الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique Populaire
وزارة التعليم العالي و البحث العلمي
Ministère De L'enseignement Supérieur Et De La Recherche Scientifique
جامعة عمار ثليجي بالأغواط
Université Amar Telidji Laghouat
كلية العلـوم
Faculté des sciences
قسم الإعلام الآلي
Département d'informatique

Filiére : Informatique
Domaine : Matémathique et Informatique
Option : Systéme d'information décisionnel

Réaliser par:

Chouiref Safia
Graa Elbatoul

THEME

Étude comparative entre BDD Relationnelle et BDD non Relationnelle

Soutenu publiquement le 25-06-2023 devant le jury composé de:

Mr. B. ZIANI	M.C(B)	Encadreur
Mr. M. BOUZIDI	M.C(B)	Examinateur
Mr. H. MAICHA	M.C(A)	Examinateur
Mr. Y. OUINTON	M.C(B)	Président

Année Universitaire 2022/2023

REMERCIEMENT

Nous remercions Dieu le tout puissant de nous avoir donné la volonté, La patience et la santé pour accomplir ce travail.
Nous tenons à remercier notre encadreur Dr.Benameur ZIANI pour ses conseils et ses recommandations.
Nos vifs remerciements aussi pour les membres du jury qui ont accept e d'examiner et d' evaluer notre travail.
Sans oublier de remercier tous les enseignants du département d'informatique et à tous ceux qui ont contribu e de prés ou de loin à la réalisation de ce travail.

DEDICACE

Je dédie ce mémoire à mes parents bien-aimés.
Je les remercie infiniment pour leur soutien sans faille depuis toujours, leur encouragement et leurs conseils tout au long de ma vie.
À mes chers frères et à ma très chère sœur, qui ont été une source d'inspiration et d'encouragement.
À tous mes amis proches, ainsi qu'à ma collègue BATOUL pour son soutien et sa collaboration.

.

DEDICACE

Je dédie ce travail à toute ma famille, Particuliérement
à mes chers parents, qui m'avez toujours soutenus
et encouragés durant ces années d'études, je vous remercie énormément pour tous ce que vous m'avez appris dans cette vie.
À les fleurs de ma vie,Douaa,Douha et Anfel,À mon frère yassine,
merci d'être à mes côtés quand j'avais besoin de vos avis, vous êtes
ma source de motivation et d'inspiration
À ma petite Lune.
À mes chéres Nour, Khouala et Batoul d'avoir toujours été là pour moi.
Je tiens surtout à dédier ce travail à mon binôme et mon amie Safia de m'avoir accompagné durant toutes ces années d'université.

Résumé

À l'ère de la révolution numérique, la gestion efficace des données est devenue essentielle pour les entreprises et les organisations. Avec la croissance exponentielle des volumes de données générés à partir de sources diverses et complexes, il est essentiel d'avoir des systèmes de gestion de données adaptés pour garantir la disponibilité, l'accessibilité et la fiabilité des informations. Dans ce contexte, les bases de données relationnelles (SQL) et les bases de données non relationnelles (NoSQL) offrent des approches différentes pour la gestion et l'exploitation des données. L'objectif de cette étude est d'analyser et expliquer les caractéristiques, les performances et les avantages de chaque modèle. Nous examinons les aspects tels que la structure des données, les requêtes, la scalabilité et la flexibilité des deux types de bases de données.
Mots clés:Bases de données NoSQL, bases de données SQL, SGBDR, MongoDB, oriente documents, orienté colonnes, orienté graphes, clé-valeur, CAP, ACID.

Abstract In the era of the digital revolution, efficient data management has become essential for businesses and organizations. With the exponential growth of data volumes generated from diverse and complex sources, it is crucial to have suitable data management systems to ensure the availability, accessibility, and reliability of information. In this context, relational databases (SQL) and non-relational databases (NoSQL) offer different approaches to data management and utilization. The objective of this study is to analyze and explain the characteristics, performance, and advantages of each model. We examine aspects such as data structure, queries, scalability, and flexibility of both types of databases.
Keywords: NoSQL Databases, SQL Databases, DBMS, Document Store, Column-Family Store, Graph Store, Key-Value Store, MongoDB, CAP, ACID.

ملخص:

في عصر الثورة الرقمية، أصبح إدارة البيانات بفعالية أمرًا أساسيًا للشركات والمنظمات. مع الزيادة الهائلة في حجم البيانات الناتجة من مصادر متنوعة ومعقدة، من الضروري أن يكون لديك أنظمة إدارة بيانات مناسبة لضمان توفر وسهولة الوصول والموثوقية في المعلومات. في هذا السياق، تقدم قواعد البيانات العلاقية SQL وقواعد البيانات غير العلاقية NOSQL نهجًا مختلفًا لإدارة واستغلال البيانات. هدف هذه الدراسة هو تحليل وشرح ميزات وأداء وفوائد كل نموذج. نحن نستعرض جوانب مثل هيكل البيانات، واستعلامات البيانات، وقابلية التوسع والمرونة للنوعين من قواعد البيانات.
الكلمات المفتاحية: قواعد بيانات NOSQL ، قواعد بيانات SQL ، نظام إدارة قواعد البيانات العلائقية (SGBDR)، قاعدة بيانات موجهة حسب الوثائق، قاعدة بيانات موجهة حسب الأعمدة، قاعدة بيانات موجهة حسب الرسومات (الجرافات)، مفتاح قيمة، MongoDB ، CAP ، ACID .

Contents

List of Figures

List of Tables

SGBD système de gestion de base de données

BDD base de données

ACID Atomicité, Cohérence, Isolation, Durabilité

transaction une suite d'opérations qui font passer la base de données d'un état A - antérieur à la transaction - à un état B postérieur.

JSON JavaScript Object Notation

BDNOSQL Base de données Non relationnel

BDR base de données relationnel

API Application Programming Interface

CSV Coma Separated Values

CRUD Create (créer), Read ou Retrieve (lire), Update (mettre à jour), Delete ou Destroy (supprimer)

CAP consistency, availability, Partition aux tolerance

CA consistency, availability

CP consistency, Partition aux tolerance

AP availability, Partition aux tolerance

BA Basically Available

NF Normal Form

CPU Central Process Unit

RAM Random Access Memory

Chapter 1

INTRODUCTION GENERALE

Depuis leur apparition dans les années 70 du dernier siècle, les systèmes de bases de données ont considérablement évolué. Dans de tels systèmes, le modèle relationnel a été pour longtemps le choix par défaut pour le stockage de données, en particulier pour les applications commerciales. Ceci est dû principalement à sa puissance de modélisation et aux langages d'interrogation riche et facile à utiliser.

Aujourd'hui, avec la croissance des capacités de stockage, nous assistons à une explosion du volume de données recueillies dans les organismes dans différents domaines tels que les ventes, la médecine, les finances et les domaines sociaux. Ceci a entraîné une évolution des bases de données relationnelles vers les bases de données massives, connues aussi sous le terme Big data, en raison de leur volume, de leur vélocité avec laquelle elles sont collectées et de leur variété de formes (semi-structurées ou non structurées). De telles bases de données sont généralement stockées dans des systèmes dits NoSQL (Not Only SQL). Les bases de données NoSql se répartissent en quatre catégories : colonnes, documents, graphes et clé-valeur. Chacun de ces types offre des fonctionnalités spécifiques et est conçu pour des cas d'utilisation particuliers.

Par conséquent, le choix de la technologie de stockage à mettre en place peut influer les performances des systèmes de bases de données. L'objectif de ce mémoire est de décrire, dans un premier temps, les avantages et inconvénients des deux systèmes. Nous présentons par la suite une étude comparative expérimentale. Plus précisément; nous nous focalisons sur les bases orientés documents et plus particulièrement la base MongoDB qui figure parmi les plus utilisés actuellement. Cette base s'appuie sur le format JSON où Un "document" a une structure arborescente qui contient une liste de champs qui ont une valeur et qui peuvent à leur tour contenir une structure avec une liste de champs et ainsi de suite. Les documents sont regroupés en collections mais il n'y a pas de type imposé pour les documents d'une collection. Cette organisation des données est couplée à des opérateurs qui permettent d'accéder aux données imbriquées.

Afin d'estimer les performances des deux système, nous avons manipuler des données de différentes tailles avec les opérations de base: création, lecture, mise à jour et suppression. Ce manuscrit est organisé comme suit:

- Le chapitre 1 constitue une introduction à notre travail.
- Le chapitre 2 présente le modèle relationnel des données. L'accent est mis en particulier sur les avantages et les limites de ce modèle.
- Le chapitre 3 est consacré aux systèmes NoSql. Il illustre les différentes représentations des données ainsi que les avantages et les limites de cette technologie.

- Le chapitre 4 présente la partie expérimentale et les différents résultats obtenus.
- Le chapitre 5 présente une conclusion générale et dresse quelques perspectives du travail réalisé.

Chapter 2

Les base de données relationnelle

2.1 introduction

Le terme de modèle relationnel désigne une manière de stocker et structurer les données sous la forme de tables ou relations. Ce modèle, très simple, est de loin le plus répandu dans les Systèmes de Gestion de Bases de Données (SGBD).
Nous parlerons dans ce chapitre sur les bases de données relationnelle historiquement,comment ça fonctionne et quelles sont les caractéristique les plus importantses de ce type de base de données.Enfin, les avantages et les inconvénients.

