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

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

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

Présenté par :
Belghali Abdallah
Bendani Mustapha

THÈME :

An Approach for moving from a relational model to a column-oriented NoSQL model in a decision-making information system

Soutenu le : 25/06/2024

Devant le jury composé de :

Betouati Fatiha	MCB	Université de Mostaganem	Président
Filali Fatima Zohra	MCB	Université de Mostaganem	Examinateur
Abdallah Bensalloua Charef	MCA	Université de Mostaganem	Encadreur

Année Universitaire 2023-2024

Résumé

Le bon fonctionnement des entreprises et des organisations repose largement sur la gestion efficace des vastes quantités de données non structurées telles que les médias sociaux, le contenu Web et les documents. Pour ce faire, la technologie de Data Warehouse (DW) est essentielle, permettant le stockage, la visualisation, l'analyse et la comparaison de ces données, facilitant la génération de rapports et de graphiques.

Dans ce projet, nous nous concentrons sur l'application de solutions de pointe dans le domaine des différentes techniques de modélisation NoSQL pour les entrepôts de données. Nous détaillerons ensuite le processus d'implémentation d'un entrepôt de données multidimensionnel avec un modèle NoSQL basé sur les colonnes. Des règles de mappage seront définies pour transformer le modèle conceptuel multidimensionnel en modèles logiques orientés colonnes, créant ainsi une instance de l'entrepôt de données qui alimentera une application d'aide à la décision développée à cet effet.

Mots-clés: Modèle NoSQL de colonne, Aide à la décision, entrepôt de donnée, NoSQL data base

Abstract

The smooth operation of businesses and organizations heavily relies on the efficient management of vast amounts of unstructured data such as social media, web content, and documents. To achieve this, Data Warehouse (DW) technology is crucial, enabling the storage, visualization, analysis, and comparison of such data, thereby facilitating the generation of reports and graphics.

In this project, our focus lies on the implementation of cutting-edge solutions in the realm of various NoSQL modeling techniques for data warehouses. Subsequently, we elaborate on the implementation process of a multidimensional data warehouse with a column-based NoSQL model. Mapping rules will be defined to transform the conceptual multidimensional model into column-oriented logical models, thereby creating an instance of the data warehouse that will fuel a decision support application developed for this purpose.

Keywords: Column NoSQL model, the decision support, DW Data warehouse, NoSQL data base

ملخص

تعتمد سلامة عمل الشركات والمؤسسات إلى حد كبير على إدارة الكميات الهائلة من البيانات غير المهيكلة (DW) مثل وسائل التواصل الاجتماعي ومحتوى الويب والمستندات. ولتحقيق ذلك، فإن تكنولوجيا مستودع البيانات ضرورية، حيث تمكّن من تخزين وتصور وتحليل ومقارنة هذه البيانات، مما يُيسّر إنشاء التقارير والرسوم البيانية.

لمستودعات NoSQL في هذا المشروع، نركز على تطبيق حلول متطورة في مجال مختلف تقنيات النمذجة مبني على الأعمدة. سيتم تحديد NoSQL البيانات. بعد ذلك، نوضح عملية تنفيذ مستودع بيانات متعدد الأبعاد بنموذج قواعد التعيين لتحويل النموذج المفهومي متعدد الأبعاد إلى نماذج منطقية موجهة نحو الأعمدة، مما يخلق نسخة من مستودع البيانات التي ستُغذي تطبيق دعم القرار الذي تم تطويره لهذا الغرض.

كلمات مفتاحية : نموذج العمود ,NoSQL دعم القرار، مستودع بيانات,DW قاعدة بياناتNoSQL

Dédicaces

Tout d'abord, je tiens à remercier DIEU

De m'avoir donné la force et le courage de mener

Je dédie cette œuvre à mes parents qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de cette œuvre.

Mon père pour ses encouragements et ma mère qu'elle a été le fondement de mon existence et mon bonheur illimité pour l'effort qu'elle a suscité en moi.

À mes frères hamza et ahmed el amine pour m'avoir encouragé tout au long de mon voyage.

Et mon professeur, abdelah Ben salloua charef et Ben Amer Abdelkader, qui m'a beaucoup soutenu et encouragé tout au long de mon parcours.

A mes proches, notamment le Sahraoui Afif, Sebihi Yassin les remercie pour leur tendresse, leur complicité et leur présence.

A tous mes amis qui m'encouragent toujours, et je leur souhaite encore plus de réussite. Merci!

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
ACID	Atomicité, cohérence, isolation et durabilité	15
SQL	Structured Query Langage	19
DW	Data Warehouse	14
EDD OU ED	Entrepôt de données	14
BDR	Base de donnée relationnel	15
OLAP	Online Analytical Processing	22
HOLAP	Online Analytical Processing Hybride	22
MOLAP	Online Analytical Processing Multidimensionnel	22
OLTP	Online Transaction Processing	22
NOSQL	Not Only SQL	36
SGBD	System Gestions Base de Données	18
BDD	Base de données	68
CAP	Consistency, Availability, Partition Tolérance	39
Cqlsh	Cassandra Query Language Shell	69

Table des matières

Introduction Générale

Avec l’augmentation de la bande passante sur Internet et la croissance continue des données provenant des réseaux sociaux ainsi que la diminution des coûts des matériels informatiques, de nouvelles possibilités ont vu le jour dans le domaine de l’informatique. Le volume de données de certaines entreprises a considérablement augmenté. L’informatisation croissante des traitements de tout genre a entraîné une augmentation exponentielle de ce volume de données, souvent appelées big data ou données massives, où de grandes quantités de données sont collectées et stockées chaque jour. Plus il y a de données générées, plus il devient important d'avoir la capacité d'accéder et d'analyser ces données pour les utiliser efficacement. De manière générale, un entrepôt de données est une base de données qui stocke des données actuelles et historiques, de sorte qu'elles peuvent être analysées pour des études de marché, des rapports d'analyse et une prise de décision.

La propagation d'un volume important de données et la diversité des schémas de données ont abouti à la difficulté de stocker, modifier, et gérer ces données par un SGBD (système de gestion de base de données) relationnel basé sur le langage SQL (Structured Query Language). Ces dernières années, les géants du Web comme Google et Amazon ont vu leurs besoins en termes de charge et de volumétrie de données croître de façon exponentielle. Et c’est pour répondre à ces besoins que des solutions alternatives ont vu le jour. Les architectes de ces organisations ont procédé à des compromis sur le caractère ACID des SGBDR. Ces intelligents compromis sur la notion de relationnel ont permis de dégager les SGBDR de leurs freins à la scalabilité et à l’évolutivité. Puis des entreprises comme Facebook, Twitter ou encore LinkedIn ont migré une partie de leurs données sur des bases NoSQL.

Le NoSQL (Not Only SQL ou pas seulement SQL) représente une approche alternative à SQL permettant le stockage et l'analyse de données structurées et non structurées du Big Data. Cette adoption croissante des bases NoSQL conduit à une multiplication et à une amélioration des offres Open Source des moteurs. Des chercheurs ont proposé des approches pour la migration (passage) des bases de données relationnelles vers des bases de données NoSQL (approches indirectes). Cependant, peu de travaux se sont concentrés sur la transformation du modèle conceptuel multidimensionnel vers un modèle logique NoSQL (approches directes).

Notre sujet inclut cette solution de passage du relationnel au NoSQL, et nous l'étudions et en discutons dans 4 chapitres :

Le premier chapitre, nous en apprenons sur les données relationnelles et l'architecture des systèmes d'information décisionnels, puis nous étudions les concepts de base d'un entrepôt de données et de modèles conceptuels et logiques.

Dans le deuxième chapitre, la valeur de NoSQL dans le monde des données et ses avantages sont démontrés. Nous arrivons ensuite au cœur de notre sujet., discutant des dernières avancées technologiques qui étudient les techniques permettant de passer d'un modèle conceptuel relationnel à un modèle logique, NoSQL, et nous mentionnons un exemple des modèles NoSQL qui fonctionne sur l'entrepôt de données.

Nous passons au troisième chapitre à la modélisation. e sont les études et le passage d'un modèle relationnel à un modèle logique (NoSQL), en prenant un exemple d'un ensemble de données Excel à **Sales.csv** Du point de vue de la mise en œuvre, nous passons à un dernier chapitre où nous implémentons notre modèle sur la base de données Cassandra en utilisant le gestionnaire pour Cassandra et tableau et en utilisant le langage Python pour traiter ces données dans Cassandra et l'analyse.

Chapitre 1

ENTREPOT DE DONNES

1.1. Introduction

Le modèle relationnel a trouvé ses limites dans la gestion du big web data, qui a connu une révolution avec l'émergence de sites à fort trafic comme Facebook, Amazon ou LinkedIn. Les grands acteurs du Web ont été rapidement bridés par les réglementations précitées pour deux raisons principales [12] :

– Les gros volumes de données.

– Les montées en charge.

Aucune solution n'ayant été trouvée sur le marché pour répondre à ces problématiques, chacun d'entre eux a décidé de développer son propre SGBD en interne.

Les entrepôts de données ED (Data Warehouse DW) sont apparus vers les années 1990 en réponse au besoin de collecter toutes les informations de l'entreprise dans une base de données unique pour les analystes et les managers. [1] [2]

Toutes les données, y compris leur historique, sont utilisées dans de nombreux domaines, tels que : l'analyse des données et l'aide à la décision (gestion et analyse des marchés, gestion et analyse des risques, gestion et détection des fraudes), dans d'autres applications (recherche de texte, dans des documents Web, dans l'astronomie scientifique). [3] [4]

Dans ce chapitre, nous définissons le modèle relationnel et ses limites dans la gestion du Big Web Data et de ses systèmes, analysons chacune des caractéristiques des référentiels et définissons les modèles conceptuels et logiques qui le composent.

1.2. Bases de données relationnelles

1.2.1. Définition :

Une base de données relationnelle (en anglais : Relational DataBase Management System (RDBMS)) est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables1, selon le modèle introduit par Edgar F. Codd en 1960. Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations. Les lignes de ces relations sont appelées des nuplets ou enregistrements. Les colonnes sont appelées des attributs. [5]

Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion de bases de données relationnelles (SGBDR).

Pratiquement tous les systèmes relationnels utilisent le langage SQL pour interroger les bases de données. Ce langage permet de demander des opérations d'algèbre relationnelle telles que l'intersection, la sélection et la jointure. [5]

1.2.2. Propriétés ACID :

L'acronyme ACID désigne les quatre propriétés clés qui définissent une transaction : **atomicité, cohérence, isolement** et **durabilité (voir figure 1)**. Si une opération de base de données présente ces propriétés ACID, elle peut être qualifiée de transaction ACID. Les systèmes de stockage de données qui appliquent ce type d'opérations sont appelés « systèmes transactionnels ». Avec les transactions ACID, chaque lecture, écriture ou modification d'une table présente les caractéristiques suivantes :[6]

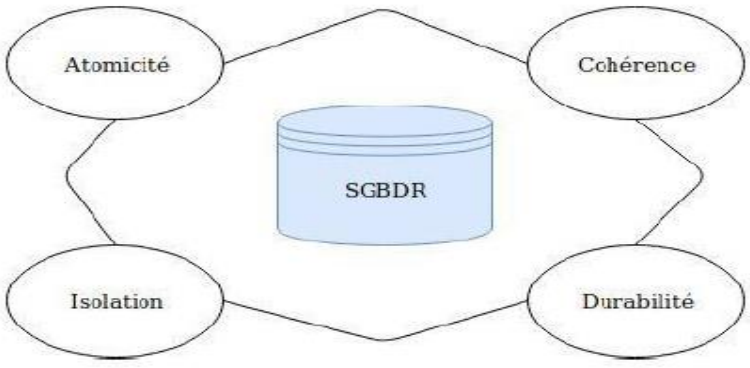

Figure 1 - Propriétés ACID [7]

1) Atomicité (Atomicity) :

Chaque instruction d'une transaction (lecture, écriture, mise à jour ou suppression de données) est traitée comme une seule unité. Soit l'instruction est exécutée dans son intégralité, soit elle ne l'est pas du tout. Cette propriété empêche que des données soient perdues ou corrompues si, par exemple, la source de données de streaming s'interrompt à mi-parcours.

2) Cohérence (Consistency) :

Les transactions ne modifient les tables que de façon prédéfinie et prévisible. La cohérence transactionnelle veille à ce qu'une corruption ou une erreur dans les données n'ait pas de conséquences inattendues sur l'intégrité de votre table.

3) Isolation (Isolation) :

Lorsque plusieurs utilisateurs lisent et écrivent dans la même table simultanément, l'isolement de leurs transactions garantit que les transactions effectuées simultanément ne s'impactent pas les unes les autres. Les requêtes sont exécutées comme si elles étaient successives alors qu'elles sont en réalité simultanées.

4) Durabilité (Durability) :

Les modifications apportées à vos données par les transactions réussies sont enregistrées, même en cas de défaillance du système.

1.2.3. Limites des bases de données relationnelles :

Dans ce qui suit, nous citons quelques limites relatives aux bases de données relationnelles [8] :

1) Difficultés avec les propriétés ACID dans un environnement distribué :

Coût élevé du stockage et des mises à jour : Garantir la cohérence entre plusieurs serveurs implique la duplication des données et des processus de mise à jour complexes, ce qui affecte les performances et les coûts.

2) Requêtes non optimales dues aux jointures :

Traitement lent de grandes quantités de données : Les jointures entre des tables volumineuses, comme la recherche d'amis sur Facebook, peuvent s'avérer très lentes et gourmandes en ressources.

3) Gestion limitée des objets hétérogènes :

Difficulté à modéliser des données complexes : Les modèles relationnels ne s'adaptent pas facilement aux structures de données complexes et variées, limitant leur flexibilité.

4) Surcharge sémantique :

Décalage entre la réalité et la représentation des données : Le modèle relationnel abstrait peut ne pas capturer pleinement la complexité des relations et des concepts du monde réel.

5) Types de données limités :

Prise en charge insuffisante des données complexes : Les modèles relationnels ne gèrent pas efficacement les données non structurées, multimédia ou complexes, comme les images, les vidéos ou les documents.

6) Langage de manipulation incomplet :

Nécessité de langages externes et problèmes de compatibilité des types : SQL n'est pas un langage de programmation complet, ce qui implique l'utilisation d'autres langages et peut engendrer des problèmes de conversion de types.

7) Incompatibilités de types ("impedance mismatch")

Conversions de types coûteuses : Des conversions de types entre le SGBD et le langage de programmation peuvent s'avérer nécessaires et gourmandes en ressources.

8) Pauvreté sémantique pour les données multimédia :

Difficulté à gérer des données complexes et multivaluées : Le modèle relationnel n'est pas adapté à la gestion efficace de données complexes comme les images, les vidéos ou les informations multimédia.

9) Problèmes de partitionnement de données :

Impact de la normalisation sur la distribution des données : La normalisation peut entraîner une dispersion des données liées, affectant les performances des requêtes et la gestion des transactions.

1.3. Le système de gestion de base de données (SGBD)

1.3.1. Définition :

Les systèmes de gestion des bases de données (SGBD) sont des programmes informatiques qui permettent aux utilisateurs d'interagir avec une base de données. Pour cela, le SGBD doit avoir un modèle qui définit la manière dont les données sont organisées. Le modèle relationnel est une approche d'organisation des données très populaire. [9]

Il existe de nombreux systèmes de gestion de bases de données, en voici une liste non exhaustive de Quelques SGBD relationnels connus et utilisées :

- **Oracle**
- **MySQL s**
- **Microsoft SQL Server**
- **PostgreSQL**
- **IBM Db2**

1.3.2. Limites Le système de gestion de base de données (SGBD)

Les bases de données relationnelles (SGBDR) sont des outils puissants pour stocker et gérer des données structurées. Cependant, elles présentent certaines limites, en particulier lorsqu'il s'agit de traiter des volumes de données volumineux, complexes ou distribués.

1) Difficultés de modélisation du monde réel [10]:

- **Entités complexes :** Les SGBDR peuvent avoir du mal à représenter des entités du monde réel complexes, comme des objets hiérarchiques ou en réseau.

- **Multiplication des relations :** La normalisation des données peut entrainer une multiplication des tables et des jointures, rendant les requêtes plus lentes.

2) Surcharge sémantique [11] :

- **Abstraction limitée :** Le modèle relationnel utilise un seul concept (la relation) pour tout représenter, ce qui peut simplifier excessivement la réalité.

3) Contraintes d'intégrité rigides [11] :

Difficulté à exprimer des règles complexes : Les SGBDR peuvent ne pas supporter toutes les contraintes d'intégrité métier souhaitées.

4) Langage de manipulation limité [11] **:**

Opérateurs SQL basiques : Le langage SQL ne permet pas d'étendre ses fonctionnalités à de nouveaux besoins métiers.

5) Scalabilité insuffisante pour de gros volumes de données [11] :

Problèmes de performance : Les SGBDR relationnels peuvent rencontrer des limitations de performance lorsqu'ils gèrent des ensembles de données volumineux.

6) Difficultés avec les propriétés ACID en environnement distribué [11] **:**

Coût élevé du maintien de la cohérence : Garantir la cohérence ACID dans un environnement distribué peut s'avérer coûteux et impactant sur les performances.

7) Gestion limitée des données complexes [11] :

- **Objets hétérogènes :** Les SGBDR relationnels ne sont pas conçus pour gérer efficacement des données complexes comme des images ou des vidéos.

- **Besoin de pré-définir les champs des objets :** La modélisation d'objets dans les bases de données relationnelles nécessite une définition préalable de tous les champs, ce qui peut être rigide.

1.4. Système d’Information Décisionnel

Un système d'information décisionnel, SID, est un système qui réalise la collecte, la transformation des données brutes issues de sources de données et le stockage dans d'autres espaces ainsi que la caractérisation des données résumées en vue de faciliter le processus de prise de décision. [32]

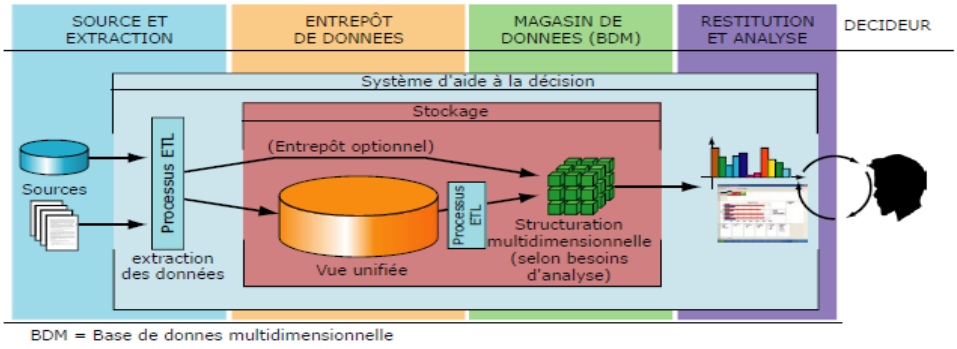

Figure 2- Architecture SID [13]

Les systèmes décisionnels peuvent avoir besoin de données issues de différentes sources et de différents formats de stockage. Ceci regroupe les fichiers texte, les rapports, les fichiers de base de données issus de différents systèmes de gestion de bases de données (SGBD), . . . etc.

Pour ce fait ces données doivent passer par le processus d'Extraction, Transformation et chargement **(ETL)** pour qu'elles soient exploitables. Ce processus peut être résumé comme suite :

1. Extract (E) : Dans la phase d'extraction seules les données destinées à l'exploitation pour l'analyse qui sont gardées en connectant aux différentes applications ou bases en production.

2. Transform (T) : Dans la phase de transformation, la mise au format des données, la fusion ou l'éclatement des informations et l'agrégation des données peuvent être effectuées.

3. Load (L) : Enfin, dans la phase de chargement, les informations sont stockées dans les entrepôts de données.

1.5. Les entrepôts de données

1.5.1. Définition

Un entrepôt de données est un type de système de gestion de données conçu pour permettre et soutenir les activités de veille stratégique, en particulier l'analyse. Les entrepôts de données sont uniquement destinés à l'exécution des interrogations et des analyses. Ils contiennent souvent de grandes quantités de données historiques. Les données dans un entrepôt de données sont habituellement issues d'un large éventail de sources, tels les fichiers journaux des applications et les applications de transaction. [14]

1.5.2. Architecture d'un entrepôt de données

L'architecture des entrepôts de données repose souvent sur un SGBD séparé du système de production de l'entreprise qui contient les données de l'entrepôt.

Le processus d'extraction des données permet d'alimenter périodiquement ce SGBD. Néanmoins avant d'exécuter ce processus, une phase de transformation est appliquée aux données opérationnelles. Celle-ci consiste à les préparer (mise en correspondance des formats de données). Les nettoyer, les filtrer, pour finalement aboutir à leur stockage dans l'entrepôt. [15]

Un entrepôt de données est l'espace de stockage centralise d'un extrait des sources de données pertinentes pour les décideurs. Son organisation doit faciliter la gestion des données sous la forme d'une vision unifiée et doit permettre la conservation des évolutions nécessaires pour les prises de décisions. [16]

Dans la Figure 3, nous présentons une architecture simplifiée d'un entrepôt. Les différents composants ont été intègres dans trois parties : les sources de données, l'entrepôt et les outils existants dans la marche. [2]

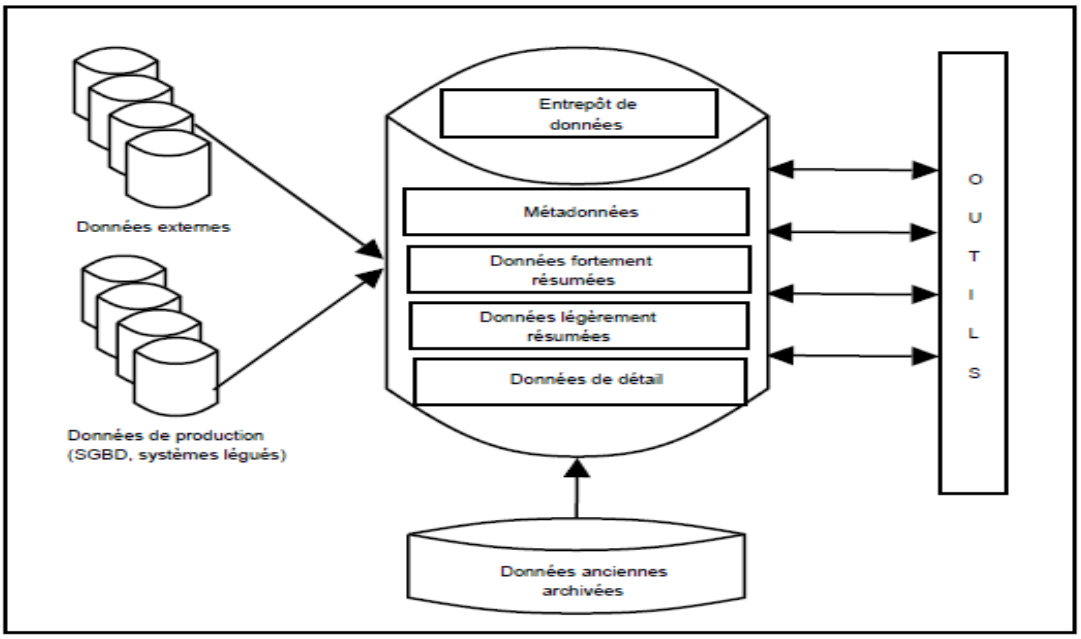

Figure 3 - Une architecture entrepôt de données [17]

- **Les sources**

Les données de l'entrepôt sont extraites de diverses sources souvent réparties et hétérogènes, et qui doivent être transformées avant leur stockage dans l'entrepôt. Nous avons deux types de sources des données [17]: internes et externes à l'organisation

- **Internes :** La plupart des données sont saisies à partir des différents systèmes de production qui rassemblent les divers SGBD opérationnels, ainsi que des anciens systèmes de production qui contiennent des données encore exploitées par l'entreprise.

