panne, nous le remplacerions simplement.

3.5.1 – Schéma en étoile de la base de données utilisée :

La figure [3.14] illustre le schéma en étoile correspondant.

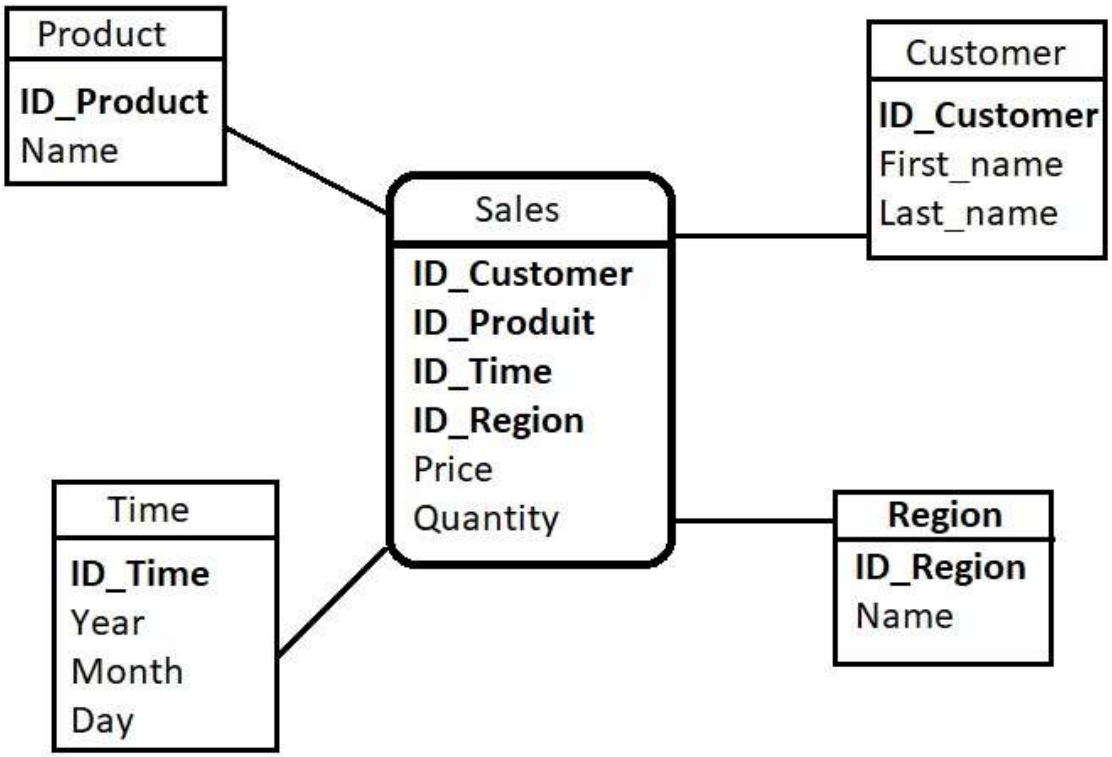


Figure [3.14] – Le schéma en étoile de la base de données utilisée.

3.5.2 – Création des tables de l'entrepôt :

La première étape pour la mise en œuvre de notre conception consiste en la création des tables qui constituent notre entrepôt. La figure suivante [3.15] montre le script de création de la table « customer ».

```
hive> CREATE TABLE customer (IDcustomer int, firstname string, lastname string)
row format delimited fields terminated by ',';
OK
Time taken: 0.776 seconds
hive>
```