2.2 les bases de données relationnelle

En 1970,Edgar Frank Codd a introduit pour la première fois les modèles relationnels. Les bases de données relationnelles sont un ensemble structuré de données avec des relations entre elles. Ces données sont bien organisées dans des tableaux, composés de colonnes et de lignes. Chaque colonne présente un attribut et les attributs du même tableau partagent des propriétés similaires. Les lignes, qui représentent des enregistrements, peuvent être reliées entre elles par des clés étrangères. Ces données sont gérées par un système de gestion de bases de données relationnelles (Relational Database Management System, RDBMS). Le langage SQL (Structured Query Language) permet de gérer et de manipuler les bases de données relationnelles.[9]

2.2.1 la normalisation

La normalisation est l'étape la plus importante de la conception d'une base de données relationnelle, et elle se compose de trois étapes principales: Première forme normale (1NF) :si tout les attribut contient une valeur atomique .
Deuxième forme normale (2NF) : si elle est en première forme normale et si chaque attribut non clé dépend de la clé primaire.
Troisième forme normale (3NF) : Une relation est en troisième forme normale si, elle est en deuxième forme normale et siles attributs non clés ne dépendent pas transitivement des autres attributs non clés.[10]

2.2.2 ACID

Le terme "transaction" désigne un ensemble d'opérations qui apportent des modifications aux données d'une base de données.
ACID : l'Atomicité, la Cohérence, l'Isolation et la Durabilité ces quatre principes assurent la fiabilité de la transaction. **Atomicité :**Une transaction peut consister en plusieurs tâches, et l'atomicité implique que soit toutes ces tâches sont exécutées avec succès, soit aucune modification n'est apportée à la base de données. En d'autres termes, chaque transaction est traitée comme une unité indivisible. Si l'une des tâches échoue, toutes les modifications effectuées jusqu'à ce point sont annulées, et la base de données reste dans son état initial.[13]
Cohérence: La propriété de cohérence dans le contexte des transactions fait référence au respect de toutes les contraintes d'intégrité du système. Cela signifie que lorsqu'une transaction est exécutée, toutes les règles et contraintes définies dans la base de données

doivent être respectées.. [13]
Isolation : La troisième propriété, la propriété d'isolation, signifie que même s'il y a concurrence, le résultat de la transaction doit être le même que si les transactions sont séquentielles. Les différentes transactions concurrentes ne doivent pas avoir d'impact les unes sur les autres.[13]
Durabilité : La durabilité signifie qu'aucune des transactions réussies ne doit être perdue, même en cas de défaillance du système. [13]

2.3 Les avantages du modèle relationnel

les avantages du modèle relationnel sont:

2.3.1 Cohérence des données

Le modèle relationnel est le plus éfficace pour la cohérence des données entre la base de données et ces instances, elle assure que plusieurs instances d'une base de données contiennent les mêmes données à tout moment.[2]

2.3.2 Les procédures stockées

Les procédures stockées sont des éléments essentiels des bases de données relationnelles. Elles permettent de stocker et d'exécuter des blocs de code réutilisables directement au sein de la base de données. Les procédures stockées sont généralement écrites dans des langages de programmation spécifiques aux bases de données, tels que SQL, et peuvent être appelées par des applications ou d'autres procédures stockées. L'un des avantages des procédures stockées est leur capacité à réduire la duplication de code. Plutôt que d'écrire la même requête encore et encore dans différentes parties de l'application, la logique de la requête peut être encapsulée dans une procédure stockée. Cela permet de réutiliser le code existant et de simplifier le processus de développement en évitant la duplication.[2]

2.3.3 Verrouillage de la base de données et simultanéité

les techniques de verrouillage et de simultanéité jouent un rôle essentiel dans la préservation de l'intégrité des données et la gestion des conflits au sein des bases de données. Le verrouillage permet de contrôler l'accès concurrentiel aux données lors des mises à jour, tandis que la simultanéité garantit que les transactions concurrentes s'exécutent de manière cohérente. Les choix de mise en œuvre de ces techniques peuvent varier selon les systèmes de gestion de bases de données, avec des implications sur les performances et la granularité de l'accès concurrentiel. [2]

2.3.4 Sécurité

Dans les bases de données relationnelles, le concept d'utilisateurs et de droits d'utilisateurs est utilisé pour gérer l'accès et les autorisations des utilisateurs aux données. Chaque utilisateur est identifié par un nom d'utilisateur et peut se voir attribuer différents droits et privilèges en fonction de ses besoins et de son rôle dans l'organisation.
Les droits d'utilisateurs dans les bases de données relationnelles sont associés aux relations, qui sont les tables contenant les données. Chaque relation peut avoir des privilèges spécifiques qui définissent les actions que les utilisateurs sont autorisés à effectuer sur cette relation.
Voici quelques exemples de privilèges couramment utilisés dans les bases de données relationnelles :
Le privilège de création (CREATE) : autorise un utilisateur à créer de nouvelles relations ou tables dans la base de données.
Les privilèges de sélection (SELECT) : autorisent un utilisateur à interroger et consulter les données dans une relation spécifique.
Les privilèges d'insertion (INSERT) : autorisent un utilisateur à insérer de nouvelles données dans une relation spécifique.
Les privilèges de suppression (DELETE) : autorisent un utilisateur à supprimer des données d'une relation spécifique. [1]

2.3.5 Vues dynamiques

les vues dynamiques sont un outil puissant offert par les bases de données relationnelles. Elles permettent d'améliorer les performances des requêtes, de simplifier les relations complexes, de stocker les résultats des requêtes et de renforcer la sécurité des données.[1]

2.3.6 Non-redondance des données

la non-redondance des données est un principe fondamental dans les bases de données relationnelles. Elle garantit que chaque donnée est stockée une seule fois, ce qui améliore l'efficacité, l'intégrité et la cohérence des informations dans la base de données.[15]

2.3.7 Résistance aux pannes

la résistance aux pannes est un aspect essentiel des bases de données relationnelles, et des mécanismes appropriés sont mis en place pour restaurer les données dans un état cohérent après une panne et assurer la continuité des opérations.[15]

2.4 Les limites du modèle relationnel

cette approche connait quelques limites à savoir :

2.4.1 Modélisation des entités réelles

Dans la modélisation des bases de données relationnelles, il est courant de représenter des entités du monde réel sous forme de relations. Cependant, cette approche ne convient pas toujours aux objets complexes ou aux cas où les relations se multiplient à la suite du processus de normalisation.

2.4.2 Scalabilité limitée

les SGBDR relationnelles peuvent présenter des limites en termes de scalabilité lorsqu'il s'agit de gérer de grandes quantités de données et des charges de travail élevées. [9]

2.4.3 Des propriétés ACID en milieu distribué

Les propriétés ACID, bien que nécessaires à la logique du relationnel, nuisent fortementaux performances surtout la propriété de cohérence. En effet, la cohérence est très difficile à mettre en place dans le cadre de plusieurs serveurs (environnement distribué), car pour que celle-ci, pour pouvoir satisfaire cette cohérence, il faut que tous les serveurs doivent être des miroirs les uns des autres, de ce fait deux problèmes apparaissent :

- Le coût en stockage est énorme car chaque donnée est présente sur chaque serveur.
- Le coût d'insertion/modification/suppression est très grand.[15].

Chapter 3

les bases de données non relationnelle(NOSQL)

3.1 les bases de données non relationnelle(NOSQL)

En 1998, Le terme NoSQL a été introduit pour la première fois par carlo Strozzi pour désigner une base de données relationnelle open source qui n'utilisait pas SQL. Une base de données non relationnelle, également connue sous le nom de NoSQL (Not Only SQL), est un type de système de gestion de base de données qui ne suit pas le modèle relationnel traditionnel. Les bases de données non relationnelles peuvent stocker des données sous diverses formes, telles que des documents, des graphiques, des clés-valeurs ou de colonnes, et elles sont souvent conçues pour traiter des quantités massives de données, avec une grande évolutivité et une haute disponibilité. Les bases de données non relationnelles n'utilisent pas de schéma prédéfini pour organiser les données, ce qui permet une grande flexibilité dans la façon dont les données sont stockées et traitées. Cependant, cela peut également rendre plus difficile l'interrogation des données et l'assurance de la cohérence des données. Les bases de données non relationnelles sont souvent utilisées pour des applications web et mobiles, ainsi que pour des cas d'utilisation tels que l'analyse de données.

3.2 le principe et la conception de NOSQL

3.2.1 Le théorème CAP

également connu sous le nom de théorème de Brewer, cette theorie est basé sur l'impossibilité de garantir simultanément trois caractéristiques.
La cohérence:Lorsqu'une mise à jour est effectuée, toutes les données réparties dans un système de base de données sont mises à jour de manière simultanée.
La disponibilité:fait référence à la capacité d'un système de base de données distribuées à fournir un accès continu aux données, même en cas de défaillance de certaines parties du système.
La tolérance aux partitions: C' est la capacité d'un système de base de données distribué à fonctionner même en cas de défaillance de certaines parties du système.

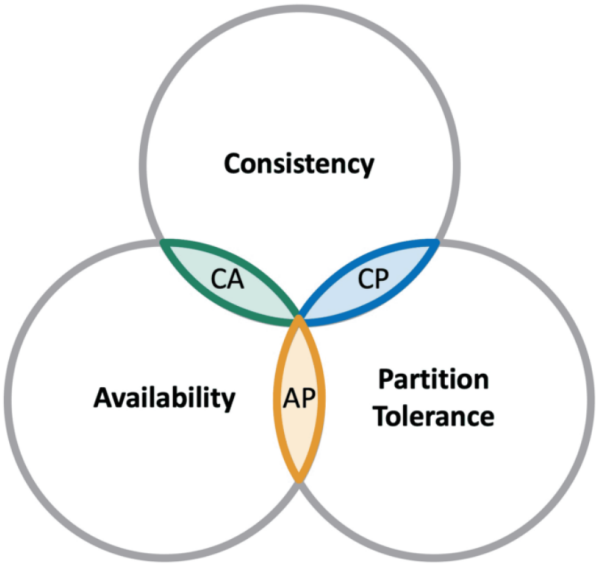


Figure 3.1: théoréme de CAP[6]

-CA : Les données sont consistantes entre tous les noeuds. Toutes les lectures/écritures les noeuds concernent les mêmes données. Mais si un problème de réseau apparait, certains noeuds seront désynchronisés au niveau des données (et perdront donc la consistance). Nous ne pouvons en respecter seulement deux à la fois.[14]

- **CP** : Les données sont consistantes entre tous les noeuds et le système possède une tolérance aux pannes, mais il peut aussi avoir des problèmes de latence ou plus généralement, de disponibilité .[14]
- **AP** : Le système affiche des performances élevées et une résilience face aux pannes, mais il ne peut pas garantir la cohérence des données entre les différents nœuds.[11]

- Les SGBD NoSQL sont des systèmes CP (Cohérent et Résistant au partitionnement) ou AP (Disponible et Résistance au partitionnement).