- **Externes :** Ils représentent des données externes à l'entreprise et qui sont souvent achetées. Par exemple, les sources de données démographiques.

- **Les entrepôts de données ont une architecture à trois niveaux :**

Niveau inférieur : Le niveau inférieur est composé d'un serveur d'entrepôt de données, généralement un système de base de données relationnelle, qui collecte, nettoie et transforme les données provenant de plusieurs sources de données via un processus appelé Extraction, Transformation et Chargement (ETL ou parfois ELT).

Niveau intermédiaire : Le niveau intermédiaire est composé d'un serveur OLAP (traitement

analytique en ligne) qui permet des vitesses de requête rapides. Trois types de modèles OLAP : ROLAP, MOLAP et HOLAP.

Niveau supérieur : Le niveau supérieur est représenté par une sorte d'interface utilisateur frontale ou d'outil de rapports, qui permet aux utilisateurs finaux d'effectuer des analyses de données ad hoc sur leurs données métier.

- **Outils :**

Il existe sur la marche différents outils pour l'aide à la décision, comme les outils de fouille données ou datamining (pour découvrir des liens sémantiques), outils d'analyse en ligne (pour la synthèse et l'analyse des données multidimensionnelles), outils d'interrogation (pour faciliter l'accès aux données en fournissant une interface conviviale au langage de requêtes), [17] ...

1.5.3. Composantes d'un entrepôt de donnée

Un entrepôt de donnée est généralement composé de plusieurs éléments clés qui permettent de stocker, organiser et analyser les données.

- **Sources de données** : les sources de données sont les différentes sources d'où les données sont extraites et chargées dans le Data Warehouse. Ces sources peuvent inclure des systèmes opérationnels, des fichiers plats, des bases de données, des applications, des services web, etc.
- **Extraction, transformation et chargement (ETL)** : l'ETL est le processus qui permet d'**extraire** les données des sources, de les **transformer** en un format standardisé et de les **charger** dans le Data Warehouse. Ce processus implique souvent la suppression des doublons, la normalisation des données et la vérification de leur qualité.
- **Stockage de données** : les données sont stockées dans le Data Warehouse de manière à faciliter leur analyse. Les données peuvent être stockées dans des tables, des vues, des cubes ou des fichiers.
- **Modèle de données** : le modèle de données est la structure qui définit la manière dont les données sont organisées dans le Data Warehouse. Le modèle de données peut être en étoile, en flocon ou en constellation.
- **Outils d'analyse** : enfin, les outils d'analyse sont utilisés pour interroger le Data Warehouse et fournir des informations précieuses sur les activités de l'entreprise. Ces outils peuvent

inclure des rapports, des tableaux de bord, des graphiques, des diagrammes, des analyses statistiques, etc.

1.5.4. Type L'entrepôt de données

Il existe plusieurs types de données dans un entrepôt, qui correspondent à diverses utilisations, comme :

- **Les données de détail courantes** : Elles comprennent les informations quotidiennes très utilisées et souvent stockées sur disque pour un accès rapide, comme les ventes actuelles dans différents magasins.

- **Les données de détail anciennes** : Elles concernent les événements passés et sont utilisées pour des analyses de tendances ou prévisionnelles, comme les ventes des années précédentes, souvent stockées dans des archives.

- **Les données résumées ou agrégées** : Elles sont moins détaillées et permettent de réduire le volume de données stockées. Elles peuvent être légèrement ou fortement résumées, par exemple les ventes mensuelles par magasin versus les ventes semestrielles par région sur plusieurs années.

- **Les métadonnées** : Elles sont indispensables pour exploiter efficacement le contenu d'un entrepôt de données. Elles fournissent des informations sur la signification des données, leur origine, les règles d'agrégation, le format de stockage (par exemple, en euro ou en francs), et leur utilisation par les programmes.

1.5.5. Schéma multidimensionnel

Pour arriver à construire un modèle approprié pour un entrepôt de données, nous Pouvons choisir, soit un schéma relationnel (le schéma en étoile, en flocon de neige ou en constellation) ; soit un schéma multidimensionnel. Avant de décrire les différents schémas, nous commençons par quelques concepts de base[17].

La modélisation multidimensionnelle consiste à considérer un sujet analysé comme un point dans un espace à plusieurs dimensions. Les données sont organisées de manière à mettre en évidence le sujet (le fait) et les différentes perspectives de l’analyse (Les

dimensions).

1) Niveau conceptuel :

- **Concept de fait :**

Le **fait** modélise le sujet de l'analyse. Un fait est formé de **mesures** correspondant aux informations de l'activité analysée.

— Fait : c'est le concept qui modélise le sujet de l'analyse. Il est formé des informations de l'activité analysée.

— Mesure : C'est la valeur d'attribut quantitatif ou qualitatif qui évalue un fait.

Nous distinguons trois types de mesures :

1. Mesures de type < flux > : mesures qui peuvent être sommées selon toutes les dimensions (e.g. "montant de vente").

2. Mesures de type < stock > : mesures qui ne peuvent pas être sommées selon certaines dimensions (e.g. "population").

3. Mesures de type < valeur par unité > : mesures qui ne peuvent jamais être sommées (e.g. "température").

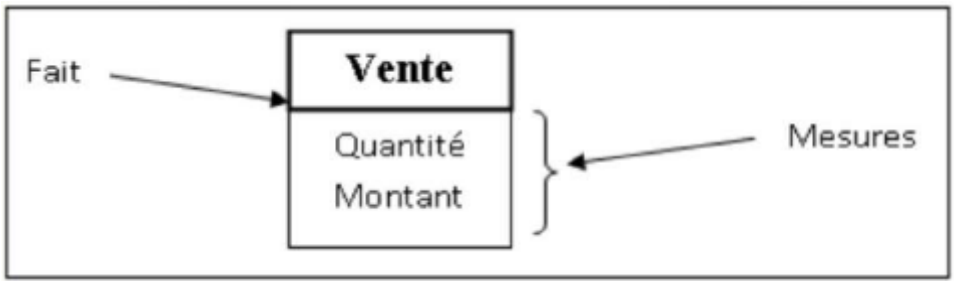

Figure 4 - exemple fait et mesure

EXEMPLE : Considérons le fait de *Vente* pouvant être constitué des mesures d'activités suivantes : quantité de produits vendus et montant total des ventes. Nous représenterons le fait par un rectangle englobant les différentes mesures d'activité qu'il contient. En outre le symbole d'un cube estampille le fait.

- **Concept de dimension**

Une **dimension** modélise une perspective de l'analyse. Une dimension se compose de **paramètres** (ou attributs) correspondant aux informations faisant varier les mesures de l'activité.

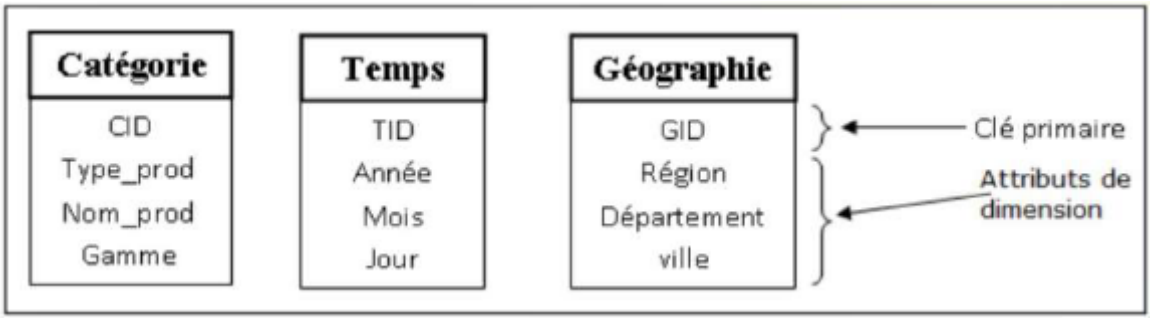

Figure 5 - exemple dimension et attributs

EXEMPLE : Poursuivons l'exemple précédent. Le fait peut être analysé suivant différentes perspectives correspondant à trois dimensions : la dimension *Temps*, la dimension *Géographie* et la dimension *Catégorie*.

Nous représenterons une dimension par un rectangle englobant les différents paramètres qu'elle contient. En outre un symbole représentant trois axes estampille les dimensions pour les distinguer du fait.

- **Concept Hiérarchie dimensionnelle**

Les analyses multidimensionnelles utilisent des hiérarchies d'attributs pour visualiser les données à déférents niveaux de détail. Ces hiérarchies organisent les attributs entre eux selon une dépendance de hiérarchie. Par exemple, dans la dimension temps, il est possible de trouver déférentes hiérarchisations :

_ Année ➔ Mois ➔Semaine ➔Jour

_ Année ➔Trimestre ➔Mois

- **Schémas relationnels**

Dans les schémas relationnels nous trouvons deux types de schémas. Les premiers sont des schémas qui répondent fort bien aux processus de type OLTP qui ont été décrits précédemment, alors que les deuxièmes, que nous appelons des schémas pour le décisionnel, ont pour but de proposer des schémas adaptés pour des applications de type OLAP.

Nous décrivons les différents types des schémas relationnels pour le décisionnel.

- **Le schéma en étoile :**

Il se compose du fait central et de leurs dimensions. Dans ce schéma il existe une relation pour les faits et plusieurs pour les différentes dimensions autour de la relation centrale. La relation de faits contient les différentes mesures et une clé étrangère pour faire référence à chacune de leurs dimensions. [17]

La figure 6 montre le schéma en étoile en décrivant les ventes réalisées dans les différents magasins de l'entreprise au cours d'un jour. Dans ce cas, nous avons une étoile centrale avec une table de faits appelée Ventes et autour leurs diverses dimensions : Temps, Produit et Magasin.

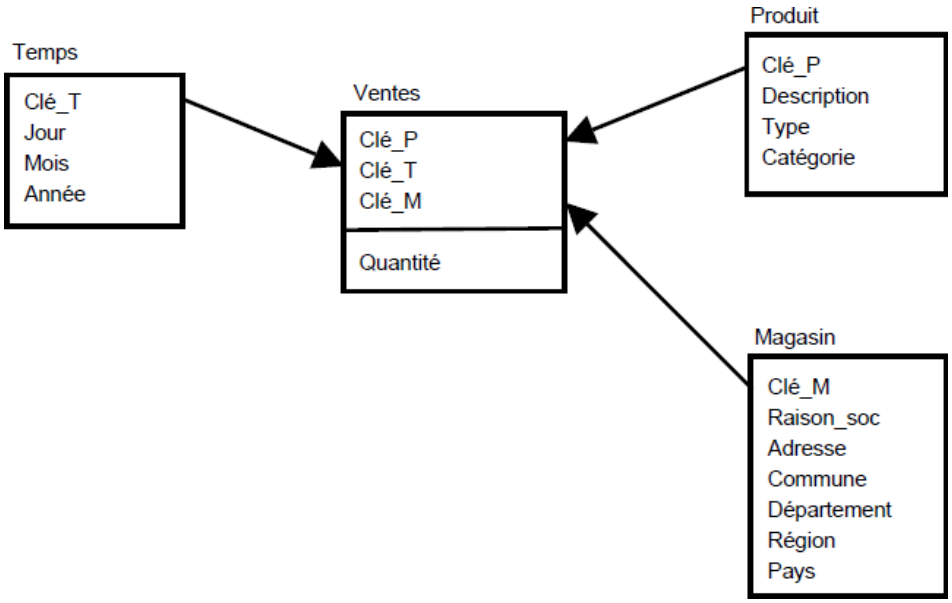

Figure 6 - Exemple de modélisation en étoile. **[17].**

- **Le schéma en flocon :**

Il dérive du schéma précédent avec une relation centrale et autour d'elle les différentes dimensions, qui sont éclatées ou décomposées en sous hiérarchies. L'avantage du schéma en flocon de neige est de formaliser une hiérarchie au sein d'une dimension, ce qui peut faciliter l'analyse. Un autre avantage est représenté par la normalisation des dimensions, car nous réduisons leur taille. En effet, ce type de schéma augmente le nombre

de jointures à réaliser dans l'exécution d'une requête. [17].

La figure 7 montre le schéma en flocon de neige avec les dimensions Temps et magasin éclatées en sous hiérarchies.

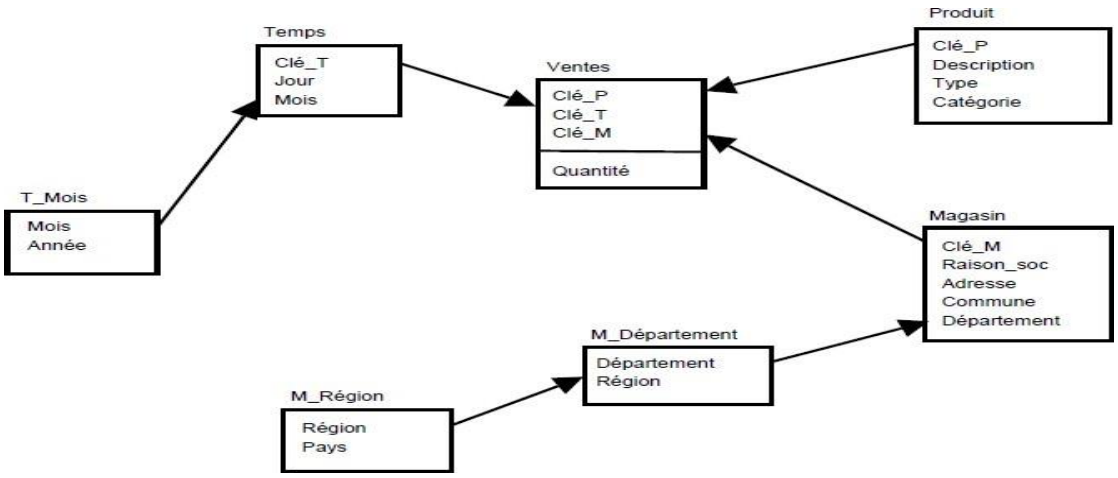

Figure 7 - Exemple de modélisation en flocon de neige. **[17].**

- **Le schéma en constellation**

Le schéma en constellation représente plusieurs relations de faits qui partagent des dimensions communes. Ces différentes relations de faits composent une famille qui partage les dimensions mais où chaque relation de faits à ses propres dimensions. [17].

La figure 8 montre le schéma en constellation qui est composé de deux relations de faits. La première s'appelle Ventes et enregistre les quantités de produits qui ont été vendus dans les différents magasins pendant un certain jour. La deuxième relation gère les différents produits achetés aux fournisseurs pendant un certain temps.

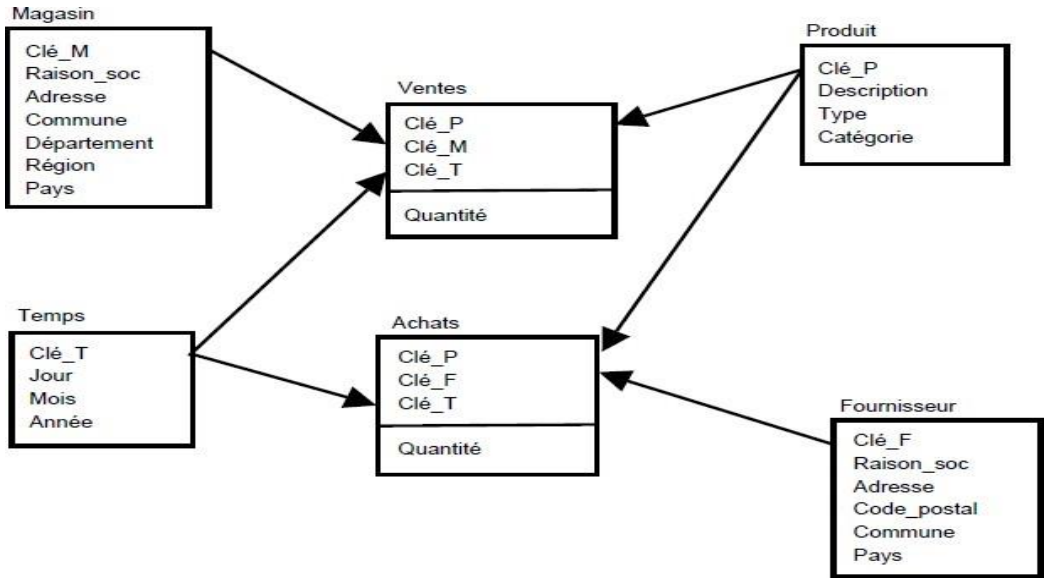

Figure 8 - Exemple de modélisation en constellation **[17].**

- **Schéma multidimensionnel (Cube)**

Dans le modèle multidimensionnel, le concept central est le cube, lequel est constitué des éléments appelés cellules qui peuvent contenir une ou plusieurs mesures [17].

La localisation de la cellule est faite à travers les axes, qui correspondent chacun à une dimension. La dimension est composée de membres qui représentent les différentes valeurs. En reprenant une partie du schéma en étoile de la Figure 9, nous pouvons construire le schéma multidimensionnel suivant.

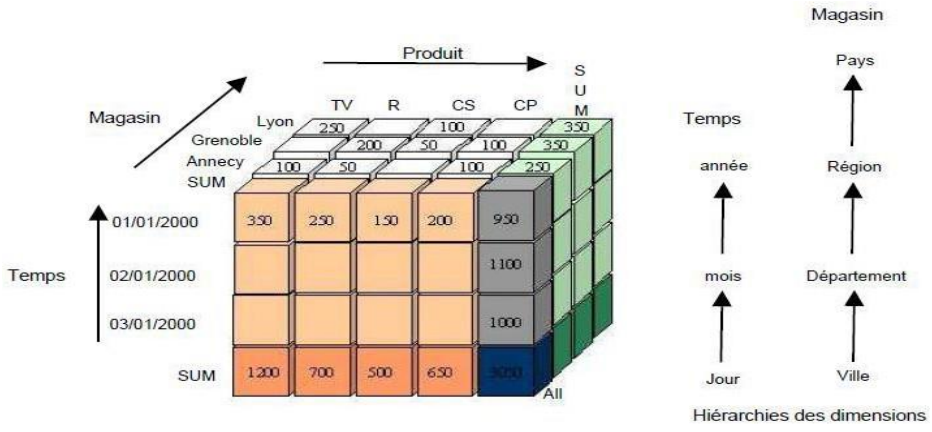

Figure 9 - Exemple de schéma multidimensionnel. **[17].**

2) Niveau logique : [17].

1. Serveurs OLAP (On-Line Analytical Processing)

Les données opérationnelles constituent la source principale d'un système d'information décisionnel. Les systèmes décisionnels complets reposent sur la technologie OLAP, conçue pour répondre aux besoins d'analyse des applications de gestion

Nous exposons dans la suite les divers types de stockage des informations dans les systèmes décisionnels.

- **ROLAP (Relational OLAP)**

Cette méthode repose sur le fonctionnement des données stockées dans la base de données relationnelle pour présenter la fonction de découpage OLAP. Essentiellement, chaque opération de découpage équivaut à ajouter une clause « WHERE » à l'instruction SQL. [17]

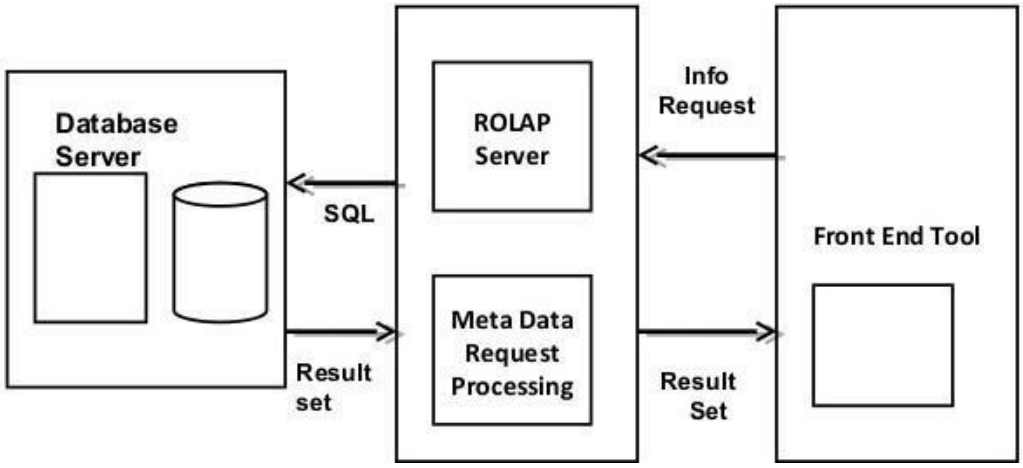

Figure 10 - Architecture ROLAP **[17].**

- **MOLAP (Multidimensionnel OLAP)**

Il s'agit de la méthode d'analyse OLAP la plus traditionnelle. Dans MOLAP, les données sont stockées dans des cubes multidimensionnels. Le stockage n'est pas dans une

base de données relationnelle, mais dans un format propriétaire. [17]

La figure 10 montre une architecture pour les systèmes MOLAP.

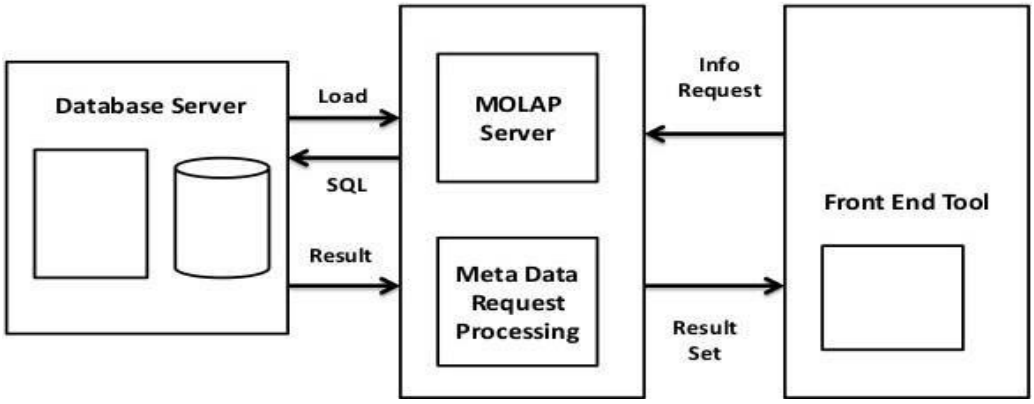

Figure 11 - Architecture MOLAP. **[17].**

– **HOLAP (Hybrid OLAP)**

Un système HOLAP est un système qui supporte et intègre un stockage des données multidimensionnel et relationnel d'une manière équivalente pour profiter des caractéristiques de correspondance et des techniques d'optimisation. [17]

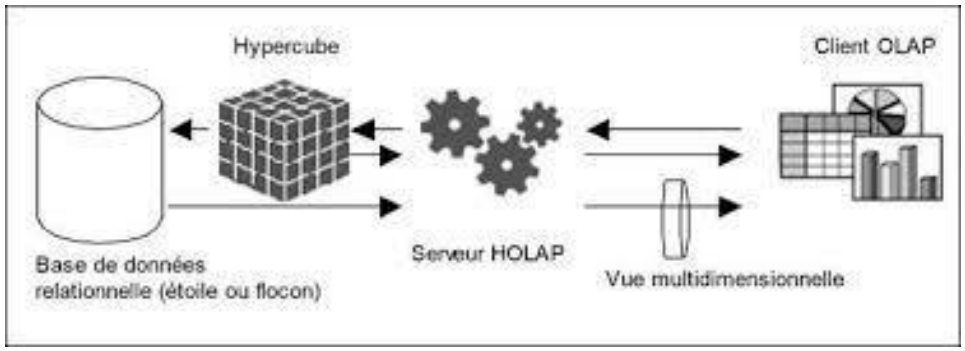

Figure 12 - Architecture HOLAP **[17].**

2. Limite des systèmes OLAP

Les systèmes OLAP ont montré leur efficacité notamment pour assurer l'analyse multidimensionnelle des données entreposée, non volatiles et à différents niveaux de granularité. Ils sont considérés comme outils d'aide à la décision.

