République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma –

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'étude Master

Filière : Informatique

Option : STIC

Thème :

Contrôle de qualité de données NoSQL

Encadré par :

Dr. AGGOUNE Aicha

Présenté par :

Hemaizia Bassem

Juin 2022

Remerciement

Avant tout, je remercie Dieu le tout puissant de nous avoir donné le courage et la patience pour réaliser ce travail malgré les difficultés rencontrées.

Je tiens à remercier mon encadreur de mémoire de fin d'étude

« Dr. Aicha AGGOUNE», pour ses précieux conseils et ses orientation tout au long du projet.

Et je tiens à remercier toutes les personnes qui ont contribué au succès de mon projet et qui m'ont aidée lors de la rédaction de ce mémoire.

Je tiens à remercier tous les membres de jury pour l'honneur qu'ils nous ont fait en acceptant ce travail.

Dédicace

A

Ma famille

Ma mère, Mon père, Mes sœurs et Mes frères pour leurs sacrifices, leurs patiences, leurs amours et leurs confiances en moi, ils ont tout fait pour mon bonheur et ma réussite.

Nulle dédicace ne peut exprimer ce que je leurs dois; Que dieu leurs réserve la bonne santé et une longue vie.

A

Mes Amis

Aidaoui Houssem, Djebarnia Nour El Islem, Zardoudi Mohamed, Touati Islem.

Résumé :

Ce travail s'articule autour de contrôle de la qualité des données NoSQL, plus particulièrement les données orientées documents. En fait, il s'appuie sur une méthode qui permet de détecter et de réparer les problèmes de chevauchement schématique, de duplication, et de l'incomplétude en se basant sur la fréquence des éléments de base de données. A cet effet, une nouvelle méthode nommée MFU (Most Frequently Used) a été proposée.

La méthode MFU comporte trois phases : (1) Détection des problèmes de la qualité, (2) Réparation des données, et (3) Vérification de la qualité. Dans chaque phase, on traite trois types de problèmes de qualité de données.

Notre méthode MFU a été validée par l'implémentation de l'outil QoDB (Quality of Document oriented Database) et évaluée sur la base de données MongoDB COVID19 qui a été publiée en 2022. Les résultats obtenus sont intéressants.

Mots clés : Qualité de données, données NoSQL, MongoDB, Détection des problèmes, Réparation, Vérification.

Abstract:

This work revolves around quality control of NoSQL data, specifically document-oriented data. In fact, it relies on a method that allows to detect and repair the problems of schematic overlap, duplication, and incompleteness based on the frequency of database elements. For this purpose, a new method named MFU (Most Frequently Used) has been proposed.

The MFU method consists of three phases: (1) quality problem detection, (2) data repair, and (3) quality verification. In each phase, three types of data quality problems are addressed.

Our MFU method has been validated by implementing the Quality of Document oriented Database (QoDB) tool and evaluated on the MongoDB COVID19 database that was released in 2022. The results obtained are interesting.

Keywords: Data quality, NoSQL data, MongoDB, Problem detection, Repair, Verification.

Table des matières

Liste des figures:

Liste des tableaux:

Introduction générale

Avec l'augmentation de la bande passante sur internet, 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 augmenter considérablement. L'informatisation croissante de traitement en tout genre a eu pour conséquence une augmentation exponentielle de ce volume de données appelées souvent Big data ou données massives.

La propagation du volume important de données et la diversité de schéma de données ont abouti à la difficulté de stoker, modifier, et gérer ces données par un SGBD (système de gestion de base de données) relationnel basé sur le langage SQL (Structured Query Language). Le NoSQL (Not Only SQL ou pas seulement SQL) représente une approche alternative de SQL permettant le stockage et l'analyse de données structurées et non structurées du Big data.

Contrairement aux bases de données relationnelles, les bases de données NoSQL sont capable d'y stocker des données sous une forme non structurée, sans suivre de schéma fixe et sans utilisation des jointures entre objets.
L'objectif principal du NoSQL est de faciliter le stockage de données volumineuses sans restriction et avec une distribution et disponibilité assurées.

Cependant, la quantité massive et l'hétérogénéité de données NoSQL présentent un défi quant à leur collecte et leur manipulation. L'un des problèmes majeurs soulevés dans les bases de données NoSQL est celui de la qualité des données. En effet, contrôler et gérer la qualité de données NoSQL permet de faciliter leur compréhension et leur utilisation pour par exemple l'exploitation de données, l'analyse de données, l'expérimentation des datasets NoSQL, etc. Plus de données avec qualité plus de précision est de bon résultat.

Notre but ultime de ce travail, c'est alors de proposer une méthode de contrôle de qualité de données NoSQL, plus précisément les données orientées documents de MongoDB. De plus, la détection des problèmes de qualité de données doit être munie d'une méthode de réparation des données de mauvaise qualité. La vérification de données après la réparation de données est encore indispensable pour évaluer les performances de nos propositions.

Afin d'atteindre les objectifs cités ci-dessus, les contributions principales de ce travail sont les suivantes :

- Etude de données NoSQL, plus précisément les données orientées documents de MongoDB.

- Etat de l'art des concepts relatifs à la qualité de données.

- Etat de l'art sur les techniques les plus récentes sur la gestion de qualité de données NoSQL.

- Proposition de méthode MFU (Most Frequently Used) pour le contrôle de qualité de données selon trois dimensions de qualité : la consistance représentationnelle, la complétude et la concision. Ces trois dimensions de qualité affectent d'autres dimensions de qualité, comme la cohérence, la facilité d'utilisation et la facilité de compréhension.

- Implémentation de l'outil QoDB (**Q**uality **o**f **D**ocument oriented data**B**ase) pour valider et évaluer les performances de nos propositions.

Ce mémoire est organisé en quatre parties :

Chapitre 01 : Les données NoSQL

Dans ce chapitre nous présentons un état de l'art des données NoSQL, ensuite nous décrivons le passage de SQL vers le NoSQL ainsi que les caractéristiques de données NoSQL. Nous terminerons le chapitre par les différentes familles de bases de données NoSQL.

Chapitre 02 : Qualité des données dans les bases de données NoSQL

Dans ce chapitre, nous présentons au début les dimensions de qualité de données d'une façon générale, et nous nous focalisons beaucoup plus sur la nature de données NoSQL. Par la suite, nous présentons les techniques d'aides au contrôle de données et quelques travaux récents sur le contrôle de données NoSQL.

Chapitre 03 : MFU Method : Une méthode de contrôle de qualité de données NoSQL orientées documents

Dans le chapitre 3 nous présentons les problèmes de qualité de données étudiées. Ensuite nous présenterons notre proposition pour le contrôle de qualité des données orientées documents MFU (Most Frequently Used)

Chapitre 04 : Implémentation de l'outil QoDB

Dans ce chapitre nous présentons l'implémentation de l'outil QoDB (Quality of Document oriented dataBase) pour valider la méthode MFU de contrôle de qualité de données NoSQL orientées document.

Chapitre 01: Les données NoSQL

1.1. Introduction

À la naissance de l'informatique, plusieurs modèles de stockage de données sont explorés, comme le modèle hiérarchique, réseau et le modèle relationnel. Ce dernier est le plus performant qui permet de mieux assurer le contrôle de l'intégrité des données, grâce à un modèle théorique puissant et simple basé sur le langage SQL. Néanmoins, ces dernières années, le volume de données ne cesse de croître. Une alternative de données SQL appelée NoSQL permet de remédier les problèmes du volumétrie, d'hétérogénéité, de scalabilité, etc.

Dans ce premier chapitre nous allons présenter un état de l'art des données NoSQL. Nous décrivons le passage de SQL vers le NoSQL et les caractéristiques de données NoSQL. Nous terminerons le chapitre par les différentes familles de bases de données NoSQL.

1.2. Données SQL

Le modèle relationnel a été introduit par **Edgar Frank Codd** du centre de recherche d'IBM à travers son article « A Relational Model of Data for Large Shared Data Banks » publié en juin 1970, sa simplicité et ses fondations mathématiques sont faites l'objet de très nombreuses recherches qui ont débouché sur la réalisation de nombreux SGBD relationnels. Un seul type de structure utilisé pour représenter les données stockées dans la machine : la relation ou la table. Ces données sont gérées par un système de gestion de bases de données relationnel [1].

Le modèle relationnel basé est sur un modèle mathématique qui est celui de la notion des ensembles. Chaque ensemble ici est représenté par une table, et ces attributs sont quant à eux représentés par des colonnes. L'un des principes fondamentaux est justement cette notion de relation entre tables à l'aide de cardinalités, clés primaires et clé étrangères, ceci implique au préalable une étude minutieuse sur la modélisation du schéma de la base de données. Comme par

exemple les tables dont le système aura besoin, ses attributs, les relations possibles entre différentes tables, etc. Une fois ce modèle mis en place dans un SGBDR, il est difficile de changer la structure de celui-ci et ceci pose par conséquent des problèmes lors de sa réingénierie [2].

1.2.1 Transactions

Une transaction est un ensemble d'opérations faisant passer l'état du système d'un point A (antérieur à la transaction) à un point B (postérieur à la transaction) [3].

Par exemple lorsqu'une transaction bancaire a lieu (un virement de 100 euros entre un compte A et un compte B), la transaction assure (et rassure) que le virement s'est bien déroulé correctement. Il n'y a pas eu d'erreur, tout a bien été enregistré, les 2 comptes ont bien pris note des mouvements et n'ont pas engendré d'erreur. Si un problème est détecté durant l'opération, la transaction est déconstruite et les données sont restaurées dans leur état initial. (4)

Dans les SGBDR une transaction doit respecter les propriétés ACID afin que celle-ci soit validée. Le respect de ces propriétés dans un environnement distribué est pénible et il existe un risque de diminution des performances proportionnelle aux nombres de serveurs.

1.2.2 Propriétés ACID

Les propriétés ACID (atomicité, cohérence, isolation, durabilité) sont un ensemble de propriétés garantissant la fiabilité d'une transaction informatique telle qu'un virement bancaire par exemple.

- **Atomicité:** cette propriété assure qu'une transaction soit effectuée complètement, ou pas du tout. Quelle que soit la situation, par exemple lors d'une panne d'électricité, du disque dur ou de l'ordinateur, si une partie de la transaction n'a pu être effectuée, il faudra effacer toutes les étapes de celle-ci et remettre les données dans l'état où elles étaient avant la transaction.

