الجزائرية الديمقراطية الشعبية الجمهورية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة العليا للإعلام الآلي - 08 ماي 1945 – بسيدي بلعباس

Ecole Supérieure en Informatique

-08 Mai 1945- Sidi Bel Abbes

MEMOIRE

En vue de l'obtention du diplôme de **Master**

Filière : **Informatique**

Spécialité : **Ingénierie des Systèmes Informatiques (ISI)**

Thème

Étude des techniques d'optimisation de performances des requêtes analytiques dans un environnement Big Data

---

Présenté par **:**

- Mr Ahmed Derkaoui
- Mlle Asmaa Bouchikhi

Soutenu le : **30/09/2020** Devant le jury composé de :

-	M. AMAR BENSABER Djamel	MCA	Président
-	M. KECHAR MOHAMED	MCB	Encadreur
-	M. BENCHERIF Khayra	MCB	Examinatrice

Année Universitaire : 2019 / 2020

Hommage

À mon cher défunt père Derkaoui Nourredine, mort le 29 Juillet 2020 du au COVID-19

À l'honneur de sa mémoire, je lui dédie cette section, pour qu'il soit part de ma soutenance. lui qui était présent dans tout mes moments, qui m'a formé dans la vie et à sacrifier la sienne pour me voir réussir.

Remerciements

”au nom de dieu clément et miséricordieux”

Nous tenons à exprimer notre gratitude et nos sincères remerciements à notre encadreur **Dr m.Kechar**, pour nous avoir guidé et orienté pendant toute la durée de la réalisation de notre projet, ainsi que pour ces précieux conseilles et ses encouragement.

Nous tenons aussi à remercier nos enseignants durant toutes nos années de formation, ainsi que staff administratif de notre école ESI-SBA.

Nous souhaitons remercier aussi nos familles respectives pour tout leurs sacrifices et leur soutien et encouragements durant toute notre vie.

Aux chers membres du Jury, nous vous remercions pour avoir accepter de d'évaluer notre travail et nous avoir accordé l'honneur de le juger.

Cordialement,

a.Derkaoui
a.Bouchikhi

Résumé

Avec l'apparition du Big data, les bases de données sont devenu énormes en volume et ont des modèles de données différents . de nouvelles techniques de traitement et d'optimisation des requêtes ont été développées pour gérer les requêtes complexes et interactives. L'objectif principal de notre recherche est d'améliorer les performances de l'exécution des requêtes pour le Big Data.Dans ce mémoire on a représenté les déférents techniques et approches utilisées pour optimiser les performances des requêtes analytique dans l'environnement de Big Data .

Les Mots-Clés : Big Data ,Bases de données, Optimisation, Requête analytique, Data Analysts, Hadoop, MapReduce, NoSQL,Hive....

Abstract

With the advent of Big Data, databases have become huge in volume and have different data models. New query processing and optimization techniques have been developed to handle complex and interactive queries. The main objective of our research is to improve the performance of the execution of queries for Big Data.

In this brief, we have presented the techniques and approaches used to optimize queries Big Data .

Keywords: Big Data ,Databases, Optimization, Analytical Query, Data Analysts, Hadoop, MapReduce, NoSQL,Hive....

Contents

List of Figures

List of Tables

Chapter 1

Introduction générale

1.1 Contexte

Avant la révolution du Big Data, les entreprises ne pouvaient pas stocker toutes leurs archives pendant de longues périodes ni gérer efficacement d'énormes ensembles de données. En effet, les technologies traditionnelles ont une capacité de stockage limitée, des outils de gestion rigides et sont coûteuses. Ils manquent d'évolutivité, de flexibilité et de performances nécessaires dans le contexte du Big Data.
Les Big data sont définies comme l'ensemble de données dont la taille dépasse la capacité de traitement des bases de données ou des ordinateurs traditionnels. Ces données sont stockées dans des Lac de données (data lake) : une technique qui nous permet de stocker n'importe quel type de donnée pour les interroger, traiter et les analyser en fonction du besoin..
L'analyse des mégas donnés peut-être vue comme l'extraction ou le traitement de données massives, récupérant ainsi des informations utiles à partir d'un vaste ensemble de données . L'analyse des mégas données peut-être caractérisée par plusieurs propriétés, telles qu'un volume important, une variété de sources différentes et une vitesse (vitesse) qui augmente rapidement .
Les outils et applications existantes et traditionnelles deviennent insuffisants pour traiter une grande quantité de données. Hadoop a expliqué et répondu le problème de la manipulation et du traitement de ces téraoctets et pétaoctets de données . Hadoop est un cadre logiciel pour une informatique fiable, évolutive, parallèle et distribuée. Au lieu de compter sur le matériel coûteux et systèmes coûteux pour le traitement et le stockage des données, Apache Hadoop autorise le traitement en parallèle sur Big Data sur le matériel de base. La norme de programmation HDFS et MapReduce s'accorde à ces tâches analytiques gour-mandes en Big Data en

raison de son architecture évolutive et de sa capacité à traiter les données en parallèle dans des clusters multi-nœuds..
En effet, la gestion du Big Data nécessite des ressources importantes, de nouvelles méthodes et des techniques puissants. Plus précisément, le Big Data nécessite de nettoyer, traiter, analyser, sécuriser et fournir un accès granulaire à des ensembles de données en constante évolution. Les entreprises et les industries sont de plus en plus conscientes que l'analyse des données devient de plus en plus un facteur essentiel pour être compétitif, pour découvrir de nouvelles perspectives et pour personnaliser les services.
L'objectif principal de notre recherche est d'étudier les différentes techniques et approches d'optimisation des requêtes en matière de recherche et en matière de résultat . Tel que le partitionnement, sélection d'indexé et sélection des vues matérialisées, afin d'améliorer les performances de l'exécution des requêtes pour le Big Data .

1.2 Challenges

Travailler dans l'industrie du Big Data apporte beaucoup de satisfaction ainsi que de nombreux défis. Vous gérez une quantité (presque) incalculable de données. Comme on peut le deviner, le principal défi ici est le suivant: à quelle vitesse pouvez-vous gérer les données? Combien de temps faut-il pour insérer une ligne dans un tableau avec des milliards de lignes? À quelle vitesse pouvez-vous rechercher une ligne parmi des milliards d'enregistrements ?

Gestion de Big Data : Concernant le traite du Big Data. L'un des défis est de savoir comment collecter, intégrer et stocker, avec moins d'exigences matérielles et logicielles . Un autre défi est la gestion du Big Data. Il est crucial de gérer efficacement le Big Data afin de faciliter l'extraction d'informations fiables et d'optimiser les dépenses.
La gestion du Big Data signifie nettoyer les données pour la fiabilité, agréger les données provenant de différentes sources et encoder les données pour la sécurité et la confidentialité. Cela signifie également assurer un stockage Big Data efficace et un accès basé sur les rôles à plusieurs points de terminaison distribués. En d'autres termes, l'objectif de la gestion du Big Data est de garantir des données fiables, facilement accessibles, gérables, correctement stockées et sécurisées[41].

Nettoyage Big Data : Ces cinq étapes (Nettoyage, Agrégation, Encodage, Stockage et Accès) ne sont pas nouvelles et sont connues dans le cas de la gestion traditionnelle des données. Le défi du Big Data est de savoir comment gérer la complexité de la nature du Big Data (vitesse, volume et variété) (Khan et al., 2014)[23] et la traiter dans un environnement distribué avec un mélange d'applications. En fait, pour des résultats d'analyse fiables, il est essentiel de vérifier

la fiabilité des sources et la qualité des données avant d'engager des ressources. Cependant, les sources de données peuvent contenir des bruits, des erreurs ou des données incomplètes. Le défi est de savoir comment nettoyer ces énormes ensembles de données et comment décider quelles données sont fiables, quelles données sont utiles.

Agrégation Big Data : Un autre défi est de synchroniser les sources de données externes et les plateformes Big Data distribuées (y compris les applications, référentiels, capteurs, réseaux, etc.) avec les infrastructures internes d'une organisation. La plupart du temps, il ne suffit pas d'analyser les données générées au sein des organisations. Afin d'extraire des informations et des connaissances précieuses, il est important d'aller plus loin et d'agréger les données internes avec des sources de données externes. Les données externes peuvent inclure des sources tierces, des informations sur les fluctuations du marché, les prévisions météorologiques et les conditions de circulation, les données des réseaux sociaux, les commentaires des clients et les commentaires des citoyens. Cela peut aider, par exemple, à maximiser la force des modèles prédictifs utilisés pour l'analyse [41].

Capacités des systèmes déséquilibrés : Un problème important est lié à l'architecture et à la capacité de l'ordinateur. En effet, on sait que les performances du processeur doublent tous les 18 mois suivant la loi de Moore, et que les performances des disques durs doublent également au même rythme. Cependant, les opérations d'E / S ne suivent pas le même modèle de performances. (Par exemple, les vitesses d'E / S aléatoires se sont améliorées modérément tandis que les vitesses d'E / S séquentielles augmentent lentement avec la densité) (Chen et Zhang, 2014) [19]. Par conséquent, ces capacités système déséquilibrées peuvent ralentir l'accès aux données et affecter les performances et l'évolutivité des applications Big Data. Sous un autre angle, on peut remarquer les différentes capacités des appareils sur un réseau (e.i, capteurs, disques, mémoires). Cela peut ralentir les performances du système [41].

Analyse de Big Data : Le Big Data apporte de grandes opportunités et un potentiel de transformation pour divers secteurs; d'autre part, il présente également des défis sans précédent pour exploiter des volumes de données aussi importants et croissants. Une analyse avancée des données est nécessaire pour comprendre les relations entre les entités et explorer les données. Par exemple, l'analyse des données permet à une organisation d'extraire des informations précieuses et de surveiller les modèles susceptibles d'affecter positivement ou négativement l'entreprise.
D'autres applications basées sur les données nécessitent également une analyse en temps réel, comme la navigation, les réseaux sociaux, la finance, la biomédecine, l'astronomie, les systèmes de transport intelligents. Ainsi, des algorithmes avancés et des méthodes efficaces d'exploration

de données sont nécessaires pour obtenir des résultats précis, pour surveiller les changements dans divers domaines et pour prédire les observations futures. Cependant, l'analyse du Big Data reste difficile pour de nombreuses raisons: la nature complexe du Big Data, y compris les 5V, le besoin d'évolutivité et de performance pour analyser ces énormes ensembles de données hétérogènes avec une réactivité en temps réel (Wang et al., 2016, Tsai, 2016) [37] [32] .

1.3 Problématique

Il existe diverses techniques analytiques, notamment l'exploration de données, la visualisation, l'analyse statistique et l'apprentissage automatique. De nombreuses études abordent ce domaine en améliorant les techniques utilisées, et en proposant de nouvelles ou en testant la combinaison de divers algorithmes et technologies.
Ainsi, le Big Data a poussé le développement des architectures systèmes, du matériel ainsi que des logiciels. Cependant, nous avons encore besoin d'un avancement analytique pour faire face aux défis du Big Data et au traitement des flux.
L'un des problèmes est de savoir comment garantir la rapidité de la réponse lorsque le volume de données est très important? [27]
Dans le cadre de gestion Big Data, une requête est exécutée en accédant à des données réparties sur un cluster de stockage matériel ou des nœuds de données en tant que système de fichiers distribué (DFS) par des tâches MapReduce. Par conséquent, le coût de traitement des requêtes n'est pas seulement le coût d'accès aux lignes de tables de base stockées sur le disque. La surcharge DFS de la distribution des données dans les nœuds de données, la cartographie et le suivi du traitement, puis la réduction des résultats sont également impliquées [23] .

1.4 Organisation du mémoire

Dans le chapitre 2 nous allons introduire des notions de bases sur les différents aspects de l'analyse données dans le contexte du Big Data puis dans le Chapitre 3 nous allons étudier les différents techniques d'optimisation de performances des requêtes analytiques dans un environnement Big Data et quelques approches basées sur ces techniques et en fin conclure notre étude dans le chapitre 4 .

Chapter 2

Background

De nos jours les systèmes d'informations sont omniprésent dans diffèrent domaines avec une croissance sur le traitement et stockage des données.
Les systèmes d'informations jouent un rôle important dans la prise de décision, d'où l'apparition de l'informatique décisionnelle.
L"informatique décisionnelle (en anglais business intelligence (BI) ou décision support système (DSS) est l'informatique à l'usage des décideurs et des dirigeants d'entreprises. Elle désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider, modéliser et restituer les données, matérielles ou immatérielles, d'une entreprise en vue d'offrir une aide à la décision et de permettre à un décideur d'avoir une vue d'ensemble de l'activité traitée.

Dans ce chapitre nous allons définir les systèmes informatiques et les méthodes utilisé pour extraire les connaissances et les informations stockés.

2.1 Entrepôts des données

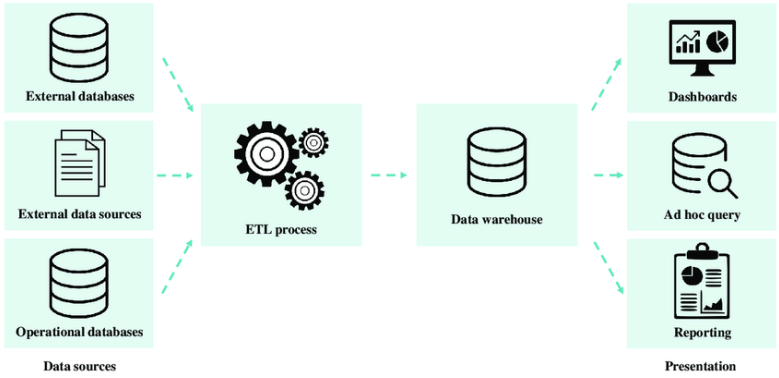
Pour subvenir aux besoins analytiques, l'expression des requêtes est souvent complexe et nécessite un accès á différentes tables d'une base de données relationnelles classiques.
Les bases de données classiques sont destinées pour sauvegarder les transactions du système et assurer une cohérence de son état, permettant ainsi la gestion des transactions concurrentes. Ce modèle de systèmes ne répond pas aux exigences de la nature des requêtes analytiques, d'où l'apparition des Entrepôts de données.

2.1.1 Définition d'un entrepôt de données

Un entrepôt de données est un système informatique dédié aux besoins analytiques, il est séparé du système transactionnel.
Un entrepôt de données centralise et consolide de grandes quantités de données provenant de

plusieurs sources. La capacité analytique d'un entrepôt de donnée permet aux décideur d'avoir un accès à des informations stratégiques pour la prise de décision. Un entrepôt de données est souvent vue comme repère historique d'une organisation et souvent considéré comme sa source unique de vérité.

Figure 2.1: Processus de création d'un entrepôt



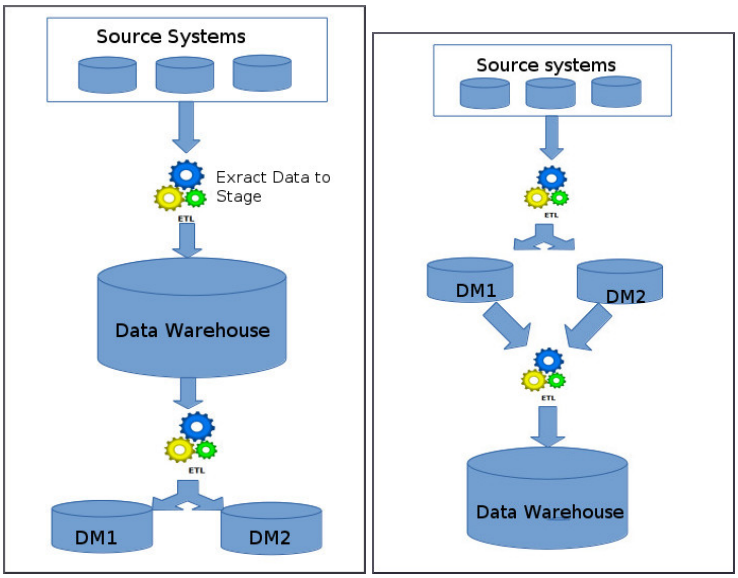
Le terme Entrepôt de données à été formalisé par William H. (Bill) Inmon qui est considéré un des fondateurs. "Un entrepôt de données est une collection de orientées sujet, intégrées, non-volatiles et variante dans le temps qui sert á la prise de décision dans le management. L'entrepôt de données contient des données granulaires de l'entreprise" [1] .

2.1.2 Modélisation d'un entrepôt de données

Approches de modélisation

Pour construire l'entrepôt de données, le processus ETL (Extract, Transforme and Load) se charge de collectionner les données des différentes sources et les transformer pour intégrer dans le modèle de l'entrepôt.
Il existe deux approches :

- BOTTOM-UP : proposée par R.Kimball, qui définie la création de l'entrepôt de données à partir de la collection des différents Data Mart (Magasin de données).
- TOP-DOWN : proposée par W.H.Inmon qui définie la création de l'entrepôt de données est de générer en suite les Data Mart (Magasin de données) .



(a) (b)

Figure 2.2: (a) TOP-DOWN (b) BOTTOM-UP

Data Mart et Data Warehouse

Un Data Mart, ou magasin de données est référencées comme sous-ensemble de l'entrepôt il se construit par une collections de données dédiées á une fonction spécifique de l'entreprise.

	Data Warehouse	**Data Mart**
Application	Data warehouse est indépendant de l'application.	Les Data mart sont spécifiques à l'application du système d'aide à la décision.
Type de système	Centralisé	Décentralisé
Facilité de construction	Difficile à construire	Simple à construire
Type de schéma utilisé	Constellation de faits	Étoile et flocon de neige
Modèle de données	Top-down	Bottom-up
Utilisation de la dénormalisation	Les données sont légèrement dénormalisées.	Les données sont fortement dénormalisées.
Forme de données	Détaillé	Résumé

Table 2.1: Tableau comparatif entre data warehouse et data mart.

Modèles de données

Par définition les données d'un entrepôt de données sont :

- **Orientées sujet** : Les données sont mondialise au tour d'un besoin analytique spécifique, par exemple, L'analyse des ventes .

