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

RAPPORT DE MINI-PROJET

Option : **Ingénierie des Systèmes d'Information**

THEME :

Modélisation et Implémentation d'entrepôt de données en systèmes NoSQL orientés colonnes

Etudiant(e): « **Lakhdari Mohamed Abderaouf** »

« **Ghoul Mohamed el Amine** »

Encadrant(e) : « **Bensalloua Abdallah Charef** »

Année Universitaire 2020-2021

Remerciements

En premier, nous aimerions remercier le bon Dieu le tout puissant de nous avoir donné le courage et la volonté de réaliser ce projet.

Nous désirons remercier nos chers parents qui nous ont soutenus et encourage durant toute notre vie et pendant notre cursus d’étude.

Nos remerciements les plus chaleureux vont à Mlle B.CHAREF ABDELAH pour leur disponibilité et leur très précieux conseils ainsi que leur remarques qui nous ont permis d'améliorer la qualité de ce travail.

Nous tenons à exprimer toute notre grande gratitude aux membres de jury d'avoir accepté de juger ce travail.

Nos vifs remerciements s’adressent également à tous nos enseignants de la faculté sciences exactes et d’Informatique de l'université ABDELHAMID IBN BADIS - MOSTAGANEM pour la formation qu'ils ont eu le soin de nous apporter le long de notre cursus universitaire.

Résumé

Le fonctionnement normal des entreprises et autres organisations dépend de la gestion, de la compréhension et de l'utilisation efficace de grandes quantités des données non structurées telles que les médias sociaux, le contenu Web, les documents, etc.

À cette fin, la technologie DW (Data Warehouse) est utilisée pour le stockage, la visualisation, l'analyse et la comparaison, permettant aux utilisateurs de générer des rapports et des graphiques.

Dans ce projet, nous devons appliquer des solutions de pointe dans le domaine des différentes techniques de modélisation NoSql pour l'entrepôt de données. Ensuite, nous proposerons le processus d'implémentation d'un data Warehouse multidimensionnel avec un modèle NoSQL de colonne. Des règles de mappage qui transforment le modèle de données conceptuel multidimensionnel en modèles logiques orientés colonnes seront définies pour créer une instance de l'entrepôt de données qui sera utilisée dans l'application d'aide à la décision qui sera développée à cet effet.

Mots-clés :

Aide à la décision, DW Data Warehouse, NoSQL data base, modèle NoSQL de colonne

Abstract

The normal functioning of businesses and other organizations depends on the management, understanding and effective use of large amounts of unstructured data such as social media, web content, documents, etc.

For this purpose, DW (Data Warehouse) technology is used for storage, visualization, analysis and comparison, enabling users to generate reports and charts.

In this project, we need to apply state-of-the-art solutions in the area of different NoSQL modeling techniques to the data warehouse. Next, we propose the process of implementing a multidimensional data warehouse with a column NoSQL model. Mapping rules that transform the multidimensional conceptual data model into column-oriented logic models will be defined to create a data warehouse instance that will be used in the decision support application that will be developed for this purpose.

Keywords:

the decision support, DW Data warehouse, NoSQL data base, column NoSQL model

ملخص:

يتوقف الأداء العادي للأعمال التجارية وغيرها من المنظمات على إدارة وفهم واستخدام كميات كبيرة من البيانات غير المنظمة مثل وسائط الإعلام الاجتماعية ومحتوى الشبكة والوثائق وما إلى ذلك.

ولهذا الغرض ، تستخدم تكنولوجيا "مستودع البيانات" للتخزين والتصور والتحليل والمقارنة ، مما يمكن المستخدمين من إنتاج التقارير والرسوم البيانية.

وفي هذا المشروع ، نحتاج إلى تطبيق أحدث الحلول في مجال مختلف تقنيات نمذجة NSQL **على مستودع البيانات. وبعد ذلك ، نقترح عملية تنفيذ مستودع بيانات متعدد الأبعاد مع نموذج** NSQL **للعمود. وسيجري تعريف قواعد رسم الخرائط التي تحول نموذج البيانات المفاهيمية المتعددة الأبعاد إلى نماذج منطقية موجهة نحو الأعمدة من أجل إنشاء جهاز لتخزين البيانات يستخدم في دعم اتخاذ القرارات**

كلمات مفتاحية:

دعم القرار، مستودع بيانات **DW**، قاعدة بيانات **NSQL**، العمود نموذج **NSQL**

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
SGBD	System Gestions Base de Données	3
BDD	Base de données	3
SQL	Structured Query Langage	3
ED	Entrepôt de données	4
DW	Data Warehouse	4
OLAP	Online Analytical Processing	11
OLTP	Online Transaction Processing	11
ROLAP	Online Analytical Processing Relationnel	20
MOLAP	Online Analytical Processing Multidimensionnel	21
HOLAP	Online Analytical Processing Hybride	22
NOSQL	Not Only SQL	22
CAP	Consistency, Availability, Partition Tolérance	25
ACID	Atomicité, cohérence, isolation et durabilité	26
BLOB	Binary large object	26
JSON	JavaScript Object Notation	28

Table of Contents

Introduction Générale

Nous vivons dans un monde axé sur les données 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 les 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'elle peut être analysée pour des études de marché, des rapports d'analyse et une prise de décision. La principale différence avec les bases de données opérationnelles traditionnelles est que les entrepôts de données sont généralement conçus pour fournir des informations historiques plutôt que les dernières données.

Les bases de données relationnelles, comme celles de MySQL et PostgreSQL, stockent les données en utilisant un schéma explicite. Un schéma décrit comment écrire des données dans la base de données, en particulier la structure, les types et les structures des tableaux et des enregistrements. Dans une base de données NoSQL, les utilisateurs ne définissent pas de schéma. Au lieu de cela, ils peuvent stocker des données en utilisant n'importe quelle structure, avec une requête SQL relationnelle fournissant des données que les utilisateurs peuvent ensuite utiliser pour mettre à jour ces données. Les deux types de systèmes de base de données offrent également différents moyens d'accéder aux données.

Dans ce travail, nous avons effectué une étude bibliographique concernant les entrepôts de données et la modélisation multidimensionnelle avec les systèmes NOSQL, nous allons baser sur les systèmes NoSQL orienté colonne.

Ce mémoire est organisé comme suite, Le premier chapitre traite les entrepôts de données leurs constructions avec les SGBDR et les bases multidimensionnelles telle que ROLAP, HOLAP, MOLAP. Le deuxième chapitre on va présenter la méthode dite NoSQL ainsi que ces différents types, En fin on terminera par une conclusion générale.

Le troisième chapitre on va présenter le thème sur lequel en va construire notre entrepôt de données, c'est le thème de l'application web SOLVE. Le quatrième chapitre c'est la modélisation d'un entrepôt de donnée NoSQL orienter colonne avec le SGBD Cassandra. En fin on terminera par une conclusion générale.

Chapitre 1

Les entrepôts de données Multidimensionnels

1.1 Introduction

Les entrepôts de données ED (Data Warehouse DW) sont apparus vers les années 1990 en réponse à la né cécité de rassembler toutes les informations de l'entreprise en une base de données unique destinée aux analystes et aux gestionnaires. L'ensemble des données, y compris leur historique, est utilisé dans de nombreux domaines, tels que : l'analyse de données et l'aide à la décision (gestion et analyse de marché, gestion et analyse du risque gestion et détection des fraudes), dans antres applications (recherches dans des textes, dans les documents web, dans l'astronomie.)

Dans ce chapitre, nous analysons aussi bien les caractéristiques des entrepôts que leurs aspects temporels. Ce chapitre décrit les différents types de modélisation multidimensionnelle.

1.2 Les entrepôts de données

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.

1.3 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.

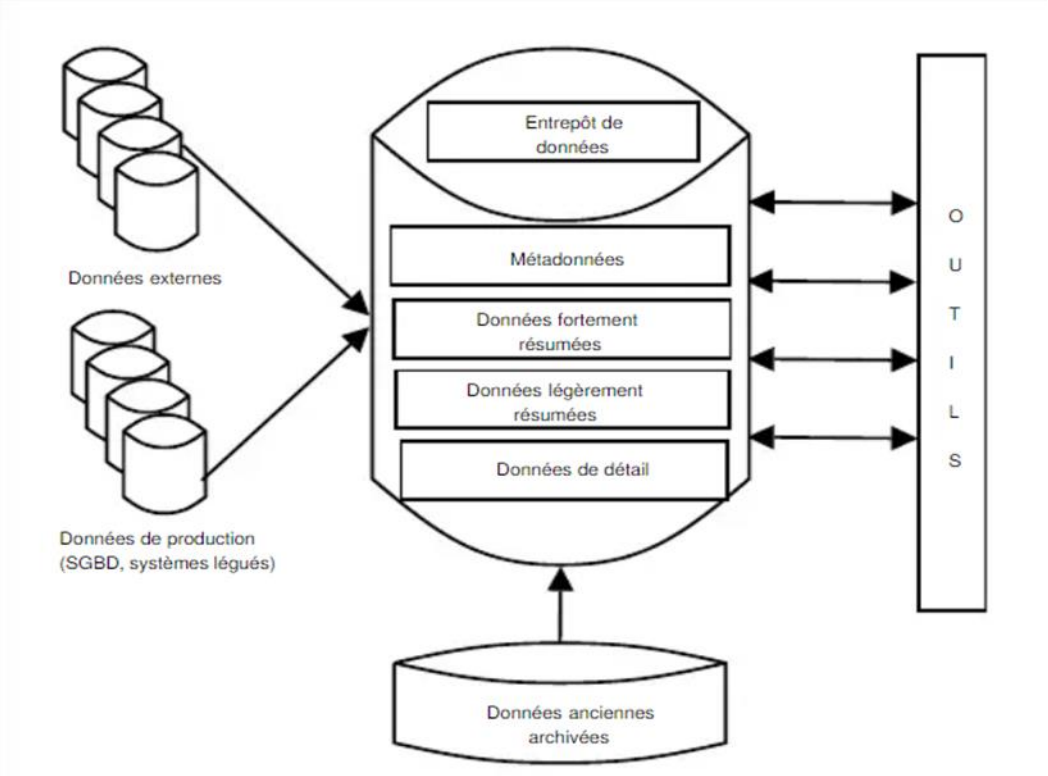


Figure 1.1 Une architecture entrepôt de données

1.3.1 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 : 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.

1.3.2 L'entrepôt de données

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

- **Données de détail courantes :** Ce sont l'ensemble des données quotidiennes et plus couramment utilisées. Ces données sont généralement stockées sur le disque pour avoir un accès rapide. Par exemple, le détail des ventes de l'année en cours, dans les différents magasins.
- **Données de détail anciennes :** Ce sont des données quotidiennes concernant des événements passés, comme par exemple le détail des ventes des deux dernières années. Nous les utilisons pour arriver à l'analyse des tendances ou des requêtes prévisionnelles. Néanmoins ces données sont plus rarement utilisées que les précédentes, et elles sont souvent stockées sur des mémoires d'archives.
- **Données résumées ou agrégées :** Ce sont des données moins détaillées que les deux premières et elles permettent de réduire le volume des données à stocker. Le type de données, en fonction de leur niveau de détail. Permet de les classifier comme des données légèrement ou fortement résumées. Par exemple, les ventes mensuelles par magasin des dix dernières années sont des données faiblement résumées, tandis que les ventes semestrielles, par région, des dix dernières années sont fortement résumées.

- **Les métadonnées :** Ce sont des données essentielles pour parvenir à une exploitation efficace du contenu d'un entrepôt. Elles représentent des informations nécessaires à l'accès et l'exploitation des données dans l'entrepôt comme : la sémantique (leur signification). L'origine (leur provenance), les règles d'agrégation (leur périmètre), le stockage (leur format. Par exemple : francs, euro…) et finalement l'utilisation (par quels programmes sont-elles utilisées).

1.3.3 Outils

Il existe sur le marché différents outils pour l'aide à la décision, comme les outils de fouille de 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 requetés), ...

1.4 Entrepôts et les bases de données

Dans l'environnement des entrepôts de données, les opérations, l'organisation des données, les critères de performance, la gestion des métadonnées, la gestion des transactions et le processus de requêtes sont très différents des systèmes de bases de données opérationnelles. Par conséquent, les SGBD relationnels orientés vers l'environnement opérationnel, ne peuvent pas être directement transplantés dans un système d'entrepôt de données.

Les SGBD ont été créés pour les applications de gestion de systèmes transactionnels. Par contre, les entrepôts de données ont été conçus pour l'aide à la prise de décision. Ils intègrent les informations qui ont pour objectif de fournir une vue globale de l'information aux analystes et aux décideurs.

Le tableau 1.1 résume ces différences entre les systèmes de gestion de bases de données et les entrepôts de données.

	SGBD	Entrepôts de données
Objectifs	Gestion et production	Consultation et analyse
Utilisateurs	Gestionnaires de production	Décideurs, analystes
Taille de la base	Plusieurs giga-octets	Plusieurs téraoctets
Organisation des données	Par traitement	Par métier
Type de données	Données de gestion (courantes)	Données d'analyse (Résumées, historiées)
Requetés	Simples, prédéterminées données détaillées,	Complexes, spécifiques, Agrégations et group by
Transactions	Courtes et nombreuses, temps réel	Longues, peu nombreuses

Tableaux 1.1 Différences entre SGBD et entrepôts de données

1.5 Caractéristiques des entrepôts de données

Les données d'un entrepôt de données possèdent les caractéristiques suivantes :

1.5.1 Orientées sujet

Les données des entrepôts sont organisées par sujet plutôt que par application. Par exemple, une chaîne de magasins d'alimentation organise les données de son entrepôt par rapport aux ventes qui ont été réalisées par produit et par magasin, au cours d'un certain temps.

1.5.2 Intégrées

Les données provenant des différentes sources doivent être intégrées, avant leur stockage dans l'entrepôt de données. L'intégration (mise en correspondance des formats, par exemple), permet d'avoir une cohérence de l'information.

1.5.3 Non volatiles

A la différence des données opérationnelles, celles de l'entrepôt sont permanentes et ne peuvent pas être modifiées. Le rafraîchissement de l'entrepôt, consiste à ajouter de nouvelles données, sans modifier ou perdre celles qui existent.

