République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLEB – Blida 1

Faculté des Sciences

Département d'Informatique

PROJET DE FIN D'ÉTUDES

Pour l'obtention du diplôme de Master en Informatique

Option : Ingénierie des Logiciels

Thème :

Définition d'une algèbre NoSQL-OLAP adaptée aux environnements Big Data

Réalisé par : LOUMACHI Lyna et CHETOUH Sara.

Soutenu le 27 septembre 2022 devant le jury composé de :

Mme Saliha OUKID	Professeure	Dépt d'Informatique, Blida 1	Présidente
M. Ishak RIALI	Maitre de Conférences (B)	Dépt d'Informatique, Blida 1	Examinateur
M. Mahfoud BALA	Maitre de Conférences (B)	Dépt d'Informatique, Blida1	Promoteur

Promotion : 2021/2022

REMERCIEMENTS

Tout d'abord, nous tenons à rendre grâce à DIEU tout puissant pour nous avoir donné le courage et la détermination nécessaire pour finaliser ce travail et le mener à terme.

Nous tenons à remercier vivement Professeure Saliha OUKID d'avoir accepté honorer ce jury comme présidente

Nous remercions M. Ishak RIALI d'avoir accepté examiner notre travail

Nos vifs remerciements s'adressent à notre promoteur M. BALA qui a endossé son rôle de la meilleure façon qui soit. Nous retiendrons sa patience, ses conseils avisés et ces idées riches ainsi que son aide précieuse

Nous tenons à témoigner toute notre reconnaissance aux personnes suivantes, pour leur aide dans la réalisation de ce mémoire :

Mme Djamila Loumachi, pour avoir relu et corrigé notre mémoire. Ses conseils de rédaction ont été très précieux.

Samir Bettahar, qui a eu l'amabilité de répondre à nos questions et de fournir les explications nécessaires.

Enfin, nous remercions Mme Guessabi Nadjia et notre chère amie Amira Tahri qui ont toujours été là pour nous. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

Résumé

Avec l'avènement du **Big Data**, le volume des données a atteint des tailles critiques, défiant les approches classiques d'entreposage de données, dont les solutions actuelles reposent principalement sur des bases de données relationnelles. Pour stocker, traiter, analyser et diffuser ces mégadonnées, de nouveaux modèles de données appelés modèles **NoSQL** ont vu le jour C'est dans ce contexte que se situe le travail de notre PFE et qui consiste à stocker des cubes de données dans une structure NoSQL en vue de leur exploitation pour des fins d'analyse et d'aide à la décision. Pour ce faire, une nouvelle **algèbre OLAP** est définie. Contrairement à l'algèbre OLAP classique qui est basée sur une structure relationnelle, l'algèbre OLAP que nous proposons est, quant à elle, adaptée à des structures de cubes NoSQL. Nous avons, dans la suite de notre travail, mis en œuvre une plateforme OLAP permettant de manipuler des cubes de données, stockés dans un **SGBD HBase**, avec des opérateurs du type SLICE, DICE adaptés au modèle NoSQL.

Mot clés : NoSQL, OLAP, Big Data, Algèbre OLAP, Orienté colonnes, Apache HBase.

Abstract

With the advent of **Big Data**, the volume of data has reached critical sizes, challenging the classical approaches of data warehousing, whose current solutions are mainly based on relational databases. To store, process, analyze and disseminate these megadata, new data models called **NoSQL** models have emerged. It is in this context that the work of our PFE is situated and which consists in storing data cubes in a NoSQL structure in order to exploit them for analysis and decision support purposes. For this purpose, a new **OLAP algebra** is defined. Contrary to the classical OLAP algebra which is based on a relational structure, the OLAP algebra that we propose is adapted to NoSQL cube structures. In the rest of our work, we have implemented an OLAP platform allowing to manipulate data cubes, stored in a **HBase DBMS**, with operators such as SLICE, DICE adapted to the NoSQL model.

Key-words: NoSQL, OLAP, Big Data, OLAP algebra, Column oriented, Apache HBase.

Table des matières

Introduction générale 1
Contexte 1
Problématique 1
Objectif 2
Structuration du mémoire 2

Chapitre 1 : Entrepôt de données et analyse en ligne OLAP 3

Introduction 4

1.1. Les systèmes d'aide à la décision 4

1.2. L'entrepôt de données 5
1.2.1. Définition et caractéristiques 5
1.2.2. Concept de base 6
1.3. Modélisation multidimensionnelle 7

1.4. L'analyse en ligne des données OLAP 11
1.4.1 Définition 11
1.4.2 L'algèbre OLAP 11

Chapitre 2 : Les données massives : « Big Data » 15

2.1. Définition et caractéristiques 16

2.2 Paradigmes et technologies des Big Data 17
2.2.1. Hadoop 17
2.2.2. Présentation de HDFS 17
2.2.3 Le paradigme MapReduce 18

2.3 Le NoSQL 19
2.3.1 Définition 19
2.3.2 Les modèles NoSQL 20
2.3.4 Les SGBD NoSQL 24

Conclusion 27

Chapitre 3 : Etat de l'art 28

Introduction 29

3.1 Travaux connexes 29
3.1.1 Approches basées sur un modèle orienté clé-valeur 29
3.1.2 Approches basées sur un modèle orienté colonne 30
3.1.3 Approches basées sur un modèle orienté document 32
3.1.4 Approches basées sur un modèle orienté graphe 35

3.3 Discussion 37

Conclusion 39

Chapitre 4 : Conception de l'environnement OLAP adapté aux modèles NoSQL 41

Introduction 42

4.1 Proposition d'un processus de mappage modèle Conceptuel-modèle Logique NoSQL 42
4.1.1 Processus de mapping orienté clé valeur 43
4.1.2 Processus de mapping orienté Colonne 46
4.1.2 Processus de mapping orienté document 48
4.1.3 Processus de mapping orienté graphe 50

4.2 Processus de passage Logique orienté colonne Physique HBase 52

4.3. Proposition d'une algèbre OLAP adaptée au modèle NoSQL orienté colonnes 53
4.3.1. Opération NoSQL-SLICE 54
4.3.2. Opération NoSQL-DICE 55
4.3.3. Algorithme opérateur/opération 55

Chapitre 5 Implémentation de l'environnement OLAP adaptée au modèle NOSQL .. 59

Introduction 60

6.1 Environnement de développement 60
6.1.1. Matériel 60
6.1.2. Plateforme 60
6.1.3. Langage de programmation 61
6.1.4. Outils et librairies 61

6.2. Présentation de jeu de données 61

6.3. Présentation de l'application OLAP-HBase 62

Conclusion générale et perspective : ***67***

Listes d'abréviation ***69***

Bibliographie ***70***

Table des figures

Figure 1.1 Architecture classique des systèmes d'aide à la décision 5

Figure 1.2 : Exemple d'analyse des ventes d'une chaîne de magasins informatiques……7

Figure 1.3: Schématisation multidimensionnelle de l'exemple du cube de données précédent. 8

Figure 1.4 : Hiérarchie de la table de dimension magasins.....9

Figure 1.5 : Représentation de modèle R-OLAP.....10

Figure 1.6 : Représentation de modèle M-OLAP.....10

Figure 1.7 : Représentation de modèle H-OLAP 11

Figure 1.8 Exemple du Slice.....12

Figure 1.9 Exemple du DICE.....12

Figure 1.10 : Exemple du ROLL UP.....13

Figure 1.11 : Exemple du Drill Down.....13

Figure 1.12: PIVOT des axes température et time.....14

Figure 1.13: Exemple du PIVOT.....14

Figure 2.1: L'architecture HDFS.....18

Figure 2.2: Architecture MapReduce.....19

Figure2.3 : Représentation de la base de données des ventes en relationnel…… 20

Figure 2.4 : Organisation de la table livre dans un modèle clé-valeur.....21

Figure 2.5 : Organisation de la collection de documents vente dans un modèle orienté-documents.....21

Figure 2.6 : Organisation des ventes dans un modèle orienté graphes 22

Figure 2.7 : Organisation d'une famille de colonnes dans un modèle orienté colonnes…23

Figure 3.1 Exemple de document par traduction plate……33

Figure 3.2 Exemple de document par traduction imbriquée……34

Figure 3.3 Exemple de document par traduction hybride. 35

Figure 3.4 Exemple de document par traduction imbriquée.....36

Figure 3.5 Représentation de la première transformation. 37

Figure 3.6 Représentation de la deuxième transformation.....38

Figure 4.1 : Modèle en étoile du besoin d'analyse vente.....43

Figure 4.2 : Représentation graphique d'une ligne de la table de faits vente dans un modèle clé valeur……..44

Figure 4.3 : Exemple d'une ligne de la table de faits vente dans le modèle orienté colonnes……47

Figure 4.4 : Contenu de la collection C…….50

Figure 4.5 : Un schéma représentatif de la table de faits vente en orienté graphe……..51

Figure 5.1: Illustration du fichier CSV utilisé comme dataset……..……62

Figure 5.2 : Importations des données vers le HDFS de Hadoop…….62

Figure 5.3 : Importations des données vers HBase……...……62

Figure 5.4 : Capture de l'interface Sujet D'analyse……..63

Figure 5.5 : construction d'une requête OLAP…….....64

Figure 5.6 : Représentation de la partie filtre……..64

Figure 5.7 : Résultat de l'exécution de la requête SLICE……65

Figure 5.8 : Résultat de l'exécution de la requête DICE…….65

Figure 5.9 : Résultat de la requête SLICE dans MSSQL……...66

Liste des tables

Tableau 2.1 **:** Comparatif des modèles NoSQL ……..……24

Tableau 3.1 : Comparaison des travaux connexes……...38

Introduction générale

Contexte

Le monde numérique ne cesse de se développer très rapidement et les données ainsi générées deviennent de plus en plus complexes en volume (téraoctet à pétaoctet), en variété (structurée et non structurée et hybride) et en vélocité (grande vitesse de croissance). C'est dans ce contexte que le Big Data est né.

Les modèles relationnels classiques ont montré leurs limites quant au stockage et à la gestion des Big Data [Leavitt, 2010]. En effet, ce sont les grands acteurs du web tels que Yahoo, Google, Facebook, Twitter et LinkedIn qui ont signalé, en premier, les limites du modèle relationnel. Le constat était que les Systèmes de Gestion de Bases de Données Relationnels (SGBDR) ne sont pas adaptés aux environnements distribués requis par les volumes gigantesques de données.

Afin de répondre à ces différentes problématiques, les bases de données NoSQL (Not Only SQL) sont apparues. En effet, grâce à leur flexibilité et leur souplesse, ces bases de données non relationnelles vont permettre de gérer de gros volumes de données hétérogènes sur un ensemble de serveurs de stockage distribués, avec une capacité de montée en charge très élevée

D'autre part, les SGBD NoSQL suscitent un intérêt particulier, celui de revisiter plusieurs aspects déjà cernés sur les environnements traditionnels basés sur le modèle relationnel en les adaptant aux nouveaux environnements.

Problématique

L'avènement des modèles de données NoSQL constitue une voie intéressante pour la construction des entrepôts de données multidimensionnels capables de supporter des données à grande échelle, d'où la nécessité de revisiter les principes de modélisation des cubes de données multidimensionnels dans ces nouveaux environnements.

En revanche, le système NoSQL ne dispose pas d'une algèbre OLAP qui lui est adaptée ; il devient donc impératif de concevoir un nouvel environnement OLAP approprié basé sur un modèle NoSQL.

Objectif

Dans le cadre de notre projet nous nous sommes intéressées à la redéfinition d'une algèbre OLAP adaptée au système de stockage NoSQL. En termes plus concrets, il s'agira de concevoir et d'implémenter les opérateurs OLAP usuels tels que DICE et SLICE déjà définis et largement utilisés dans un environnement relationnel, sur une structure de données NoSQL.

Par la suite, nous nous sommes penchées sur la mise en œuvre d'une application d'analyse basée sur une interface conviviale permettant aux utilisateurs de visualiser le résultat des requêtes.

Structuration du mémoire

Notre mémoire s'articule autour de quatre chapitres à savoir :

Le premier chapitre est consacré à la présentation des environnements OLAP, plus précisément les concepts de cubes de données, table de faits, mesures, tables de dimensions, hiérarchies des dimensions et l'algèbre OLAP. Dans la deuxième partie nous avons défini les données massives (Big Data) selon les règles existantes, ensuite nous avons défini le système NoSQL et nous avons indiqué ses quatre modèles. Enfin, dans la troisième partie nous avons évoqué quelques exemples des systèmes de gestion de données NoSQL qui existent.

Dans le deuxième chapitre, nous avons présenté une étude de l'existant dans le domaine de la représentation des données multidimensionnelles dans les différents modèles NoSQL.

Le troisième chapitre a été consacré à la description du processus de mappage d'un modèle conceptuel (modèle en étoile ou flocon) vers les différents modèles NoSQL logiques du cube de données.

Le quatrième chapitre est dédié à l'ultime étape qui nous a conduit à la concrétisation des opérateurs OLAP basés sur une structure NoSQL orientée colonnes sur le SGBD HBase avec le langage de programmation python.

Enfin, nous terminons ce mémoire par une conclusion générale qui sera l'occasion de faire le bilan de notre travail et de présenter quelques perspectives.

Chapitre 1 : Entrepôt de données et analyse en ligne OLAP

Introduction

Si aujourd'hui les systèmes d'information sont devenus de plus en plus complexes et diversifiés, c'est en raison notamment de l'émergence de nouvelles technologies, ainsi que de l'accroissement continu de la volumétrie des données numériques, de la multiplicité des sources de données et de leur hétérogénéité. [Bradji, 2012]

C'est dans ce contexte que l'informatique décisionnelle (en anglais «Decision Support Systems») a vu le jour, elle permet aux entreprises de transformer des données brutes en informations précieuses, plus pertinentes et accessibles, leur facilitant ainsi la prise de décision et d'avoir une vision globale sur les informations circulant dans leur organisation [Rmdani K, Ramdani R, 2012].

1.1. Les systèmes d'aide à la décision

L'architecture des systèmes d'aide à la décision est structurée en trois couches [Teste 2009] [Bimonte,Pinet, 2012] comme l'illustre la Figure 1.1.

-La première correspond au chargement les données hétérogènes distribués grâce au processus d'Extraction de Transformation et de Chargement communément appelé ETL (Extract-Transform-Load), En effet il permet d'extraire les données à partir des systèmes sources, les préparer et les transporter vers l'entrepôt.

-La deuxième est constitué d'un espace de stockage pour l'entreposage des données (data Warehouse).

-La troisième correspond au système OLAP qui fournit une réponse rapide à des requêtes analytiques à partir d'entrepôts de données.

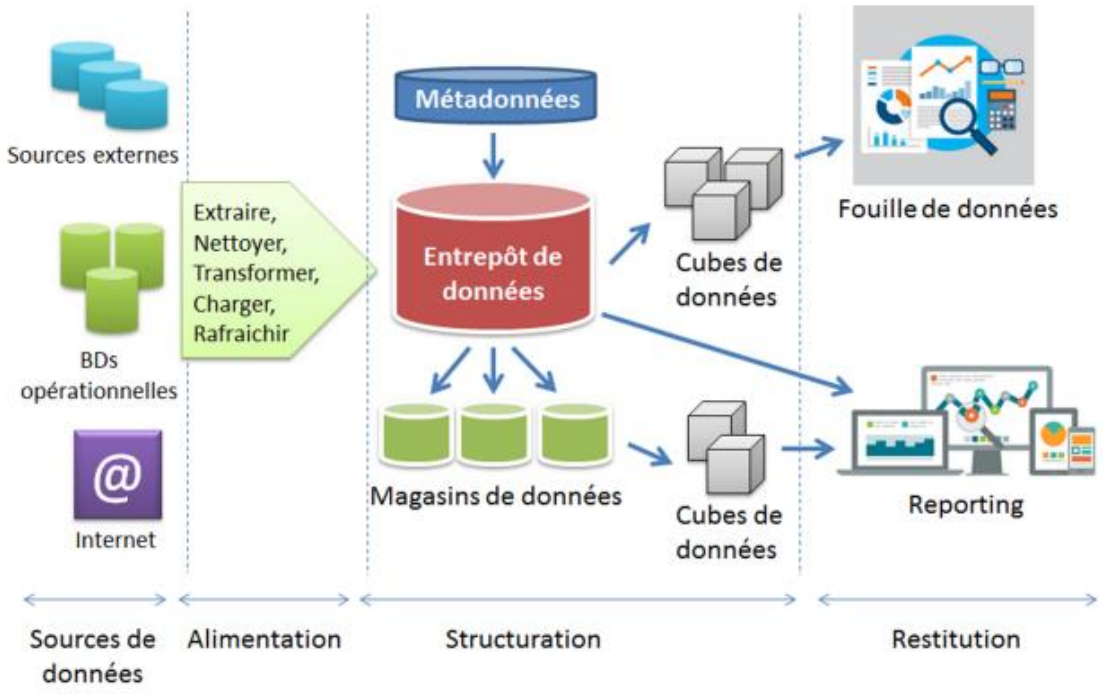

Figure 1.1 Architecture classique des systèmes d'aide à la décision. [Bala, 2017]

1.2. L'entrepôt de données

1.2.1. Définition et caractéristiques

Bill Inmon [Inmon, 1996] a défini l'entrepôt de donné comme : « une collection de données orientées sujet, intégrées, variant selon le temps et non volatiles, qui sert de support au processus de prise de décision aux acteurs du management (les décideurs) ». Plus précisément la collection de données est :

- Orientée sujet : les données constituent des granules d'information concernant des sujets d'analyse plutôt que les opérations de gestion se déroulant au sein de l'entreprise.

- Intégrée : les données sont centralisées dans un entrepôt à partir d'un ensemble de sources de données variées. Les données sont fusionnées et agencées au sein d'une vision cohérente.

- Variant selon le temps : Toutes les données d'un entrepôt sont identifiées par des périodes temporelles spécifiques. On parle d'historisation des données [Kimball,1996], [Inmon, 1996], [Teste, 2000]

- Non volatile : Les données d'un entrepôt sont stables. Il est possible d'ajouter des données, mais on ne modifie pas les données déjà intégrées. Il est toutefois possible de les archiver.

1.2.2. Concept de base

La modélisation des données décisionnelles s'appelle modélisation multidimensionnelle ou modélisation dimensionnelle. Le formalisme utilisé est basé sur deux concepts clés : le fait et la dimension.