- **Intégrées** : Les données sont regroupées de différentes sources et transformer pour l'intégration avec le modèle du sujet a traiter .
- **Non-volatile** : Les données ne changent suite aux résultats des traitement.
- **Variante dans le temps** : Chaque nouvelle données est insérées. Un référentiel de temps doit être mis en place afin de pouvoir identifier chaque donnée dans le temps.

La modélisation des données dans un entrepôt est une modélisation multidimensionnelle qui repose sur le choix des :

- **Tables de faits** : Les tables de faits contiennent les données que l'on souhaite voir apparaître dans les rapports d'analyse, sous forme de métriques. Les données des tables de faits sont agrégées à partir des tables de dimensions qui leur sont associées.
- **Tables de dimensions** : Les tables de dimensions sont utilisées pour décrire les données que l'on souhaite stocker dans le Data Warehouse.
- **Les niveaux de granularité** : Déterminent le niveaux de détails des tables de faits et des tables de dimensions [7].

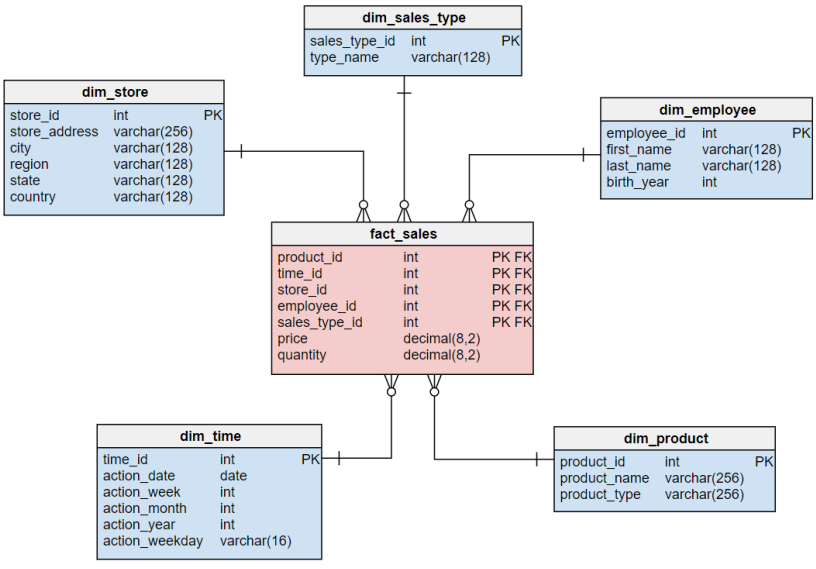


Figure 2.3: Exemple de table de fait et de dimensions

Les modèles de données

- **Modèles en étoile** : dans ce modèle la table des faits contient des mesures et les clef de références vers les tables de dimensions. Les tables de dimensions sont démoralisées.

- **Modèles en flocon** : similaire au modèle en étoile sauf que les tables de dimensions sont normalisées.
- **Modèles en constellation** : est la collection de plusieurs tables de faits ayant des dimensions commune, utilisé généralement pour regrouper les data Mart.

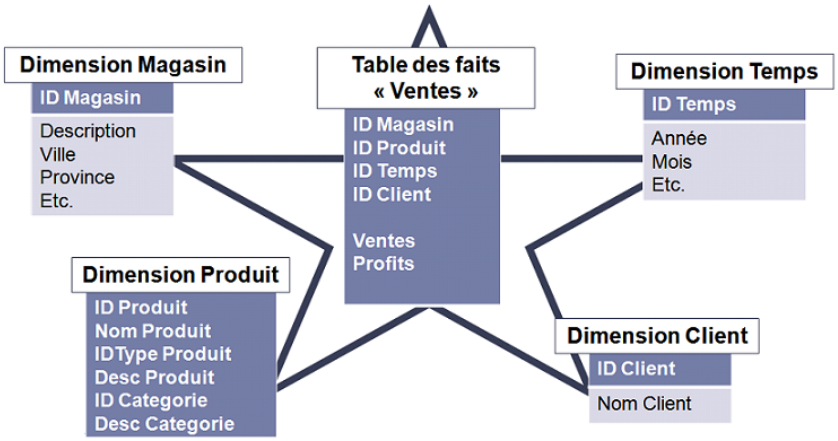


Figure 2.4: Modèle en étoile

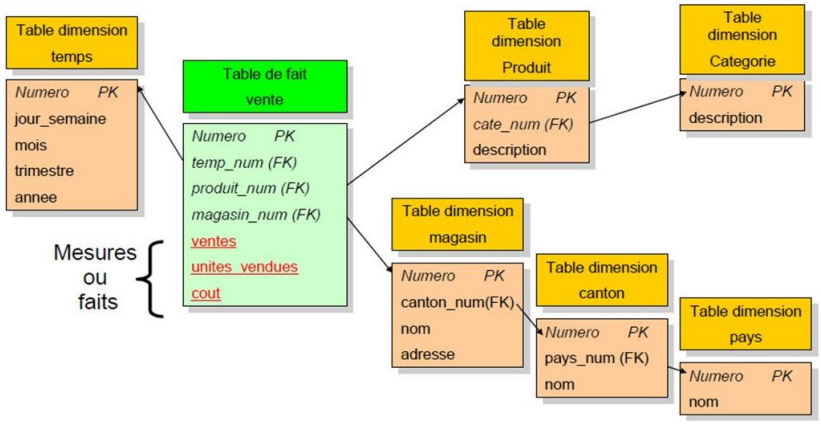


Figure 2.5: Modèle en flocon

la différence entre le modèle en étoile et le modèle en flocon est dans la normalisation et démoralisation des tables de dimensions:
Le schéma en étoile est facile à interroger et nécessite peu de jointure mais occupe plus d'espace sur disque et un effort dans la phase ETL.
Le schéma en flocon occupe moins d'espace disque grâce à la normalisation des dimensions, mais par contre pose un problème de jointures multiples et de requêtes complexes .

2.1.3 L'analyse des données

la modélisation multidimensionnelle dans un entrepôt de données permet de les analyser sous forme de cubes grâce au concept OLAP.

OLAP - Online Analytical Processing

Le OLAP, ou Online Analytical Processing, est une technologie de traitement informatique utilisé dans des systèmes d'aide á la décision et du Business Intelligence, elle permet de traiter et comparer des données complexes pour générer des rapports. Les données OLAP sont stockées sur une base multidimensionnelle, aussi appelées Cubes OLAP, pour faciliter ce type d'analyses.

OLAP cube

Un cube OLAP est une méthode de stockage de données sous forme multidimensionnelle. les données souvent référencées comme mesures sont classées par dimensions. Les données des dimensions des cubes OLAP sont souvent traités pour accélérer considérablement l'interrogation par rapport aux bases de données relationnelles.

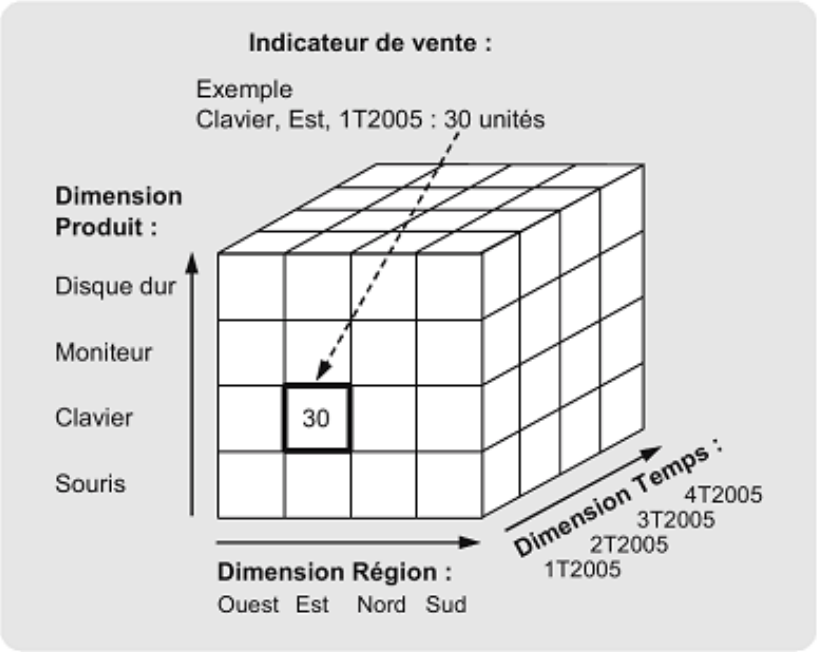


Figure 2.6: Cube OLAP

Les opérations sur un cube OLAP

- Roll-Up/Drill-UP: est l'agrégation d'une dimension .

- Drill Down : est l'operation inverse de Roll-Up, qui permet de decomposer la dimension.
- Slice : Une sélection d'une dimension .
- Dice : similaire au slice mais avec la sélection de plus d'une seule dimension .
- Pivot : Rotation sur les axes de dimensions .

Les types des systèmes OLAP

- **ROLAP** :permet d'accéder directement aux données stockées dans les bases de données relationnelles. Ainsi, les entreprises peuvent directement exploiter leurs SGBDR déjà en place. On effectue les requêtes avec un produit ROLAP en utilisant le langage SQL.
  Le problème des produits ROLAP est qu'ils sont trop dépendants du SQL. Or, ce langage est limité est souvent inflexible grammaticalement. Un autre inconvénient est que les données doivent être extraites et reformatées avant de pouvoir effectuer une requête.
- **MOLAP** : Les produits MOLAP permettent aux utilisateurs de modéliser les données au sein d'un environnement multidimensionnel, plutôt que de fournir une vue multimensionnelle de données relationnelles comme le font les produits ROLAP.
- **HOLAP** : Les produits HOLAP combinent les meilleures fonctionnalités du MOLAP et du ROLAP dans une seule architecture. Ainsi, ces produits corrigent les inconvénients de ces deux types de produits. Ils peuvent être utilisés aussi bien sur une base de données multidimensionnelle que sur une base de données relationnelle.
- **HTAP** Le terme HTAP a été inventé en 2014 par Gartner. Ce terme décrit les systèmes in-memory data permettant d'effectuer à la fois des traitements OLAP (online analytical processing) et OLTP (online transaction processing). Le HTAP repose sur un traitement plus puissant, plus récent, généralement distribué.

2.2 Big Data analytique

2.2.1 Introduction au Big Data

Définition du Big Data

Contrairement aux données traditionnelles, le terme Big Data fait référence à de grands ensembles de données croissants comprenant des formats hétérogènes: données structurées, non structurées et semi-structurées. Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés. Ainsi, les outils traditionnels de Business Intelligence statiques ne peuvent plus être efficaces dans le cas des applications Big Data [41].

Gartner (une entreprise americaine de conseil et de recherche dans le domaine technique avancées) définit le big data comme suit: **Des données au volume, à la vélocité et à la variété conséquents, exigeant un traitement innovant et rentable de l'information pour en améliorer la visibilité et favoriser la prise de décisions [53] .**

Caracteristique de Big Data

La plupart des data scientists et des experts définissent le Big Data par les trois principales caractéristiques suivantes (appelées les 3V) (Furht et Villanustre, 2016) [41] :

- **Volume** : De grands volumes de données numériques sont générés en continu à partir de millions d'appareils et d'applications (TIC, smartphones, codes produits, réseaux sociaux, capteurs, journaux, etc.). Selon McAfee et al. (2012)[11], on estime qu'environ 2,5 exaoctets ont été générés chaque jour en 2012. Ce montant double tous les 40 mois environ. En 2013, le total des données numériques créées, répliquées et consommées a été estimé par l'International Data Corporation (une société qui publie des rapports de recherche) à 4,4 zettaoctets (ZB). Il double tous les 2 ans. En 2015, les données numériques sont passées à 8 ZB . Selon le rapport d'IDC, le volume de données atteindra 40 octets Zeta d'ici 2020 et augmentera de 400 fois maintenant (Kune et al., 2016) [36].

- **Vitesse**: les données sont générées de manière rapide et doivent être traitées rapidement pour extraire des informations utiles et des informations pertinentes. Par exemple, Wallmart (une chaîne internationale de vente au détail à prix réduits) génère plus de 2,5 Po de données par heure à partir des transactions de ses clients. YouTube est un autre bon exemple qui illustre la vitesse rapide du Big Data.

- **Variété**: les Big Data sont générées à partir de diverses sources distribuées et dans plusieurs formats (par exemple, des vidéos, des documents, des commentaires, des jour-

naux). Les grands ensembles de données sont constitués de données structurées et non structurées, publiques ou privées, locales ou distantes, partagées ou confidentielles, complètes ou incomplètes, etc. Emani et coll. (2015) [41] et Gandomi et Haider (2015) [27] indiquent que plus de V et d'autres caractéristiques ont été ajoutées par certains acteurs pour mieux définir le Big Data:**Vision** (un objectif),**Vérification** (données traitées conformes à certaines spécifications), **Validation** (l'objectif est rempli), de la **valeur** (des informations pertinentes peuvent être extraites pour de nombreux secteurs), de la complexité (il est difficile d'organiser et d'analyser le Big data en raison de l'évolution des relations de données) et de l'immuabilité (les Big data collectées et stockées peuvent être permanentes si elles sont bien gérées).

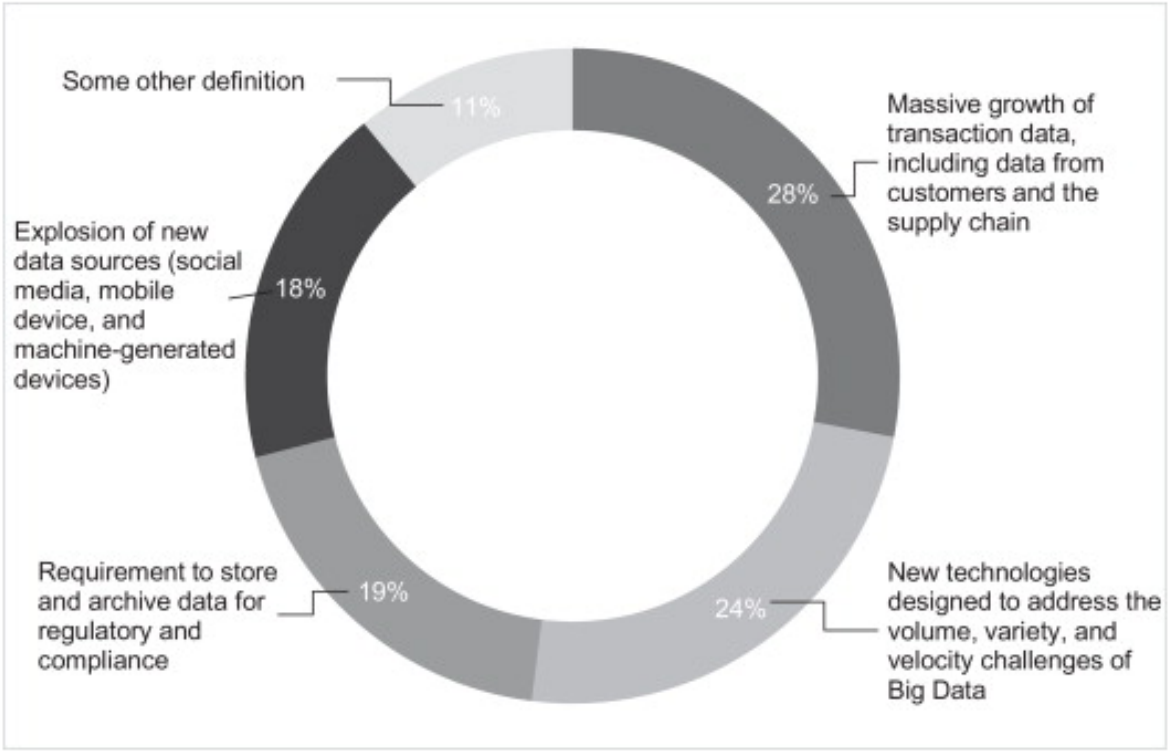


Figure 2.7: Définitions Big Data basées sur une enquête en ligne

Définition de l'analytique de Big Data

L'Analytique fait référence aux processus d'examen, de collecte, d'analyse et de transformation du big data pour en tirer les informations utiles qui se cachent dans les données. L'objectif est de tirer des conclusions et de résoudre les éventuels problèmes. Cependant, l'Analytique n'est pas seulement un outil grâce auquel vous pouvez analyser un événement passé avec des données passées, elle vous permet également de décrire des modèles futurs avec les données d'aujourd'hui [14] .

L'analyse Big Data offre de nouvelles dimensions d'analyse comme la prise en compte de la chronologie des événements et du contexte des évènements. A la différence du Data Analytiques, l'analytique Big Data applique des traitements différents pour traiter plusieurs problématiques simultanément et n'est pas prisonnier d'un schéma de relations prédéfini .

En résumé, le big data fait référence à des ensembles de données non structurées et assez complexes qui nécessitent des outils spécifiques pour les traiter. L'Analytique consiste quant à elle à structurer ces données de manière à les transformer en de précieuses contributions au processus d'analyse [53] .

Catégories de l'analytique

Gartner divise donc cette discipline en quatre catégories différentes :

- **l'Analytique descriptive** : décrit la situation actuelle comme elle est.(fournit des informations sur ce qui a été réalisé et permet donc de comprendre ce qui s'est produit).
- **l'Analytique diagnostique** : permet de comprendre pourquoi des événements et des modèles particuliers se réalisent .
- **l'Analytique prédictive** : permet d'anticiper le comportement futur en s'appuyant sur des données passées.(a pour objectif de fournir des modèles afin de prévoir ce qui pourrait se réaliser. Elle s'appuie sur le Data Mining qui fournit des modèles statistiques. Une des techniques courantes est l'analyse de régression qui prédit les valeurs de plusieurs variables liées entre elles).
- **l'Analytique prescriptive** : en se basant sur les prévisions futures et les données actuelles, nous conseille sur les actions que nous devriez entreprendre.