1.5.4 Historisées

La prise en compte de l'évolution des données est essentielle pour la prise de décision qui, par exemple, utilise des techniques de prédiction en s'appuyant sur les évolutions passées pour prévoir les évolutions futures.

1.6 Objectifs d'un entrepôt de données

- Les données de l'organisation doivent être accessible facilement (données parlantes, et signification évidente pas seulement pour le développeur mais surtout pour l'utilisateur

- La présentation des informations de manière cohérente : Les données doivent être assemblées à partir des différentes sources de l'entreprise et il faut contrôler la qualité et éviter de les mettre à disposition quand elles ne sont pas nettoyées. Attention si deux mesures sont identiques, elles doivent porter le même nom pour éviter toute confusion et inversement ne pas mettre le même nom pour deux mesures qui ne sont pas calculées [10]

- Il doit être adaptable

- Il doit efficacement protégées les informations de l'entreprise

- Il doit être utilisé lors de la prise de décision et par l'ensemble de la communauté

- Il doit servir de stockage d'information

1.7 Modélisation multidimensionnelle

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.[1]

	S. Transactionnel	S. Décisionnel
Données	Exhaustives Courantes Dynamiques Orientées applications	Résumées Historiques Statiques Orientées sujets (d'analyse)
Utilisateurs	Nombreux Varie (employés, directeurs, ...) Concurrents Mises à jour et interrogations Requetés prédéfinis Réponses immédiates Accès a peu d'information	Peu nombreux Uniquement les décideurs Non concurrents Interrogations Requêtes imprévisibles et complexes Réponses moins rapides Accès à de nombreuses Informations

Tableaux 1.2 Comparaison des systèmes

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).[2][3][4][6]

1.7.1 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.

La figure 1.2 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.

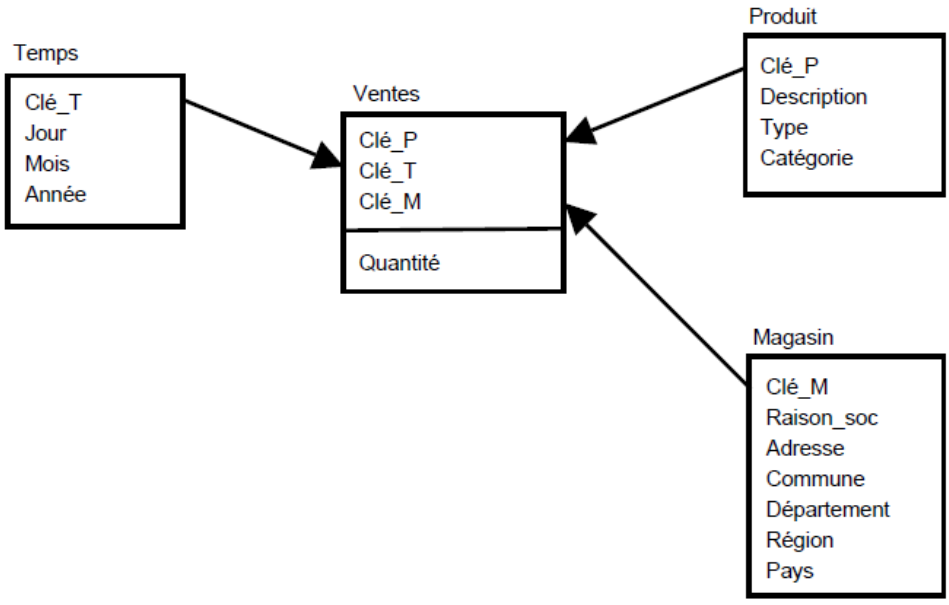


Figure 1.2 Exemple de modélisation en étoile.

- **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. [5]

La figure 1.3 montre le schéma en flocon de neige avec les dimensions Temps et magasin éclatées en sous hiérarchies.

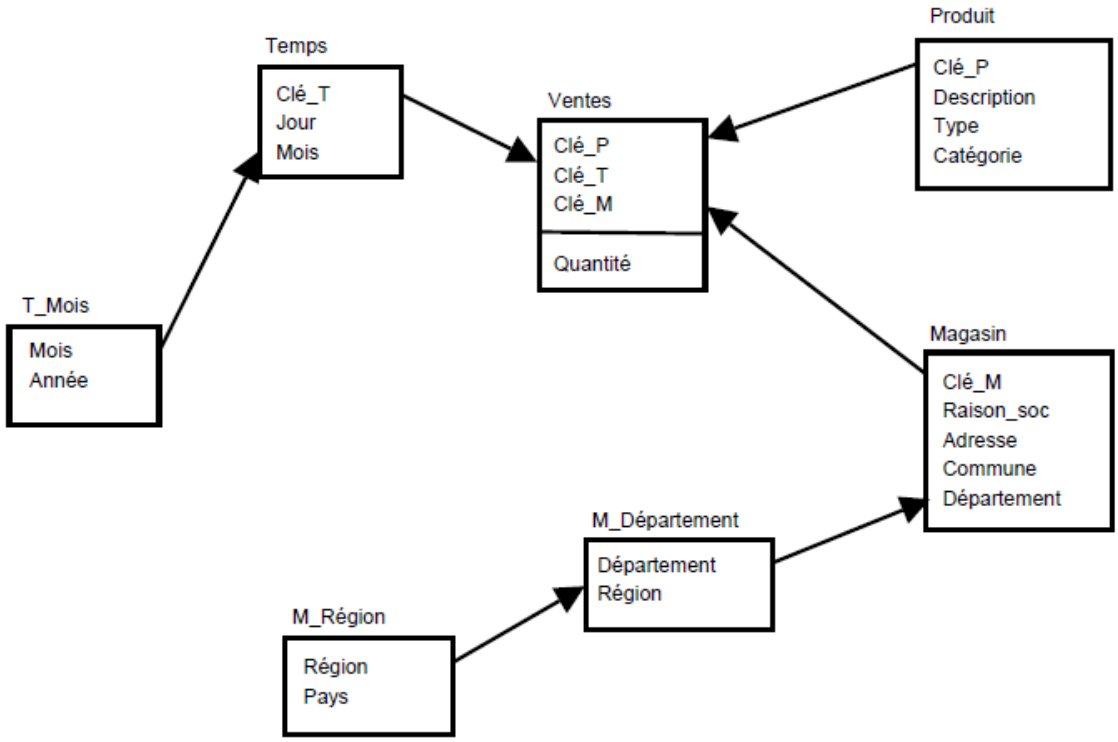


Figure 1.3 Exemple de modélisation en flocon de neige.

- **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. [7]

La figure 1.4 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.

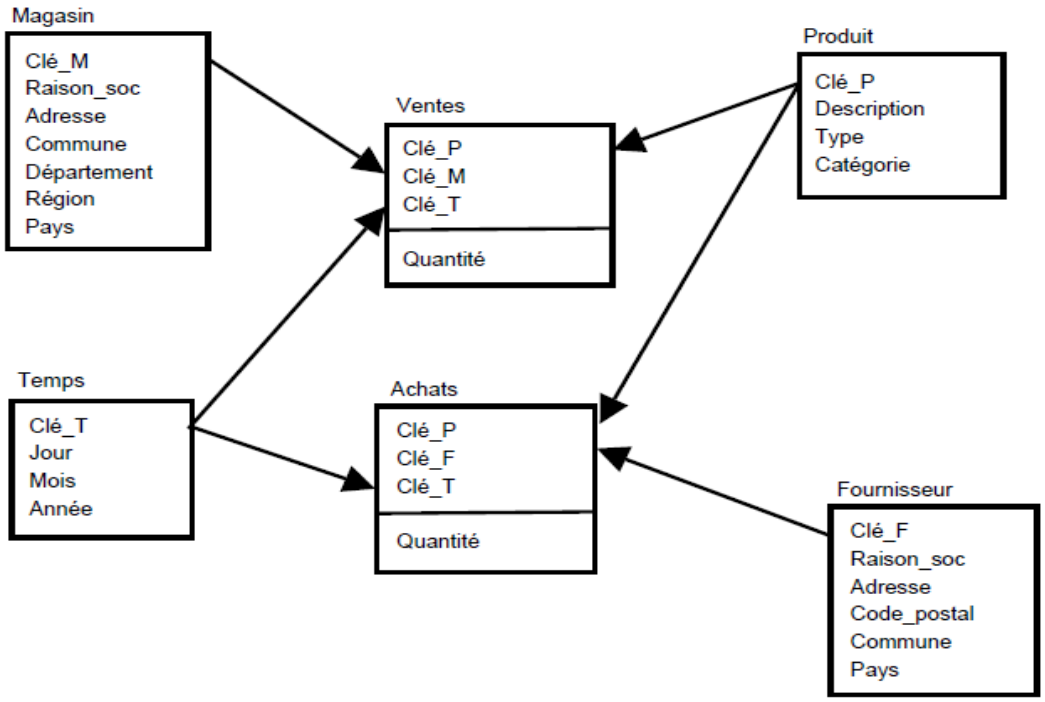


Figure 1.4 Exemple de modélisation en constellation.

- **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.
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 1.2, nous pouvons construire le schéma multidimensionnel suivant.

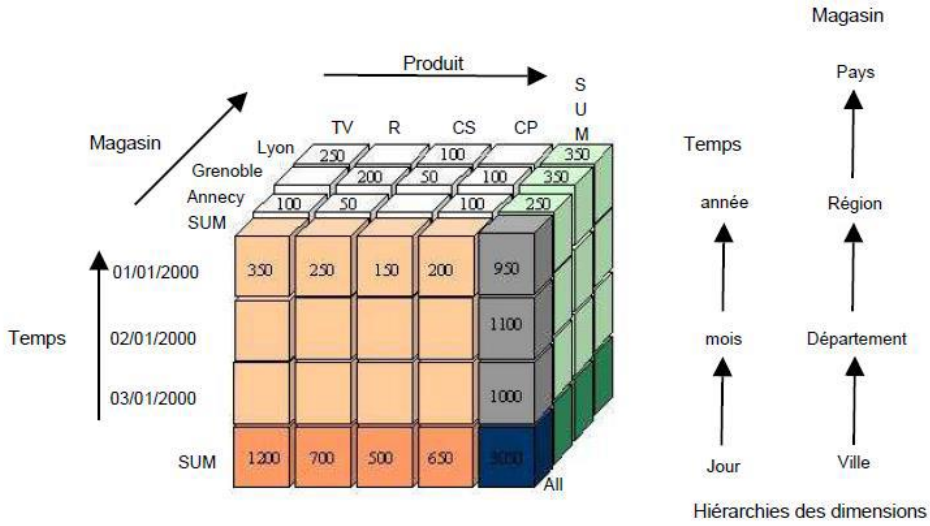


Figure 1.5 Exemple de schéma multidimensionnel.

1.8 Manipulation des données multidimensionnelles

Pour visualiser les données multidimensionnelles, nous pouvons utiliser la représentation sous forme d'une table de données, qui est la plus courante. Dans une table, nous représentons les différentes combinaisons des valeurs choisies pour constituer les noms de lignes et de colonnes.[8][6]

Nous présentons les opérations pour la manipulation des données multidimensionnelles, en les divisant selon leur impact sur la façon de présenter les différentes vues des données analysées.[9]

1.8.1 Opérations classiques

Ces opérations correspondent aux opérations relationnelles de manipulation des données :

La sélection : Résulte en un sous-ensemble de données qui respecte certaines conditions d'appartenance. Nous pouvons avoir une sélection avec des conditions soit sur les mesures, soit sur les membres. Par exemple, une sélection des ventes supérieur à 350 est une sélection sur les mesures, tandis qu'une sélection des ventes réalisés dans la région « Rhône Alpes » de l'année « 2000 » est une sélection sur les membres d'une dimension.

La projection : Résulte en un sous-ensemble des attributs d'une relation, qui sont soit des dimensions, soit des niveaux de granularité. Dans les systèmes décisionnels, les opérations de sélection et de projection sont appelées souvent « slice-and-dice ».
La jointure : Permet d'associer les données de relations différentes. Par exemple, en utilisant les tables 1.3 et 1.4, nous faisons une jointure sur la dimension Produit. L'objectif est de représenter sur une même table la quantité de produits vendue et leur prix (*cf*. Figure 1.6).

Ventes (01/01/2000)	Mag1	Mag2	Mag3
Téléviseur	100		250
Radio	50	200	
Caméscope		50	100
Camera photo	100	100	

Tableaux 1.3 Table de vente par magasin

Produit	Prix (01/01/2000)
Téléviseur	1
Radio	2
Caméscope	3
Camera photo	4

Tableaux 1.4 Table des prix des produits

Les opérations ensemblistes : D'union, d'intersection et de différence sont des opérations qui agissent sur des relations qui ont le même schéma. Par exemple, l'union des tables 1.5 et 1.6 donne comme résultat la table 1.7.

1.8.2 Opérations agissant sur la structure

Les opérations agissant sur la structure visent à présenter une vue (face du cube) différente en fonction de leur analyse citons :

La rotation (*rotate*) : Consiste à pivoter ou à effectuer une rotation du cube, de manière à présenter une vue différente des données à analyser.

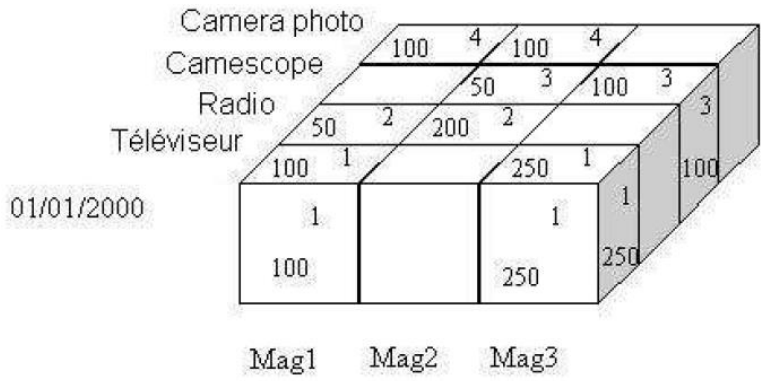


Figure 1.6 Exemple de l'opération Jointure.