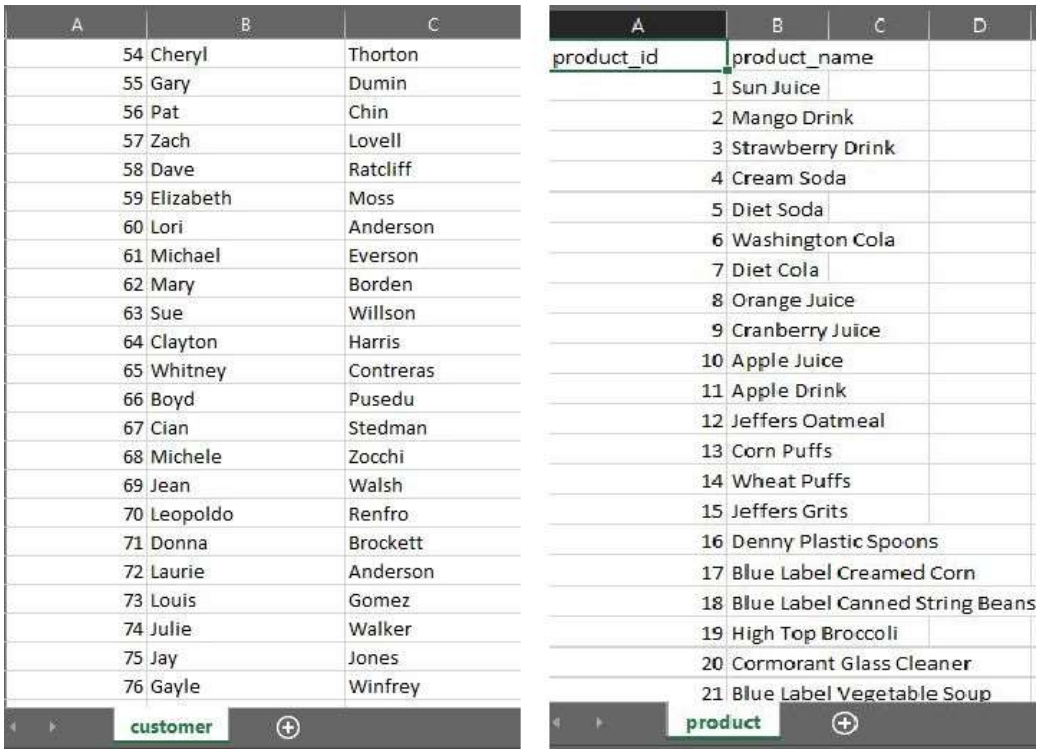
Figure [3.15] – La création de la table client.

Des scripts similaires sont utilisés pour créer le reste des tables.

3.5.3 – Données Sources :

Dans cette expérience et puisque nous parlons de BIG DATA, nous sommes censés le faire avec de très gros entrepôts de données pour ressentir la puissance de Hadoop mais avec toutes les erreurs qui nous ont été rencontrées et le temps qui nous reste, nous avons été obligés de faire ce test avec quelques petits fichiers.

Les figures suivantes [3.16], [3.17], [3.18] montrent une partie du contenu de notre base de données que nous utilisons au format csv.



	A	B	C
54		Cheryl	Thorton
55		Gary	Dumin
56		Pat	Chin
57		Zach	Lovell
58		Dave	Ratcliff
59		Elizabeth	Moss
60		Lori	Anderson
61		Michael	Everson
62		Mary	Borden
63		Sue	Willson
64		Clayton	Harris
65		Whitney	Contreras
66		Boyd	Pusedu
67		Cian	Stedman
68		Michele	Zocchi
69		Jean	Walsh
70		Leopoldo	Renfro
71		Donna	Brockett
72		Laurie	Anderson
73		Louis	Gomez
74		Julie	Walker
75		Jay	Jones
76		Gayle	Winfrey

customer

	A	B	C	D
product_id		product_name		
1		Sun Juice		
2		Mango Drink		
3		Strawberry Drink		
4		Cream Soda		
5		Diet Soda		
6		Washington Cola		
7		Diet Cola		
8		Orange Juice		
9		Cranberry Juice		
10		Apple Juice		
11		Apple Drink		
12		Jeffers Oatmeal		
13		Corn Puffs		
14		Wheat Puffs		
15		Jeffers Grits		
16		Denny Plastic Spoons		
17		Blue Label Creamed Corn		
18		Blue Label Canned String Beans		
19		High Top Broccoli		
20		Cormorant Glass Cleaner		
21		Blue Label Vegetable Soup		

product

Figure [3.16] – Les tables produit et client.