3.2.2 Les propriétés BASE

L'acronyme de BASE a été définie par Eric Brewer, qui est également connu pour formuler le théorème CAP. Les propriétés de BASE sont également utilisées dans le contexte des bases de données NoSQL pour décrire les caractéristiques des systèmes de gestion de données qui privilégient la disponibilité et la tolérance aux partitions plutôt que la cohérence forte. Le principe de BASE se compose comme suit :

- **BA** pour Basically Available (disponibilité de base) : Cela signifie que le système doit être toujours disponible, même en cas de panne de certains nœuds du système.

- **S** pour Soft-state (état souple) : Les données stockées dans un système NoSQL peuvent être dans un état souple, ce qui signifie qu'elles ne sont pas nécessairement cohérentes en temps réel, mais elles sont mises à jour progressivement.

- **E** pour Eventually Consistent (cohérence éventuelle) : cela signifie que le système finira par atteindre un état cohérent, même si cela prend du temps en raison de la propagation asynchrone des mises à jour.

Ces propriétés permettent aux systèmes NoSQL de gérer des charges de travail massives et d'offrir une haute disponibilité tout en réduisant les coûts de gestion de données.

3.2.3 les types de base de données NOSQL

Les bases de données NoSQL peuvent être divisées en quatre types pour satisfaire des cas d'utilisation et des besoins.
Clé-valeur:
Les bases de données de type clé-valeur moins complexes et accès rapide aux informations. Son principe est très simple, chaque valeur stockée est associée à une clé unique.
Les implémentations les plus populaires sont : **Riak, Redis, Voldemort.**

Figure 3.2: Base de données clé-valeur[4]

Orientées document:
Les bases de données de type Orientées document sont utilisées pour gérer les données semi-structurées. Les systèmes de type documentaire sont composés de collections de documents. La représentation en document est une sorte d'extension du concept clé/valeur. La valeur est représentée sous forme de document contenant des données. Les types de fichier de documents en Orientées document sont:**JSON,XML**
Les implémentations les plus populaires sont:**MongoDB,CouchDB**.

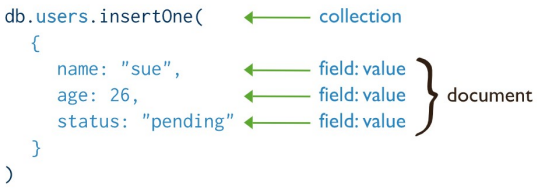


Figure 3.3: Base de données orientée document[4]

Orientée graphes:
Les bases de données de type Orientées graphes se base sur la théorie des graphes et s'appuie sur la notion de nœuds, composé par plusieurs nœuds et chaque nœud possède plusieurs attributs et entre chaque nœuds il y a des relations. Ce modèle facilite la représentation du monde réel comme les réseaux sociaux. Les implémentations les plus populaires sont:**Neo4j, Hypergraph DB et FlockDB.**

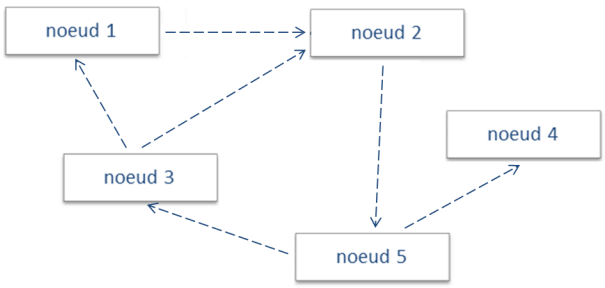


Figure 3.4: Base de données orientée graphe[4]

Orientée colones:
Les bases de données de type orientées colonnes est un système de gestion de base de données (SGBD) qui stocke les tableaux de données par colonne et non par ligne. Le nombre de colonnes est dynamique.
Les implémentations les plus courantes sont:**HBase ,Cassandra.**

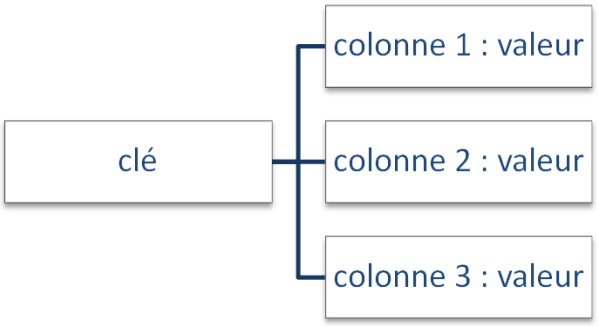


Figure 3.5: Base de données orientée colone[4]

3.2.4 Les avantages de nosql

Comme la plupart des outils de stockage de données, la base de données NoSQL présente une grande série d'avantages, les bases de données NoSQL sont plus évolutives et offrent des performances supérieures, et leur modèle de données corrige plusieurs faiblesses du modèle relationnel Il s'agit d'un des principaux points forts de NoSQL:

- **La scalabilité horizantale:** également connu sous le nom croissance externe. Le principe de la scalabilité horizontale consiste à simplement rajouter des serveurs en parallèle. On part d'un serveur basique et on rajoute des nouveaux serveurs identiques afin de répondre à l'augmentation de la charge.[8]

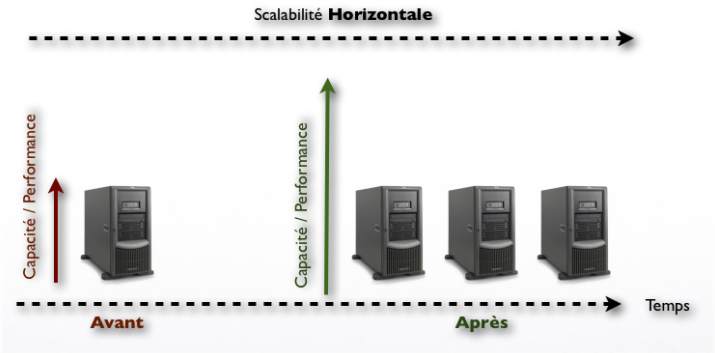


Figure 3.6: scalabilité_horizantale[5]

- **Flexibilité:** Cette flexibilité permet aux utilisateurs de stocker des données semi-structurées ou non structurées dans la base de données, sans avoir à respecter un schéma strict de données. Les données peuvent être ajoutées ou modifiées sans nécessiter de modifications au schéma, ce qui permet une plus grande agilité et une plus grande rapidité dans le développement de l'application.

- **Performance en écriture :** Les géants de la technologie comme Google, Facebook et Twitter, gérant des masses de données qui augmentent considérablement chaque année. Pour réduire le temps nécessaire au stockage des données, ils distribuent l'écriture sur un groupe de machines, en utilisant des techniques telles que MapReduce, la réplication, la tolérance aux pannes et la cohérence. [12]

- **Performances en lecture**: Les données sont distribuées sur plusieurs machines ce qui permet une lecture trés rapide et éfficace de grandes quantités de données.
- **Haute disponibilité** : Les SGBD NoSQL sont conçus pour être hautement disponibles. Ils peuvent répliquer des données sur plusieurs nœuds de serveur pour offrir une tolérance aux pannes. Cela signifie que si un nœud de serveur échoue, les autres nœuds peuvent continuer à fonctionner normalement, réduisant ainsi les temps d'arrêt et les pertes de données.
- **Coût** : Les SGBD NoSQL sont souvent moins coûteux que les SGBD relationnels en termes de licences et de matériel requis. Les SGBD NoSQL sont souvent open source, ce qui signifie qu'ils sont gratuits à télécharger et à utiliser, tandis que les SGBD relationnels peuvent être coûteux en termes de licences et de frais de maintenance.
- **Évolutivité horizontale** : Les SGBD NoSQL ont été conçus pour gérer des données volumineuses et offrent une évolutivité horizontale facile. Cela signifie qu'il est possible d'ajouter des nœuds de serveur supplémentaires pour gérer la charge et la taille des données sans avoir besoin de modifier le schéma de la base de données.

3.3 les inconvénients de nosql

Les avantages apportés par les bases de données NoSQL ne sont pas sans contreparties. En effet comme toute avancée technologique les bases de données NoSQL ont aussi des inconvénients, on en citera :

- **Connaissance technique :**Les systémes de gestion de base de données NoSQL sont généralement plus complexes que les systémes de gestion de base de données relationnels traditionnels, ce qui peut nécessiter une expertise technique plus avancée ainsi que des efforts conséquents lors de leur maintenance pour les configurer et les utiliser efficacement.[8]
- **Limitations de requêtes :** Les requêtes dans les systèmes de gestion de bases de données NoSQL sont souvent conçues pour effectuer des opérations simples et rapides sur les données, ce qui peut entraîner certaines limitations en termes de types de requêtes complexes .
- **Conflits de données :** Les SGBD NoSQL répliquent souvent des données sur plusieurs nœuds de serveur pour offrir une haute disponibilité et une tolérance aux pannes. Cependant, cela peut parfois entraîner des conflits de données entre les différents nœuds.
- **Manque de maturité :** Certains SGBD NoSQL peuvent être relativement nouveaux et manquer de maturité en termes de fonctionnalités et de stabilité par rapport aux SGBD relationnels plus établis.
- **Difficulté de migration :** La migration des données depuis un SGBD relationnel traditionnel vers un SGBD NoSQL peut être complexe et nécessite

une révision et une vérification complète de la structure des données.

3.4 conclusion:

En conclusion, les bases de données NoSQL sont des solutions de gestion de données flexible et evolutives qui offrent des avantages par rapport aux bases de relationnelles traditionelles.

Chapter 4

IMPLEMENTATION ET EXPERIMENTATION

4.1 Environnement de travail

4.1.1 Microsoft SQL Server

Microsoft SQL Server est une base de données relationnelle qui stocke les données sous forme de tables et utilise le langage SQL pour communiquer avec la base de données. Il permet de créer des requêtes simples et complexes afin d'extraire et de structurer les données selon vos besoins, ainsi que d'établir des relations entre les tables.
Pour mesurer les performances des requêtes SQL, nous avons utiliseé SQL-Querystress.
SQLQuerystress est un outil de test de performance simple et léger, conçu pour tester la charge de requêtes de sql server.