- **Cohérence:** la cohérence assure que chaque transaction préserve la cohérence de données, en transformant un état cohérant en un autre état de mêmes données cohérentes. La cohérence exige que les données concernées par une potentielle transaction soient protégés sémantiquement [3].
- **Isolation:** L'opération doit se faire en toute autonomie sans dépendance à une autre opération. En d'autres termes, l'isolation assure que chaque transaction voit l'état du système comme si elle était la seule à manipuler la base de données, par exemple si pendant la transaction *T1*, la transaction

  T2 s'exécute au même moment, *T1* ne doit pas la voir, tout comme *T2* ne doit pas voir *T1* [3].

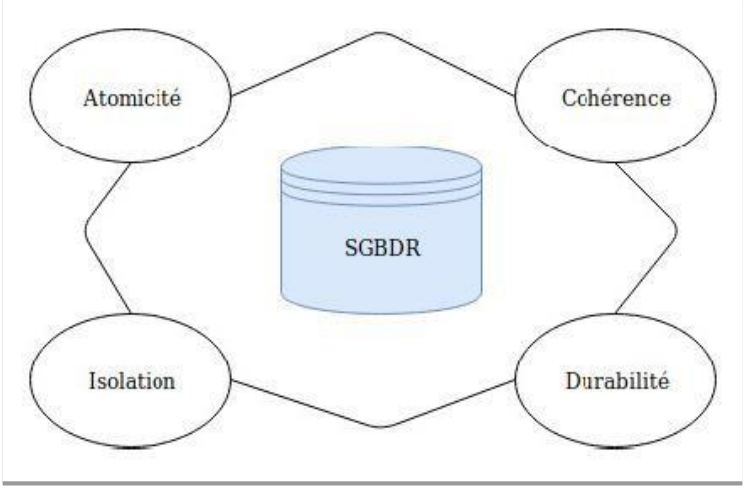

Figure 1.1. Propriété ACID [4]

1.2.3 Théorème de CAP

Le théorème CAP s'appelle également théorème de Brewer, car il a été avancé pour la première fois par le professeur Eric A. Brewer lors d'un discours sur l'informatique en réseau prononcé en 2000. Deux ans plus tard, les professeurs Seth Gilbert et Nancy Lynch du MIT publient une preuve de « la conjecture de

Brewer ». Le théorème CAP nous dit qu'un système distribué ne peut fournir que deux des trois caractéristiques souhaitées : cohérence (Consistency), disponibilité (Availability) et tolérance au partitionnement (Partition Tolérance).

- **Cohérence (Consistency) :** La cohérence signifie que tous les clients voient les mêmes données en même temps, quel que soit le nœud auquel ils se connectent. Pour cela, chaque fois que des données sont écrites sur un nœud, elles doivent être instantanément transférées ou répliquées vers tous les autres nœuds du système pour que l'écriture soit considérée comme « réussie ».
- **Disponibilité (Availability) :** La disponibilité signifie qu'un client qui effectue une demande de données obtient une réponse, même si un ou plusieurs nœuds sont en panne. Autre façon de l'exprimer : tous les nœuds actifs du système réparti renvoient une réponse valide à toute demande, sans exception.
- **Tolérance au partitionnement (Partition Tolérance):** Un *partitionnement* est une rupture de communication au sein d'un système réparti, une perte ou un retard temporaire de connexion entre deux nœuds. La tolérance au partitionnement signifie que la grappe (cluster) doit continuer à fonctionner quel que soit le nombre d'interruptions entre les nœuds du système [5].

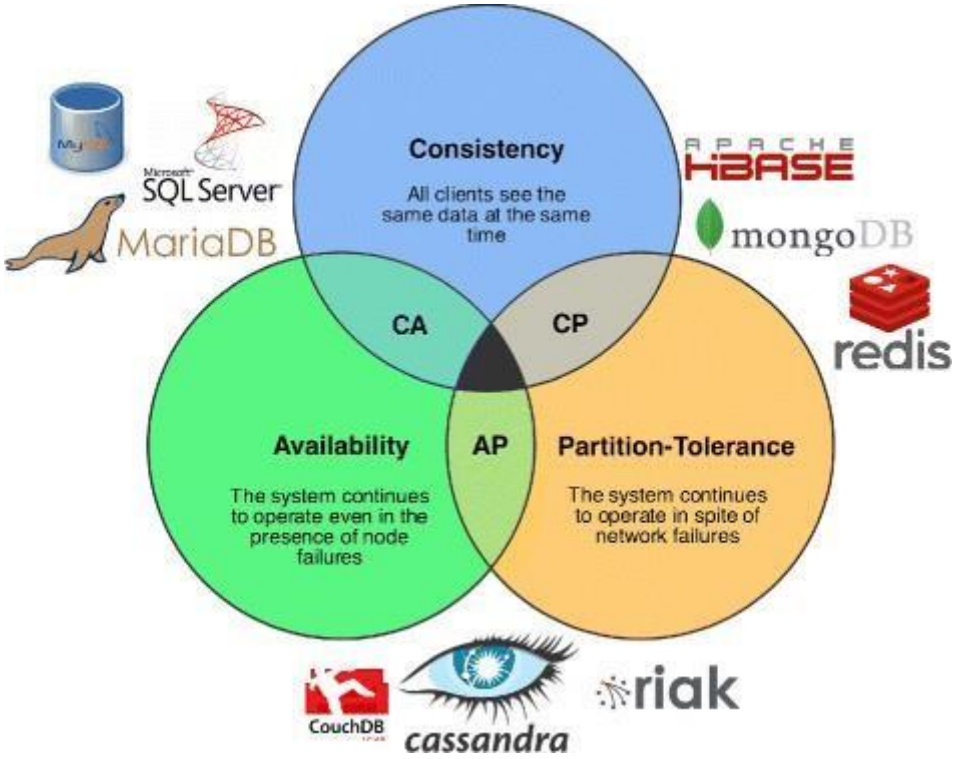

Figure 1.2 – Théorème de CAP [6]

1.3. Limites de données SQL

Les bases de données SQL (Structured Query Language) existent depuis plus de quatre décennies. L'utilisation a explosé à la fin des années 1990 avec l'augmentation des applications Web et des solutions open source telles que MySQL, PostgreSQL et SQLite, au fil des années certaines limites sont apparus lors de manipulation de ces données, les plus importantes sont [4] [7]:

Volume : Le volume de données augmente à un rythme exponentiel, c'est à dire que la création de données ne cesse de s'accélérer, le modèle sql n'arrive plus à suivre.

Vélocité : l'évolution rapide des données nécessite beaucoup de temps pour les traiter, le modèle SQL classique n'arrive plus à suivre.

Variété : le volume gigantesque de données SQL entraîne l'hétérogénéité de données qui traite le même sujet, ce qui a posé un problème lors des

interrogations des données, gérer et traiter cette énorme quantité de données n'est pas à la portée du modèle relationnel.

Simplicité de structure: Dans une base de données SQL, il est impossible d'ajouter des données tant que vous ne définissez des tables et des types de champs dans ce que l'on appelle un schéma. De plus, ce schéma SQL contient d'autres informations : Clés primaires – index, contraintes, fonction, procédures stockées, le schéma de données doit être conçu et mis en œuvre avant que toute logique métier puisse être développée pour manipuler ces données. Il est possible de faire des mises à jour plus tard, mais de gros changements peuvent être compliqués.

Scalabilité limité : les données SQL ne sont pas adaptées aux changements de l'environnement requis par le volume important de données, la seule solution est d'appliquer une segmentation de tables relationnelles.

Théorème de CAP limité : ce théorème est limité en milieu distribué, il engendre un coût énorme de stockage et de gestion, car il est difficile de valider une transaction sur tous les serveurs et en temps raisonnable.

Tous ces problèmes cités auparavant ont poussé les chercheurs à trouver une solution alternative, qui résout ces problèmes, et qui est mieux adaptée à certains types de projets, cette solution est appelée NoSQL.

1.4. De SQL au NoSQL

Le terme **" NoSQL** " signifie en fait **" Not Only SQL "** (pas seulement SQL). En effet, les bases de données relationnelles utilisent la syntaxe SQL pour le stockage et la manipulation de données.

Une base de données NoSQL est une base de données "non relationnelle". Il est possible d'y stocker des données sous une forme non structurée, sans suivre de schéma fixe. Les jointures ne sont plus nécessaires, et le scaling est facilité.

À l'heure du Big Data, les bases de données relationnelles ne sont plus adaptées. Pour prendre en charge les immenses volumes de données, les stocker et les analyser, il est impératif de s'en remettre à de nouvelles solutions [8].

1.4.1 Pourquoi l'alternative du 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 Amazone 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 destinées 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 [2].

1.4.2 Différences entre le NoSQL et le SQL

Pour commencer, il est important de savoir que le SQL n'est pas un modèle relationnel en soit, mais un langage de manipulation de données conçu autour du modèle relationnel. Les bases de données NoSQL n'ont pas pour but de s'éloigner de ce langage mais du modèle relationnel. Nous allons voir les facteurs différenciateurs [9].

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 clévaleur 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.

Table 1.1: Comparaison entre SQL et NoSQL.

1.5. Familles de bases de données NoSQL

Les bases de données NoSQL sont donc une catégorie de base de données qui n'est plus fondée sur l'architecture classique des bases relationnelles, et non pas un type à part entière. Quatre grandes familles se distinguent parmi celles-ci.

1.5.1 Bases de données clé-valeur

En principe, le modèle clé-valeur est l'un des types de bases de données les moins complexes. Ces bases de données sont idéales pour accéder aux données par le biais d'une clé. La spécificité est que les données peuvent être stockées sans définir de schéma spécifique. Il utilise des fonctions très simples pour stocker, obtenir et supprimer des données qui sont stockées sous forme de paires clé / valeur. Ce type des bases de données est très efficace pour la lecture et l'écriture, et conçues pour s'adapter massivement et proposer un temps de réaction extrêmement rapide.

Les données sont entreposées dans un tableau de " hash " au sein duquel chaque clé est unique. La valeur peut être un JSON, un objet BLOB, une ligne de code ou autre. Par défaut, ce SGBD supporte trois opérations :

- PUT ajoute ou met à jour une nouvelle paire de clé-valeur.
- GET retourne la valeur associée à une clé spécifique.
- DELETE supprime une clé et sa valeur d'une table.

Les références dans cette famille de bases de données sont Redis, Riak, Oracle NoSQL et Microsoft Azure Table Storage [1] [10].