Processus d'analyse Big Data

Le processus global d'extraction des informations à partir du big data peut être divisé en cinq étapes (Labrinidis et Jagadish, 2012) [27]. Ces cinq étapes forment les deux sous-processus principaux: la gestion des données et l'analyse.

- '**La gestion des données** implique des processus et des technologies de support pour acquérir et stocker des données et pour les préparer et les récupérer pour analyse.
- **L'analyse** quant à elle, fait référence aux techniques utilisées pour analyser et acquérir des informations à partir du big data. Ainsi, l'analyse de Big Data peut être considérée comme un sous-processus dans le processus global d ' extraction d'informations à partir de Big Data.

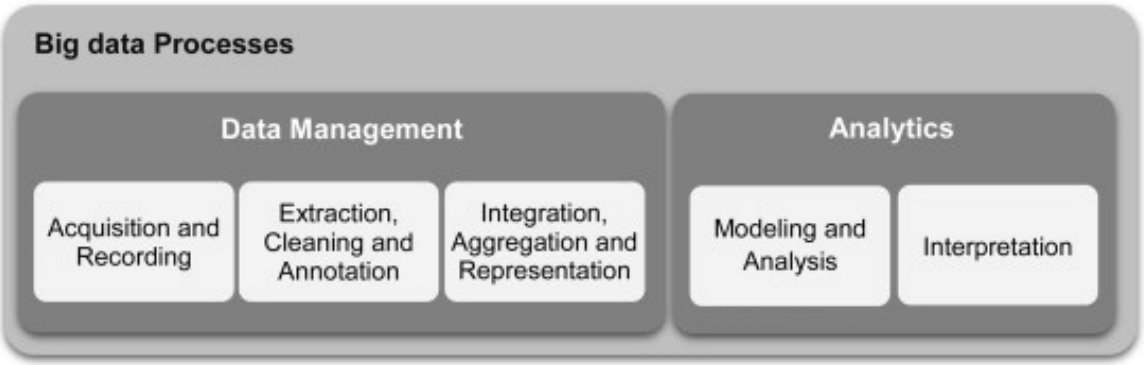


Figure 2.8: Processus d'analyse Big Data

2.2.2 Data lake

Dans l'ère du Big Data, le terme Data Lake a vu le jour, souvent confondu au Data Warehouse le Data Lake se différencie de ce dernier sur différents aspects dont on peu résumer sur ces points [48] :

- **Données**: Le Data Warehouse utilise des donnes de nature structurées, par contre le Data Lake permet d'intégrer les donne de nature semi-structurés ou non structuré grâce aux technologies Big Data .
- **Traitement**: Le Data Warehouse repose sur le concept du schéma en écriture ce qui limite l'espace d'interactions avec le système aux requêtes analytiques dont le système a été conçu via le processus ETL, par-contre le DATA Lake permet l'interaction avec diffèrent types de données est possible, seul lorsque les données sont interrogées, ils

seront transformée selon l'application de l'utilisateur via le processus ETL en se basant sur le concept de schéma en lecture.

- **Stockage et coûts** : les Data Lake sont souvent conçus en se basant sur des technologies big data open source qui sont conçu pour des serveur et un matériel de commodité contrairement aux coûts élevée des licences et du stockage dans un environnement Data Warehouse classique .
- **Flexibilité**: Le data warehouse est un environnement hautement structuré avec le modèle schéma en écriture tout changement dans la conception de ce dernier engendre des coûts de maintenance. Le Data Lake se base sur le concept de schéma en lecture qui offre une flexibilité et une évolution plus souple .
- **Sécurité**: Les environnements Data Warehouse ont atteint un niveau de maturité en terme de mécanismes de sécurité, les Data Lake sont en constante amélioration et des recherche dans le domaine de la sécurité de ces derniers sont en productions.
- **Utilisateurs**: Plusieurs expert sont habitués aux concepts des Data Warehouse par contre les data scientistes sont les principaux utilisateur des data lake due a la nature et l'hétérogénéité de ces données.

Déférence entre Data lake et Data warehouse

Le Data Lake peut être vu comme la nouvelle approche du Data Warehouse en utilisant les technologies Big Data et le traitement distribué et le stockage distribué offert par le paradigme MapReduce et le système de stockage HDFS. Dans la prochaine section nous allons détailler Sur Hadoop et son Écosystème tel que Hive, qui constituent la base des Data Lake récents .

Critère	Data Lake	Data Warehouse
Données	Structurées, semi-structurées et non structurées	Structuré, données traitées
Traitement	Schéma en Lecture	Schéma en Écriture
Stockage et coûts	coûts faible	coûteux
Flexibilité	Agile, configuration flexible	Moins agile et configuration fixée
Sécurité	En développement	Mature
Utilisateurs	Décideurs et Professionnels du Business	Data Scientistes et Analystes connaisseur du domaine

Table 2.2: Tableau comparatif entre Data Lake et Data Warehouse

Entrepôt de données dans un environnement du Big Data

L'entrepôt de données dans le contexte du Big Data repose à utiliser les principes du Data Warehousing dans un environnent distribué sous des technologies Big Data. Dans un environnement Big Data, Il existe plusieurs contraintes vis-a-vis de la variété de données et l'aspect non structuré de ces dernières. Le processus ETL permet alors d'intégrer les données de différentes sources et les stocker souvent dans un système de fichier distribué comme HDFS ou dans une base de donnée NoSQL.

Le traitement analytique des données se fait sous le paradigme du MapReduce, Il existe aussi des technologies permettant l'abstraction de ce concept et d'interroger les données d'une manière similaire au SQL, tel que HiveQL de Apache Hive.

Le concept du big data repose sur l'utilisation du calcule distribué.

2.2.3 Apache Hadoop

Définition et utilisation

Hadoop est un framework open source qui permet de stocker et de traiter des big data dans un environnement distribué sur des clusters d'ordinateurs à l'aide de modèles de programmation simples. Il est conçu pour passer d'un serveur unique à des milliers de machines, chacune offrant un calcul et un stockage locaux [21].
Hadoop a été créé par Doug Cutting et Mike Cafarella en 2005. Il a été initialement développé pour soutenir la distribution du projet de moteur de recherche Nutch. Doug, qui travaillait chez Yahoo! à l'époque et est maintenant architecte en chef de Cloudera [18].
Hadoop est basé sur Java qui prend en charge le traitement de grands ensembles de données dans un environnement informatique distribué. Il est basé sur le système de fichiers Google ou GFS (lu comme G-F-S).
Hadoop est un cadre logiciel pour une informatique fiable, évolutive, parallèle et distribuée. Au lieu de compter sur matériel coûteux et systèmes coûteux pour le traitement et le stockage des données, Apache Hadoop autorise le traitement en parallèle sur Big Data sur le matériel de base.
La norme de programmation HDFS et MapReduce s'accorde à ces Tâches analytiques gourmandes en Big Data en raison de son architecture évolutive et de sa capacité à traiter les données en parallèle dans des clusters multi-nœuds.
MapReduce, et son projet open source existant appelé Hadoop, permet le traitement parallèle d'une énorme quantité de données et la partition automatique des données, la distribution des

données, la tolérance aux pannes et la gestion de l'équilibrage de charge, ce qui permet enfin une informatique fiable et évolutive.
Le taux de croissance rapide et élevé des données défis pour les grandes entreprises comme Facebook, Google, Amazon et Yahoo. Ces entreprises doivent exécuter et exécuter quotidiennement des téraoctets et des pétaoctets de données pour déduire les demandes et les requêtes de leurs utilisateurs.
Les outils et applications existants et traditionnels deviennent insuffisants pour traiter une grande quantité de données. Hadoop a expliqué et répondu le problème de la manipulation et du traitement de ces téraoctets et pétaoctets de données [30] .

Architecture et composants

Les Composants de Hadoop Sont :

- Hadoop Common .
- Hadoop Distributed File System (HDFS).
- Hadoop Yarn .
- Hadoop MapReduce .

Système de fichiers distribués Hadoop (HDFS)

un système de fichiers distribué qui stocke les données sur les machines de base, fournissant une bande passante globale très élevée à travers le cluster.

HDFS stocke des données sur plusieurs machines. Les données sont automatiquement répliquées sur diverses machines pour éviter la perte de données. Dans HDFS, les données sont divisées en plusieurs blocs (afin d'être compatibles avec le stockage du matériel de base.); HDFS fournit un accès à haut débit aux blocs de données. Lorsque des données non structurées sont téléchargées sur HDFS, elles sont converties en blocs de données de taille fixe par défaut de 128 Mo.[28] HDFS fournit une interface limitée pour gérer le système de fichiers. Il garantit que l'on peut effectuer une augmentation ou une réduction des ressources dans le cluster Hadoop.

HDFS crée plusieurs répliques de chaque bloc de données et les stocke dans plusieurs systèmes à travers le cluster pour permettre un accès fiable et rapide aux données.

HDFS a deux composants qui s'exécutent sur différentes machines. Elles sont:

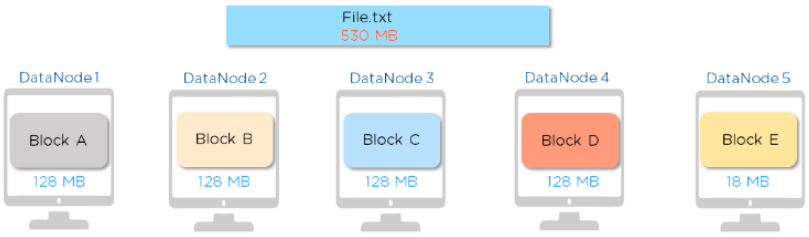


Figure 2.9: HDFS DataNode

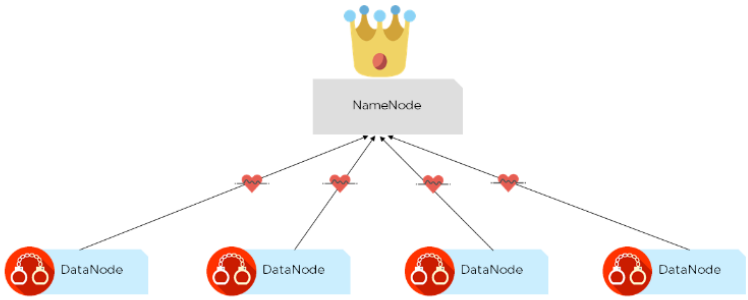


Figure 2.10: HDFS NameNode

- **NameNode** - NameNode est le maître de la couche de stockage HDFS. Il stocke toutes les métadonnées. Si la machine sur laquelle traite le NameNode tombe en panne, le cluster sera indisponible.

- **DataNode** - Les DataNodes sont appelés nœuds esclaves. Ils stockent les données réelles et effectuent les opérations de lecture / écriture. Fondamentalement, le NameNode gère tous les DataNodes. Les signaux appelés pulsations sont envoyés par les DataNodes au NameNode pour fournir des mises à jour de statut[28] .

Yarn :

une plate-forme de gestion des ressources chargée de gérer les ressources de calcul dans les clusters et de les utiliser pour la planification des applications des utilisateurs [18].
Apache YARN comprend:

- **Gestionnaire de ressources -** Il agit comme le démon maître. Il examine l'affectation du processeur, de la mémoire, etc.

- **Node Manager -** Il s'agit du démon esclave. Il signale l'utilisation au gestionnaire de ressources.

- **Maître d'application -** Cela fonctionne à la fois avec le gestionnaire de ressources et le gestionnaire de nœuds lors de la négociation des ressources [28].

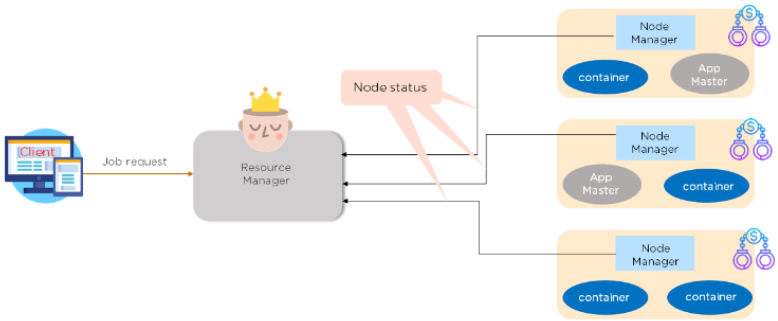


Figure 2.11: Hadoop Yarn

certaines des fonctionnalités de YARN:

- YARN est responsable du traitement des demandes d'emploi et de l'allocation des ressources .
- Différentes versions de MapReduce peuvent s'exécuter sur YARN, ce qui permet de gérer une mise à niveau de MapReduce .
- Selon nos besoins, on peut ajouter des nœuds à volonté [28] .

Hadoop MapReduce

un modèle de programmation pour le traitement de données à grande échelle Le composant MapReduce de Hadoop est responsable du traitement des travaux en mode distribué.
Certaines des principales fonctionnalités du composant Hadoop MapReduce sont les suivantes:

- Il effectue un traitement de données distribué à l'aide du paradigme de programmation MapReduce.
- Il vous permet de posséder une phase de mappage définie par l'utilisateur, qui est un traitement parallèle et sans partage des entrées.
- Il agrège la sortie de la phase de mappage, qui est une phase de réduction définie par l'utilisateur après un processus de mappage [28] .

À partir du diagramme suivant, c'est comment chaque étape est exécutée dans MapReduce [28] :

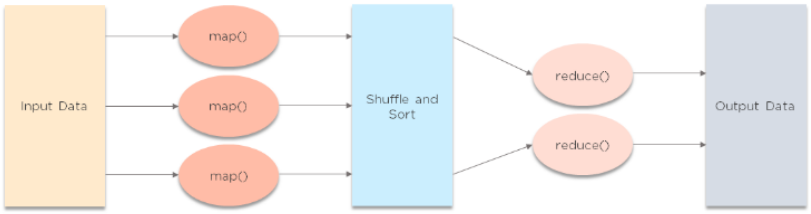


Figure 2.12: Hadoop MapReduce

Autre composants apache Hadoop

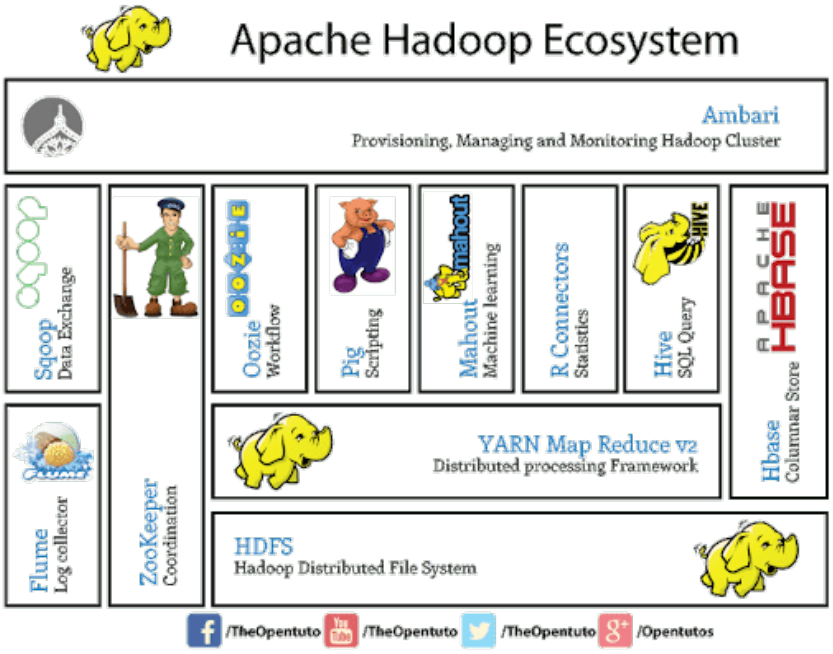


Figure 2.13: Composants de l'ecosysteme Hadoop

- **Hive**
- **HBase** :
  Apache HBase est une base de données distribuée, orientée colonne, construite sur HDFS.il peut évoluer horizontalement vers des milliers de serveurs de base et des péta octets de données en indexant le stockage.Apache HBase est une base de données non relationnelle open source, distribuée et versionnée, modelée d'après Bigtable de Google: un système

de stockage distribué pour les données structurées. HBase prend en charge les opérations CRUD aléatoires en temps réel (lues comme C-R-U-D). CRUD signifie créer, lire, mettre à jour et supprimer. L'objectif de HBase est d'héberger de très grandes tables avec des milliards de lignes et des millions de colonnes, au sommet de grappes de matériel de base.

- **ZooKeeper**:
  ZooKeeper est un service de coordination open source et hautes performances pour les applications distribuées. Il propose des services tels que: Naming (Appellation) Serrures et synchronisation Gestion de la configuration Services de groupe ZooKeeper fournit un noyau simple et performant pour créer des primitives de coordination plus complexes chez le client. Il fournit également des services de coordination distribués pour les applications distribuées. ZooKeeper suit FIFO, c'est-à-dire l'approche First-In-First-Out en ce qui concerne l'exécution des travaux. Il permet la synchronisation, la sérialisation et la coordination des nœuds dans un cluster Hadoop. Il est livré avec une architecture de pipeline pour réaliser une approche sans attente. ZooKeeper s'occupe des problèmes en utilisant des algorithmes intégrés pour la détection et la prévention des blocages. Il applique une approche multi-traitement pour réduire le temps d'attente pour l'exécution du processus.

- **Sqoop** :
  sqoop est un projet d'écosystème Apache Hadoop dont la responsabilité est d'importer ou d'exporter des opérations à travers des bases de données relationnelles telles que MySQL, MSSQL et Oracle vers HDFS. Il s'agit d'un outil conçu pour transférer des données de Hadoop vers un RDB et vice versa. Il transforme les données dans Hadoop à l'aide de MapReduce ou Hive sans codage supplémentaire [33] .