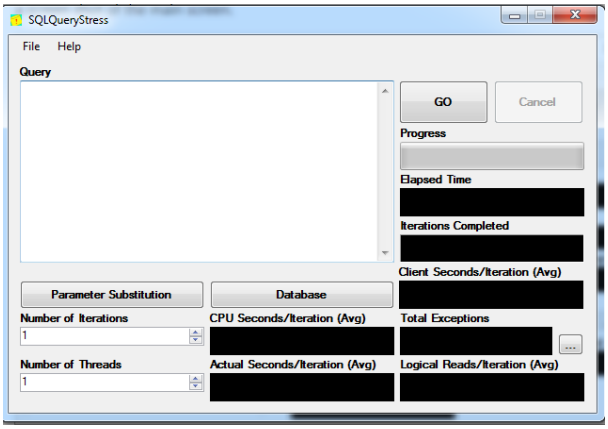


Figure 4.1: Capture d'ecran de SQLQueryStress

4.1.2 MongoDB

MongoDB est un système de gestion de base de données (SGBD) orienté document. Il a été lancée en 2009. Dans MongoDB, les données sont stockées sous forme de documents JSON (JavaScript Object Notation) MongoDB possède des tables comme les bases de données relationnelles, mais elles sont appelées collections, et chaque collection contiens un ou plusieurs documents. Un document est une structure de données composée de paires clé-valeur, où les valeurs peuvent être de différents types comme:chaînes de caractères, des nombres, des tableaux, des objets imbriqués, etc.

4.1.3 Studio 3T

Studio 3T est une GUI de mongodb utilisee pour exporter ou importer des collections, des vues ou encore des requêtes de mongodb.
Pour nous, nous avons utilisé Studio 3T pour la migration de MongoDB vers SQL, il est considéré comme le moyen le plus simple pour migrer entre SQL et MongoDB. Importez Oracle, PostgreSQL, MySQL, SQL Server,

Sybase et IBM DB2 vers MongoDB, ou exportez les collections MongoDB vers un fichier SQL ou une base de données.
Mockaroo: est une plateforme en ligne qui permet d'offrir un service gratuit pour générer des données de test aléatoire et réaliste et contient plusieurs format des fichiers: JSON, CSV, EXCEL, SQL, CASSANDRA...

Matérielle Notre étude a été menée dans deux machines physique avec la configuration suivante :
1 ere machine :

* Processeur : Intel® Core(TM) i3-2348M CPU @ 2.30 GHz
* RAM : 4.00 Go
* Système : Microsoft Windows7 Professional 64 bits
* Disque dur: 4.00 Go

2 eme machine :

* Processeur : Intel® Core(TM) i7-5500U CPU @ 2.40 GHz
* RAM : 8.00 Go
* Système : Microsoft Windows10 Professional 64-bits
* Disque dur: HDD 207GO

4.2 Approche de migration d'une base de données relationnelle vers une base de données non relationnelle

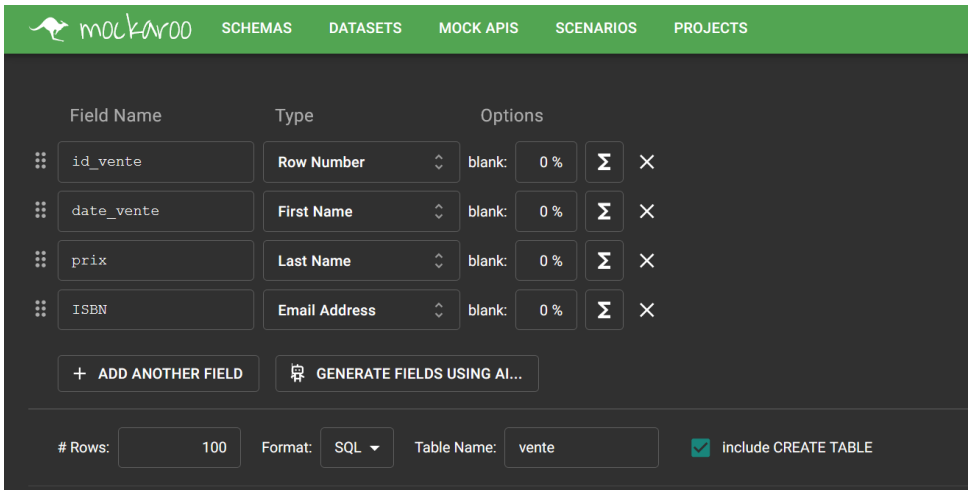
La migration des bases de données est un processus qui consiste a convertir toutes les composantes de la base de données source (BDR) vers leur équivalent dans la base de données cible (NoSQL). Cela inclut schéma de données (le modèle physique de données), les données elles-mêmes(les enregistrements stocke dans les tables), les procédures stockées, et les vues.[9]

1. Traduction du modèle de données source en modèle cible: Cette étape implique l'analyse du schéma SQL et du modèle de données actuels pour identifier les problèmes potentiels. Il peut s'agir de revoir les relations entre les données, dénormaliser ou répartir les données sur plusieurs tables.
2. Conversion de données de la BDR à la BDNOSQL: Cette phase comprend l'extraction des données de la base de données source en sélectionnant les champs pertinents, le traitement ou l'enrichissement des données extraites, et enfin, l'injection des données dans la base de données cible, qui est une base de données NoSQL.

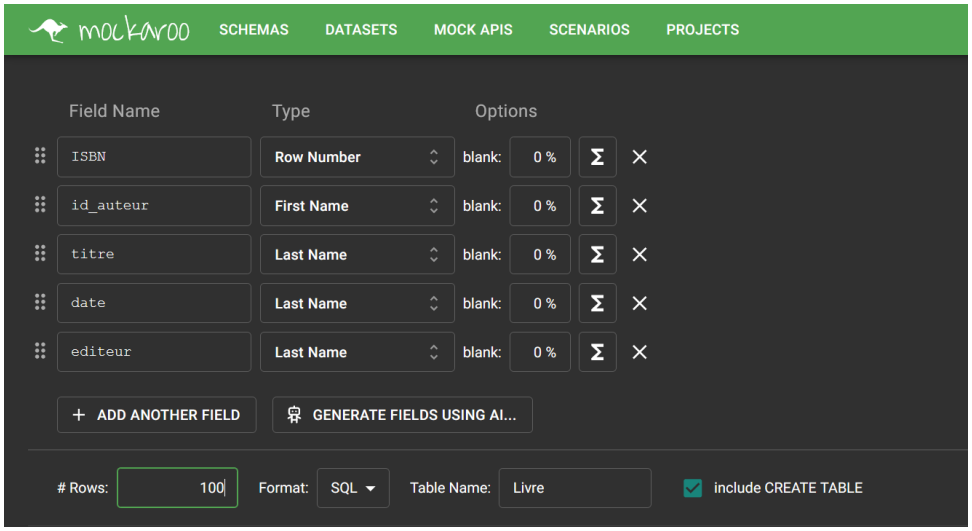
4.2.1 Exemple

La première base de données que nous avons utilisée concerne les ventes de livres. Cette base de données contient des informations sur les ventes de livres, telles que les titres des livres, les auteurs, les éditeurs, les dates de vente. La démarche que nous avons suivre pour faire la migration est la suivante : on a fait la conception: pour définir les table .

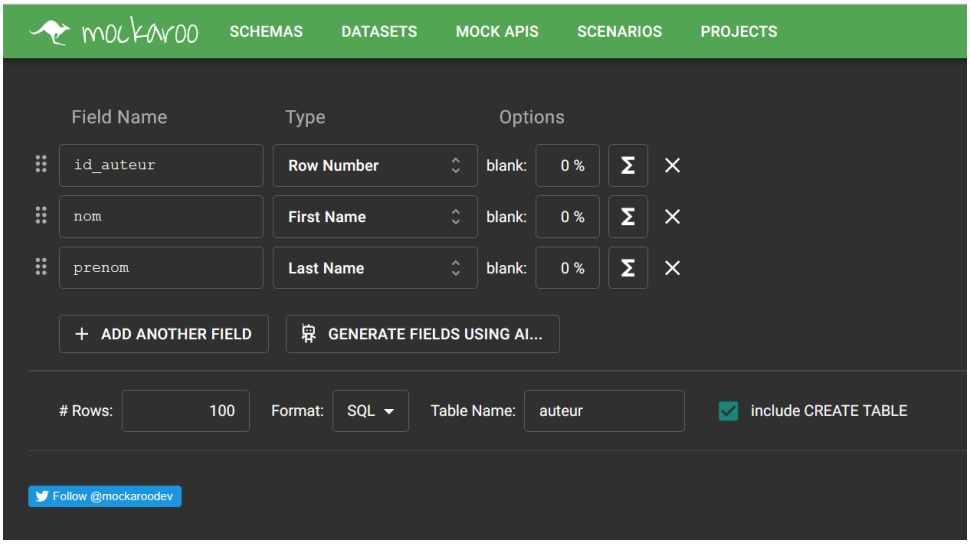
Après la création des tables de la base de données, nous avons utilisé un générateur de données (Mockaroo est un outil qu'il falait définir dans la section précédente) pour remplir toutes les tables avec les données correspondantes.
la table vente:



la table livre:



la table auteur



. Les images ci-dessus représentent les trois tables au format SQL, chacune contenant 100 lignes de données. Après avoir généré ces données, nous les importons dans SQL Server. Ensuite, nous effectuons une transformation des données au format JSON et les importons dans MongoDB.
fichier JSON:


```
[{
  "_id": {
    "$oid": "63f94d87342d1b80d5d399ef"
  },
  "id_vente": 1,
  "date": "02/01/2021",
  "prix": 3.25,
  "livre": {
    "isbn": {
      "$numberLong": "9782844871435"
    },
    "titre": "Love Story a Abidjan",
    "auteur": {
      "id_auteur": 100,
      "nom": "Assaley",
      "prenom": "-N"
    },
    "date": "28/11/2002"
  }}
```


Figure 4.2: fichier JSON

4.3 Approche de migration d'une base de données non relationnelle vers une base de données relationnelle

Elle est composè d'une analyse detaille de la structure de la base de donnèes non relationnelle. Ensuite, une modelisation conceptuelle est realiser pour identifier les entites, les relations et les attributs qui seront utilisès dans la base de données relationnelle. Une fois la modélisation conceptuelle ètablie, la migration des donnèes peut commencer, cette étape C'est-à-dire l'extraction des donnèes de la bdd non relationnelle leur transformation et leur chargement dans la nouvelle base de données relationnelle. Des outils et des techniques spécifiques sont utilisés pour faciliter ce processus comme (Studio 3T).