Le schéma suivant montre un exemple de données stockées sous forme de paires clé-valeur:

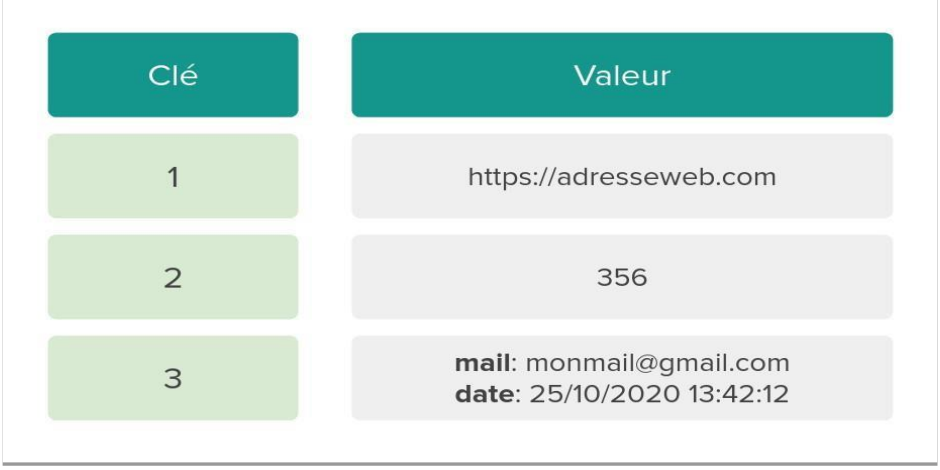

Figure 1.3. Exemple de données orientées clé-valeur

1.5.2 Bases de données orientées colonnes

Comme leur nom l'indique, repose sur des colonnes, ce type de modèle de BD est une extension des bases de données clé- valeurs, avec des colonnes et des caractéristiques des bases de données relationnelles. Elles sont peu adaptées aux mises à jour fréquentes et préfèrent les ajouts. Les relations ne sont pas supportées, ce qui implique la création d'une structure en réseau pour les données connectées, ce qui n'est pas la solution la plus efficace. Elles sont utilisées quand une grande quantité de données doit être stockée et qu'une grande disponibilité est demandée. La plupart des systèmes de type orienté colonne peuvent supporter les données géospatiales, mais surtout avec des données intégrées et des requêtes simples avec une demande de réponse rapide. HBase, Cassandra, Big Table, DynamoDB et Accumulo sont des exemples de bases de données orientées colonnes qui sont
utilisées par Amazon, Google et Facebook [10].

Ce schéma montre un exemple de données stockées dans une base de données orientée colonnes :

Figure 1.4 : Exemple de données orientées colonnes [11]

1.5.3 Bases de données orientées document

Les bases de données orientées documents consistent à gérer une grande quantité de données. Une clé est assignée à un document qui peut lui-même posséder plusieurs couples clé-document intégrés. Un document peut être importé sous différents formats standards comme XML (Extensible Markup Language), JSON (JavaScript Option Notation) ou BSON (Binary JSON). La différence aux bases de données clé-valeurs est que les recherches peuvent être effectuées sur le contenu des documents et pas uniquement leur clé. La présence des documents intégrés permet d'effectuer des recherches avancées et ne requiert pas de schémas prédéfinis.

Les BD orientées documents permettent à un document d'être décrit par un grand nombre de valeurs et peuvent supporter un schéma flexible. Elles peuvent accueillir une grande quantité de données sous des formats fortement différents. La

relation entre les documents est représentée par l'intégration de ceux-ci (Embedded documents) ou utilisation des références. Ainsi, pour des requêtes qui interrogent les documents d'une même collection (ensemble de documents), la réponse sera très rapide. L'utilisation de document XML peut améliorer les BD orientées documents en offrant des fonctionnalités complémentaires (XQuery, Xpath, XPointer) et en offrant le principe de relation qui n'existait pas sans le XML. On retrouve principalement CouchDB et MongoDB comme solutions basées sur le concept de base de données orientées documents et Twitter est un d'utilisateur [10].

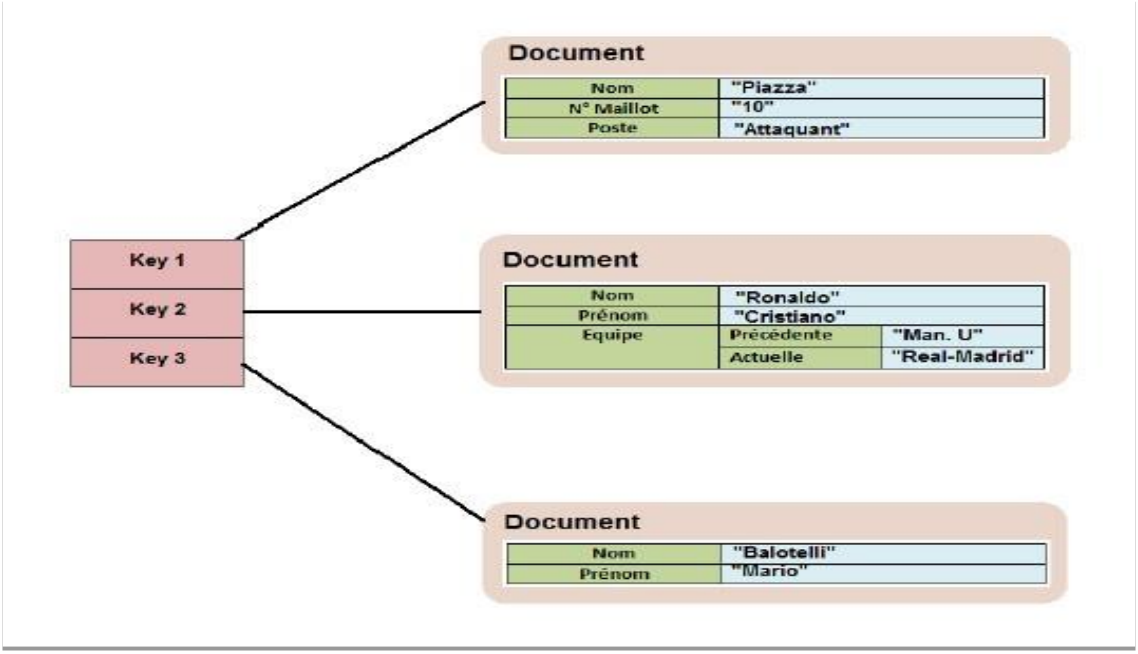

Figure 1.5. Exemple de données orientées document [2]

1.5.4 Base de données orientées graphe

La base de données orientée graphe est un type de base de données NoSQL utilisant la théorie des graphes pour stocker, cartographier et effectuer des requêtes sur les relations entre les données

Les bases de données orientées graphes sont constituées de nœuds et de bords.

Chaque nœud représente une entité, et chaque bord représente une connexion entre les nœuds. Elles gagnent en popularité dans le domaine des analyses d'interconnexions. Ce modèle est capable de gérer des données relationnelles. Les BD graphes sont utiles pour les données constituées en réseau comme les routes, les réseaux sociaux. Contrairement aux autres types de bases de données NoSQL, les BD orientées graphes ne sont pas optimales pour les grandes quantités de données si celles ne sont pas assez connectées.

Oracle Graph, Neo4J, Arango et OrientDB sont des exemples de ce modèle. Elles sont utilisées par Walmart et Ebay [10].

Voici un exemple des données orientées graphe :

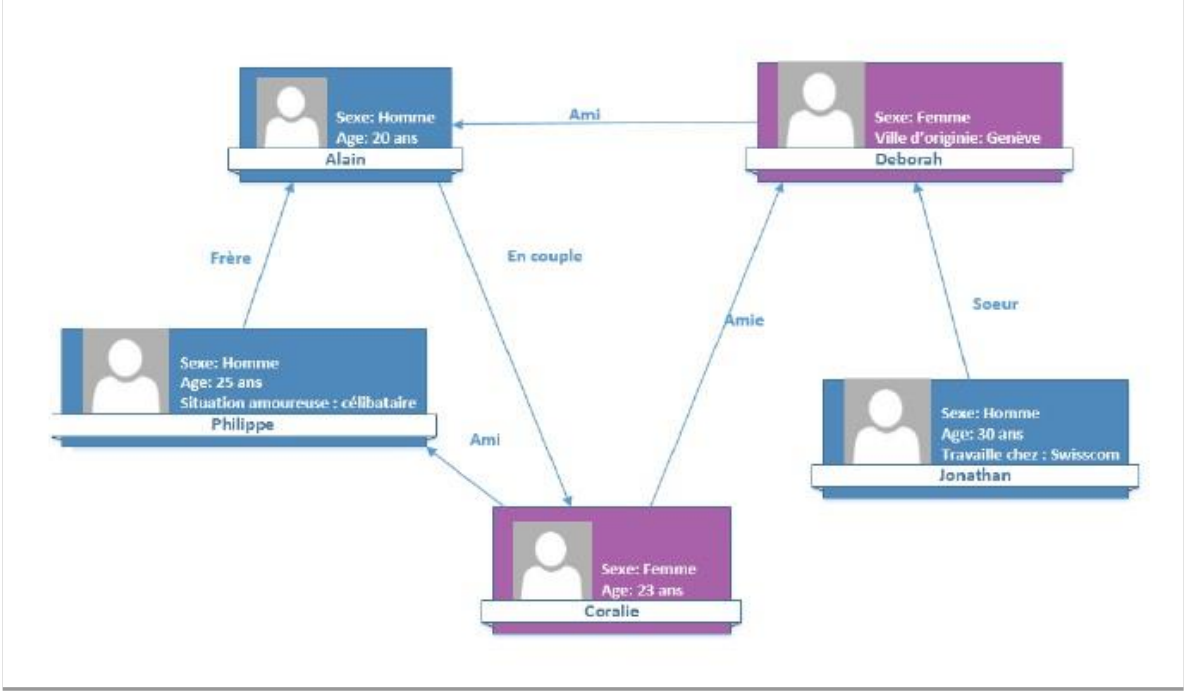

Figure 1.6. Exemple de données orientées graphe [2]

1.6. Conclusion

Nous avons présenté les différents types de bases de données (SQL et NoSQL) avec les limites des bases de données SQL et pourquoi choisir le NoSQL. De plus, nous avons présenté les différentes familles des bases de données NoSQL.

Dans le chapitre suivant, nous allons voir la notion de qualité de données dans les bases de données NoSQL, et voir aussi les dimensions avec quelques métriques de contrôle de qualité. Et enfin, nous présenterons les travaux récents sur le contrôle de qualité de données NoSQL.

Chapitre 02. Qualité des données dans les bases de données NoSQL

2.1. Introduction

La qualité de données fait partie des enjeux majeurs pour les organisations sur le plan de performance d'analyse de données et le plan financière. Les données de qualité permettent aux entreprises d'améliorer leurs performances opérationnelles, de satisfaire la clientèle et d'être plus compétitives en réorientant rapidement leur stratégie d'entreprise.

Dans ce chapitre, nous découvrons les dimensions de qualité de données d'une façon générale, et nous nous focalisons beaucoup plus sur la nature de données

NoSQL. Par la suite, nous présentons les techniques d'aides au contrôle de données et quelques travaux récents sur le contrôle de données NoSQL.