2.2.4 Apache Spark

Définition et utilisation

Apache Spark est un moteur de traitement parallèle de données open source permettant d'effectuer des analyses de grande envergure par le biais de machines en clusters. Codé en Scala, Spark permet notamment de traiter des données issues de référentiels de données comme Hadoop Distributed File System, les bases de données NoSQL, ou les data stores de données relationnels comme Apache Hive. Ce moteur prend également en charge le traitement In-memory, ce qui permet d'augmenter les performances des applications analytiques du Big Data. Il peut aussi être utilisé pour un traitement conventionnel sur disque, si les ensembles de données sont trop volumineux pour la mémoire système .

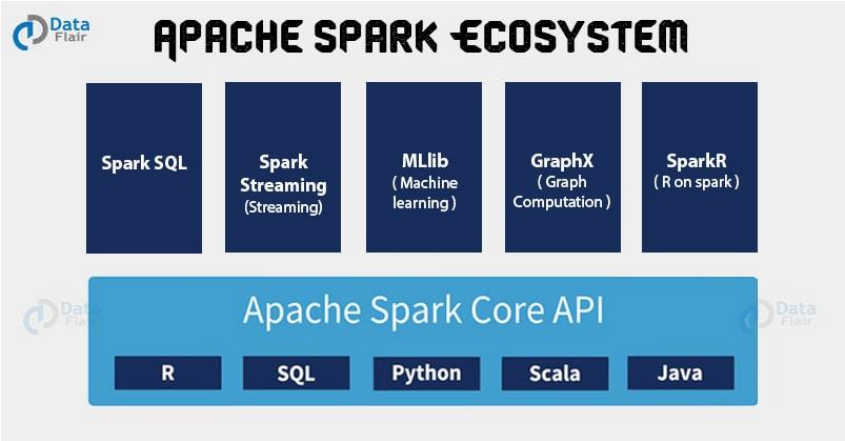


Figure 2.14: Composants de l'écosystème Spark

Spark a été introduit par Apache Software Foundation pour accélérer le processus du logiciel de calcul informatique Hadoop. Contrairement à une idée reçue, Spark n'est pas une version modifiée de Hadoop et ne dépend pas vraiment de Hadoop car il possède sa propre gestion de cluster. Hadoop n'est qu'un des moyens d'implémenter Spark.
Spark utilise Hadoop de deux manières: la première est le stockage et la seconde le traitement. Étant donné que Spark possède son propre calcul de gestion de cluster, il utilise Hadoop à des fins de stockage uniquement [38] .

Caractéristiques de apache Spark

1. **Traitement rapide :** En utilisant Apache Spark, nous atteignons une vitesse de traitement des données élevée environ 100 fois plus rapide en mémoire et 10 fois plus rapide sur le disque. Ceci est rendu possible en réduisant le nombre de lecture-écriture sur le disque.

2. **Dynamique dans la nature :** Nous pouvons facilement développer une application parallèle, car Spark fournit 80 opérateurs de haut niveau.

3. **Calcul en mémoire dans Spark :** Avec le traitement en mémoire, nous pouvons augmenter la vitesse de traitement. Ici, les données sont mises en cache, nous n'avons donc pas besoin de récupérer les données du disque à chaque fois, ce qui permet de gagner du temps. Spark dispose d'un moteur d'exécution DAG qui facilite le calcul en mémoire et le flux de données acyclique, ce qui se traduit par une vitesse élevée.

4. **Tolérance aux pannes dans Spark :** Apache Spark offre une tolérance aux pannes via Spark abstraction-RDD. Les RDD Spark sont conçus pour gérer la défaillance de tout nœud de travail dans le cluster. Ainsi, il garantit que la perte de données est réduite à zéro. Découvrez différentes façons de créer un RDD dans Apache Spark.

5. **Traitement de flux en temps réel :** Spark a une disposition pour le traitement de flux en temps réel. Auparavant, le problème avec Hadoop MapReduce était qu'il peut gérer et traiter les données qui sont déjà présentes, mais pas les données en temps réel. mais avec Spark Streaming, nous pouvons résoudre ce problème.

6. **Prise en charge de plusieurs langues :** Dans Spark, il existe un support pour plusieurs langages comme Java, R, Scala, Python. Ainsi, il fournit de la dynamicité et surmonte la limitation de Hadoop qu'il ne peut créer des applications qu'en Java.

7. **Prise en charge d'une analyse sophistiquée :** Spark est livré avec des outils dédiés pour la diffusion de données en continu, des requêtes interactives / déclaratives, un apprentissage automatique qui s'ajoutent pour mapper et réduire [49].

Figure 2.15: Caractéristiques de Apache Spark

2.2.5 Apache Hive

Définition

Hive est défini comme un système d'entrepôt de données pour Hadoop qui facilite les requêtes ad hoc et l'analyse de grands ensembles de données stockés dans Hadoop. Hive est un système de gestion et d'interrogation de données non structurées dans un format structuré. Il utilise le concept de MapReduce pour l'exécution de ses scripts et le système de fichiers distribués Hadoop ou HDFS pour le stockage et la récupération des données. Les performances sont meilleures dans Hive puisque le moteur Hive utilise le meilleur script intégré pour réduire le temps d'exécution, permettant ainsi une sortie élevée en moins de temps [33].

Construit sur Apache Hadoop ™, Hive fournit les fonctionnalités suivantes:

- Outils pour permettre un accès facile aux données via SQL, permettant ainsi des tâches d'entreposage de données telles que l'extraction / transformation / chargement (ETL), la création de rapports et l'analyse des données.
- Un mécanisme pour imposer une structure à une variété de formats de données Accès aux fichiers stockés directement dans Apache HDFS ™ ou dans d'autres systèmes de stockage de données tels que Apache HBase ™ .
- Exécution de requêtes via Apache Tez ™, Apache Spark ™ ou MapReduce Langage procédural avec HPL-SQL.
- Récupération de requête en sous-seconde via Hive LLAP, Apache YARN et Apache Slider.

Architecture du système

Dans cette partie, nous présentons brièvement l'architecture de Hive. La figure illustre les principaux composants du système [51].

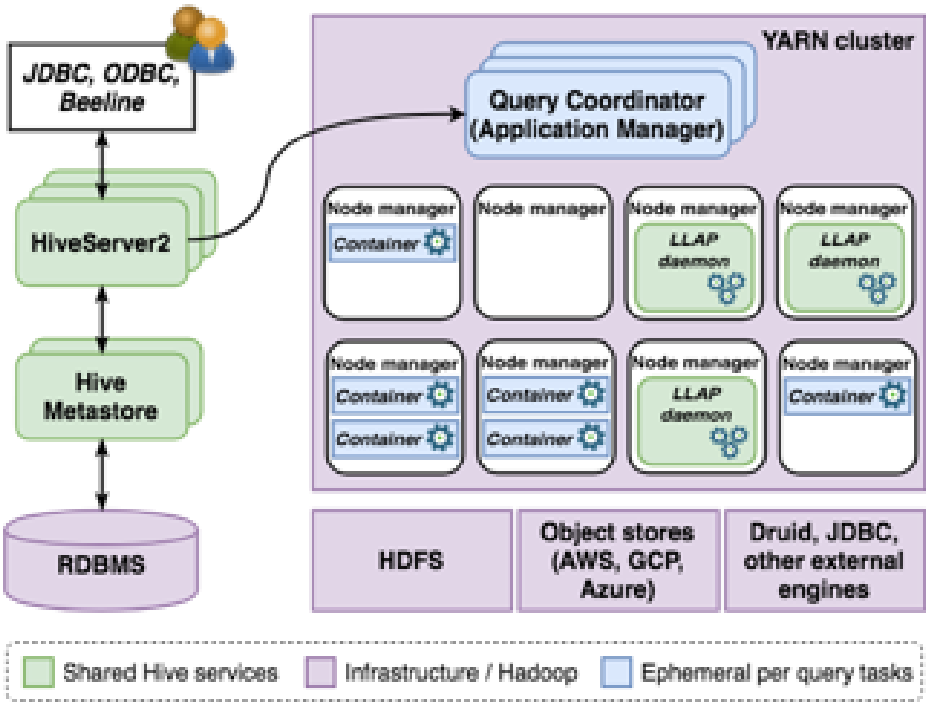


Figure 2.16: Architecture de apache Hive

Stockage des données

Les données dans Hive peuvent être stockées à l'aide de l'un des formats de fichiers pris en charge dans n'importe quel système de fichiers compatible avec Hadoop. À ce jour, les formats de fichiers les plus courants sont ORC [44] et Parquet [45]. À leur tour, les systèmes de fichiers compatibles incluent HDFS, qui est l'implémentation de système de fichiers distribué la plus couramment utilisée, et tous les principaux magasins d'objets cloud commerciaux tels que AWS S3 et Azure Blob Storage. En outre, Hive peut également lire et écrire des données sur d'autres systèmes de traitement autonomes, tels que Druid [42] ou HBase [43].

Cataloque des données

Hive stocke toutes les informations sur ses sources de données à l'aide du Hive Metastore (ou HMS, en bref). En un mot, HMS est un catalogue de toutes les données interrogeables par Hive. Il utilise un SGBDR pour conserver les informations et s'appuie sur DataNucleus [30], une implémentation de mappage objet-relationnel Java, pour simplifier la prise en charge de plusieurs SGBDR au niveau du backend. Pour les appels qui nécessitent une faible latence, HMS peut contourner DataNucleus et interroger directement le RDMBS. L'API HMS prend en charge plusieurs langages de programmation et le service est implémenté à l'aide de Thrift [46],

un framework logiciel qui fournit un langage de définition d'interface, un moteur de génération de code et une implémentation de protocole de communication binaire.

Runtime de traitement de données échangeables

Hive est devenu l'un des moteurs SQL les plus populaires sur Hadoop et s'est progressivement éloigné de MapReduce pour prendre en charge des temps d'exécution de traitement plus flexibles compatibles avec YARN [50]. Bien que MapReduce soit toujours pris en charge, actuellement le runtime le plus populaire pour Hive est Tez .
Tez offre plus de flexibilité que MapReduce en modélisant le traitement des données sous forme de DAG avec des sommets représentant la logique d'application et des arêtes représentant le transfert de données.

Serveur de requêtes

HiveServer2 abrégé HS2 permet aux utilisateurs d'exécuter des requêtes SQL dans Hive. HS2 prend en charge les connexions JDBC et ODBC locales et distantes; La distribution Hive comprend un client léger JDBC appelé Beeline.

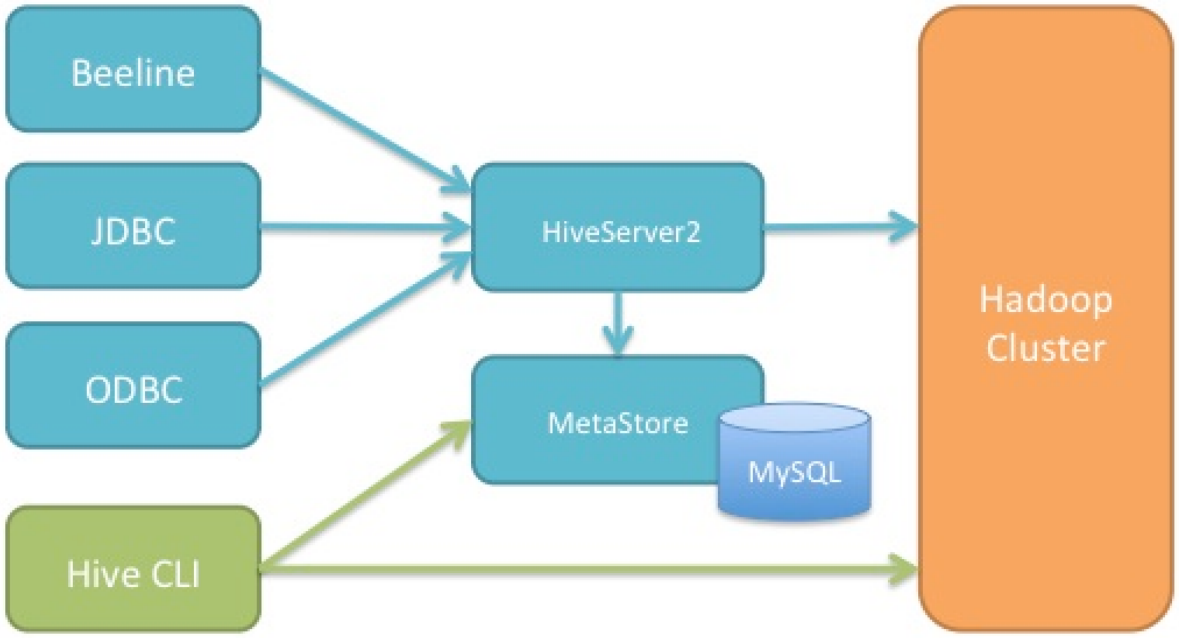


Figure 2.17: Hive Server

Support SQL et ACIDE

Prise en charge SQL

Pour remplacer les entrepôts de données traditionnels, Hive devait être étendu pour prendre en charge davantage de fonctionnalités du SQL standard. Hive utilise un modèle de données imbriqué prenant en charge tous les principaux types de données SQL atomiques ainsi que les types non atomiques tels que STRUCT, ARRAY et MAP. En outre, chaque nouvelle version de Hive a augmenté sa prise en charge des constructions importantes qui font partie de la spécification SQL. Par exemple, il existe une prise en charge étendue des sous-requêtes corrélées, c'est-à-dire des sous-requêtes qui référencent des colonnes de la requête externe, des opérations OLAP avancées telles que le regroupement d'ensembles ou de fonctions de fenêtre, des opérations d'ensemble et des contraintes d'intégrité, entre autres. D'autre part, Hive a conservé plusieurs fonctionnalités de son langage de requête d'origine qui étaient précieuses pour sa base d'utilisateurs. L'une des fonctionnalités les plus populaires est de pouvoir spécifier la disposition du stockage physique au moment de la création de la table à l'aide d'une clause PARTITIONED BY columns. En un mot, la clause permet à un utilisateur de partitionner une table horizontalement. Ensuite, Hive stocke les données pour chaque ensemble de valeurs de partition dans un répertoire différent du système de fichiers. Pour illustrer l'idée, considérez la définition du tableau suivant et la disposition physique correspondante illustrée à la figure 2.18
.

```
1 CREATE TABLE store_sales (
2 sold_date_sk INT, item_sk INT, customer_sk INT, store_sk INT,
3 quantity INT, list_price DECIMAL(7,2), sales_price DECIMAL(7 , 2)
4 ) PARTITIONED BY (sold_date_sk INT) ;
```

L'avantage de l'utilisation de la clause PARTITIONED BY est que Hive pourra ignorer facilement l'analyse des partitions complètes pour les requêtes qui filtrent sur ces valeurs.

Mise en oeuvre d'ACIDE

Au départ, Hive ne prenait en charge que l'insertion et la suppression de partitions complètes d'une table [55]. Bien que le manque d'opérations au niveau des lignes était acceptable pour les charges de travail ETL, à mesure que Hive évoluait pour prendre en charge de nombreuses charges de travail d'entreposage de données traditionnelles, il y avait un besoin croissant de prise en charge complète de DML et de transactions ACID. Par conséquent, Hive inclut désormais

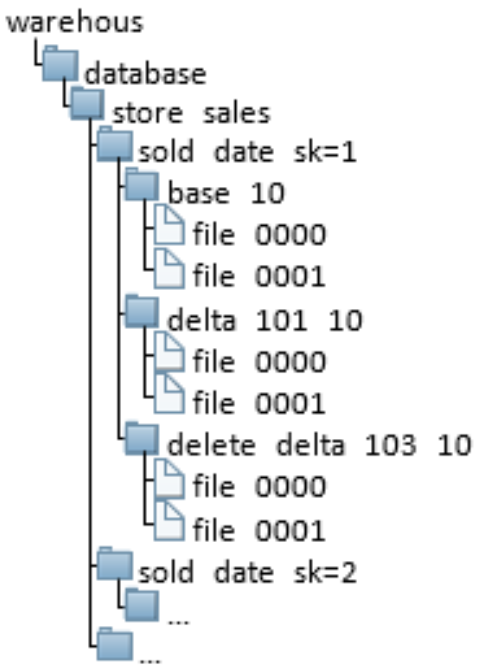


Figure 2.18: Disposition physique pour la table partitionnée.

la prise en charge de l'exécution des instructions INSERT, UPDATE, DELETE et MERGE. Il fournit des garanties ACID via Snapshot Isolation [8] sur la lecture et une sémantique bien définie en cas d'échec à l'aide d'un gestionnaire de transactions construit au-dessus du HMS. Cependant, il est possible d'écrire dans plusieurs tables au sein d'une même transaction à l'aide des instructions multi-insert Hive [10].

2.2.6 Les bases de données NoSQL

Définition de NoSql

Le terme NoSQL peut véhiculer deux connotations différentes - l'une impliquant que le système de gestion des données n'est pas conforme à SQL, tandis que l'implication la plus acceptée est que le terme signifie non seulement SQL, suggérant des environnements qui combinent SQL traditionnel (ou Langages de requête de type SQL) avec d'autres moyens d'interrogation et d'accès.[14] ces bases non relationnelles permettent de gérer de gros volumes de données hétérogènes sur un ensemble de serveurs de stockage distribués grâce à leur flexibilité et leur souplesse.

Fonctionnement

les concepts généraux de NoSQL incluent la modélisation sans schéma dans laquelle la sémantique des données est intégrée dans un modèle de connectivité et de stockage flexible; cela permet une distribution automatique des données et une élasticité par rapport à l'utilisation de l'informatique, du stockage et de la bande passante du réseau de manière à ne pas forcer la liaison spécifique des données à être stockées de manière persistante dans des emplacements physiques particuliers. Les bases de données NoSQL prévoient également une mise en cache des données intégrée qui aide à réduire la latence d'accès aux données et à accélérer les performances. Les base de données NoSQL reposent sur les schéma dynamiques (pas de schéma prédéfinies) et favorisent le théorème CAP (Consistency, Availability, Partition) vis-que les SGBDR reposent sur les modèles relationnelles et visent les propriétés ACID (Atomicity, Consistency, Isolation, Durability)[9].