Néanmoins, ils présentent aussi des difficultés remarquables aux différents stades de modélisation notamment conceptuelle, logique et physique.

- Les systèmes OLAP ne font pas la mise à jour des données, mais seulement l'archivage des versions.
- On se retrouve parfois devant des situations de gestion de Big data.
- Ceci est relatif à la périodicité de l'application du processus ETL.
- Les techniques et méthodes proposés pour résoudre ces problèmes sont efficaces dans quelques cas.
- Des propositions font complexifier encore ces outils destinés au départ pour l'analyse rapide et facile des données.
- problèmes se posent relatives à la qualité des analyses effectuées par les utilisateurs.

Cette qualité dépond de plusieurs facteurs notamment la consistance des données entreposées, la consistance des agrégations et la consistance de l'exploitation c'est-à-dire la formulation des requêtes.

1.6. Conclusion

Dans ce chapitre, nous avons parlé des bases de données relationnelles et des systèmes de gestion de bases de données (SGBD), de leurs limites, de ce que sont les entrepôts de données, et avons développé certains concepts et techniques de bases de données traditionnelles pour créer des systèmes d'aide à la décision.

Nous avons décrit différents modèles multidimensionnels pour construire un entrepôt de données, ainsi que différentes opérations de traitement de données multidimensionnelles.

Dans le chapitre suivant, nous parlerons des bases de données NOSQL. Il s'agit d'une nouvelle technologie dans le monde des données relationnelles et du Big Data et de la solution la plus adaptable aux limitations des bases de données. Et discutons de l'État de l'art sur le passage du modèle conceptuel multidimensionnel vers le modèle logique NoSql.

Chapitre 2

System NoSQL

2.1. Introduction

Dans ce chapitre consistera à expliquer ce que sont des données NoSQL, ensuite nous décrivons et les différents de SQL vers le NoSQL ainsi que les caractéristiques de données NoSQL les types de bases de données NoSQL. Nous terminerons le chapitre par présentons un Etape I : état de l'art sur :

- Techniques de passage du modèle conceptuel multidimensionnel vers modèle logique NoSql

- Exemples réels d'implémentation d'EDD Nosql.

2.2. Les bases de données non relationnelles

2.2.1. Définition

NoSQL est une base de données non relationnelle qui stocke et accède aux données en utilisant des valeurs clés. Au lieu de stocker des données dans des lignes et des colonnes comme une base de données traditionnelle, un SGBD NoSQL stocke chaque élément individuellement avec une clé unique. De plus, une base de données NoSQL ne nécessite pas de schéma structuré définissant chaque table et les colonnes associées. Cela offre une approche beaucoup plus flexible du stockage des données qu'une base de données relationnelle. La base de données NoSQL signifie « Pas seulement SQL ». [20]

2.2.2. La Technologie NoSQL pour SQL

NoSQL est une combinaison de deux mots : NO et SQL. Qui pourrait être mal interprétée, car l'on pourrait penser que cela signifie la fin du langage SQL et qu'on ne devrait donc plus l'utiliser. En fait, le « No » est un acronyme qui signifie « Not only »,c'est-à-dire en français, « non seulement »,c'est donc une manière de dire qu'il y a autre chose que les bases de données relationnelles. [12]

NoSQL est un mouvement très récent, qui concerne les bases de données. L'idée du mouvement est simple : proposer des alternatives aux bases de données relationnelles pour coller aux nouvelles tendances et architectures du moment, notamment le Cloud Computing. Les axes principaux du NoSQL sont une haute disponibilité et un partitionnement horizontal des données, au détriment de la cohérence. Alors que les bases de données relationnelles actuelles sont basées sur les propriétés ACID (Atomicité, Consistance ou Cohérence, Isolation et Durabilité).

En effet, NoSQL ne vient pas remplacer les BD relationnelles, mais proposer une alternative ou compléter les fonctionnalités des SGBDR pour donner des solutions plus intéressantes dans certains contextes. Le NoSQL regroupe de nombreuses bases de données, récentes pour la plupart, qui se différencient du modèle SQL par une logique de représentation de données non relationnelle. Leurs principaux avantages sont leurs performances et leur capacité à traiter de très grands volumes de données. [22]

2.2.3. Choisir NoSQL

Le besoin fondamental auquel répond le NoSQL est la performance. Afin de résoudre les problèmes liés au « big data », les développeurs de sociétés telles que Google et Amazon ont procédé à des compromis sur les propriétés ACID des SGBDR. Ces compromis sur la notion relationnelle ont permis aux SGBDR de se libérer de leurs freins à la scalabilité horizontale. Un autre aspect important du NoSql est qu'il répond au théorème de CAP qui est plus adapté pour les systèmes distribués.

Le NOSQL sert à manipuler de gros volumes de données destinés aux grandes entreprises. Dès 2010, le NoSQL a commencé à s'étendre vers les plus petites entreprises. Majoritairement des start-ups n'ayant pas les moyens d'acquérir des licences Oracle qui ont donc développé leurs propres SGBD en imitant les produits Google et Amazon [23].

2.2.4. NoSQL versus SQL

• SQL prononcé comme « S-Q-L » ou comme « Sée-Quel » est principalement appelé SGBDR ou bases de données relationnelles alors que NoSQL est une base de données non relationnelle ou distribuée.

• En comparant les bases de données SQL et NoSQL, les bases de données SQL sont des bases de données basées sur des tables, tandis que les bases de données NoSQL peuvent être basées sur des documents, des paires clé-valeur, des bases de données graphiques.

• Les bases de données SQL sont évolutives verticalement tandis que les bases de données NoSQL sont évolutives horizontalement.

• Les bases de données SQL a un schéma prédéfini tandis que les bases de données NoSQL utilisent un schéma dynamique pour les données non structurées.

• En comparant les performances de NoSQL à celles de SQL, SQL nécessite un matériel de base de données spécialisé pour de meilleures performances, tandis que NoSQL utilise du matériel de base.

Tableau 1 - différences entre SQL et NoSQL [24]

SQL	NoSQL
Les bases de données SQL sont évolutives verticalement (scalabilité verticale). Ils peuvent être mis à l'échelle en augmentant la capacité matérielle (CPU, RAM, SSD, etc.) sur un seul serveur.	Les bases de données NoSQL sont évolutives horizontalement (scalabilité horizontale). Ils peuvent être mis à l'échelle en ajoutant plus de serveurs à l'infrastructure pour gérer une charge importante et réduire le tas.
Les bases de données SQL sont principalement des bases de données relationnelles (SGBDR).	Les bases de données NoSQL sont principalement des bases de données non relationnelles ou distribuées.
Une technologie vieillie.	Technologie relativement jeune.
Ils ont un schéma prédéfini bien conçu pour les données structurées.	Ils ont le schéma dynamique pour les données non structurées. Les données peuvent être stockées de manière flexible sans avoir une structure prédéfinie (Schemaless).
Coûteux à l'échelle.	Moins cher à l'échelle que les bases de données relationnelles.
Ils conviennent parfaitement aux requêtes complexes, car SQL dispose d'une interface standard pour gérer les requêtes. La syntaxe des requêtes SQL est fixe.	Ce n'est pas un bon choix pour les requêtes complexes car il n'y a pas d'interface standard dans NoSQL pour gérer les requêtes. Les requêtes dans NoSQL ne sont pas aussi puissantes que les requêtes SQL. Il est appelé UnQL et la syntaxe d'utilisation du langage de requête non structuré varie d'un SGBD à l'autre
Les bases de données SQL ne conviennent pas au stockage hiérarchique des données.	Les bases de données NoSQL conviennent le mieux pour le stockage hiérarchique des données car elles suivent la méthode de la paire clé valeur pour stocker les données.
Les bases de données SQL suivent correctement les propriétés ACID.	Les bases de données NoSQL suivent correctement le théorème CAP de Brewers.
L'ajout de nouvelles données dans la base de données SQL nécessite des modifications telles que le remplissage des données, la modification des schémas.	De nouvelles données peuvent être facilement insérées dans les bases de données NoSQL car elles ne nécessitent aucune étape préalable.
Ne convient pas au stockage hiérarchique des données.	Convient pour le stockage hiérarchique des données et le stockage de grands ensembles de données (par exemple, Big Data).
Exemple de bases de données SQL : MySQL, Oracle, MS-SQL, SQLite.	Exemples de bases de données NoSQL : MongoDB, Apache CouchDB, Redis, HBase.

2.2.5. Théorème de CAP

Contrairement à une base de données relationnelle et pour fonctionner avec une architecture distribuée, les moteurs NoSQL ne respectent pas les propriétés ci-dessus. Ils sont conçus pour s'appuyer sur le **théorème CAP (Consistency, Availability, Partition tolerance)**. Ce théorème définit trois exigences de base qui sont nécessaires à une relation informatique (la cohérence, la disponibilité et la tolérance au partitionnement) lors de la conception d'applications dans une architecture distribuée (voir figure 13). [25]

- **Cohérence (Consistency)**

Impose que la base de données doit être cohérente après chaque opération. Ce qui signifie que toutes les applications ou « clients » accèdent à la même donnée (valeur) à un instant T, et que ceux-ci sont cohérents entre eux.

- **Disponibilité (Availability)**

Impose que le système soit toujours disponible, et que même en cas de panne le système ne connaisse aucun arrêt de service, ce qui garantit la continuité de service.

- **Résistance au partitionnement (Partition tolerance)**

Permet au système de continuer à fonctionner même en cas de problème de communication entre les serveurs (ex : problème de réseau). Les serveurs sont partitionnés en plusieurs groupes qui ne pourront pas communiquer entre eux.

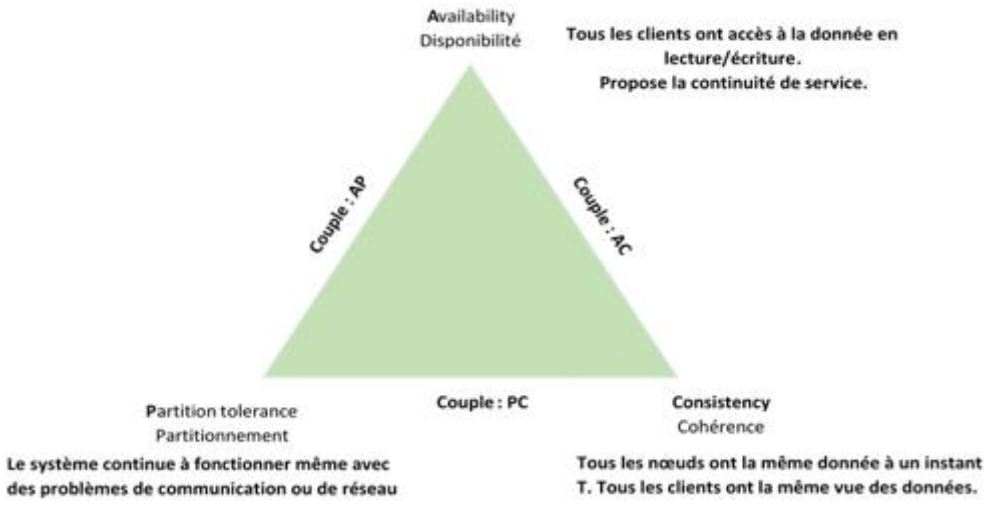

Figure 13- Théorème CAP

Le premier constat est qu'un NoSQL ne pourra pas satisfaire toutes les exigences **et ne pourra répondre qu'à 2 des 3 exigences** ci-dessus. Par conséquent, **les moteurs NoSQL répondent à un couple de combinaisons de C, A, P** du théorème CAP (voir figure 14).

Consistency & Availability, soit le couple AC : cluster de site unique, donc tous les nœuds sont toujours en contact. Lorsqu'une partition se produit, le système s'arrête (pas de gestion du split-brain).

Consistency & Partition Tolerance, soit le couple PC : Certaines données peuvent ne pas être accessibles, mais les données restent cohérentes. Moteur de type CP : Mongo DB, Hbase, Redis

Availability & Partition Tolerance, soit le couple AP : Le système est toujours disponible même sous partitionnement, mais certaines des données retournées peuvent être inexactes. Moteur de type CP : Cassandra, Couche DB, DynamoDB.

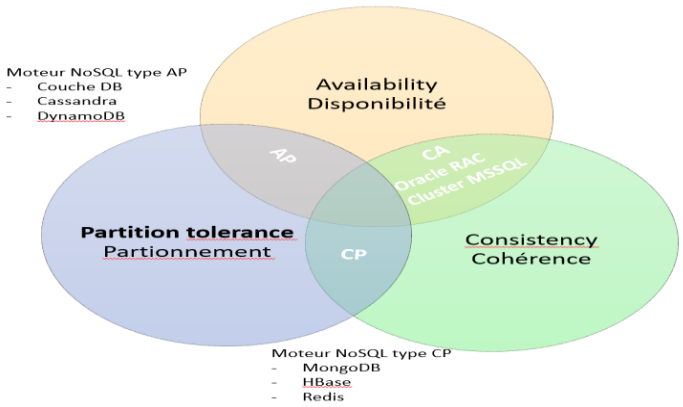

Figure 14 - Répartition des bases NoSQL selon théorème CAP

Note : ce théorème est toujours soumis à discussion et considéré comme « simpliste » car il impose d'abandonner la cohérence au profit de la disponibilité. Des travaux sont réalisés afin de le faire évoluer, notamment avec l'étude **du théorème PACELC (Partition tolerance, Availability, Consistency, Availability Else Latency or Consistency)** qui offre un arbitrage entre la latence et la cohérence.

2.2.6. Caractéristiques de NoSQL

1) **Non relationnel**

- Les bases de données NoSQL ne suivent jamais le modèle relationnel
- Ne fournit jamais de tables avec des enregistrements plats à colonne fixe
- Travailler avec des agrégats autonomes ou des BLOB (binary large object)
- L'intégrité référentielle se joint à l'ACID [26]

2) **Sans schéma**

- Les bases de données NoSQL sont exemptes de schémas ou ont des schémas détendus
- N'exigent aucune définition du schéma des données
- Offre des structures hétérogènes de données dans le même domaine

3) **API simple**

- Offre des interfaces faciles à utiliser pour le stockage et l'interrogation des données fournies
- Protocoles textuels principalement utilisés avec France REST avec JSON
- Utilise la plupart du temps aucun langage de requête standard
- Bases de données Web fonctionnant sous forme de services Internet

4) **Réparti**

- Plusieurs bases de données NoSQL peuvent être exécutées de manière distribuée
- Offre des capacités d'auto-mise à l'échelle et d'échec
- Souvent, le concept ACID peut être sacrifié pour l'évolutivité et le débit
- Fournir seulement la cohérence éventuelle

2.2.7. Les propriétés BASE

Dans le premier chapitre consacré aux bases de données relationnelles, nous avons rappelé les propriétés ACID auxquelles doivent répondre les SGBD de type relationnel.

Les SGBD NoSQL qui, selon le théorème CAP, privilégient la disponibilité ainsi que la tolérance à la partition plutôt que la cohérence, répondent aux propriétés de BASE.

Le principe de BASE est le fruit d'une réflexion menée par Eric Brewer [12]

Les caractéristiques de BASE sont fondées sur les limites que montrent les SGBD relationnelles. Voici sa description [12] :

1) **Basically Available (Disponibilité basique)**

Même en cas de désynchronisation ou de panne d'un des nœuds du cluster, le système reste disponible selon le théorème CAP.

2) **Soft-state (Cohérence légère)**

Cela indique que l'état du système risque de changer au cours du temps, sans pour autant que des données soient entrées dans le système.

3) **Eventual consistancy (Cohérence à terme)**

Cela indique que le système devient cohérent dans le temps, pour que pendant ce laps de temps, le système ne reçoive pas d'autres données.

2.2.8. Types de bases de données NoSQL [29]

Les bases de données NoSQL peuvent être classées en utilisant différentes approches et divers critères. Certains experts les classent selon leur modèle de données et la plupart d'entre eux décrivent quatre grands types de bases de données NoSQL : bases de données orientés clé-valeur, bases de données orientés documents, bases de données orientées colonnes et bases de données orientés graphes. Dans les sections suivantes, nous présentons ces quatre grands types.

1) **Bases de données orientés clé-valeur**

Ce modèle est implémenté à l'aide d'une table de hachage où il y a une clé unique et un pointeur vers un élément de données particulier en créant une paire clé-valeur (voir Figure 15). La table de hachage convient aux recherches pour des valeurs simples ou complexes dans des ensembles de données extrêmement volumineux.

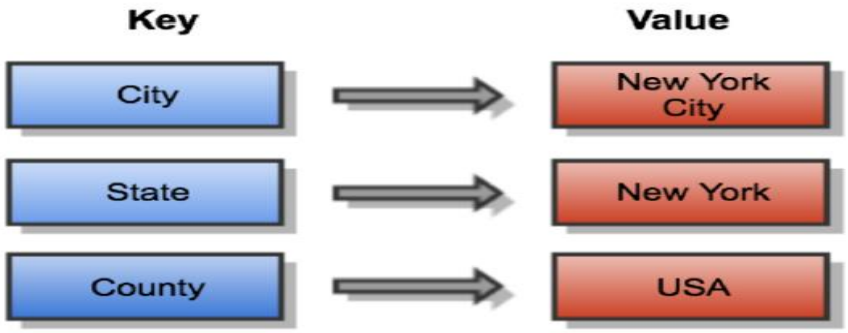

Figure 15 - Modèle orienté clé-valeur **[29]**

Exemples de cas d'utilisation :

•Gestion des sessions

•Profils, préférences & configurations

•Mise en cache des données

•Stockage de fichiers multimédias ou d'objets volumineux.

Nous choisissons la base de données orientée clés-valeurs si on a besoin :

1. Schéma simple
2. Lecture/écriture à grande vitesse sans mises à jour fréquentes
3. Hautes performances et évolutivité
4. Pas de requêtes complexes impliquant plusieurs clés ou jointures. [29]

2) Bases de données orientés documents

Les bases de données orientés documents ont été conçues pour gérer le stockage et la gestion des documents à grande échelle. Ce type de base de données attribue une valeur clé à chaque document. Les documents peuvent contenir plusieurs paires clé-valeur, ou paires clé-tableau, ou même les documents imbriqués (voir FIG 16). Les documents sont encodés dans un standard de format d'échange de données tel que XML, JSON ou BSON

Figure 16 - Modèle orienté document **[29]**

Les bases de données orientés documents sont reconnues comme un outil puissant, flexible et agile pour stocker les BigData.

Exemples de cas d'utilisation :

•Systèmes de gestion de contenu

•Sites de commerce électronique

•Applications middleware qui utilisent JSON.

On choisir la base de données orientée documents si on besoin :

1. Schéma flexible avec requêtes complexes

2. Formats de données JSON/BSON ou XML

3. Tirez parti d'index complexes (multi-clés, géo spatiales, recherche plein texte, etc.)

4. Haute performance et rapport R : W équilibré. [29]

3) Bases de données orientées colonnes

Bases de données orientée colonnes ou à colonnes étendues (également appelées magasins d'enregistrement extensible) représentent une extension de l'architecture clé-valeur avec des colonnes (voir Figure 17). Ils ont été conçus pour traiter des données sur un pool d'infrastructures. Les bases de données à colonnes étendues reposent sur une approche hybride qui s'appuie sur les caractéristiques déclaratives des bases de données et divers schémas de magasins clé-valeur.

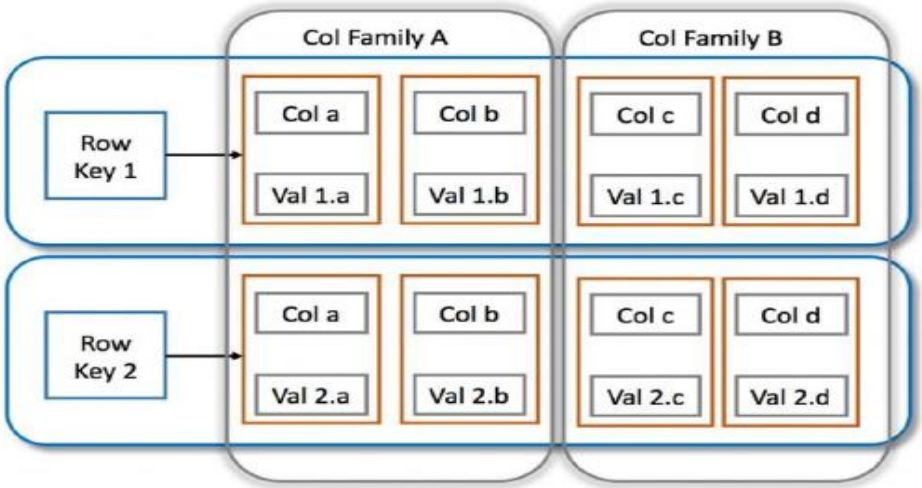

Figure 17 - Modèle orienté colonnes **[29]**

Exemples de cas d'utilisation :

•Données de séries chronologiques

•Applications IoT

•Journalisation et autres applications lourdes en écriture.

On choisir la base de données orientée colonnes si on a besoin :

1. Volume élevé de données

2. Vitesses d'écriture extrêmes avec des lectures relativement moins rapides

3. Extractions de données par colonnes à l'aide de clés de ligne

4. Pas de modèles de requête ad-hoc, d'indices complexes ou de niveau élevé d'agrégations.

4) Bases de données orientées graphes

Les bases de données orientées graphes sont adaptées pour stocker non seulement des informations sur les objets mais aussi toutes les relations qui existent entre eux (voir Figure 18). Ils reposent sur un modèle de graphe sans schéma afin de modéliser et représenter facilement les données connectées. Telle modèle comprend des sommets (par exemple, des objets ou des éléments représentés par nœuds) et des arêtes pour représenter les connexions entre les données.