4.3.1 Exemple

La deuxième base de données (fichier JSON) que nous avons utilisée concerne les listes des chaînes YouTube les plus populaires sur YouTube. Ce fichier JSON est composé des colonnes suivantes et sa taille est de 121.203 MO, qui ont été obtenues en utilisant l'API YouTube.Le fichier JSON contient les colonnes suivantes, qui sont obtenues à l'aide de l'API YouTube. [3]

Le fichier JSON contient les colonnes suivantes, qui sont obtenues à l'aide de l'API YouTube :

colonnes	type
total views channel elapsed time	décimal
channelid	chaîne
videocategoryid	entier
channelviewcount	entier
likessubscriber	décimal
viewssubscribers	décimal
videocount	entier
compte d'abonnés	entier
videoid	chaîne
dislikesviews	décimal
channelelapsedtime	entier
commentairesabonné	décimal
likesviews	décimal
channel comment count	entier
videos viewcount	entier
likes dislikes	décimal
comments views	décimal
totvideosvideocount	décimal
elapsed time	entier
videolike count	entier
videos dislike count	nombre entier
dislikes subscriber	décimal
totviewstotsubs	décimal
views elapsed	temps décimal
vidéo publiée	datehorodatage
videos commentcount	nombre entier

Table 4.1: Les colonnes du fichier JSON

Une fois que le fichier JSON a été importé dans MongoDB et que la connexion a été établie avec Studio 3T, nous pouvons commencer la migration.

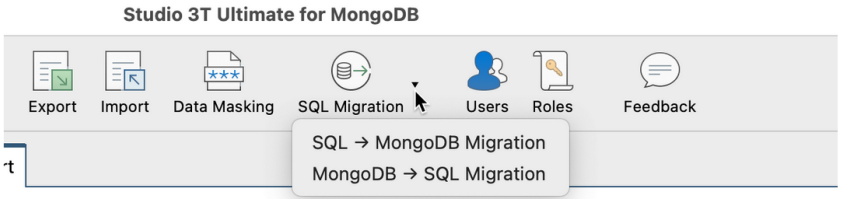


Figure 4.3: migration dans studio 3T [7]

Après la connexion à MongoDB, nous avons sélectionné la base de données et la collection que nous souhaitions migrer.

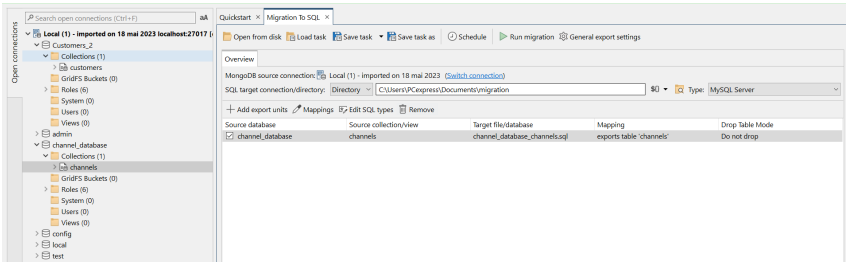


Figure 4.4: BDD et collection

Ensuite, nous avons effectué le mapping des données, c'est-a-dire associer les champs de la collection MongoDB aux colonnes de la base de données relationnelle cible. Une fois le mapping terminé et les données migrer avec succès dans la table SQL cible, nous avons téléchargé la table et l'avons importée dans SQL server.

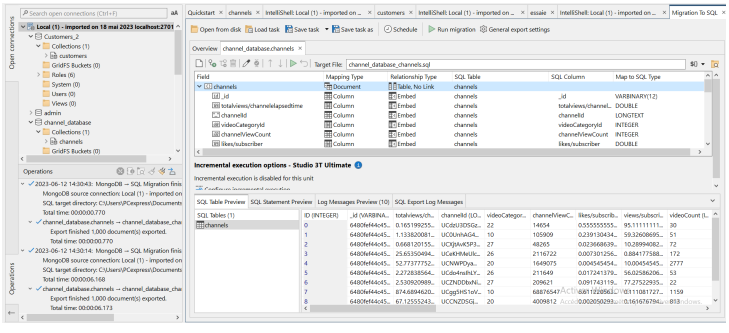


Figure 4.5: resultat de migration

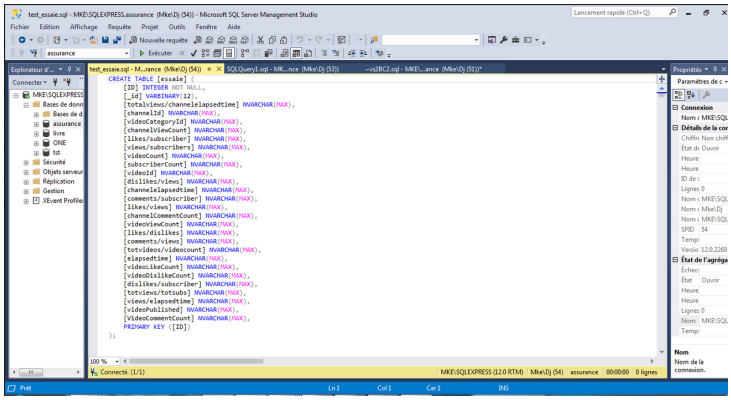


Figure 4.6: fichier SQL

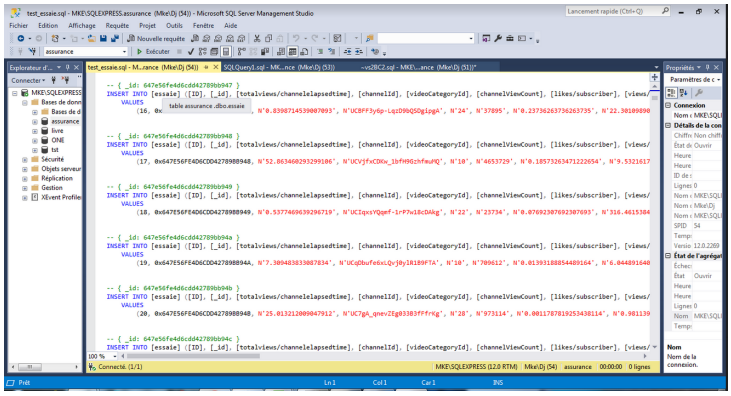


Figure 4.7: fichier SQL 2

4.3.2 méthode

Pour la troisième base de données (réelle) que nous avons utilisée, nous l'avons obtenue à partir de la CNAS. Elle est composée de deux tables CSV : "ayant" et "assuré". La taille totale des fichiers est de 11 Mo. Nous n'avons pas utilisé la migration SQL vers NoSQL ou NoSQL vers SQL. Au lieu de cela, nous avons importé les deux fichiers CSV dans MongoDB et SQL Server, puis nous avons effectué la jointure dans SQL Server et utiliser la fonction "lookup" dans MongoDB pour effectuer des opérations de jointure entre les tables.
la fonction "lookup": est une operation dans MongoDB effectuer une jointure entre deux collections.
La forme de la fonction lookup est:
db.collection.aggregate([lookup: from: "collection", localField: "champlocal", foreignField: "champetranger", as: "resultat"])

4.4 IMPLEMENTATION ET EXPERIMENTATION

L'expérience menée dans le cadre de cette étude comporte des scénarios en fonction de l'opération de requête exécutée, telle que l'insertion, la sélection, la mise à jour et la suppression. Chaque scénario a été réalisé en exécutant une requête avec différentes quantités de lignes (1, 10, 100, 1000 lignes). La requête a été exécutée 4 fois et la moyenne a été calculée pour obtenir le résultat final.
Pour chaque scénario, le temps de réponse de la requête a été représenté dans un graphique. Nous avons utilisé une formule pour calculer la valeur de y (temps de réponse de la requête (seconde) en fonction de la valeur de x (nombre de lignes ou nombre d'itérations).

4.5 Expérimentation avec la base N°1

Résultat de l'opération insert: La comparaison des résultats expérimentaux est visible dans le graphique qui se divise en deux catégories: le temps de réponse à une requête sur une table ou une collection avec relation (l'insertion sur une table qui contient une clé étrangère 'livre' ou une collection constituée de sous-documents intégrés pour MongoDB) et le temps de réponse à une requête sur une table ou une collection sans relation (l'insertion sur une table qui ne contient pas de clé étrangère 'auteur' ou une collection qui n'a pas de sous-document intégré pour MongoDB).

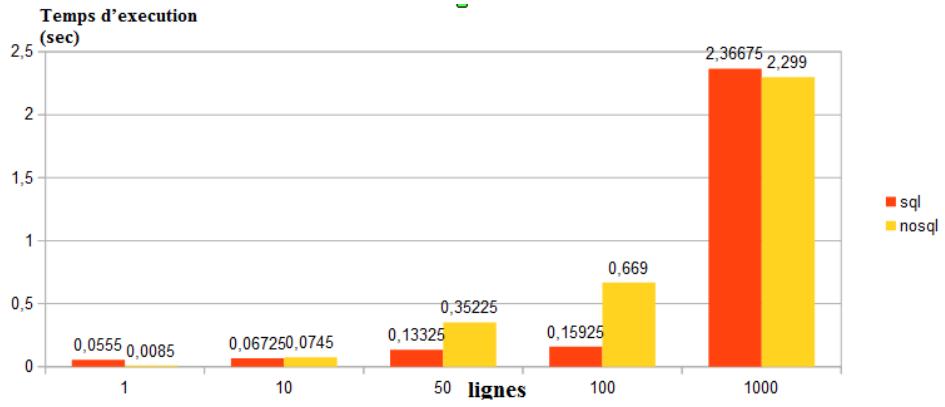


Figure 4.8: le temps d'exécution de l'inseretion sur un tableau ou un document sans relation

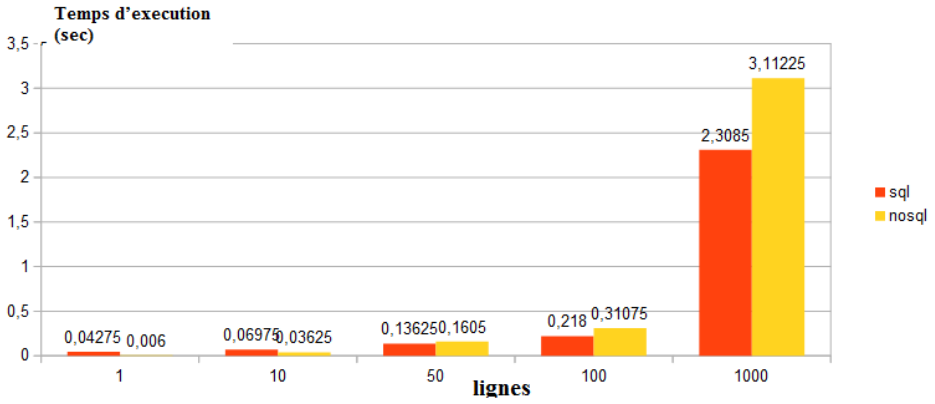


Figure 4.9: le temps d'exécution de l'Inseretion sur un tableau ou un document avec relation (base N°1)

les Figure n°4.8 4.9 montre la comparaison du temps d'insertion entre les deux systemes de base de données. les exemples de requetes exécutées dans le cadre de cette expérience sont les suivants:

- SQL SERVER :

```
insert into auteur values(@id,'jack' +convert(varchar(20), @id),
convert(varchar(20), @id))
```

- MONGODB :

```
db.memoi.insertOne(
 {
   "auteur": {
   "id_auteur":125 ,
   "nom": " sa",
   "prenom": "chch"
}  })
```

.