Les types de bases de données NoSQL

Le NoSQL regroupe 4 grandes familles de base de données qui permettent d'offrir une représentation différentes des données, chacune dispose d'avantages et d'inconvénients en fonction du contexte dans lequel on souhaite l'utiliser., en voici quelques uns avec le nom des solutions ou moteurs associés :

- **Orientées colonnes** : HBase et BigTable .
- **Orientées graphes** : Neo4J, OrientDB, TITAN.
- **Orientées clé-valeur** : Voldemort , Redis, Amazon DynamoDB et Riak.
- **Orientées document** : CouchDB, MongoDB, RavneDB et Cassandra.

Chapter 3

État de l'art

Il existe plusieurs techniques pour obtenir de meilleures performances de requête et réduire le temps de réponse , comme le : partitionnement la table de base, indexation sur les colonnes requises, matérialisation et création (des vus).

3.1 Les techniques de partitionnement

Les techniques de partitionnement des données sont l'un des facteurs critiques de succès dans la conception de bases de données.
En tirant parti du théorème 'Divide Conquer', le Big Data peut être partitionné en unités logiques distinctes, en modifiant la relation des métadonnées, chaque grande unité de jeu de données peut se déplacer 'roam around' rapidement autour de différentes tables sous des exigences changeantes, ce qui augmente vraiment les performances et la flexibilité d'utilisation de la possibilité d'application à l'ère de l'explosion des données / informations.

3.1.1 Définition et principe

Le partitionnement consiste à diviser les tables en tables plus petites et plus faciles à gérer, appelées partitions.
Un partitionnement s'effectue toujours en fonction d'une clé, soit un ou plusieurs attributs dont la valeur sert de critère à l'affectation d'un document à un fragment. La première décision à prendre est donc le choix de la clé.
Un bon partitionnement répartit les documents en fragments de tailles comparables. Cela suppose que la clé soit suffisamment discriminante pour permettre de diviser la collection avec une granularité très fine (si possible au niveau du document lui-même).
Un partitionnement doit être dynamique: en fonction de l'évolution de la taille de fragments doit pouvoir évoluer. C'est important pour optimiser l'utilisation de l'espace disponible et

obtenir les meilleures performances. C'est aussi, techniquement, la propriété la plus difficile à satisfaire [54] .

3.1.2 Types de partitionnement

La plupart des systèmes NoSQL utilisent soit partitionnement basé sur le hachage, soit sur plage (ou un mélange des deux) :

Partitionnement basée sur plages

Dans le partitionnement basé sur des plages, l'espace de clés est divisé en plages et chaque plage est attribuée à un serveur et potentiellement répliquée sur d'autres. Le principal avantage du partitionnement par plage est que deux clés consécutives sont susceptibles d'apparaître dans la même partition, ce qui est avantageux lorsque les requêtes de type scan de plage sont fréquentes. Les schémas de partitionnement basés sur des plages maintiennent généralement une carte qui stocke des informations sur les serveurs responsables de quelles plages de clés [16] .

Certaines bases de données NoSQL comme Apache H.Base , MongoDB utilisent le partitionnement par plage .

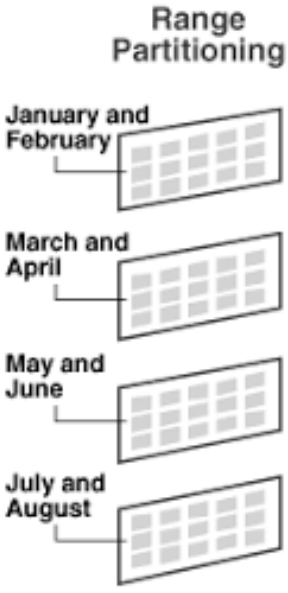


Figure 3.1: Partitionnement par plages

Partitionnement basée sur le hachage

Le partitionnement basé sur le hachage utilise simplement le hachage des données pour déterminer le serveur responsable du stockage de ces données. d'autres bases de données NoSQL comme Google Big Table, Amazon Dynamo, Cassandra utilisées par Facebook utilisent le partitionnement par hachage.

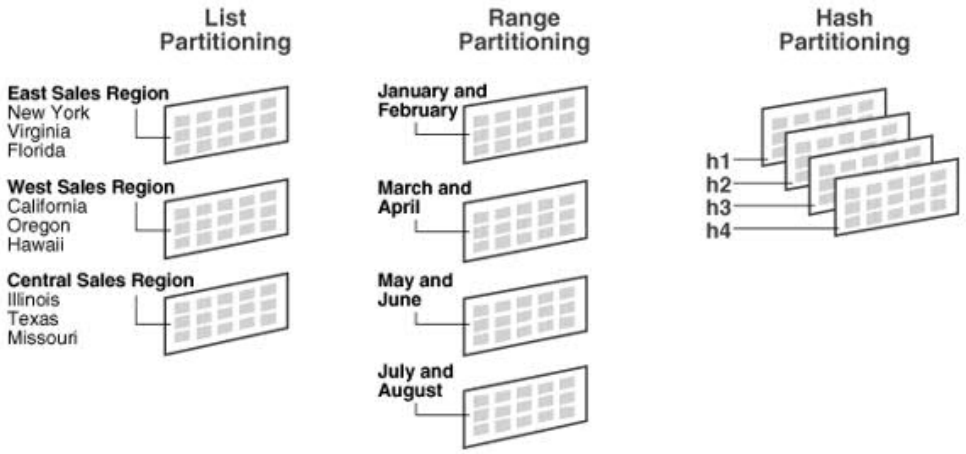


Figure 3.2: Partitionnement par hachage

Les anneaux de hachage cohérents sont un mélange de schémas de partitionnement basés sur la plage et le hachage et de nombreux systèmes NoSQL tels que Cassandra, Dynamo, Voldemort et Riak adoptent ce schéma [54] .

3.1.3 Travaux connexes

- **Travaux de Ata Turk, R. Oguz Selvitopi, Hakan Ferhatosmanoglu, et Cevdet Aykanat 2014** [16] (Partitionnement répliqué tenant compte de la charge de travail temporelle pour les réseaux sociaux):
  (Ata Turk et al. a proposé une méthode de partitionnement des données basées sur l'hypergraphe qui est construit pour modéliser correctement les opérations multi-utilisateurs. Cette méthode utilise les informations temporelles des charges de travail précédentes pour essayer d'améliorer Cassandra NoSQL)
  Dans ce travail, ils ont proposé une méthode de partitionnement et de réplication sélective pour la distribution des données dans les réseaux sociaux en utilisant les informations de charge de travail et de temps.
  Cette méthode utilise un nouveau modèle hypergraphique (appelé modèle hypergraphique

d'activité temporelle) pour représenter la structure du réseau social et les interactions entre ses utilisateurs. Ce modèle évalue le temps des interactions entre les utilisateurs et prédit les interactions susceptibles de se produire dans un proche avenir.
Ils ont montré que le partitionnement et la réplication simultanés (partitionnement répliqué) de ce modèle hypergraphique peuvent capturer avec précision l'objectif de réduction de la durée des requêtes multi-utilisateurs, sous réserve d'équilibrage de charge et de contraintes de réplication.
Après avoir effectué un partitionnement répliqué de ce modèle hypergraphique, ils ont décodé le résultat obtenu sous la forme d'un mappage donné-serveur. Ce schéma réduit considérablement la durée moyenne des requêtes tout en équilibrant les charges du serveur. Il limite également l'augmentation de la charge d'E/S due aux réplications en respectant un seuil fourni par l'utilisateur sur la quantité de réplication et en effectuant une réplication sélective.

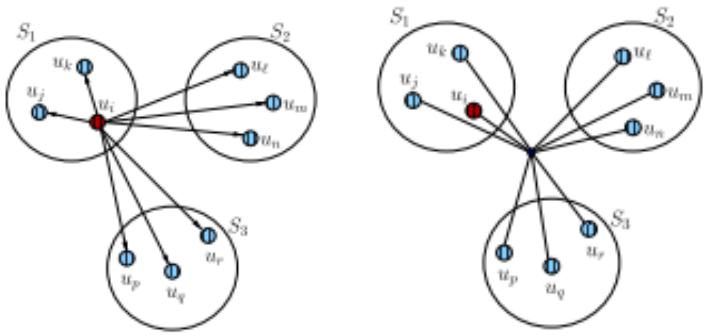


Figure 3.3: (a) modèle graphique (b) modèle hypergraphique

- **Travaux de TE-YUAN LIN, CHIOU-SHANN FUH 2015** [29]
(Technologie de partitionnement et mouvements rapides de contenu de Big Data)
Dans ce travail, il a développé une stratégie pour faire glisser la structure de partitionnement en supprimant la partition la plus ancienne et en étendant une nouvelle partition supplémentaire, pendant le processus, non seulement en actualisant la structure, mais également en supprimant les données historiques et en chargeant les dernières données en quelques secondes . Pour automatiser davantage ce processus, il suffit de mettre ces étapes dans les travaux de planification avec une fréquence appropriée, cette idée est totalement différente du SQL DML traditionnel (Data Manipulation Langage) et extrêmement efficace pour gérer le Big Data pour toute application de niveau entreprise.
Le partitionnement a également ses inconvénients et ses limites. Le cadre a la forte dépendance du moteur de base de données. Le même scénario avec de bonnes perfor-

mances et une bonne faisabilité dans une plate-forme de base de données peuvent ne pas bien fonctionner dans les autres en raison de la prise en charge de la version et des différences de comportement du moteur interne. Des exigences de conception un peu complexes avant de se lancer dans un voyage sont également le showstopper largement utilisé dans les opérations de données quotidiennes. Cependant, il a encore tiré plus d'avantages de l'utilisation de l'idée que de ses limites. Au fur et à mesure de la maturité du cadre, la véritable influence du partitionnement vaut la peine d'être attendue .

- **Travaux de Sally M. Elghamrawy • Aboul Ella Hassanien2** [39]
  (À partitionner framework for Cassandra NoSQL database using rendez-vous hashing)
  Dans ce travail, un framework MRRHVH (Virtual hierarchical) basé sur le hachage MapReduce Rendez-vous est proposé pour le partitionnement de la base de données Cassandra NoSQL. Son objectif principal est d'améliorer les performances de partitionnement de Cassandra en utilisant le hachage Rendez-vous qui distribue uniformément les enregistrements de la base de données sur les nœuds, contrairement au hachage cohérent.L'idée de base du hachage Rendez-vous basé sur l'algorithme du poids aléatoire le plus élevé (HRW), proposé pour la première fois par Yao Z and Ravishankar [3], est de donner à chaque nœud un poids pour chaque clé et d'affecter la clé au poids le plus élevé.
  De plus, un algorithme de hachage Rendez-vous basé sur un algorithme d'équilibrage de charge LBRH est proposé pour gérer l'équilibrage entre les nœuds dans le processus de partitionnement. MR-RHVH améliore la synchronisation du hachage en utilisant un évaluateur de filtre de floraison dans chaque nœud de la plage de hachage de rendez-vous. De plus, l'utilisation d'une fonction de hachage effrayante améliore la synchronisation du hachage.

D'autres travaux connexes : les chercheurs s'intéressent énormément aux stratégies de partitionnement des bases de données NoSQL en raison de leur impact sur les performances des systèmes. Les partitionnements de hachage, de plage et les hybrides de hachage entre le partitionnement de hachage et de plage sont les stratégies les plus couramment utilisées par les systèmes No sql.
Un certain nombre de chercheurs ont développé différentes méthodes pour améliorer les performances de Cassandra et dans les stratégies de partitionnement de différentes bases de données No Sql:

- **Abramova et coll**.[17] ont analysé l'évolutivité de Cassandra en testant les stratégies de réplication et de partitionnement des données utilisées

- **Lakshminarayanan**. [31] a proposé un schéma de partitionnement adaptatif pour un hachage cohérent qui a un effet sur l'hétérogénéité des systèmes .
- **Wang et Loguinov**.[6] ont proposé un algorithme glouton pour positionner le nouveau nœud dans la plus grande plage de l'espace en divisant la plage en deux parties.
- **Ramakrishnan et coll** [34]. ont proposé un pipeline de traitement, utilisant le partitionneur aléatoire de Cassandra, pour permettre à tout exécutable non Java d'utiliser le NoSQL et permettant le traitement hors ligne pour Cassandra. Kuhlenkamp et coll.
- **Zhikun et coll**. [12] [11] ont proposé une stratégie de partition HRCH (Hybrid range Consistency Hash) pour la base de données NoSQL afin d'améliorer le degré de traitement et la vitesse de chargement des données. La plupart de ces approches envisagent un test d'évolutivité de la base de données NoSQL ou une amélioration du hachage cohérent de Cassandra.

3.1.4 Synthèse

Le tableau résume les différentes approches basées le partitionnement qu'on a déjà vu dans les travaux connexes .IL présente les avantages de chaque approche et sur quel principe elles sont basées, ainsi que les futurs travails concernant chacune d'elles..

L'approche	**Basée sur**	**Ces avantages**	**Travail Future**
Ata turk et all Partitionnement répliqué tenant compte de la charge de travail temporelle pour les réseaux sociaux	modèle hypergra-phique d'activité temporelle	-réduit considérablement la durée moyenne des requêtes tout en équilibrant les charges du serveur. - Il limite également l'augmentation de la charge d'E / S due aux réplications en respectant un seuil fourni par l'utilisateur sur la quantité de réplication et en effectuant une réplication sélective.	l'étude des mécanismes de repartitionnement qui évitent la migration des éléments de données dans les itérations de partitionnement ultérieurs et l'ajout de mécanismes pour plus de performance .
Te yuan Technologie de partitionnement et mouvements rapides de contenu de Big Data	Partionne-ment vertical et horizontal	pendant le processus, non seulement en actualisant la structure, mais également en supprimant les données historiques et en chargeant les dernières données en quelques secondes	
Sally M. Elghamrawy MR-RHVH : Plateforme de partionnement pour casandra no sql	le hachage MapReduce Rendez-vous	-améliore la synchronisation du hachage en utilisant un évaluateur de filtre de floraison dans chaque nœud de la plage de hachage de rendez-vous -très efficace en matière de débit et de latence par rapport aux systèmes récents - MR-RHVH divise le système en région de rendez-vous, ce qui permet aux nœuds d'être plus flexibles dans la sélection des nœuds maîtres, tout en conservant l'avantage de la localité dans le traitement des nœuds dans la même région.	vise à réduire la surcharge de MR-RHVH et à tester son évolutivité lors de l'augmentation du nombre de nœuds. De plus, le hachage cohérent de saut sera implémenté en tant que technique de hachage dans MR RHVH et remplacera le Hadoop par le Spark.

Table 3.1: Caractéristiques des approches basées sur le partitionnement

3.2 Les techniques de sélection d'index

les techniques d'indexation jouent un rôle majeur pour accéder et traiter les requêtes plus rapidement . L'indexation est un moyen d'optimiser les performances d'une base de données en minimisant le nombre d'accès au disque requis lorsqu'une requête est traitée[35].

3.2.1 Définition et principe

les index peuvent être considérés comme une liste de balises, de noms, de sujets, etc. d'un ensemble de données qui référence où les données peuvent être trouvées.
Une stratégie d'indexation est la conception d'une méthode d'accès à un élément recherché, ou simplement, un index. Il décrit également comment les données sont organisées dans un système de stockage pour faciliter la recherche d'informations.
L'idée de l'indexation Big Data est de fragmenter les ensembles de données selon des critères qui seront fréquemment utilisés dans la requête . Les fragments sont indexés avec chaque valeur contenant satisfaisant certains prédicats de requête. Cela vise à stocker les données de manière plus organisée, facilitant ainsi la récupération d'informations [26].
Les dernières recherches sur les techniques d'indexation suggèrent des moyens d'améliorer les performances de précision de l'indexation afin que la qualité de l'indexation ne se détériore pas. Ces techniques servent à optimiser les performances de recherche dans le Big Data avec un meilleur compromis entre l'indice espace-temps [35] .

3.2.2 Types d'indexation

les stratégies d'indexation peuvent être classées en approche d'intelligence artificielle (IA) et en approche d'intelligence non artificielle (NAI) [26] .

Approche d'intelligence artificielle

Les approches d'indexation de l'intelligence artificielle (IA) sont ainsi appelées en raison de leur capacité à détecter des comportements inconnus dans le Big Data. Ils établissent des relations entre les éléments de données en observant des modèles et en catégorisant des éléments ou des objets ayant des caractéristiques similaires. Bien que cela donne aux approches d'indexation IA un avantage sur NAI, les premières prennent généralement plus de temps dans la recherche d'informations et sont parfois considérées comme inefficaces par rapport aux approches d'indexation NAI [35].

L'indexation sémantique latente (LSI) et le modèle de Markov caché (HMM) sont deux approches d'indexation d'IA populaires :

a- Indexation sémantique latente

L'indexation sémantique latente, en abrégé LSI, est une stratégie d'indexation (méthode de récupération / d'accès)((retrieval/access method) qui identifie les modèles(paternes) entre les termes d'un ensemble de données non structuré (en particulier, du texte).
Il utilise une approche mathématique connue sous le nom de décomposition en valeurs singulières (SVD) pour l'identification du modèle ou de la relation.
La principale caractéristique de LSI est la capacité à obtenir le contenu conceptuel (sémantique) des ensembles de données et à établir des relations entre des termes avec des contextes similaires.