	A	B	C	D
Time_id		Year	Month	Day
367		2008	1	1
368		2008	1	2
369		2008	1	3
370		2008	1	4
371		2008	1	5
372		2008	1	6
373		2008	1	7
374		2008	1	8
375		2008	1	9
376		2008	1	10
377		2008	1	11
378		2008	1	12
379		2008	1	13
380		2008	1	14
381		2008	1	15
382		2008	1	16
383		2008	1	17
384		2008	1	18
385		2008	1	19
386		2008	1	20
387		2008	1	21

time

	A	B	C	D
region_id		region_name		
1		Alabama		
2		Arizona		
3		California		
4		Florida		
5		Georgia		
6		Hawaii		
7		Idaho		
8		Kansas		
9		Maryland		
10		Nevada		
11		Dakota		
12		Ohio		
13		Washington		

region

Figure [3.17] – Les tables temps et région.

A	B	C	D	E	F
customer_id	product_id	Time_id	region_id	price	quantity
6280	337	371	4	50	2
6280	1512	371	1	62	3
4018	963	963	11	40	1
4018	181	371	13	79	3
4018	1383	371	5	18	2
4018	1306	371	13	41	3
1418	1196	371	6	84	2
1418	360	371	8	62	2
1418	1242	371	3	96	2
1418	154	371	5	96	1
4382	483	371	13	88	2
1293	77	371	1	60	2
1293	533	371	8	84	2
1293	310	371	4	76	1
1293	1392	371	5	83	1
9305	1303	394	4	36	2
9305	748	394	7	40	2
9305	1270	394	13	69	1
5649	311	394	9	45	3
5649	194	394	5	36	1

sales

Figure [3.18] – La table vente.

3.5.4 – Chargement de données :

Après avoir créé les tables de l'entrepôt de données, il est maintenant temps pour nous de les charger dans Hive.

Comme SQL, Hive utilise également la commande LOAD pour importer des données, les figures [3.19], [3.20] montre le script que nous utilisons dans cette opération.

```
hive> load data local inpath '/home/ayoub/Documents/test/sales.csv' into table s
ales ;
Loading data to table default.sales
OK
Time taken: 2.499 seconds
hive>
```

```
hive> load data local inpath '/home/ayoub/Documents/test/region.csv' into table
region ;
Loading data to table default.region
OK
Time taken: 3.105 seconds
hive>
```

Figure [3.19] et [3.20] – Chargement des tables.

3.5.5 – Résultats :

Comme nous l'avons mentionné précédemment, nous n'attendions pas grand-chose d'Hadoop en ce qui concerne les petites données, cependant, nous avons effectué un test de requête pour le voir nous-mêmes.

Nous avons essayé de répondre aux questions que nous avons posées dans la section « 3.5 » de ce chapitre, les résultats sont dans le tableau ci-dessous en les comparant avec une exécution similaire dans un seul nœud avec MYSQL.

Le tableau suivant [3.0] montre cette comparaison.

Requête	Hive Multi Node Cluster	MYSQL
Requête 1 :	Temps pris **= 77.509 sec**	Temps pris **= 8.801**
Requête 2 :	Temps pris **= 46.178 sec**	Temps pris **= 5.397**
Requête 3 :	Temps pris **= 60.295**	Temps pris **= 5.717**
Requête 4 :	Temps pris **= 37.089**	Temps pris **= 2.015**

Tableau [3.0] – La comparaison entre exécution Hive multi nœud et MYSQL.

La figure suivante [3.21] est un exemple de ce test sur HIVE.