En cas d'insertion d'une seule ligne ou document, on peut observer que MongoDB peut insérer les données beaucoup plus rapidement que SQL Server. Il est important de confirmer et vérifier les contraintes d'intégrité, telles que le type de données valide (intégrité du domaine) avec les valeurs qui seront insérées, la clé primaire doit être unique et non nulle (intégrité des entités), et la clé étrangère doit correspondre à la clé primaire référencée dans une autre table dans le cas du SQL.

En ce qui concerne l'insertion de plusieurs lignes, MongoDB et SQL Server ont obtenu des résultats assez équivalents avec 10 et 50 lignes. Cependant, à mesure que le nombre d'éléments augmente, notamment avec 100 et 1000 lignes, la différence entre les deux systèmes a commencé à s'accentuer..

Résultat de l'opération Select :

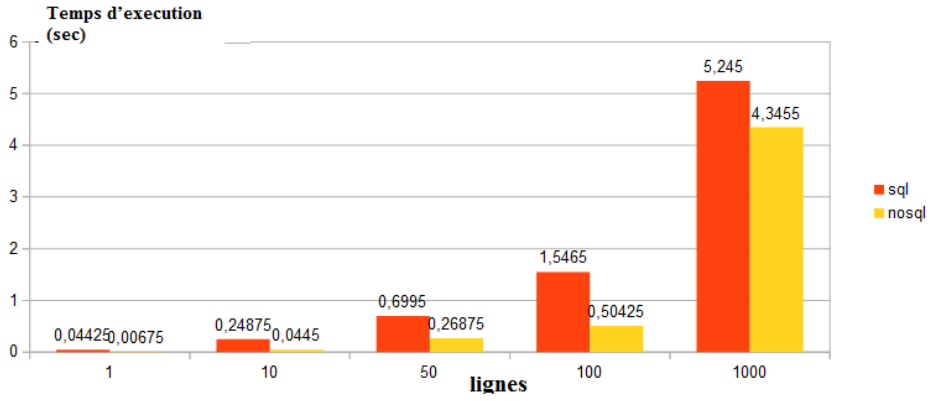


Figure 4.10: le temps d'exécution de select simple (base N°1)

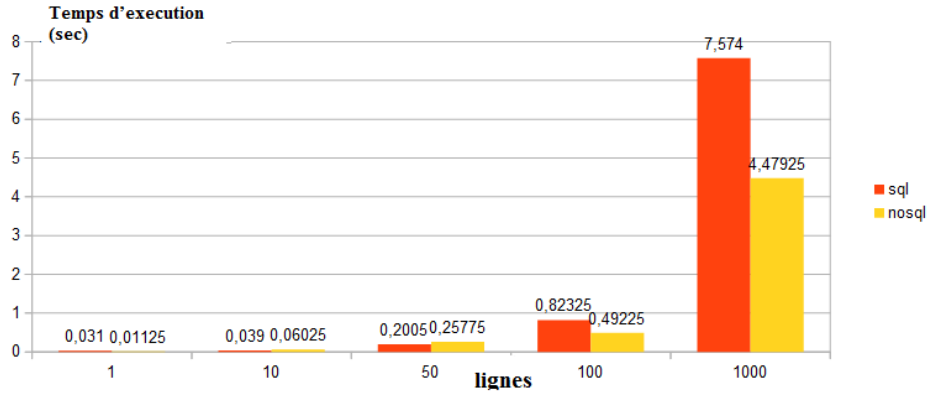


Figure 4.11: le temps d'exécution de select complex (base N°1)

Les figures 4.10 et 4.11 présentent le temps de sélection et de comparaison entre les deux systèmes de bases de données. La requête utilisée dans cette expérience comprend une requête simple et une autre complexe impliquant une opération de jointure (pour SQL) et certaines fonctions d'agrégation. Les résultats montrent que MongoDB est plus rapide que SQL.

L'opération de sélection dans MongoDB avec des documents liés à des conditions (document intégré) est beaucoup plus rapide, car MongoDB peut récupérer les données directement à partir des documents d'une collection sans avoir à prendre en compte les relations avec d'autres documents. Les données sont stockées dans un document qui contient un sous-document.

Il est important de noter que les performances peuvent varier en fonction de la taille des données, de la complexité des requêtes et de la configuration du système.

Exemple d'une requête exécutée pour select complexe:

- SQL SERVER :

```
SELECT
auteur.nom,auteur.prenom,livre.titre,SUM(vente.prix ) AS sumprix
FROM auteur
LEFT JOIN livre ON livre.idautr = auteur.id
LEFT JOIN vente ON vente.isbn10 =livre.isbn10
where   auteur.nom like '[AEIOU]%[AEIOU]' and
((select avg(prix) from vente)>3)
group by
auteur.nom,auteur.prenom
```

- MONGODB :

```
db.memoi.aggregate([{$group:{_id:"livre.auteur.nom",moyenne:
{$avg:"prix"}}},
{$match:{moyenne:{$gt:3}}},{$match:
{"livre.auteur.nom":/^[AEIOUY].*[AEIOUY]$/}},
{$project:
{"livre.auteur.nom":1,"livre.titre":1,"livre.auteur.prenom":1}}])
```

Résultat de l'opération MAJ :

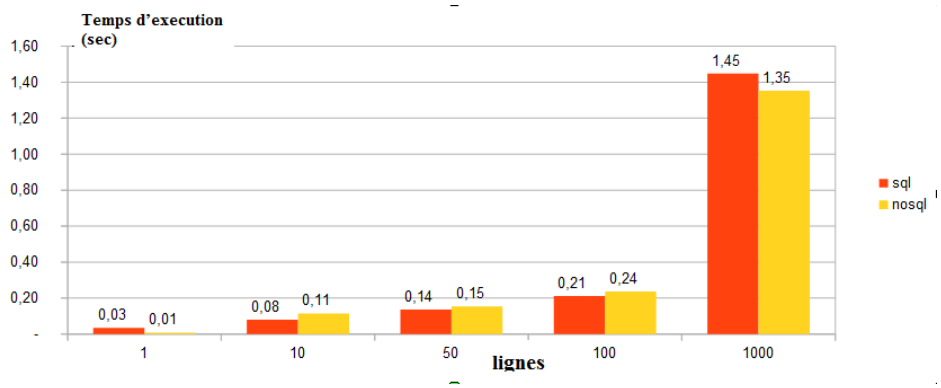


Figure 4.12: le temps d'exécution de mise à jour simple (base N°1)

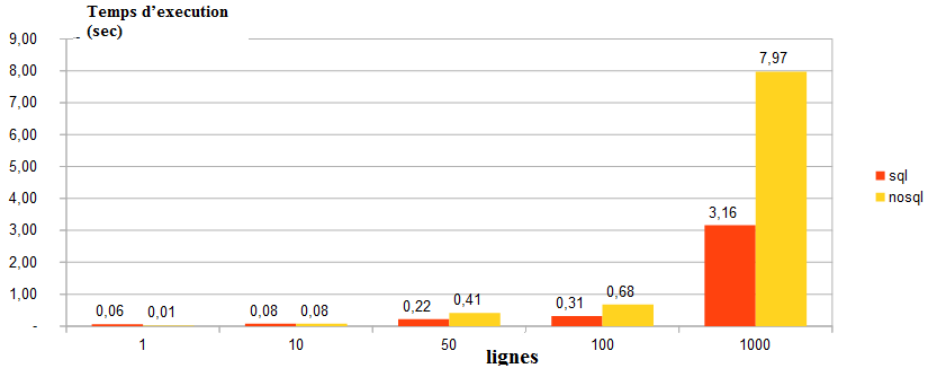


Figure 4.13: le temps d'exécution de mise à jour simple (base N°1)

Exemple d'une requête exécutée pour mise à jour:

- SQL SERVER:

```
declare @id int
select @id =1
while @id >=1 and @id <=1000
begin
update auteur set nom= 'john'+convert(varchar(20), @id)
where id=@id
 select @id = @id + 1
end.
```

- MONGODB :

```
db.memoi.updateOne({"livre.auteur.nom":"sa"},
                   {$set:{"livre.auteur.nom":"saa"}})
```

Les figures 4.12 et 4.13 mettent en évidence les performances de MongoDB par rapport à SQL Server lors de la mise à jour d'un seul enregistrement. Nous constatons que MongoDB est plus performant dans ce cas. Cependant, lorsque nous augmentons le nombre d'enregistrements à modifier (10, 50, 100, 1000 enregistrements), les résultats deviennent similaires ou SQL Server dépasse MongoDB.

Il est important de noter que ces résultats peuvent être influencés par la taille de la base de données et la complexité des opérations de mise à jour. Dans notre cas, avec une base de données ne dépassant pas 100 lignes, les différences de performances sont moins prononcées. Cependant, à mesure que la base de données et le nombre d'enregistrements augmentent, il est possible que les performances de SQL Server surpassent celles de MongoDB.