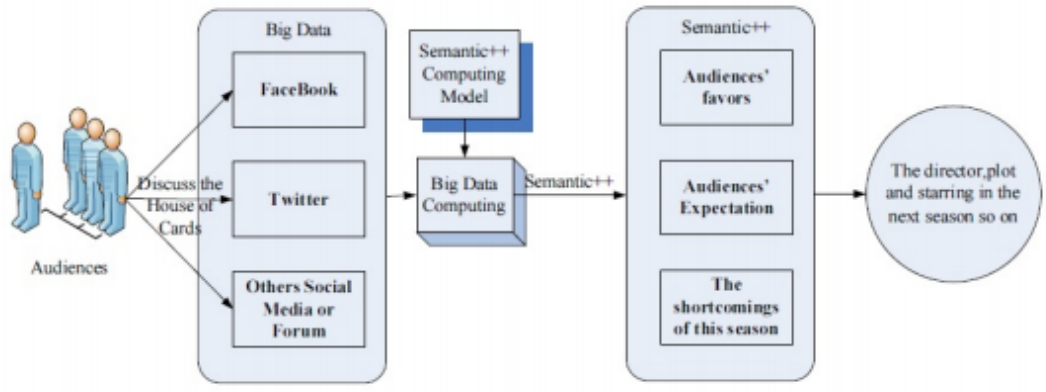


Figure 3.4: Indexation sémantique latente LSI

Dans la figure, le public discute du programme House of cards sur le social médias et forums. LSI est utilisé ici pour catégoriser ou indexer les commentaires du public en audience favors, audience expectations, and the shortcomings of the season . Cela permet au réalisateur de prendre plus facilement des décisions en vue de l'amélioration de la saison suivante, etc.
LSI utilise Ressource Description Framework (RDF), qui est une norme pour la description des ressources Web.
Dans la stratégie LSI, le texte ou les documents sont affectés à des catégories en fonction de leurs similitudes contextuelles. Lors de la catégorisation, les contextes de l'ensemble de texte à catégoriser sont comparés aux contextes des exemples de documents, et les catégories sont attribuées en fonction des documents correspondants.
Les principaux défis rencontrés lors de l'utilisation de LSI sont l'évolutivité et les performances

. La stratégie LSI exige des performances de calcul très élevées ainsi que de la mémoire pour indexer le Big Data. LSI prend en charge les requêtes par mots-clés sur des données textuelles qui peuvent être sous forme de contenu Web (images, audio, etc.), de documents, d'e-mails ou de tout élément pouvant être converti en texte [26].

b- indexation du modèle de Markov caché

L'approche d'indexation du modèle de Markov caché (HMM) est une méthode d'accès développée à partir du modèle de Markov. Un modèle de Markov est composé d'états reliés par des transitions, où les états futurs sont uniquement dépendants de l'état actuel et indépendants des états historiques.
Semblable à la stratégie LSI, le HMM utilise la reconnaissance de formes et la relation entre les données. Dans l'approche d'indexation HMM, les données ou caractéristiques dont les états dépendent la requête sont classées et stockées à l'avance. Les résultats de la requête sont généralement des prédictions des états futurs d'un élément, en fonction de l'état actuel . L'état actuel est utilisé pour prédire les états futurs en utilisant les données dépendantes ou les caractéristiques des états.
Par exemple, dans l'étude de Matsui et al [24], HMM a été utilisé pour classer et stocker les données de mouvement utilisées par les robots. La classification était basée sur les informations d'accélération, constituées des informations de position et de la force pure. Par conséquent, la prédiction de la prochaine série (séquence) de mouvements dépendait de l'informateur de position et de la force pure. Les données de mouvement sont stockées et classées à l'avance, avant que la recherche rapide de mouvement ne soit effectuée.
L'étude de Widodo et al. [25] ont utilisé HMM et LSI pour classer ou indexer des documents. Dans leur travail, les mots sont développés dans chaque document et stockés à l'avance. Le développement des documents avant l'indexation donne plus de place pour la prédiction et une recherche plus rapide sur les éléments [26] .

Approche d'intelligence non artificielle

Dans l'approche d'indexation NAI, la formation d'index ne dépend pas de la signification de la donnée ou de la relation entre les textes. Au contraire, les index sont formés en fonction des éléments les plus interrogés ou recherchés dans un ensemble de données particulier. La stratégie d'indexation arborescente (B-tree , R-tree et X-tree), approche d'indexation inversée , le hachage et l'indexation personnalisée (GiST [et GIN) sont des approches d'indexation NAI .

a- Les stratégies d'indexation arborescentes

Les structures d'indexation Tree sont le B-tree, R-tree, Xtree, etc. . Dans la stratégie d'indexation d'arbre, la récupération des données est effectuée dans un ordre trié, suivant les relations de branche de l'élément de données. Cela satisfait les requêtes des voisins les plus proches .
Les stratégies d'indexation par arborescence sont :

- **B-tree**
  Un B-tree fonctionne comme la recherche d'arbre binaire, mais d'une manière plus complexe. En effet, les nœuds de B-tree ont de nombreuses branches, contrairement à l'arbre binaire qui a deux branches par nœud.
  Les index B-tree satisfont les requêtes de plage et les requêtes de similarité également connues sous le nom de recherche du voisin le plus proche (NNS) Nearest Neighbor Search , en utilisant des opérateurs de comparaison (¡, ¡=, =,¿,¿ =). Dans l'arborescence B, les clés et tous les enregistrements sont normalement stockés dans des feuilles, mais des copies (de la clé) sont stockées dans des nœuds internes comme illustrés à la figure 2. En outre, les feuilles peuvent inclure des pointeurs vers le nœud suivant, montrant le chemin à l'élément recherché.
  D'autres variantes de l'arbre B sont l'arbre B + , l'arbre B *, l'arbre KDB , etc.

- **R-tree**
  il s'agit d'une stratégie d'indexation utilisée pour les requêtes spatiales ou de plage. Il est principalement appliqué dans les systèmes géospatiaux avec chaque entrée ayant des coordonnées X et Y avec des valeurs minimales et maximales . L'avantage d'utiliser un R-tree sur un B-tree est que, le R-tree satisfait les requêtes multidimensionnelles ou de plage, alors que le B-tree ne le fait pas. Étant donné une plage de requêtes, l'utilisation de l'arborescence R permet de trouver rapidement des réponses aux requêtes .
  Un exemple est de trouver toutes les auberges dans un campus, ou trouver tous les hôtels

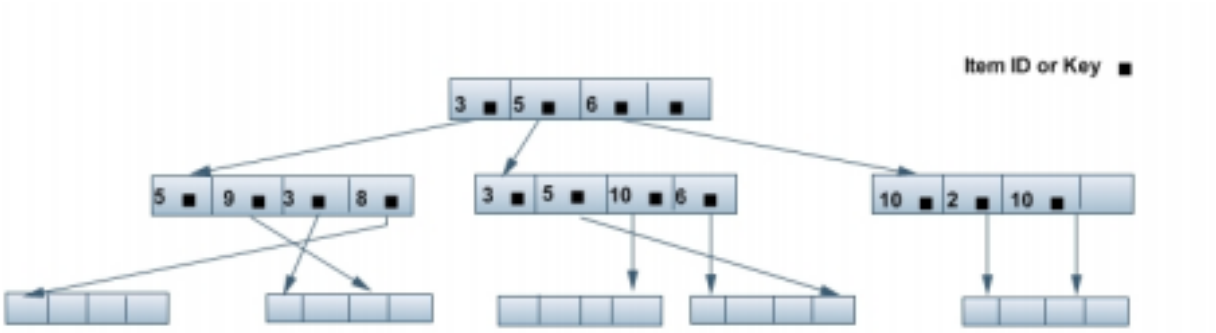


Figure 3.5: B-TREE

dans un kilomètre donné à partir d'un certain emplacement. L'idée est de regrouper les éléments de données en fonction de leur distance les uns des autres et de leur attribuer des limites minimales et maximales. Chaque enregistrement au nœud feuille décrit un seul élément (avec des valeurs minimales et maximales). Chaque nœud interne décrit une collection d'éléments ou d'objets comme illustré dans les deux figures :

Figure 3.6: R-TREE

- **X-tree**
  Ce type de stratégie d'indexation, basé sur l'arbre R, satisfait les requêtes de plage. L'arbre X est similaire à l'arbre R et fonctionne exactement comme l'arbre R. Bien que, contrairement à l'arbre R qui satisfait les requêtes de plage de 2 à 3 dimensions, l'arbre X satisfait les requêtes de nombreuses dimensions. Cela implique que le X-tree est une version plus compliquée du R-tree. L'avantage de l'arbre X par rapport à l'arbre R est qu'il couvre plus de dimensions, sinon, l'arbre X consomme également de l'espace mémoire en raison du stockage des coordonnées.

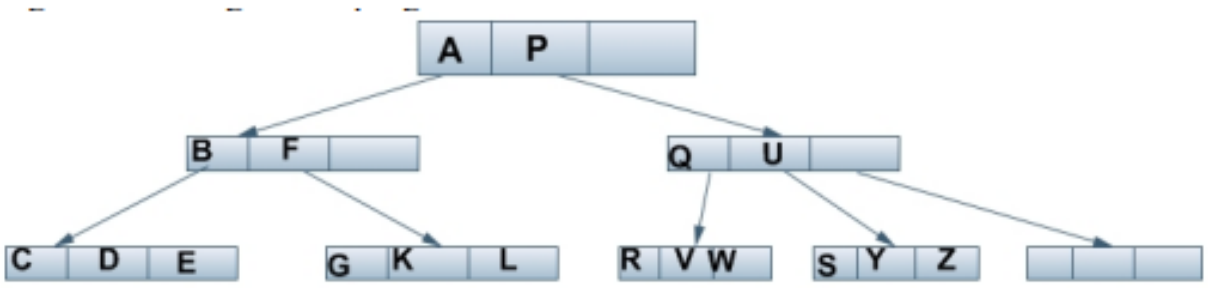


Figure 3.7: R-TREE

b- Stratégie d'indexation de hachage

Le hachage permet une comparaison d'égalité. L'indexation par hachage accélère la recherche d'informations en détectant les doublons dans un grand ensemble de données .
Un exemple de mise en œuvre d'une stratégie d'indexation par hachage est expliqué dans l'étude de Giangreco et al (2014)[20]. Giangreco et al ont conçu une stratégie qui prend des diagrammes esquissés à la main et récupère des images similaires à partir d'une grande collection d'images, basée sur une stratégie d'indexation par hachage.
La stratégie d'indexation de hachage est utilisée dans les systèmes de vérification de mot de passe, la correspondance de séquence ADN, etc.
Le hachage est utilisé dans l'indexation Big Data pour indexer et récupérer des éléments de données (dans un ensemble de données) qui sont similaires à l'élément recherché. Il utilise une clé hachée (qui est calculée par la fonction de hachage et généralement plus courte que la valeur d'origine) pour stocker et récupérer les index. Pour cette raison, l'indexation par hachage est plus efficace que l'indexation arborescente en matière d'égalité ou de requête ponctuelle . Simplement, la recherche sur des clés hachées plus courtes peut être plus rapide que la recherche sur une clé de longueur imprévisible (trouvée dans les index basés sur des arbres).
Bien que la technique de hachage fonctionne correctement avec une taille de données limitées, elle a tendance à présenter une surcharge de calcul d'indexation à mesure que la taille des données augmente .

Stratégie d'indexation personnalisée (Custom Indexing Strategy)

L'indexation personnalisée prend en charge l'indexation de champs multiples basée sur des index arbitraires ou définis par l'utilisateur . Ils sont généralement basés sur des stratégies d'indexation telles que B-tree, R-tree, index inversé et stratégie d'indexation par hachage.

Deux types de stratégies d'indexation personnalisées sont l'arbre de recherche généralisé (Generalized Search Tree)(GiST) et l'index inversé généralisé (Generalized Inverted Index)(GIN):

- **Gist: Generalized Search Tree**
  L'arbre de recherche généralisé ou stratégie d'indexation GiST, est une stratégie d'indexation basée sur le B-tree ou le R-tree . Il permet la création de champs personnalisés ou arbitraires sous forme d'index.
  Le GiST a la même implémentation (pour l'indexation et la récupération) que l'arbre R pour ceux basés sur le Rtree et que l'arbre B pour ceux basés sur l'arbre B. Par conséquent, ils prennent en charge l'indexation et les requêtes sur des données unidimensionnelles, ainsi que des données multidimensionnelles ou spatiales.
  Comme toute autre stratégie d'indexation basée sur des arbres, Gist a des nœuds racines, des nœuds feuilles, des pointeurs et d'autres caractéristiques d'une structure arborescente équilibrée. Malgré ces similitudes entre les stratégies Gist et arborescentes, la première présente l'avantage de prendre en charge les requêtes ad hoc par rapport à la seconde.
  En prenant Gist basé sur R-tree par exemple, chaque nœud contient une paire clé-pointeur, où la clé est la clé recherchée, et le pointeur pointe ou fait référence au nœud correspondant (ou élément de données, dans le cas d'un nœud feuillé) comme illustré à la figure . En outre, chaque nœud contient des valeurs minimums et maximum, à l'exception du nœud racine.
  Le gist fonctionne bien en matière de recherche de requêtes, mais est généralement considéré comme plus lent que la GIN .

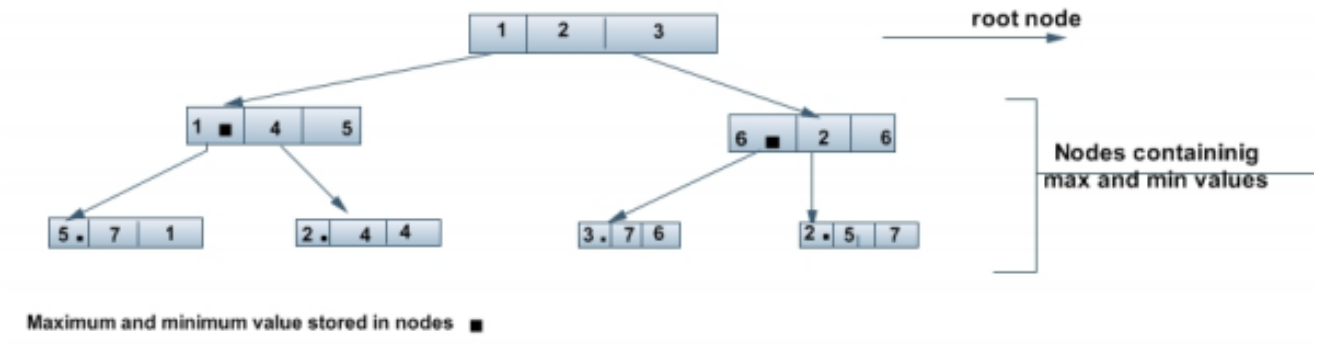


Figure 3.8: indexation GIST

- **GIN: Generalized Inverted Index**
  Tout comme dans le Gist, la stratégie d'indexation (ou méthode d'accès) d'index inversé généralisé ou GIN utilise des champs personnalisés ou arbitraires comme index . Il est

conçu pour les besoins spécifiques des utilisateurs. Bien que la GIN soit implémentée comme le B-tree et ait les propriétés de l'index inversé, GIN diffère du B-tree qui a des opérations basées sur la comparaison qui sont prédéfinies. GIN est constitué d'un index B-tree qui comprend un arbre ou une liste d'entrées (ET) et un arbre de publication (PT) ou une liste de publications (PL) [52]. Dans l'ET, chaque entrée représente un élément de la clé recherchée ou de la valeur indexée, par exemple, des tableaux. Le PL est un pointeur vers une liste d'éléments, ou un pointeur vers un arbre B (pour les nœuds feuilles), auquel cas il est appelé un PT, comme illustré à la figure 3.9 .
. Alors que l'arbre B est bon pour un correspond aux index ou aux requêtes de plage, le GIN fonctionne mieux pour les index ayant de nombreux doublons. En effet, le GIN interroge les données uniquement par correspondance de points ou d'égalité. Ceci est souvent considéré comme une limitation.

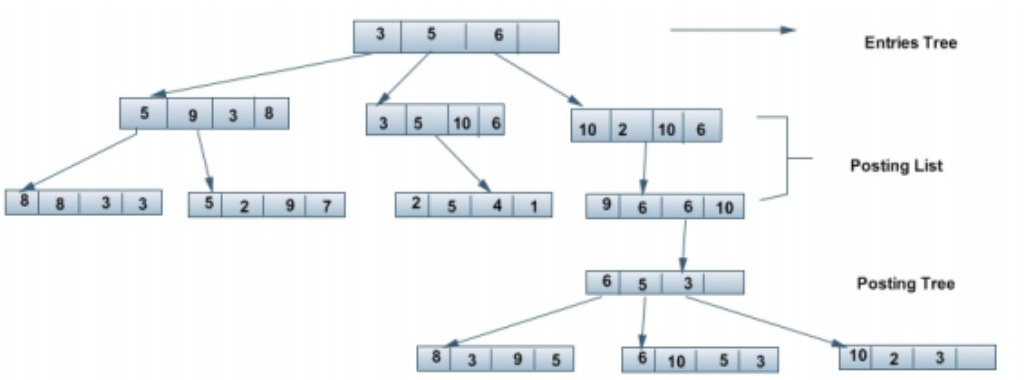


Figure 3.9: indexation GIN

d- Stratégie d'indexation inversée

La stratégie d'indexation inversée permet de concevoir des index inversés qui sont utilisés pour la recherche en texte intégral comme ce qui est disponible dans Google et d'autres moteurs de recherche .
Un index inversé est constitué d'une liste de tous les mots uniques qui apparaissent dans les documents et d'une liste de documents dans lesquels chaque mot apparaît.
Avec un index inversé, plusieurs documents peuvent avoir la même clé que l'index. En outre, plusieurs clés peuvent être utilisées pour indexer un document. Par exemple, un article de blog peut avoir plusieurs balises (en tant que clé) et chaque balise peut faire référence à plusieurs articles de blog.

Bien que certaines stratégies d'indexation inversée utilisent des arbres B ou peuvent être remplies avec des lignes, il convient de noter que tous les index inversés ne sont pas basés sur des arbres B. La différence entre un index inversé et un B-tree est que les Btrees utilisent des données structurées en lignes, contrairement aux index inversés.
L'indexation inversée est mise en œuvre en stockant ou en indexant un ensemble de paires de listes de points clés, où clé est l'index recherché et liste de publicationS est une collection de documents où la clé apparaît.
Le problème avec les index inversés est que deux mots ou plus (clés) peuvent être des termes distincts, mais apparaîtront à l'utilisateur comme le même terme. En outre, les synonymes (des clés) peuvent ne pas être reconnus ou récupérés lors de la recherche par requête [13] .