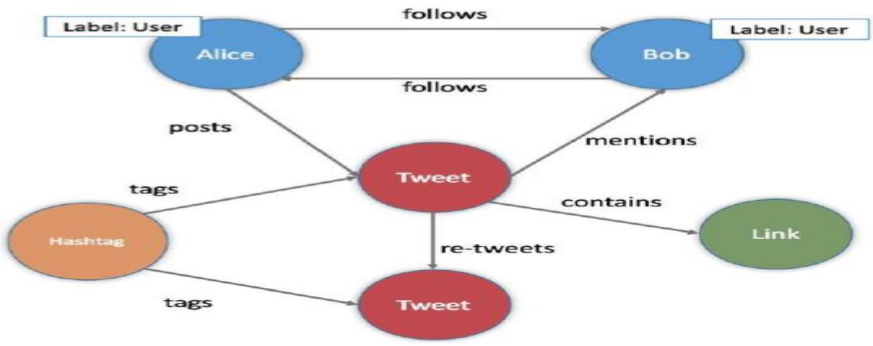

Figure 18 - Modèle orienté graphes **[29]**

Exemples de cas d'utilisation :

•Services basés sur la localisation et la navigation

•Infrastructure réseau et informatique

•Détection de fraude

•Gestion des métadonnées. [29]

On choisir la base de données orientée graphes si on besoin :

1. Applications nécessitant une traversée entre les points de données

2. Possibilité de stocker les propriétés de chaque point de données

3. Requêtes complexes pour déterminer les relations entre les points de données

4. Besoin de détecter des modèles entre les points de données. [29]

2.2.9. Les avantage du NoSQL

- **Plus évolutif**

NoSQL est plus évolutif. C'est en effet l'élasticité de ses bases de données NoSQL qui le rend si bien adapté au traitement de gros volumes de données. Au contraire, les bases de données relationnelles ont souvent tendance à utiliser la scalabilté verticale, quand celui-ci atteint ses limites [30].

- **Plus flexible**

N'étant pas enfermée dans un seul et unique modèle de données, une base de données NoSQL est beaucoup moins restreinte qu'une base SQL. Les applications NoSQL peuvent donc stocker des données sous n'importe quel format ou structure, et changer de format en production. En fin de compte, cela équivaut à un gain de temps considérable et à une meilleure fiabilité. Par contre une base de données relationnelle doit être gérée attentivement, car un changement, aussi mineur, peut entraîner un ralentissement ou un arrêt du service [30]

- **Plus économique**

Les serveurs destinés aux bases de données NoSQL sont généralement bon marché contrairement à ceux qui sont utilisés par les bases relationnelles. De plus, la très grande majorité des solutions NoSQL sont Open Source, ce qui reflète une économie importante sur le prix des licences [30]

- **Plus simple**

Les bases de données NoSQL ne sont pas forcément moins complexes que les bases relationnelles, mais elles sont beaucoup plus simples à déployer. La façon dont elles ont été conçues permet une gestion beaucoup plus légère [30]

2.3. Etat de l'Art sur

2.3.1. Techniques de passage du modèle conceptuel multidimensionnel vers modèle logique NoSql

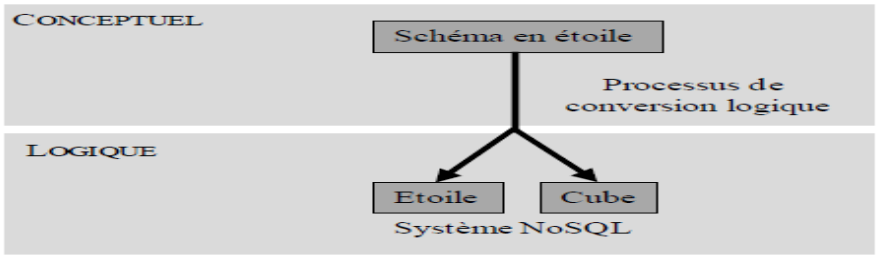

Figure 19 - Processus de conception

Le figure 19 que vous avez fourni illustre le processus de transformation des modèles conceptuels en modèle logiques dans un système NoSQL.

1) Entrepôts de données sous NoSQL

Dans cette section, nous présentons les travaux de recherche portant sur le développement des entrepôts de données avec ces nouveaux systèmes (cf. Figure 20).

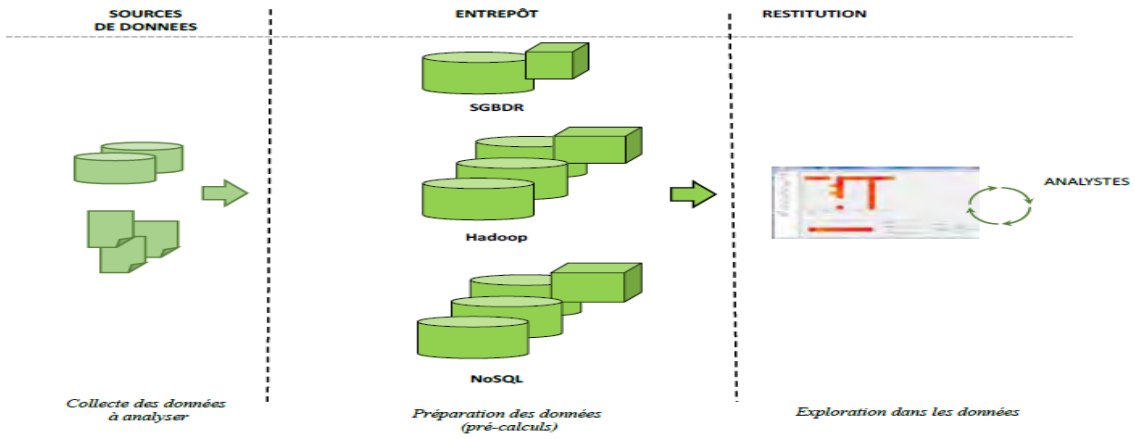

Figure 20 - Nouvelle architecture des systèmes d'aide à la décision intégrant le NoSQL

Nous étudions tout particulièrement les travaux visant à implanter les modèles multidimensionnels du niveau conceptuel à l'aide les modèles NoSQL [32] [33].Ils se résument en deux catégories :

La première catégorie consiste à traduire de manière indirecte les schémas multidimensionnels. Le processus réutilise les règles de traduction du conceptuel vers le R-OLAP [32] [34] [35] [36] puis est étendu par de nouvelles règles permettant un passage du modèle intermédiaire R-OLAP vers le modèle NoSQL cible.

La seconde catégorie consiste à traduire directement les schémas multidimensionnels du niveau conceptuel vers le modèle NoSQL cible.

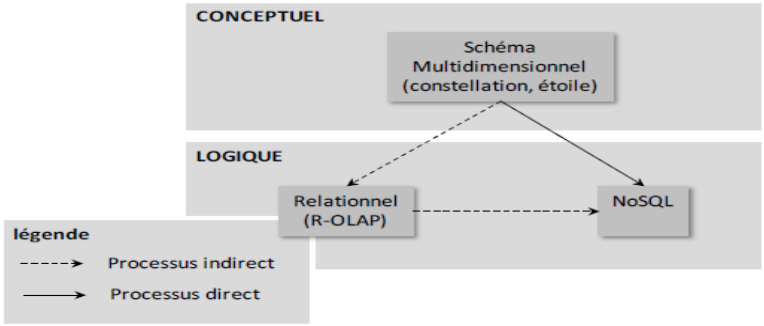

Figure 21 - Processus de transformation des entrepôts de données multidimensionnelles du niveau conceptuel vers le niveau logique [32]

2) Migration des entrepôts de données des systèmes relationnels vers les systèmes NoSQL

Dans la littérature, de nombreux chercheurs ont reconnu les limitations du modèle de données OLAP traditionnel et ont proposé des approches pour la migration des bases de données relationnelles vers des bases NoSQL [31]. Il s'agit de deux approches, l'approche indirecte et l'approche directe, qui sont les suivantes :

1. Approche Indirecte

1) Principe :

L'approche indirecte consiste à traduire les schémas multidimensionnels conceptuels en un modèle relationnel OLAP (R-OLAP) avant de les convertir en un modèle NoSQL. Ce processus en deux étapes implique d'abord l'application des règles de transformation du schéma conceptuel vers le schéma relationnel, puis l'application de nouvelles règles pour convertir le schéma relationnel en un modèle NoSQL.

– Cette approche traditionnelle consiste d'abord à convertir le modèle multidimensionnel (vue conceptuelle de l'entrepôt de données) en un modèle relationnel (niveau logique).

– Ensuite, le modèle relationnel est converti en modèle NoSQL (niveau logique).

– Cette approche est moins complexe mais peut ne pas exploiter pleinement les avantages des systèmes NoSQL.

2) Étapes du Processus de Traduction Indirecte :

1. Traduction Conceptuel vers R-OLAP :

- Utilisation des règles de transformation pour convertir les schémas conceptuels multidimensionnels en schémas relationnels OLAP.

- Ces règles ont été bien documentées et utilisées depuis longtemps dans les systèmes R-OLAP traditionnels.

- **Identification des Entités et Relations :**

- Analyse des modèles conceptuels multidimensionnels pour identifier les entités principales (dimensions et faits) et leurs relations.

- Exemple : Dans un modèle de ventes, les entités pourraient inclure les "Produits", "Clients", "Temps", et "Ventes".

- **Application des Règles de Transformation Conceptuel-R-OLAP :**

- Utilisation des règles documentées pour convertir les entités conceptuelles en tables relationnelles.

- Les dimensions deviennent des tables dimensionnelles avec des clés primaires, et les faits deviennent des tables de faits avec des clés étrangères référenciant les tables de dimensions.

- Exemple : La dimension "Produit" devient une table "Produit" avec des attributs comme "ProduitID", "NomProduit", "Catégorie", etc.

- **Définition des Clés et des Relations :**

- Définition des clés primaires pour chaque table dimensionnelle et des clés étrangères dans les tables de faits pour maintenir les relations.

- Exemple : La table "Ventes" contient des clés étrangères telles que "ProduitID", "ClientID", "TempsID" qui référencent les tables "Produit", "Client", "Temps" respectivement.

2. **Transformation R-OLAP vers NoSQL** :

- Application de nouvelles règles spécifiques pour convertir les tables relationnelles en structures NoSQL, telles que les modèles orientés colonnes ou orientés documents.

- Les modèles orientés colonnes restructurent le schéma relationnel en utilisant des familles de colonnes, par exemple, dans HBase.

- Les modèles orientés documents convertissent les relations en collections de documents dénormalisés, par exemple, dans MongoDB.

- **Choix du Modèle NoSQL Cible :**

- Décision sur le type de modèle NoSQL à utiliser, soit orienté colonnes, soit orienté documents.

- Exemple : HBase pour un modèle orienté colonnes, MongoDB pour un modèle orienté documents.

- **Application des Règles de Transformation Relationnel-NoSQL :**

- Utilisation des règles spécifiques pour convertir les tables relationnelles en structures NoSQL adaptées.

- Les tables relationnelles sont transformées en familles de colonnes ou collections de documents selon le modèle NoSQL choisi.

- **Restructuration des Données :**

- Dénormalisation des données pour réduire le besoin de jointures coûteuses dans un environnement distribué.

- Exemple : Dans HBase, chaque table relationnelle devient une famille de colonnes. Dans MongoDB, chaque relation est regroupée dans une collection et chaque tuple est traduit en un document.

3) **Concepts Clés :**

- **Dénormalisation :** Processus de conversion des données relationnelles normalisées en un format dénormalisé pour éviter les jointures dans les bases NoSQL.

- Familles de Colonnes : Utilisées dans les modèles orientés colonnes pour regrouper des ensembles de colonnes dans HBase.

- Collections de Documents : Utilisées dans les modèles orientés documents pour regrouper des ensembles de documents en MongoDB.

- Règles de Transformation : Ensembles de règles définissant comment convertir les schémas relationnels en structures NoSQL.

4) Exemples :

- [41] et [42] utilisent HBase pour restructurer les schémas relationnels en familles de colonnes [32].

- [43], [44] et d'autres proposent des méthodes pour convertir les schémas relationnels en documents MongoDB [32].

5) Synthèse :

Les approches indirectes visent à réutiliser les connaissances et les outils existants pour la conversion de modèles relationnels vers NoSQL. Bien que cette approche permette une transition progressive et une réutilisation des technologies existantes, elle peut parfois manquer de flexibilité et d'efficacité comparé à une approche directe.

2. Approche Directe

1) Principe :

Les modèles multidimensionnels sont un moyen de représenter les données pour les entrepôts de données, en se concentrant sur divers aspects (dimensions) tels que le temps, le produit et l'emplacement.

L'approche directe consiste à traduire les schémas multidimensionnels conceptuels directement en un modèle NoSQL sans passer par une phase intermédiaire de schéma relationnel. Cette approche cherche à tirer parti des caractéristiques uniques des bases de données NoSQL pour optimiser le stockage et les requêtes.

2) Étapes du Processus de Traduction Directe :

1. Identification des Entités et des Relations Conceptuelles :

- Analyse des schémas multidimensionnels conceptuels pour identifier les entités (dimensions et faits) et leurs relations.

- **Analyse du Schéma Conceptuel :**

- Analyse approfondie des schémas conceptuels multidimensionnels pour identifier les dimensions et les faits.

- Exemple : Identifier les dimensions "Produit", "Client", "Temps" et la table de faits "Ventes".

- **Définition des Attributs et des Hiérarchies :**

- Identification des attributs de chaque dimension et des hiérarchies éventuelles.

- Exemple : La dimension "Produit" pourrait avoir des hiérarchies comme "Catégorie > Sous-Catégorie > Produit".

2. **Application des Règles de Transformation NoSQL :**

- Application de règles spécifiques pour convertir directement les entités et les relations conceptuelles en structures NoSQL.

- Les données peuvent être structurées en utilisant des familles de colonnes ou des collections de documents, selon le type de base NoSQL cible.

- **Choix du Modèle NoSQL Cible :**

- Décision sur le type de modèle NoSQL à utiliser, soit orienté colonnes, soit orienté documents.

- Exemple : Utilisation de HBase pour un modèle orienté colonnes ou MongoDB pour un modèle orienté documents.

- **Application des Règles de Transformation Conceptuel-NoSQL** :

- Utilisation de règles spécifiques pour convertir les entités conceptuelles directement en structures NoSQL.

- Exemple : Définir des familles de colonnes ou des collections de documents pour chaque dimension et fait.

- **Restructuration et Optimisation des Données :**

- Structuration des données pour optimiser les requêtes et les performances dans le système NoSQL choisi.

- Exemple : Dans HBase, création de familles de colonnes pour chaque dimension fréquemment interrogée. Dans MongoDB, imbriquer les documents pour éviter les jointures

3) **Concepts Clés :**

- Cube de Données : Représentation multidimensionnelle des données permettant des opérations analytiques complexes.

- Indexation : Utilisation d'index pour optimiser les requêtes dans les bases de données NoSQL.

- Langage de Requête NoSQL : Utilisation de langages spécifiques comme HQL dans Hive pour interroger les données.

4) Exemples :

- [37] proposent des méthodes pour construire des cubes de données dans des modèles orientés colonnes [32].

- [33] et [38] travaillent sur l'implantation de cubes OLAP dans des modèles orientés colonnes comme Hbase [32].

- [39] proposent une approche R2NoSQL pour transformer un modèle en étoile directement dans MongoDB [32].

- [40] proposent des approches automatiques pour transformer des schémas multidimensionnels en modèles NoSQL orientés colonnes et documents [32].

5) Synthèse :

Les approches directes offrent une solution plus adaptée aux caractéristiques spécifiques des bases de données NoSQL, permettant des performances optimisées et une meilleure utilisation des capacités de ces systèmes. Cependant, elles peuvent nécessiter des connaissances et des outils spécialisés pour leur mise en œuvre.

3. Caractéristiques de l'approche indirecte et directe pour la modélisation multidimensionnelle dans les bases NoSQL

Tableau 2 Caractéristiques de l'approche indirecte et directe

Fonctionnalité	Approche indirecte	Approche directe
Processus de transformation	Modèle conceptuel -> R-OLAP -> NoSQL	Modèle conceptuel -> NoSQL
Représentation des données	Structurée (R-OLAP) -> NoSQL	Modèle conceptuel -> NoSQL
Complexité	Potentiellement plus complexe en raison de l'étape de transformation supplémentaire	Potentiellement moins complexe en raison du mappage direct

Redondance des données	Possibilité de redondance des données lors de la traduction NoSQL	La redondance des données doit être gérée attentivement pendant le mappage
Besoins d'expertise	Nécessite une compréhension à la fois de R-OLAP et de NoSQL	Nécessite une compréhension approfondie des modèles de données NoSQL et du stockage
Potentiel d'optimisation	Peut nécessiter des efforts supplémentaires pour optimiser le stockage des données dans NoSQL	Un mappage personnalisé peut conduire à un stockage et une récupération optimisés des données

Les deux approches, directe et indirecte, présentent des avantages et des inconvénients. L'approche indirecte permet une transition plus progressive et réutilise des technologies existantes, tandis que l'approche directe offre des performances optimisées en tirant pleinement parti des caractéristiques des bases de données NoSQL. Le choix de l'approche dépend des besoins spécifiques du projet et des ressources disponibles.

4. Type de la classification

Dans le modèle NoSQL oriente colonne, il existe deux classifications faites par les auteurs, le premier d'entre eux est pour Les auteurs *Rania Yangui, Ahlem Nabila, Faiez Gargouri,* [31] et cela est classé en deux les règle de transformation, transformation simple et Hiérarchique, et la deuxième classification est pour l'auteur *Mohamed Malki* [32] a Quatre Processus de traduction, Processus de traduction Plate, Imbrication, Hybride et Éclatée

1) Les règles de transformation

Les règles de transformation ont pour objectif de proposer des règles pour la conversion des schémas de Data Warehouse (DW) vers des modèles NoSQL. Un schéma de DW se compose de faits avec des mesures, ainsi que d'un ensemble de dimensions avec des attributs. Nous cartographions les dimensions selon leurs attributs et les faits selon leurs mesures. Étant donné qu'il existe plusieurs alternatives possibles, certaines de ces alternatives seront détaillées afin de choisir la meilleure. Deux types de transformations sont définis : les transformations simples et les transformations hiérarchiques. [31]

Règle 1 : Transformation Simple vers le Modèle Orienté Colonne (ST-C)

Les transformations simples consistent à stocker les faits et les dimensions dans des familles de colonnes ou des collections, en utilisant uniquement des colonnes ou des documents simples pour représenter les mesures et les attributs de dimension.

Pour chaque schéma multidimensionnel ($MS\ (MS\hat{}N, MS\hat{}\{Fact\}, MS\hat{}\{Dim\}, Func)$, la transformation vers un keyspace $KS\ (KS\hat{}N, KS\hat{}\{Val\})$ s'effectue de la manière suivante :

– Le nom du keyspace est le nom du $MS/KS^{N} \leftarrow MS\hat{}N$.

– Chaque fait ($F \in MS\hat{}\{Fact\}$) est transformé en une famille de colonnes ($CF\ (CF\hat{}N, CF\hat{}\{Val\})$) où :

- Le nom de la famille de colonnes est le nom du fait ($CF\hat{}N \leftarrow F\hat{}N$).

- Chaque mesure ($M \in F$) est transformée en une colonne ($C \in CF\hat{}\{Val\} / C\hat{}N \leftarrow M$).

- Chaque identifiant des dimensions liées est transformé en une colonne ($C \in CF\hat{}\{Val\} / C\hat{}N \leftarrow D\hat{}\{id\}$).

– Chaque dimension ($(D\hat{}N, D\hat{}\{Att\}, D\hat{}\{Hier\}) / D \in MS\hat{}\{Dim\}$) est transformée en une famille de colonnes ($CF\ (CF\hat{}N, CF\hat{}\{Val\})$) où :

- Le nom de la famille de colonnes ($CF\hat{}N$) équivaut au nom de la dimension ($CF\hat{}N \leftarrow D\hat{}N$).

- Chaque attribut ($A \in D\hat{}\{Att\}$) est transformé en une colonne ($C \in CF\hat{}\{Val\}$) où le nom de la colonne est l'attribut ($C\hat{}N \leftarrow A$).

Le processus global de transformation simple du schéma conceptuel DW au modèle orienté colonnes est présenté dans la (Figure 22).

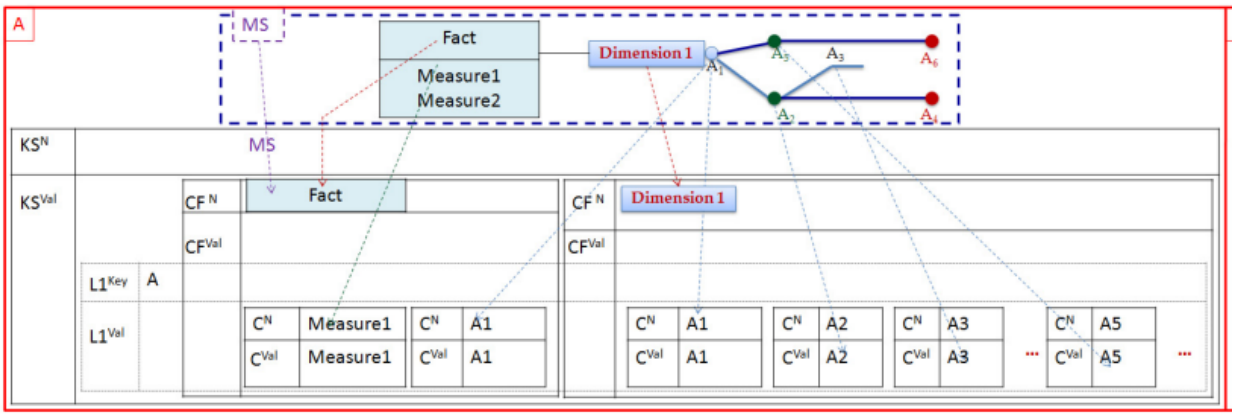

Figure 22 - transformation simple exemples

Règle 2 : Transformation Hiérarchique vers le Modèle Orienté Colonne (HT-C)

Les transformations hiérarchiques consistent également à utiliser différentes familles de colonnes ou collections pour stocker les faits et les dimensions, mais utilisent des colonnes super et des attributs composés pour représenter les attributs de dimension tout en expliquant les hiérarchies.

Pour chaque schéma multidimensionnel ($MS\ (MS\^N, MS\^\{Fact\}, MS\^\{Dim\}, Func)$), la transformation vers un keyspace ($KS\ (KS\^N, KS\^\{Val\})$) s'effectue de la manière suivante :

– Le nom du keys pace est le nom du MS \($KS\^N \leftarrow MS\^N$).

– Chaque fait ($F \in MS\^\{Fact\}$) est transformé en une famille de colonnes ($CF\ (CF\^N, CF\^\{Val\})$) où :

- Le nom de la famille de colonnes est le nom du fait ($CF\^N \leftarrow F\^N$).

- Chaque mesure ($M \in F$) est transformée en une colonne ($C \in CF\^\{Val\}$ / $C\^N \leftarrow M$).

- Chaque identifiant des dimensions liées est transformé en une colonne \($C \backslash in\ CF\^\{Val\}$ / $C\^N \leftarrow D\^\{id\}$).

– Chaque dimension ($(D\^N, D\^\{Atr\}, D\^\{Hier\})$ / $D \in MS\^\{Dim\}$) est transformée en une super-colonne-famille ($SCF\ (SCF\^N, SCF\^\{Val\})$) où :

- Le nom de la super-colonne-famille ($SCF\hat{}N$) équivaut au nom de la dimension ($D\hat{}N \leftarrow D\hat{}N$).

- Chaque hiérarchie ($H\ (H\hat{}N, H\hat{}P, PF\hat{}\{Att\})$) est transformée en une super-colonne ($SC\ (SC\hat{}N, SC\hat{}\{Val\})$) dans la ligne ($M\_i \in SCF\hat{}\{Val\}$) où :

- Le nom de la super-colonne est le nom de la hiérarchie ($SC\hat{}N \leftarrow H\hat{}N$).

- Chaque attribut (fort et faible) p_j devient une colonne dans ($SC\hat{}\{Val\}$).