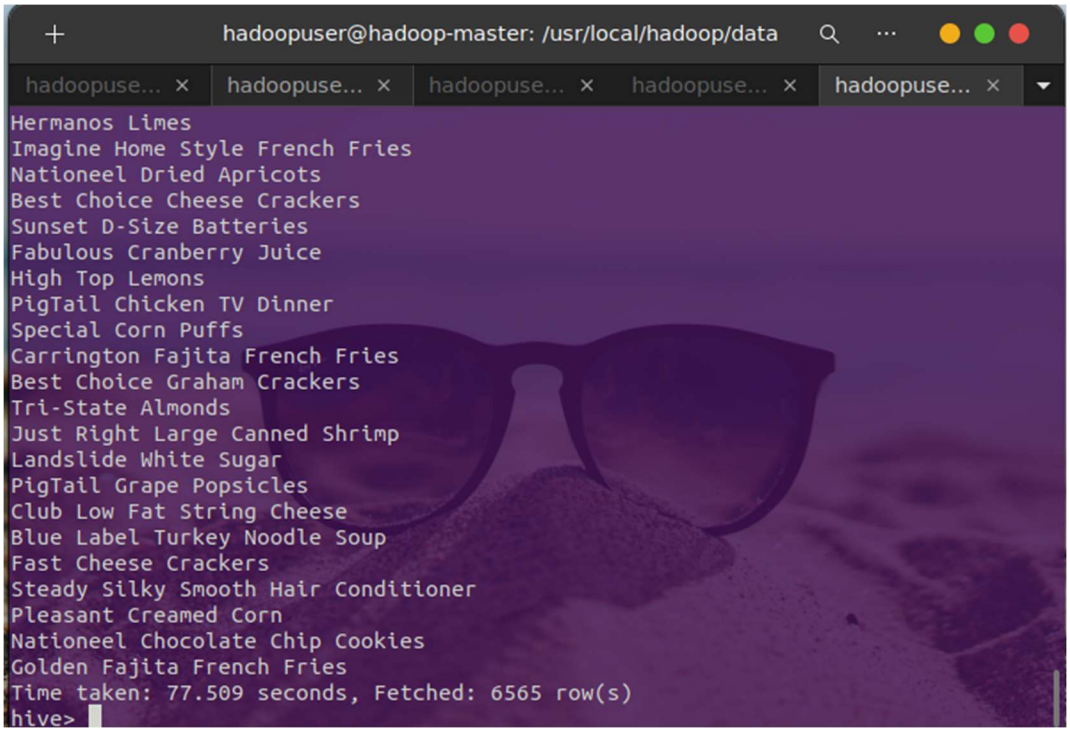


Figure [3.21] – Exemple de test sur Hive.

3.5.6 – Discussion des résultats :

Dès le premier regard, il est clair que l'exécution de ces requêtes sous HIVE est une situation perdante, car comme nous l'avons vu précédemment, la taille de bloc HDFS par défaut est de 128 Mo et le traitement de données qui sont significativement plus petites que la taille de bloc par défaut dégradera ses performances à cause de la fonction MapReduce.

La fonction MapReduce est une arme à double tranchant, au lieu de l'utiliser pour distribuer des données volumineuses à des morceaux plus petits et faciliter le traitement. Des données plus petites poseront problème car la fonction Map traite un bloc à la fois et si nous avons programmé 100 de ces fichiers de données plus petits de 5 Mo chacun, par exemple, MapReduce produira 100 fonctions Mapper pour traiter chaque bloc.

Et c'est exactement ce que nous avons remarqué lors de cette expérimentation, le temps de travail était beaucoup plus lent que de l'exécuter dans un seul nœud avec MYSQL.

Conclusion et Travaux Futurs

Les techniques de Big data (données massives) deviennent désormais un standard pour le stockage et l'analyse des grands volumes de données pour la prise de décisions. Dans ce mémoire nous nous sommes intéressés à la mise en œuvre d'un entrepôt de données en utilisant de telles techniques.

Nous avons présenté dans un premier lieu les concepts et les définitions relatifs à l'architecture et la modélisation des entrepôts de données. Nous avons expliqué par la suite comment les techniques du Big data permettent un traitement efficace de grandes quantités de données. Nous avons fini par illustrer l'installation et la configuration de l'écosystème Hadoop. Pour pouvoir interroger efficacement les données structurées avec le langage SQL, nous avons également installé l'outil Apache Hive qui est un système d'entrepôt de données distribué et tolérant aux pannes permettant des analyses à grande échelle.

En raison des contraintes de temps et des données disponibles, les résultats des tests effectués étaient prévisibles et confirment le fait que pour profiter pleinement des techniques Big data, il faut manipuler des données de tailles. Nous avons confirmé dans notre expérimentation que Hadoop n'est tout simplement pas conçu pour les petites données.

Ce projet a été une opportunité pour nous afin de mettre en valeur et d'approfondir les connaissances et les compétences acquises au cours de notre formation en Master sur les systèmes d'information décisionnels. Au cours du projet nous étions amenés à jouer le rôle du développeur et de l'utilisateur. Ainsi, l'expérience s'avère très positive et nous sommes convaincus que le travail effectué nous servira à l'avenir dans notre vie professionnelle.