2.2. Qu'est-ce que la qualité des données ?

La qualité des données (QD ou data quality en anglais) est un concept bien connu au sein de la communauté des bases de données et constitue un domaine de recherche actif depuis de nombreuses années [15]. D'après [12], la qualité des données n'est pas facile à définir, ses définitions sont sensibles au domaine des données.

Sidi et al [13] ont défini la qualité de données comme la pertinence de l'utilisation et la satisfaction des besoins des utilisateurs.

Devillers et al [16] défini la qualité de données comme l'accord entre les caractéristiques des données et les besoins explicites et/ou implicites d'un utilisateur pour une application donnée dans un domaine donné.

Juran et al [14] ont défini la qualité comme "l'aptitude à l'emploi" (ou « Fitness for use ») pour répondre aux besoins des membres de société. Fitness for use indique l'efficacité d'une conception (Effectiveness of a design), d'une méthode de fabrication (manufacturing method) et d'un processus de support utilisés pour fournir un système ou un service qui correspond à l'objectif des clients, dans des conditions opérationnelles prévues ou spécifiées. La qualité de données appelée aussi Fitness to use.

La définition de Joseph M. Juran (auteur et éditeur de handbook cité dans [14]) a obtenu un accord officiel entre les organismes de normalisation (ISO) et les organisations internationales comme IEEE.

La norme ISO/CEI 25012 (ISO/CEI, 2008) identifie alors, la qualité de données selon deux points de vue [17] :

- ***Qualité des données inhérentes :*** Les caractéristiques inhérentes de qualité des données ont un potentiel intrinsèque pour satisfaire les besoins de données implicites ou explicites.
- ***Qualité des données dépendante du système :*** fait référence à la mesure dans laquelle la qualité des données est atteinte et préservée grâce à un système d'information et dépend du contexte technologique spécifique dans lequel les données sont utilisées.

2.3. Dimensions et métriques d'évaluation de qualité de données

La qualité des données peut être décrite à l'aide de nombreuses caractéristiques ou dimensions comme la précision, la performance, la cohérence, la fiabilité, la complétude, l'actualité, etc. Il existe dans la littérature plusieurs classifications des dimensions, ainsi que dans chaque type de classification, les auteurs ont donné leurs propres définitions pour chaque dimension [18]. En se basant sur les différentes synthèses et comparaisons entre les classifications des dimensions [12, 18, 13], Wang et Strong ont été les premiers qui ont établi les

dimensions du point de vue de l'utilisateur de données adapté à l'utilisation en 1996 [19]. Wang et Strong [19] ont classifié 20 dimensions en quatre catégories : Dimensions intrinsèques, Dimensions contextuelles, Dimensions représentationnelles et les dimensions d'accessibilité.

Les différentes dimensions de qualité de données doivent être évaluées à l'aide des métriques utilisées pour par exemple le nettoyage de données (Data cleaning or cleansing), réparation de données (Data repairing), …etc.

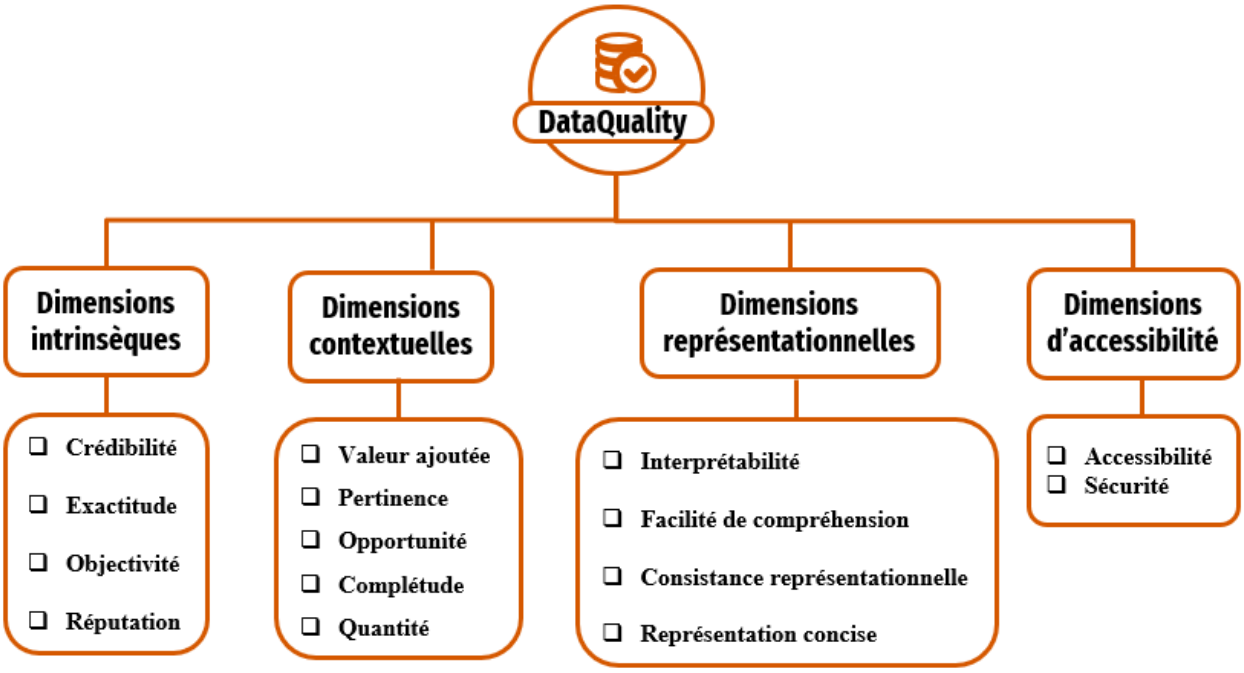

Figure 2.1. Les dimensions de qualité de données [19]

2.3.1 Dimensions intrinsèques

Les dimensions intrinsèques concernent les caractéristiques objectifs et natives de données qui sont liées à la crédibilité, l'exactitude, l'objectivité et la réputation.

Crédibilité (Believability) : Les données extraites sont valides dans leur finalité d'utilisation. La crédibilité est fortement liée à la validité (validity). En effet, les

données doivent être collectées selon des règles de gestion (business rules) et des paramètres commerciaux définis, et doivent se conformer au bon format.

Exactitude (Accuracy) : L'exactitude ou la précision représente la mesure dans laquelle les données correspondent (ou sont proches) des vraies valeurs des items, également comprises comme les valeurs correctes et précises (c'est la précision syntaxique). La précision sémantique concerne le degré d'exactitude des données par rapport aux valeurs du monde réel

Métrique d'exactitude : L'exactitude peut être calculée en divisant les éléments ou enregistrements exacts par le nombre total d'éléments ou d'enregistrements. Par exemple, un registre local de la population contient 932 904 numéros de téléphone et 813 942 ont été confirmés, ce qui donne une précision de 87,25 % (813 942/932 904 * 100).

Objectivité (Objectivity) : L'objectivité des données dans un dataset est la mesure dans laquelle les données sont créées par des moyens objectifs. Il existe une relation directe entre l'objectivité et la crédibilité. L'objectivité étends la crédibilité.

Réputation (Reputation) : les données répondent aux attentes des utilisateurs. La réputation d'un ensemble de données est l'opinion des utilisateurs sur les données en raison de ce qui s'est passé dans le passé. La réputation est liée à la crédibilité.

2.3.2 Dimensions contextuelles

Elles sont liées aux données elles même par rapport le contexte de la tâche à accomplir. Elles concernent la valeur ajoutée, la pertinence, l'opportunité, la complétude et la quantité de données.

Valeur ajoutée (added value) : représente la mesure dans laquelle les données sont utiles et offrent des avantages de leur utilisation.

Pertinence (Relevancy) : Est-ce que les informations stockées sont d'une quelconque utilité pour l'entreprise ? La collecte de données non pertinentes est une perte de temps et d'argent.
La pertinence représente la mesure dans laquelle les données répondent aux attentes et aux exigences de l'utilisateur.

Métrique de pertinence : La pertinence peut être évaluée qualitativement à la discrétion et en fonction des exigences de l'utilisateur, par exemple en utilisant une scorecard (un tableau de bord stratégique dont l'objectif est de prendre en compte l'ensemble des dimensions concourant à la performance au-delà des simples mesures financières.).

Opportunité (Timeliness) : la disponibilité de données à tout moment. Les données peuvent être mises à jour en temps réel pour s'assurer qu'elles sont facilement disponibles et accessibles. L'opportunité est donc fortement liée à l'actualité de données (Currency) qui fait référence à la mesure dans laquelle l'information est à jour avec le monde qu'elle modélise et si elle est correcte malgré d'éventuels changements liés au temps.
Métrique: L'actualité peut être mesurée en fonction du taux de fréquence attendu auquel les éléments de données de base doivent être mis à jour, ainsi que la vérification que les données sont à jour, ce qui nécessite potentiellement des processus automatisés et manuels. Des règles d'actualité peuvent être définies pour affirmer la "durée de vie" d'une valeur de données avant qu'elle ne doive être vérifiée et éventuellement actualisée.
Complétude (Completeness) : Est-ce que les informations sont complètes ? Les champs à renseigner le sont-ils ? D'autres champs utiles pourraient-ils être ajoutés ?

La complétude est l'une des dimensions de la qualité des données les plus couramment appliquées et fournit généralement une mesure de la présence de données.

Métrique: Parmi les métriques utilisées, par exemple le calcul des valeurs manquantes (missing values) pour un attribut. Plusieurs approches existent pour traiter les données manquantes. L'approche la plus simple consiste à supprimer de l'ensemble de données qui ont des valeurs manquantes. Une autre approche basée sur l'imputation des données manquantes qui permet d'attribuer des valeurs de remplacement à des données manquantes

Quantité (Quantity) : la quantité de données requise pour la tâche à accomplir.
La quantité de données (nombre d'enregistrements et de variables) requises doit être suffisante pour un contexte donné.

2.3.3 Dimensions représentationnelles

Les dimensions représentationnelles comprennent des aspects liés au format des données (représentation concise et cohérente) et à la signification des données (interprétabilité et facilité de compréhension).

Interprétabilité (Interpretability) : réfère à la clarté et la description des concepts de données. C'est la mesure dans laquelle les données sont bien présentées et définies. L'interprétabilité est liée la facilité de compréhension.

Facilité de compréhension (ease to understanding) : La facilité de compréhension des données dans un ensemble de données est la mesure dans laquelle les données d'un ensemble de données sont claires, sans ambiguïté et peuvent être comprises.