3.2.3 Synthèse

La taxonomie des stratégies d'indexation populaires utilisées dans le Big Data est illustrée à la figure . Les deux principales catégories sont l'IA et le NAI. L'IA utilise la signification contextuelle des données pour établir des relations (entre les éléments de données) qui servent de base à la création d'index. Les approches d'indexation IA les plus populaires sont LSI et HMM. NAI, en revanche, ne détecte pas le comportement inconnu des données [35]. Les techniques NAI les plus populaires sont les B-tree, R-tree (et leurs variantes), Hash, Inverted et Custom Indexing.

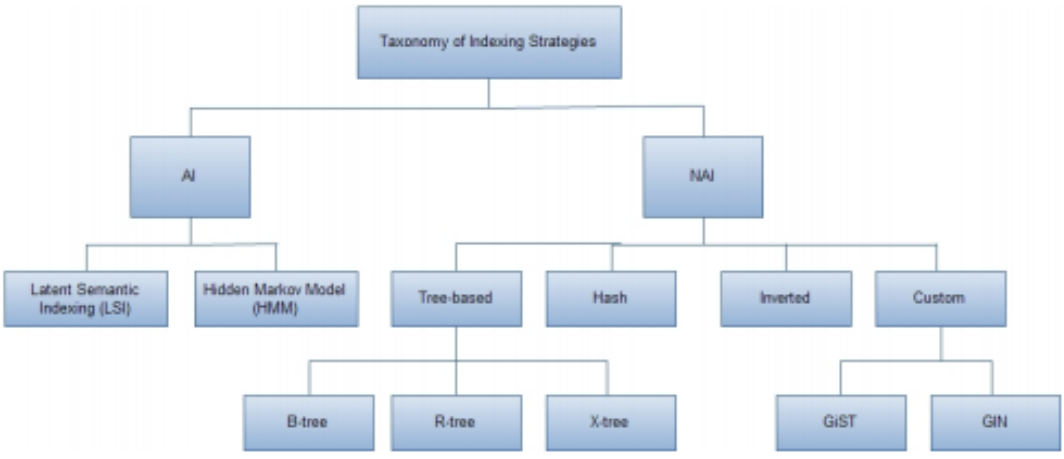


Figure 3.10: Taxonomie de la stratégie d'indexation

Le tableau 1 résume les différents types de stratégies d'indexation, ainsi que le type possible de données et de requêtes qu'elles prennent en charge.
Le tableau 2 présent les principales caractéristiques de chaque stratégie d'indexation. Les principales caractéristiques et défis rencontrés dans chacun d'eux sont décrits.

Stratégie d'indexation	**Type de données**	**Type de requête**
B-tree	données de journal (Log data) données multimédias	requêtes de plage ,requêtes de similarité (NNS) : 1 dimension
R-tree	données spatiale données multimédias	requêtes de plage ,requêtes spatiale : 2-3 dimension
X-tree	données spatiale	requêtes de plage ,requêtes spatiale : multi dimension
Hash	données de journal (log) données multimédias	requête de point (recherche d'égalité)
Gist	données de journal (log) données spatiales	requêtes de plage requêtes ad-hoc
Gin	données de journal (log) données multimédias	requête de similarité requêtes ad-hoc
Inverted	données multimédias documents	requêtes de mots clés
Lsi	données multimédias données spatiale	requêtes de mots clés
HMM	données multimédias signals instables	requêtes ad-hoc

Table 3.2: Déférents techniques d'indexation

Stratégie d'indexation	**Propriétés**	**Challenges**
B-tree	-méthode d'accès unidimensionnelle -Arborescence avec nœuds et pointeurs -Échelle linéairement	-gaspillage de espace de stockage -Ne convient pas pour un accès multidimensionnel -Consomme d'énormes ressources informatiques
R-tree	-Plus évolutif que le B-tree -Méthode d'accès de 2 à 3 dimensions	-Index consomme plus d'espace mémoire
X-tree	-Méthode d'accès multidimensionnelle	-Consomme de l'espace mémoire
Hash	-Présente la réponse exacte (utilise l'opérateur '=') -Récupération rapide des informations	-Les frais généraux de calcul
Gist	-Index arbitraires -Basé sur les structures arborescentes	-Réponse à la requête plus lente
Gin	-Index arbitraires -Basé sur les structures arborescentes	-Temps de traitement plus long
Inverted	-Index consomme moins d'espace -recherche plein texte (recherche par mot-clé)	-Temps de traitement des données plus long -Limite l'espace de recherche, ne produisant pas nécessairement la réponse exacte -Peut présenter de mauvaises réponses en raison de synonymes et de polysémie
Lsi	-Utilise les données et la signification des données pour l'indexation -Présente des résultats de requête précis (car il utilise plus d'informations)	-Demande de hautes performances de calcul -Consomme plus d'espace mémoire
HMM	-Basée sur le modèle Markov -Reconnaît les relations entre les données	-Demande de hautes performances de calcul

Table 3.3: Caractéristiques des stratégies d'indexation

3.3 Technique de sélection les vues matérialisées

Pour accélérer le traitement des requêtes sur le Big Data, des sous-requêtes ou vues fréquentes peuvent être matérialisées de sorte que le coût de traitement des requêtes soit minimisé avec un coût optimal de maintenance des vues et / ou requêtes matérialisées.

3.3.1 Définition et principe

Les vues sont un ensemble de données relationnelles logiques dérivées des tables de base afin que les requêtes complexes puissent être simplifiées en accédant à ces vues. Si les données extraites des vues intermédiaires de ces requêtes sont sauvegardées ou matérialisées, au lieu de générer et de récupérer des données des tables de base pour ces relations dérivées encore et encore, elles peuvent être directement lues à partir des vues matérialisées.
La matérialisation de sous-requêtes et de vues fréquentes signifie que l'ensemble de données résultant des vues réside dans la mémoire d'un ou plusieurs nœuds du cluster, ce qui réduit le coût MapReduce, le coût de soumission et de planification des travaux du système de fichiers distribués pour le traitement des requêtes.

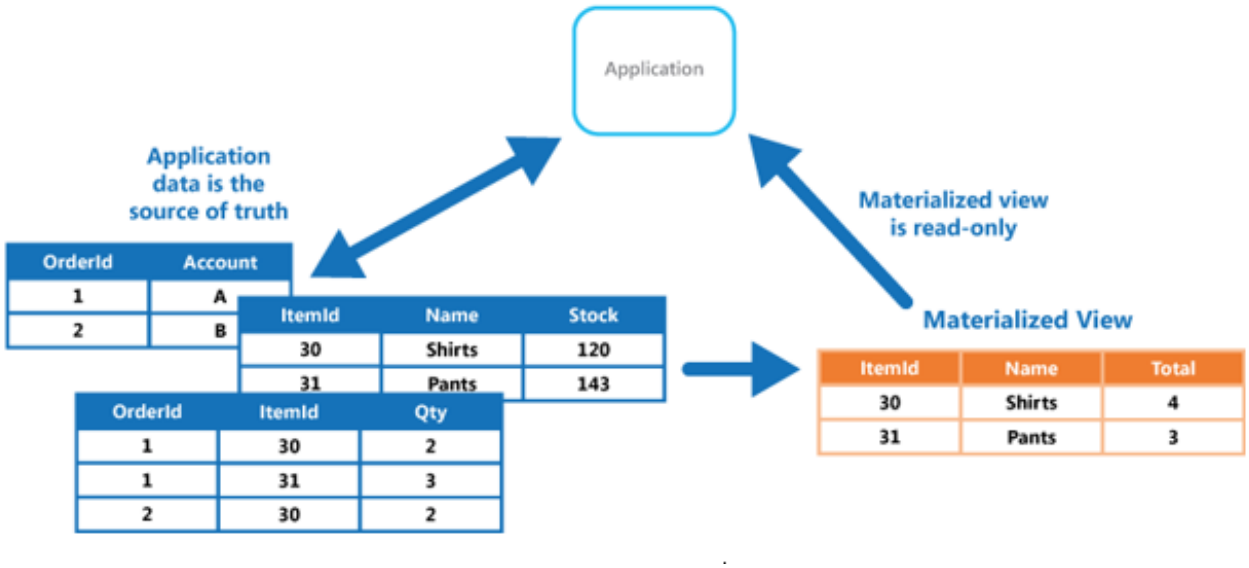


Figure 3.11: materialized-view-pattern-diagram

La matérialisation des vues temporaires intermédiaires générées lors du traitement des requêtes signifie un besoin d'espace supplémentaire. Par conséquent, il est nécessaire de sélectionner un ensemble optimal de vues à matérialiser pour augmenter les performances de traitement des requêtes, avec un coût de maintenance de vue matérialisée optimisé et un coût d'espace de matérialisation.

Les vues matérialisées peuvent être indexées pour augmenter encore les performances de la requête. Ces vues matérialisées sont conçues par des fonctions d'agrégation sur des tables de base, ou comme des copies de sous-requêtes fréquemment exécutées d'un ensemble de requêtes fréquentes [40].

3.3.2 Problème de sélections des vues

En basant sur la définition de Chirkova et al. (Chirkova et al., 2001): Soit R l'ensemble des relations de base (comprenant les tables de faits et de dimensions), S l'espace de stockage disponible, Q une charge de travail sur R, L la fonction d'estimation du coût du traitement des requêtes . Le problème de sélection de vue est de trouver l'ensemble des vues V (configuration de vue) sur R dont la taille totale est au plus S et qui minimise L (R, V, Q) [2].

3.3.3 Modèle de coût

Le modèle de coût à utiliser pour traiter le problème de sélection de vue matérialisée pour HDFS est différent des modèles de coût utilisés dans les approches utilisées pour l'architecture client-serveur conventionnelle avec l'entreposage de données basé sur le SGBDR. La principale raison derrière cela est que, dans un système classique basé sur un SGBDR, le modèle d'accès aux données est principalement dominé par les recherches et le temps de recherche, tandis que dans HDFS ou dans un cadre distribué similaire, le modèle d'accès aux données est principalement dominé par le taux de transfert de données et les coûts de MapReduce [40] .
MapReduce overheads sont composés des couts de transfert de données liés au transfert des données vers un certain nombre de nœuds de données à travers le cluster DFS, l'exécution de traqueurs de travaux (job trackers] et de traqueurs de tâches (task trackers), la création de mappeurs(mappers) et de réducteurs (reducers), overheads importants dans la soumission et la planification des travaux.
Dans DFS de gestion de Big Data, la taille du bloc et la taille de la division sont fixes. Par conséquent, dans le cadre HDFS / MapReduce, un petit nombre de gros fichiers sont préférables à un grand nombre de petits fichiers . Cela signifie que dans le cas de HDFS, un plus petit nombre de vues plus grandes doit être préféré pour la matérialisation. Ce critère n'est pas applicable dans le cas des vues matérialisées traditionnelles de l'entrepôt de données basées sur le SGBDR.

Les différents coûts et bénéfices à optimiser pour matérialiser les vues afin d'améliorer le traitement des requêtes dans l'entrepôt Big Data sont définit :

- **Coût de traitement les requêtes :**
  Le coût total de traitement des requêtes d'un ensemble de requêtes fréquentes peut être considéré comme le total des sur coûts MapReduce pour l'exécution de l'ensemble de requêtes. Si les résultats de certaines sous-requêtes et fonctions d'agrégation utilisées dans ces requêtes sont matérialisés ou enregistrés, lors de l'exécution ultérieure des requêtes, les frais généraux de MapReduce liés à l'exécution de ces sous-requêtes ou vues sont enregistrés. Si un ensemble de sous-requêtes d'un ensemble de requêtes fréquentes est traité et composé en tant que vues et matérialisé dans HDFS, le coût de traitement des requêtes de l'ensemble de requêtes considéré peut être défini comme Définition :
  Pour un ensemble de n nombre de requêtes fréquentes Q = q1, q2, •••, qn sur un entrepôt de données, où V est l'ensemble des m vues intermédiaires générées par Q, et n m, si V V est l'ensemble des vues V = v1, v2, •••, vp qui sont matérialisées, le coût total de traitement des requêtes HDFS peut être défini par l'expression suivante :

$$C_{V'}^{Q} = C_{|V'|=0}^{Q} - \sum_{i=1}^{p} M_{v_i}$$

- **Coût de maintenance des vues matérialisées :**
  Dans l'analyse Big Data sur un entrepôt basé sur HDFS, les opérations de mise à jour sont généralement très rares. Mais chaque fois qu'il y a un changement dans les données de base, les vues matérialisées doivent être mises à jour. En cas de matérialisation des requêtes en mémoire, un rafraîchissement fréquent est nécessaire car dans ce cas les requêtes peu fréquentes doivent être rejetées après chaque période de temps fixe . La maintenance des vues matérialisées signifie le retraitement des fonctions d'agrégation et / ou des sous-requêtes correspondantes, puis la mise à jour des vues dans des disques ou des disques SSD. Il y aura donc un autre ensemble de DFS overheads . Le coût de maintenance de la vue matérialisée peut donc être défini comme suit [40].
  Pour un ensemble de vues matérialisées V = v1, v2, •••, vp pour traiter un ensemble de requêtes Q, le coût de maintenance des vues matérialisées peut être exprimé comme : Ou Uvi , i = 1, 2 ••• , p , sont les frais MapReduce overheads de maintenance pour

$$U(V') = \sum_{i=1}^{p} U_{v_i}$$

  l'ensemble des vues matérialisées vi V' , i = 1, 2 ••• p.

Nombre de vues à matérialiser et le besoins en espace de stockage

Dans HDFS, un plus petit nombre de tables plus grandes est préféré (comme indiqué dans la section 2.1). L'espace de stockage requis pour p nombre de vues matérialisées V, où —V'— = p, peut être défini comme :
Si Svi est l'espace de stockage requis par la ième vue matérialisée, alors l'espace total requis pour matérialiser p nombre de vues est :

$$S(V') = \sum_{i=1}^{p} S_{vi}$$

Conception de vue (View Design)

La conception de vue détermine quelles vues doivent être matérialisées. D'une part, l'espace pour stocker les vues est limité; D'autre part, le coût de recherche pour trouver une vue associée est proportionnel à la quantité de vues matérialisées. Ainsi, il n'est pas pratique de matérialiser toutes les vues. Ils doivent être sélectionnés avant la matérialisation. La conception de la vue comporte deux étapes: la vue de l'énumération et la sélection de la vue.

- **Afficher l'énumération:** le but de l'énumération de la vue est de réduire le nombre de vues candidates à prendre en compte par la phase de sélection. Il filtre les vues non liées. Cette phase est facultative.
- **Sélection de vues:** la sélection de vues est basée sur un modèle de rentabilité. Étant donné que les bases de données relationnelles utilisent l'optimiseur de requêtes basé sur les coûts, View Sélection peut être facilement intégré à l'optimiseur de requêtes.

La vue est avantageuse si: (1) elle est coûteuse à calculer; et (2) elle peut être réutilisé par d'autres requêtes. Le coût provient de deux aspects:

1. la construction des vues - les overheads pour sélectionner, créer et stocker les vues; et
2. la maintenance des vues - les frais généraux pour maintenir les vues à jour. Les vues seront sélectionnées par l'optimiseur de requêtes en fonction de leurs avantages et de leurs coûts [15].

3.3.4 Travaux connexes

Travaux de Julian Hyde [22]

propose une extension aux vues matérialisées appelées Discardable, In-Memory Materialized Query - DIMMQ for Hadoop (Hyde 2014). DIMMQ propose que l'ensemble de données résultant de certaines requêtes fréquentes réside dans la mémoire d'un ou plusieurs nœuds du cluster. Discardable signifie que le système peut supprimer les requêtes matérialisées en mémoire lorsqu'elles ne sont pas utilisées pendant une longue période. Ici, il est proposé que certaines sous-requêtes puissent être sauvegardées dans la mémoire du cluster matériel avec un mappage vers leurs données résultantes sur le disque. Mais même ici, les frais généraux de MapReduce pour la soumission et la planification des travaux restent avec le coût de maintenance pour actualiser le mappage entre les requêtes en mémoire et les données du disque. Par conséquent, qu'il s'agisse de vues matérialisées ou de requêtes matérialisées en mémoire, un sous-ensemble de l'ensemble candidat de vues ou de requêtes doit être sélectionné pour se matérialiser de telle sorte que tous les coûts associés soient minimisés avec un coût total de traitement des requêtes minimisé d'un ensemble de requêtes définies comme des requêtes d'entrepôt fréquents pour une application d'entrepôt de données spécifique.

Travaux de Rajib Goswami1 , D. K Bhattacharyya1 ,Malayananda Dutta [40]

Ils ont présenté une tentative de conception d'un système pour trouver un ensemble de solutions de vues à matérialiser afin d'optimiser le coût de traitement des requêtes, le coût de maintenance des vues matérialisées et le stockage HDFS à partir des vues générées lors du traitement d'un ensemble de requêtes sur l'entreposage Big Data dans le cadre HDFS. Le problème est défini comme un problème d'optimisation multi-objectif pour trouver un ensemble de vues de solutions non dominé à l'aide de l'algorithme d'évolution différentielle DE multi-objectif et de l'algorithme génétique de tri non dominé-NSGA-II (Deb et al. 2002). [4] Gong et Tuson (2006) présentent l'utilisation de l'analyse forma pour exploiter l'utilisation de DE pour un problème d'optimisation discrète. Ici, nous avons personnalisé l'analyse de forma basée sur un objectif simple DE présenté dans Gong et Tuson (2006) [5] en DE multi-objectifs pour les données codées binaires (MODE-BE) pour sélectionner des vues à matérialiser dans le cadre de l'entrepôt de données HDFS en promouvant la diversité dans l'espace vectoriel de décision. Dans NSGA-II, la diversité d'un grand nombre de solutions est favorisée en calculant les distances d'encombrement entre les solutions dans l'espace de valeurs de fonction objective. ils ont également développé un prototype du système de recommandation utilisant NSGA-II sur ce problème pour analyser les performances entre les systèmes basés sur NSGA-II et les systèmes

de recommandation basés sur MODE-BE pour la sélection de vues matérialisées dans le cadre de gestion Big Data [40].