- **Fait :** est un point de vue d'analyse. Il s'agit d'une table appelée table de fait qui contient deux types de champs : les champs qui correspondent à des mesures que l'on souhaite analyser et les champs qui correspondent à des clés renvoyant aux tables de dimension.
- **Mesures :** c'est l'indicateurs d'analyse qui sera analysées en fonction des axes d'analyse.
- **Dimension :** est un axe d'analyse. Il s'agit d'une table qui accompagne la table de fait. Elle est composée d'attributs, appelés paramètres, chaque dimension est définie par sa clé primaire qui assure l'intégrité référentielle avec la ou les tables de faits à laquelle elle est liée.
- **Hiérarchie des dimensions** : les tables de dimensions sont agencées de manière hiérarchique, modélisant ainsi les différents niveaux de détails des axes d'analyse.
- **Cube de donnée :** Un cube de donnée ou bien un magasin de donnée est un extrait de l'entrepôt qui correspond à un besoin d'analyse particulier. Il s'agit d'une méthode permettant de stocker les données sous forme multidimensionnelle, notamment pour le reporting. En général, ces cubes sont pré-résumés pour accélérer le temps de requête par rapport aux entrepôts de données. La figure 1.2 représente un exemple d'un cube de données

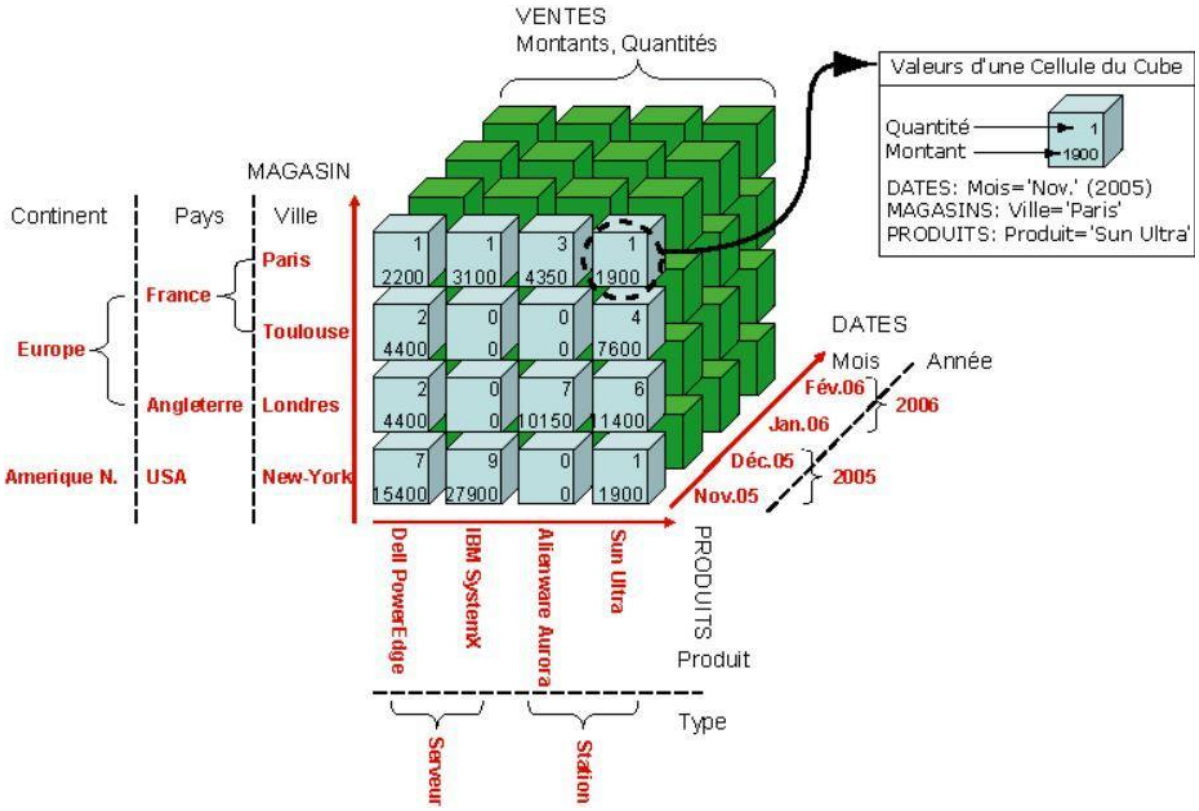

Figure 1.2 : Exemple d'analyse des ventes d'une chaîne de magasins informatiques. [Tournier, Ronan, 2007]

Pour concevoir des cubes de données des structures ont été définies pour la modélisation de sujets d'analyse en faits, et d'axes d'analyse en dimensions [Kimball, Ross, 2011].

1.3. Modélisation multidimensionnelle

Concevoir un système décisionnel nécessite une phase de modélisation des données multidimensionnelles. Plusieurs approches ont été proposées selon trois niveaux d'abstraction conceptuelle, logique et physique

- Conceptuel : Ce niveau d'abstraction consiste à représenter l'espace de données multidimensionnelles indépendamment des techniques informatiques.

- Logique : Ce niveau d'abstraction désigne une technique de modélisation (relationnel, objet, NoSQL, etc.)

− **Physique :** Ce niveau d'abstraction correspond à un logiciel particulier choisi dans la technologie logique adoptée (Oracle, PostgreSQL, MongoDB, HBase…).

1.3.1 Modélisation conceptuelle

Différents concepts sont définis pour représenter les données multidimensionnelles. Les sujets d'analyse (faits), regroupent un ensemble d'indicateurs (mesures). Les valeurs de ces indicateurs sont observables selon des axes d'analyse (dimensions).

L'exemple en cube dans la figure 1.2 est représenté en modèle multidimensionnelle dans la figure 1.3

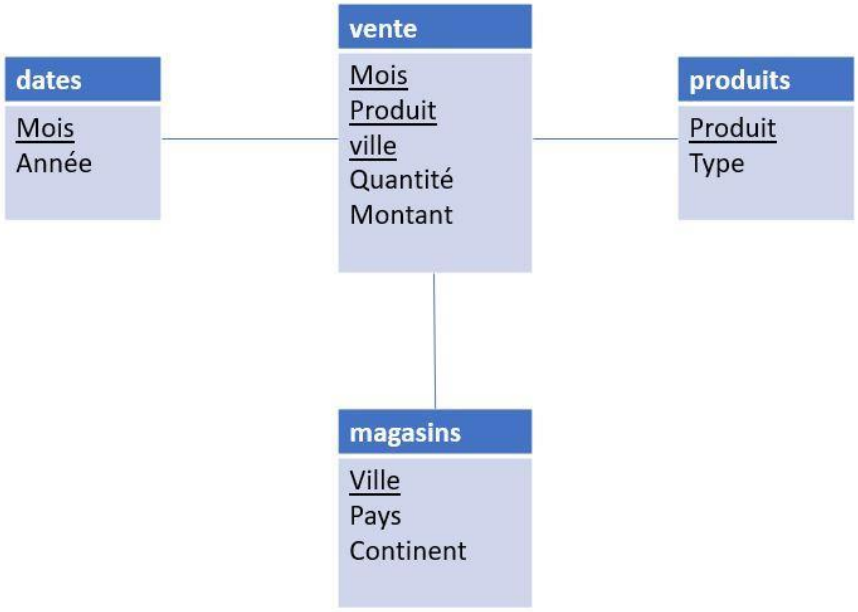

Figure 1.3: Schématisation multidimensionnelle

Ces dimensions sont constituées de différents niveaux de détail, eux-mêmes organisés en hiérarchies ; par exemple, comme le montre la figure 1.3.1 nous pourrions analyser le fait vente au travers des mesure montant et quantité. Ces mesures pouvant être observés en fonction d'une dimension magasin constituée de trois niveaux de détails (ville, pays, continent) organisés au sein d'une hiérarchie définissant la ville comme un niveau de détail inférieur au pays, lui-même inférieur à continent. Ces différents concepts permettent de concevoir des schémas multidimensionnels, appelés constellation. Les dimensions peuvent ainsi être partagées entre les faits. Un cas particulier consiste à ramener la constellation à un seul fait, on parle alors de schéma en étoile (star schema) [Kimball, Ross, 2011]. La Figure 1.3 présente un exemple de schéma en étoile.

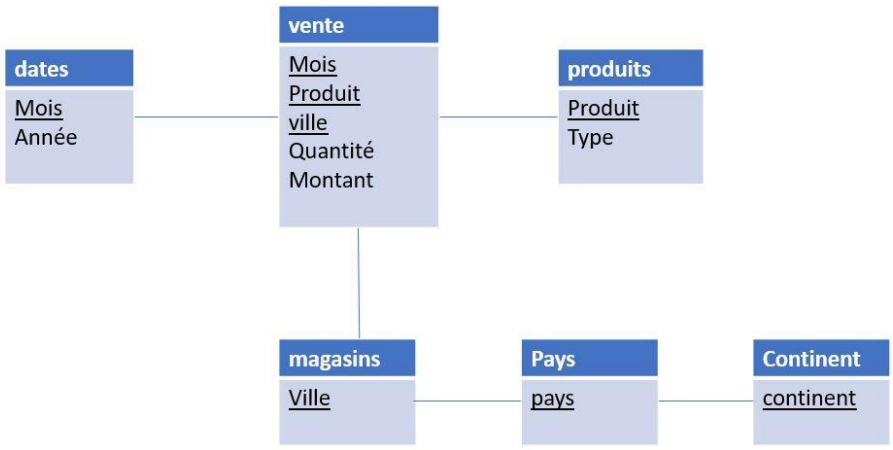

Figure 1.4: Hiérarchie de la table de dimension magasins

Dans la figure 1.4 nous avons représenté la hiérarchie de la table de dimension magasins.

1.3.2 Modélisation logique

Au niveau logique, plusieurs possibilités sont envisageables pour la modélisation multidimensionnelle. En effet, il est possible d'utiliser soit, un système de gestion de bases de données (SGBD) existant tel que les SGBD relationnels et dans ce cas l'environnement OLAP utilisé est le R-OLAP ; soit un système de gestion de bases de données multidimensionnelles (M-OLAP) qui stocke les données des faits et des dimensions de façon multidimensionnelle. Enfin, un système qui utilise en même temps l'approche R-OLAP (pour les dimensions) et M-OLAP (pour les faits) s'appelle modèle hybride (H-OLAP)

− L'approche R-OLAP consiste à utiliser le modèle relationnel pour représenter un schéma en constellation [Vassiliadis, Sellis, 1999] [Morfonios et al., 2007]. Elle est de loin l'approche la plus utilisée. Ce modèle traduit chaque fait dans une table appelée table de fait. Chaque dimension est traduite dans une table appelée table de dimension. Dans la table de fait on retrouve les attributs représentants les mesures d'activités, les attributs et les clés étrangères permettant la relation avec chacune des tables de dimensions. La table de dimension est constituée des paramètres et de la clé primaire.

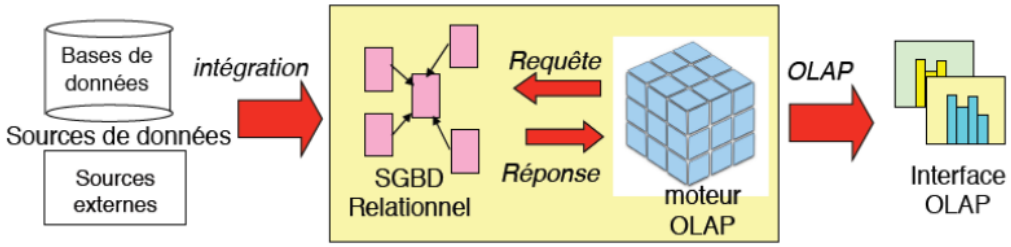

Figure 1.5 : Représentation de modèle R-OLAP [Ludovic ,Laure,2017]

− L'approche M-OLAP consiste à utiliser un système où les données sont organisées sous forme de tableaux multidimensionnels formant des hypercubes de données. Chaque arrête de l'hypercube correspond à une dimension et les cellules correspondent au fait.

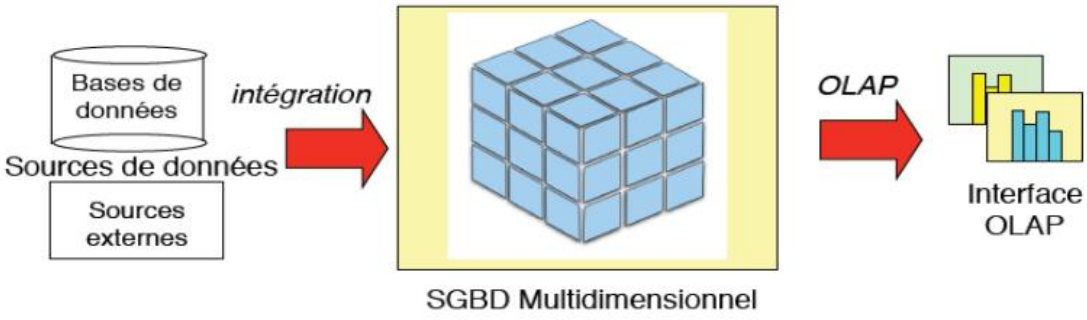

Figure 1.6 : Représentation de modèle M-OLAP [Ludovic, Laure,2017]

− L'approche H-OLAP vise à cumuler les avantages des deux approches précédentes. Les données agrégées sont stockées sous formes multidimensionnelles tandis que les données détaillées sont stockées dans des structures relationnelles.

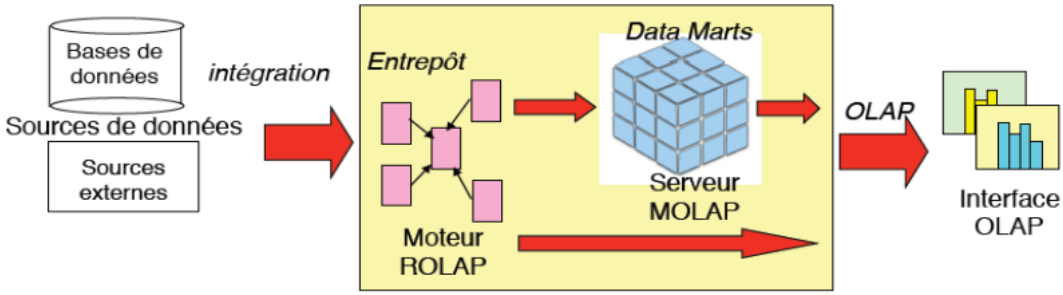

Figure 1.7 : Représentation de modèle H-OLAP [Ludovic, Laure,2017]

1.4. L’analyse en ligne des données OLAP

1.4.1 Définition

En 1993, E.F. Codd, suggère l’emploi de systèmes qui permettent d’améliorer les processus de prise de décision par la consultation et l’analyse de grandes masses de données : les systèmes d’analyse en ligne, « On-Line Analytical Processing » (OLAP) [Codd, 1993]

Ces systèmes consistent à visionner des indicateurs définis par différents axes d’analyse. Cette vision multidimensionnelle des données peut être vue comme un cube de données [Gray, et al., 1996].

1.4.2 L’algèbre OLAP

Les systèmes OLAP offre une algèbre multidimensionnelle permettant de manipuler les cubes de données et la navigation à travers les cellules et les niveaux des différentes dimensions. [Rafanelli, 2003]. Les opérateurs standards de l’algèbre OLAP sont : *SLICE, DICE, ROLL-UP, DRILL-DOWN, PIVOT, etc.* dans cette partie nous allons définir les cinq opérateurs les plus connu.

-SLICE : sélectionnent une seule valeur pour l'une de ses dimensions et crée un sous-ensemble de données. Il est illustré dans la figure 1.8.

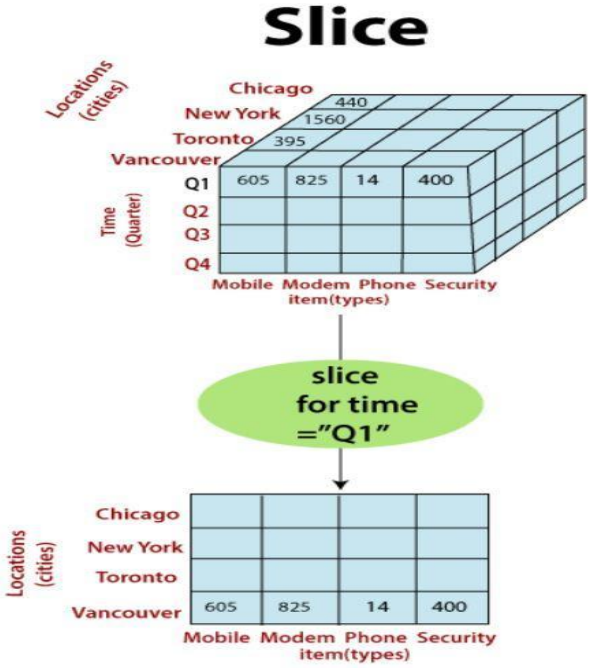

Figure 1.8 Exemple du Slice [Dharmendra, 2021]

DICE : comme illustré dans la figure 1.9 DICE sélectionnent une ou plusieurs valeurs pour un minimum de deux dimensions et crée un sous-ensemble du cube

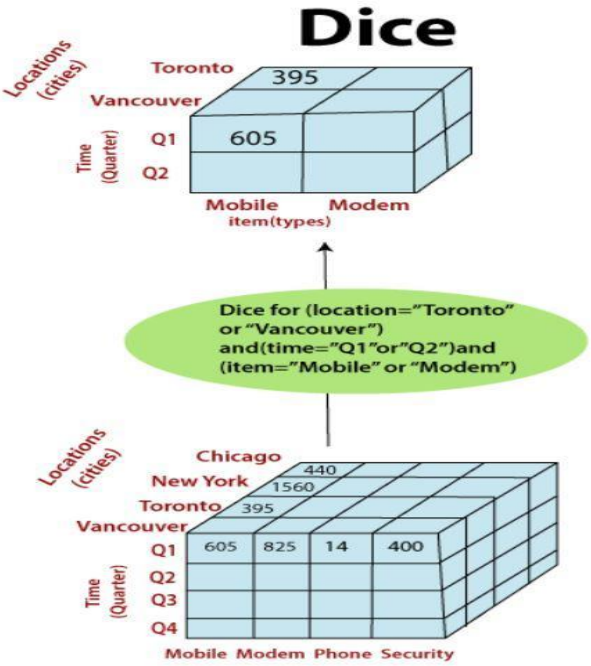

Figure 1.9 : exemple du DICE [Dharmendra ,2021]

ROLL-UP : consiste à remonter d'un niveau dans une hiérarchie de dimension vers un niveau plus agrégé. Une représentation de ROLL-UP est illustré dans la figure 1.10.

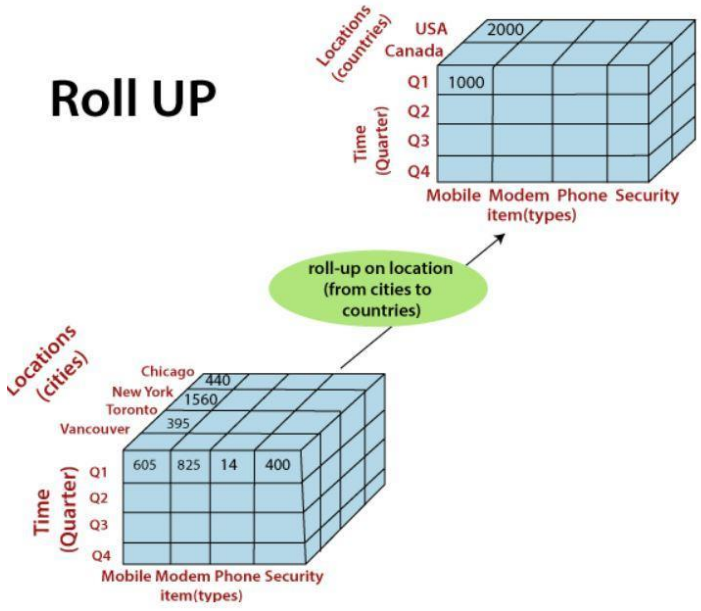

Figure 1.10 : exemple du ROLL UP [Dharmendra ,2021]

DRILL-DOWN : permet de descendre dans une hiérarchie de dimensions vers un niveau plus détaillé. Comme le montre la figure 1.11.