Consistance représentationnelle (Representational consistency): La mesure dans laquelle les données sont toujours présentées dans le même format et sont compatibles avec les données précédentes. La consistance ou encore la cohérence de données (consistency) permet de détecter si une même donnée contenue dans plusieurs bases présente des résultats différents, donc elle ne peut pas être considérée comme fiable et une investigation doit être menée.

Métrique : La cohérence est liée à la satisfaction aux contraintes d'intégrité dans les bases de données relationnelles. Elle peut être représentée comme la proportion d'éléments ou d'enregistrements jugés cohérents. Par exemple, la date de naissance dans un registre de population doit être stockée dans le format "AAAA-MM-JJ" (année-mois-jour). Dans 61 196 cas sur 930 611 au total, la date de naissance était stockée à l'envers sous la forme "JJ-MM-AAAA", ce qui donne
93,42 %

Représentation concise (Concise representation) : est la mesure dans laquelle les données sont représentées de manière compacte sans être écrasantes (c'est-àdire, brève dans la présentation, mais complète et pertinente). La notion de concision se réfère à la mesure dans laquelle le dataset est exempt de redondances.

Métrique : par exemple détection et traitement de données redondantes ou dupliquées (duplicate data) par l'élimination de duplications notamment dans le cas de Big data.

2.3.4 Dimensions d'accessibilité

Elles concernent la facilité d'accès aux données du système avec toute sécurité.

Accessibilité : L'accessibilité fait généralement référence à la disponibilité des données. Il décrit avec quelle facilité l'utilisateur peut y accéder. Aucune mesure quantitative n'est suggérée pour mesurer l'accessibilité ; au lieu de cela, cette dimension devrait être évaluée qualitativement.

Sécurité: la sécurité est fortement liée à l'accessibilité et l'accès aux données. Elle mesure à quel point les données sont protégées. Elle concerne le niveau de protection, les droits d'accès, la politique de stockage et les contraintes de sécurité.

D'autres dimensions (cinq dimensions) ne figurent pas dans la figure 2, comme :

- **Traçabilité (Tracability)** : La mesure dans laquelle les données sont bien documentées, vérifiables et facilement attribuables à une source.

- **Rentabilité (Cost-effectiveness)** : La mesure dans laquelle le coût de la collecte des données appropriées est raisonnable.
- **Facilité d'utilisation (Ease of operation)** : la mesure dans laquelle les données sont facilement gérées et manipulées (c'est-à-dire mises à jour, déplacées, agrégées, reproduites, etc.).
- **Variété de données et de sources de données (Variety of data and data sources)** : la mesure dans laquelle les données sont disponibles à partir de plusieurs sources de données différentes.
- **Flexibilité (Flexibility)** : La mesure dans laquelle les données sont extensibles, adaptables et facilement applicables à d'autres besoins.

Les problèmes de qualité de données sont alors liés à une ou plusieurs dimensions comme par exemple l'incomplétude, l'incohérence, l'inexactitude, etc. Ils sont également en relations avec des erreurs, des anomalies, des valeurs d'attribut manquantes, des valeurs d'attribut incorrectes ou des représentations différentes des mêmes données. Ces problèmes de qualité de données peuvent être détectés soit au niveau du schéma, au niveau de l'instance, ou au niveau des métadonnées.

Dans le contexte de données NoSQL et en raison de la flexibilité de schéma et la structuration des données (schemaless), nous nous focalisons beaucoup plus sur les problèmes détectés au niveau du schéma, plus précisément le chevauchement schématique lié aux problèmes de représentation concise et la consistance représentationnelles de données NoSQL orientées documents de MongoDB. En ce qui concerne les valeurs de données, nous nous focalisons sur le contrôle de complétude et la concision de données. Ces deux dimensions sont respectivement liées aux problèmes de valeurs manquantes et la redondance de données (plus de détail dans le chapitre 3).

2.4. Travaux récents relatifs au contrôle de qualité de données NoSQL

De nombreux travaux et Framework ont été proposés pour contrôler et gérer les données relationnelles ainsi que les données ouvertes liées du web [24], [25].

La qualité des données est un facteur clé qui détermine la performance des systèmes d'information, notamment pour le big data qui sont souvent stockées dans des systèmes NoSQL. Le plus grand défi pour le big data est la qualité du big data lui-même, c'est ce que l'on appelle Big data quality [12].

Nous présentons par ordre chronologique quatre travaux d'évaluation de la qualité de données NoSQL, que nous jugeons intéressant dans le contexte de notre travail.

2.4.1 Approche de Störl et al, 2018

Störl et al [26] présentent un middleware appelé Darwin qui permet d'extraire une description de schéma NoSQL, découvrir l'historique des versions de schéma et proposer des mappages entre ces versions. Les bases de données traitées par Darwin sont MongoDB et CouchDB. Darwin se compose de quatre principaux composants : Gestionnaire d'extraction de schéma, Gestionnaire d'évolution de schéma, Gestionnaire de migration de données et Gestionnaire de réécriture de requêtes. Le contrôle de qualité de schéma de données est basé sur la détection de différentes opérations appliquées sur le schéma telles que l'ajout d'un nouvel attribut, renommage d'un attribut, …etc.

2.4.2 Approche de Möller et al, 2019

Le travail de Möller et al [23] présente une approche de gestion de qualité de données NoSQL avec des schémas évolutifs dont le but de contrôler l'actualité et la complétude de données sous l'augmentation continue de volume de données. Ce travail fait partie du projet de recherche de développement de Darwin [26]. L'évolution de schéma est basée sur cinq opérations populaires : ajouter, supprimer,

renommer, copier et déplacer. C'est sur la base de ces opérations, le contrôle de qualité de données est maintenu.

2.4.3 Approche de Pablo et al, 2020

Pablo et al [20] ont proposé une approche pour maintenir la qualité de données NoSQL durant l'évolution du schéma. Ils ont basé sur la détermination de modèle conceptuel de données NoSQL et sur la base de ce modèle, toutes modifications du modèle conceptuel impliquent une évolution nécessaire du schéma de la base de données afin d'assurer la consistance entre le schéma et le modèle. Le travail présenté dans [20] traitent les données NoSQL orientées colonnes de Cassandra qui possèdent une structure proche de données relationnelles et qui peuvent être modélisées à l'aide d'un modèle conceptuel guidé par requêtes appelé le modèle Chebotko [21]. Parmi les changements étudiés dans ce travail, l'ajout d'une entité, mise à jour de clé primaire, l'ajout d'une relation, changement de cardinalités, etc. Les auteurs donnent pour chaque type de changement du modèle conceptuel de données NoSQL, la transformation correspondante dans le schéma relationnel.

L'approche proposée est applicable seulement sur les données NoSQL orientées colonnes où leurs modèles conceptuels sont préalablement définis, sinon une présentation manuelle du modèle est obligatoire.

2.4.4 Approche de Conrad et al, 2021

Conrad et al [22] proposent un Framework pour l'évaluation de systèmes NoSQL avec des schémas évolutifs. Il repose sur deux composants : json-datagenerator pour générer un fichier json à évaluer et EvoBench runner pour l'évaluation de performance des évolutions de données NoSQL orientées documents de MongoDB et aussi orientées colonnes de Cassandra. Ce Framework traite la qualité de données dans le contexte de migration de données vers une autre

source (l'opération move) ou la génération d'une nouvelle version par copy. Il est basé aussi sur Darwin [26] pour la validation de EvoBench Framework.

2.5. Conclusion

Nous avons présenté les définitions de la qualité des données, les dimensions avec quelques métriques de contrôle de qualité. De plus, nous avons présenté les travaux récents sur le contrôle de qualité de données NoSQL.

Nous nous intéressons notamment à des questions de qualité liées à la structuration des données au sein de systèmes de données orientées document, type MongoDB. Le chapitre suivant décrit en détail notre proposition.

Chapitre 03. MFU Method : Une méthode de contrôle de qualité de données NoSQL orientées documents

3.1. Introduction

Dans les bases de données NoSQL orientées documents, il est difficile d'assurer la qualité des données dans les documents des collections à cause de flexibilité de schéma, l'absence des contraintes d'intégrité, le volume important de données et l'hétérogénéité de données. Les sociétés informatiques ont rencontré beaucoup de problèmes de qualité quand elles ont essayé de travailler avec des données NoSQL.

Après avoir effectué un état de l'art sur les données NoSQL et les concepts relatifs à la qualité de données ainsi que les différents travails récents sur la gestion de qualité de données NoSQL, nous allons présenter dans ce chapitre les problèmes de qualité de données étudiées qui sont: le chevauchement schématique, la duplication des données et l'incomplétude de données. Ensuite nous présenterons notre proposition pour le contrôle de qualité des données orientées documents. Il s'agit d'une méthode de contrôle de la qualité basée sur le cacul de fréquence de données appelée la méthode MFU (Most Frequently Used).

3.2. Aperçu de la proposition

Notre objectif est de proposer une méthode pour contrôler et améliorer la qualité des données NoSQL orientées documents.
Cette méthode appelée MFU (Most Frequently Used), permet de détecter et réparer les problèmes de la qualité selon les éléments les plus fréquemment utilisés.
Nous avons défini la qualité de données orientées documents selon trois dimensions : la consistance représentationnelle, la représentation concise, et la complétude. Ces

dimensions affectent d'autres dimensions de qualité, comme la précision, la concision et la cohérence de données, Facilité de compréhension.

Notre méthode MFU permet de détecter et réparer les problèmes suivants :

- **Le chevauchement schématique:** en se basant sur le schéma le plus fréquemment utilisé dans les documents d'une collection.
- **L'incomplétude dans une collection:** la détection de ce problème se fait par la recherche des valeurs nulles ou des attributs manquants. MFU offre trois solutions possibles : imputation par la moyenne(mean), imputation par la valeur la plus utilisée(mode) ou suppression de document contenant des données manquantes.
- **La duplication des données:** est détecter lorsque la fréquence d'apparition de données est supérieure à 1. La réparation est tout simplement la suppression des documents dupliqués.

3.3. Les problèmes de qualité de données NoSQL orientées documents

- Le chevauchement schématique:

Un des problèmes majeurs des base de données NoSQL est le chevauchement schématique. Comparer par une base de données SQL, où il est impossible d'ajouter des données tant qu'on ne définit des tables et des types de champs dans ce que l'on appelle un schéma. De plus, ce schéma SQL contient d'autres informations [16]: Clés primaires – index, contraintes, fonction, procédures stockées.

Le schéma de données doit être conçu et mis en œuvre avant que toute logique métier puisse être développée pour manipuler des données. Il est possible de faire des mises à jour plus tard, mais de gros changements peuvent être compliqués.

Dans une base de données NoSQL, la logique est toute autre ! Les

données peuvent être ajoutées n'importe où, à tout moment. Il n'est pas nécessaire de spécifier une conception de document ou même une collection à l'avance.