Résultat de l'opération Suppression:
Exemple d'une requête exécutée pour Suppression:

- SQL SERVER :

```
declare @id int
select @id =1
while @id >=1  and @id <= 1000
begin
DELETE FROM  vente
WHERE
    idvente = @id
end .
```

- MONGODB :

```
  {
db.memoi.deleteOne({"id_vente = @id
end .
":i})}
```

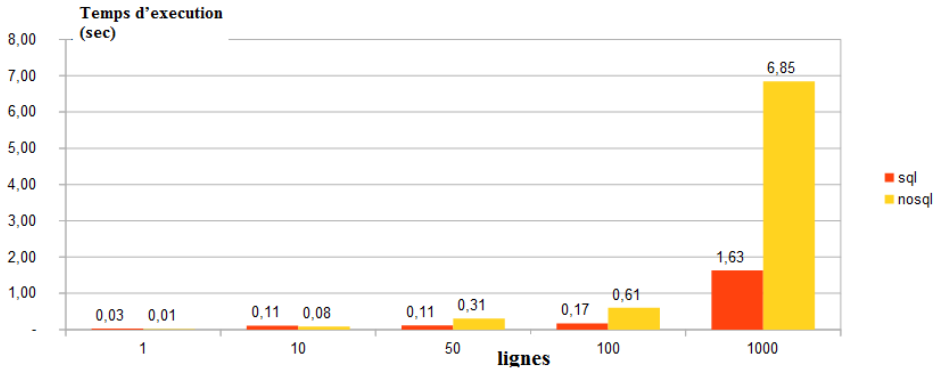


Figure 4.14: le temps d'exécution de la suppression simple (base N°1)

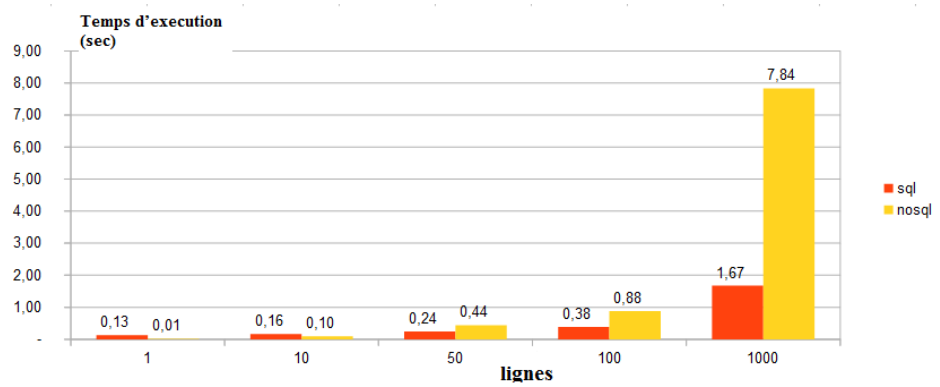


Figure 4.15: le temps d'exécution de la suppression complexe (base N°1)

.

MongoDB n'a rencontré aucun problème pour supprimer un document de la base de données et a maintenu un temps d'exécution stable jusqu'à 50 documents.
Cependant, avec 100 et 1000 enregistrements, MongoDB est plus lent que SQL Server.
La même remarque a été mentionnée lors de la mise à jour de bases de données de petite taille.

4.6 Expérimentation avec la base N°2

nous avons utilisé la même méthode et les mêmes opérations que pour la première base de données.
Résultat de l'opération insert:

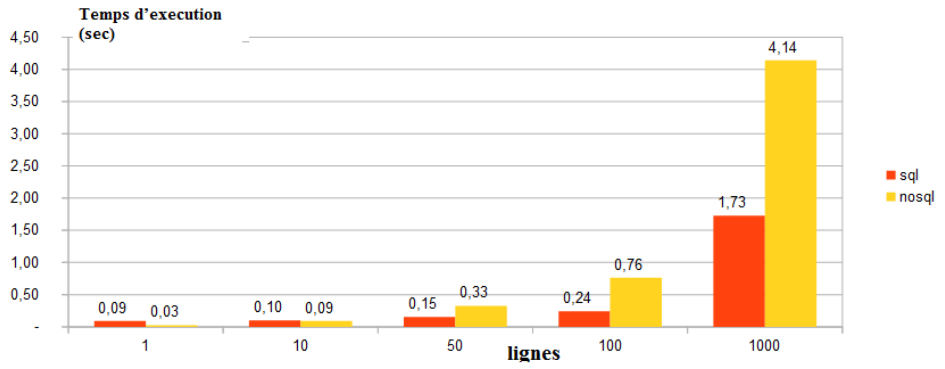


Figure 4.16: le temps d'exécution de l'inseretion sur un tableau ou un document sans relation

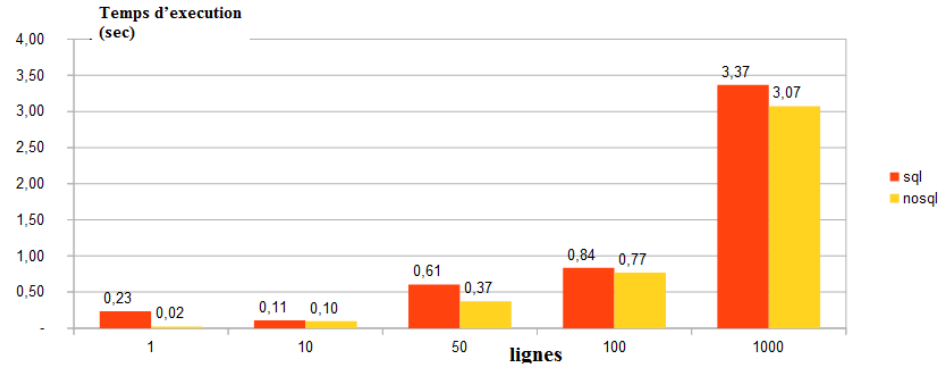


Figure 4.17: le temps d'exécution de l'Inseretion sur un tableau ou un document avec relation

les exemples de requetes exécutées dans le cadre de cette expérience sont les suivants:

- SQL SERVER :

```
declare @id float
select @id = 7424
declare @int float
select @int =991890003831
while @id >=7424 and @id <=7424
begin
insert into assurer values(@int,'jack'+convert(varchar(20),@int),
'jack'+convert(varchar(20),@int),
'19882307','F','c','BRIDA','3000',1,'9','9','3','3','331741278','10309'
,500,'1','2','2012-02-21 ','9','N','4',311,'2002-06-22','1998-07-
29',@id)
insert into ayantdroit values(@id,@int,'80','jack'+convert(varchar(20)
, @int),'jack'+convert(varchar(20), @int),'F','M','2012-02-21','1','1'
,'20020622','9','0','19980729','null')
select @id = @id + 1
select @int = @int + 1
end
```

- MONGODB :

```
db.combinedCollection.insertOne({
  "NO_ASSURE": 82017853937,
  "NOM": "Jacques",
  "PRENOM": "Jacque",
  "D_NAISS": "07012000",
  "SEXE": "M",
  "SIT_FAMILLE": "<value>",
  "ADRESSE": "BRIDA",
  "TAUX": 1,
  "OFFICINES_PERMISES": 9,
  "STRUCTURES_PERMISES": 9,
  "STATUT": 3,
  "POSIT": 3,
  "EMPLOYEUR": 317241278,
  "CENTRE": 10309,
  "SALAIRE_REF": "5,00",
  "ORGANISME_AFFIL": 1,
  "DROITS": 2,
  "MEDECINS_PERMIS": 9,
  "CONVOQUE": "C",
  "N_NATIONAL": "N",
  "NATIONALITE": 4,
  "DATE_NAISSANCE": "11/9/2000",
  "DATE_PRESUME": 0,
  "ID": 46541958,
```

```
  "matchedData": [
    {
      "NO_ASSURE": "8117853937",
      "RANG_AD": 1,
      "NOM": "KHELIFI",
      "PRENOM": "MOHAMED",
      "SEXE": "M",
      "SITUATION": "N",
      "TAUX": 1,
      "POSIT": 1,
      "OFFICINES_PERMISES": 9,
      "BLOCAGE": 0,
      "DATE_FIN_DROIT": "1/10/2006"
    }
  ]
})
```

.

Résultat de l'opération Select :

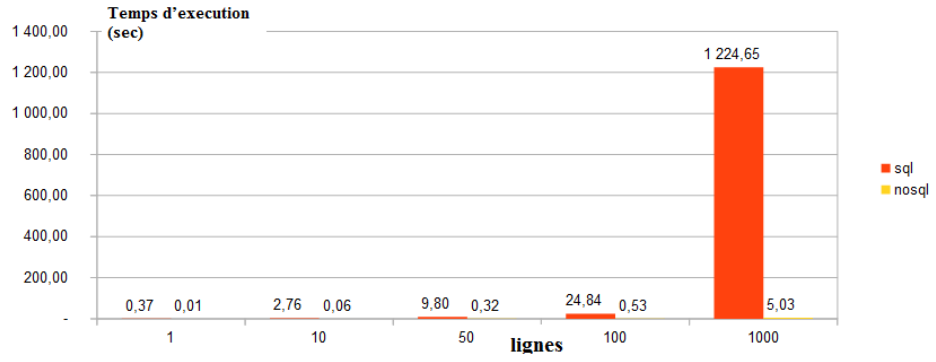


Figure 4.18: le temps d'exécution de select simple (base N°1)

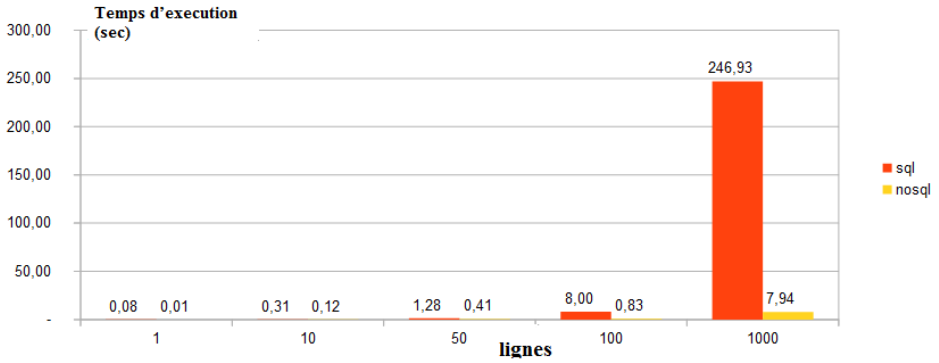


Figure 4.19: le temps d'exécution de select complex (base N°1)

les exemples de requetes exécutées sont les suivants :