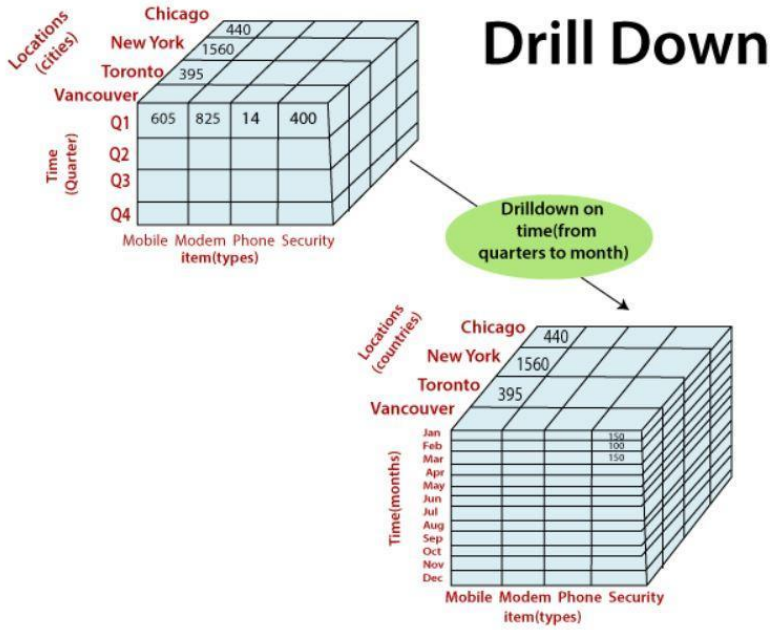

Figure 1.11: exemple du Drill Down [Dharmendra ,2021]

PIVOT : Le pivot peut désigner l'opération triviale qui consiste à permuter les axes du cube de donnée. La Figure 1.12 et 1.13 représente l'opérateur PIVOT

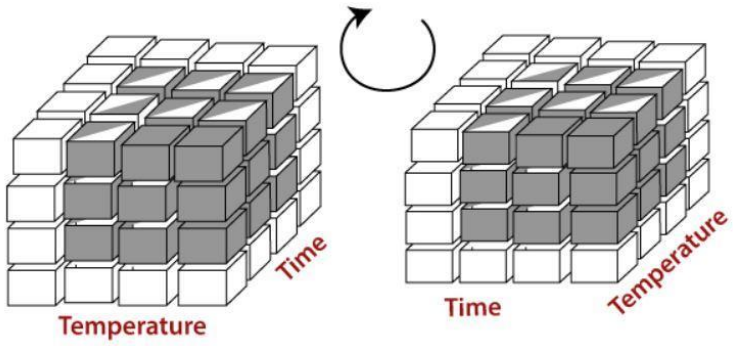

Figure **1.12** : PIVOT des axes température et time [Dharmendra ,2021]

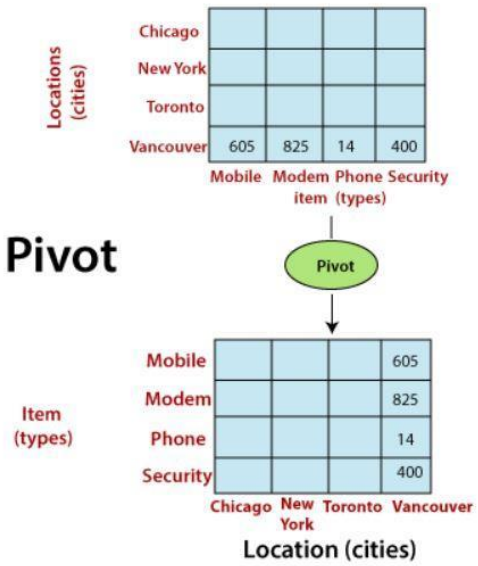

Figure 1.13: Exemple du PIVOT [Dharmendra ,2021]

Chapitre 2 : Les données massives : « Big Data »

2.1. Définition et caractéristiques

Le Big Data est une collection de données caractérisée par une complexité selon des dimensions en Vs (Volume, Variété, Vélocité…), nous présentons dans cette section les trois règles selon lesquelles le Big Data est définie « 3V », « 4V » ou en « 5V ».

Règle des « 3V » : la description du Big Data est établie dans une étude réalisée en 2001 par l'analyste du Groupe Gartner Douglas Laney [Douglas, 2001]. Elle définit les défis reliés à la croissance des données comme tridimensionnel, il s'agit du volume, de la variété et de la vitesse. En 2012, Gartner donne une définition plus détaillée des mégadonnées comme suit : « Les Big Data sont des données volumineuses, très variées, générées et traitées à grande vitesse. Ces données exigent des formes efficaces et innovantes de traitement de l'information pour permettre une meilleure prise de décision » [Douglas, 2001].

Règle des « 4V » : les analystes d'IDC 2 (International Data Corporation), ont décrit dans le rapport de 2011 [Gantz, Reinsel, 2011] quatre dimensions pour caractériser le Big Data, à savoir : le Volume, la Variété, la Vélocité et la Valeur. Gantz et Reinsel ont défini le Big Data comme suit : « Les Technologies Big Data décrivent une nouvelle génération de technologies et d'architectures conçues pour extraire économiquement de la valeur à partir de grands volumes de données très variées, en permettant leur capture et leur analyse à grande vitesse » [Gantz, Reinsel, 2011].

Règle des « 5 V » : deux nouveaux V sont apparus en 2013 suite à une étude réalisée par [Demchenko et al., 2013] dans laquelle le groupe SNE 3 (système et réseau Engineering) donne une définition plus large du Big Data en règle 5V.

Selon Demchenko et al « Les Technologies Big Data visent à traiter des données de grand volume, grande vélocité et grande variété pour extraire de la valeur, assurer une forte véracité des données originales et obtenir des informations qui exigent des formes novatrices de traitement des données, afin d'améliorer la prise de décision et le contrôle des processus » [Demchenko et al., 2013].

2.2 Paradigmes et technologies des Big Data

2.2.1. Hadoop

Hadoop est un framework open source développée en java, faisant partie des projets de la fondation Apache depuis 2009. Il a été conçu pour :

- Stocker des volumes de données très importants ;

- Supporter des données de formats variés, structurées, semi- structurées ou non structurées.

Hadoop est basé sur un ensemble de machines formant un cluster Hadoop. Chaque machine est appelée nœud. C'est l'addition des capacités de stockage et traitement de ses nœuds qui lui assure un important système de stockage et une puissance de calcul. Le système de stockage est appelé HDFS (Hadoop Distributed File System) [Borthakur, 2008]. La puissance de calcul repose sur le paradigme de programmation parallèle MapReduce [Dean, Ghemawat, 2010].

2.2.2. Présentation de HDFS

HDFS est le composant chargé de stocker les données dans un cluster Hadoop. Basé sur une architecture « maître-esclave », comme illustré dans la figure 2.1 le système HDFS repose dans son fonctionnement sur deux types de nœuds :

- Le namenode : le nœud maître est l'orchestrateur du système HDFS, au moyen de métadonnées. Il maintient l'arborescence de tous les fichiers du système et gère l'espace de nommage, autrement dit, il gère la correspondance entre un fichier et les blocs qui le constitue. Il prend en charge également la localisation des blocs des données dans le cluster. Il n'y a qu'un namenode par cluster HDFS.

-Le secondary namenode : il sert à effectuer à intervalle réguliers des sauvegardes du namenode. Il est utilisé pour prendre la relève en cas de panne du namenode.

- Les datanode : ils servent d'espace de stockage et de calcul des blocs de données. Dans un cluster hadoop il y a donc une machine jouant le rôle de namenode, une autre machine sert de secondary namenode tandis que le reste des machines sont utilisées comme des datanodes.

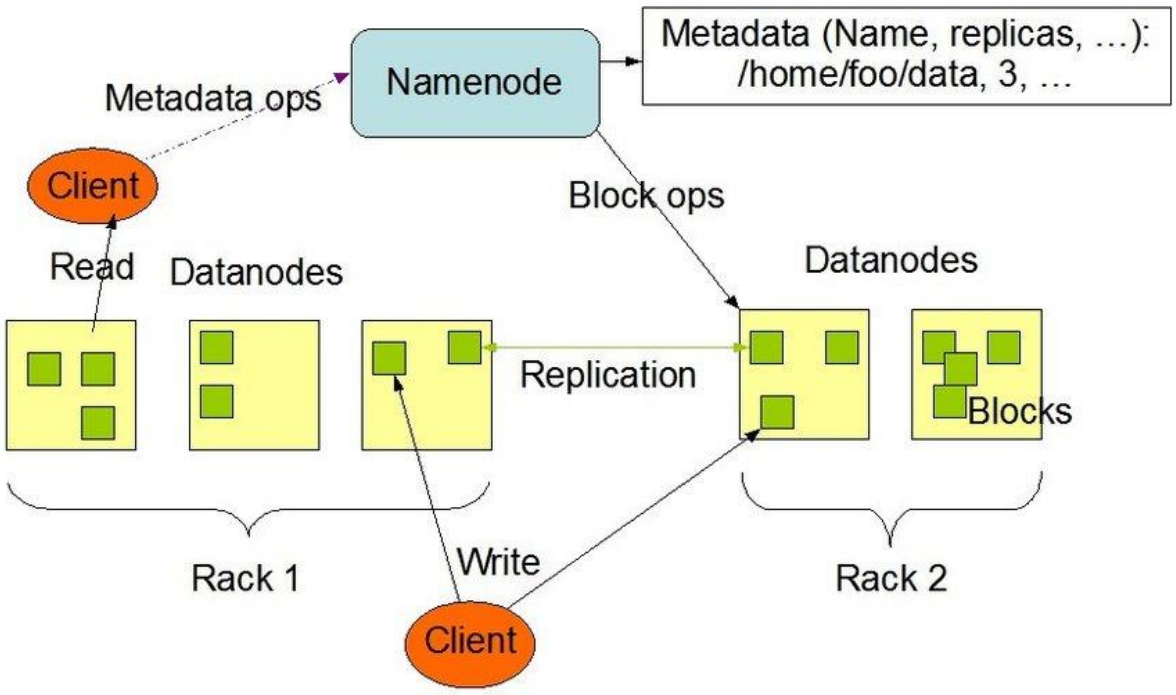

Figure 2.1 : L'architecture HDFS. [Dean, Ghemawat, 2010].

2.2.3 Le paradigme MapReduce

Introduit en 2004 par Google, le paradigme MapReduce est un modèle de programmation parallèle développé spécifiquement pour lire, traiter et écrire des volumes de données très importants dans un environnement distribué. Google l'utilise pour gérer les gigantesques tâches portant sur des téraoctet- ou pétaoctet de données [O'Malley, 2008].

Son principe de fonctionnent est très simple : il s'agit de décomposer une tâche en tâches plus petites, autrement dit, découper une tâche portant sur de très gros volumes de données en tâches identiques portant sur des sous-ensembles de ces données.

La décomposition consiste à découper le volume de données initial en N volumes plus petits, qui seront manipulés séparément. Le programme MapReduce est réalisé à partir de deux fonctions, Map et Reduce. La fonction Map découpe la tâche en un ensemble de petites tâches et les reparti sur l'ensemble des nœuds de sorte à ce que chaque petite tâche s'exécute sur le nœud qui héberge les données concernées. La fonction Reduce rassemble et agrège les résultats issus des fonctions Map parallélisées.

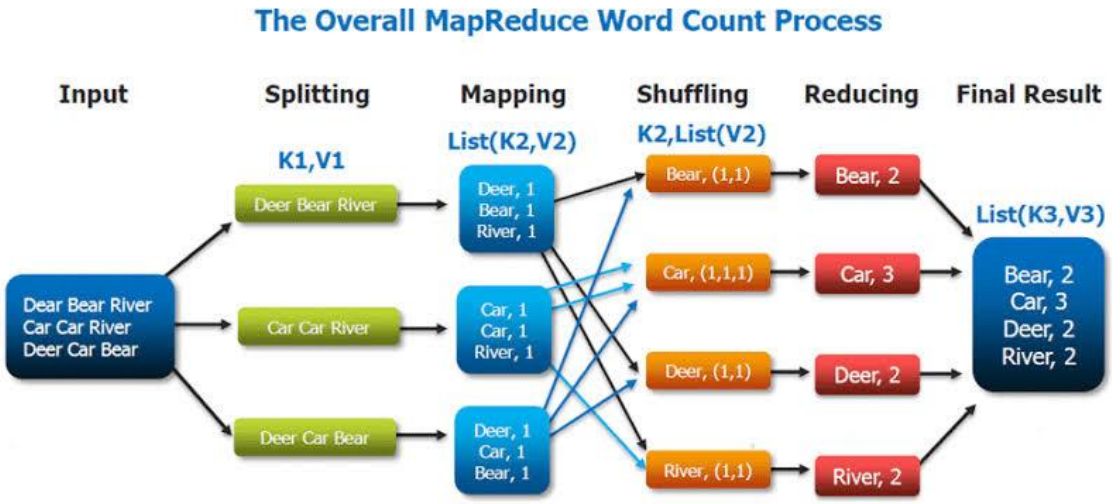

Figure 2.2: Architecture MapReduce [Bastien, 2017].

2.3 Le NoSQL

2.3.1 Définition

Le terme NoSQL a été utilisé pour la première en 2009 lors d'une rencontre sur les bases de données distribuées [Cattell, 2011] [Leavitt, 2010] [Han et al. 2011]. Ce terme désigne une nouvelle approche des bases de données capables de gérer une grande quantité d'informations, d'assurer une haute disponibilité de service et d'avoir une bonne scalabilité pour gérer la montée en charge.

Le NoSQL adopte une approche non relationnelle, son développement a été marqué par l'abandon ou le compromis fait sur les propriétés : Atomicité, cohérence, Isolation et Durabilité (ACID) des bases de données relationnelles, qualifiées jusqu'ici comme un obstacle à la scalabilité horizontale.

Il existe différentes manières de structurer les données NoSQL mais l'ensemble des solutions développées peut être divisé en quatre modèles : le modèle orienté clé-valeur, le modèle orienté colonnes, le modèle orienté documents et le modèle orienté graphes. Ces approches ont été développées pour surmonter les limites des bases de données relationnelles en permettant le stockage selon des modèles de données différents et en introduisant une plus grande flexibilité au niveau des schémas. [Sadalage, Fowler, 2012].

2.3.2 Les modèles NoSQL

Les modèles NoSQL peuvent être globalement regroupés en quatre catégories correspondant à différents paradigmes de modélisation [Morfonios, et al., 2007], que nous décrivons dans la figure 1.3.1 à l'aide de l'exemple d'une base de données relationnel qui contient les ventes des livres et ses auteurs :

Table vente (a)

ticket	date	livre
1	01/01/2022	22
1	01/01/2022	16
3	03/01/2022	16

Table livre (b)

isbn	titre	auteur
22	NoSQL	001
16	OLAP	002

Table auteur (c)

id	Nom	Prénom
001	Bruchez	Rudi
002	Thomsen	Erik

Figure 2.3 : Représentation des tables Vente (a), Livre (b) et Auteur (c) de la base de données relationnelle traitant sur les ventes.

a- Le modèle clé-valeur : Il s'agit du modèle non-relationnel le plus simple. Il organise les données sous forme d'un couple composé d'une clé unique et une valeur, où la clé est le point d'entrée unique qui permet d'accéder à la donnée. [El Malki, 2016]

A titre d'exemple la figure 1.3.2 représente la table livre dans un modèle clé valeur

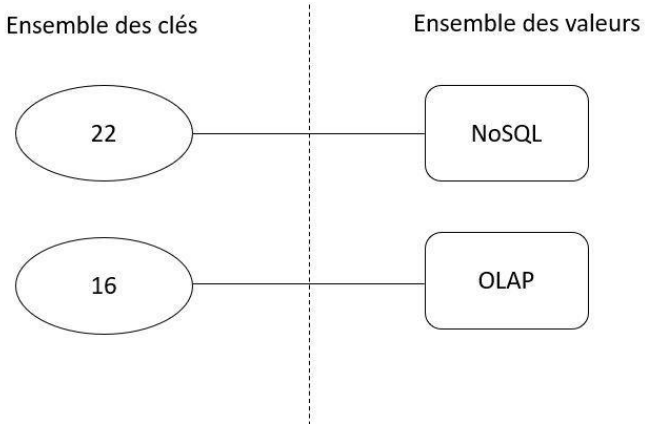

Figure 2.4 : Organisation de la table livre dans un modèle clé-valeur

b- Le modèle orienté documents : Il s'agit d'une extension du modèle clé valeur de façon à ce que les données sont organisées en collections de documents. Un document est identifié par une clé à laquelle correspond un agrégat de couples clé-valeur qui peuvent être hiérarchisés. Cela est schématisé à l'aide de la figure 1.3.3. [Brahim, 2018]

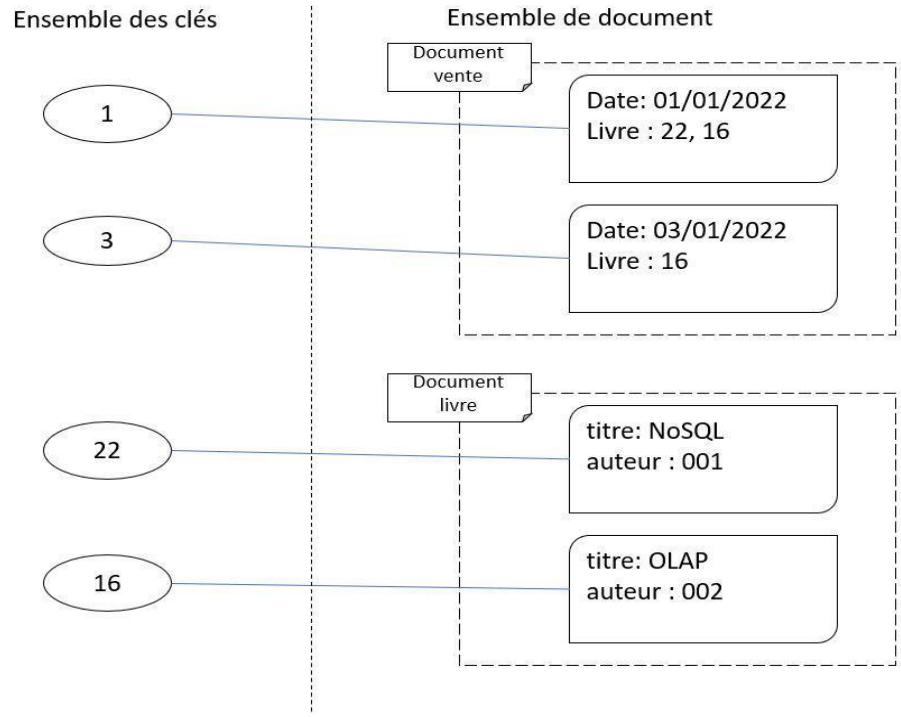

Figure 2.5 : Organisation de la collection de documents vente dans un modèle orienté-documents.

c- Le modèle orienté graphes : est un modèle spécialisé dans la structuration de données en nœuds et relations formant un graphe. Chaque nœud représente une entité, et chaque relation (arc) représente une connexion entre les nœuds. Ce modèle est utile pour stocker et interroger des données complexes fortement liées ; c'est le cas par exemple des données issues des réseaux sociaux, les routes… [Sellami, et al., 2020]
Ce modèle nous l'avons schématisé dans la figure 1.3.4

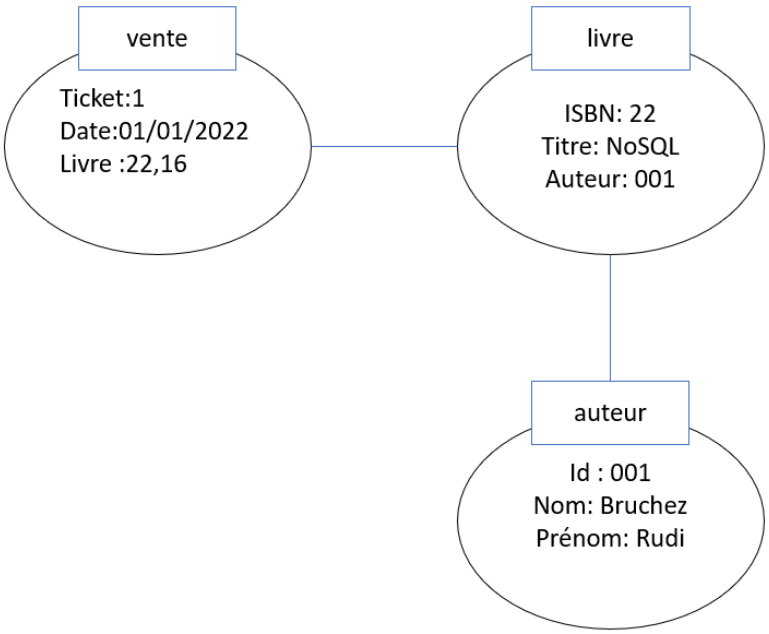

Figure 2.6 : Organisation des ventes dans un modèle orienté-graphes

d- Le modèle orienté colonnes : Il s'agit d'une extension du modèle clé valeur de façon qu'il structure les données verticalement, par regroupement de colonnes en familles de colonnes.