Travaux de Ordonez-Ante, L., Van Seghbroeck, G., Wauters, T., Volckaert, B. and De Turck, F[50]

Ils ont développé un mécanisme automatique de sélection de vue matérialisée au-dessus des données modélisées dimensionnellement distribuées. Le mécanisme repose sur l'analyse syntaxique des charges de travail de requête en utilisant une matrice d'attributs représentative comme structure de données d'entrée, assemblée sous la forme d'une collection de vecteurs de caractéristiques codant toutes les clauses de chaque requête individuelle dans la charge de travail . Avec cette entrée, une stratégie pour sélectionner un ensemble limité de vues matérialisables candidates est mise en œuvre, comprenant l'utilisation d'un regroupement hiérarchique avec une fonction de distance de requête personnalisée conforme à la structure des vecteurs de caractéristiques, et l'estimation d'un score matérialisable sur le résultat configuration de clustering, permettant d'identifier sans ambiguïté des groupes de requêtes matérialisables.
Ils ont exploré les techniques conçues pour extraire les vecteurs de caractéristiques des instructions de requête, regrouper les requêtes liées en fonction d'une estimation de leur similitude par paires et dériver un ensemble limité de vues matérialisées capables de répondre aux requêtes regroupées sous chaque cluster.

Travaux de Alekh Jindal Konstantinos Karanasos Sriram Rao Hiren Patel [47]

ils ont exploré une nouvelle approche hautement évolutive du problème bien étudié de la réutilisation des calculs: algorithme BIGSUBS, ils ont concentré sur la sélection de sous-expressions, une spécialisation du problème de sélection de vues qui considère les sous-arbres de plans de requête logiques comme des candidats de vue. ils ont suive une approche holistique qui considère périodiquement l'ensemble de la charge de travail avec des dizaines de milliers de requêtes, et sélectionne les sous-expressions les plus prometteuses à matérialiser pour améliorer l'évaluation des requêtes ultérieures.
En particulier, ils ont mappé la sélection de sous-expressions à un problème d'étiquetage de graphe bipartite. Les sommets du graphique représentent les requêtes et les sous-expressions candidates, tandis que les arêtes relient chaque requête à ses sous-expressions. Ensuite, ils ont divisé l'étiquetage du graphe en deux sous-problèmes: (i) étiqueter les sommets de la sous-expression, qui dicte les sous-expressions qui seront matérialisées, et (ii) étiqueter les arêtes, qui détermine les sous-expressions matérialisées qui seront utilisées pour évaluer chaque requête .

Pour s'adapter aux tailles de charge de travail susmentionnées, ils ont suive un modèle de traitement de graphe centré sur les sommets qui effectue de manière itérative les étapes d'étiquetage ci-dessus en parallèle jusqu'à ce qu'elles convergent. Pour la partie étiquetage de vertex, ils ont suive une approche probabiliste, tandis que pour la partie étiquetage d'arête, ils ont résolve des ILP locaux par requête.

3.3.5 Synthèse

Le tableau résume les différentes approches basées sur les vues matérialisées qu'on a déjà vu dans les travaux connexes .IL présente les avantages de chaque approche et sur quel principe elle sont basées , ainsi que les futurs travails concernant chacune d'elles .

L'approche	Basée sur	Ces avantages	Travail Future
Julian Hyde Discardable, In-Memory Materialized Query - DIMMQ for Hadoop	expression algébrique, règles de réécriture, modèle de coût	- le système peut supprimer les requêtes matérialisées en mémoire lorsqu'elles ne sont pas utilisées pendant une longue période. - tous les coûts associés sont minimisés avec un coût total de traitement des requêtes minimisé.	DIMMQ lui-même est une méta-gestion supplémentaire de la couche de stockage,
Rajib Goswami Sélection de vues matérialisées à l'aide d'un algorithme évolutif pour accélérer le traitement des requêtes Big Data	-algorithme évolutif multi-objectif -NSGA-II	-optimiser le coût de traitement des requêtes, le coût de maintenance des vues matérialisées et le stockage HDFS . - l'approche est générique, elle peut être mise en œuvre facilement pour tout type de cadre similaire qui utilise un cluster distribué de matériel de base	le processus d'analyse sémantique pour l'extraction et la composition des vues candidates reste à développer.
Alekh Jindal BIGSUBS Selecting Subexpressions to Materialize at Datacenter Scale	subexpres-sions scope	BIGSUBS gère d'importantes charges de travail de production comprenant des dizaines de milliers de tâches - diviser le problème global en deux sous-problèmes, que nous résolvons séparément de manière itérative - optimisations d'élagage peuvent effectivement réduire l'espace de recherche de plus de 90 pourcents dans la plupart des cas .	
Leandro Ordonez-Ante, Gregory Van Seghbroeck : Sélection automatique de vue pour les données dimensionnelles distribuées	l'analyse syntaxique	Les vues proposées par le mécanisme conçu se sont révélées être une méthode efficace pour contourner les opérations de jointure distribuée coûteuses et réduire par la suite le temps de traitement des requêtes de 89 à 98 pour-cent par rapport au temps d'exécution sur la dimension distribuée de base	La mise en œuvre de la fonction de prise de conscience de cardinalité pour le mécanisme de sélection de vue proposé, ainsi que la stratégie de maintenance

Table 3.4: Caracteristique des approches basées sur les vues materialisées

Chapter 4

Conclusion Générale

De nos jours, les Big Data attirent de plus en plus l'attention du monde universitaire, de l'industrie et du gouvernement d'où L'analyse de Big Data est devenu une exigence très courante aujourd'hui. Le paradigme de programmation Hadoop MapReduce et HDFS sont de plus en plus utilisés pour traiter des ensembles de données volumineuses et non structurés.

Dans notre mémoire nous avons pu définir le Data warehouse et Big Data analytique, présenter une revue des principales fonctionnalités d'Apache Hadoop, Spark et Hive pour l'analyse du Big Data et enfin nous avons vu les différentes techniques et approches pour l'optimisation des requêtes big data tel que l'optimisation par partitionnement, indexation ou matérialisation ainsi que problèmes et le l'avantage de chaque technique qui sont un peu différentes Dues à la nature distribuée des environnements Big Data:

- l'approche d'optimisation par partitionnement a un problème important lié à la façon dont les données sont partitionnées et attribuées aux nœuds est la répartition de charge, car si les nœuds sont affectés à différentes charges de travail, les avantages du calcul parallèle diminuent parce que les ressources peuvent être sous-utilisées, et la durée totale d'exécution devient celle du nœud le plus lent. L'un des défis liés à ce problème est que, dans de nombreux cas, la localité des données et la répartition de charge sont contradictoires et un compromis doit être trouvé.
  Par contre, cette technique a des avantages comme : elle fait progresser les fonctionnalités de requête. Les requêtes peuvent être facilement et rapidement résolues pour une collection de partitions au lieu de les résoudre pour une base de données géante. Par conséquent, la fonctionnalité et le niveau de performance sont améliorés. Le temps d'entracte prévu est abrégé. elle facilite les procédures d'administration des informations telles que le chargement d'informations, ainsi que la sauvegarde et la reprise au stade de la partition. En conséquence, les processus deviennent plus rapides.

- une deuxieme approche d'optimisation est l'optimisation par indexation, d'où le choix

d'index constitue un des problèmes majeurs. un autre problème, si le tableau est mis à jour tout le temps, les frais généraux du la gestion et la réorganisation des indices peuvent nous conduire à ne pas choisir l'index. Chaque saisie de données de modification implique potentiellement la mise à jour des index, ce qui peut conduire à une performance plus lente des mises à jour des données.
d'autres problèmes liés à l'indexation du Big Data sont: Évolutivité, Création d'index coûteuse , Indexation dimensionnelle élevée, Quel attribut pour créer un index, réindexation de l'ensemble du document (lors des index devenir corrompu ou de nouvelles fonctionnalités sont ajoutés), Taille de l'index (La taille de l'index doit être une fraction des données originales).
Des avantages de la technique d'indexation Big Data sont principalement: l'accélération et l'amélioration de la récupération et l'interrogation des données, la couverture d'un large éventail de types de données(notamment des données du type relationnelles, des données spatiales et des données JSON semi-structurés).

- une autre approche de materialisation des vues qui a un défi majeur pour gérer le problème de sélection de vues pour la matérialisation dans les Mega données et de réduire la complexité des algorithmes de sélection de vues.
  Les vues matérialisées peuvent être utilisées pour satisfaire plusieurs objectifs, comme l'amélioration de la performance des requêtes: permet de réduire significativement le temps de réponse pour des requêtes d'agrégation ou de jointure.

Notre travail fait montrer qu'il n'existe pas une solution universelle qui répond au problème de meilleure approche d'optimisation de performances des requêtes Big Data, mais chaque mécanisme dépend de type de donnée et type de requêtes à traiter ainsi que le contexte global du système et ses contraintes. Donc Le choix d'une approche particulière dépendrait l'administration du système et de nombreux facteurs tels que les caractéristiques des programmes, l'architecture du système, les capacités du réseau, etc.

Bibliography

[1] William H. Inmon. *Building the Data Warehouse*. USA: John Wiley Sons, Inc., 1992. ISBN: 0471569607.

[2] Rada Chirkova, Alon Halevy, and Dan Suciu. "A Formal Perspective on the View Selection Problem". In: *VLDB Journal* 11 (Sept. 2001). DOI: 10.1007/s00778-002-0070-0.

[3] Yao Z and Ravishankar Cv. "Hash-based virtual hierarchies for caching in hybrid contentdelivery networks". In: (2001).

[4] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[5] Antony W. Iorio and Xiaodong Li. "Incorporating Directional Information within a Differential Evolution Algorithm for Multi-Objective Optimization". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. GECCO '06. Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 691–698. ISBN: 1595931864. DOI: 10.1145/1143997.1144119. URL: https://doi.org/10.1145/1143997.1144119.

[6] X. Wang and D. Loguinov. "Load-Balancing Performance of Consistent Hashing: Asymptotic Analysis of Random Node Join". In: *IEEE/ACM Transactions on Networking* 15.4 (2007), pp. 892–905.

[7] W. H. Inmon. In: *Building the Data Warehouse*. Robert Ipsen, 2008, p. 43. ISBN: 0-471-08130-2.

[8] Michael J. Cahill, Uwe Röhm, and Alan D. Fekete. "Serializable Isolation for Snapshot Databases". In: *ACM Trans. Database Syst.* 34.4 (Dec. 2009). ISSN: 0362-5915. DOI: 10.1145/1620585.1620587. URL: https://doi.org/10.1145/1620585.1620587.

[9] Rudi Bruchez. "Les bases de données NoSQL et le Big Data - Comprendre et mettre en oeuvre". In: 2010.

[10] A. Thusoo et al. "Hive - a petabyte scale data warehouse using Hadoop". In: *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 2010, pp. 996–1005.

[11] Andrew McAfee and Erik Brynjolfsson. "Big Data: The Management Revolution". In: *Harvard business review* 90 (Oct. 2012), pp. 60–6, 68, 128.

[12] Z. Chen et al. "Hybrid Range Consistent Hash Partitioning Strategy – A New Data Partition Strategy for NoSQL Database". In: *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. 2013, pp. 1161–1169.

[13] R. Irudeen and S. Samaraweera. "Big data solution for Sri Lankan development: A case study from travel and tourism". In: *2013 International Conference on Advances in ICT for Emerging Regions (ICTer)*. 2013, pp. 207–216.

[14] D. Loshin. "Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph". In: 2013.

[15] Hui Shang. "Reusing Results in Big Data Frameworks". In: *KTH Information and Communication Technology* 49 (2013), p. 27.

[16] A.Turk et al. "Temporal Workload-Aware Replicated Partitioning for Social Networks". In: vol. 26. 11. 2014, pp. 2832–2845.

[17] Veronika Abramova, Jorge Bernardino, and Pedro Furtado. "Testing Cloud Benchmark Scalability with Cassandra". In: *2014 IEEE World Congress on Services* (2014), pp. 434–441.

[18] Sachin P Bappalige. *An introduction to Apache Hadoop for big data*. Aug. 2014.

[19] Min Chen et al. *Big Data: Related Technologies, Challenges and Future Prospects*. Springer Publishing Company, Incorporated, 2014. ISBN: 3319062441.

[20] Ivan Giangreco, Ihab Kabary, and Heiko Schuldt. "ADAM - A Database and Information Retrieval System for Big Multimedia Collections". In: June 2014, pp. 406–413. DOI: 10.1109/BigData.Congress.2014.66.

[21] Alex Holmes. *Hadoop in Practice: Includes 104 Techniques*. Manning Publications, Oct. 2014. Chap. 1. ISBN: 1617292222.

[22] J.Hyde. "Discardable in-memory, materialized query for hadoop." In: 49 (2014), p. 27.

[23] Nawsher Khan et al. "Big data: survey, technologies, opportunities, and challenges". In: *The scientific world journal* 2014 (2014).

[24] Ayaka Matsui, Satoshi Nishimura, and Seiichiro Katsura. "A classification method of motion database using hidden Markov model". In: June 2014, pp. 2232–2237. ISBN: 978-1-4799-2399-1. DOI: 10.1109/ISIE.2014.6864965.

[25] Widodo and W. C. Wibowo. "Improving classification performance by extending documents terms". In: *2014 International Conference on Data and Software Engineering (ICODSE)*. 2014, pp. 1–5.

[26] Fatima Adamu et al. "A Survey On Big Data Indexing Strategies". In: Dec. 2015. DOI: 10.13140/RG.2.1.1844.0721.

[27] Haider andomi. "Beyond the hype: Big data concepts, methods, and analytics". In: *International Journal of Information Management* 35 (Apr. 2015), pp. 137–144.

[28] Mohd Rehan Ghazi and D. Gangodkar. "Hadoop, MapReduce and HDFS: A Developers Perspective". In: *Procedia Computer Science* 48 (2015), pp. 45–50.

[29] Te-Yuan Lin and Chiou-Shann Fuh. "Partitioning Technology and Fast Content Movements of Big Data". In: 2015.

[30] Durgaprasad Gangodkara Mohd Rehan Ghazia. "Hadoop, MapReduce and HDFS: A Developers Perspective". In: *International Conference on Intelligent Computing, Communication Convergence (ICCC-2014* 19 (2015).

[31] L. Srinivasan and Vasudeva Varma. "Adaptive Load-Balancing for Consistent Hashing in Heterogeneous Clusters". In: *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2015), pp. 1135–1138.

[32] Chun-Wei Tsai et al. "Big data analytics: a survey". In: *Journal of Big data* 2.1 (2015), pp. 1–32.

[33] Tom white. *hadoop , the definitive guide*. 4th ed. 2015.

[34] E. Dede et al. "Processing Cassandra Datasets with Hadoop-Streaming Based Approaches". In: *IEEE Transactions on Services Computing* 9.1 (2016), pp. 46–58.

[35] Abdullah Gani et al. "A Survey on Indexing Techniques for Big Data: Taxonomy and Performance Evaluation". In: *Knowl. Inf. Syst.* 46.2 (Feb. 2016), pp. 241–284. ISSN: 0219-1377. DOI: 10.1007/s10115-015-0830-y. URL: https://doi.org/10.1007/s10115-015-0830-y.

[36] Raghavendra Kune et al. "The anatomy of big data computing". In: *Softw. Pract. Exp.* 46 (2016), pp. 79–105.

[37] Hai Wang et al. "Towards felicitous decision making: An overview on challenges and trends of Big Data". In: *Information Sciences* 367 (2016), pp. 747–765.

[38] Matei Zaharia et al. "Apache spark: A unified engine for big data processing". In: *Communications of the ACM* 59 (Nov. 2016), pp. 56–65. DOI: 10.1145/2934664.

[39] Sally M. Elghamrawy and Aboul Ella Hassanien. "A partitioning framework for Cassandra NoSQL database using Rendezvous hashing". In: *The Journal of Supercomputing* 73 (2017), pp. 4444–4465.

[40] Dutta Goswami Bhattacharyya. "Materialized view selection using evolutionary algorithm for speeding up big data query processing". In: *Intell Inf Syst* 49 (2017), p. 27.

[41] Ahmed Oussous et al. "Big Data technologies: A survey". In: *Journal of King Saud University - Computer and Information Sciences* 30 (2017), pp. 431–448.

[42] *Apache Druid: Interactive analytics at scale*. http://druid.io. 2018.

[43] *Apache HBase*. http://hbase.apache.org/. 2018.

[44] *Apache ORC: High-Performance Columnar Storage*. http://orc.apache.org/. 2018.

[45] *Apache Parquet*. http://parquet.apache.org. 2018.

[46] *Apache Thrift*. http://thrift.apache.org/. 2018.

[47] Alekh Jindal et al. "Selecting subexpressions to materialize at datacenter scale". In: *Proceedings of the VLDB Endowment* 11 (Mar. 2018), pp. 800–812. DOI: 10.14778/3192965.3192971.

[48] Pwint Phyu Khine and Zhao Shun Wang. "Data lake: a new ideology in big data era". In: *ITM Web of Conferences* 17 (2018). Ed. by Kei Eguchi and Tong Chen, p. 03025. DOI: 10.1051/itmconf/20181703025. URL: https://doi.org/10.1051/itmconf/20181703025.

[49] Srini Penchikala. *Big data processing with apache spark*. Lulu. com, 2018.

[50] Leandro Ante et al. “Automatic View Selection for Distributed Dimensional Data”. In: May 2019. DOI: 10.5220/0007555700170028.

[51] J. Camacho-Rodríguez et al. “Apache Hive: From MapReduce to Enterprise-grade Big Data Warehousing”. In: *Proceedings of the 2019 International Conference on Management of Data* (2019).

[52] Louise Grandjonc. “PostgreSQL’s indexes - GIN”. In: July 2019.

[53] Erik Jaspers. *Quelles différences entre le big data et l’Analytique ?* Dec. 2019.

[54] Philippe Rigaux. *Systèmes NoSQL*. 2019.