La flexibilité de schéma c'est un grand avantage, mais elle est considérée comme un inconvénient lorsqu'on veut structurer ces données.

L'exemple suivant montre deux documents dans une collection, avec un ordre des champs différent [27]

-

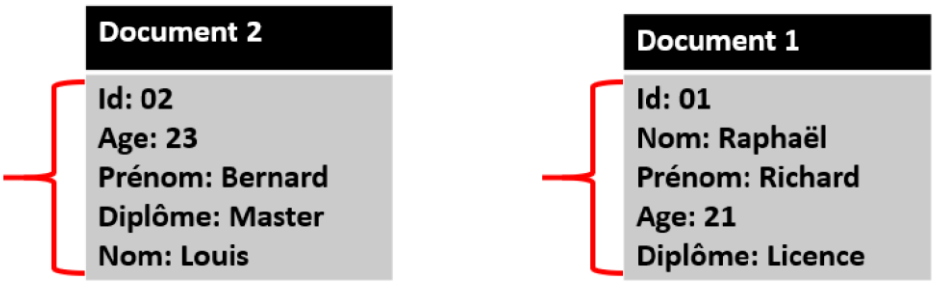

Figure 3.1. Exemple de chevauchement schématique

- **L'incomplétude de données**

L'incomplétude peut être issue d'une absence d'information ou d'une valeur exacte sur certaines champs (ou attributs) de documents. Une donnée incomplète est une donnée pour laquelle la valeur de certains attributs est inconnue, ces valeurs sont dites manquantes [28]. Ainsi dire que l'information générée ou acquise est incomplète représente le fait qu'une ou plusieurs valeurs d'attributs d'une donnée sont manquantes ou inconnues.

La figure suivante illustre un exemple de document avec données incomplètes.

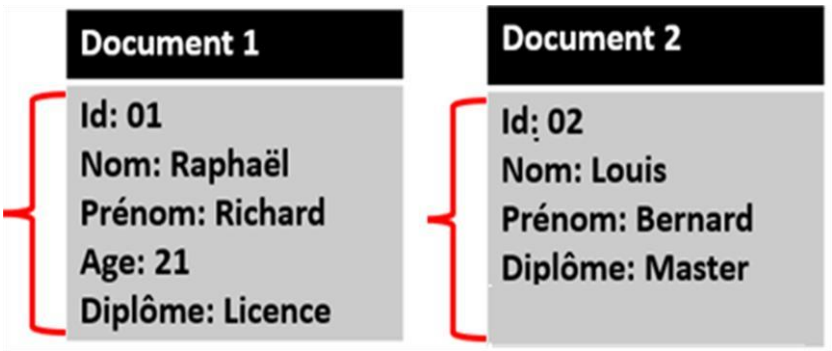

Figure 3.2. Exemple de l'incomplétude de données

Dans cet exemple le document 2 ne comprend pas un champ Age car sa valeur est inconnue.

- **La duplication de données**

La duplication des données dans les bases de données NoSQL orientée document, est un problème courant.

Par conséquent, on ne peut pas avoir la duplication au niveau de l'ID, mais il peut y avoir la duplication au niveau des autres attributs du document, ce qui implique une mauvaise qualité de notre base de données.

Voici un exemple de collection avec des données dupliquées :

	Name	Age	City
0	jack	34	Sydeny
1	Riti	30	Delhi
2	Aadi	16	New York
3	Riti	30	Delhi
4	Riti	30	Delhi
5	Riti	30	Mumbai
6	Aadi	40	London
7	Sachin	30	Delhi

	Name	Age	City
3	Riti	30	Delhi
4	Riti	30	Delhi

Figure 3.3. Exemple de données dupliquées

3.4. Présentation de la méthode proposée "MFU"

La méthode de contrôle de qualité de de données orientées documents MFU (Most Frequently Used) repose sur 3 étapes :

3.4.1 Détection de qualité de données

On distingue 3 types de détection de qualité

- **Schéma:** l'ordre des champs n'est pas important dans les bases de données NoSQL, ce qui rend la qualité des données mauvaise, et pose beaucoup de problèmes dans la manipulation et la compréhension de ces données via les requêtes.
  MFU parcourt les documents de la collection, et détecte si le schéma n'est pas le même, si c'est le cas elle affiche la fréquence d'apparition du même schéma de données et passe par l'étape de réparation, sinon elle affiche le schéma utilisé.

- **Duplication:** MFU parcourt les documents de la collection, et détermine si on a de redondance selon la fréquence d'apparition des documents (sans prendre en compte l'ID du document qui est toujours unique), et affiche les documents redondants. Si la collection contient de duplication, MFU passe à l'étape de réparation.
- **L'incomplétude:** MFU parcourt la liste des documents de la collection, et cherche les documents avec les attributs qui ont des valeurs 'NAN', INF et
  NULL et les affiche. Si les documents contiennent de valeurs manquantes, MFU passe à l'étape de réparation.

3.4.2 Réparation de données

- **Chevauchement schématique**

La réparation du problème de chevauchement schématique sert à utiliser le résultat de la détection de chevauchement schématique qui est un ensemble de couple (schéma, fréquence). Le couple ayant la fréquence la plus élevée est sélectionné comme schéma correcte. Ensuite, on applique ce schéma pivot à notre base de données comme suit :

o Parcourir la collection, et appliquer le schéma sur chaque document o Si la colonne a une valeur dans le document on la récupère et on l'insère dans la nouvelle collection o Sinon on met la valeur 'None' à la place.

- **Duplication des données**

Un des problèmes majeurs des base de données MongoDB orientée document est la duplication des données.

Pour résoudre ce problème, on a utilisé l'algorithme suivant :

- Choisir une collection dans la base de données
- Récupérer un document sans l'ID
- Parcourir tous des documents et vérifier s'il existe un document similaire à celui qu'on a récupéré auparavant
- Refaire les 2 étapes précédente pour chaque document
- Afficher la liste des documents dupliqués
- C'est à l'utilisateur de décider la suppression ou non des documents

- **Réparation de l'incomplétude**

Plusieurs propositions ont été faites pour la réparation de l'incomplétude dans les données :

- Suppression des données comportant des valeurs manquantes ou des données incomplètes et/ou suppression d'un attribut du jeu de données si celui-ci est souvent non renseigné. Cette méthode n'est appropriée que si les valeurs manquantes sont rares, car si le pourcentage de valeurs manquantes est élevé, la perte d'information résultant de la suppression des données incomplètes n'est pas acceptable. De plus, la représentativité statistique de l'échantillon n'est pas toujours applicable.

- Imputation des valeurs manquantes :

Le remplissage par une valeur statistique (moyenne, médiane), difficilement applicable aux gros volumes de données, permet d'obtenir des résultats qui varient de manière importante selon l'estimation réalisée. Par ailleurs, l'imputation doit être la plus proche possible de la réalité pour éviter d'introduire un biais trop important dans les données. Il s'agit, de fait, de traitement par imputation des données.

3.4.3 Vérification de données

La vérification de qualité de données après la détection et l'amélioration de la qualité s'avère nécessaire pour évaluer la méthode proposée. Nous avons proposé un processus de vérification de la qualité qui consiste à évaluer les données qui ont été déjà réparés. En d'autres termes, la sortie de la méthode
MFU devient son entrée et sa sortie doit être identique aux données d'entrée si la MFU est bien gérée la qualité de données.

- **Chevauchement schématique**

Après la réparation du schéma de la collection, on vérifie que le schéma choisit est bien appliqué à tous les documents de la collection, et tous les documents ont le même ordre des colonnes.

- **Duplication des données**

Après la suppression des données dupliquées dans la collection, on vérifie que la collection ne contient plus de documents dupliqués.

- **L'incomplétudes des données**

Après la réparation des données manquantes avec une des méthodes de réparation proposée par MFU (Mean, Mode ou Delete), on vérifie que la collection ne contient plus de valeurs null.

3.5. Conclusion

La recherche dans le domaine de qualité des données reste vaste, dans ce chapitre nous avons traité quelques problèmes de qualité de données MongoDB orienté document. Nous avons présenté en détail notre méthode MFU proposée.

Le dernier chapiter présente l'implémentation de l'outil QoDB (Quality of Document oriented database) validant notre méthode MFU.

Chapitre 04. Implémentation de l'outil QoDB

4.1. Introduction

Nous allons présenter dans ce chapitre, l'implémentation de l'outil QoDB (**Q**uality **o**f **D**ocument oriented data**B**ase) pour valider la méthode MFU de contrôle de qualité de données NoSQL orientées document. Pour cela, nous allons présenter dans un premier temps les paramètres expérimentaux et la base de données orientées documents de MongoDB COVID19. Dans un second temps, nous présenterons quelques exemples d'utilisation de l'outil QoDB.

4.2. Paramètres expérimentaux

4.2.1 Caractéristiques de la machine

La méthode **MFU** proposée est développé en Python 3.10.4. Le choix de ce langage est soutenu par sa facilité d'utilisation, ainsi que par sa capacité de calcul et gestion des grandes quantités de données, en plus une grande communauté utilise ce langage.

Notre solution s'appuie sur les bases de données MongoDB directement. Toutes les expériences ont été réalisées sur une machine Lenovo IdeaPad P400 Touch ayant les caractéristiques suivantes :

- Système d'exploitation : Windows 10
- Processeur (CPU) : I7 2.20 GHz*2
- RAM : 12 GB

4.2.2 Logiciels et langages utilisés

- **Python :**

Python est un langage de programmation interprété, orienté objet et de haut niveau, facile à apprendre en raison de la simplicité de sa syntaxe [29]. Python supporte les modules et les paquets qui encouragent la modularité des programmes et la réutilisation du code. Il contient de nombreuses bibliothèques utilisées dans l'apprentissage automatique et l'IA. Nous avons utilisé la version 3.10.4

- **MongoDB :**

Le modèle orienté documents de MongoDB permet de faire face aux exigences les plus complexes à n'importe échelle. Pourtant, il n'en reste pas moins facile à comprendre, à assimiler et à utiliser pour les développeurs. [30]

- **PyQT5 :**

Il y a tellement d'options fournies par Python pour développer une application graphique et PyQt5 en fait partie. PyQt5 est une boîte à outils d'interface graphique multiplateforme, un ensemble de liaisons python pour Qt v5. On peut développer une application de bureau interactive avec autant de facilité grâce aux outils et à la simplicité fournis par cette bibliothèque. [31]

- **Pycharm:**

C'est un IDE offre la saisie de code intelligente, des inspections de code, la mise en évidence des erreurs à la volée et des correctifs rapides [32]. Il est utilisé pour le développement de l'outil QoDB par le langage python.