Les lignes possèdent un identifiant appelé Row Keys et sont composées d'un ensemble de valeurs ; chacune est associée à une colonne. Les valeurs dans chaque colonne disposent d'un Timestamp pour sauvegarder les valeurs antérieures. Ainsi, la recherche d'une valeur revient à parcourir la séquence : clé de ligne -> famille de colonnes -> colonne. [El Malki, 2016]

Le modèle orienté-colonnes est très proche du modèle relationnel, on y retrouve le principe de « table », par contre il se diffère du modèle relationnel par le fait que les

colonnes sont dynamiques et l'historisation de la donnée se fait à la valeur et non à la ligne.

Famille vente

ticket	date	livre
1	01/01/2022	22 16
3	03/01/2022	16

Famille livre

isbn	livre	nom	prénom
22	NoSQL	Bruchez	Rudi
16	OLAP	Thomsen	Erik

Figure 2.7 : Organisation d'une famille de colonnes dans un modèle orientée colonnes.

Dans le Tableau 1, nous synthétisons les caractéristiques de ces quatre modèles NoSQL ainsi que le modèle relationnel. Nous considérons 3 caractéristiques principales :

- La structure de la valeur peut être soit atomique soit composée.

- Le schéma peut être statique (schéma unique pour un ensemble d'enregistrements) ou dynamique (schéma différent pour chaque enregistrement).

- Le requêtage fournit des fonctions d'accès aux valeurs. L'acronyme CRUD représente les fonctions élémentaires de tous systèmes de gestion de données (écriture, lecture, modification, suppression). Il peut être complété par des fonctions avancées développées dans un langage propriétaire du système comme par exemple SQL, HQJ et CQL

	Relationnel	Clé-valeur	Colonnes	Documents	Graphes
Structure	atomique	atomique	atomique & composée	atomique & composée	atomique & composée
Schéma	statique	dynamique	dynamique (colonnes)	dynamique	dynamique
Requêtage	SQL	CRUD	CRUD + langage	CRUD + langage	CRUD + langage
Exemple de systèmes	Oracle, PostgresSQL	Redis	HBase, Cassandra	MongoDB, CouchDB	Neo4j, OrientDB

Tableau 2.1 : Comparatif des modèles NoSQL

2.3.4 Les SGBD NoSQL

Les plus populaires sont :

a- Riak

Riak est développé par la société Basho sur le modèle de dynamo d'Amazon, Riak est un modèle NoSQL orienté clé-valeur qui offre de bonnes performances et une simplicité d'administration. Il est adopté par les grandes entreprises telles que the water company, c'est le plus grand fournisseur météo dans le monde. Le partitionnement de la donnée se fait dans une architecture décentralisée (maître-maître) par hachage. Aucun serveur maître n'est désigné et chaque nœud est indépendant. Le client peut adresser sa requête à n'importe quel nœud. Si ce dernier n'a pas la donnée, la requête est redirigée vers le nœud contenant la donnée. [El Malki, 2016]

b- MongoDB

La solution MongoDB open source (extrait du mot humongous ce qui veut dire gigantesque) a été créée en 2009 par la société 10gen [Banker 2011] [Chodorow 2013]. Dans son modèle de stockage, les données sont stockées dans des documents. Les documents sont regroupés dans une collection où chaque document est constitué d'un ensemble de paires clé-valeur pouvant être organisées de manière hiérarchique. Un document est identifié par une clé unique (de taille maximale de 96 octets) gérée automatiquement par le système (comme elle peut être gérée par le client). Chaque document est stocké dans une représentation comprise et optimisée par MongoDB,

BJSON (Binary JSON) et peut posséder sa propre structure, on parle de schemaless. Dans son mécanisme de stockage, MongoDB stocke l'ensemble des collections dans des bases de données regroupées dans des espaces de noms. Chaque collection est stockée séparément et indépendamment sur disque. Pour les objets de taille large, MongoDB les découpe en petits segments (considérés comme des documents). Ces segments sont stockés dans une collection appelée Chunks tandis que les métadonnées des segments sont stockées dans une autre appelée Files, permettant d'identifier chaque segment de la collection Chunks. [El Malki, 2016]

c- HBase

HBase est une solution initiée par l'entreprise Powerset en 2006 et supportée par Apache depuis 2008. HBase est un moteur orienté colonnes [George 2011] [Vora 2011]. Il est considéré comme une contribution Hadoop car il s'interface avec Hadoop et ne peut pas fonctionner sans. Il utilise HDFS comme espace de stockage et utilise MapReduce pour le traitement. Il a été développé pour gérer de grandes quantités de données avec une architecture entièrement distribuée. Plusieurs distributions fournissent HBase directement dans leurs solutions telles que Cloudera et Hortonworks.

Dans son modèle de données HBase, les données sont stockées dans des tables, chaque table est un ensemble de lignes qui regroupent un ensemble de familles de colonnes. Chaque famille de colonnes regroupe un ensemble de colonnes, variable d'une ligne à l'autre. Chaque valeur de La colonne est "versionné" avec des horodatages. Chaque famille de colonnes est stockée séparément sur disque dans un fichier appelé HFile.

Pour l'interrogation des données, HBase offre peu de fonctions d'analyse (scan, get, put, delete). Il est nécessaire d'écrire ses propres fonctions MapReduce avec des langages d'interrogations externes tels que Hive, Phoenix, Java, Python... [El Malki, 2016]

d- Neo4j

Aujourd'hui, Neo4j est considéré comme le premier modèle de données de graphes au monde et est un membre potentiel de Famille NoSQL [Hoff, 2009]. Il s'agit d'un modèle de données graphique open source, robuste et basé sur disque qui prend en charge les transactions ACID et implémenté en Java.

La nature graphique de Neo4j lui attribue l'agilité et la rapidité et par rapport aux bases de données relationnelles, pour un ensemble similaire d'opérations ; il est cité

beaucoup plus rapide et surpasse le premier avec plus de 1000 x pour plusieurs potentiellement importants scénarios en temps réel [Eifrem, 2009].

Semblable à un graphique de propriétés ordinaires (simples paires clé-valeur), Neo4j le modèle de données graphique est constitué de nœuds et d'arêtes, où chaque nœud représente une entité et une arête entre deux nœuds correspond à la relation entre ces entités attachées.

Comme l'emplacement est devenu un aspect important des données aujourd'hui, la plupart des applications doivent faire face à la très des données associées, formant un réseau (ou graphe) ; les sites de réseaux sociaux en sont des exemples évidents. Contrairement aux modèles de bases de données relationnelles ; qui nécessitent des schémas initiaux qui restreignent absorption des données agiles et ad-hoc, Neo4j est un modèle de données sans schéma, qui fonctionne de manière ascendante approche qui permet une expansion facile de la base de données pour s'imprégner de données ad hoc et dynamiques. Semblable à n'importe quel autre modèle de données,

Neo4j possède son propre langage de requête déclaratif et expressif, appelé Cypher [Johnson, 2010] qui a les capacités de correspondance de modèles entre les nœuds et la relation pendant les données l'exploration et la mise à jour des données. [Sharma, Sugam, 2015]

e- Cassandra

Facebook a mis en place une solution pour traiter des données grandissantes notamment pour les besoins de stockage et de recherche dans sa messagerie [Lakshman, Malik, 2010] [Aniceto , et al., 2015]. En 2009, une solution libre est publiée, appelée Cassandra. Il s’agit d’un moteur orienté colonnes totalement distribué.

Dans son modèle de données, Cassandra stocke les données dans une table regroupant des lignes. Chaque ligne est constituée d’une ou plusieurs familles de colonnes, chacune est constituée d’un ensemble de colonnes. Une colonne est apparentée à une paire clé-valeur.

Le schéma n’est pas fixé à l’avance ; le nombre de colonnes contenues dans la famille de colonnes peut varier d’une ligne à l’autre. Les données d’une même famille de colonnes sont stockées sur le disque dans un fichier qui lui est propre.

Cassandra, diffère légèrement de la définition du modèle orienté colonnes puisqu'il ajoute la notion de super colonnes, c'est-à dire un regroupement de famille de colonnes.

Pour l'interrogation des données, Cassandra utilise le langage d'analyse CQL, proche du langage SQL. Toutefois CQL ne permet pas de faire des agrégations. Pour ce faire il faut définir ses propres fonctions en java par exemple. L'entreprise qui supporte Cassandra accorde un effort particulier à cette problématique, ce qui laisse entrevoir des évolutions possibles lors de prochaines versions.

Conclusion

Dans ce deuxième chapitre nous avons abordé sur les généralités sur les mégadonnées, paradigme MapReduce et modèles NoSQL.

Chapitre 3 : Etat de l’art

Introduction

L'objectif de ce chapitre est de présenter les travaux connexes et ce en vue de faire toute la lumière sur les propositions et contributions faites dans le domaine de l'analyse des données en ligne (OLAP) adaptée aux modèles NoSQL, nous exposons les approches de modélisation multidimensionnelle des modèles NoSQL les plus intéressantes tout en montrant :

-Le mapping de modèle conceptuel -modèle logique NoSQL

-Le mapping de modèle logique NoSQL- un modèle physique

-Les requête OLAP -NoSQL

3.1 Travaux connexes

3.1.1 Approches basées sur un modèle orienté clé-valeur

Il existe très peu de travaux sur la transformation conceptuelle multidimensionnelle dans un modèle orienté clé-valeur en raison de la simplicité de ce type de modèle.

L'approche la plus récente est celle de [Khalil 2020] qui présente trois modèles logiques

1-Modèle logique plat le fait F et ses dimensions associées $D_{1..n}$ sont transformés en une table de données clé-valeur T définie par T(Tid, TAtt, TR) où :

· T_{Id} est la clé de fait.

· T_{Att} est un ensemble d'attribut simple qui peut être une clé étrangère vers une clé dimension ou mesure

· T_R Ensemble d'enregistrements imbriqués stockant des éléments de données de dimension.

2-Modèle logique en étoile

- Le fait est mappé sur une paire clé-valeur ; la clé représente l'identifiant du fait et la valeur contient les mesures.
- Une dimension est mise en correspondance avec une paire clé-valeur enfant, et hérite de la clé de son fait correspondant, où la valeur contient les attributs de la dimension.

3-Modèle logique en flocon Ce modèle est une extension du modèle logique hiérarchique en étoile où la dimension est connectée à une ou plusieurs autres dimensions, et organisée en une structure arborescente, impliquant un seul parent pour chaque table,

afin de concevoir un schéma multidimensionnel dans ce modèle, une réplication des tables enfants des dimensions pour chaque table de faits parent est nécessaire.

Ces approches diffèrent entre

Ces trois modèles sont implémentés dans le SGBD Oracle NoSQL, chacun d'entre eux diffère en termes de formalisation et de structure. Ensuite les auteurs comparent ces modèles en fonction de deux métriques : l'espace de stockage et la performance des requêtes. Les résultats ont montré que le modèle logique plat offre de meilleures performances.

3.1.2 Approches basées sur un modèle orienté colonne

[Dehdouh et al., 2015] ont proposé trois approches logiques pour mettre en œuvre un entrepôt de données dans un système NoSQL orienté colonnes (HBase) :

1-NLA (Normalized Logical Approach) : Cette approche propose de mapper le modèle dimensionnel de l'entrepôt de données par l'approche normalisée comme le fait du relationnel.

Pour ce faire, les modèles dimensionnels classiques sont convertis en modèles logiques relationnels.

Les dimensions et les faits sont stockés séparément sur des tables différentes. Pour assurer les liens entre ces deux entités (dimension-fait), l'identifiant de la table des dimensions est dupliqué dans la table des faits. Cependant, en l'absence de contraintes d'intégrité référentielle dans les SGBD NoSQL en colonnes, c'est la responsabilité du niveau applicatif du client de vérifier cet ordonnancement.

2-DLA (Denormalized Logical Approach) : Cette approche propose de transformer le modèle conceptuel des données en un modèle basé sur une grande structure (table) appelée BigFactTable, qui conserve les faits et les dimensions associées. A l'inverse de l'approche normalisée qui sépare les faits de leurs dimensions, cette approche favorise la dénormalisation du schéma en intégrant, dans une même table, le fait et la dimension.

Ce processus est très fréquent dans la modélisation des entrepôts de données, notamment dans la gestion des hiérarchies dimensionnelles.

Pour mapper les mesures et les dimensions dans le modèle logique, cette approche utilise l'attribut simple proposé par le modèle NoSQL orienté colonne. Ainsi, les valeurs des faits et des dimensions sont maintenant identifiées par le même identifiant (row key) de la table, il n'y a plus de jointure entre les tables lorsque l'agrégation est effectuée.

3- DLA-CF (Denormalized Logical Approach) : Cette approche propose de transformer le modèle conceptuel des données en un modèle logique NoSQL orienté colonnes, comme le fait l'approche logique dénormalisée. Cependant, cette approche utilise les attributs composites pour mapper les mesures et les dimensions au lieu des attributs simples. En effet, chaque dimension est mappée dans une famille de colonnes et les attributs appartenant à la même dimension sont rassemblés dans une famille de colonnes. Cela permet aux attributs appartenant à une dimension donnée d'être partagés dans le même espace disque, ce qui améliore le temps d'accès aux colonnes, en particulier lorsque la requête décisionnelle implique plusieurs attributs de la même dimension (hiérarchie : année, trimestre et mois) pour effectuer des agrégations.

Le passage vers le modèle physique est montré au travers d'une expérimentation particulière sur le système HBase ; les performances des trois solutions sont comparées à travers des requêtes décisionnel.

Les approches dénormalisée DLA et DLA-CF montrent de meilleures performances que NLA qui représente l'approche normalisée. En effet, le NLA utilise moins de mémoire disque, mais il est assez inefficace lorsque des requêtes avec jointures sont effectuées. Par ailleurs, nous avons constaté que le DLA-CF est plus efficace que le DLA, mais seulement lorsque les attributs de traitement des requêtes appartiennent à la même dimension.

[Boussahoua et al., 2017] ont proposé une modélisation d'un entrepôt de données NoSQL un processus de quatre étapes :

1-extraction de l'ensemble des attributs de la table de faits et des tables de dimensions trouvés dans les requêtes, en particulier ceux qui apparaissent de la requête dans les clauses « Select » et « Where », à l'exception des attributs des prédicats de jointure.

2- regroupement des attributs fréquemment interrogés, ce regroupement formera des familles de colonnes qui constituent le schéma logique de DW, en utilisant l'algorithme k-means cela permettra de définir le nombre maximum (ayant k comme

paramètre d'entrée) de groupes à construire afin contrôler le nombre de familles de colonnes qui peuvent être créées., l'algorithme k-means prend en entrée la matrice d'affinité des attributs et le numéro de cluster k, et renvoie en sortie, l'ensemble des familles de colonnes S

3- génération d'un schéma logique du DW.

4- préparation et chargement des données dans le DW.

3.1.3 Approches basées sur un modèle orienté document

Dans l'approche [Chevalier et al., 2015] les auteurs ont étudié le passage d'un modèle multidimensionnel en constellation vers des modèles NoSQL orienté-documents. Quatre processus de traduction ont été proposés :

- Le processus de traduction plat : il repose sur une dénormalisation des données du schéma en constellation. A chaque fait et ses dimensions associées correspond une collection. Chaque instance du fait est traduite par un document. Les attributs représentant les mesures du fait et les attributs de dimensions sont contenus dans un même document plat sans imbrication.

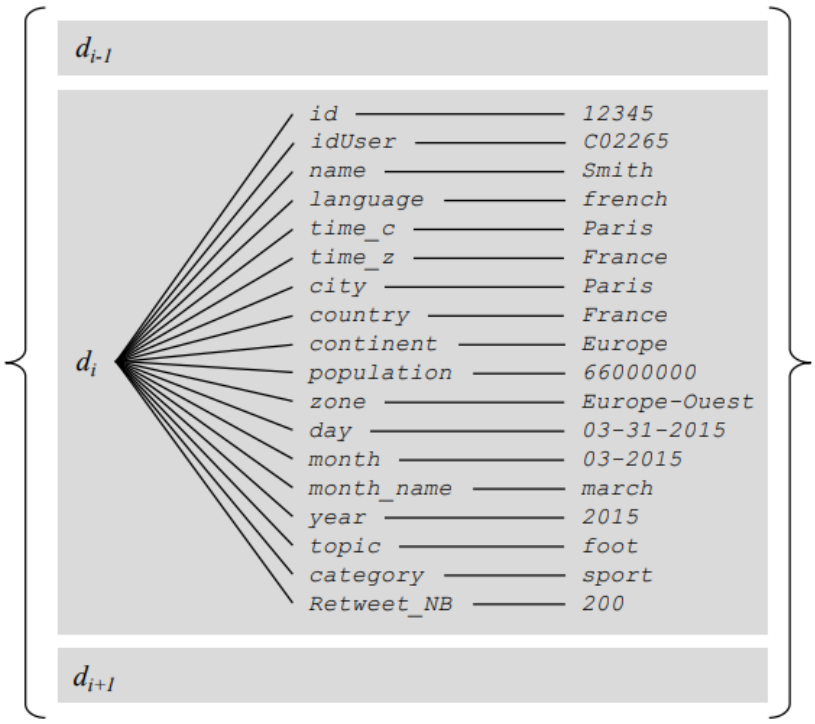

Figure 3.1 Exemple de document par traduction plate. [Chevalier et al., 2015]

Comme le montre la figure 3.1, cette implantation évite l'utilisation des jointures entre le fait et les dimensions, mais génère un important niveau de redondance par duplication des données des dimensions.

- Le processus de traduction par imbrication : il consiste à combiner l'utilisation d'attributs simples et d'attributs composés. Comme précédemment, à chaque fait et ses dimensions associées correspond une collection de même nom. Chaque instance du fait est traduite par un document. L'imbrication est utilisée pour rassembler les attributs représentant les mesures du fait. De même, les attributs de chaque dimension associée sont rassemblés dans un même attribut composé.

La Figure 3.2 Présente sous une forme graphique le document constitué par un ensemble d'attributs composés issus du fait (imbrication des mesures) et de chaque dimension associée (imbrication des attributs de la dimension).

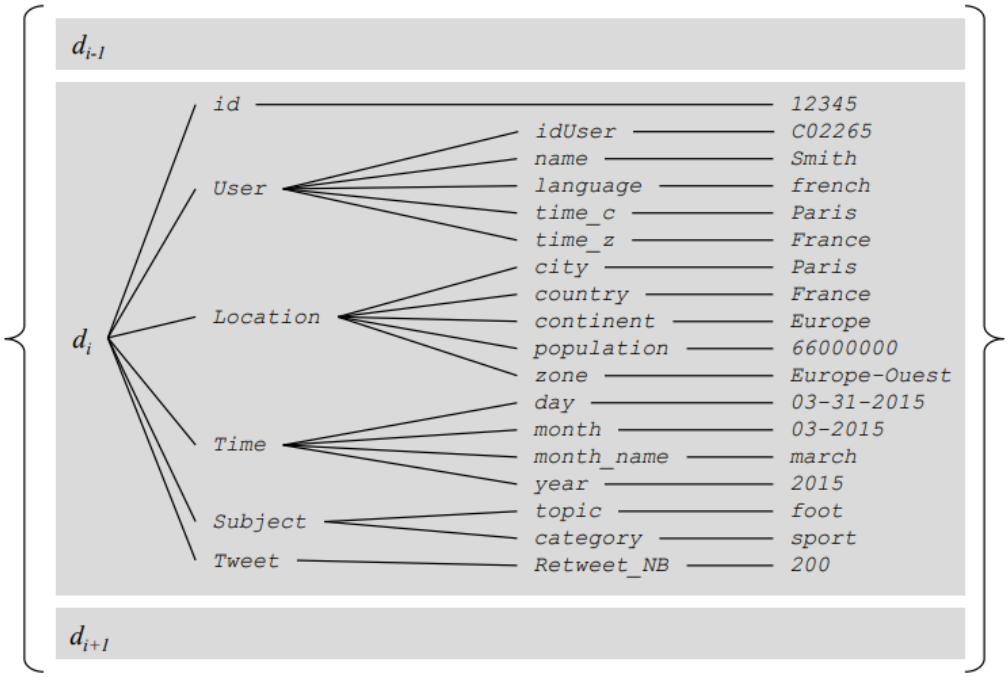

Figure 3.2 Exemple de document par traduction imbriquée. [Chevalier et al., 2015]

- Le processus de traduction hybride : Comme les approches précédentes, une traduction hybride rassemble au sein d'une même collection les données issues d'un fait et des dimensions associées. Les attributs des dimensions sont convertis en attributs simples, à plat sans imbrication, et stockés dans un document de la collection. Les mesures du fait sont converties en attributs simples, à plat sans imbrication, et stockés

dans un document de la même collection. Les documents des faits contiennent également les références aux documents représentant les dimensions associées.