La transformation hiérarchique du schéma multidimensionnel vers le modèle orienté colonnes est illustrée par (Figure 23).

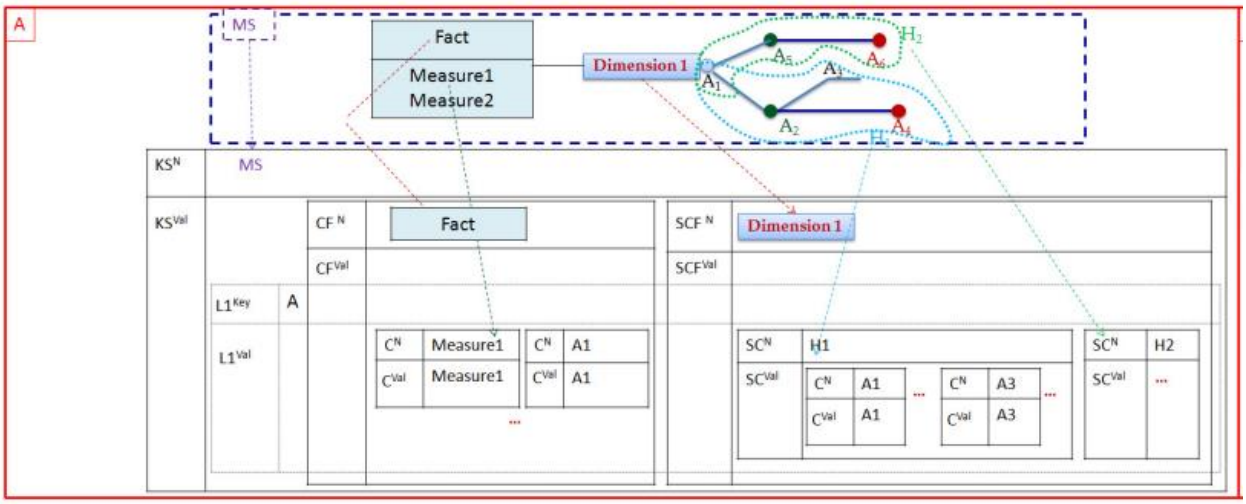

Figure 23 - transformation hiérarchique exemples.

Les transformations simples utilisent des colonnes et des documents simples pour représenter les mesures et les attributs, tandis que les transformations hiérarchiques utilisent des super-colonnes et des attributs composés pour représenter les hiérarchies au sein des dimensions. Ces transformations visent à optimiser le stockage et la représentation des schémas de DW dans des modèles NoSQL.

2) Modélisation Logique NOT-ONLY-SQL

Nos propositions portent sur modèles NoSQL orienté colonnes. Dans nous présentons le modèle orienté colonnes quatre les processus de traduction [32].

Modélisation multidimensionnelle orientée colonnes

Dans le modèle orienté colonnes, les données sont organisées en **tables** contenant un ensemble de lignes, chacune organisée en **colonnes**.

Nous adoptons le même formalisme que précédemment, basé sur deux constructeurs :

- [] pour représenter une structure formée d'un ou plusieurs colonnes.
- {} pour représenter un ensemble.

Définition : Une **table** *T* est définie par (N^T, E^T) où

- N^T est le nom de la table,
- $E^T = \{r_1, \dots, r_t\}$ est un ensemble de lignes.

La structure des lignes est définie par l'intermédiaire de **colonnes** (pouvant s'apparenter à des paires clé-valeur). Nous distinguons les **colonnes simples** dont les valeurs sont atomiques, des **colonnes composées** (appelées **familles de colonnes**) dont les valeurs sont elles-mêmes des regroupements de colonnes.

Définition : Une **ligne** r_i est définie par (S_i, V_i) où

- S_i est le schéma de la ligne, formé par un ensemble de colonnes,
- V_i est la valeur de la ligne.

Le schéma est inhérent à chaque ligne. Dans les systèmes NoSQL orientés colonnes, chaque ligne à son propre schéma, pouvant varier d'une ligne à l'autre, même au sein d'une même table.

Dans NoSQL orienté colonnes, il existe quatre Processus de traduction pour un entrepôt de données multidimensionnel.

- Parmi elles figurent des traductions plates, imbrication, hybride et éclatée.
- Nous avons utilisé ce processus de traduction plat car il est clair et facile à traduire.
- Nous expliquons les quatre processus de traduction avec un exemple.

- **Processus de traduction plate en orienté colonnes**

Chaque fait et ses dimensions associées correspondent à une table unique. Les données sont dénormalisées, et Chaque instance du fait est traduite par une ligne r_i. ce qui signifie que toutes les mesures du fait et les attributs des dimensions sont traduits

en colonnes simples au sein d'une seule famille de colonnes.

Une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

- *id* est l'identifiant de la ligne,
- chaque mesure de forme un attribut simple de la famille de colonnes,
- chaque attribut de forme un attribut simple.

Exemple : Considérons le schéma conceptuel de la Figure 30. Chaque instance du fait **Tweet** est traduite par une ligne. L'expression ci-dessous présente la table obtenue et détaille une ligne :

$$
\begin{aligned}
&\{\\
&\ldots,\ r_{i} = (\\
&\quad [id,\\
&Tweet\ [idUser, name, language, time\_c, time\_z,\\
&\qquad city, country, population, zone,\\
&\qquad day, month, month\_name, year,\\
&\qquad topic, category,\\
&\qquad Retweet\_NB]\\
&\quad],\\
&\qquad\quad [id: 12345,\\
&Tweet\ [\\
&\qquad idUser: "C02265", name: "\ Smith", language: "\ french", time\_c: "\ Paris", time\_z: "\ France",\\
&\qquad\quad city: "\ Paris", country: "\ France", population: 66000000, zone: "\ Europee\\
&\qquad\qquad\quad - Ouest",\\
&\qquad day: "03-31-2015", month: "03-2015", month\_name: "\ march", year: 2015,\\
&\qquad topic: "\ foot", category: "sport",\\
&\qquad Retweet\_NB: 200]],), \ldots\}
\end{aligned}
$$

La Figure 24 présente sous une forme graphique la ligne r_i. Elle est constituée d'une identifiant (*row key*) et d'une famille de colonnes.

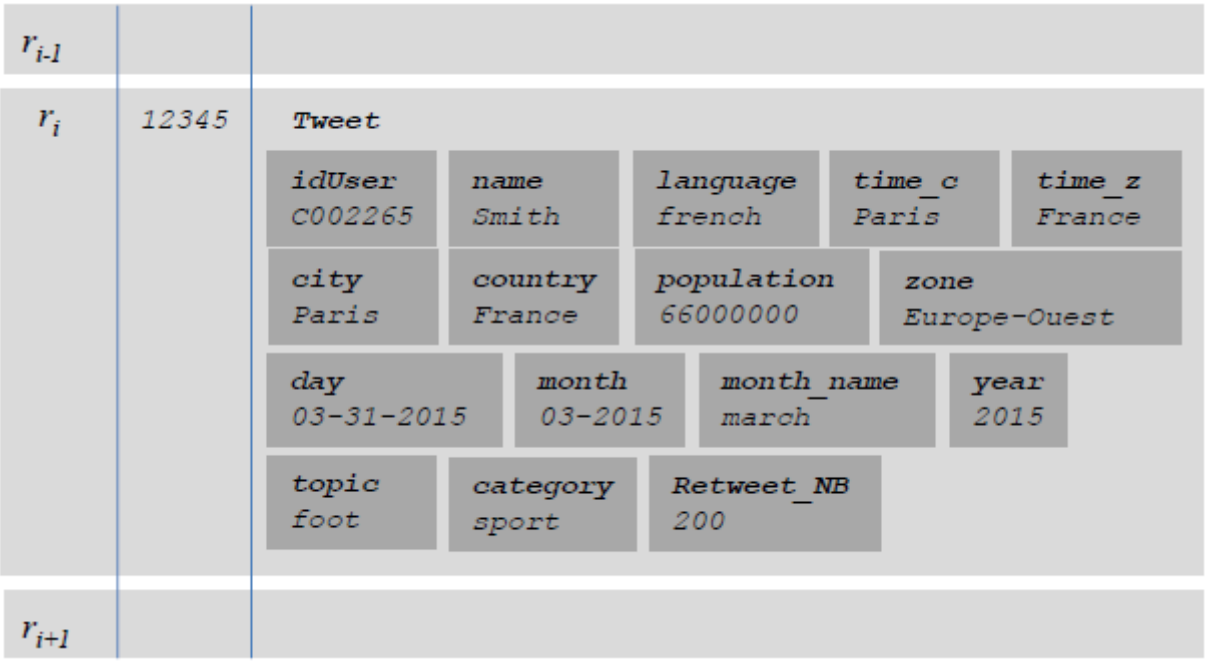

Figure 24 - Exemple de ligne par traduction plate

- **Processus de traduction par imbrication en orienté colonnes**

Comme précédemment, à chaque fait et ses dimensions associées correspond une table de même nom **(N^F, E^F).** Chaque instance du fait est traduite par une ligne r_i. Ce processus distribue les colonnes issues des mesures, et les colonnes issues de chaque dimension dans différentes familles de colonnes.

Une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

- ***id*** est l'identifiant de la ligne
- une famille de colonnes est définie pour le fait, et chaque mesure de forme une colonne imbriquée dans la famille
- une famille de colonnes est définie pour chaque dimension, chaque attribut de la dimension forme une colonne imbriquée dans la famille.

Exemple : Considérons la ligne décrite dans l'exemple précédent, que nous restructurons ci-dessous en fonction de l'approche par imbrication :

$\{ \ \ldots, ri = ($

$[id,$

$User[idUser, name, language, time_c, time_z],$

$Location[city, country, population, zone],$

$Time[day, month, month_{name}, year],$

$Subject[topic, category]$,

$Tweet[Retweet\_NB]]$,

$[id: 12345$,

$User: [idUser: "C02265", name: " Smith", language: " french", time\_c: " Paris", time\_z: " France"]$,

$Location: [city: " Paris", country: " France", population: 66000000, zone: " Europee - Ouest"]$,

$Time: [day: "3 - 31 - 2015", month: " 3 - 2015", month\_name: " march", year: 2015]$,

$Subject: [topic: " foot", category: " sport"]$,

$Tweet: [Retweet\_NB: 200]],), \ldots$

La Figure 25 présente sous une forme graphique la ligne r_i constituée par un ensemble de familles de colonnes issues du fait et de chaque dimension associée.

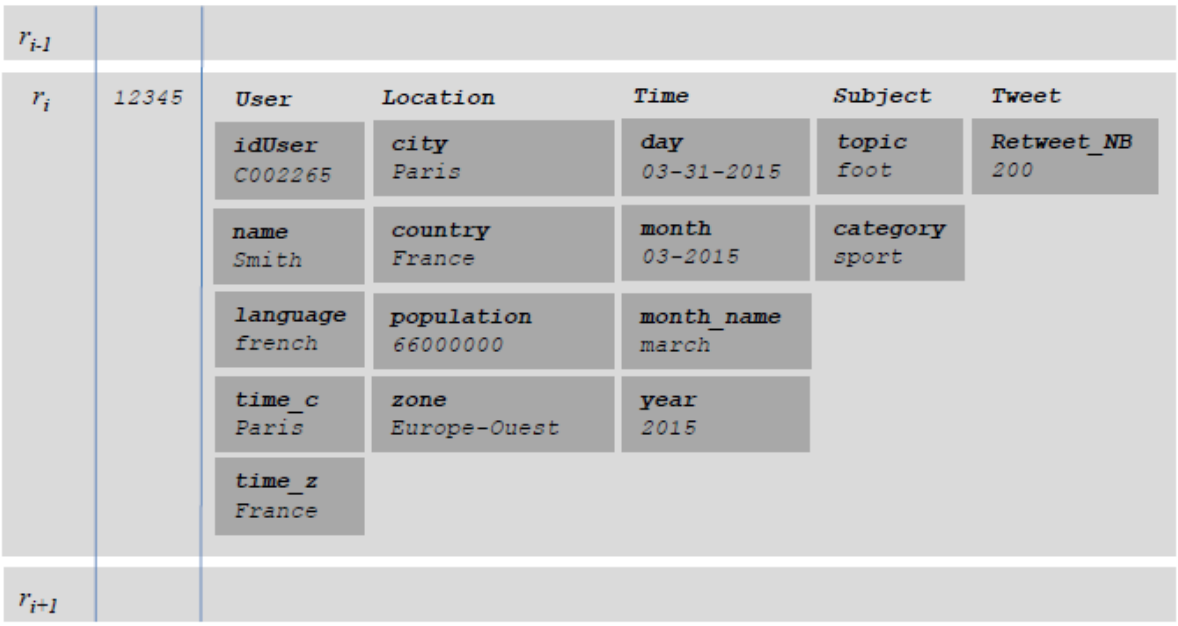

Figure 25 - Exemple de ligne par traduction imbriquée

- **Processus de traduction hybride en orienté colonnes**

L'approche hybride consiste à décomposer au sein de la table les données issues

d'un fait et de chaque dimension associée.

Les attributs des dimensions sont convertis en colonnes, sous une même famille de colonnes, et stockés comme une ligne de la table, à raison d'une ligne par dimension. Les mesures du fait sont converties en colonnes, sous une même famille de colonnes, et stockés sous forme de lignes de la même table. Les lignes des faits contiennent également les références aux lignes représentant les dimensions associées.

Pour chaque instance de dimension, une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

- *id* est l'identifiant de ligne, correspondant à la racine de la dimension.
- chaque attribut de la dimension forme une colonne imbriquée dans une unique famille de colonnes de même nom que la dimension.

Pour chaque instance du fait, une ligne $\boldsymbol{r_i}$ est définie par un schéma $\boldsymbol{S_i}$ constitué de la manière suivante :

- ***id*** est l'identifiant de ligne,
- chaque mesure de forme une colonne imbriquée dans une unique famille de colonnes de même nom que le fait.
- pour chaque dimension associée, une colonne est ajoutée référençant une ligne liée issue d'une dimension.

Exemple : Considérons l'exemple de document précédent. Dans l'approche hybride, une ligne est constituée pour chaque instance des dimensions ainsi qu'une autre pour l'instance du fait. Nous obtenons les lignes définies ci-dessous :

{..., r_i^{Tweet}=($[id, idUser, city, day, topic, Retweet\_NB]$,

$[id: 12345,$
$idUser: "C02265", city: "Paris", day: "03-31-$
$2015", topic: "foot", Retweet\_NB: 200],),$

$r_i^{User} = ([idUser, name, language, time_c, time_z],$
$[idUser: "C02265", name: "Smith", language: "french", time\_c: "Paris", time\_z: "France"],),$

$r_i^{Location} = ([city, country, population, zone],$
$[city: "Paris", country: "France", population: 66000000, zone: "Europe - Ouest"],),$

r_i^{Time}=($[day, month, month\_name, year]$,

$[day: "03-31-2015", month: "03-2015", month\_name: "march",\ year: 2015],$),

$r_i^{Subject}$=($[topic, category]$,

$[topic: "foot", category: "sport"],$),… }

La Figure 28 présente sous une forme graphique les lignes $r_i^{Tweet}, d_i^{user}, r_i^{Location}, r_i^{Subject}, r_i^{Subject}$, constituées par une famille de colonnes issues d'attributs respectivement du fait et de chaque dimension.

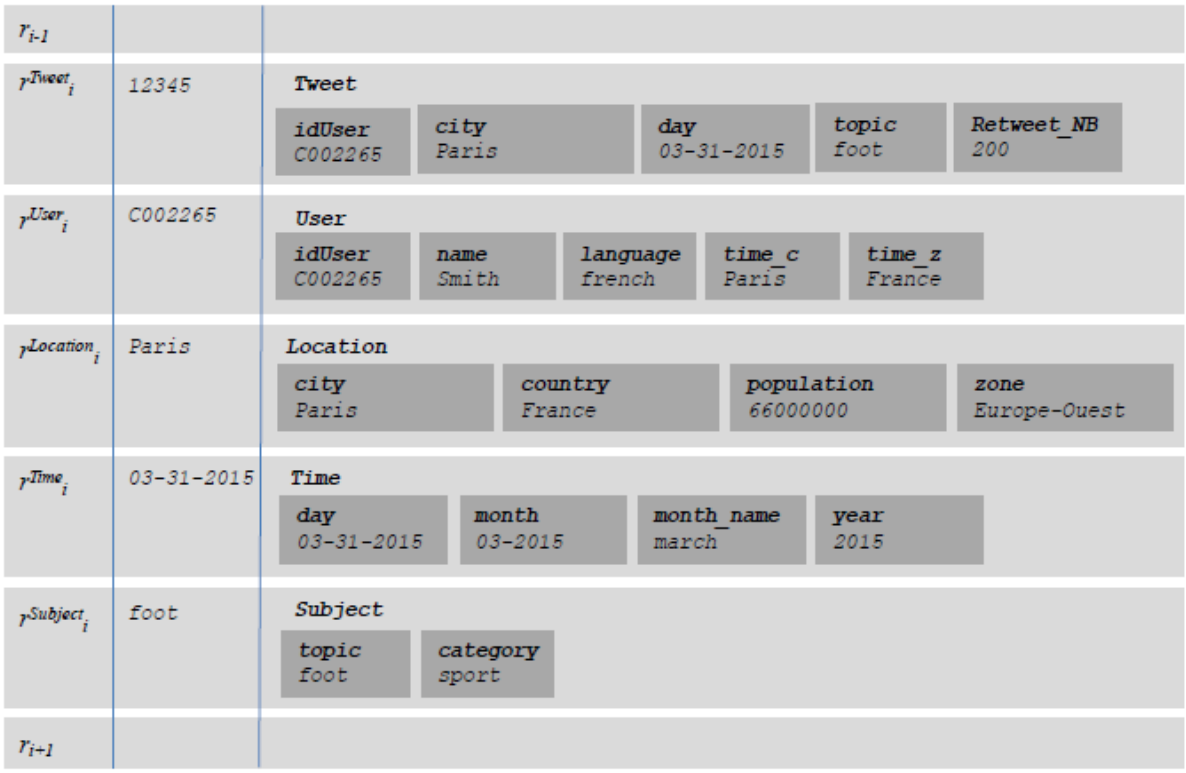

Figure 26 - Exemple de ligne par traduction hybride

- **Processus de traduction éclatée en orienté colonnes**

La traduction éclatée distribue dans plusieurs tables les données : les données issues d'un fait sont placées dans une première table, et les données de chaque dimension associée sont placées dans des tables distinctes (une table par dimension).

Les attributs des dimensions sont convertis en colonnes, sous une même famille de colonnes, et stockés dans une ligne de la table dédiée. Les mesures du fait sont converties en colonnes, sous une même famille de colonnes, et stockés dans une ligne d'une autre table.

Les lignes des faits contiennent également les références aux lignes représentant les dimensions associées, stockées dans les tables des dimensions.

Pour chaque instance de dimension, une ligne est définie par un schéma S_i constitué de la manière suivante.

- *id* est l'identifiant de la ligne, correspondant à la racine de la dimension,
- chaque attribut de la dimension forme une colonne imbriquée dans une unique famille de colonnes de même nom que la dimension.

Pour chaque instance du fait, une ligne est définie par un schéma S_i constitué de la manière suivante :

- *id* est l'identifiant de la ligne,
- chaque mesure de forme une colonne imbriquée dans une unique famille de colonnes de même nom que le fait,
- pour chaque dimension associée, une colonne est ajoutée référençant une ligne liée issue de la table de la dimension.

Exemple : Considérons l'exemple de ligne précédent. Dans l'approche éclatée, une ligne est constituée pour chaque instance des dimensions ainsi qu'une autre ligne pour l'instance du fait. Ces lignes sont placées dans des tables distinctes. Nous obtenons les lignes définies ci-dessous :

{… ,

r_i^{Tweet}=($[id, idUser, city, day, topic, Retweet\_NB]$,

$[id: 12345,$

$idUser: "C02265", city: "Paris", day: "03-31-2015", topic: "foot", Retweet\_NB: 200],), \ldots\}$

{ … ,

$r_i^{User} = (\,[idUser, name, language, time_c, time_z]$,

$[idUser: "C02265", name: "Smith", language: "french", time\_c: "Paris", time\_z: "France"],), \ldots\}$

{ … ,

$r_i^{Location} = (\,[city, country, population, zone]$,

$[city: "Paris", country: "France", population: 66000000, zone: "Europe - Ouest"],), \ldots\}$

$\{ \dots ,$

$r_i^{Time} = (\ [day, month, month\_name, year],$

$[day: "03-31-2015", month: "03-2015", month\_name: "march", \ year: 2015],), \dots \}$

$\{ \dots ,$

$r_i^{Subject} = (\ [topic, category], [topic: "foot", category: "sport"],), \dots \}$

La Figure 29 présente sous une forme graphique les lignes

$r_i^{Tweet}, d_i^{user}, r_i^{Location}, r_i^{Subject}, r_i^{Subject}$,appartenant respectivement aux tables T^{Tweet}, T^{User}, $T^{Location}$, T^{Time} $T^{SubSect}$ dont la construction est homogène pour toutes ses lignes. Les lignes sont constituées par une famille colonnes issues respectivement du fait et de chaque dimension.

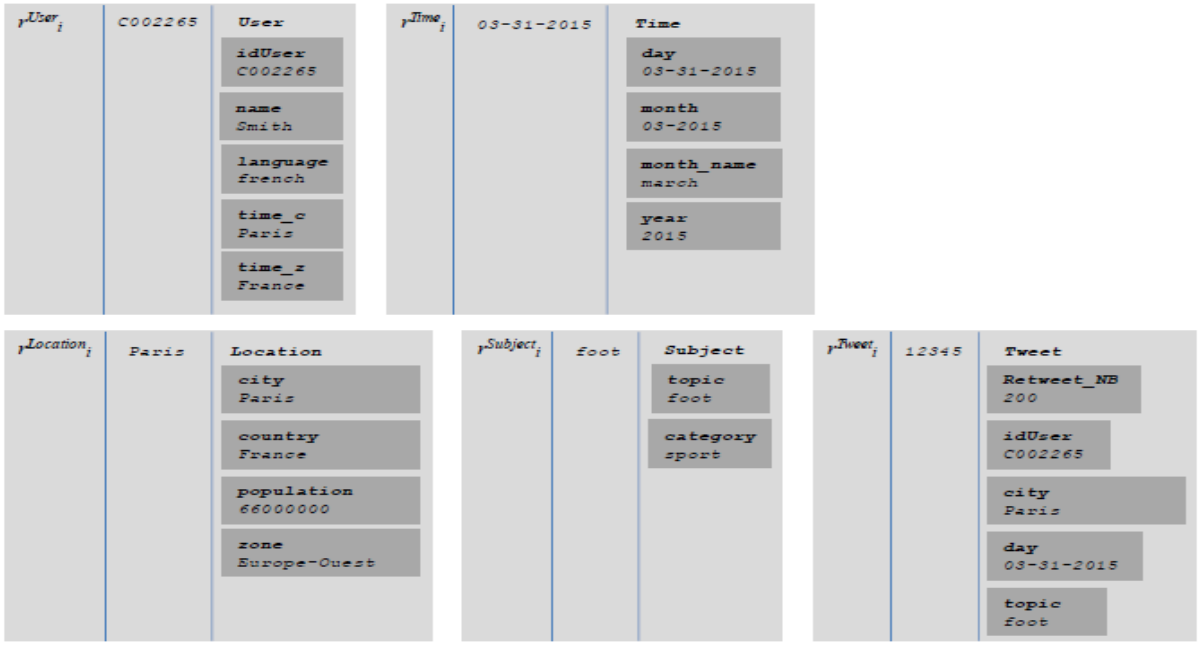

Figure 27 - Exemple de ligne par traduction éclatée

3) Exemple de Modélisation Conceptuelle pour les Données Multidimensionnelles

- **Modèle Conceptuel Multidimensionnel** :

➢ **Schéma en Constellation** :

Noté S, est défini par (F^S, D^S , $Star^S$) où :

- $F^S = \{F^1, \dots, F^n\}$: Ensemble de faits.
- $D^S = \{D^1, \dots, D^m\}$: Ensemble de dimensions.
- $Star^S : F^S \rightarrow 2\text{^}D^S$: Associe des faits aux dimensions.