Produit	Ville	Mag1	Mag2	Mag3
Téléviseur	**Grenoble**	50	100	
Radio	**Grenoble**	50		100
Caméscope	**Grenoble**	100		
Camera photo	**Grenoble**		50	100

Tableaux 1.5 Table de vente des produits par ville 1

Produit	Ville	Mag1	Mag2	Mag3
Téléviseur	**Fontaine**	100	100	50
Radio	**Fontaine**	50		100
Caméscope	**Fontaine**	100	50	100
Camera photo	**Fontaine**	100		100

Tableaux 1.6 Table des ventes de produits par ville 2

Produit	Ville	Mag1	Mag2	Mag3
Téléviseur	**Grenoble**	50	100	
Radio	**Grenoble**	50		100
Caméscope	**Grenoble**	100		
Camera photo	**Grenoble**		50	100
Téléviseur	**Fontaine**	100	100	50
Radio	**Fontaine**	50		100
Caméscope	**Fontaine**	100	50	100
Camera photo	**Fontaine**	100		100

Tableaux 1.7 Table des ventes du Département Isère

La permutation (*switch*) : Consiste à inverser des membres d'une dimension, de manière à permuter deux tranches du cube.
La division (*split*) : Consiste à présenter chaque tranche du cube en passant d'une représentation tridimensionnelle à une présentation tabulaire. Par exemple, si nous découpons par magasin le cube de ventes de la figure 1.5, nous avons les tables 1.8, 1.9 et 1.10.

Ventes Magasin 1	01/01/2000	02/01/2000	03/01/2000
Téléviseur	100	100	50
Radio	50	200	100
Caméscope			100
Camera photo	100	100	100

Tableaux 1.8 Table de ventes du Magasin 1

Ventes Magasin 2	01/01/2000	02/01/2000	03/01/2000
Téléviseur		100	150
Radio	200	200	100
Caméscope	50	100	
Camera photo	100		100

Tableaux 1.9 Table de ventes du Magasin 2

Ventes Magasin 3	01/01/2000	02/01/2000	03/01/2000
Téléviseur	250	100	150
Radio			100
Caméscope	100	50	50
Camera photo		100	100

Tableaux 1.10 Table de ventes du Magasin 3

Nous remarquons que le nombre de tables résultantes de cette opération dépend du nombre de valeurs à l'intérieur de la dimension.

L'emboîtement (*nest*) : Permet d'imbriquer les membres d'une dimension. En utilisant cette opération, nous représentons dans une table bidimensionnelle toutes les données d'un cube quel que soit le nombre de dimensions.

L'enfoncement (*push*) : Consiste à combiner les membres d'une dimension aux mesures du cube et donc de représenter un membre comme une mesure.

L'opération inverse de retrait (*pull*) : Permet de changer le statut de certaines mesures, pour transformer une mesure en membre d'une dimension.

La factualisation (*fold*) : Consiste à transformer une dimension en mesure(s) ; cette opération permet de transformer en mesure l'ensemble des paramètres d'une dimension.

La paramétrisation (*unfold*) : Permet de transformer une mesure en paramètre dans une nouvelle dimension.

L'opération Cube : Permet de calculer des sous-totaux et un total final dans le cube (Figure1.5).

1.8.3 Opérations agissant sur la granularité

Les opérations agissant sur la granularité des données analysées, permettent de hiérarchiser la navigation entre les différents niveaux de détail d'une dimension.
Dans la suite nous traitons les deux opérations de ce type :

Le forage vers le haut (*drill-up ou roll-up*) : Permet de représenter les données du cube à un niveau plus haut de granularité en respectant la hiérarchie de la dimension. Nous utilisons une fonction d'agrégation (somme, moyenne, ...), qui est paramétrée, pour indiquer la façon de calculer les données du niveau supérieur à partir de celles du niveau inférieur.

Le forage vers le bas (*drill-down ou roll-down ou scale-down*) : Consiste à représenter les données du cube à un niveau de granularité inférieur, donc sous une forme plus détaillée.

Ces types d'opérations ont besoin d'informations non représentées dans un cube, pour augmenter ou affiner des données, à partir d'une représentation initiale vers une représentation de granularité différente. Le forage vers le haut a besoin de connaître la fonction d'agrégation utilisée tandis que le forage vers le bas nécessite de connaître les données au niveau inférieur.

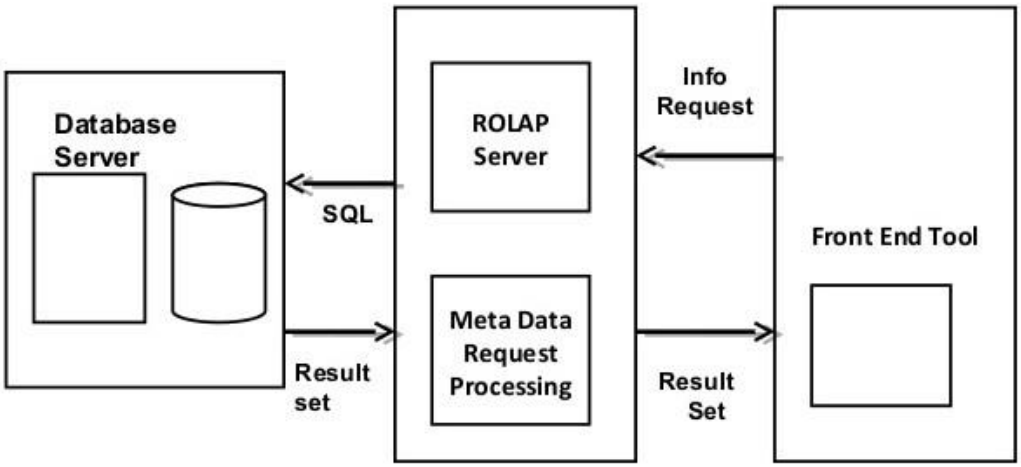
1.9 Serveurs OLAP (On-Line Analytical Processing)

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.

1.9.1 ROLAP (Relational OLAP)

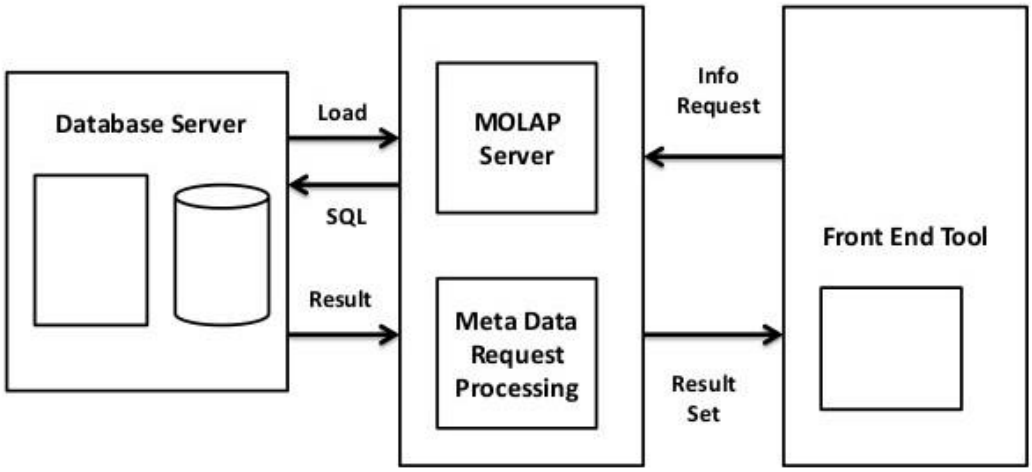
Cette méthode repose sur le fonctionnement des données stockées dans la base de données relationnelle pour présenter la fonction de découpage OLAP. Essentiellement, chaque opération de découpage équivaut à ajouter une clause « WHERE » à l'instruction SQL.[11]



La figure 1.7 Architecture ROLAP.

1.9.2 MOLAP (Multidimensionnel OLAP)

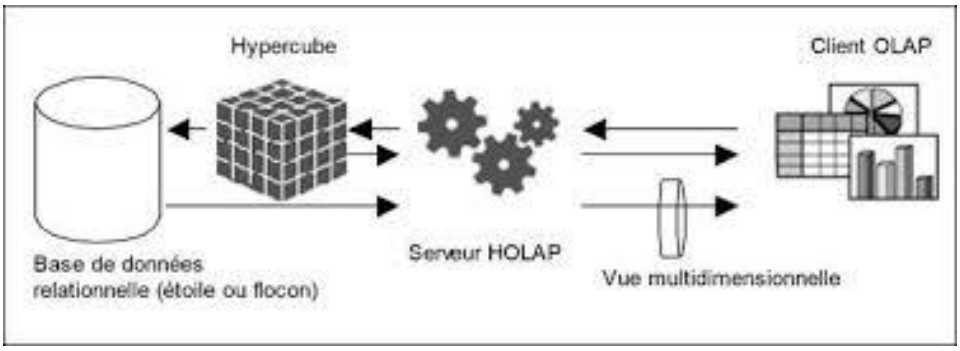
Il s'agit de la méthode d'analyse OLAP la plus traditionnelle. Dans MOLAP, les données sont stockées dans des cubes multidimensionnels. Le stockage n'est pas dans une base de données relationnelle, mais dans un format propriétaire.[11]
La figure 1.7 montre une architecture pour les systèmes MOLAP.



La figure 1.8 Architecture MOLAP.

1.9.3 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.[11]



La figure 1.9 Architecture HOLAP.

1.10 Conclusion

Dans ce chapitre, nous avons traité le sujet des entrepôts de données qui étendent les concepts et la technologie traditionnelle des bases de données pour créer des systèmes d'aide à la décision.

En utilisant l'architecture d'un entrepôt de données, nous avons expliqué les différents composants qu'il intègre, comme les diverses sources, les types de données et les différents outils pour arriver à la visualisation de l'information. Nous avons décrit les différents modelés multidimensionnels pour la construction d'un entrepôt de données, ainsi que les différentes opérations pour la manipulation des données multidimensionnelles.

Dans le chapitre suivant nous décrirons l'implémentation d'entrepôt de données dans le cas de bases de données NOSQL.

Chapitre 2

Les bases de données NoSQL

2.1 Introduction

Un NoSQL faisant à l'origine référence à non SQL ou non relationnel est une base de données qui fournit un mécanisme de stockage et de récupération des données. Ces données sont modélisées par des moyens autres que les relations tabulaires utilisées dans les bases de données relationnelles. De telles bases de données ont vu le jour à la fin des années 1960, mais n'a pas obtenu le surnom de NoSQL jusqu'à un regain de popularité au début du vingt – premier siècle. Les bases de données NoSQL sont utilisées dans les applications Web en temps réel et le Big Data et leur utilisation augmente avec le temps. Les systèmes NoSQL sont également parfois appelés Non seulement SQL pour souligner le fait qu'ils peuvent prendre en charge les langages de requête de type SQL.[12]

Dans ce chapitre, nous analysons aussi bien Caractéristiques de NoSQL. Ce chapitre décrit les différents Types de bases de données NoSQL

2.2 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 ».[12][13]

2.3 Pourquoi NoSQL ?

Le concept de bases de données NoSQL est devenu populaire auprès des géants de l'Internet comme Google, Facebook, Amazon, etc… qui traitent d'énormes volumes de données. Le temps de réponse du système devient lent lorsque vous utilisez RDBMS pour des volumes massifs de données. Pour résoudre ce problème, nous pourrions « mettre à l'échelle » nos systèmes en améliorant notre matériel existant. Ce processus est coûteux.

L'alternative pour ce problème est de distribuer la charge de base de données sur plusieurs hôtes chaque fois que la charge augmente. Cette méthode est connue sous le nom de « mise à l'échelle (scaling out) ».[14]

2.4 NoSQL versus SQL : quelles différences ?

- 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.

2.5 Le théorème CAP

Le théorème de CAP est également appelé théorème du brasseur. Il indique qu'il est impossible pour un magasin de données distribués d'offrir plus de deux garanties sur trois [14]

Cohérence (consistency) : Les données doivent rester cohérentes même après l'exécution d'une opération. Cela signifie qu'une fois les données écrites, toute demande de lecture future doit contenir ces données. Par exemple, après la mise à jour de l'état de la commande, tous les clients doivent être en mesure de voir les mêmes données

Disponibilité (availability) : La base de données doit toujours être disponible et réactive. Il ne devrait pas avoir de temps d'arrêt.

Tolérance à la partition (partition tolerance) : La tolérance de partition signifie que le système doit continuer à fonctionner même si la communication entre les serveurs n'est pas stable. Par exemple, les serveurs peuvent être divisés en plusieurs groupes qui peuvent ne pas communiquer entre eux. Ici, si une partie de la base de données n'est pas disponible, d'autres parties ne sont toujours pas affectées.