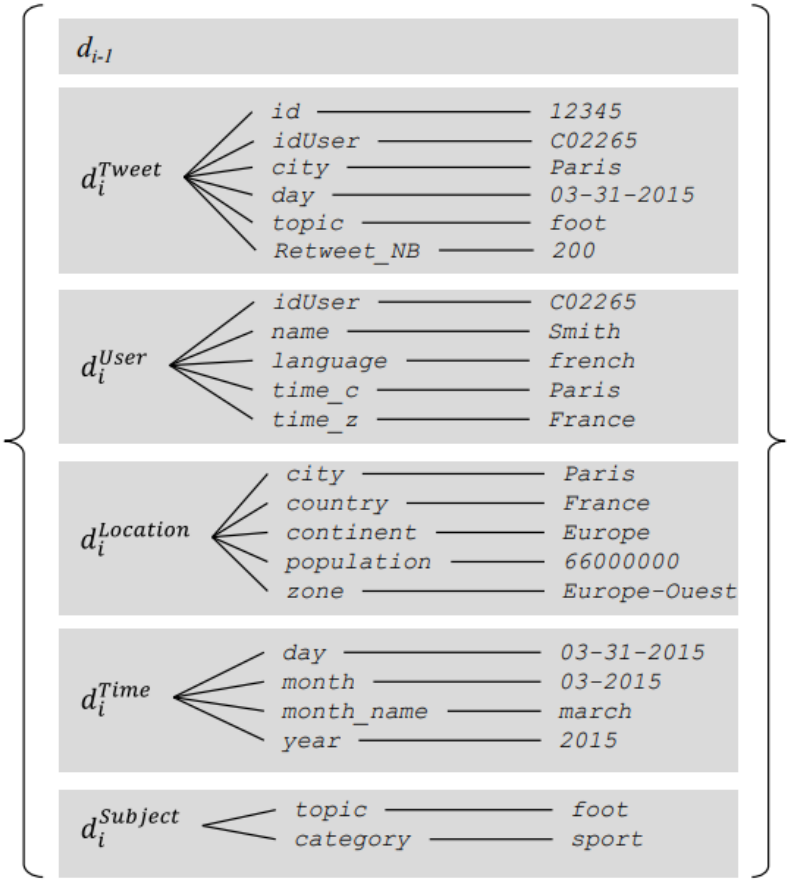

Figure 3.3 Exemple de document par traduction hybride. [Chevalier et al., 2015]

Cette implantation évite la redondance des données des dimensions des deux modèles précédents, mais introduit une hétérogénéité dans les structures des documents placés dans la même collection. Contrairement aux approches précédentes, l'approche hybride nécessite l'utilisation d'auto-jointures pour retrouver les documents de dimensions liés aux documents des faits au sein de la collection.

- Le processus de traduction éclaté : Contrairement aux approches précédentes, une traduction éclatée distribue les données au sein de plusieurs collections : les données issues d'un fait sont placées dans une première collection, et les données de chaque dimension associée sont placées dans des collections distinctes (une collection par dimension). Les attributs des dimensions sont convertis en attributs simples, à plat sans imbrication, et stockés dans un document de la collection dédiée. Les mesures du fait sont

converties en attributs simples, à plat sans imbrication, et stockés dans un document d'une autre collection. Les documents des faits contiennent également les références aux documents représentant les dimensions associées, stockées dans les collections des dimensions.

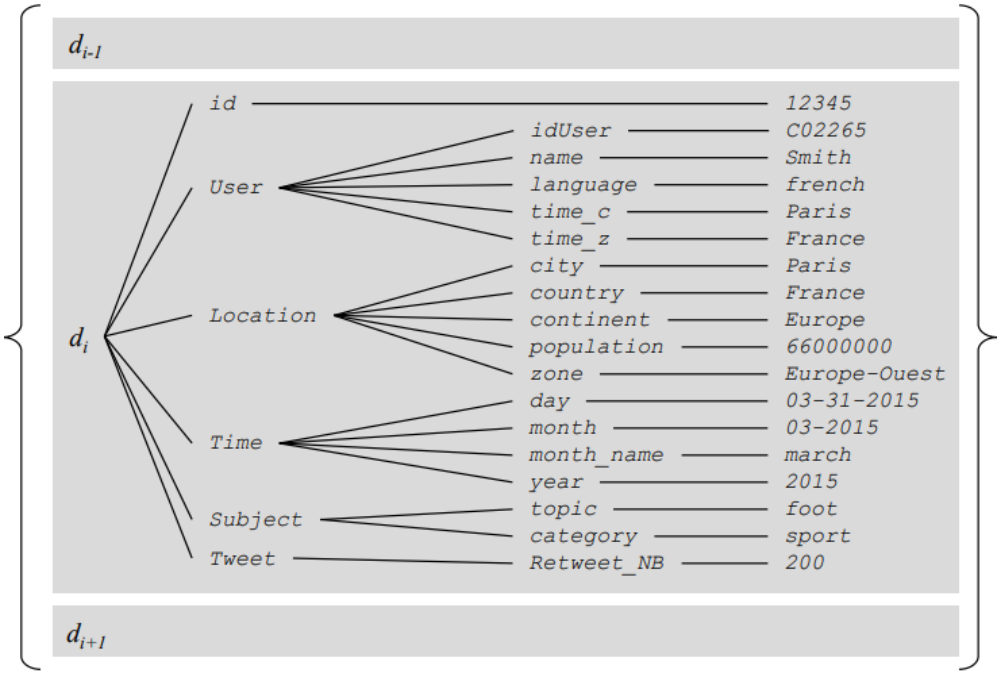

Figure 3.4 Exemple de document par traduction imbriquée [Chevalier et al., 2015]

3.1.4 Approches basées sur un modèle orienté graphe

Sellami et al [Sellami, et al., 2018] ont proposé une approche qui se compose de deux transformations.

La première transformation : chaque fait et dimension sont transformés en nœuds comme décrit par les règles suivantes

Règle.1 : Transformation des faits : Chaque fait est transformé en un nœud dont l'étiquette prend le nom du concept du modèle multidimensionnel qui est « fait » et le nom du fait comme deuxième étiquette au même nœud. Chaque mesure est transformée en propriété du nœud « fait ».

Règle.2- transformation du nom et identifiant de la dimension : Chaque dimension est transformée en un nœud dont l'étiquette du nœud prend le nom du concept du modèle multidimensionnel (dans ce cas, il s'agit de la dimension) le nom de la dimension comme deuxième étiquette au même nœud. Ensuite, l'identifiant est transformé

en une propriété dans le nœud. Enfin, tout attribut faible associé à l'identifiant est transformé en une propriété dans le même nœud.

Règle.3- Transformation des hiérarchies : Une hiérarchie est constituée d'un ensemble de paramètres et d'un lien de précédence entre les paramètres. Chaque paramètre est transformé en un nœud dont l'étiquette prend le nom du concept du modèle multidimensionnel (dans ce cas, le paramètre) et le nom du paramètre comme deuxième étiquette au même nœud. Ensuite, chaque attribut faible est représenté dans le nœud sous la forme d'une propriété. Enfin, chaque lien de précédence est transformé en une relation

Règle.4- Transformation du lien fait-dimension : Chaque lien entre fait et dimension est représenté comme une relation ayant comme nœud source le nœud modélisant le fait et comme nœud destination le nœud modélisant la dimension. La relation a pour nom « lien fait-dimension ».

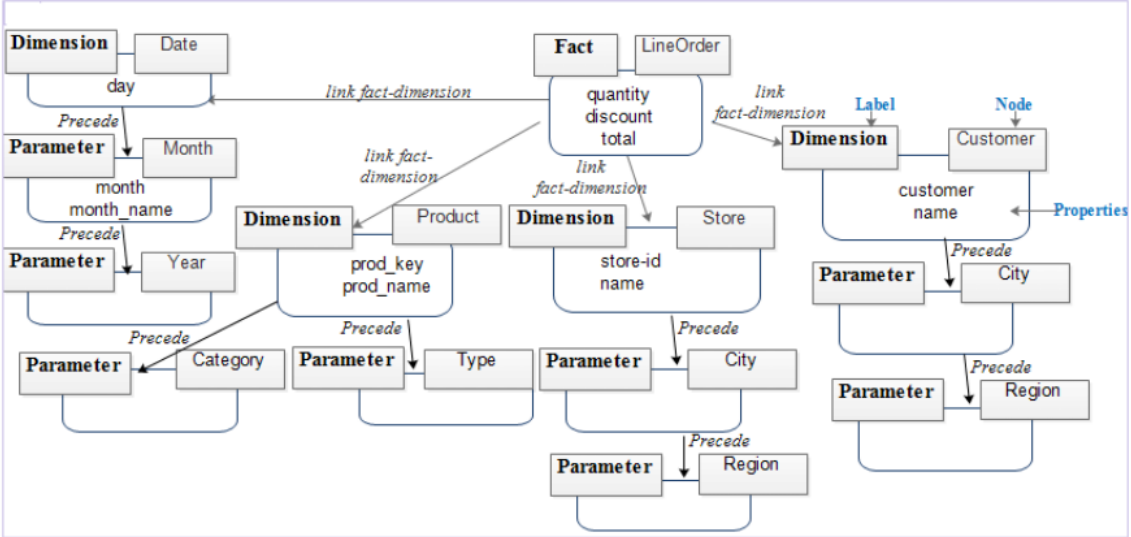

Figure 3.5 Représentation de la première transformation. [Chevalier et al., 2015]

Deuxième transformation :

Cette transformation a les mêmes règles que la première mais avec une seule différence : **Règle.4-Transformation du lien fait-dimension**, tous les identifiants de dimension sont transformés en un seul nœud, comme expliqué par la règle suivante :

Règle.5- Transformation du lien fait-dimension : Une relation a été créée entre le fait et le noeud créé avec les premiers paramètres de chaque dimension. La relation a pour nom « lien fait-iD-Dimension ».

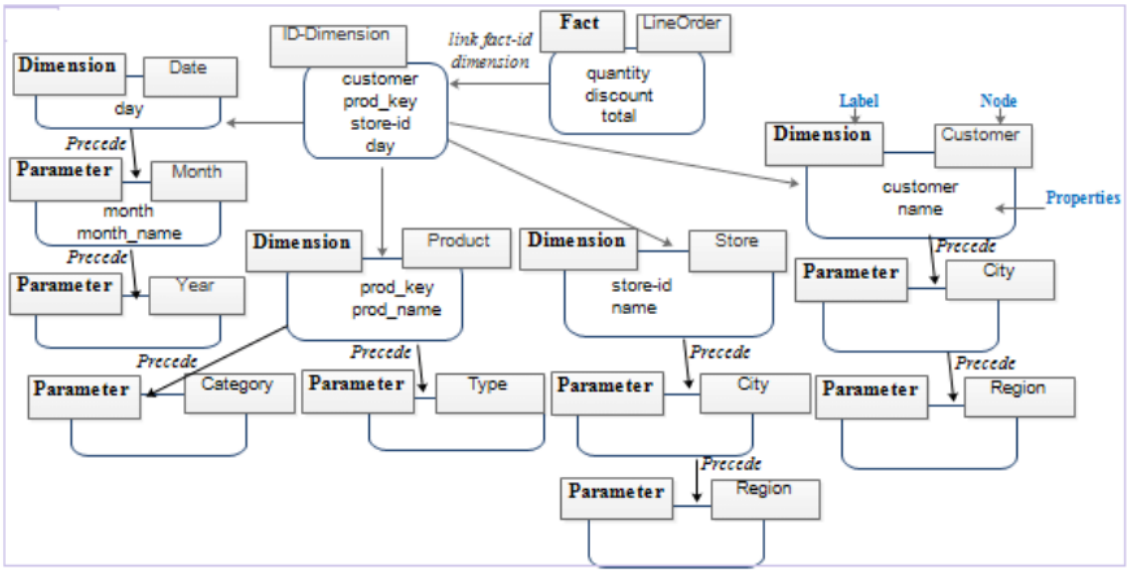

Figure 3.6 Représentation de la deuxième transformation. [Chevalier et al., 2015]

En 2014, une première approche a été proposée pour coupler le modèle orienté graphe et l'OLAP [Castelltort and Laurent 2014]. Dans cette approche les auteurs proposent de structurer les données dans le système NoSQL orienté graphes Neo4J et présentent deux formalismes pour représenter le fait et les dimensions au niveau du modèle logique orienté graphes. Le formalisme assure deux types de relations, celles liant le fait aux dimensions, et celles reliant les attributs des dimensions entre eux ; ces dernières permettent de préserver la relation hiérarchique.

3.3 Discussion

Les deux modèles les plus utilisés comme une alternative au modèle relationnel pour implanter des entrepôts de données multidimensionnelles sont les modèles orientés colonnes et documents.

Le modèle orienté colonnes est la première solution adoptée dans ce contexte. L'entreposage orienté colonnes repose généralement sur le SGBD HBase. HBase permet de gérer de gros volumes de données dans un environnement distribué. Il est implémenté dans une couche au-dessus de Hadoop (il stocke les données dans l'espace de stockage HDFS).

La second approche, basée sur le modèle orienté documents, utilise souvent le système MongoDB. Ce choix se justifie par la richesse de son lagunage d'interrogation et son moteur d'agrégation.

Dans le Tableau 2, nous synthétisons les processus de transformation directe. Nous considérons les caractéristiques suivantes :

- Type de schéma conceptuel peut être en constellation ou en étoile.

- La catégorie du modèle logique NoSQL utilisé. Le modèle peut être orienté colonnes, orienté documents, orienté graphe ou orienté clé/valeur

- Système de stockage ou SGBD (HBase, MongoDB...) pour implanter l'entrepôt de données.

-Type de schéma logique, il s'agit du schéma normalisé ou dénormalisé.

Critères > Approches V	Type de schéma conceptuel	Catégorie de modèle logique	Système de stockage ou SGBD	Type de schéma logique
[Khalil, Belaissaoui, 2020]	Etoile	Clé-valeur	Oracle NoSQL	Dénormalisé
[Dehdouh et al., 2015]	Constellation	Colonnes	HBase	Dénormalisé
[Boussahoua et al., 2017]	Etoile	Colonnes	HBase	-
[Chevalier et al., 2015]	Etoile	Document	MongoDB	Dénormalisé Normalisé
[Sellami, et al., 2020]	Etoile	Graphe	Neo4j	Dénormalisé Normalisé
[Castelltort, Laurent 2014]	-	Graphe	Neo4j	-

Tableau 3.1 Comparaison des travaux connexes.

Ces contributions proposent un processus de passage allant d'un modèle conceptuel multidimensionnel vers un modèle NoSQL par ailleurs l'algèbre OLAP n'est pas pris en charge. C'est là que notre contribution s'intervient, notre approche consiste à définir une algèbre OLAP adapté aux systèmes NoSQL.

Conclusion

Ces dernières années, les systèmes NoSQL ont connu un important développement, Ils ont atteint aujourd'hui un certain niveau de maturité, laissant entrevoir la possibilité d'utiliser ces systèmes pour le développement d'entrepôts de données multidimensionnelles. Plusieurs travaux de recherche ont exploré cette voie en se focalisant sur un seul type de système NoSQL : orienté colonnes, orienté documents et également orienté graphes mais très peu se sont focalisé sur le système NoSQL orienté clé valeur. Contrairement à notre travail qui vise à étudier tous les modèles NoSQL, afin de développer des processus de passage conceptuel-logique.

Concernant les processus de transformation présentés dans notre mémoire, il est indépendant d'une plateforme particulière.

La majorité des travaux existants décrivent les opérations OLAP spécifiques à leurs implémentations au niveau physique. Cependant, très peu de travaux se sont concentrés sur la représentation formelle de l'algèbre OLAP.

Dans ce contexte, nous proposant une algèbre OLAP pour des cubes de données basés sur NoSQL. L'algèbre OLAP proposée est conçue sur la base de la sémantique formelle des opérateurs OLAP et est ensuite mise en œuvre sur des cubes de données NoSQL orienté colonnes.

Chapitre 4 : Conception de l'environnement OLAP adapté aux modèles NoSQL

Introduction

L'émergence des systèmes NoSQL permettent d'envisager de nouvelles approches pour implanter le schéma d'un entrepôt, d'un magasin ou d'un cube de données.

Ces systèmes offrent l'avantage de gérer de grandes masses de données (mégadonnées ou Big Data).

Nous proposons d'adopter une approche de modélisation des entrepôts de données basée sur les trois niveaux d'abstraction conceptuel, logique et physique.

Au niveau conceptuel, la modélisation multidimensionnelle repose sur les concepts de faits, de dimension et de hiérarchie. Cette modélisation consiste à décrire les données analysées au travers d'un schéma en étoile ou en constellation [Kimball, Ross, 2013]. Le schéma conceptuel est ensuite traduit en un modèle logique lui-même traduit en modèle physique. Pour ce faire, des règles de passage sont alors énoncées pour convertir les structures multidimensionnelles (modèles en étoile et en flocons) du niveau conceptuel en un modèle logique NoSQL.

Ce chapitre présente nos propositions pour modéliser un entrepôt de données structurés de manière multidimensionnelle dans les systèmes NoSQL. Notre processus convertit les mécanismes conceptuels dans le modèle logique NoSQL cible. Par la suite nous proposons une algèbre OLAP-NoSQL adaptée au modèle NoSQL orienté colonnes.

4.1 Proposition d'un processus de mappage d'un modèle Conceptuel-multidimensionnel vers un modèle Logique NoSQL

Définition : Un schéma multidimensionnel S est un tuple S = (F, D, R), où F est un ensemble fini de faits F= {F1, … , Fn}, D est un ensemble fini de dimensions D= { D1,…, Dn }, R : F→D c'est l'association entre les faits et ses ensembles de dimensions

Définition : Une dimension D est un tuple D= (Nd, A), où Nd c'est le nom de D et A un ensemble d'attributs A= {A1, An}.

Définition : Un fait F est un tuple F=(Nf, M), où Nf c'est le nom de F, M est un ensemble des mesures M={M1,…, Mn}.

Nous présentons, dans cette section, la transformation dénormalisée assurant le mappage aux quatre modèles NoSQL orientés à savoir modèle clé-valeur, modèle orienté document, modèle orienté colonne et modèle orienté graphe tout en mettant en évidence les concepts du Schéma Multidimensionnel.