  - **Faits** :
    - N^F: Nom du fait.
    - M^F: Ensemble de mesures avec des fonctions d'agrégation (SUM, MAX, MIN, COUNT, AVG).
  - **Dimensions** :
    - N^D: Nom de la dimension.
    - A^D: Ensemble d'attributs.
    - H^D: Ensemble de hiérarchies.
  - **Hiérarchies** :
    - N^{Hj}: Nom de la hiérarchie.
    - $Param^{Hj}$: Ensemble ordonné d'attributs formant la hiérarchie.
    - $WeaK^{Hj}$: Fonction associant des paramètres à des attributs faibles.

➢ **Exemple de Schéma conceptuel en étoile**

Un schéma en étoile est un cas particulier de schéma en constellation où il n'y a qu'un seul fait. Voici un exemple d'analyse OLAP portant sur les tweets, représentant la popularité des tweets en fonction du nombre de retweets.

☐ Fait:

- Représente la popularité des tweets mesurée par les retweets.

☐ Dimensions:

- D_{User}: Attributs relatifs à l'auteur du tweet.
- $D_{Location}$: Attributs relatifs à la localisation du tweet.
- $D_{Subject}$: Attributs relatifs au sujet du tweet.
- D_{Time}: Attributs relatifs au moment du tweet.

Le fait est ainsi défini par $F_{Tweet} = (N^{F_{Tweet}}, count(M^{F_{Tweet}}))$.

Le fait est associé à quatre dimensions :

$$Star^S(F_{Tweet}) = \{D_{User} \ ,, D_{Location} ,, D_{Subject} ,, D_{Time}\}.$$

La dimension D_{User}décrit l'auteur du tweet. Elle comporte six attributs

$A^{User} = \{id^{User}, name\_u, time\_c, language, time\_z, all^{User}\}$ Organisés selon trois hiérarchies :

- $(H_{TC}, \langle id^{User}, time\_c, all^{User}\rangle, \{(id^{User}, \{name\_u\})\})$
- $(H_{TM}, \langle id^{User}, language, all^{User}\rangle, \{(id^{User}, \{name\_u\})\})$
- $(H_{TZ}, \langle id^{User}, time\_z, all^{User}\rangle, \{(id^{User}, \{name\_u\})\})$

La dimension $D_{Location}$ décrit le lieu d'où est émis le tweet. Elle comporte six attributs $A^{Location} = \{city, country, population, continent, zone, all^{Location}\}$ organisés selon deux hiérarchies :

- $(H_{cont}, \langle city, country, continent, all^{Location}\rangle, \{(country, \{population\})\})$
- $(H_{Zn}, \langle city, country, zone, all^{Location}\rangle, \{(country, \{population\})\})$

La dimension $D_{Subject}$ décrit le sujet du tweet (classiquement, le sujet est déterminé en fonction de la fréquence d'apparition de termes dans le tweet). Elle comporte trois attributs $A^{Subject} = \{topic, category, all^{Subject}\}$ organisés selon une hiérarchie :

- $(H_{Top}, \langle topic, category, all^{SubSect}\rangle, \{\})$

La dimension D_{Time} décrit la date à laquelle le tweet est émis. Elle comporte cinq attributs $A^{Time} = \{day, month, month\_name, year, all^{Time}\}$ organisés selon une hiérarchie :

- $(H_{Time}, \langle day, month, year, all^{Time}\rangle, \{(month, \{month\_name\})\})$

Ce Figure 25 montre les relations entre le fait central (tweets) et les dimensions associées, ainsi que les hiérarchies dans chaque dimension et les paramètres racines des dimensions .Notons que les attributs racines des dimensions $D_{Location}, D_{Subject}$ et D_{Time} correspondent respectivement aux paramètres *city*, *topic* et *day* (au lieu de id^D).

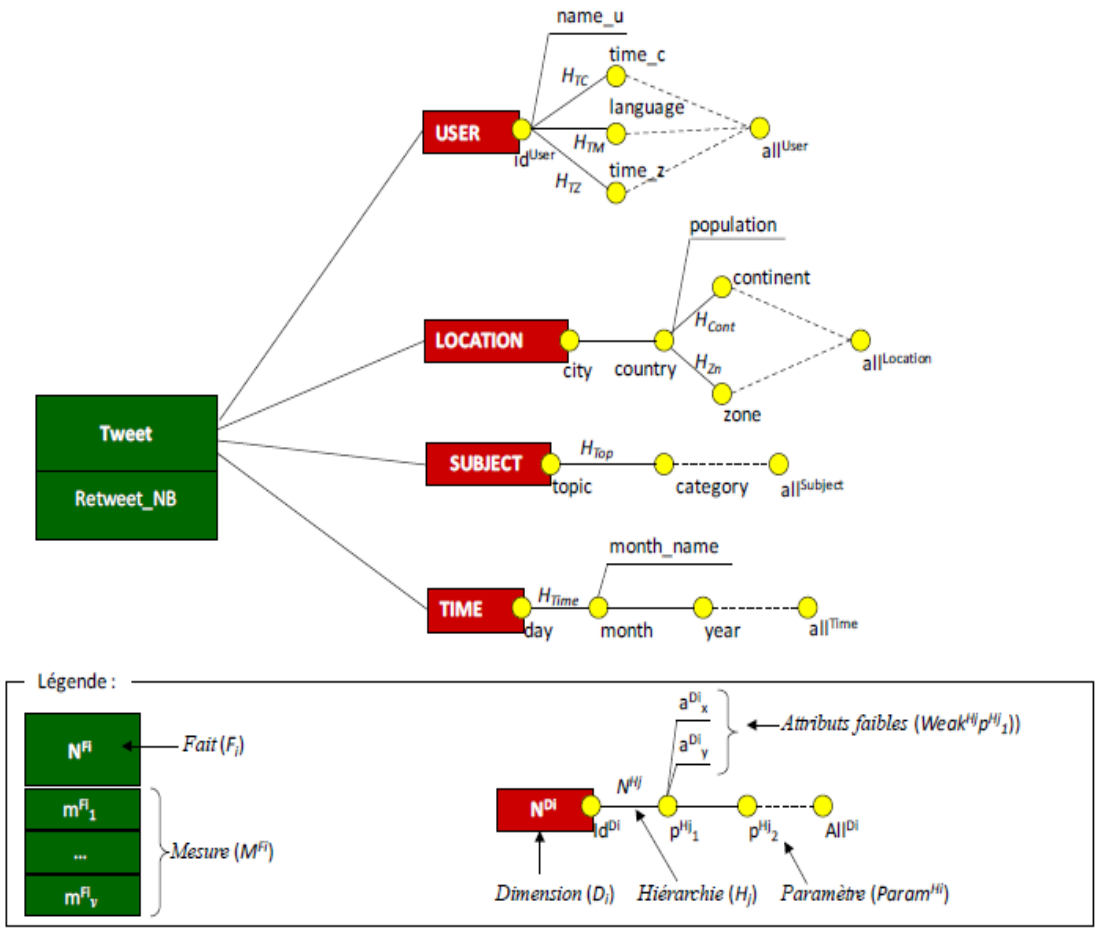

Figure 28 - Exemple de schéma conceptuel en étoile [32]

4) Choisir modèle conceptuel et base de donne Nosql :

Dans le Tableau 3, nous synthétisons les processus de transformation directe. Nous considérons les caractéristiques suivantes :
– Le schéma conceptuel multidimensionnel. Il s'agit de vérifier si le schéma multidimensionnel a été utilisé.

– Le type de schéma 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.

– Le système NoSQL utilisé (Cassandra, HBase, MongoDB...) pour implanter l'entrepôt de données.

– Le type de stockage utilisé pour implanter l'entrepôt de données, il s'agit du schéma normalisé ou dénormalisé.

Tableau 3-Comparatif des travaux de transformation directe des schémas conceptuels en NoSQL [32].

	Conceptuel		Logique		
	Multi-dimensionnel	**Type de schéma**	**Catégorie**	**Système NoSQL**	**Type**
[33]	Oui	Constellation	Colonnes	HBase	Dénormalisé
[37]	Oui	Etoile	Colonnes	HBase	Dénormalisé
[38]	Oui	Etoile	Colonnes	HBase	Dénormalisé
[39]	Oui	Etoile	Colonnes, Documents	MongoDB	Dénormalisé
[48]	Oui	-	Graphes	Neo4J	-
[49]	Non	-	Colonnes	HBase	Dénormalisé
[40]	Oui	Etoile	Colonnes, Documents	Cassandra HBase	Dénormalisé
[50] [51] [52] [53] [54] [55] [56] [57] [58]	Oui	Constellation	Colonnes, Documents	MongoDB HBase	Dénormalisé Normalisé

2.3.2. Exemples réels d'implémentation d'EDD Nosql : base de donnée NoSQL 2024

Le classement DB-Engines classe les systèmes de gestion de bases de données en fonction de leur popularité. Le classement est mis à jour mensuellement.

Rank May 2024	Rank Apr 2024	Rank May 2023	DBMS	Database Model	Score May 2024	Score Apr 2024	Score May 2023
1.	1.	1.	Oracle	Relational, Multi-model	1236.29	+2.02	+3.66
2.	2.	2.	MySQL	Relational, Multi-model	1083.74	-3.99	-88.72
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	824.29	-5.50	-95.80
4.	4.	4.	PostgreSQL	Relational, Multi-model	645.54	+0.49	+27.64
5.	5.	5.	MongoDB	Document, Multi-model	421.65	-2.31	-14.96
6.	6.	6.	Redis	Key-value, Multi-model	157.80	+1.36	-10.33
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	135.35	+0.57	-6.28
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	128.46	+0.97	-14.56
9.	9.	↑ 11.	Snowflake	Relational	121.33	-1.87	+9.61
10.	10.	↓ 9.	SQLite	Relational	114.32	-1.69	-19.54
11.	11.	↓ 10.	Microsoft Access	Relational	104.92	-0.49	-26.26
12.	12.	12.	Cassandra	Wide column, Multi-model	101.89	-1.97	-9.25

Figure 29 - Classification des SGBD selon leur popularité [9]

Exemples concrets d'implémentations NoSQL EDD pour chaque base de données :

1) MongoDB :

Facebook : Facebook utilise MongoDB pour stocker ses vastes données de graphiques sociaux, qui incluent les profils d'utilisateurs, les connexions et les interactions. La flexibilité et l'évolutivité de MongoDB permettent à Facebook de gérer une quantité massive de données et d'offrir une expérience de réseautage social fluide.

PayPal : PayPal utilise MongoDB pour gérer les données de transaction de ses utilisateurs et les informations de paiement. La structure orientée document de MongoDB permet à PayPal de stocker et d'analyser les divers formats de données associés aux transactions financières.

2) Apache Cassandre :

Netflix : Netflix utilise Cassandra pour gérer ses préférences utilisateur et ses données de recommandation. Les capacités d'évolutivité et de distribution de Cassandra permettent à Netflix de gérer l'énorme quantité de données utilisateur et de fournir efficacement des recommandations personnalisées.

eBay : eBay emploie Cassandra pour stocker et gérer les données de son catalogue de produits. La tolérance aux pannes et la haute disponibilité de Cassandra garantissent que les informations sur les produits eBay restent accessibles aux utilisateurs même pendant les périodes de trafic élevé.

3) Apache HBase :

Twitter : Twitter utilise HBase pour gérer ses données de tweet en temps réel. Le stockage de données orienté colonnes et le débit d'écriture élevé de HBase permettent à Twitter d'ingérer et d'analyser efficacement l'énorme volume de données de tweet pour les sujets d'actualité, les recommandations des utilisateurs et l'analyse des sentiments.

Yahoo : Yahoo utilise HBase pour stocker et analyser ses données de parcours, qui incluent les interactions des utilisateurs avec ses sites Web et son moteur de recherche. L'évolutivité et les performances de HBase permettent à Yahoo d'obtenir des informations sur le comportement des utilisateurs et d'améliorer ses produits et services.

4) Neo4j :

Walmart : Walmart utilise Neo4j pour analyser ses données de ventes de produits et identifier les modèles d'achat des clients. La capacité de Neo4j à interroger efficacement les relations entre les points de données permet à Walmart d'obtenir des informations sur les préférences des clients et d'optimiser le placement des produits et les promotions.

Microsoft : Microsoft utilise Neo4j pour gérer sa base de connaissances pour son système de support client. La capacité de Neo4j à gérer des relations complexes entre les entités de données permet à Microsoft de fournir une assistance plus rapide et plus précise à ses clients.

5) Apache CouchDB :

BBC : La BBC utilise CouchDB pour stocker et gérer ses données de programmes de radio et de télévision. Les capacités de réplication de CouchDB garantissent que les données des programmes de la BBC restent cohérentes sur ses différents systèmes et accessibles aux utilisateurs même dans des endroits éloignés.

TravelPerk : TravelPerk utilise CouchDB pour gérer ses données de réservation de voyages et fournir un accès hors ligne à son application mobile. La fonctionnalité hors ligne de CouchDB permet aux utilisateurs de TravelPerk d'accéder à leurs informations de réservation même lorsqu'ils ne sont pas connectés à Internet.

Ces solutions offrent une variété d'approches pour répondre aux besoins spécifiques en termes de traitement et de stockage des données dans des environnements Big Data.

2.4. Conclusion

L'objectif de NoSQL dans ce chapitre est de trouver des solutions aux limitations des bases de données relationnelles et de surmonter les défis posés par les Big Data. Parmi ces solutions, la transition du modèle conceptuel au modèle logique NoSQL est essentielle. Cette transition peut se faire via deux approches : une approche indirecte et une approche directe.

Ensuite, nous passons au chapitre suivant pour la modélisation Dataset et leur analyse. Nous utiliserons la modélisation logique NoSQL en choisissant une méthode appropriée pour la modélisation des données. Cela permettra de structurer efficacement les données et d'améliorer les performances des requêtes et des analyses.

Chapitre 3

Modélisation conceptuelle multidimensionnelle

3.1. Introduction

Dans ce chapitre, nous avons travaillé sur le dataset "Retail Sales Data" du site kaggle.com. Nous avons effectué la modélisation et l'analyse de ce dataset, en commençant par la modélisation conceptuelle. Ensuite, nous sommes passés à la modélisation logique NoSQL, et choisi la *processus de traduction plate*, car elle est la plus simple et plus appropriée pour notre sujet. Notre attention s'est concentrée sur les modèles NoSQL orientés colonnes, spécifiques à notre étude.

3.2. La dataset et le modèle conceptuel

En modélisation nous pouvons utiliser n'importe quel ensemble de données (Dataset) sur notre thèse, le modéliser sur modèle étoile et l'implémenter sur *processus de traduction plate* et nous avons pris les données du site kaggel sur les ventes (Sales) .

3.2.1. Contenu

Ces données sont collectées auprès d'une entreprise de vente (Sales) au détail turque. La période s'étend de 2017 à fin 2019 [45][1] et la taille du fichier csv 1.03 GB.

3.2.2. Analyse du Dataset

Le tableau 4 montre d'analyse du Dataset avec des colonnes et leurs descriptions correspondantes :

Tableau 4 - Analyse le Dataset

[1] Retail Sales Data (kaggle.com)

colonnes	Description
Product id	Identifiant unique du produit
store id	Identifiant unique du magasin
date	Date de la transaction ou de l'enregistrement
sales	Quantité de produits vendus
revenue	Revenu généré par les ventes
stock	Quantité de stock disponible
Price	Prix du produit
promo_type_1	Type de promotion 1
promo_bin_1	Information complémentaire sur la promotion 1
promo_type_2	Type de promotion 2
promo_bin_2	Information complémentaire sur la promotion 2
promo_discount_2	Montant de la réduction pour la promotion 2
promo_discount_type_2	Type de réduction pour la promotion 2

3.2.3. Identification des Entités

- **Entités Principales**

– **Dimension**

- **Produit (Product)**

  - `Produit_id`

  - `Price` (peut varier par produit, même si le prix est constant dans cet exemple)

- **Magasin (Store)**

  - `Store_id`

- **Date (Date)**

  - `date`

- **Promotion (Promotion)**

  - `promo_type_1`, `promo_bin_1`

  - `promo_type_2`, `promo_bin_2`, `promo_discount_2`, `promo_discount_type_2`

– **Fait**

- **Ventes (Sales)**

- `Produit_id`, `Store_id`, `date`

- `sales`, `revenue`, `stock`

- `promo_type_1`, `promo_bin_1`

- `promo_type_2`, `promo_bin_2`, `promo_discount_2`, `promo_discount_type_2`

3.2.4. Modèle Conceptuel

Le modèle conceptuel identifie les entités et leurs relations. Voici comment elles se regroupent :

Tableau 5 - Produit

Attribut	Type	Description
Produit_id	VARCHAR	Identifiant produit
Price	FLOAT	Prix du produit

Tableau 6 - Magasin

Attribut	Type	Description
Store_id	VARCHAR	Identifiant magasin

Tableau 7 - Date

Attribut	Type	Description
date	DATE	Date de la transaction
Year	INT	Année
Month	INT	Mois
Day	INT	Jour
Week	INT	Trimestre

Tableau 8 - Promotion

Attribut	Type	Description
promo_type 1	VARCHAR	Type de promotion
promo_type 2	VARCHAR	Type de promotion
promo_bin 1	VARCHAR	Information complémentaire promotion
promo_bin 2	VARCHAR	Information complémentaire promotion
promo_discount 2	FLOAT	Montant de la réduction

promo_discount_type 2	VARCHAR	Type de réduction

Tableau 9 - Ventes (Sales)

Attribut	Type	Description
Produit_id	VARCHAR	Identifiant du produit
Store_id	VARCHAR	Identifiant du magasin
date	DATE	Date de la vente
sales	FLOAT	Quantité vendue
revenue	FLOAT	Revenu généré
stock	FLOAT	Stock disponible
Price	FLOAT	Prix du produit
promo_type_1	VARCHAR	Type de promotion 1
promo_bin_1	VARCHAR	Information complémentaire sur promo 1
promo_type_2	VARCHAR	Type de promotion 2
promo_bin_2	VARCHAR	Information complémentaire sur promo 2
promo_discount_2	FLOAT	Montant de la réduction pour promo 2
promo_discount_type_2	VARCHAR	Type de réduction pour promo 2

Choisir le modèle étoile :

Le modèle étoile est utilisé dans la modélisation logique Not-only-SQL car il fournit une structure simple et performante pour gérer et analyser de grandes quantités de données, tout en étant bien adapté aux caractéristiques des systèmes NoSQL.

La figure 30 montre un modèle de données en étoile pour l'analyse des ventes, avec une table des faits centrale et plusieurs tables de dimensions.

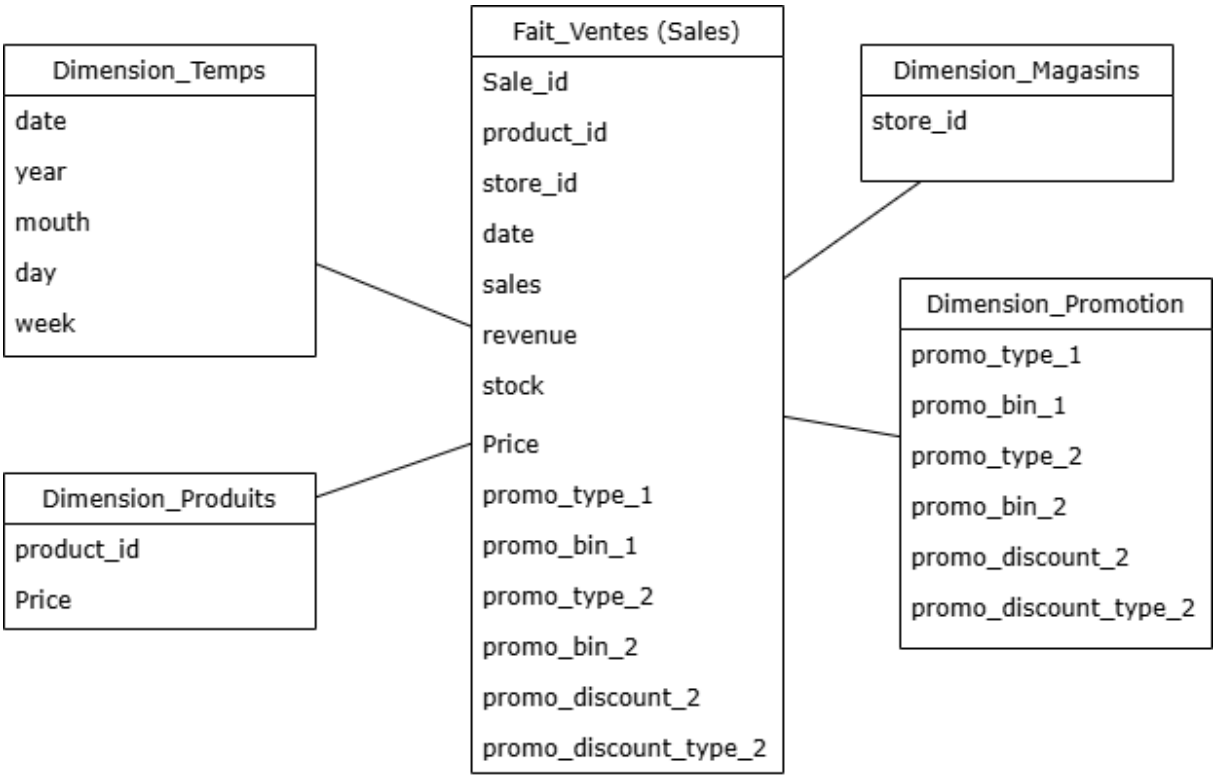

Figure 30- Modèle en Etoile

3.3. Modélisation de processus de traduction plats pour un ensemble de données de vente Dataset sale.csv :

Cela se réfère à un *processus de traduction plate*, qui implique de convertir un modèle multidimensionnel complexe en une structure plus simple et plate. Ce processus est crucial dans les contextes où la simplicité et l'efficacité sont prioritaires par rapport à la complexité du modèle multidimensionnel original.

Le processus de traduction plate est probablement utilisé pour simplifier la représentation des structures de données complexes en une forme plus facile à gérer et à interroger dans un environnement NoSQL. Cela implique de mapper les données multidimensionnelles dans un format qui peut être stocké et accédé efficacement dans un système de base de données Not-Only-SQL.

La Figure 31 et tableau 10 présente sous une forme graphique la ligne *ventes* Elle est constituée d'une identifiant (*id*) et d'une famille de colonnes.

- **Modélise modèle en Etoile vente au schéma plate**

Nous avons fait notre propre modélisation sur Dataset de Ventes (Sales) voir tableau 10 comme dans la figure 24 précédente .