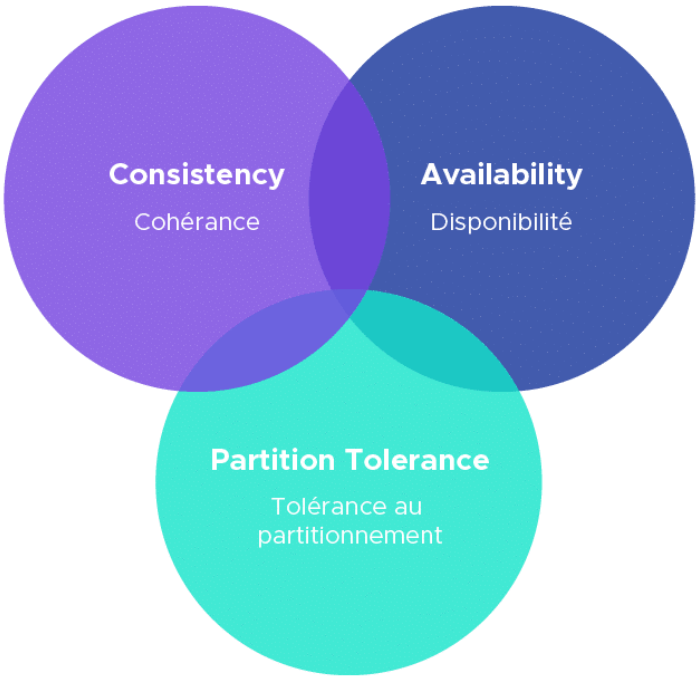


Figure 2.1 Le théorème CAP

2.6 Caractéristiques de NoSQL

2.6.1 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 [14]

2.6.2 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

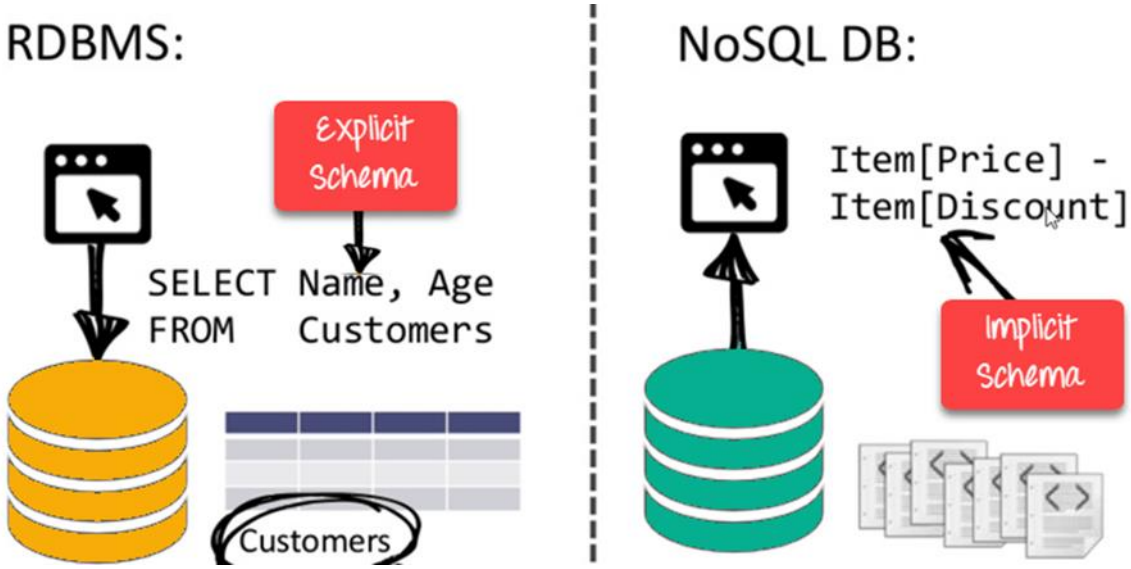


Figure 2.2 NoSQL est sans schéma

2.6.3 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

2.6.4 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

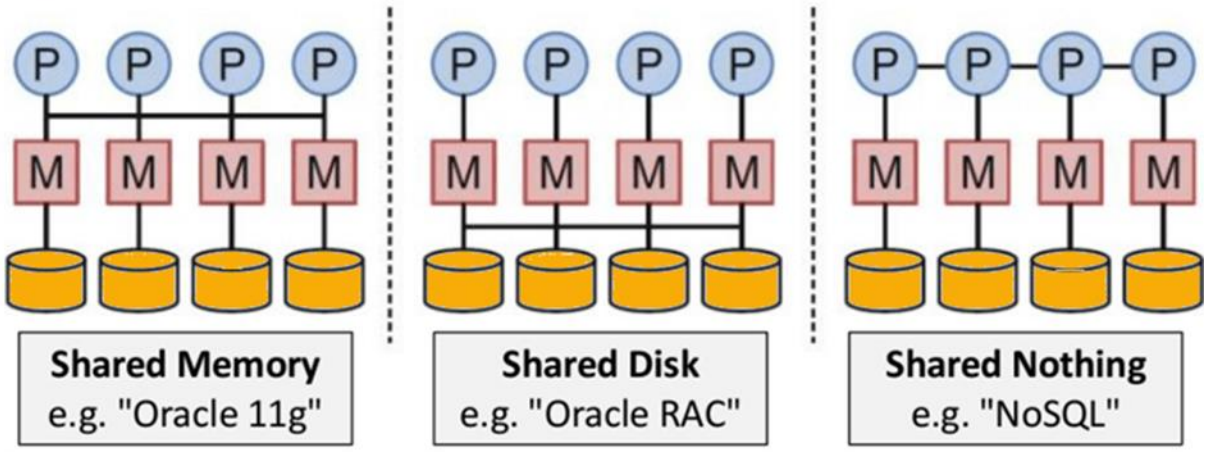


Figure 2.3 NoSQL est rien partagé (Shared Nothing).

2.7 Types de bases de données NoSQL

La plupart des bases de données hautement performantes non relationnelles, parfois appelées « pas seulement SQL », peuvent aussi gérer des données hautement structurées. Elles ne se limitent pas à des modèles de données fixes, tels que les bases de données relationnelles (SQL).[14] [

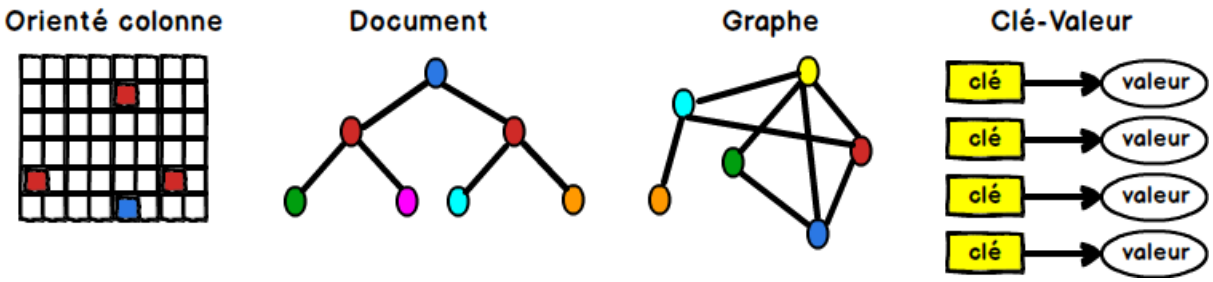


Figure 2.4 Types de bases de données NoSQL

Les quatre principaux types de bases de données NoSQL sont les suivants :

2.7.1 Clé-valeur

Clé-valeur stocke des paires de clés et de valeurs à l'aide d'une table de hachage. Les types clé-valeur sont particulièrement adaptés lorsqu'une clé est connue et que la valeur associée à la clé est inconnue.

2.7.2 Basé sur des colonnes

Les bases de données en colonnes, en colonnes larges ou en familles de colonnes stockent efficacement les données et interrogent les lignes de données éparses, et offrent la possibilité d'interroger les colonnes spécifiques d'une base de données.

2.7.3 Orienté vers les documents

Les bases de données de documents étendent le concept de base de données clé-valeur en organisant des documents entiers dans des groupes appelés collections. Elles prennent en

charge les paires clé-valeur imbriquées et autorisent les requêtes sur tous les attributs d'un document.

2.7.4 Graphique-basé

Les bases de données graphes utilisent un modèle basé sur les nœuds et les bords pour représenter les données interconnectées (relations entre membres d'un réseau social, par exemple), et offrent un stockage et une navigation facilités en présence de relations complexes.

2.8 Les avantages et les Inconvénients de NoSQL

Les bases de données NoSQL par leur conception sont différentes des bases relationnelles classiques. Elles répondent également à d'autres problématiques et besoins. Voici quelques-uns des avantages et inconvénients :

Avantages :

Rapidité : Le modèle NoSQL ne suit pas le modèle relationnel. Pas de schéma de bases avec les contraintes sur les champs. Cela apporte de la flexibilité dans la gestion des données et de la rapidité.

La scalabilité est facilitée : Elles offrent un niveau de scalabilité excellent sur des environnements clustérisés

Moins cher : Les solutions de gestion de base de données non relationnelles sont pour la plupart en Open Source. Les entreprises qui l'utilisent bénéficient d'un bon rapport qualité/prix

Inconvénients :

Support limité : due à la jeunesse des bases de données NoSQL, le support de la communauté est parfois limité.

Manque de standardisation : Pas de langage « NoSQL" standard sur les différentes bases de données.

Interopérabilité : La passage d'une base de données NoSQL vers une autre n'est pas transparente pour une application.

2.9 Conclusion

Dans ce chapitre en a vu la notion de l'approche NoSQL nous avons expliqué les différents types et la déférence entre SQL et NOSQL. On a présenté aussi les Avantages et les Inconvénients de ce dernier.

Chapitre 3

Modélisation de la solution décisionnelle

3.1 Introduction

La modélisation de notre application d'aide à la décision repose sur deux parties essentielles. La première partie concerne la modélisation de l'application SOLVE et son modèle de données. La deuxième partie concerne la conception de notre entrepôt de donnée en se basant sur l'analyse des questions et réponses sur un large choix de thèmes concernant différents domaines.

3.2 Présentation de l'application « SOLVE »

L'idée de notre application web est inspirée de site stack overflow, on l'a nommée solve.

Cette application est une communauté de questions et réponses en ligne passionnante qui vise à révolutionner la façon dont les questions sont répondues en ligne. Que vous ayez besoin de connaître la réponse à une question brûlante ou que vous soyez débordé de réponses et d'expériences que vous avez hâte de partager, Solve vous fournira les réponses que vous recherchez.

3.3 Création du modèle de donnée

3.3.1 Comprendre le flux de travail de SOLVE

Notre choix de base de données NoSQL orienté colonne étant CASSANDRA qui a un modèle de données spécifique.

En effet, avec Cassandra, plutôt que de commencer par le modèle de données, la meilleure pratique consiste à commencer par le flux de travail de l'application. Cette approche est appelée « **conception axée sur la recherche d'abord** » [15], qui consiste à construire notre modèle de données en fonction des types des requêtes que la base de données devra prendre en charge. La séquence des étapes du flux de travail est importante car elle nous aide à déterminer deux informations qui sont nécessaires pour soutenir les requêtes ultérieures, impactant la conception de la table. Ces informations étant « **user_id** » et « **post_id** ».

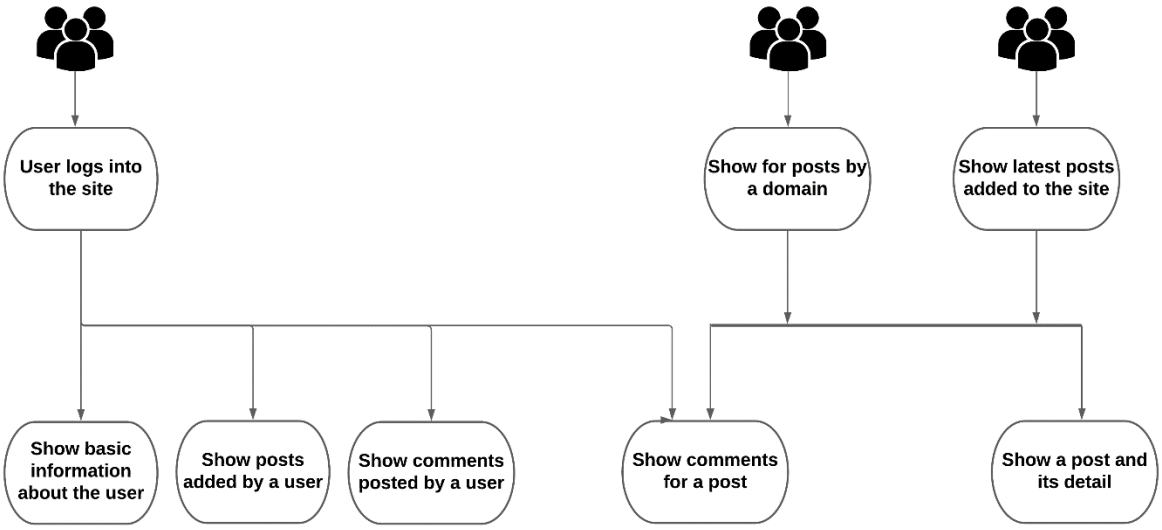


Figure 3.1 Diagramme du flux de travail de l'application

3.3.2 Modélisation les requêtes requises par SOLVE

Adopter une approche axée sur la recherche d'abord signifie non seulement réfléchir à la séquence des tâches requises, mais aussi aider à déterminer quelles données seront requises et quand.

Le diagramme des relations d'entités ci-dessous montre les entités (utilisateurs, publications et commentaires), les éléments de données et les relations requises par l'application SOLVE. Une fois le flux de travail de l'application et les objets de données clés identifiés, il est possible de déterminer les requêtes que l'application doit prendre en charge.