La figure 4.1 présente un exemple de table de fait « vente » analysé selon trois dimensions : client (informations de client), Temps (date de vente) et Produit (caractéristiques de produit). «TF_Vente » possède deux mesures : la quantité vendu (Quantité) et le montant (Montant)

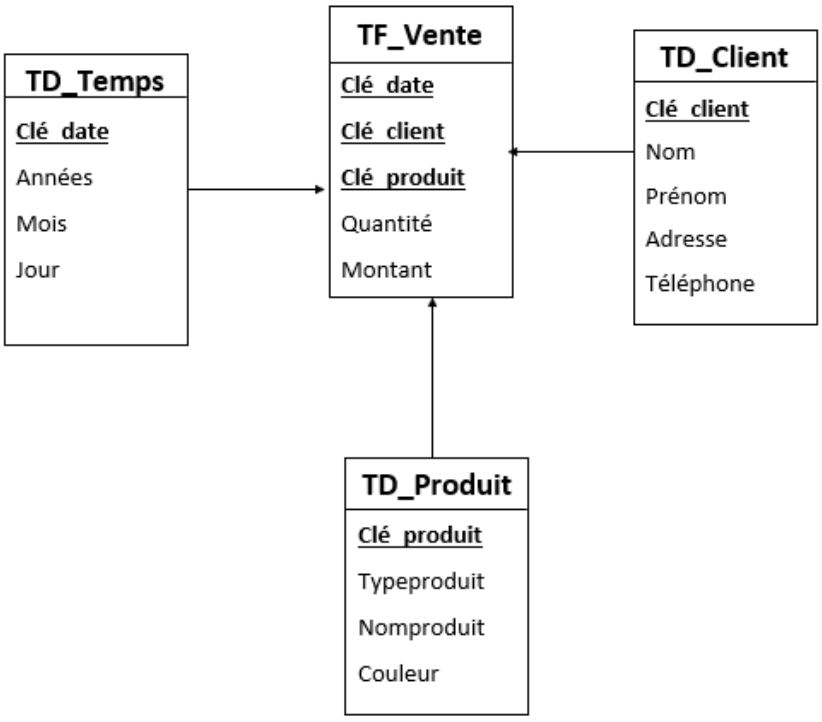

Figure 4.1 : Modèle en étoile du besoin d'analyse vente

4.1.1 Processus de mapping orienté clé valeur

Définition : Un ensemble clé-valeur est un tuple KV = (K, V), où K c'est une clé unique, V est une valeur.

Le passage vers le modèle de donnés orienté clé-valeur est défini comme suit :

– Chaque schéma multidimensionnel (chaque fait Fi de F, et ses dimensions R(Fi)) est transformé en un ensemble clé-valeur KV.

– Le fait Fi est transformé en un ensemble clé valeur Kf Vf ∈ V dans laquelle les mesure sont également un ensemble clé valeur KmVm ∈ Vf.

– Chaque dimension Di ∈ R(Fi) est transformée en un ensemble KdVd ∈ V où Vd est un ensemble clé valeur KaVa.

La Figure 4.2 présente le mapping de l'exemple de schéma en étoile du cube de données (figure 4.1) vers le modèle de donnés orienté clé-valeur.

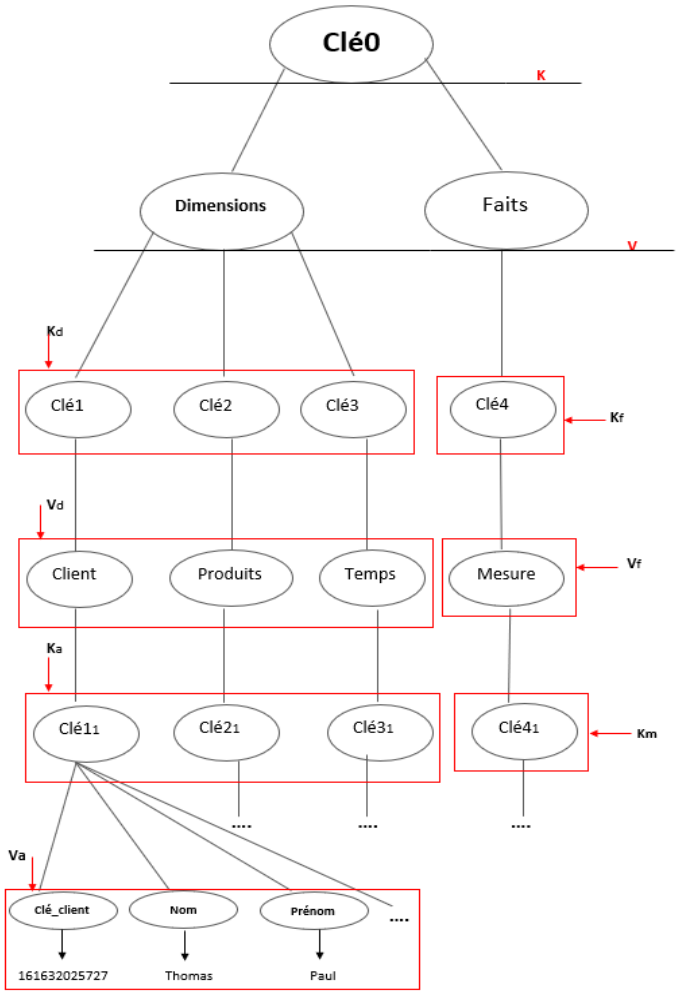

Figure 4.2 : Représentation graphique d'une ligne de la table TF vente dans un modèle clé valeur

ALGORITHME de passage vers un modèle orienté clé-valeur

Algorithme CV(C)

VARIABLES

ENTRÉES

C **:** cube de données ;

SORTIES

KV : ensemble de clé-valeur ;

DÉBUT

Créer (KV) ; // création de l'ensemble clé-valeur

Créer (Kf Vf) ; // création de l'ensemble clé-valeur ∈ V

Kf VfßC.F ; //insertion du fait dans l'ensemble de valeur V

V.inserer (Kf Vf) ; // insertion de l'ensemble Kf Vf dans l'ensemble de valeur V

Pour i de 0 à p faire //parcourir toutes les mesures de la table de fait

Créer (Km Vm) ; // création de l'ensemble clé-valeur KmVm ∈ Vf.

Km VmßC.F.m(i) ; // insertion des mesures dans l'ensemble clé-valeur KmVm

Vf.inserer (Km Vm) ; insertion de l'ensemble Km Vm dans l'ensemble de valeur Vf.

FIN

Pour j de 0 à n faire //parcourir toutes les tables de dimensions

Créer (Kd Vd) ; // création de l'ensemble clé-valeur Kd Vd ∈ V.

Kd VdßC.D(j); // insertion de la dimension Dj dans l'ensemble clé-valeur KdVd

V.inserer (Kd Vd) ; //insertion de l'ensemble (Kd Vd)dans l'ensemble de valeur V.

FIN

FIN

RETOURNER (KV) ;

4.1.2 Processus de mapping orienté Colonne

Définition : Un espace clé K est un tuple K = (Nk, F), où Nk est le nom de K, F est un ensemble de famille de colonne.

Définition : Une famille de colonne f ∈ F est un tuple f = (Nf, Sc), où Nf est le nom de f, Sc est un ensemble d'ensemble de colonne.

Définition : Un ensemble de colonne cs ∈ Sc est un tuple cs = (Ncs, Cols), où Ncs est le nom de cs, Cols est un ensemble de colonne.

Définition : Une colonne c ∈ Cols est un tuple c = (Nc, v), où Nc est le nom de c, v est une valeur atomique.

Le passage vers le modèle de donnés orienté colonnes est défini comme suit :

– Chaque schéma multidimensionnel (chaque fait Fi de F, et ses dimensions R(Fi)) est transformé en une table T.

– Le fait Fi est transformé en une famille de colonnes CFfi de T dans laquelle chaque mesure mi est une colonne Cmi ∈ CFfi.

– Chaque dimension Di ∈ R(Fi) est transformée en une famille de colonnes CFdi où chaque attribut Ai de dimension Di est transformé en une colonne Cdi de la famille de colonnes CFdi (Ci ∈ CFdi).

-L'instance du fait est donc composée de la famille de colonnes CFm (les mesures et leurs valeurs respectives) et des familles de colonnes des dimensions CFd (les attributs de chaque dimension et leur valeurs respectives).

Le résultat de ce passage est représenté dans la figure 4.3

ALGORITHME de passage vers un modèle orienté colonne

ALGORITHME OC ;

VARIABLES

ENTRÉES

C : cube de données ;

SORTIES

T : table de famille de colonnes ;

DÉBUT

Créer (T) ; // création table

Pour i de 0 à n faire //parcourir toutes les tables de dimension

Créer (CFd) ; // création de la famille de colonne dimension

CFd(i) =C.D(i) ; // insertion de la dimension D(i) du cube C dans la famille de colonne CFd(i)

T.inserer(CFd(i)); // insertion de la famille de colonne CFd(i) dans la table T

FIN

Pour i de 0 à m faire //parcourir toutes les mesures de de fait F

Créer (CFm) ; // création de la famille de colonne mesure

CFm =C.F.m(i) ; // insertion de la mesure m(i) du cube C dans la famille de colonne CFm

T.inserer(CFm);

FIN

FIN

RETOURNER (T) ;

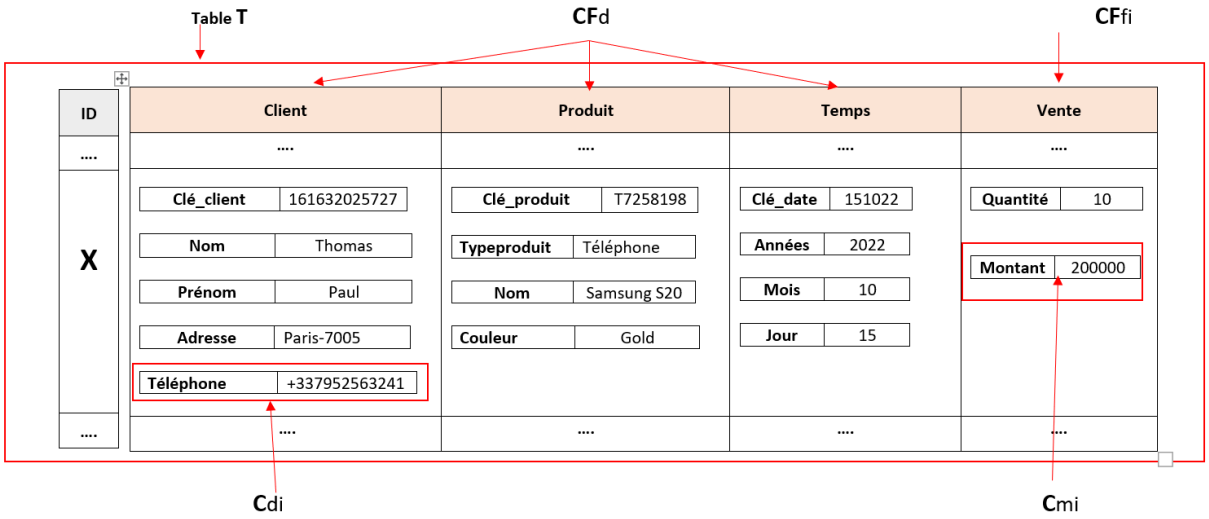

Figure 4.3 : Exemple d'une ligne de la table TF vente dans le modèle orienté colonne

4.1.2 Processus de mapping orienté document

Définition : Une collection de document C c'est un tuple C = (nC, Docs), où nC le nom de C, Docs est un ensemble de document.

Définition : Un document d ∈ Docs est un tuple d = (nd, A), où nd est le nom de d, A est un ensemble des attributs A= {A1,.., An}.

Définition : Un attribut a ∈ A est un tuple a = (na, v), où na est le nom de a, v contient une valeur.

Comme illustré dans la figure 3.4 un cube de donnée NoSQL orienté document après sa transformation à partir d'un modèle multidimensionnel selon les étapes suivantes :

-Chaque schéma multidimensionnel (chaque fait Fi et ses dimensions associées R(Fi)) est traduit en une collection C.

-Une instance de fait Fi est convertie en un document Dfi.

-Chaque dimension Di est convertie également en un document imbriqué Ddi ∈ Dfi contenu dans le même document que l'instance de fait Fi.

-Les valeurs des mesures M sont combinées au sein d'un document imbriqué attM.

- Chaque Mi ∈ M est un attribut simple att m.

-Chaque attribut Ai de la dimension Di est converti en un attribut simple Att d.

ALGORITHME de passage vers un modèle orienté document

ALGORITHME OD ;

VARIABLES
ENTRÉES
C : cube de données ;
SORTIES
CD: collection de document ;
DÉBUT
Créer (CD) ; // création de la collection de document
Pour i de 0 à n faire //parcourir toutes les tables de dimension
Créer (Dd) ; // création du document dimension
Dd(i) =C.D(i) ; // insertion de la dimension D(i) du cube C dans le document Dd
CD.inserer(Dd(i)); // insertion du document Dd(i) dans la collection CD
FIN
Pour j de 0 à m faire //parcourir toutes les mesures de de fait F
Créer (Dm) ; // création du document mesure
Dm =C.F.m(j) ; // insertion de la mesure m(j) du cube C dans le document
CD.inserer(Dm);
FIN
FIN
RETOURNER (CD) ;

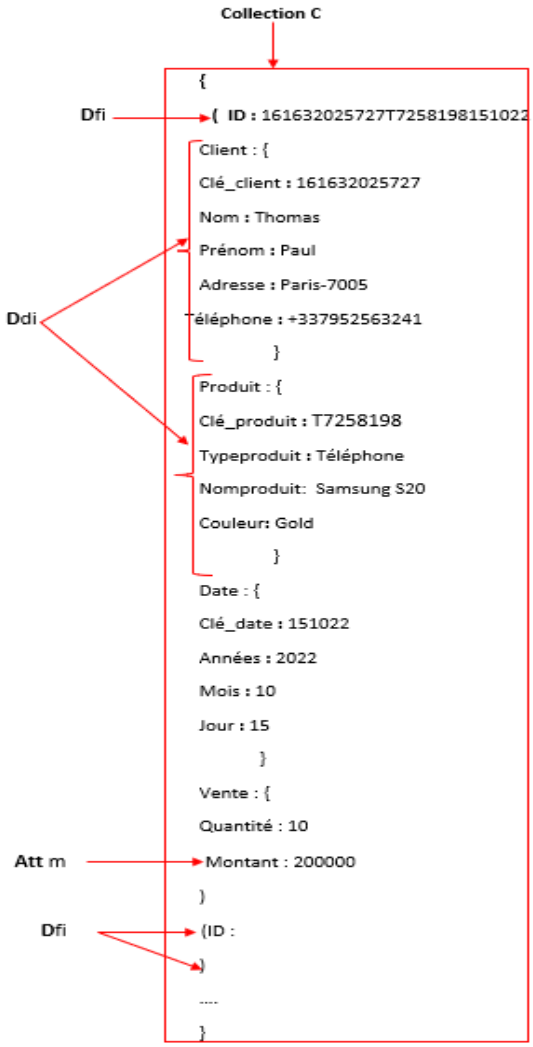

Figure 4.4 : Contenu de la collection C

4.1.3 Processus de mapping orienté graphe

Définition : Un graphe G c'est un tuple G = (nG, N, Ar), où nG c'est le nom de G, N est un ensemble de nœuds, Ar : N → N est une relation entre les nœuds.

Définition : Un nœud N c'est un tuple N = (nN, V), où nN c'est le nom de N, V est un ensemble de propriétés de N.

Dans cette transformation chaque fait et ses dimensions sont transformés en nœuds selon les règles suivantes :

-Chaque schéma multidimensionnel (chaque fait Fi et ses dimensions associées R(Fi)) est traduit en un graphe G

-Chaque fait Fi est transformé en nœud Nfi avec l'étiquette du nœud-fait contenant le nom de fait. Chaque mesure est transformée en propriété de Noeud de Fi.

-Chaque dimension Di est transformée en un nœud-dimension Ndi avec une étiquette contenant le nom de Di.

-Chaque paramètre de Di est transformé en une propriété du nœud-dimension Ndi.

-Chaque lien entre fait et dimension est représenté par un arc d'un nœud fait vers un nœud dimension.

L'application de ces règles sur le cube de données de la figure 1 nous donne le formalisme graphique montré dans la figure 4.5 :

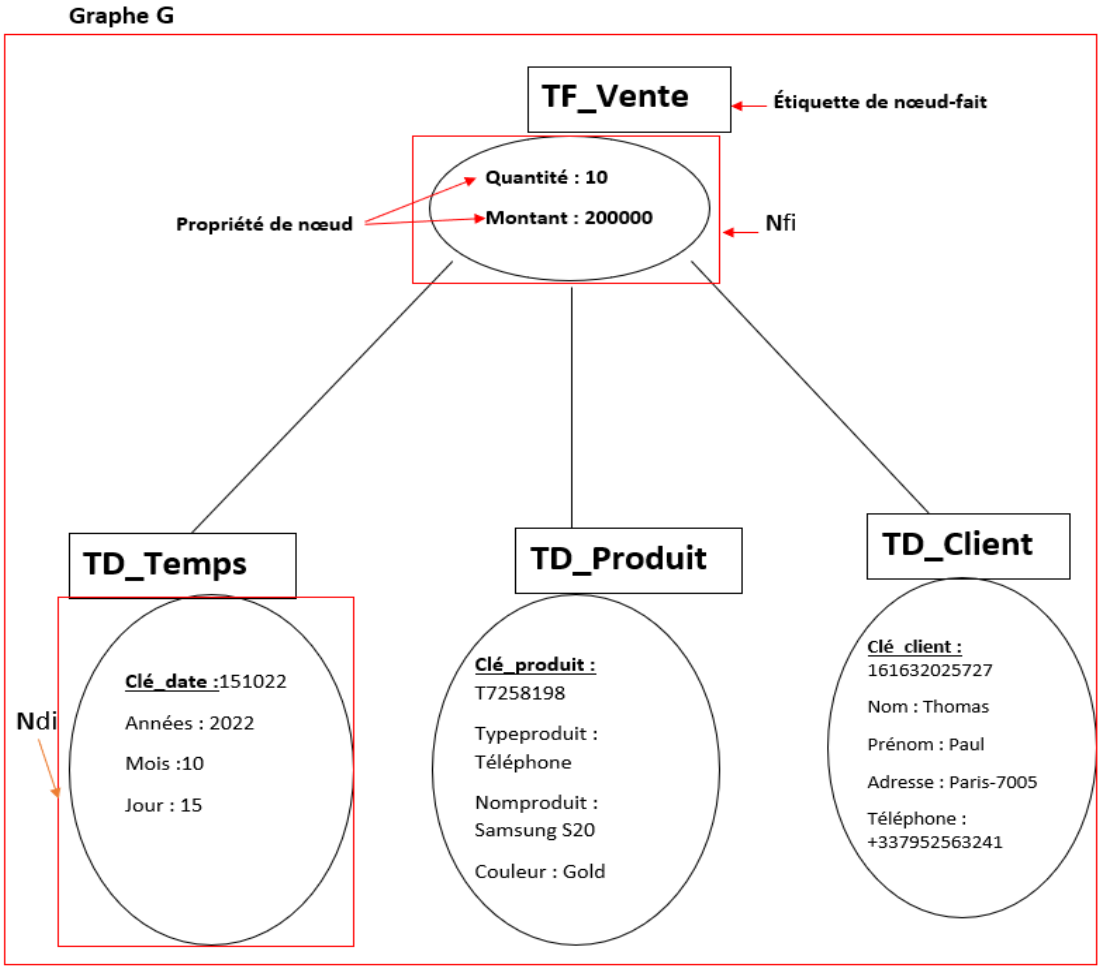

Figure 4.5 : Schéma représentatif de TF vente en orienté graphe

ALGORITHME de passage vers un modèle orienté graphe

Algorithme OG(C)

VARIABLES

ENTRÉES

C **:** cube de données ;

SORTIES

G : ensemble de clé-valeur ;

DÉBUT

Créer (G) ; // création de graphe

Créer (Nf) ; // création de nœud du fait

NfßC.F ; //insertion du fait dans le nœud Nf

G.inserer (Nf) ; // insertion de nœud de fait dans la graphe G

Pour j de 0 à n faire //parcourir toutes les tables de dimensions

Créer (Nd) ; // création de nœud de dimension ∈ G.

NdßC.D(j); // insertion du dimension Dj dans le nœud Nd dans la graphe G

G.inserer (Nd) ; //insertion de nœud de dimension dans la graphe G

FIN

FIN

RETOURNER (G) ;

4.2 Processus de passage d'un modèle Logique orienté colonne vers le modèle Physique HBase

Définition : Table T est un tuple T=(nT,RK,CF),où nT c'est le nom de T, RK est un ensemble de clés, CF est un ensemble de famille de colonne.