Tableau 10 : schéma plate des Ventes

r_{i-1}		
r_i	1343	Vente Store_id 1234 | Produit_id P0001 | Promo_id PR0001 | Date 12-03-2020 | Prix 12 | Promo_type pr14 Promo_bin 1 | Promo_discount | promo_discount_type | Year 2012 | Month 12 Day 02 | Quarter 2 | Sales 0.0 | Revenue 0.0 | Stock 8,0
r_{i+1}		

– **Affichage la implémentation du schéma plate une interface migration**

Pour extraire des données dans ce format et y appliquer le *processus de traduction plate*, nous avons utilisé le langage Python sur PyCharm (Chapitre 4 pour utilise). Nous avons créé une méthode algorithmique qui nous permet d'appliquer le *processus de traduction plate* à n'importe quelles données.

Voire figure 31 affiche id est ligne, affiche attribue et mesure sur colonne

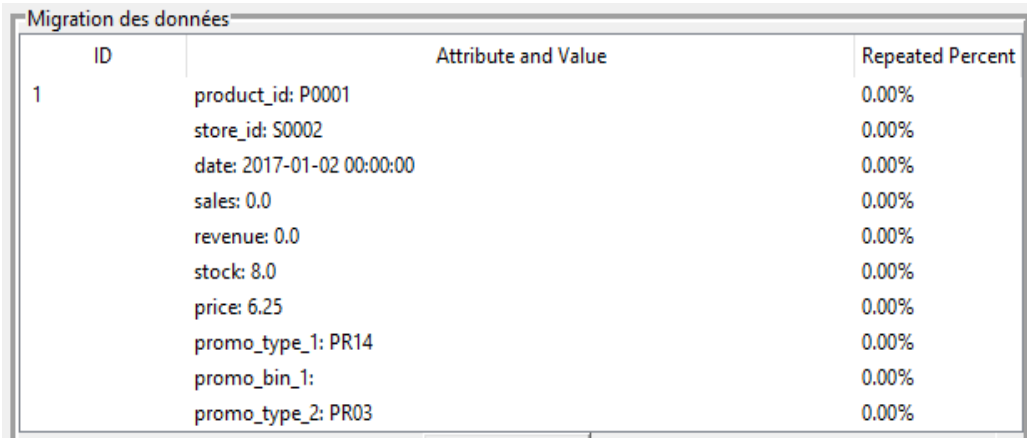

Figure 31 - affichage schéma plate

3.4. Conclusion

En résumé, ce chapitre a été consacré à la présentation de la modélisation du cas d'étude et à l'exploration des étapes nécessaires à la création d'un modèle de données dans une base de données NoSQL orientée colonnes. Nous avons également examiné en détail le schéma en étoile de notre modèle, mettant en évidence sa structure et ses composants clés et l'utilisation de processus traduction plate sur un modèle.

Dans le prochain chapitre, nous plongerons dans les détails de l'implémentation de la base de données, ainsi que dans l'analyse des données. Nous aborderons également les outils spécifiques utilisés pour modéliser notre système, en mettant l'accent sur leur rôle dans la création d'une infrastructure robuste et efficace pour notre application.

Chapitre 4

Implémentation

4.1. Introduction

Dans ce chapitre, nous montrerons l'implémentation sur une base de données Cassandra dans le modèle en colonnes, en utilisant des outils d'analyse de tables pour l'analyse des données et l'analyse des requêtes, et pour la connecter à Cassandra, nous utilisons ODBC, qui à son tour est un intermédiaire entre eux. Et créé notre application qui nous permet de gérer plus facilement Cassandra et de mettre en œuvre la transition de SQL vers NoSQL orienté colonnes.

4.2. Présentation de Cassandra et des outils nécessaires pour la BDD

Apache Cassandra est une base de données NoSQL distribuée et open source, orientée colonnes. Elle est largement adoptée par des milliers d'entreprises en raison de son évolutivité et de sa haute disponibilité, sans compromis sur les performances. Son évolutivité linéaire et sa tolérance éprouvée aux pannes, que ce soit sur du matériel de base ou sur une infrastructure cloud, en font une plateforme idéale pour les données critiques.

4.2.1 Caractéristiques de Cassandra

Dans cette section, nous détaillons les principales caractéristiques de Cassandra :

• **Base de données à colonnes :** Cassandra est structurée pour stocker des données en colonnes, ce qui la distingue des bases de données traditionnelles.

• **Cohérence, tolérance aux pannes et évolutivité :** Cassandra assure une forte cohérence des données, une tolérance aux pannes robustes et une capacité à s'adapter aux augmentations de charge sans dégradation des performances.

• **Origine et open source :** Initialement développée par Facebook, Cassandra a été par la suite rendue open source, permettant à une large communauté de développeurs de contribuer à son évolution.

• **Modèle de données inspiré de Google Bigtable :** Le modèle de données de Cassandra s'inspire de Google Bigtable, un système de stockage de données distribué à grande échelle.

• **Conception distribuée basée sur Amazon Dynamo :** La conception distribuée de Cassandra est influencée par Amazon Dynamo, garantissant une robustesse et une flexibilité accrues.

4.2.2 Installation de Cassandra et lance

Pour installer Apache Cassandra [46] sur un système Windows, il est nécessaire d'avoir Java 8. De plus, le Shell en ligne de commande de Cassandra (cqlsh) requiert Python 2.7 pour fonctionner correctement Voici les étapes pour lancer le serveur Cassandra :

- Assurez-vous que Java 8 est installé sur votre système.
- Vérifiez que Python 2.7 est installé pour pouvoir utiliser cqlsh.
- Exécutez la commande suivante pour démarrer le serveur Cassandra : **Cassandra** Après lancement serveur voire Cassandra

```
C:\Users\PC>cassandra
Detected powershell execution permissions.  Running with enhanced startup scripts.
```

Figure 32 - La commande Cassandra

```
Invite de commandes
356995377029703620, 761838145847967221 7, 766437515662241165, 783035799305476168 0, 790161524131831449 5, 796726754849796 2
324, 798164362178802068 1, 801100174029973672 6, 806722869806247273 4, 813279618921931571 4, 816809338375739640 6, 818905618 1
634329804, 820051468847379048 2, 832547932276779107 1, 833347838722229850 0, 845929688748938733 8, 847193517857104964 6, 8491
842223634677240, 854098054502319045 3, 858980648014778164 9, 864830246339068526 1, 866978010079762524 8, 867602673470041278 7
, 869121618832916230 5, 869824858186240667 1, 876657037919719494 0, 877707696710551243 0, 878585618906375366 8, 879073125241 7
456075, 893180459474415205 2, 902603877054795619 5, 903149432765326623 3, 906082281601748391 6, 929143468129571899, 9320529 9
3379124382, 938126495186818791]
INFO  [main] 2024-06-19 16:37:43,338 StorageService.java:1670 - JOINING: Finish joining ring
INFO  [main] 2024-06-19 16:37:43,419 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql1', ColumnFamily='magasins')
INFO  [main] 2024-06-19 16:37:43,421 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql1', ColumnFamily='produits')
INFO  [main] 2024-06-19 16:37:43,422 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql1', ColumnFamily='temps')
INFO  [main] 2024-06-19 16:37:43,425 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql1', ColumnFamily='promotions')
INFO  [main] 2024-06-19 16:37:43,426 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql', ColumnFamily='conversation_by_id_user')
INFO  [main] 2024-06-19 16:37:43,427 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql', ColumnFamily='message')
INFO  [main] 2024-06-19 16:37:43,428 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql', ColumnFamily='postulation')
INFO  [main] 2024-06-19 16:37:43,429 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql', ColumnFamily='profile_user')
INFO  [main] 2024-06-19 16:37:43,430 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql', ColumnFamily='find_user_by_nom_utilisateur')
INFO  [main] 2024-06-19 16:37:43,433 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='cassan
drademocql', ColumnFamily='time_line_offeres')
INFO  [main] 2024-06-19 16:37:43,497 StorageService.java:2595 - Node localhost/127.0.0.1 state jump to NORMAL
```

Figure 33 - lancement serveur Cassandra

4.3. Présentation de PyCharm et Installation

PyCharm est un environnement de développement intégré (IDE) spécialement conçu pour le langage de programmation Python. Développé par JetBrains, PyCharm est reconnu pour ses fonctionnalités robustes qui facilitent le développement de logiciels, la gestion des projets et l'analyse de code. [47]

Nous utilisons Visual studio code pour faciliter notre

Figure 34- Installation PyCharm

4.3.1. Présentation de Spark et Installation via PyCharm

Apache Spark est un moteur de traitement de données distribué open source, conçu pour des traitements rapides à grande échelle de grandes quantités de données. Spark est largement utilisé dans les applications de big data en raison de ses capacités de traitement en mémoire et de ses bibliothèques pour le SQL, le streaming de données, l'apprentissage automatique et le traitement de graphes.

Spark peut être installé facilement avec la commande pip.

Dans l'invite de commande, on exécute la commande suivante : pip Install pyspark. Cette commande télécharge et installe Spark, voir Figure 35. [21]

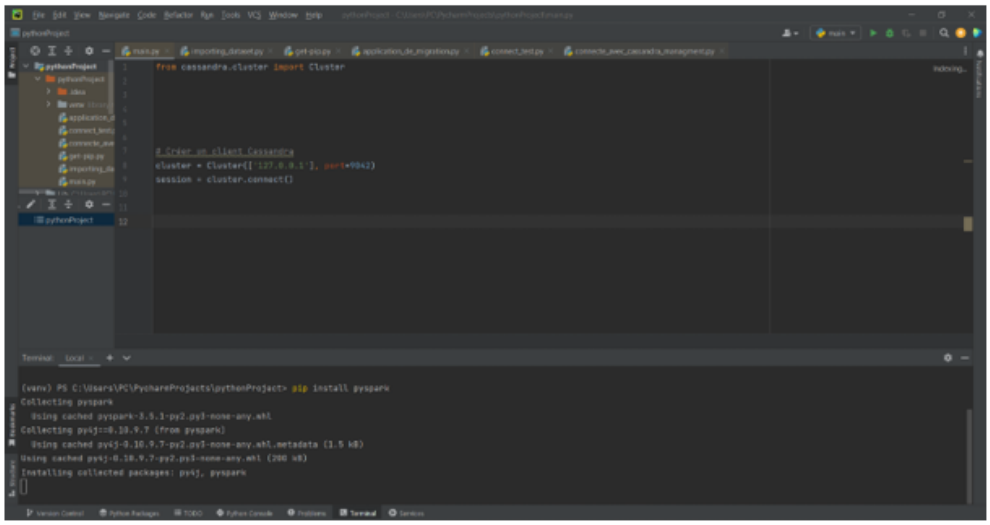

Figure 35- Téléchargement et installation Spark

But utilise Spark sur Cassandra :

Nous avons un fichier de 1 Go que nous divisons en 10 partitions de 100 Mo. Cela simplifie la lecture des données pour notre application et notre base de données Cassandra, tout en permettant à Spark de gérer efficacement chaque partition. Grâce à ce partitionnement, nous pouvons non seulement améliorer les performances de notre application, mais aussi garantir que chaque donnée est accessible et contrôlable de manière optimale.

En fin utilisant Spark pour diviser et gérer nos données, nous améliorons considérablement les performances et l'efficacité de notre système, tout en facilitant la lecture et le contrôle des données.

4.3.2. Création projet MIGRATION

- **Étape 1 :**

1. Ouvrez PyCharm et sélectionnez Create New Project.

2. Choisissez un emplacement pour votre projet et donnez-lui un nom.

3. Sélectionnez New environnent using et choisissez Virtualenv.

4. Assurez-vous que l'option Inherit global site-packages est décochée et cliquez sur Create.

- **Étape 2 :**

1.Ouvrez les paramètres du projet en allant dans File > Settings > Project : [Nom du projet] > Python Interpreter.

2.Cliquez sur l'icône d'engrenage à droite de la liste des interpréteurs et sélectionnez Add.

3.Choisissez Virtualenv Environnent, puis cliquez sur OK.

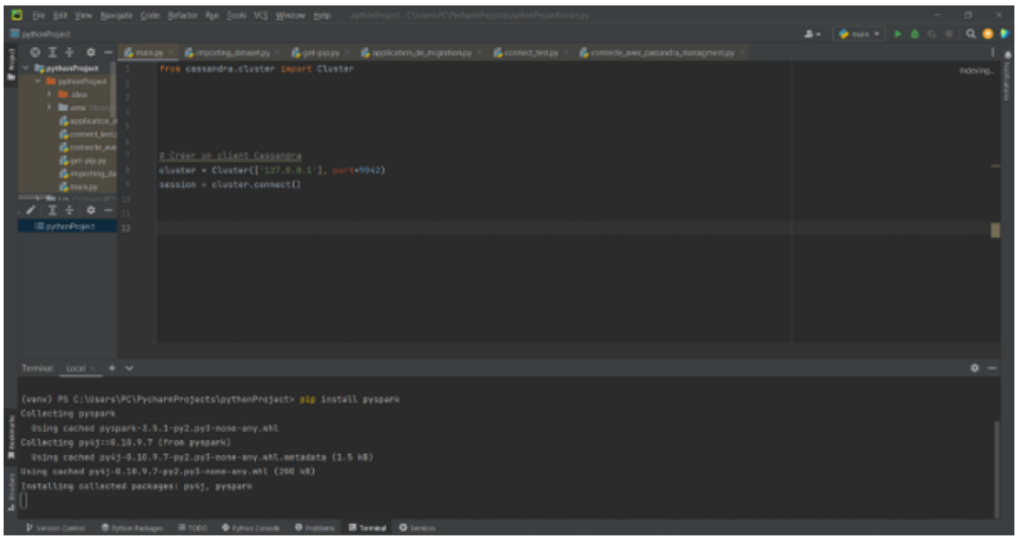

Figure 36- création Projet migration

4.3.3. Configuration Cassandra avec Spark

- **Étape 01** installation recommandée par commande **pip Install Cassandra-driver**, voir Figure 37
- **Étape 02** Nous avons vérifié si Cassandra-driver est installer

Figure 37- Installation Cassandra driver

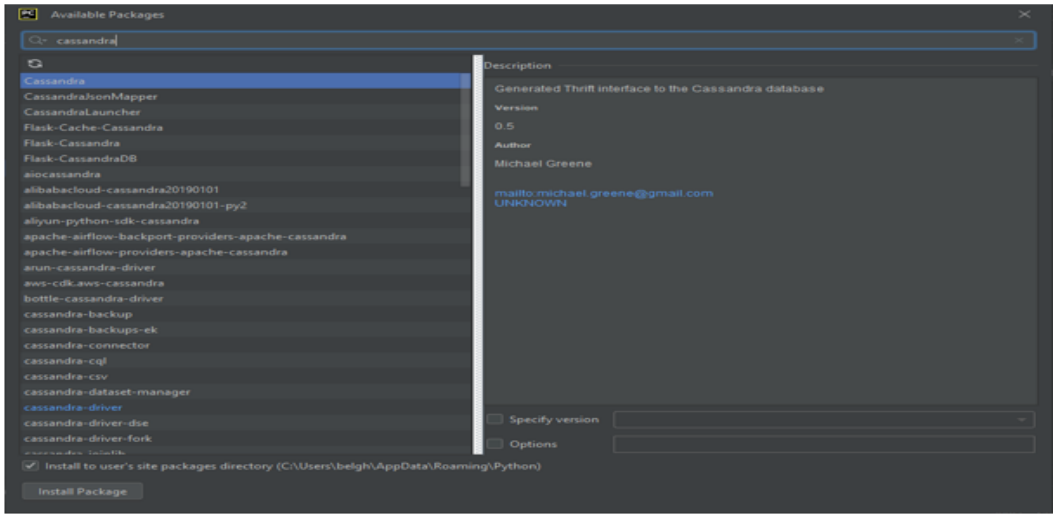

Figure 38- vérification de Cassandra-driver

- **Étape 03** Nous avons créé le code notre application

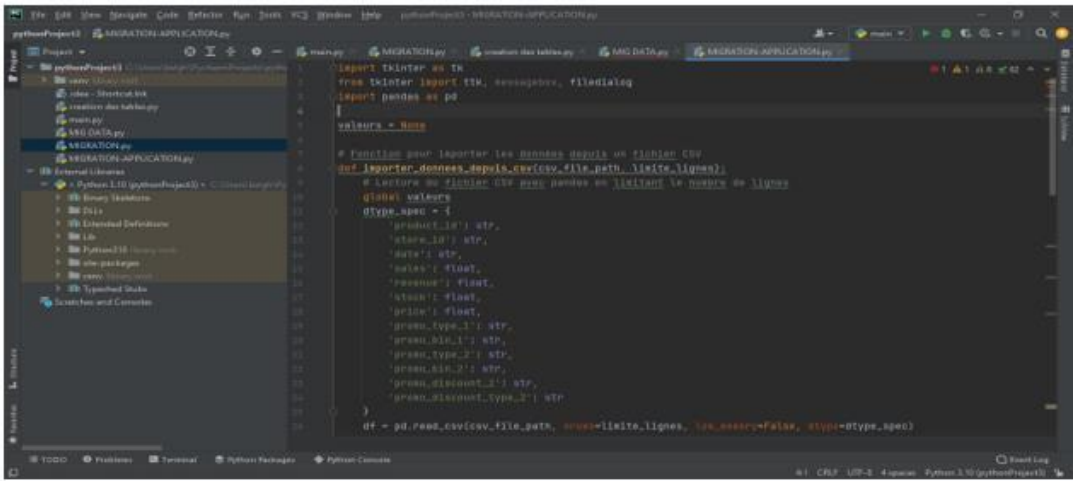

Figure 39- Code de l'application

4.3.4. Explication de l’application

Notre application à propos de deux fenêtres :

La première fenêtre (Connexion Cassandra et visualisation des données) compose de deux parties :

Premier partie qui va se connecte à Cassandra en utilisant l’hôte (127.0.0.1) et le port (9042) du serveur Cassandra après de choisir RDBMS(Cassandra). Ensuite, on clique sur le bouton "Connecter" et on attend jusqu'à voir le message de connexion. Puis, la deuxième partie contient l’Affichage des keyspace et leur famille de colonnes, en cliquant sur le bouton Rafraichir pour voir l’Affichage des keyspace qui déjà créé dans Cassandra .Ensuite pour voir leurs famille de colonnes il suffit juste choisir le keyspace et sélectionner, et pour voir leurs données en choisir un famille de colonne et sélectionner , et nous laissé le deuxième bouton qui nous avons nommé Migration à la fin de l’explication de premier fenêtre (Connexion Cassandra et visualisation des données) car cette dernier qui va faire un appelle a la deuxième fenêtre ,voir la Figure 40

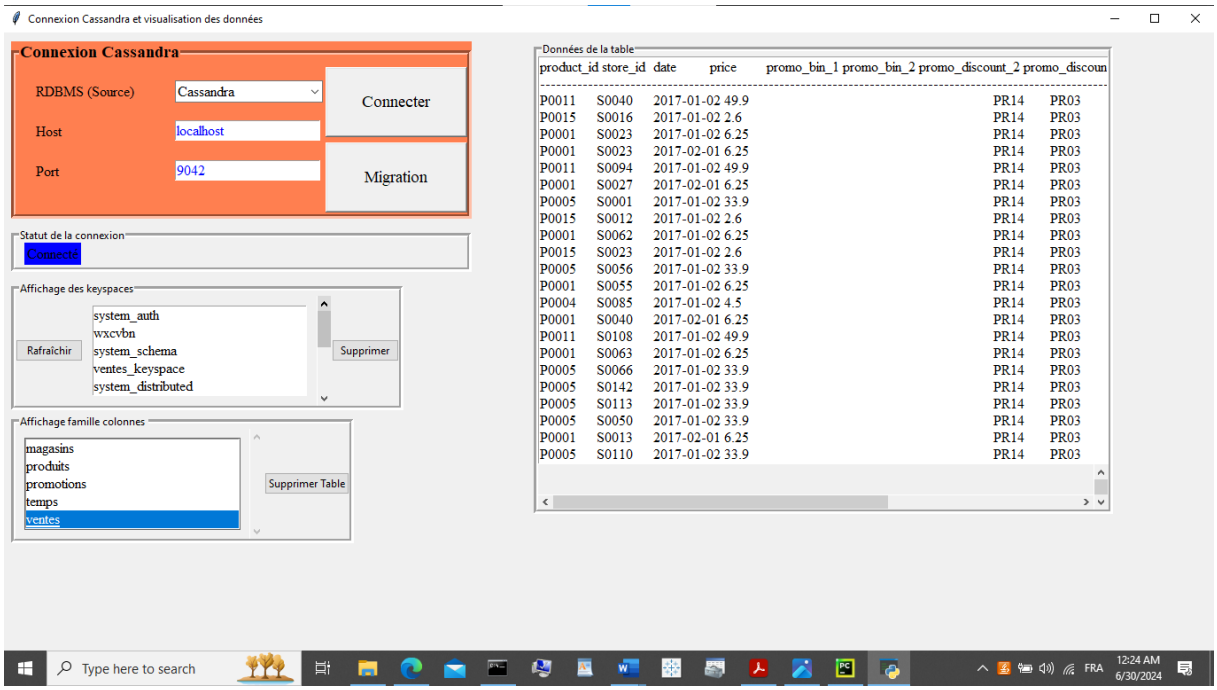

Figure 40 - - La première fenêtre (Connexion Cassandra et visualisation des données)

La deuxième fenêtre (Create keyspace et famille de colonnes et insertion des données) compose de deux espaces le premier pour créer le keyspace, et leur famille de colonnes ainsi leur colonnes et un affichage de colonnes puis affichage des tables si elles existe dans un keyspace apprêt la vérification par la Botton (vérifier keyspace), et le deuxième pour créer leur famille de colonnes par requête dans Cassandra, ainsi nous avons créé un Botton d'insertion un data depuis un fichier csv par choisir une table par un nom qui appartenir le même nom de fichier csv, et on a créé un label pour afficher le lien de fichier qui on a partitionner qui nous mentionner le lien et leur nom sur le code, donc ça fonctionne automatique juste apprêt le clic sur le Botton(PARTITIONE TABLE DE FAIT). Puis, on a un grand espace pour l'Affichage de notre modèle de traduction de plat, et tous ces espaces fonctionner par leurs bottons, ainsi il Ya la Botton de (rafraichir la connexion). Voir la figure 41

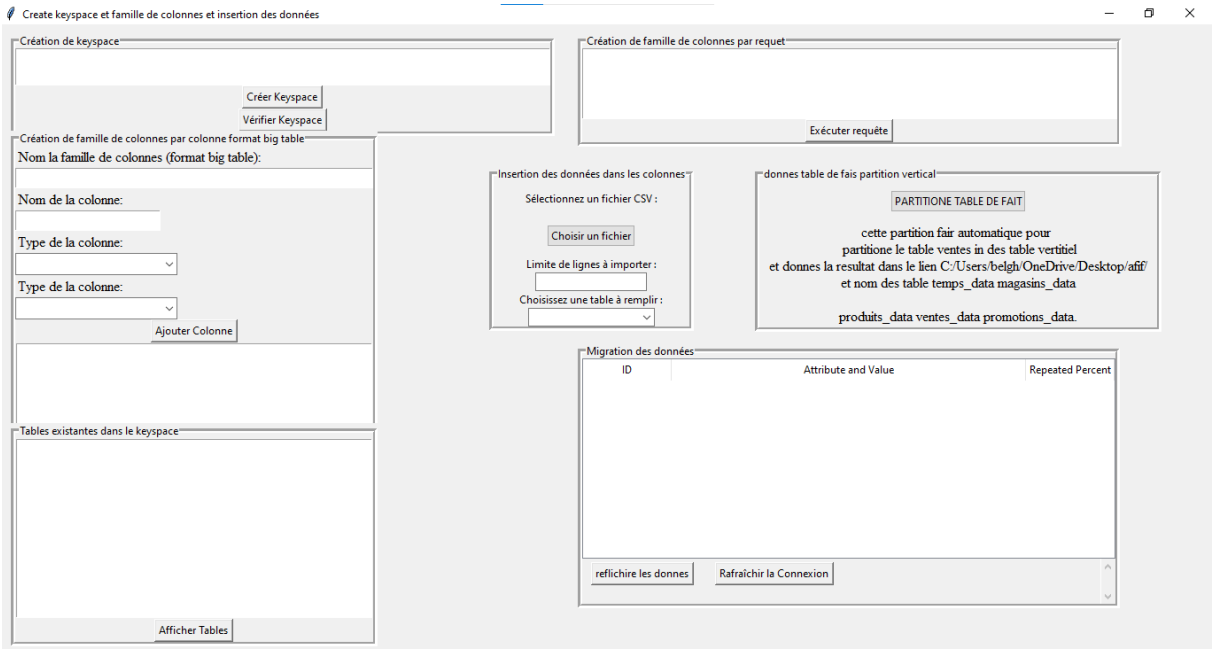

Figure 41 - La deuxième fenêtre (Create keyspace et famille de colonnes et insertion des données)

4.4. Présentation de ODBC CData

Le pilote ODBC CData pour Cassandra est un pilote ODBC conforme aux normes, disponible en versions pour les systèmes d'exploitation Windows et Unix, qui permet de faire une connexion de Cassandra avec Tableau utilisant l'hôte (127.0.0.1) et le port (9042) du serveur Cassandra et User (admin) ainsi Password (admin). Voir la figure 42.