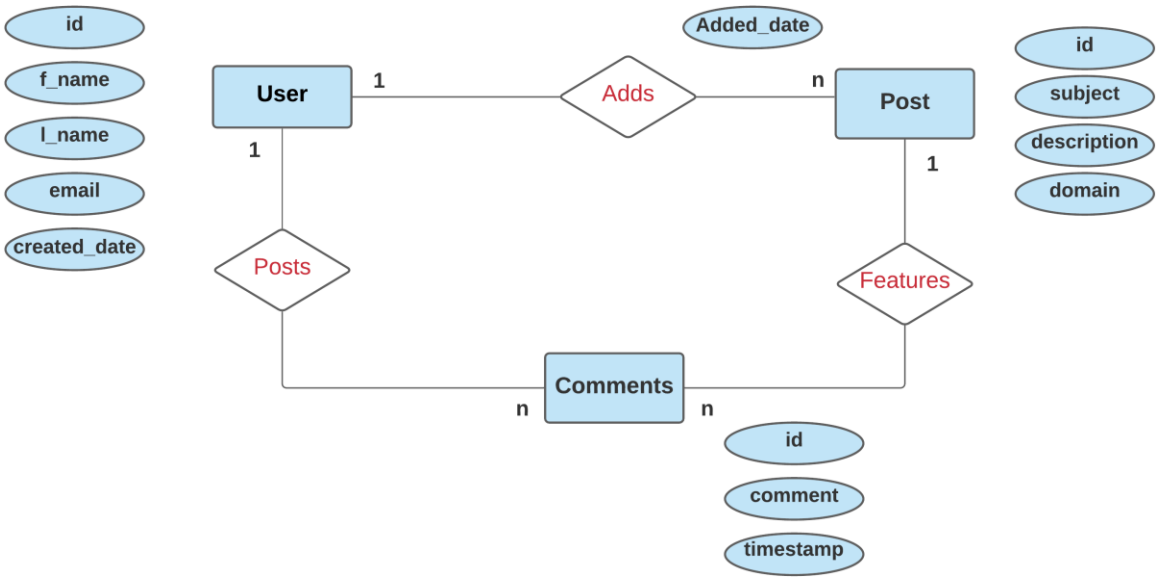


Figure 3.2 Diagramme Entité-Relationship

3.3.3 Concevoir les tableaux

Dans Cassandra, les tableaux peuvent être regroupés en deux catégories distinctes :

- **Tables avec des partitions à une ligne**

Ces types de tables ont des clés primaires qui sont aussi des clés de partition [15]. Ils sont utilisés pour stocker les entités et sont généralement normalisés. Il est recommandé que les tableaux soient basés sur l'entité pour une référence facile (e, g.., Users).

```
class Users(DjangoCassandraModel):
    user_id = columns.UUID(primary_key=True)
    f_name  = columns.Text()
    l_name  = columns.Text()
    email    = columns.Text()
    created_date = columns.DateTime()
```

Figure 3.3 Table avec des partions à une ligne

- **Tables avec des partitions multi lignes**

Ces types de tables ont des clés primaires composées de clés de partition et de clustering [15]. Ils sont utilisés pour stocker les relations et les entités liées. On rappelle que Cassandra ne supporte pas les jointures, donc nous allons structurer les tables pour supporter les requêtes qui se rapportent aux éléments multiples data.

```
class Latest_post(DjangoCassandraModel):
    yyyymmdd = columns.Text(primary_key=True)
    added_date = columns.DateTime(primary_key=True , clustering_order ="DESC")
    post_id = columns.UUID(primary_key=True , clustering_order ="ASC")
    user_id = columns.UUID()
    subject  = columns.Text()
    description = columns.Text()
```

Figure 3.4 Table avec des partitions multi lignes

Des colonnes de regroupement supplémentaires (added_date et post_id) spécifient comment les enregistrements sont regroupés et classés dans chaque partition. En concevant la table de cette façon, les requêtes ne toucheront qu'une partition pour la journée en cours et éventuellement une autre partition pour la veille. Ce niveau d'optimisation et d'efficacité

permet d'expliquer comment Cassandra peut prendre en charge des applications avec un nombre énorme de requêtes sur de très grands ensembles de données.

3.3.4 Utilisation d'un diagramme de Chebotko pour représenter notre schéma

Un bon outil pour cartographier le modèle de données qui prend en charge une application est connu comme un diagramme de Chebotko [15]. Conçu pour développer les modèles de données logiques et physiques nécessaires pour soutenir l'application, le diagramme de Chebotko capture le schéma de la base de données, montrant les noms de table, les colonnes clés de partition (K), les colonnes clés de regroupement (C) et leur ordre, et des colonnes régulières avec des types de données. Les tableaux sont organisés en fonction du flux de travail de l'application pour prendre en charge des étapes de flux de travail et des requêtes d'application spécifiques.

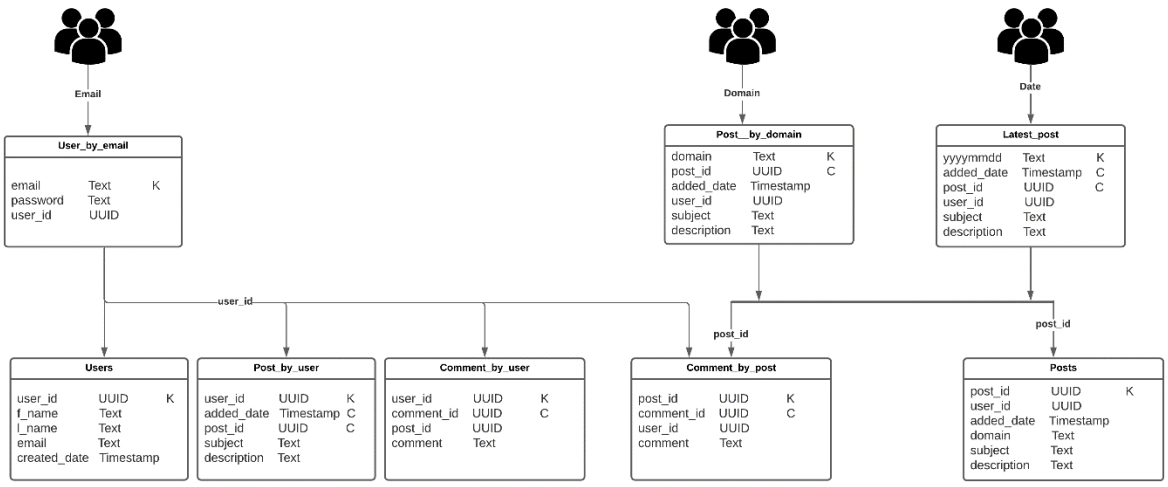


Figure 3.5 Diagramme Chebotko

3.4 Sources de donnée

Les sources de donnée utilisées dans notre création de l'entrepôt de donnée sont des Fichiers CSV qui est récupérer dans internet via le site de génération de données (www.generatedata.com).

Notre base de données est créée à partir du SGBD Cassandra avec Framework Django et l'interface NoSQL Manager for Cassandra.

3.5 Schéma en constellation SOLVE

Pour la modélisation de l'entrepôt de données, nous avons opté pour le schéma en constellation. Ce schéma est composé par deux schémas en étoile (Post_fact et Comment_fact).

Ces schémas sont simples permettant l'analyse de différentes mesures des deux tables des faits (Post_fact et Comment_fact) selon différentes dimensions qui représentent les axes d'analyse.

3.5.1 Schéma en étoile Post_fact

Notre schéma se compose de dimensions et une table de fait, voir Figure 3.6

1-Dim_ Domain
2- Dim_ User
3- Dim_ Time
4- Dim_ Localisation

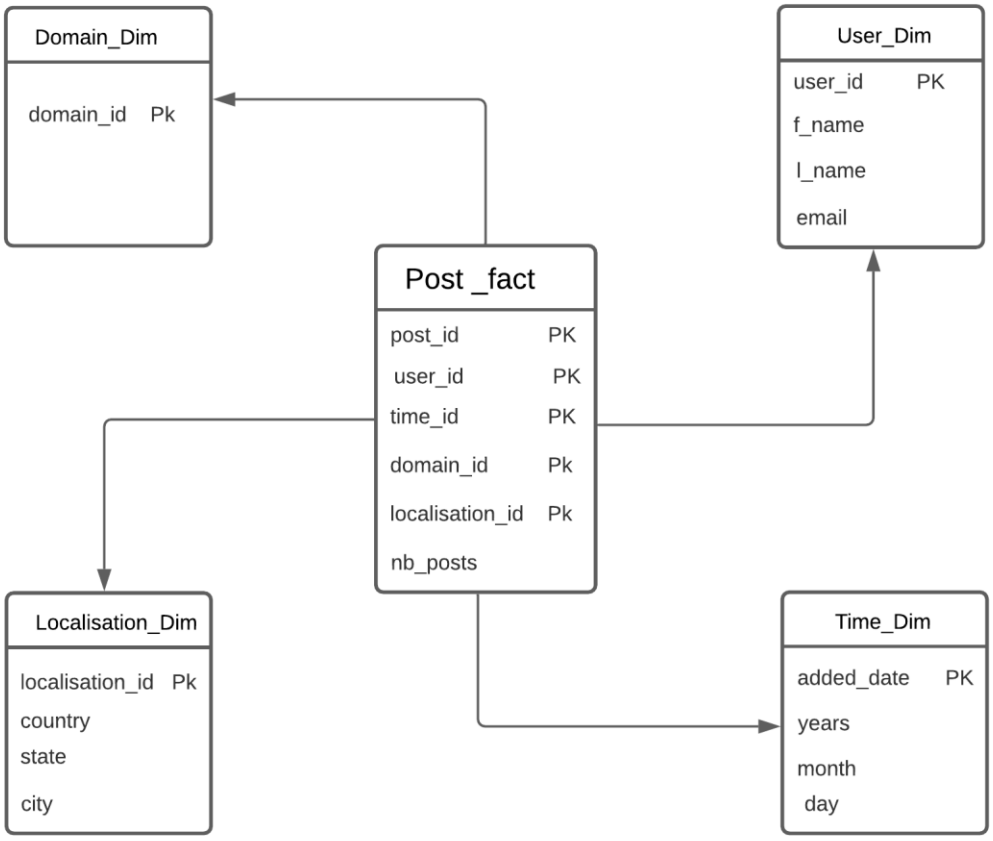


Figure 3.6 Schéma en étoile Post_fact

3.5.2 Schéma en étoile Comment_fact

Notre schéma se compose de dimensions et une table de fait, voir Figure 3.7

1- Dim_ User

2- Dim_ Time

3- Dim_ Localisation

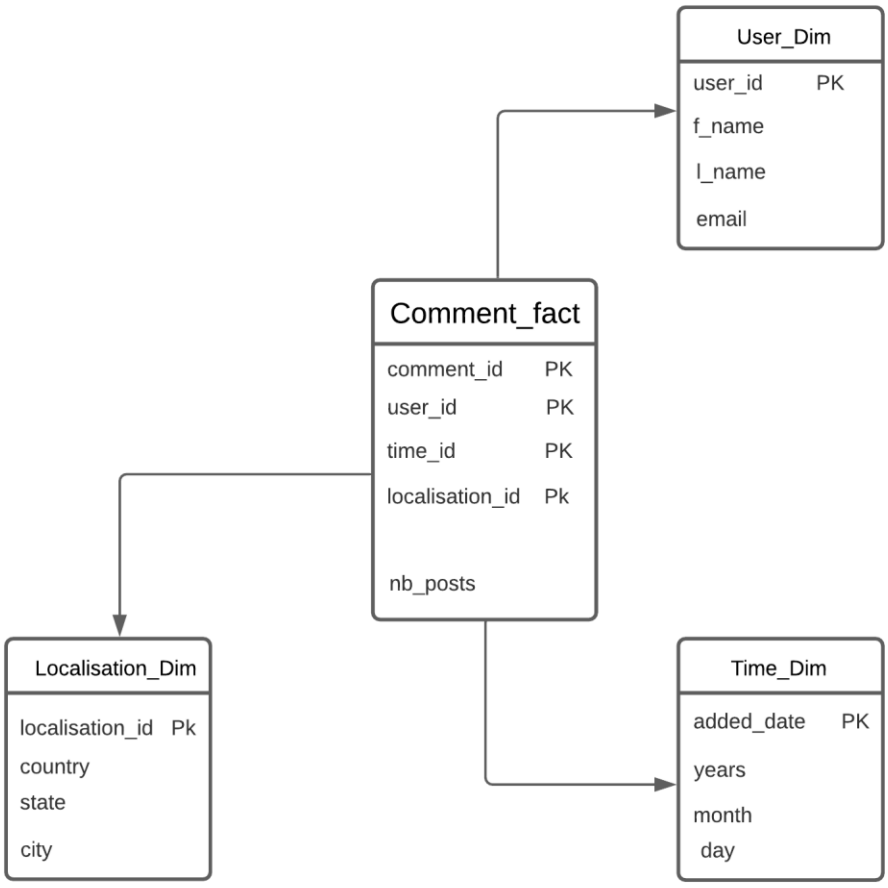


Figure 3.7 Schéma en étoile Comment_fact

3.5.3 Schéma en constellation

Les deux schémas que nous venons de présenter (publications et commentaires) peuvent être considérés comme deux Data Marts, deux compartiments du Data Warehouse : un Data Mart pour les statuts et un Data Mart pour les commentaires. Chacun d'entre eux ne contient qu'une seule table des faits entourée de quelques tables de dimensions

Voici comment se présente la modélisation en constellation appliquée aux deux schémas en étoile étudiée plus haut :

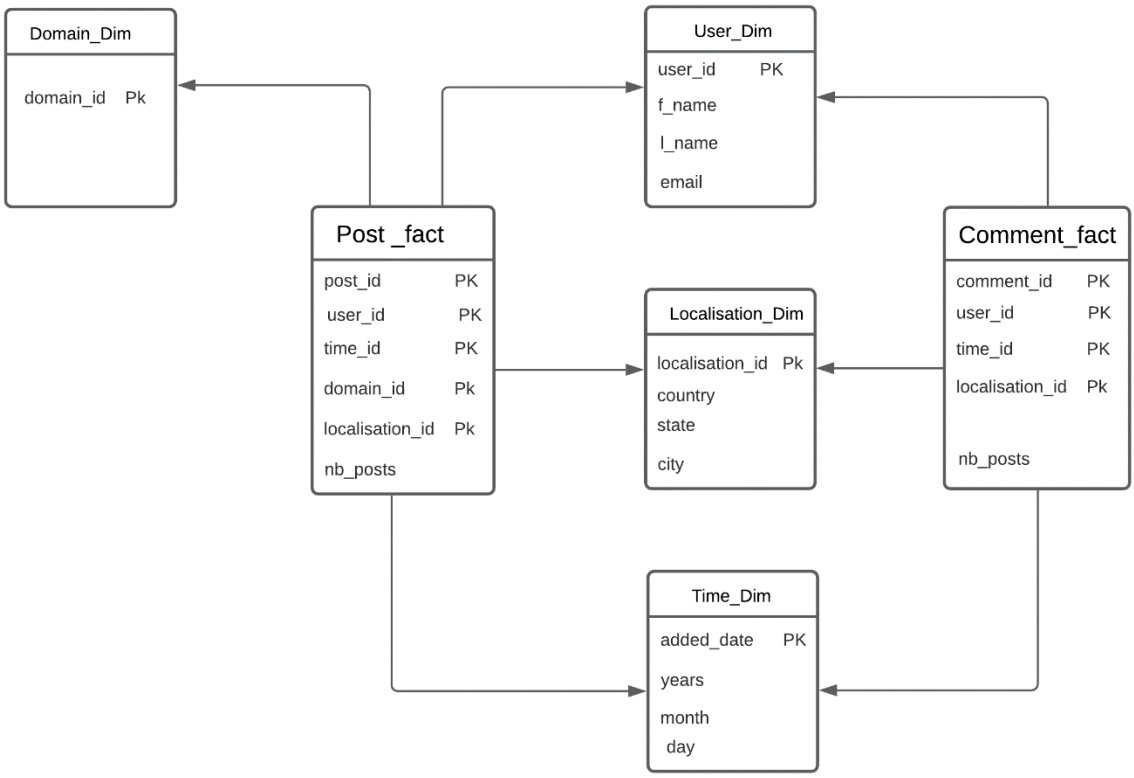


Figure 3.8 Schéma en constellation

3.6 Conclusion

Dans ce chapitre nous avons présenté notre application SOLVE, et nous avons traité les étapes pour crée un modèle de donnée dans NOSQL oriente colonne, depuis on a présenté le schéma en constellation de notre modèle.