- **Git et GitLab :**

GitLab est une plateforme de développement collaborative open source éditée par la société américaine du même nom. Elle couvre l'ensemble des étapes du DevOps. Elle permet de piloter des dépôts de code source et de gérer leurs

différentes versions. Son usage est particulièrement indiqué pour les développeurs qui souhaitent disposer d'un outil réactif et accessible. [33]

4.2.3 Base de données COVID19

La base de données MongoDB choisie pour cette étude c'est COVID19 version publiée le 14 Février 2022, et fournie par l'université américaine Johns-Hopkins University (JHU) [34]. D'après le développeur MongoDB Maxime Beugnet, la base de données COVID19 représente un excellent ensemble de données (dataset) à des fins éducatives et pour les projets favoris [34].

La base de données COVID19 est une combinaison de deux bases de données des statistiques de propagation de la pandémie covid19 dans les pays d'Europe.

Le tableau suivant contient les caractéristiques de la base de données Covid19 :

Nom de la base de données	Covid19		
Nombre de collections	3		
Description des collections	Benford	Benford_view	Vaccination
	1 document	193 documents	11000 documents
Nombre total des documents	11194 documents		

Table 4.1: Les caractéristiques de la base de données Covid19

4.3. Description de l'outil QoDB

Nous avons développé l'outil QoDB, qui permet de contrôler la qualité des données dans une base de données NoSQL orientée documents selon les étapes de notre méthode MFU présentée dans le chapitre précédent.

Les différentes étapes de la méthode MFU sont modélisées en différentes interfaces de l'outil QoDB.

La figure suivante présente l'interface principale de QoDB.

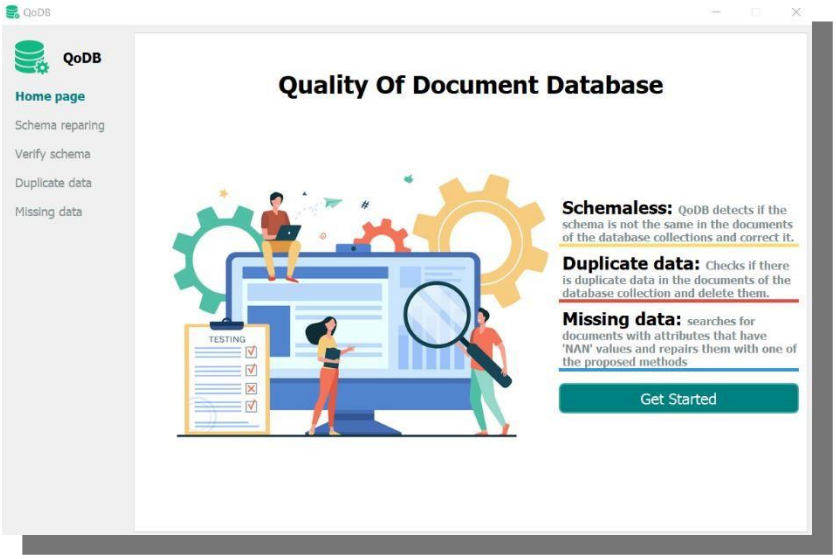

Figure 4.1. L'interface principale de l'outil

Cette interface est composée de deux parties :

- La partie gauche indique la progression de processus de validation des données décrit dans la méthode MFU, commençant par le chevauchement schématique, et finissant par la réparation des données manquantes, dans un ordre séquentiel.

- La partie droite présente le traitement et le détail de chaque étape du processus de validation.

Se connecter à MongoDB :

Pour utiliser notre outil de contrôle de la qualité, la première étape sert à se connecter à MongoDB. Ensuite sélectionner la base de données et la collection qu'on veut la contrôler le chevauchement schématique (Schema Repairing).

Figure 4.2. Se connecter à MongoDB

Après avoir se connecter à MongoDB et choisir la base de données et la collection, il s'avère nécessaire de visualiser la collection avant l'exécution de MFU.

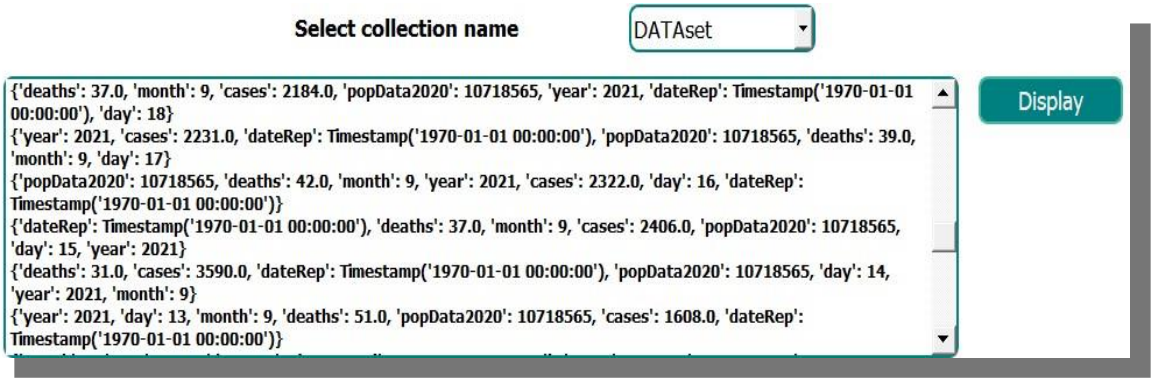

Figure 4.3. L'affichage de la collection avant le traitement

Notre processus commence par la réparation du chevauchement schématique :

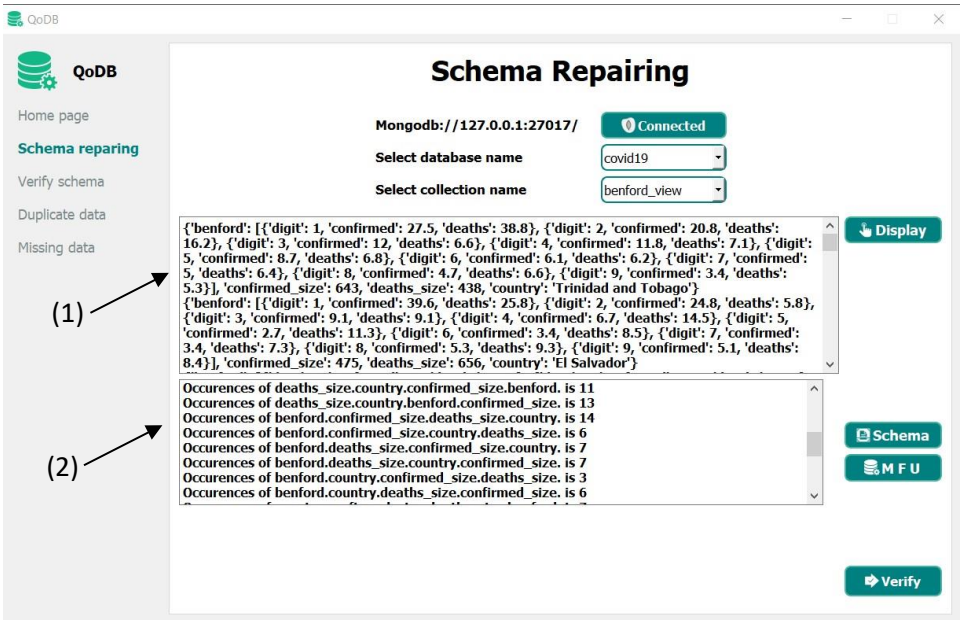

Figure 4.4 Réparation du chevauchement schématique

On applique le schéma le plus utilisé à la collection par la méthode MFU

Puis la vérification du schéma réparé, c'est-à-dire MFU parcourt toute la collection pour s'il y a un chevauchement schématique.

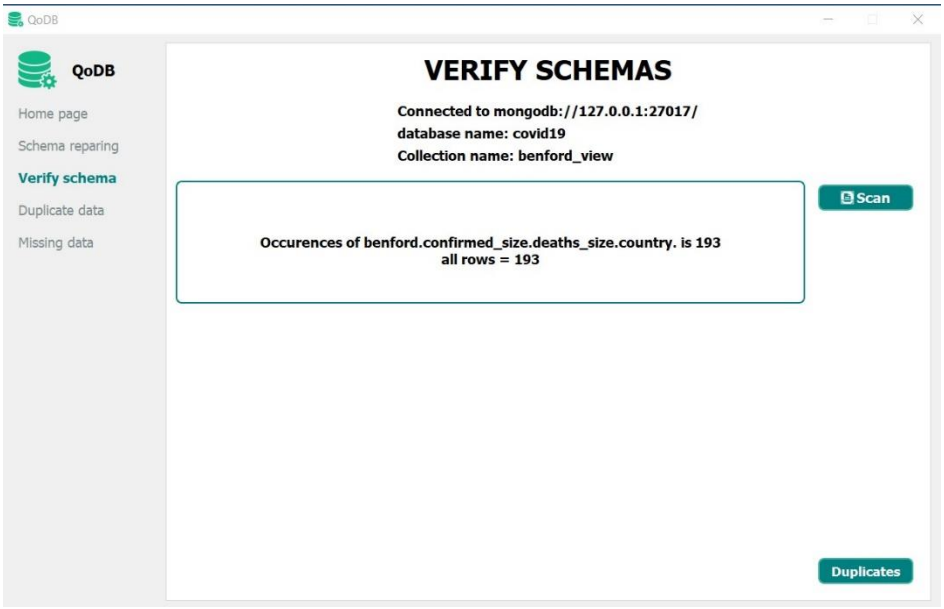

Figure 4.5 Vérification du schéma réparé

Ensuite, le traitement de la duplication : on vérifier s'il y a des champs dupliqués dans la collection, et c'est à l'utilisateur de prendre la décision de supprimer ou non.