A l'issue de ce travail, le premier constat dégagé est que la mise en œuvre d'un cluster multi-nœuds pour le big data est une tâche complexe. La partie du projet la plus fastidieuse et consommatrice en temps était la partie de configuration des différents nœuds du cluster. Nous étions de plus confronté à la problématique de disponibilité des données pour la partie expérimentale.

Bien que testée sur un ensemble réduit de données, notre installation a été conçue pour être générique et utilisée avec des tailles plus importantes de données.

Comme tout travail, le nôtre ouvre la voie pour plusieurs perspectives :

1- L'exploitation réelle de la solution proposée nécessite du matériel adéquat. Ainsi, il est recommandé d'utiliser des nœuds avec un minimum de plusieurs giga-octets de RAM et plusieurs téraoctets d'espace de stockage.

2- Étendre le jeu de données de manière à effectuer d'avantages d'expérimentations.

3- Enfin, d'un point de vue pratique, il serait intéressé que les données exploitées soient celles relatives aux parcours pédagogiques des étudiants universitaires.

Bibliographie

1. Swati Gore. **Hadoop and Map-Reduce** Université de Illinoi Chicago, Juilliet 2016, Thèse de Master.
2. Sevuga Perumal Chidambaram. **A Discussion on Testing Hadoop Applications:** International Journal of Advancements in Research & Technology, February-2014, Thèse de Master.
3. Manich P.Tembhurkar. **Overview On Performance Testing Approach In Big Data:** Shri Shankarprasad Agnihotri College of Engineering**,** Decembre 2014 Thèse de doctorat.
4. Kvn Krishna Mohan. **Query Optimization in Big Data Hadoop Using Hive,** Article de recherche, Juin 2016.
5. Anish Gupta. **Hive – Processing Structured Data in Hadoop** International Journal of Scientific & Engineering Research, June-2017, Thèse de Master.
6. Jimmi Lin. **Hadoop and MapReduce** Université de Roma TRE, 2009, Thèse de Master.
7. Barkha Jain and Manish Kakhani. **Query Optimization in Hive for Large Datasets,** Université Modi, Laxmangarh, Rajasthan, Avril 2015, Thèse de Doctorat.
8. Tien Duc Dinh. **Hadoop Performance Evaluation :** Université de Heidelberg March 10, 2009, Thèse de Master.
9. Annabel Mckenzie. **Hadoop WordCount Explained,** 2016, Article sur Internet.
10. Stella Gatziu Grivas **Data Warehousing: Concepts and Mechanisms,** Janiveir 1999, Université de ASA Northwestern Switzerland.
11. Surajit Chaudhuri. **An Overview of Data Warehousing and OLAP Technology,** Mars 1997, Université de Hewlett Packard, Thèse de Master.
12. Sreenivas Sremath Tirumala, **ETL Tools for Data Warehousing: An Empirical Study of Open Source Talend vs Microsoft SSIS,** Janvier 2015, Université de Auckland, Thèse de Master.

13. Bernard Espinass, **Entrepôt de Données : ROLAP, MOLAP et HOLAP,** Décembre 2015, Université de AIX- Marseille, Présentation.

14. Andrzej Leszek Ziora, **The Role of Big data Solutions in the Management of Organizations,** 2015, Université de Technologie Czestochowa, Thèse de Master.

15. Youssra Riahi, **Big Data and Big Data Analytics: Concepts, Types and technologies,** Septembre 2018, Université de Rabat, Thèse de Doctorat.

16. Elisabeta Zagan. **Hadoop: A comparative Study Between Single Node Cluster and Multi Node Cluster,** Université de Suceava, Roumanie**,** 2021, Thèse de Master.

17. Vineet Sajwan, **HDFS: Architecture and Internals,** May 2015, Université de Noida Sector India, Thèse de Master.

18. Sunnie Chung, **Apache Hive,** Online Courses, CIS 612.

19. S. Chafki. **Modélisation Dimensionnelle**, École de technologie supérieure Canada, 2011, Thèse de Master.

20. Archana Bhaskar, **Optimized Memory Model for Hadoop MapReduce Framework,** October 2019, Université de Riva India, Thèse de Master.