Définition : Column Familly CF c'est un tuple CF=(nCF,C), où nCF c'est le nom de CF, C'est un ensemble de colonnes.

Définition : Column C c'est un tuple C=(nC,V), où nC c'est le nom de C, V est une valeur atomic.

Définition : Column Qualifier QC c'est un tuple CQ=(nCF,nC).

Définition : Cell Cl c'est le tuple Cl= (RK,CQ,V).

-Chaque famille de colonne est transformée en une table Table T.

-Chaque famille de dimension CFd et famille de mesure CFm est transformé en famille de colonne Column Familly CF de T.

Chaque colonne Cdi ∈ CFd et Cmi ∈ CFm est transformé en une colonne Column C de CF.

4.3. Proposition d'une algèbre OLAP adaptée au modèle NoSQL orienté colonnes

L'algèbre OLAP relationnel propose des opérations de manipulation des cubes de données tel que les opération SLICE et DICE. Ces opérations sont définies avec les formules suivante :

∏ sélectionne des colonnes de la table T.

∂ extrait les lignes de la tables T, où la condition cond est vérifié.

Pour exprimer par exemple un SLICE nous allons suivre l'exemple suivant :

Soit une table de fait TF contenant la mesure m, avec ses tables de dimension TD1 avec les attributs {att1, att2} et la table de dimension TD2 avec les attributs {att3, att4}

Un SLICE sur 'att1=val' est formulé avec l'expression suivante :

$\prod_{att2,att3,att4,mesure} \partial_{(att1=val)}$ TD1,TD2,TF

L'algèbre OLAP-NoSQL que nous avons proposée est composée des deux opérateurs $\mathfrak{S}$ et $\mathfrak{f}$ définis comme suit :

1- $\mathfrak{f}$ est l'opérateur de restriction sur les lignes selon un prédicat qui peut être atomique noté P ou bien composé noté P1 <op> P2 <op> …<op> Pn, dans le prédicat composé <op> est l'opérateur logique AND ou OR.

$\mathfrak{f}$ Récupère les lignes (rows) qui vérifient le prédicat en utilisant le rowkey.

Exemple :

Soit l'ensemble des lignes de la famille de colonnes Location dans le modèle NoSQL orienté colonnes :

{ 15,Location :City :Paris , Location :Pays:France

16, Location :City :Toulouse , Location :Pays:France

20, Location :City :Milan , Location :Pays:Italy }

Une restriction sur les lignes selon Location :Pays=France est notée comme suit :

σ (Location :Pays=France)

Le résultat est le suivant :

{ 15,Location :City :Paris , Location :Pays:France

16, Location :City :Toulouse , Location :Pays:France }

2- π est l'opérateur d'extraction des colonnes qui prend comme paramètre un ensemble de colonnes de la forme {famille de colonne : colonne} et le résultat de l'opérateur σ.

Exemple :

Une extraction des colonnes Location :City avec une restriction sur Location :Pays=France est notée comme suit : π (Location :City, σ (Location :Pays=France))

Le résultat de cette extraction est le suivant :

{ 15,Location :City :Paris

16, Location :City :Toulouse }

Les opérations OLAP:

4.3.1. Opération NoSQL-SLICE

Sélectionne une seule valeur pour une dimension. La notation algébrique de l'opérateur SLICE est la suivante :

SLICE= Ş (C , F)

C’est la liste des colonnes {famille de colonne : colonne}.

F est la liste des lignes retourné par l’opérateur **ƒ(p)**. avec p un prédicat atomique.

Le résultat d’un slice selon Location : Pays=France est notée comme suit :

SLICE= Ş (Location : City,**ƒ**(Location : Pays=France)**)**

{ 15,Location :City :Paris

16, Location :City :Toulouse }

4.3.2. Opération NoSQL-DICE

Sélectionne une ou plusieurs valeurs pour un minimum de deux dimensions. La notation algébrique de l’opérateur DICE est la suivante :

DICE= Ş (C , F)

C est la liste des colonnes {famille de colonne : colonne}.

F est la liste des lignes retourné par l’opérateur **ƒ(p)**. avec p un prédicat composé.

Le résultat d’un dice selon Location : Pays=France et Location : City=Paris est notée comme suit :

DICE= Ş ({Location : Pays, Location : City} **,ƒ** (Location : Pays=France, Location : City=Paris)**)**

{ 15,Location :City :Paris }

4.3.3. Algorithme opérateur/opération

Nous allons présenter dans cette partie les algorithmes avec lesquels fonctionnent les opérateurs OLAP déjà définis.

ALGORITHME de l’opérateur restriction

ALGORITHME F(C,P)

VARIABLES
ENTRÉES

C :cube de données ;

P : prédicats ; //le prédicat selon lequel on souhaite appliquer l'opérateur

SORTIES

L :liste des rows ; //les lignes de la table sous la forme (rowkey,row) tel que rowkey est un identifiant de la ligne row

i: rowkey;

DÉBUT

Créer(L) ;

Pour i de 0 à n faire //parcourir toutes les lignes du cube de donnée

Si P=vrai faire //le prédicat est vérifié

L.inserer(C(i)) //insertion de la ligne dans la liste L

FIN

FIN

RETOURNER (L) ;

ALGORITHME de l'opérateur de sélection

ALGORITHME S(CF,F)

VARIABLES
ENTRÉES

CF: liste de colonnes ; //le nom des colonnes a sélectionné dans un cube de données

F: liste de lignes; // une ligne est de la forme (rowkey, famille de colonne1, famille de colonne2, … famille de colonne n)

SORTIE

L : liste de lignes ;

DÉBUT

Créer(L) ;

Pour i de 0 à n faire //parcourir la liste F

L.inserer(F.rowkey,F.CF); //insertion des lignes avec les familles de colonnes sélectionnées

FIN

FIN

RETOURNER (L) ;

ALGORITHME de l'opération SLICE

ALGORITHME SLICE(C,CF,P)

VARIABLES

f :liste de lignes ; //les lignes de la table sous la forme (rowkey,row) tel que rowkey est un identifiant de la ligne row

ENTRÉES

C : cube de données ;

CF : liste de colonnes ; // le nom des colonnes a sélectionné dans un cube de données

P : prédicats ; //le prédicat selon lequel on souhaite appliquer l'opérateur

SORTIES

L :liste de lignes ;

DÉBUT

f=F(C,P) ;

L=S(CF,f) ;

FIN

RETOURNER (L) ;

ALGORITHME de l'opération DICE

ALGORITHME DICE(C,CF,P)

VARIABLES

f :liste de lignes ; //les lignes de la table sous la forme (rowkey,row) tel que rowkey est un identifiant de la ligne row

ENTRÉES

C : cube de données ;

CF : liste de colonnes ; // le nom des colonnes a sélectionné dans un cube de données

P : prédicats ; //le prédicat selon lequel on souhaite appliquer l'opérateur

SORTIES

L : liste de lignes ;

DÉBUT

f=F(C,P) ; //appel à la fonction F pour filtrer les lignes qui vérifie les prédicats P

L=S(CF,f) ; //appel à la fonction S pour sélectionner les famille de colonnes a afficher

FIN

RETOURNER (L) ;

CONCLUSION

Ce chapitre présente dans la première partie les processus de conversion au niveau logique. Nous avons présenté quatre processus de conversions vers un modèle NoSQL à savoir orienter clé valeur, orienté colonne, orienté document, orienté graphe. Ensuite nous avons défini le processus de passage logique orienté colonne vers le SGBD HBase.

La troisième partie été consacré à la définition d'une algèbre OLAP-NoSQL plus particulièrement les opérations SLICE et DICE.

Chapitre 5 Implémentation de l’environnement OLAP adaptée au modèle NOSQL

Introduction

Dans ce chapitre nous allons commencer tout d'abord par la description de notre environnement de travail, le langage ainsi que les outils utilisés pour l'implémentation et le jeu de données. Par la suite nous allons présenter le fonctionnement général de notre application ainsi que ses différentes interfaces.

6.1 Environnement de développement

Nous présentons dans cette section l'environnement que nous avons utilisé pour implémenter notre système OLAP-NoSQL.

6.1.1. Matériel

Nous avons mené ce travail sur une machine virtuelle avec un processeur Intel i3-3110M CPU @ 2.40GHz, une RAM de 5 Go et un espace de stockage de 160 Go sur le système d'exploitation Linux x64-bit.

6.1.2. Plateforme

Nous avons mis en place un environnement distribué, basé sur la plateforme Hadoop 3.2.3 et le système de gestion de données NoSQL orienté colonnes HBase 2.4.12.

- **Hadoop**

Hadoop est configuré en mode pseudo-distribué, dans ce mode le Namenode, Datanode, Secondary Name node, Resource Manager, Node Manager sera exécuté séparément sur une JVM (Java Virtual Machine) distincte c'est à dire sur différents processus Java.

- **HBase**

HBase est aussi installé en mode pseudo-distribué ce qui signifie un cluster à nœud unique (un seul hôte), mais les daemons HMaster, HRegionServer et Zookeeper (QuorumPeer) s'exécute comme un processus séparé.

6.1.3. Langage de programmation

- **Python 3.10.4 :**

Python est le langage de programmation open source multi-plateformes et orienté objet. Grâce à des bibliothèques spécialisées. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels.

6.1.4. Outils et librairies

- **Visual studio code :**

VS code est un éditeur de code source créé par Microsoft pour Windows, Linux et MacOs. Les fonctionnalités incluent la prise en charge du débogage, la coloration syntaxique, la complétion de code intelligente, les extraits de code, la refactorisation de code et Git intégrer. Les utilisateurs peuvent installer des extensions qui ajoutent des fonctionnalités supplémentaires.

- **HappyBase :**

Happybase est une libraires développeur python conviviale pour interagir avec Apache HBase. HappyBase est conçu pour être utilisé dans les configurations HBase standard et offre aux développeurs d'applications une API pythonique pour interagir avec HBase.

- **Tkinter :**

C'est un module intégré à la bibliothèque standard de Python, permettant de créer des interfaces graphiques : des fenêtres, des widgets (boutons, zones de texte, cases à cocher, …), et des évènements (clavier, souris, …).
Tkinter est disponible sur Windows et la plupart des systèmes Unix : les interfaces crées avec Tkinter sont donc portables.

6.2. Présentation de jeu de données

Dans cette partie nous décrivons le jeu de données synthétiques personnalisés que nous avons générés pour mener nos expérimentations.

Nous avons pris l'entrepôt de données Northwind et nous avons extrait le chiffre d'affaires et le nombre de commandes réalisées par client *(id, nom, adresse, ville, région,*

pays, numéro de téléphone), par produit (*id, nom produit, prix unitaire*) et par mois (*mois, année*) comme le montre la figure 5.1

Figure 5.1 Illustration du fichier csv utilisé comme dataset

Pour le chargement de données dans HBase tout d'abord le fichier csv est importé dans le HDFS de Hadoop en utilisant la commande *put* comme le montre la figure 5.2.

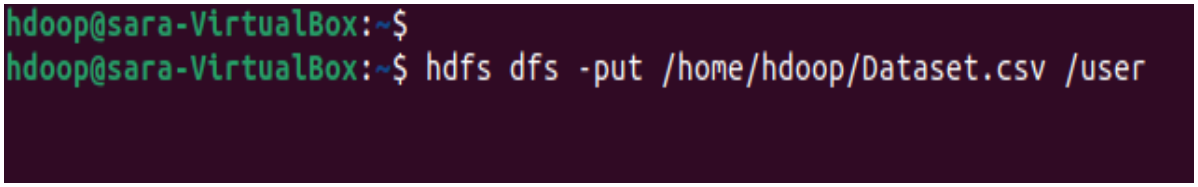

Figure 5.2 Importations des données vers le HDFS de Hadoop

Ensuite dans la table HBase nommée avec la commande ImportTsv (voir figure 5.3).

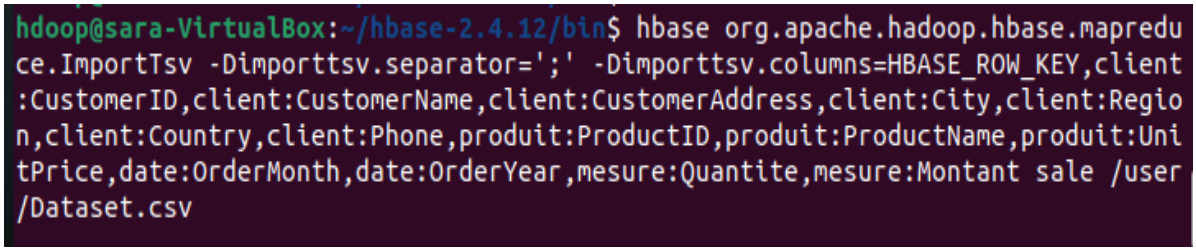

Figure 5.3 Importations des données vers HBase

6.3. Présentation de l'application OLAP-HBase

Nous présentons dans cette section notre application d'aide à la décision qui permet aux utilisateurs d'analyser des données stockées dans le SGBD orienté colonne HBase.

Comme illustré dans figure 5.4, OLAP-HBase permet aux utilisateurs de choisir le sujet d'analyse souhaité c-à-d le cube de données dans l'interface "sujet d'analyse".

Cette interface est divisée en trois compartiments, (a) cube de données, (b) vue OLAP et (c) filtre.

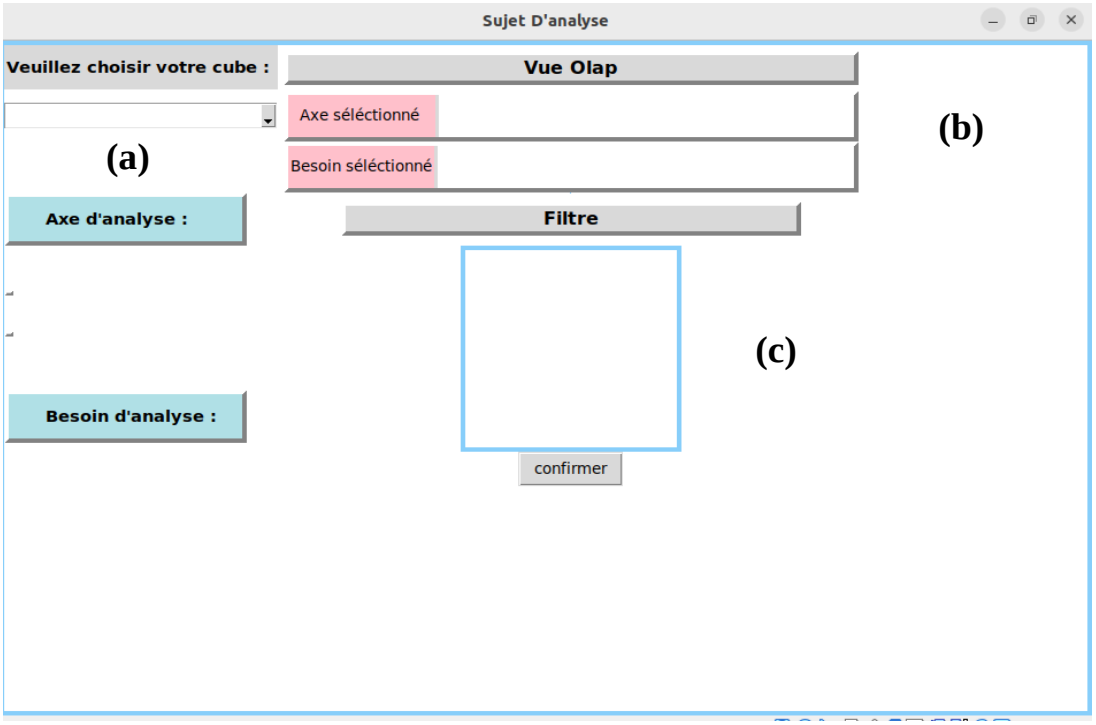

Figure 5.4 : Capture de l'interface Sujet D'analyse.

Le compartiment "axe d'analyse" permet à l'utilisateur de construire sa requête en sélectionnant les familles de colonne (*client*, *produit*, *date*) puis en cliquant à droite sur les colonnes qu'il souhaite visualiser ensuite sélectionner « *ajouter axe* ».

Comme le montre la figure 5.5, les colonnes sélectionnées sont affichées dans le compartiment vue OLAP.

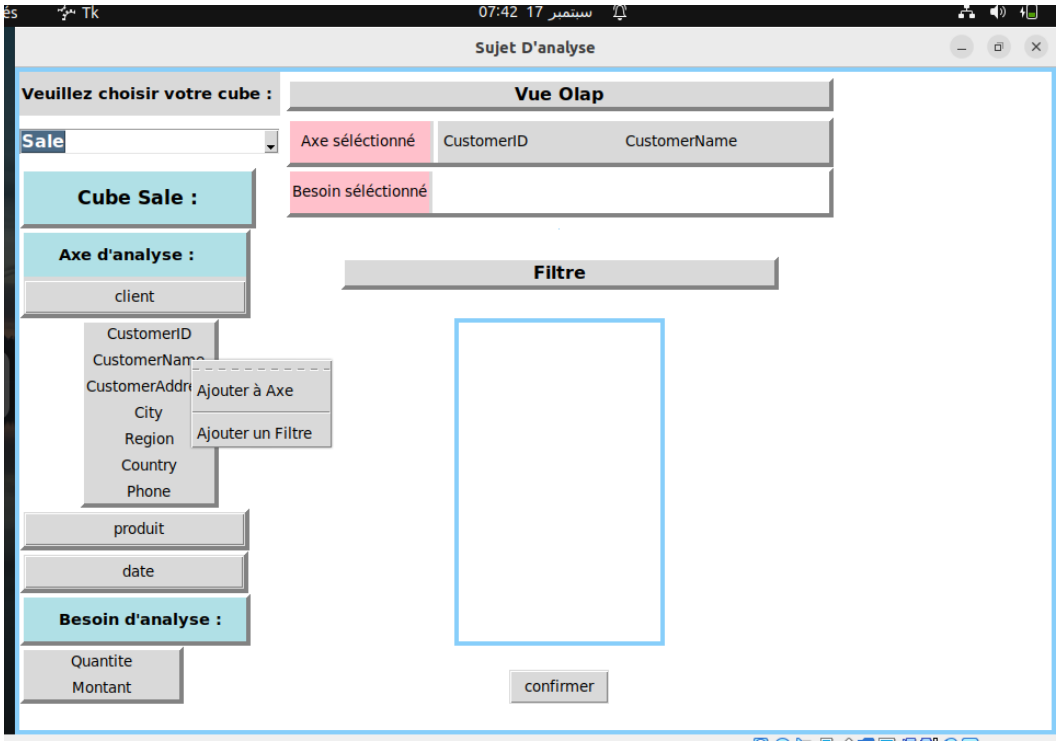

Figure 5.5 : Exemple de construction d'une requête OLAP.

Comme illustré dans la figure 5.6 l'utilisateur peut appliquer des filtres sur des dimensions (axes d'analyses) en choisissant « *ajouter filtre* » et en introduisant une valeur pour cette dimension.