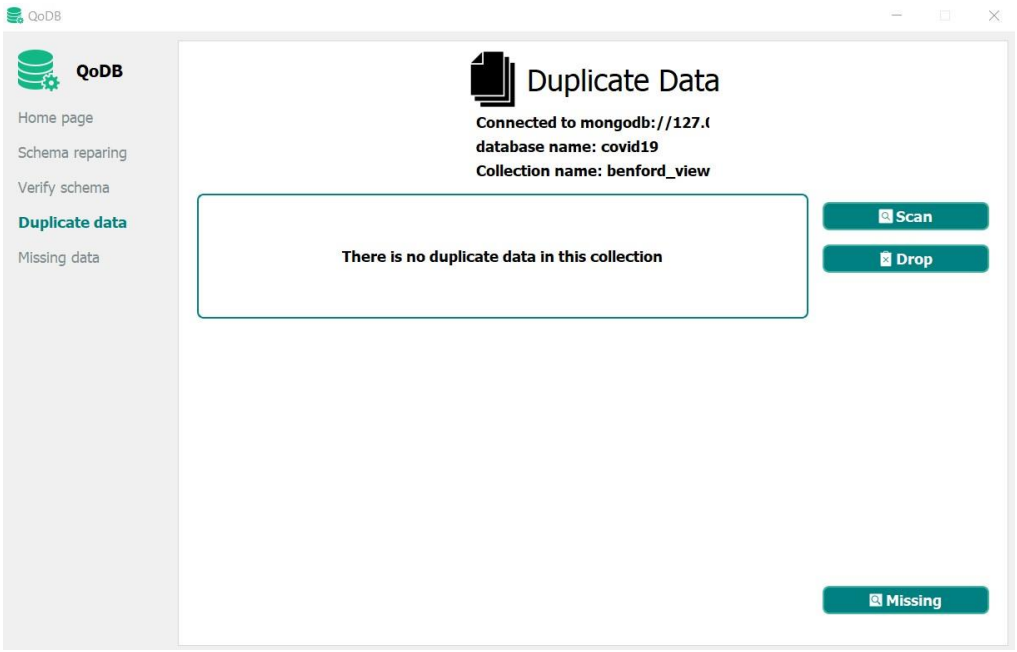

Figure 4.6 Traitement de la duplication

Pour finir avec le traitement des données incomplètes, MFU vérifie s'il y a des documents avec des valeur NAN, puis l'utilisateur choisit la méthode à appliquer pour la (MODE, MEAN, DELETE)

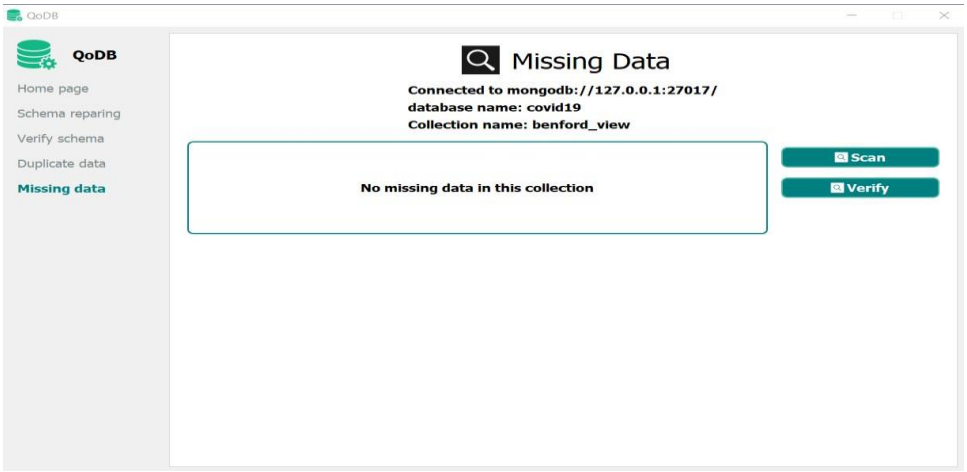

Figure 4.7 Traitement du données manquantes

4.4. Conclusion

Nous avons développé l'outil QoDB (Quality of Document Data Base) pour contrôler et réparer trois problèmes de qualité connus dans le monde des bases de données NoSQL orientées documents : le chevauchement schématique, la concision (duplication), et l'incomplétude.

L'utilisation de l'outil QoDB et les résultats obtenus montrent ses performances de contrôler la qualité de données orientée documents.

Conclusion Générale

Ce travail a pour objectif le contrôle de qualité de données NoSQL orientées documents selon trois dimensions : la consistance représentationnelle, la complétude et la concision. Ces trois dimensions de qualité affectent d'autres dimensions de qualité, comme la cohérence, la facilité d'utilisation et la facilité de compréhension. Pour cela, nous avons proposé une méthode MFU (**M**ost **F**requently **U**sed) qui contrôle et répare quelques problèmes de qualité de données NoSQL orientées documents : **le Chevauchement Schématique**", "**la duplication de données**" et "**l'incomplétude de données**"

Nous avons développé l'outil **QoDB pour** QoDB (**Q**uality **o**f **D**ocument oriented data**B**ase) pour valider et évaluer les performances la méthode **MFU**, en utilisant la base de données orientées documents MongoDB COVID19 comme données d'expérimentations.

Suite au travail réalisé, quelques perspectives sont consacrées pour les travaux futurs :

- Contrôle les types des variables dans les documents d'une collection par le principe de MFU.
- Traitement le chevauchement schématique dans les attributs imbriqués (documents imbriqués, collection des documents, liste des valeurs,…).
- Utilisation des techniques de machine learning pour la gestion des valeurs aberrantes.
- Traitement la duplication Intra-document (duplication des champs dans le même document).
- Traitement sémantique de données textuelles.

Bibliographie

[1] BEKADDOUR Abderazak & BECHLAGHEM Seyf-Allah, (2017), Etude comparative des performances des bases de données SQL et NoSQL MySQL vs MongoDB. Mémoire Master 2, Département informatique Université Abou Bakr Belkaid– Tlemcen

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

[3] Christophe (24 AVRIL 2015),Base de donnees. Consulté le 27 janvier 2022. https://www.base-de-donnees.com/acid/

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

[5] IBM Cloud Education (le 14 Novembre 2019), Théorème CAP, consulté le 28 janvier 2022, https://www.ibm.com/fr-fr/cloud/learn/cap-theorem

[6] João Ricardo Lourenço (Aout 2015) Choosing the right NoSQL database for the job: a quality attribute evaluation. Consulté le 29 janvier 2022
https://www.researchgate.net/figure/CAP-theorem-with-databases-that-choose-CA-CPand-AP_fig1_282519669

[7] A .Aggoune, M.S. Namoune. "A Method for Transforming Object-relational to Documentoriented Databases". In Proceedings of the 2nd International Conference on
Mathematics and Information Technology ICMIT'20, February 18-19, 2020, Adrar,
Algérie, pp. 154–158. IEEE. DOI. 10.1109 / ICMIT 47780 .2020. 9047011

[8] A .Aggoune, M.S. Namoune. "Practical study for handling of NoSQL data on the distributed environment systems". The 2nd Conference on Informatics and Applied Mathematics IAM'19, June 12-13, 2019, Guelma, Algérie.

[9] 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.

[10] A .Aggoune. "Switching Between Different NoSQL Databases: Approaches and Frameworks". The 3nd Conference on Informatics and Applied Mathematics IAM'20, Octobre 21-22, 2020, Guelma, Algérie.

[11] Maxime Jumelle Publié le 2 novembre 2020 Consulté le 16 février 2022 https://blent.ai/blog/debriefs/debrief-comprendre-les-bases-de-donnees-nosql

[12] Taleb, I., Serhani, M. A., & Dssouli, R. (2018, July). Big data quality: A survey. In *2018 IEEE International Congress on Big Data (BigData Congress)* (pp. 166-173). IEEE.

[13] Sidi, F., Panahy, P. H. S., Affendey, L. S., Jabar, M. A., Ibrahim, H., & Mustapha, A. (2012, March). Data quality: A survey of data quality dimensions. In *2012 International Conference on Information Retrieval & Knowledge Management* (pp. 300-304). IEEE.

[14] JURAN, J.M., GRYNA, F.M.J. and BINGHAM, R.S., 1974, Quality Control Handbook(New York: McGraw-Hill) 3rd ed.

[15] Juddoo, S. (2015, December). Overview of data quality challenges in the context of Big Data. In *2015 International Conference on Computing, Communication and Security (ICCCS)* (pp. 1-9). IEEE.

[16] Devillers, R., Bédard, Y., Jeansoulin, R., & Moulin, B. (2007). Towards spatial data quality information analysis tools for experts assessing the fitness for use of spatial data. *International Journal of Geographical Information Science*, *21*(3), 261-282.

[17] Gualo, F., Rodríguez, M., Verdugo, J., Caballero, I., & Piattini, M. (2021). Data quality certification using ISO/IEC 25012: Industrial experiences. *Journal of Systems and Software*, *176*, 110938.

[18] Batini, C., Cappiello, C., Francalanci, C., & Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, *41*(3), 1-52.

[19] Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, *12*(4), 5-33.

[20] Suárez-Otero, P., Mior, M. J., Suárez-Cabal, M. J., & Tuya, J. (2020, December). Maintaining nosql database quality during conceptual model evolution. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 2043-2048). IEEE.

[21] Chebotko, A., Kashlev, A., & Lu, S. (2015, June). A big data modeling methodology for Apache Cassandra. In *2015 IEEE International Congress on Big Data* (pp. 238-245). IEEE.

[22] Conrad, A., Möller, M.L., Kreiter, T., Mair, JC., Klettke, M., Störl, U. (2022). EvoBench: Benchmarking Schema Evolution in NoSQL. In: Nambiar, R., Poess, M. (eds) Performance Evaluation and Benchmarking. TPCTC 2021. Lecture Notes in Computer Science (), vol 13169. Springer, Cham. https://doi.org/10.1007/978-3-030-94437-7_3

[23] Möller, M. L., Klettke, M., & Störl, U. (2019). Keeping nosql databases up to datesemantics of evolution operations and their impact on data quality.

[24] Ilyas, I. F., & Chu, X. (2015). Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, *5*(4), 281-393.

[25] Hadhiatma, A. (2018, March). Improving data quality in the linked open data: a survey.
In *Journal of Physics: Conference Series* (Vol. 978, No. 1, p. 012026). IOP Publishing.

[26] Störl, U., Müller, D., Tekleab, A., Tolale, S., Stenzel, J., Klettke, M., & Scherzinger, S. (2018, April). Curating variational data in application development. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)* (pp. 1605-1608). IEEE.

[27] Missing Data | Types, Explanation, & Imputation consulté le 05 Juin 2022 https://www.scribbr.com/statistics/missing-data/

[28] https://getc.com.tn/sql-vs-nosql-quelles-differences/ consulté le 05 Juin 2022

[29] https://www.python.org/doc/essays/blurb/ consulté le 05 Juin 2022

[30] https://www.mongodb.com/fr-fr/what-is-mongodb consulté le 05 Juin 2022

[31] https://fr.acervolima.com/python-introduction-a-pyqt5/ consulté le 05 Juin 2022

[32] https://www.jetbrains.com/fr-fr/pycharm/features/ consulté le 07 Juin 2022

[33] https://www.journaldunet.com/web-tech/guide-de-l-entreprise-digitale/1443814-gitlab/ consulté le 07 Juin

[34] https://www.mongodb.com/developer/article/johns-hopkins-university-covid-19-graphqlhttps://www.mongodb.com/developer/article/johns-hopkins-university-covid-19-graphql-api/api/ Consulté le 20/03/2022