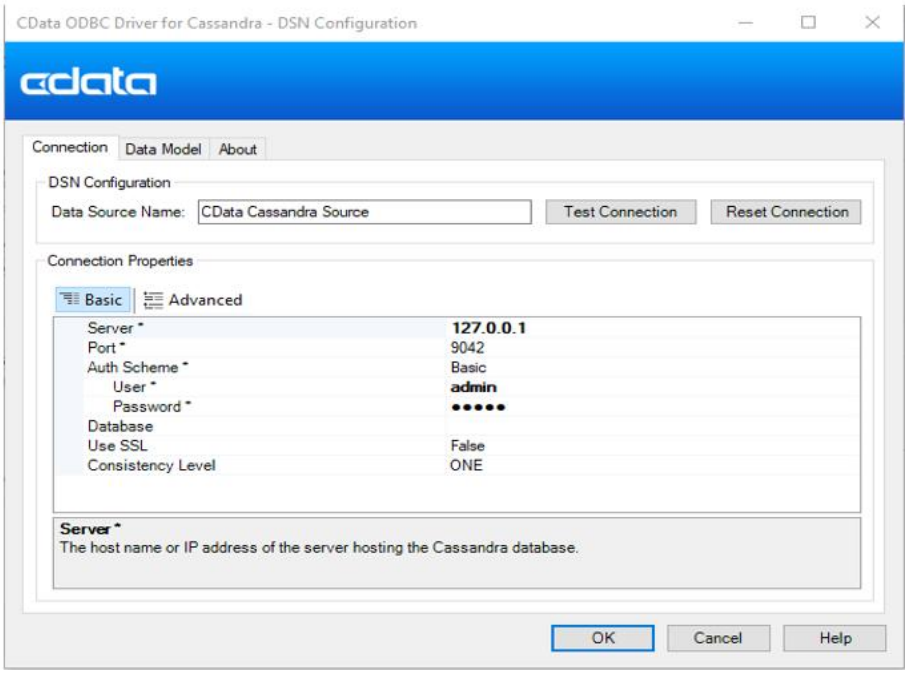

Figure 42- - CData ODBC Driver for Cassandra DSN Configuration

4.5. Présentation de l'outil tableau

Tableau est un outil puissant d'analyse et de visualisation de données conçu pour aider les analystes à voir et comprendre leurs données de manière intuitive. Cette plateforme d'analyse visuelle révolutionne l'utilisation des données en facilitant la résolution des problèmes grâce à des insights clairs et interactifs. Tableau permet aux utilisateurs d'explorer, analyser et partager des données complexes de manière simple et efficace, transformant ainsi la prise de décision basée sur les données. [28]

Après avoir la connexion avec Cassandra par ODBC, Voilà notre modèle voire Figure 43

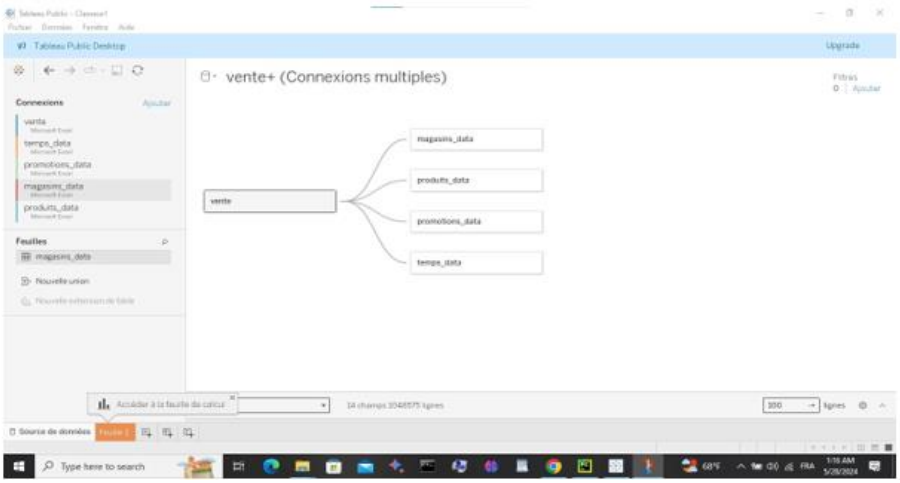

Figure 43- Les tables dans tableau

4.5.1. Exemple de requêtes analytiques

- **Requête 1**

« La valeur de prix et Revenu par jour » **La Figure 44** montre le résultat de cette requête.

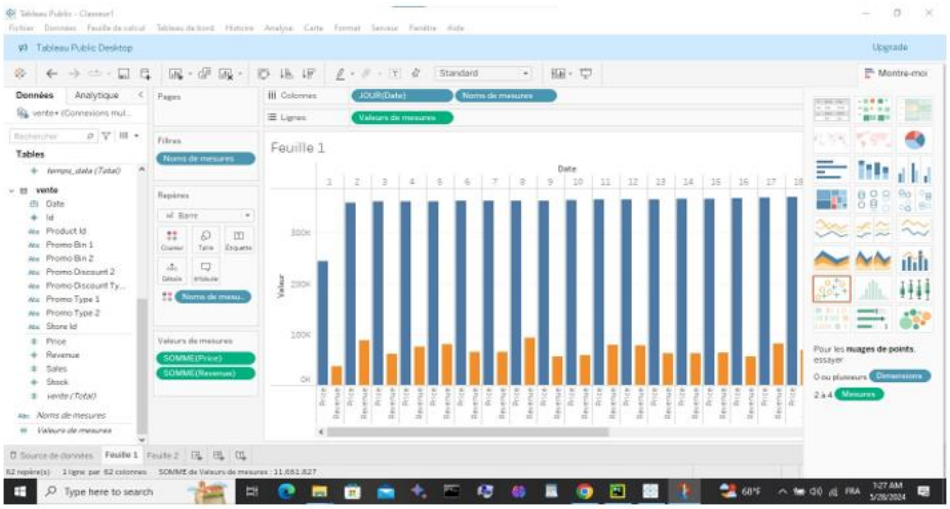

Figure 44-- Graph nombres des prix et revenu par jour

- **Requête 2**

« Produit_id par temps et Produit_id par stock et Produit_id par nombre de vents »,la Figure 45 montre le résultat de cette requête et Graph nombres de Produit_id par nombre de vents et stock et temps .

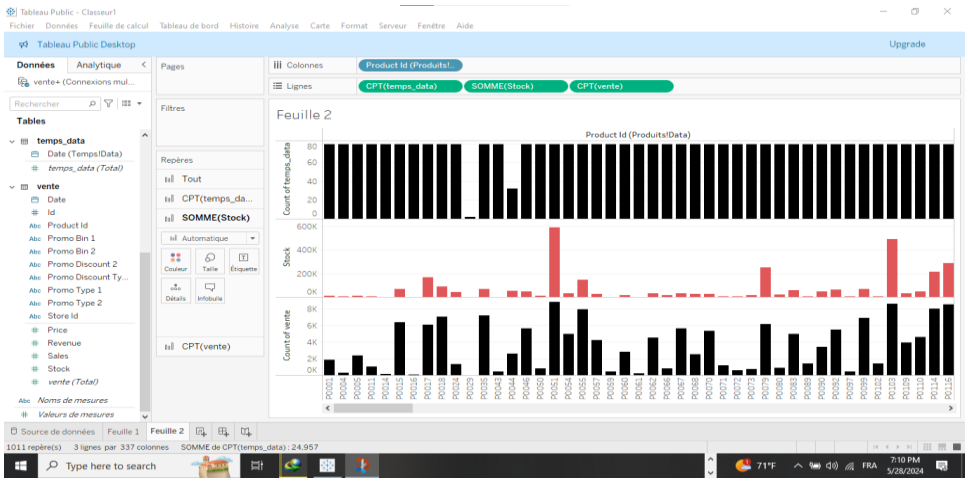

Figure 45 -nom de produit par temps et par vents et par la somme

- **Requête 3**

En applique la requête analytique qui affiche la méthode Plat **La Figure 46**« Produit_id par tous »

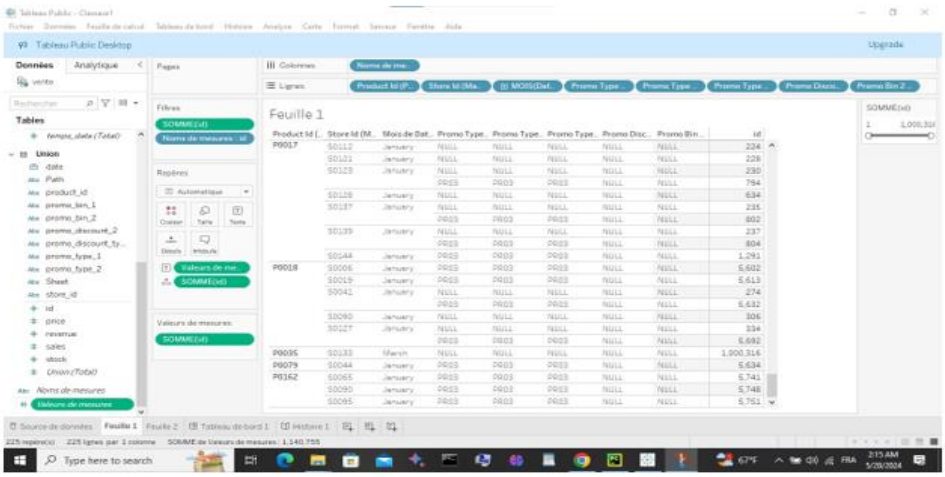

Figure 46- Produit_id par tous en famille colonne

4.6. Conclusion

Ce chapitre a détaillé les diverses étapes d'implémentation de la solution décisionnelle, ainsi que les outils spécifiques utilisés pour finaliser la création de notre modèle. Nous avons abordé chaque étape de manière systématique, illustrant comment les composants et technologies choisis ont contribué à la réalisation de notre objectif global MIGRATION.

Conclusion Générale

Le NoSQL est une technologie en pleine expansion, utilisée dans des environnements nécessitant la gestion de grandes masses de données, comme Google, Yahoo, Twitter, Facebook, etc. Les moteurs de recherche sont les premiers utilisateurs de ces technologies, car ils nécessitent une grande puissance de stockage et de traitement pour ces volumes de données. Les réseaux sociaux, eux aussi, doivent gérer une montée en charge massive due au grand nombre d'utilisateurs et de requêtes simultanées.

Dans le premier chapitre, nous avons abordé les entrepôts de données, leur construction avec les SGBDR et les bases multidimensionnelles telles que ROLAP, HOLAP, et MOLAP. Le deuxième chapitre a présenté la méthode NoSQL ainsi que ses différents types, en mettant un accent particulier sur Cassandra, une base de données NoSQL très performante pour le traitement de grandes quantités de données distribuées.

Notre projet de fin d'études s'est concentré sur la transformation d'un modèle de données conceptuel multidimensionnel en modèles logiques orientés colonnes, notamment avec Cassandra. Nous avons défini et instancié un entrepôt de données qui sera exploité dans une application d’aide à la décision, en traitant spécifiquement des données sur la migration.

Le rôle de notre projet est double : d'une part, il vise à démontrer la faisabilité et l'efficacité de l'utilisation de Cassandra pour la gestion de données complexes et volumineuses ; d'autre part, il fournit une solution concrète pour l'analyse des données, offrant ainsi un outil précieux pour les décideurs. En adoptant les technologies NoSQL, en particulier Cassandra, notre projet prépare le terrain pour des applications capables de traiter des flux de données massifs et complexes, répondant aux besoins actuels et futurs des entreprises en termes de robustesse et d'évolutivité.

Bibliographie

Les Articles et auteurs
[1] E.F. Codd. Providing OLAP (On-Line Analytical Processing) to useranalystes: an IT mandate. Technical report, 1993.

[2] A. Doucet and S. Gangarski. Entrepôts de données et Bases de Données Multidimensionnelles, Chapitre 12 du livre : Bases de Données et Internet, Modèles, langages et systèmes. Editions Hermès, 2001

[3] M. Adiba. Entrepôts de données et fouille de données, Introduction, Supports de cours, 2002.]

[4] z, C. Collet, and M. Adiba. Entrepôts de données : caractéristiques et problématique. Revue TSI, 20(2), 2001.

[5] E.F. Codd, « *A Relational Model of Data for Large Shared Data Banks* », *Communications of the ACM*, vol. 13, nº 6, 1970, p. 377–387 (DOI 10.1145/362384.362685)
Base de données relationnelle — Wikipédia (wikipedia.org) consulté le 7 /04/2024

[7] Namoune Mohamed Sofiane : Migration de données SQL vers les données NoSQL. Mémoire de Master, Université 08 Mai 1945, Guelma, 2019

[10] Pierre Senellart, « Limites des systèmes classiques de gestion de bases de données »

[11] AMARA Manel Warda, « Etude comparative des bases de données NoSQL », Année universitaire : 2014-2015

[12] Matteo Di Maglie, Adoption d'une solution NoSQL dans l'entreprise, Travail de Bachelor
réalisé en vue de l'obtention du Bachelor HES, Carouge, 12 septembre 2012 Haute École de Gestion de Genève (HEG-GE).

[13] Tournier, R. (2007). Analyse en ligne (OLAP) de documents (Doctoral dissertation, Université Paul Sabatier-Toulouse III).

[14] Lakhdari Mohamed Abderaouf, Ghoul Mohamed el Amine. Modélisation et Implémentation d'entrepôt de données en systèmes NoSQL orientés colonnes. Master 2 Ingénierie des Systèmes d'Information, UNIVERSITE ABDELHAMID IBN BADIS – MOSTAGANEM, Année Universitaire 2020-2021

[15] Serna Encinas, M. T. (2005). *Entrepôts de données pour l'aide à la décision médicale : conception et expérimentation* (Doctoral dissertation, Université Joseph Fourier (Grenoble; 1971-2015)).

[16] Tournier, R. (2007). *Analyse en ligne (OLAP) de documents* (Doctoral dissertation, Université Paul Sabatier-Toulouse III).

[17] Trinidad, S. E. M. (2005). *Entrepôts de données pour l'aide à la décision médicale : conception et expérimentation* (Doctoral dissertation, Université Joseph-Fourier-Grenoble I).

[20] Bruchez, R. (2015). *Les bases de données NoSQL et le BigData : Comprendre et mettre en oeuvre*. Editions Eyrolles.

[22] Kouedi Emmanuel, Approche de migration d'une base de données relationnelle vers une base de données NoSQL orientée colonne, Mémoire présenté en vue de l'obtention du diplôme de MASTER II RECHERCHE, Option : S.I & G.L ; Université de YAOUNDE I. Mai 2012.

[23] Adriano Girolamo PIAZZA, (2013), Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES, Haute École de Gestion de Genève (HEGGE).

[24] A. Aggoune, M.S. Namoune. "From Object-relational to NoSQL Databases: A Good Alternative to Deal with Large Data". The 1st International Conference on Innovative Trends in Computer Science CITCS'19, November 20-21, 2019, Guelma, Algérie.

[30] Adriano Girolamo PIAZZA, NoSQL Etat de l'art et benchmark ; Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES ; Genève, 9 octobre 2013
Haute École de Gestion de Genève (HEG-GE).
[31] Yangui, R., Nabli, A., & Gargouri, F. (2016). Automatic transformation of data warehouse schema to NoSQL data base: comparative study. *Procedia Computer Science*, *96*, 255-264.

[32] El Malki, M. (2016). *Modélisation NoSQL des entrepôts de données multidimensionnelles massives* (Doctoral dissertation, Université Toulouse le Mirail-Toulouse II).

[33] [Dehdouh et al., 2015] Khaled Dehdouh, Omar Boussaid, and Fadila Bentayeb. 2014a. Columnar NoSQL Star Schema Benchmark. In Yamine Ait Ameur, Ladjel Bellatreche, & George A. Papadopoulos, eds. Model and Data Engineering. Lecture Notes in Computer Science. Springer International Publishing, 281–288

[34] 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.

[35] Konstantinos Morfonios, Stratis Konakas, Yannis Ioannidis, and Nikolaos Kotsis. 2007a.

ROLAP Implementations of the Data Cube. ACM Comput Surv 39, 4.

[36] Ralph Kimball and Margy Ross. 2010. The Kimball Group Reader: Relentlessly Practical

[37] Alberto Abelló, Jaume Ferrarons, and Oscar Romero. 2011. Building Cubes with MapReduce. In Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP. DOLAP '11. ACM, 17–24.

[38] Lucas C. Scabora, Jaqueline J. Brito, Ricardo Rodrigues Ciferri, and Cristina Dutra de Aguiar Ciferri. 2016. Physical Data Warehouse Design on NoSQL Databases - OLAP Query Processing over HBase: In SCITEPRESS - Science and and Technology Publications, 111–118.

[39] Myller Claudino de Freitas, Damires Yluska Souza, and Ana Carolina Salgado. 2016. Conceptual Mappings to Convert Relational into NoSQL Databases: In SCITEPRESS - Science and and Technology Publications, 174–181. DOI:https://doi.org/10.5220/0005836301740181

[40] Rania Yangui, Ahlem Nabli, and Faiez Gargouri. 2016. Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study. Procedia Comput. Sci. 96 (2016), 255–264.

[41] MPINDA, Steve Ataky Tsham, MASCHIETTO, Luís Gustavo, et BUNGAMA, Patrick Andjasubu. From relational database to column-oriented NoSQL database: Migration process. *International Journal of Engineering Research & Technology (IJERT)*, 2015, vol. 4, p. 399-403.

[42] Li, Chongxin. "Transforming relational database into HBase: A case study." *2010 IEEE international conference on software engineering and service sciences*. IEEE, 2010.

[43] Lee, C. H., & Zheng, Y. L. (2015, October). SQL-to-NoSQL schema denormalization and migration: a study on content management systems. In *2015 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 2022-2026). IEEE.

[44] Schram, A., & Anderson, K. M. (2012, October). MySQL to NoSQL: data modeling challenges in supporting scalability. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity* (pp. 191-202).

[48] Arnaud Castelltort and Anne Laurent. 2014. NoSQL Graph-based OLAP Analysis: In SCITEPRESS - Science and and Technology Publications, 217–224.

[49] Fatma Abdelhédi, Amal Ait Brahim, Faten Atigui, and Gilles Zurfluh. 2016. Processus de transformation MDA d'un schéma conceptuel de données en un schéma logique NoSQL. In Actes du XXXIVème Congrès INFORSID, 15–30.

[50] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2016a. Document-oriented data warehouses: Models and extended cuboids, extended cuboids in oriented document. In Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, IEEE, 1–11.

[51] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2016b. Document-oriented Models for Data Warehouses - NoSQL Document-oriented for Data Warehouses.- 18th International Conference on Enterprise Information Systems, Volume 1 (ICEIS), SciTePress, 142–149.

[52] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, Ronan Tournier 2016c.Document-oriented data warehouses : complex hierarchies and summarizability. Dans : International Symposium on Ubiquitous Networking (UNet).

[53] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015a. Benchmark for OLAP on NoSQL technologies comparing NoSQL multidimensional data warehousing solutions. In 9th IEEE International Conference on Research Challenges in Information Science, RCIS 2015, IEEE, 480–485.

[54] 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.

[55] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015c. How Can We Implement a Multidimensional Data Warehouse Using NoSQL. Lecture Notes in Business Information Processing. Springer International Publishing, 108–130.
Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015d. Implementation of Multidimensional Databases in Column-Oriented NoSQL Systems. ADBIS 2015-Springer International Publishing, 79–91.

[56] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015e. Implementation of Multidimensional Databases with Document-Oriented NoSQL. 17th International Conference, DaWaK 2015, 1-4, 2015, Proceedings. Lecture Notes in Computer Science. Springer, 379–390.

[57] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015f. Implementing Multidimensional Data Warehouses into NoSQL. ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 1, SciTePress, 172–183.

[58] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, Ronan Tournier 2015g.Implantation Not Only SQL des bases de données multidimensionnelles. Dans : Colloque Veille Stratégique Scientifique et Technologique.

Documents web

[6] Que sont les transactions ACID ? (databricks.com) consulté le 8 /04/2024

[8] Synthèse d étude et projets d'intergiciels. Base NOSQL - PDF Téléchargement Gratuit (docplayer.fr) consulté le 10 /04/2024

[9] https://db-engines.com/en/ranking consulté le consulté le 12/05/2024

[18] tableau différence Différence entre OLTP et OLAP - WayToLearnX consulté le

7 /06/2024

[19] tableau différence OLTP et OLAP : différence entre les systèmes de traitement des données – AWS (amazon.com) consulté le 7 /06/2024

[25] Du SQL vers le NoSQL : Tendance ou réalité ? (veeam.com) consulté le 27 /05/2024

[26]https://www.guru99.com/nosql-tutorial.html consulté le 28/04/2024

[29] NoSQL Data Architecture & Data Governance: Everything You Need to Know - DATAVERSITY consulté le 25/04/2024

[45]Dataset Retail Sales Data (kaggle.com) ,consulté le 25 /05/2024

[46] Documentation officielle d'Apache Cassandra. Disponible à l'adresse : Apache Cassandra | Apache Cassandra Documentation, consulté le 6 /05/2024

[47]Téléchargement PyCharm https://www.jetbrains.com/pycharm/documentation/ , consulté le 5 /05/2024

[21] Téléchargement Spark https://spark.apache.org/documentation.html , consulté le 28 /04/2024

[27] http://www.mongodbmanager.com/cassandra, consulté le 02 /05/2024

[28] outils analyse Tableau https://www.tableau.com/why-tableau/what-is-tableau, consulté le 01 /05/2024