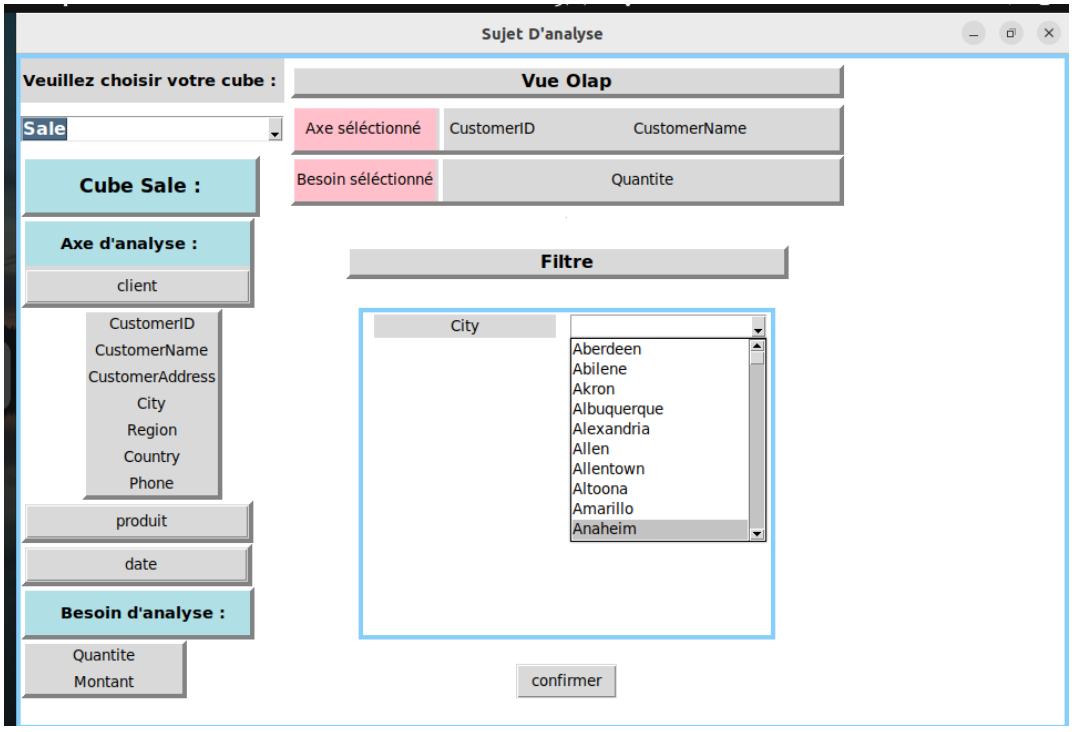

Figure 5.6 : Représentation du compartiment filtre.

Si l'utilisateur choisit une seule famille de colonnes avec une seule valeur, une requête SLICE est lancée.

La figure 5.7 montre un exemple d'une requête SLICE sur *City='Akron'*

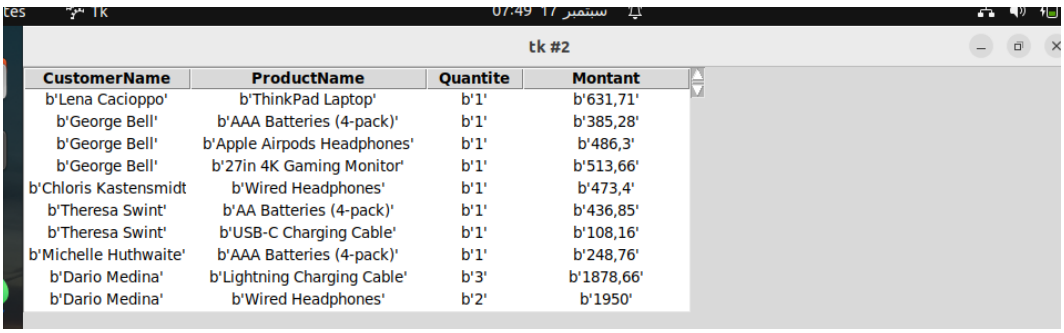

Figure 5.7 : Résultat de l'exécution de la requête SLICE

Il est possible de choisir plusieurs valeurs ou bien de sélectionner plusieurs dimensions comme représenté dans la figure 5.8 Dans ces cas, l'utilisateur lance une requête DICE, la figure montre un exemple d'une requête DICE sur *City='Akron' et OrderYear=2016.*

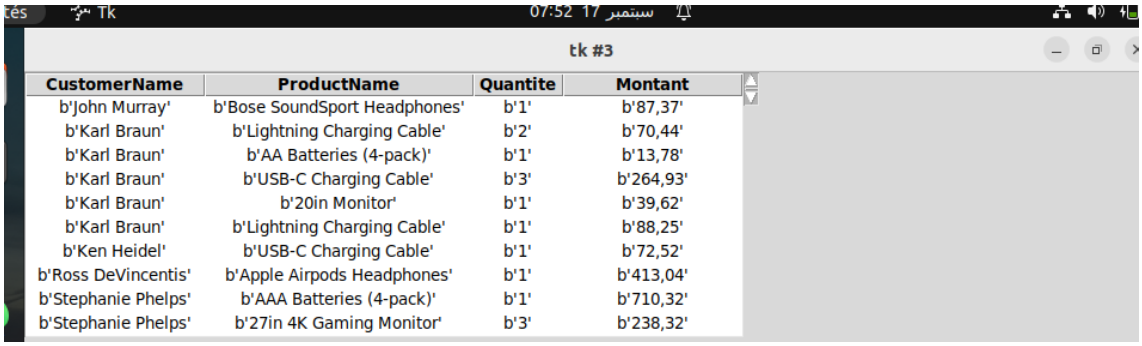

Figure 5.8 : Résultat de l'exécution de la requête DICE

6.4 Validation de l'application OLAP-HBase :

La démarche de validation de notre système consiste à comparer le nombre de lignes retourné par la même requête exécuté dans un SGBDR (MSSQL) et notre système OLAP-Hbase. Notre système est alors jugé valide car la requête retourne le même résultat, comme le montre la figure 5.9.

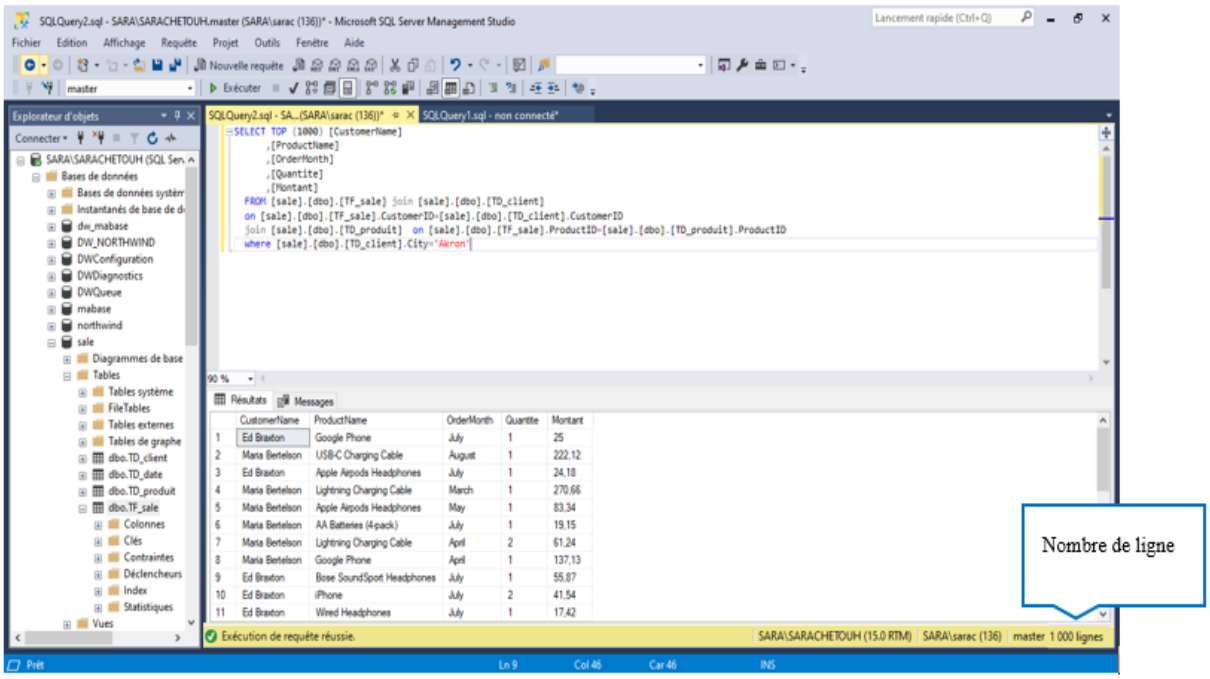

Figure 5.9 : Résultat de la requête SLICE dans MSSQL

Conclusion générale et perspective :

L'objectif principal de ce travail est la proposition d'une nouvelle approche d'analyse des données en ligne (OLAP) qui s'adapte bien aux nouveaux environnements basés sur les modèles NoSQL.

Pour attendre cet object, nous avons arrêté la démarche suivante :

1. Etude bibliographique sur les domaines traités par ce projet : consiste en une revue de la littérature sur l'informatique décisionnelle, particulièrement les environnement OLAP, les mégadonnées avec les paradigmes et les environnements qui lui sont dédiés et enfin les modèles de données NoSQL
2. Etat de l'art : consiste en une revue des travaux connexes dans le domaine de l'OLAP basé sur des modèles NoSQL
3. Proposition d'une nouvelle approche d'analyse OLAP adaptée aux modèles NoSQL, tout particulièrement au modèle orienté colonnes
4. Mise en œuvre de la plateforme NoSQL – OLAP basée sur une interface graphique d'expression des besoins d'analyse

Concernant notre contribution, nous nous sommes penchés, dans un premier temps, sur la définition des règles de passage conceptuel-logique des quatre modèles NoSQL pour mieux cerner les problèmes rencontrés lors de ces processus de mappage.

Dans un second temps, nous avons réalisé le processus du mappage du modèle logique NoSQL orienté colonne vers le modèle physique en choisissant le système de gestion de base de données HBase.

Par la suite, nous avons modélisé et implémenté avec succès des requêtes OLAP sur des données stockées dans un SGBD NoSQL orienté colonne HBase.

L'ultime étape nous a conduit à la concrétisation des requêtes que nous avons pu réaliser à travers un jeu de données.

En dehors du périmètre initial de notre projet, nous avons pu réaliser une interface graphique conviviale dédiée aux requêtes OLAP sur des cubes de données stockés dans un SGBD NoSQL orienté colonnes HBase pour permettre aux utilisateurs de visualiser les résultats de leurs requêtes.

Les problèmes que nous avons rencontré lors de la réalisation de notre travail sont liés au fonctionnement du SGBD HBase qui ne génère pas automatiquement le ID du tuple (RowKey), ce qui nous a emmené à définir nous-même une colonne dans le fichier source jouant le rôle de RowKey.

Perspectives :

Comme tout projet, celui-ci mérite des améliorations et une continuité que nous pouvons résumer de la manière suivante :

-Augmenter la taille des fichiers pour connaître les performances et la rapidité des requêtes :

1. Réaliser le passage en autre modèles de donnée NoSQL : tel que l'orienté graphe, l'orienté documents et l'orienté clé-valeur.
2. Réaliser le passage en utilisant d'autres formats de sources : base de données relationnelle, fichier Excel.
3. Essayer les requêtes dans d'autres SGBD pour les comparer :tel que Cassandra, Big Table
4. Étendre notre projet par l'ajout d'autres requêtes OLAP : ROLL-UP, DRILL-DOWN, PIVOT.

Listes d’abréviation

BD : Base de données

SGBD : Système de Gestion des Bases de Données

OLAP: Online Analytical Processing

R-OLAP: Relational OLAP

SQL: Structured Query Language.

DW: Data Warehouse

ETL : Extract Transform Load

TD : Table de Dimension

TF : Table de Fait

NoSQL: Not Only SQL

CSV: Comma-Separated Values

HDFS: Hadoop Distributed File System

CF: Column Family

Bibliographie

[Aniceto, et al., 2015] Rodrigo Aniceto et al. 2015. Evaluating the Cassandra NoSQL Database Approach for Genomic Data Persistency, Evaluating the Cassandra NoSQL Database Approach for Genomic Data Persistency. Int. J. Genomics Int. J. Genomics.

[Bala,2017] Bala Mahfoud. (2017). Parallélisation et distribution du processus d'intégration ETL pour le traitement de données massives.

[Banker 2011] Kyle Banker, MongoDB in Action (Dec. 2011), Ed.Hanning

[Bastien, 2017] Bastien L, (2017) data analytics

[Bimonte,Pinet, 2012] Bimonte Pinet, François. (2012). Conception des entrepôts de données : de l'implémentation à la restitution.

[Borthakur, 2008] Dhruba Borthakur(2008), HDFS architecture guide, *Hadoop apache project,vol 53*

[Boussahoua et al., 2017] Boussahoua, Mohamed & Boussaid, Omar & Bentayeb, Fadila. (2017). Logical Schema for Data Warehouse on Column-Oriented NoSQL Databases. 247-256. 10.1007/978-3-319-64471-4_20.

[Bradji, 2012] Louardi BRADJI .(2012). Adaptation des techniques de l'Extraction des données (Thèse de Doctorat).

[Brahim, 2018] BRAHIM, Amal. (2018). Approche dirigée par les modèles pour l'implantation de bases de données massives sur des SGBD NoSQL.

[Castelltort, Laurent 2014] Arnaud Castelltort, Anne Laurent. Exploiting NoSQL Graph Databases and In Memory Architectures for Extracting Graph Structural Data Summaries. International Journal of Uncertainty, Fuzziness and Knowledge-

Based Systems, World Scientific Publishing, 2017, 25 (1), pp.81-109. ⟨10.1142/S0218488517500040⟩. ⟨lirmm-01381083⟩

[Cattell, 2011] Cattell, R. (2011) Scalable SQL and NoSQL Data Stores. ACM SIGMOD Record, 39, 12-27. http://dx.doi.org/10.1145/1978915.1978919

[Chevalier et al., 2015] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015b. Entrepôts de données multidimensionnelles NoSQL. In Esteban Zimányi, Stijn Vansummeren, & Toon Calders, eds. Actes des 11es journées francophones sur les Entrepôts de Données et l'Analyse en Ligne, EDA 2015, RNTI. Hermann-Éditions, 161–176.

[Codd, 1993] Edgar F Codd, Sharon B Codd et Clynch T Salley. Providing OLAP (on-line analytical processing) to user-analysts : An IT mandate. Codd and Date, vol. 32, 1993.

[Dean, Ghemawat, 2010] Jeffrey Dean et Sanjay Ghemawat. MapReduce : simplified data processing on large clusters. Communications of the ACM, vol. 51, no. 1, pages 107–113, 2008.

[Dehdouh et al., 2015] K. Dehdouh, F. Bentayeb, O. Boussaid, and N. Kabachi,(2015). Using the column-oriented NoSQL model for implementing big data warehouses, Int. Conf. Parallel Distrib. Process. Tech. Appl., pp. 469–475, 2015.

[Demchenko et al., 2013] Demchenko, Yuri & Membrey, Peter & Grosso, Paola & Laat, Cees. (2013). Addressing Big Data Issues in Scientific Data Infrastructure. 10.1109/CTS.2013.6567203.

[Dharmendra ,2021] Dharmendra Kumar, Data Driven Strategies, OLTP (June 14th, 2021). OLTP vs OLAP: 9 Critical Differences - Learn | Hevo (hevodata.com)

[Douglas, 2001] Laney, D. (2001) 3D Data Management: Controlling Data Volume, Velocity and Variety. META Group Research Note, 6.

[Duda,2012] Duda, J. (2012). Business intelligence and NoSQL databases. Information Systems Management, 1, 25-37.

[Eifrem, 2009] Eifrem, E. (2009). Neo4j-The Benefits of Graph Databases. O'reilly' OSCON-Open-Source Convention. [Online] http://www.oscon.com/oscon2009/public/schedule/detail/8364 (Last accessed on Nov 17, 2013)

[El Malki, 2016] Mohammed El Malki. (2016) Modélisation NoSQL des entrepôts de données multidimensionnelles massives. Modélisation et simulation. Université Toulouse le Mirail - Toulouse II. Français. ffNNT : 2016TOU20139ff. fftel-02057102f

[Han et al. 2011] Jing Han, E Haihong, Guan Le et Jian Du. Survey on NoSQL database. In 6th international conference on pervasive computing and applications (ICPCA), 2011, pages 363–366. IEEE, 2011 [Inmon, 1996] Bill Immon, (1996) .*Building the Data Warehouse.* Wiley

[Hoff, 2009] PETER D. HOFF (2009), *A First Course in Bayesian Statistical Methods - Springer Texts in Statistics,* Springer-Verlag New York Inc

[Gantz, Reinsel, 2011] Gantz, J. and E. Reinsel. 2011. "Extracting Value from Chaos", IDC's Digital Universe Study, sponsored by EMC.

[George,2011] George Lars (2011), *HBase: The Definitive Guide*, O'Reilly

[Gray, et al., 1996] Gray, R., Owen, D. and Adams, C., 1996. Accounting and Accountability: Changes and Challenges in Corporate Social and Environmental Reporting. London: Prentice-Hall.

[Khalil, Belaissaoui, 2020] A. KHALIL and M. BELAISSAOUI, "New approach for implementing big DataMart using NoSQL key-value stores," 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), 2020, Doi: 10.1109/CloudTech49835.2020.9365897.

[Kimball,1996] Kimball, R. (1996) The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, New York.

[Kimball, Ross, 2011] Ralph Kimball and Margy Ross. (2011). The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence, John Wiley & Sons

[Kimball, Ross, 2013] Ralph Kimball and Margy Ross. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, John Wiley & Sons

[Chodorow 2013] Chodorow, K. (2013) MongoDB: The Definitive Guide. O'Reilly Media, Inc.

[Lakshman, Malik, 2010] Lakshman, Avinash & Malik, Prashant. (2010). Cassandra — A Decentralized Structured Storage System. Operating Systems Review. 44. 35-40. 10.1145/1773912.1773922.

[Leavitt, 2010] Leavitt, Neal. (2010). Will NoSQL databases live up to their promise?. Computer. 43. 12- 14. 10.1109/MC.2010.58

[Ludovic , Laure,2017], Ludovic Denoyer et Laure Soulier. (2017) ,Business Intelligence - M1DAC OLAP, C4-BI-OLAP.pdf (lip6.fr)

[Morfonios, et al., 2007] Morfonios, Konstantinos & Konakas, Stratis & Ioannidis, Yannis & Kotsis, Nikolaos. (2007). ROLAP implementations of the data cube. ACM Comput. Surv.. 39. 10.1145/1287620.1287623.

[O'Malley, 2008] Owen O'Malley. 2008. Terabyte sort on apache hadoop. Yahoo Available Online Httpsortbenchmark OrgYahoo-Hadoop PdfMay (2008), 1–3.

[Rafanelli, 2003] Maurizio Rafanelli, (2003). Multidimensional Databases: Problems and Solutions. Igi Publishing

[Ramdani K, Ramdani R, 2012] RAMDANI Karima-RAMDANI Randa. (2012). Création d'un portail décisionnel appliqué au Business Intelligence (mémoire de Master).

[Sadalage, Fowler, 2012] Sadalage P.J., Fowler M. (2012). NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Addison Wesley.

[Sellami, et al., 2020] Amal Sellami, Ahlem Nabli, and Faiez Gargouri. (2020). Graph NoSQL Data Warehouse Creation. In Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20). Association for Computing Machinery, New York, NY, USA, https://doi.org/10.1145/3428757.3429141

[Sharma, Sugam, 2015] Sharma, Sugam. (2015). An Extended Classification and Comparison of NoSQL Big Data Models.

[Teste 2009] Olivier Teste (2009) Modélisation et manipulation des systèmes OLAP : de l'intégration des documents à l'usager. Habilitation à Diriger des recherches de l'Université Toulouse 3.

[Teste, 2000] Olivier Teste. (2000). Modélisation et manipulation d'entrepôts de données complexes et historisées. Université Paul Sabatier-Toulouse III.

[Tournier, Ronan, 2007] Tournier Ronan. (2007). Analyse en ligne (OLAP) de documents.

[Vassiliadis, Sellis, 1999] VASSILIADIS, SELLIS, (1999).A survey of logical models for OLAP databases".

[Vora,2011] Vora, M.N. (2011) Hadoop-HBase for Large-Scale Data. 2011 International Conference on Computer Science and Network Technology (ICCSNT), Harbin, 24 December 2011, 601-605. https://doi.org/10.1109/ICCSNT.2011.6182030