Dans le chapitre suivant nous expliquons les différentes étapes d'implémentation de la base de données et l'analyse des données ainsi que les outils utilisés pour modéliser notre modèle.

Chapitre 4

Implémentation

4.1 Introduction

Ce chapitre est consacré à la présentation des modèles de données ainsi que les outils qui ont permis la réalisation de ses différents modèles.

4.2 Présentation de Cassandra et les outils nécessaires pour la BDD

Apache Cassandra est une base de données distribuée NoSQL open source orientée colonnes qui fait confiance à des milliers d'entreprises pour son évolutivité et sa haute disponibilité sans compromettre les performances. L'évolutivité linéaire et la tolérance éprouvée aux pannes sur le matériel de base ou l'infrastructure cloud en font la plateforme idéale pour les données essentielles à la mission [16]

4.2.1 Caractéristiques de Cassandra

Nous détaillons dans cette section les principales caractéristiques de la base de données NoSQL Cassandra.

- Il s'agit d'une base de données à colonnes.
- Elle est très cohérente, tolérante aux pannes et évolutive.
- Elle a été créée pour Facebook et a ensuite a été open source.
- Le modèle de données est basé sur Google Bigtable.

- La conception distribuée est basée sur Amazon Dynamo.

4.2.2 Installation Cassandra

Apache Cassandra nécessite Java 8 pour fonctionner sur un système Windows. De plus, le Shell en ligne de commande de Cassandra (**cqlsh**) dépend de Python 2.7 pour fonctionner correctement.

Pour lancer le serveur Cassandra, on exécute la commande suivante **cassandra.bat**

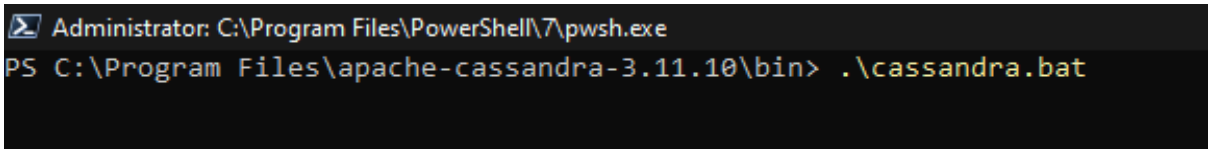


Figure 4.1 La commande cassandra.bat

Après lancement serveur voir Figure 4.2

Figure 4.2 Démarrage serveur

4.2.3 Présentation Visual studio code

Visual Studio Code est un éditeur de code source léger mais puissant. On possède souvent une solution à laquelle nous sommes habitués, qui nous convient la majorité du temps et on a peur de se retrouver perdu et de perdre par la même occasion en productivité. Pourtant, Visual Studio Code rassure la majorité des nouveaux utilisateurs dès les premières heures d'utilisation [18].

Nous utilisons Visual studio code pour faciliter notre travail.

Figure 4.3 Installation Visual studio code

4.2.4 Présentation de Django

Django est un Framework Web Python de haut niveau qui encourage un développement rapide et un design propre. Construit par des développeurs expérimentés, il prend soin de la plupart des tracas du développement Web, de sorte que vous pouvez vous concentrer sur l'écriture de votre application sans avoir besoin de réinventer la roue [17].

Django peut être installé facilement avec la commande pip.

Dans l'invite de commande, on exécute la commande suivante : **pip install Django**. Cette commande télécharge et installe Django, voir Figure 4.4

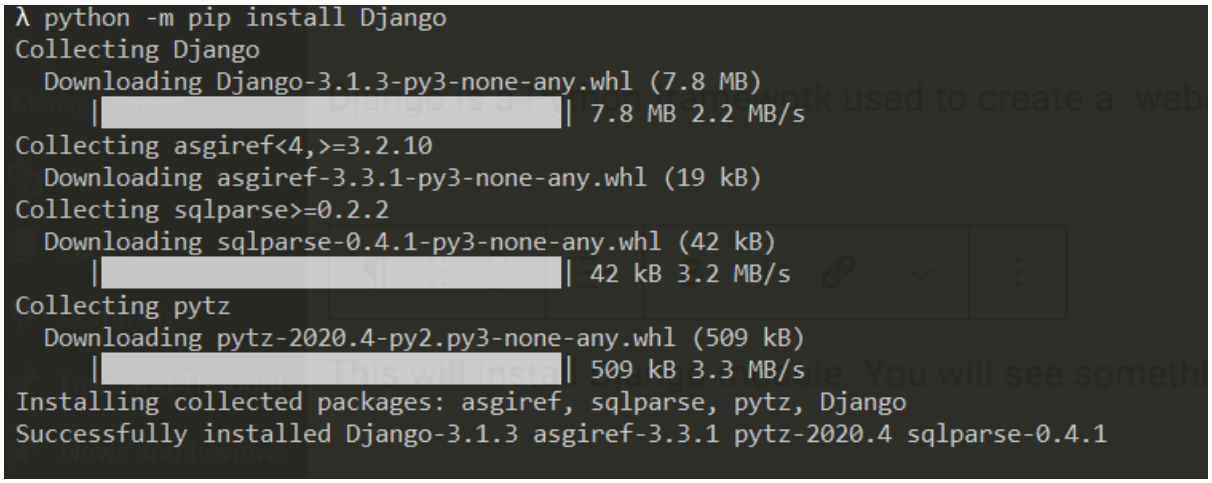


Figure 4.4 Téléchargement et installation Django

- ## Création projet solve

Pour crée projet sur Django, on exécute la commande suivante :

django-admin startproject solve, solve est le nom de projet

Solve/ composé des fichiers suivants : **(__init__.py ; settings.py; urls.py ; asgi.py ; wsgi.py)**

- ## Création application solve_model

Pour crée application sur Django, on exécute la commande suivante :

django-admin startapp solve_model, solve_model est le nom de l'application

solve_model/ composé des fichiers suivants : **(__init__.py; admin.py; apps.py; migrations/; __init__.py; models.py; tests.py; views.py)**

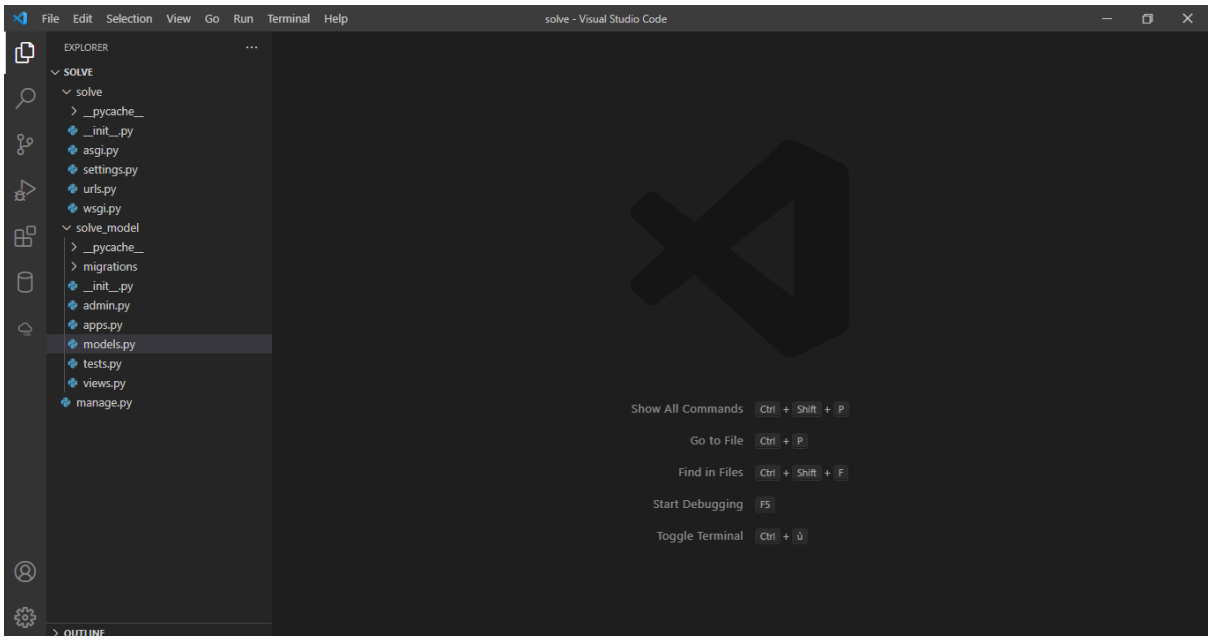


Figure 4.5 Projet SOLVE dans Visual studio code

4.2.5 Configuration Cassandra avec Django

Étape 01 installation recommandée par commande **pip install django-cassandre-engine,** voir Figure 4.6

```
Collecting django-cassandra-engine
  Using cached django_cassandra_engine-1.6.2-py3-none-any.whl
Collecting Django>=2.0
  Downloading Django-3.2.4-py3-none-any.whl (7.9 MB)
     |████████████████████████████████| 7.9 MB 312 kB/s
Requirement already satisfied: six>=1.6 in c:\users\amine\appdata\local\programs\python\python39\lib\site-packages (from django-cassandra-engine) (1.16.0)
Collecting cassandra-driver>=3.25
  Using cached cassandra_driver-3.25.0-cp39-cp39-win_amd64.whl
Collecting geomet<0.3,>=0.1
  Using cached geomet-0.2.1.post1-py3-none-any.whl (18 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Collecting asgiref<4,>=3.3.2
  Using cached asgiref-3.3.4-py3-none-any.whl (22 kB)
Collecting pytz
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting click
  Using cached click-8.0.1-py3-none-any.whl (97 kB)
Collecting colorama
  Using cached colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Installing collected packages: colorama, click, sqlparse, pytz, geomet, asgiref, Django, cassandra-driver, django-cassandra-engine
Successfully installed Django-3.2.4 asgiref-3.3.4 cassandra-driver-3.25.0 click-8.0.1 colorama-0.4.4 django-cassandra-engine-1.6.2 geomet-0.2.1.post1 pytz-2021.1 sqlpa
se-0.4.1
WARNING: You are using pip version 21.1.1; however, version 21.1.2 is available.
You should consider upgrading via the 'c:\users\amine\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.
```

Figure 4.6 Installation django-cassandra-engine

Étape 02 Nous avons ajouté **django_cassandra_engine** dans le fichier **settings.py**

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django_cassandra_engine',
    'solve_model',
]
```

Figure 4.7 Augmentation django-cassandra-engine

Étape 03 Nous avons changé **DATABASES** dans le fichier **settings.py** , voir Figure 4.8

```
DATABASES = {
    'default': {
        'ENGINE': 'django_cassandra_engine',
        'NAME': 'solve_keyspace',
        'TEST_NAME': 'solve',
        'HOST': '127.0.0.1',
        'OPTIONS': {
            'replication': {
                'strategy_class': 'SimpleStrategy',
                'replication_factor': 1
            }
        }
    }
}
```

Figure 4.8 Configuration DATABASES

4.2.6 Création les Tables de projet SOLVE

On pourrait faire des tables directement au niveau **cqlsh** ou **NoSQL Manager for Cassandra**, Mais nous avons choisi Django pour compléter notre site en future.

Voilà les tables :

User_by_ email

Users

Post_by_user

Comments_by_user

Post_by_domain

Latest_post

Posts

Comments_by_post

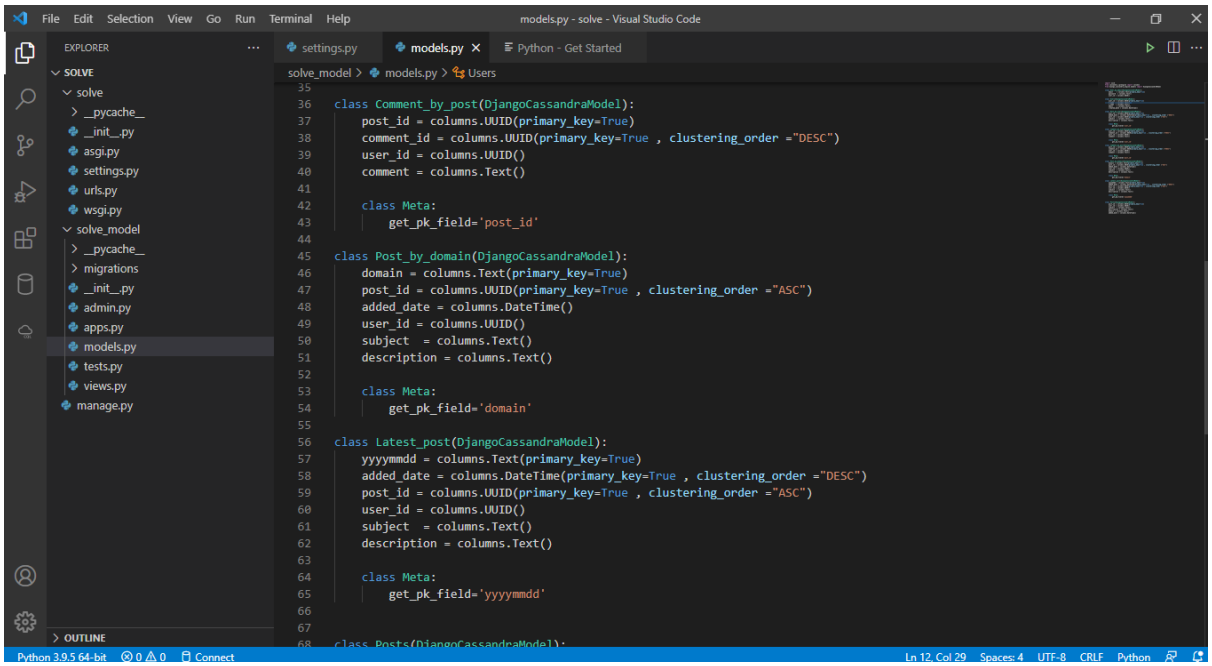


Figure 4.9 Création les tables à partir django

A la fin, on exécute la commande suivante **python manage.py sync_cassandra** , Pour la migration de nos modèles

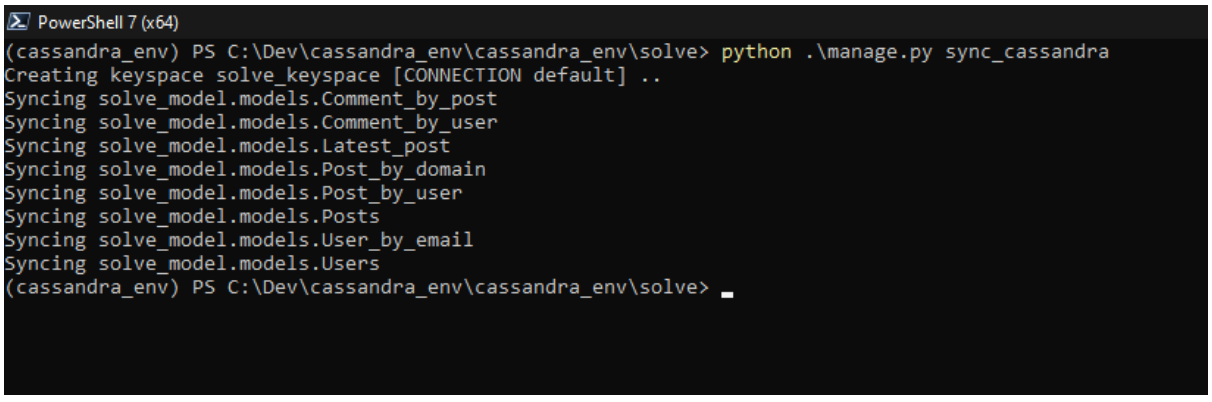


Figure 4.10 Migration les tables

Nous pouvons voir les tables dans NoSQL Manager for Cassandra après migration présenté dans la section suivante.

4.2.7 Présentation de NoSQL Manager for Cassandra

Client de bureau pour la gestion, le contrôle et le développement de la base de données Cassandra.

La Connexion NoSQL Manager for Cassandra avec serveur Cassandra par Host **127.0.0.1**

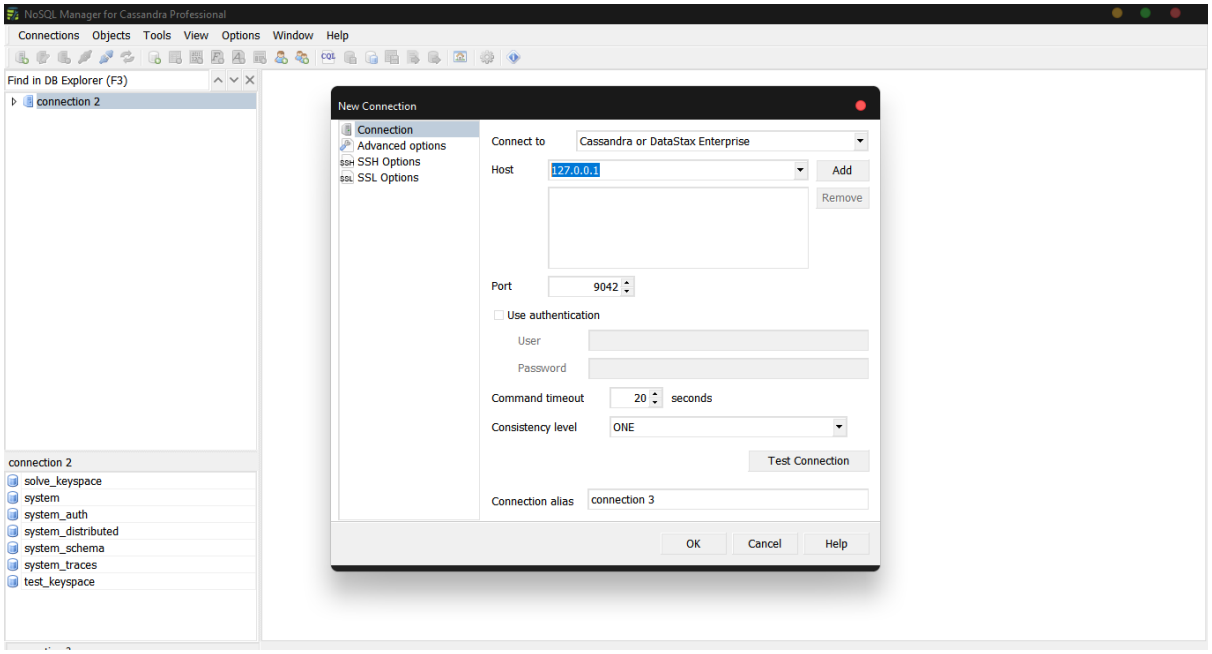


Figure 4.11 Connexion NoSQL Manager for Cassandra avec serveur Cassandra

Nous pouvons voir les tables après migration dans NoSQL Manager for Cassandra comme le montre la Figure 4.12

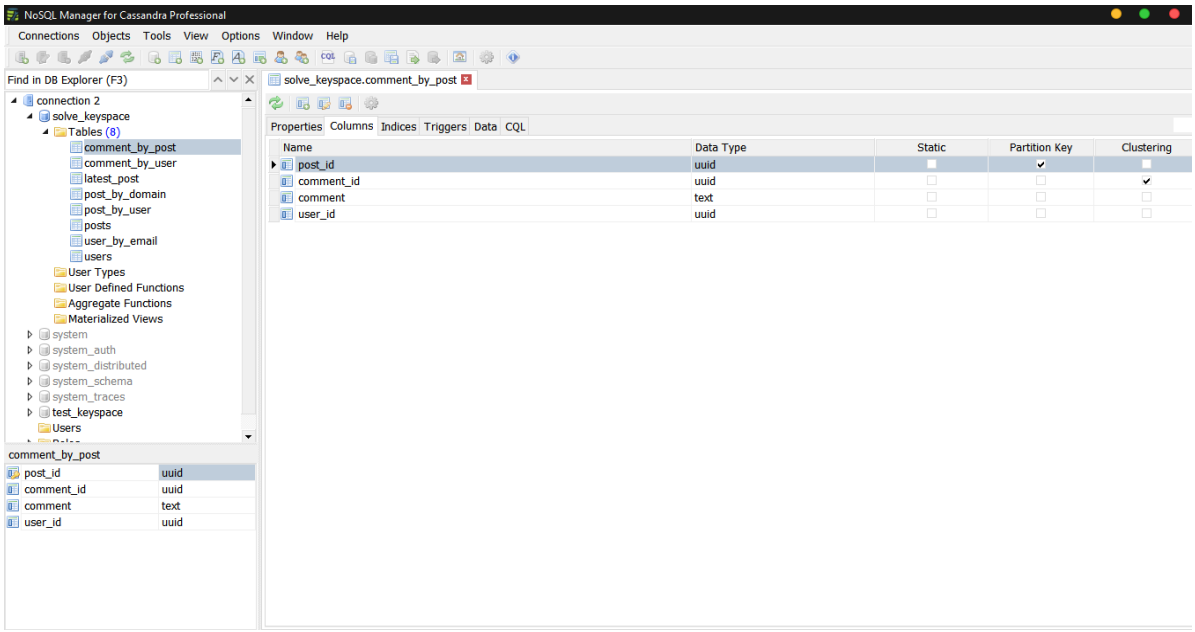


Figure 4.12 Les tables dans Manager for Cassandra

Pour remplir notre table, Nous importons fichier csv dans NoSQL Manager for Cassandra

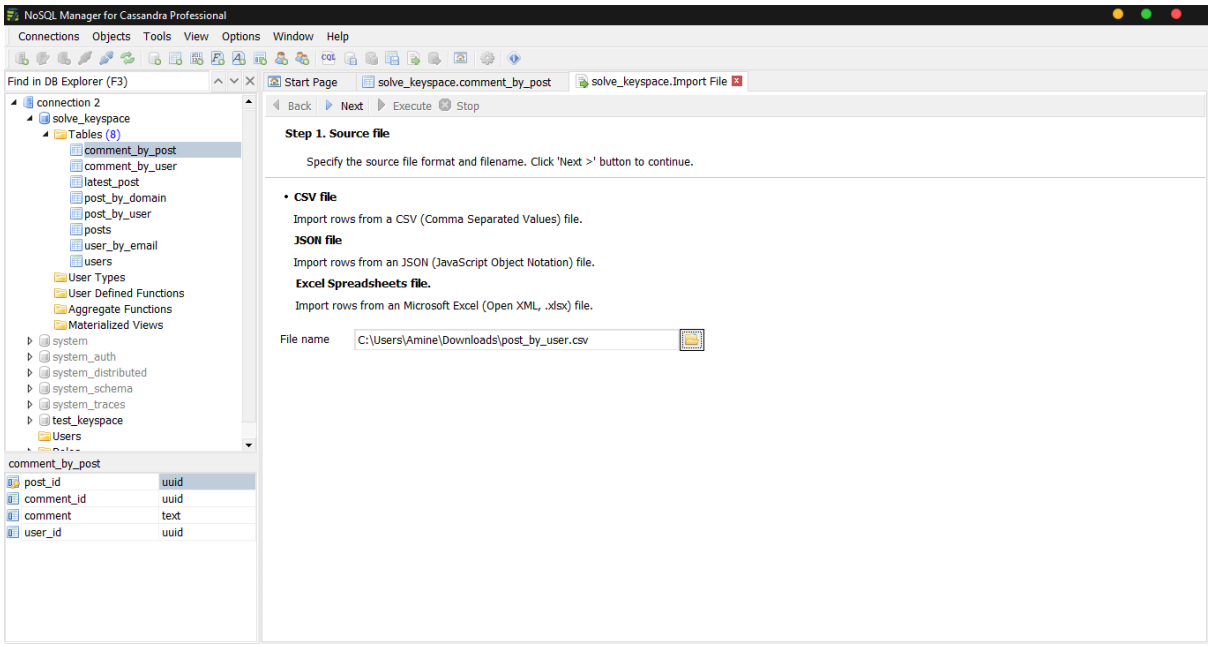


Figure 4.13 Importation fichier csv dans NoSQL Manager for Cassandra

La Figure 4.14 montre la table par exemple comment_by_post après emballage

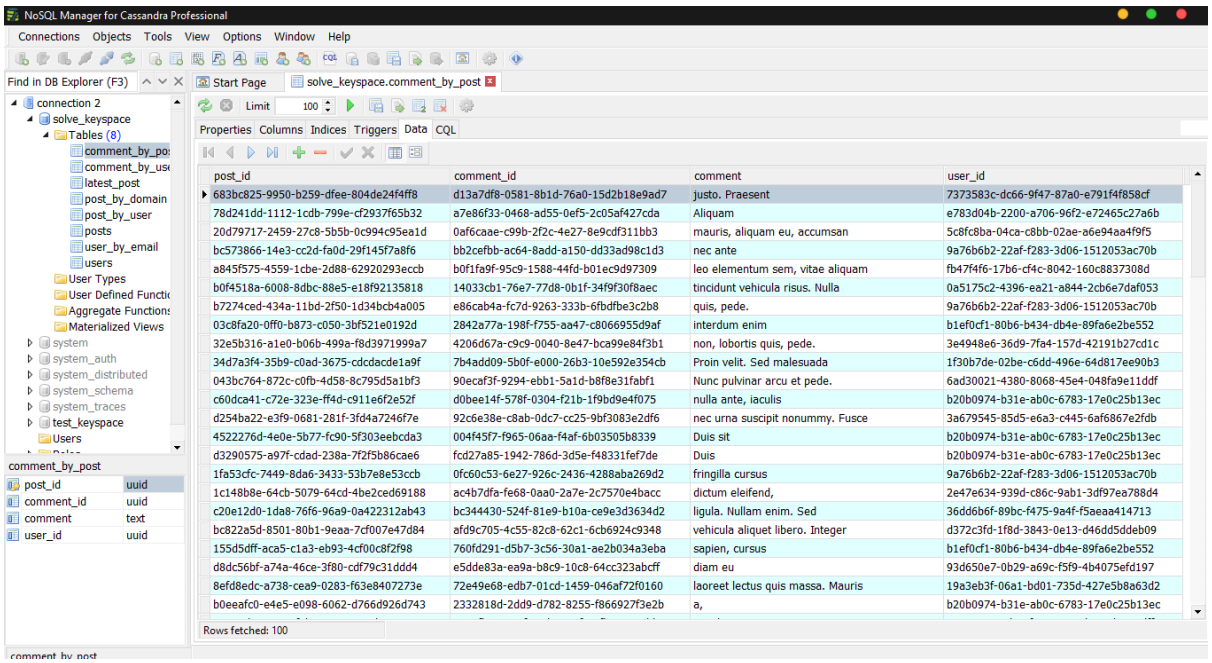


Figure 4.14 La table après l'importation

4.3 Analyse multidimensionnelle

L'objectif final de ce travail étant de posséder un outil d'analyse de données générées à partir d'un site web. Pour atteindre cet objectif nous allons exploiter la base de données implémentée sous Cassandra et alimenté grâce au site www.generatedata.com.

Nous allons par la suite montrer comment utiliser un outil d'analyse « Tableau » pour connecter aux tables de données pour réaliser logiquement les deux schémas en étoile.

4.3.1 Présentation de l'outil tableau

Tableau est un outil d'analyse qui aide les analystes à voir et à comprendre les données. C'est une plateforme d'analyse visuelle transforme la façon dont les gens utilisent les données pour résoudre des problèmes.

Après avoir sélectionné le type de base de données et l'avoir connectée à Cassandra, Voilà notre modèle de connexion voir Figure 4.15

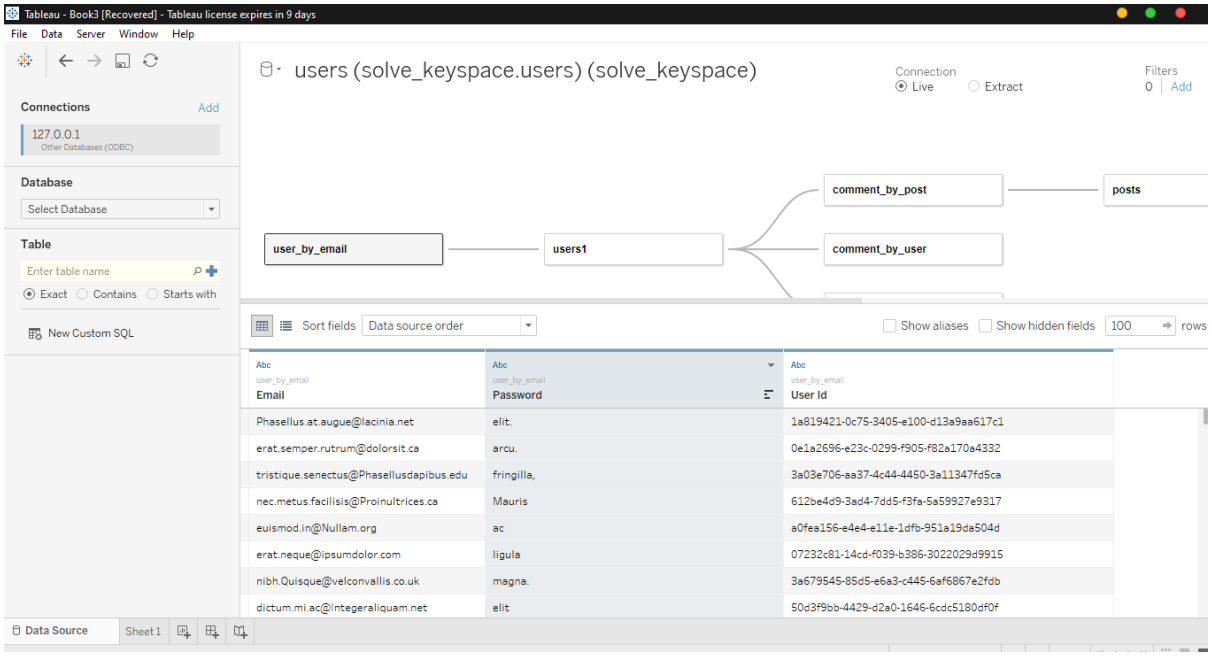


Figure 4.15 Les tables dans tableau

4.3.2 Exemple de requêtes analytiques

Requête 1

« Nombre des publications par Domain »

La Figure 4.16 montre le résultat de cette requête :

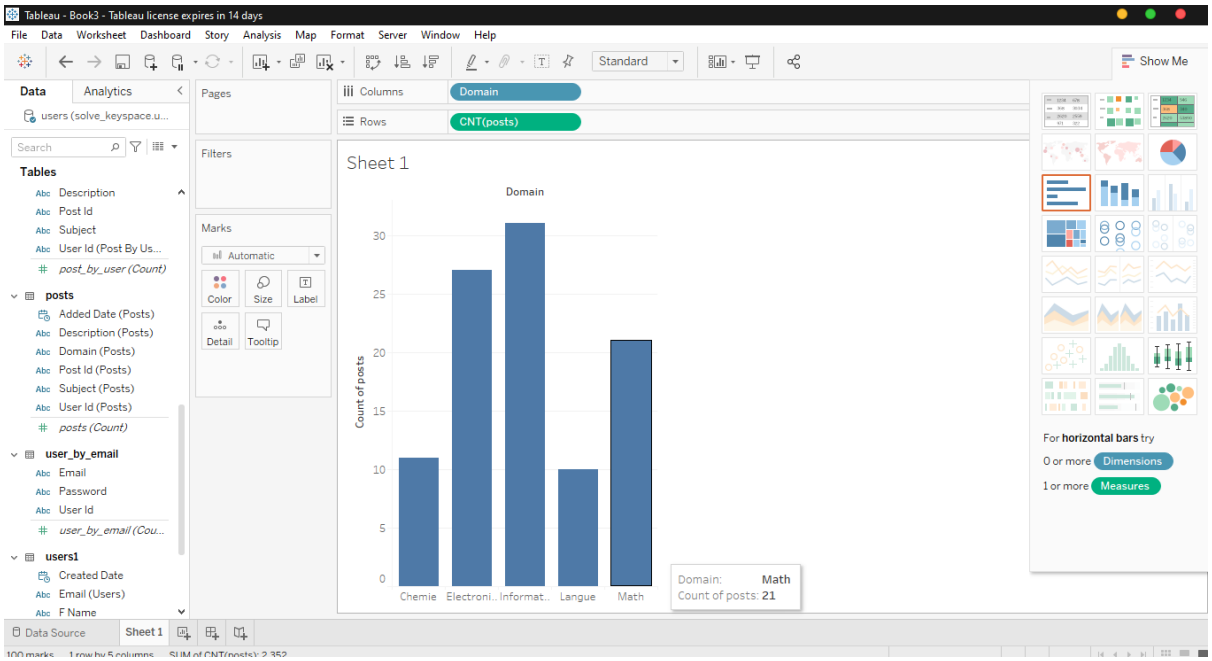


Figure 4.16 Graph nombre des publications par Domain

Requête 2

« Nombre des publications par les noms utilisateurs(f_name) »

La Figure 4.17 montre le résultat de cette requête :

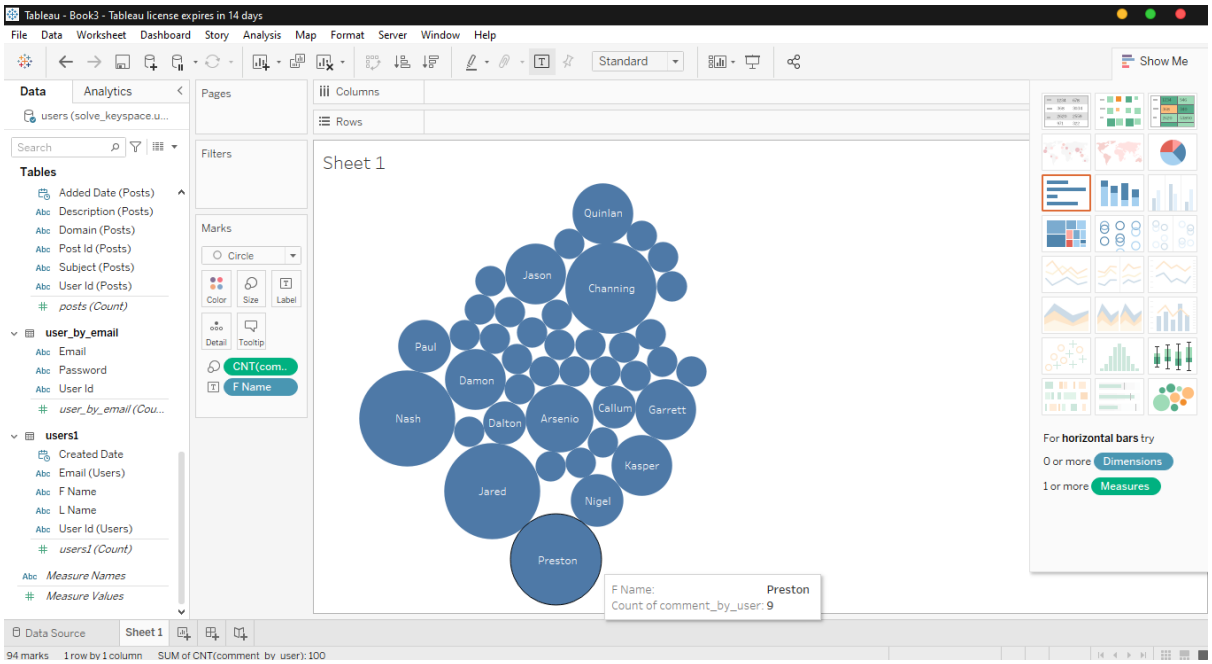


Figure 4.17 Graph nombre des publications par les noms utilisateurs(f_name)

Requête 3

Si nous considérons un exemple où on aurait représenté dans un modèle multidimensionnel (cube) : « nombre des publications sous 2 axes à savoir : Domaine et les noms utilisateurs (f_name) »

La Figure 4.18 montre le résultat de requête « Nombre des publications par Domaine et les noms utilisateurs (f_name) »

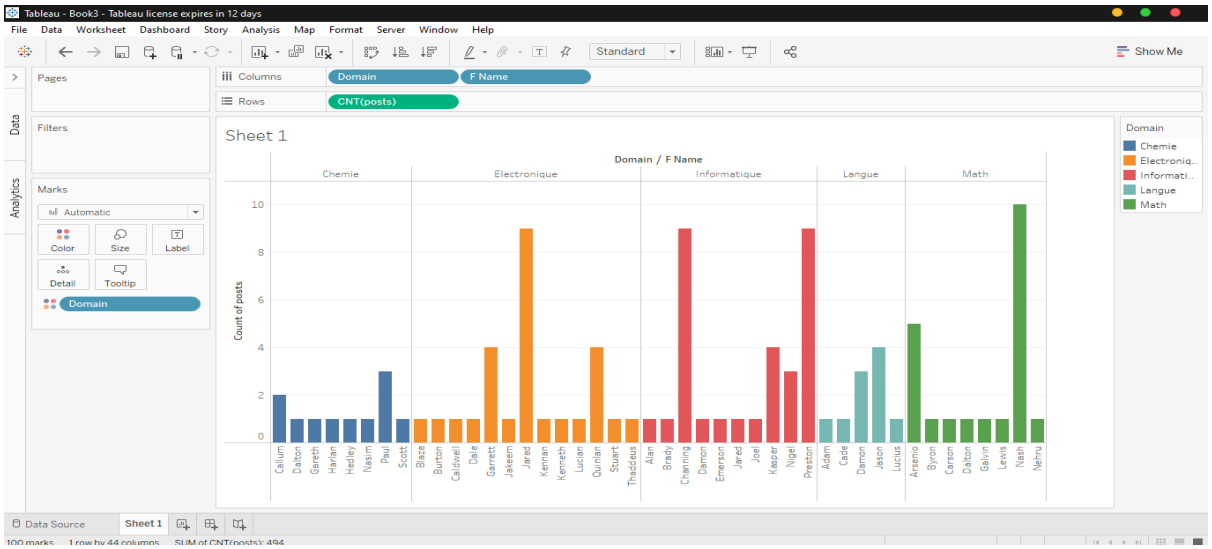


Figure 4.18 Graph nombre des publications par Domaine et les noms utilisateurs (f_name)

4.4 Conclusion

Dans ce chapitre nous avons présenté les différentes étapes d'implémentation de la solution décisionnelle ainsi que les outils utilisés pour finaliser sa création notre modèle SOLVE.

Conclusion Générale

Le NoSQL est une technologie qui émerge en puissance, il est mis en œuvre dans des environnements manipulant de grandes masses de données tels que Google, Yahoo, Twitter, Facebook, etc. Les moteurs de recherche sont les premiers utilisateurs de ces technologies puisqu'ils ont besoin d'une grande puissance de stockage et de traitement de ces volumes de données, de même pour les réseaux sociaux qui gère une très grosse montée en charge dû au grand nombre d'utilisateurs et de requêtes simultanées.

Dans Le premier chapitre on traite les entrepôts de données leurs constructions avec les SGBDR et les bases multidimensionnelles telle que ROLAP, HOLAP, MOLAP. Le deuxième chapitre on va présenter la méthode dite NoSQL ainsi que ces différents types.

Notre travail consiste à transformer un modèle de données conceptuel multidimensionnel en modèles logiques orientés colonnes vont être définies pour instancier entrepôt de donnes qui vas être exploite dans une application d'aide à la décision .et consiste à implémenter un entrepôt de données qui traite le thème des données SOLVE.

Bibliographies

[1] Rakesh Agrawal, A. Gupta, and Sunita Sarawagi. Modeling Multidimensional Databases. In Alex Gray and Per-Åke Larson, editors, Proc. 13th Int. Conf. Data Engineering, ICDE, pages 232–243. IEEE Computer Society, 7–11 1997.

[2] Luca Cabibbo and Riccardo Torlone. Querying Multidimensional Databases. In Workshop on Database Programming Languages, pages 319–335, 1997.

[3] William Inmon. Building the Data Warehouse. QED Technical Publishing Group, Wellesley, Massachusetts, U.S.A., 1992, 1992.

[4] William Inmon. What is a Data Warehouse, White paper., 1995.

[5] R. Kimball. Entrepôts de données, Guide pratique du concepteur de data warehouse. John Wiley and Sons, Inc., 1996.

[6] R. Kimball and M. Ross. Entrepôts de données, Guide pratique de modélisation dimensionnelle. Vuibert, Paris, 2003.

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

[8] P. Marcel. Manipulations de Données Multidimensionnelles et Langages

de Règles. PhD thesis, Institut National des Sciences Appliques de Lyon, France, 1998.

[9] O. Teste. Modélisation et manipulation d'entrepôts de données complexes et historisées. PhD thesis, Université Paul Sabatier de Toulouse, Décembre 2000.

[10] BI Niouses . Objectifs d'un datawarehouse https://biniouses.wordpress.com/article/datawarehouse/objectifs-datawarehouse/

[11] WayToLearn. Différence entre ROLAP et MOLAP https://waytolearnx.com/category/big-data

[12] Bruchez Rudi. 2015. Les bases de données NoSQL et le BigData: Comprendre et mettre en œuvre

[13] Bruchez Rudi. 2015. Les bases de données NoSQL et le BigData : Comprendre et mettre en oeuvre

[14] Guru99. Types of NoSQL Databases, what is & Example https://www.guru99.com/nosql-tutorial.html

[15] DataStax. La modélisation des données https://www.datastax.com/learn/data-modeling-by-example

[16] Apache Cassandra. Documentation https://cassandra.apache.org

[17] Django. Documentation https://www.djangoproject.com/

BIBLIOGRAPHIES

[18] Visual studio. Documentation

https://visualstudio.microsoft.com/