- SQL SERVER :

```
SELECT
 a.NOM ,a.PRENOM,a.NO_ASSURE,a.DATE_NAISSANCE,count(ay.NO_ASSURE)
  as 'nombre ayant droit'
FROM assurer a ,  ayantdroit ay
where ay.NO_ASSURE=a.NO_ASSURE and ay.RANG_AD<80 and a.DATE_NAISSANCE
BETWEEN '1952-01-01'  AND  '1998-02-01 '
group by
 a.NOM ,a.PRENOM,a.NO_ASSUR
```

- MONGODB :

```
db.getCollection("assure").aggregate([
  {
    $lookup: {
      from: "ayant",
      localField: "NO_ASSURE",
      foreignField: "NO_ASSURE",
      as: "ayant"
    }
  },
  {
    $project: {
      NO_ASSURE: 1,
      num_ayant: { $size: "$ayant" }
    }
  },
  {
    $group: {
      _id: "$NO_ASSURE",
      total: { $sum: "$num_ayant" }
    }
  },
{
    $match: {
      "result.RANG_AD": 80,
"result.DATE_NAISSANCE":{"$gte":"1952-01-01","$lte":"1998-02-01"}
}},
{$project:{"result.NOM":1,"result.PRENOM":1,"result.DATE_NAISSANCE":1}}])
```

.

Résultat de l'opération mise à jour :

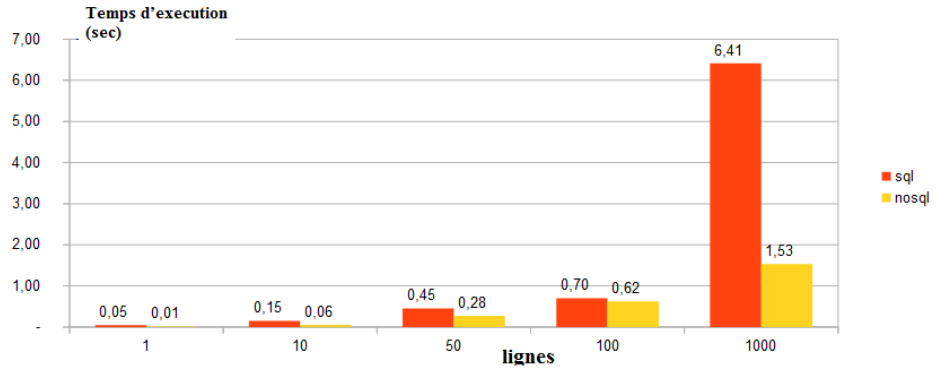


Figure 4.20: le temps d'exécution de mise à jour simple

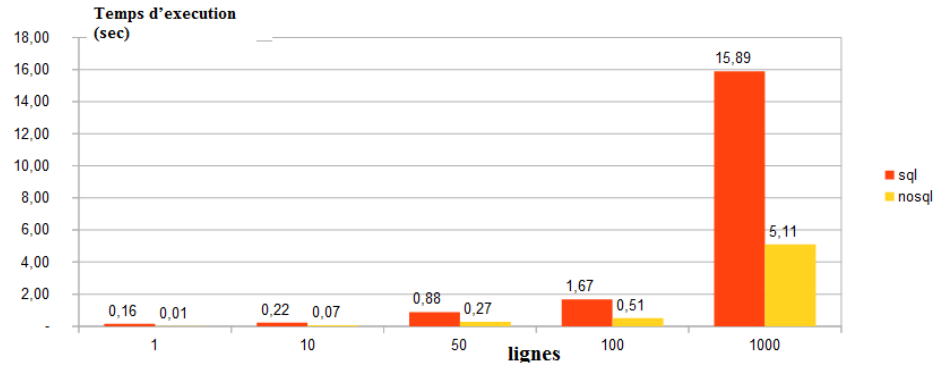


Figure 4.21: le temps d'exécution de mise à jour simple

– SQL SERVER :

```
declare @id INT
select @id = 1

while @id >=1  and @id <= 1000
begin
update   assurer   SET TAUX = 5000
   where SEXE='F'   and DATE_NAISSANCE<='1989-01-01'
  select @id = @id + 1
end
```

– MONGODB :

```
update simple:
db.assure.aggregate([
  {
$addFields: {
SALAIRE_REF: {
$toInt: {
$replaceOne: {
```

```
input: { $replaceOne: { input: "$SALAIRE_REF", find: ",",
replacement: "." } },
find: ".",
replacement: ""
          }
        }
      }
    }
  },
  {
    $match: {
      SALAIRE_REF: { $exists: true }
    }
  },
  {
    $addFields: {
      SALAIRE_REF: { $add: ["$SALAIRE_REF", 1000] }
    }
  },
  {
    $set: {
      SALAIRE_REF: "$SALAIRE_REF"
    }
  }
])
```

Résultat de l'opération Suppression:
Exemple d'une requête exécutée pour la Suppression:

- SQL SERVER :

```
declare @id float
select @id = 991890003831
while @id >=991890003831  and @id <=  991890003831
begin
delete assurer from assurer a inner join
ayantdroit ay on ay.NO_ASSURE=a.NO_ASSURE

where a.NO_ASSURE=@id
 select @id = @id + 1

end
```

- MONGODB :

```
db.combinedCollection.deleteOne({
  "NO_ASSURE": 82017853937,
  "NOM": "Jacques",
  "PRENOM": "Jacque",
  "D_NAISS": "07012000",
  "SEXE": "M",
  "SIT_FAMILLE": "<value>",
  "ADRESSE": "BRIDA",
  "TAUX": 1,
  "OFFICINES_PERMISES": 9,
  "STRUCTURES_PERMISES": 9,
  "STATUT": 3,
  "POSIT": 3,
  "EMPLOYEUR": 317241278,
  "CENTRE": 10309,
  "SALAIRE_REF": "5,00",
  "ORGANISME_AFFIL": 1,
  "DROITS": 2,
  "MEDECINS_PERMIS": 9,
  "CONVOQUE": "C",
  "N_NATIONAL": "N",
  "NATIONALITE": 4,
  "DATE_NAISSANCE": "11/9/2000",
  "DATE_PRESUME": 0,
  "ID": 46541958,
  "matchedData": [
    {
      "NO_ASSURE": "8117853937",
      "RANG_AD": 1,
      "NOM": "KHELIFI",
      "PRENOM": "MOHAMED",
```

```
        "SEXE": "M",
        "SITUATION": "N",
        "TAUX": 1,
        "POSIT": 1,
        "OFFICINES_PERMISES": 9,
        "BLOCAGE": 0,
        "DATE_FIN_DROIT": "1/10/2006"
    }
  ]
})
```

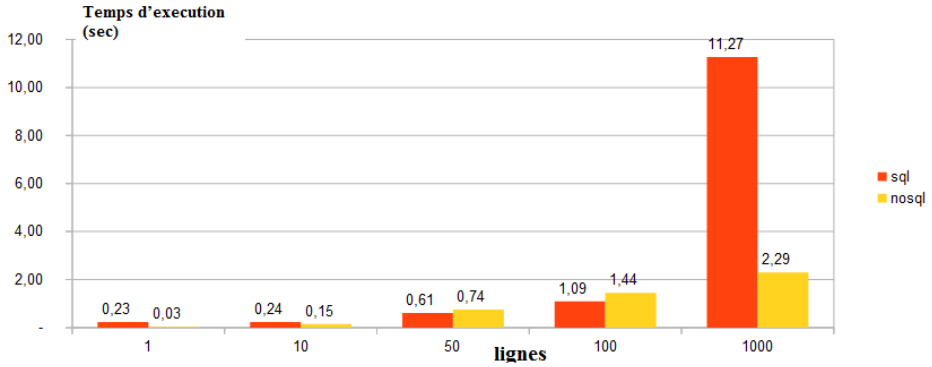


Figure 4.22: le temps d'exécution de la suppression simple

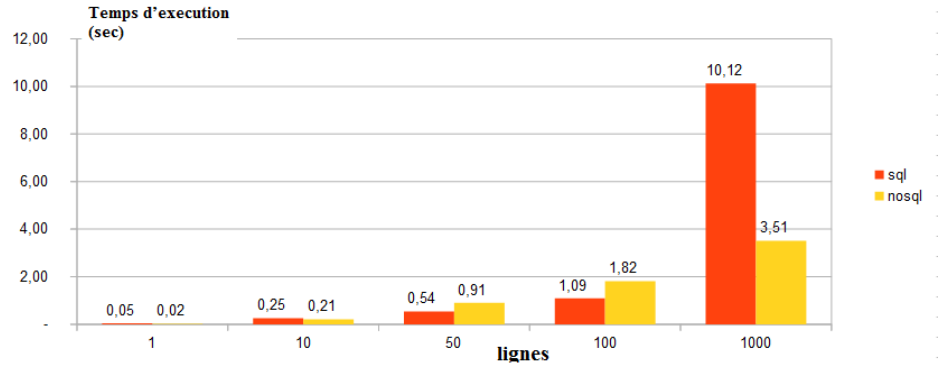


Figure 4.23: le temps d'exécution de la suppression complexe

. Nous remarquons que dans tous les histogrammes de la deuxième base de données, MongoDB est plus rapide que SQL pour toutes les opérations, sauf la suppression. On note que SQL est plus performant dans certains cas (50 et 100 lignes), mais plus lent dans le cas de 1000 lignes.
En comparant avec les résultats de la base précédente, nous constatons que la taille de la base augmente. Donc, MongoDB est le meilleur choix pour le big data.

4.7 Expérimentation avec la base N°3

Dans la base N°3, nous utilisons la même méthode, mais avec des nombres différents du lignes (1, 100, 1000, 10000,1000000 lignes).

Résultat de l'opération insert:

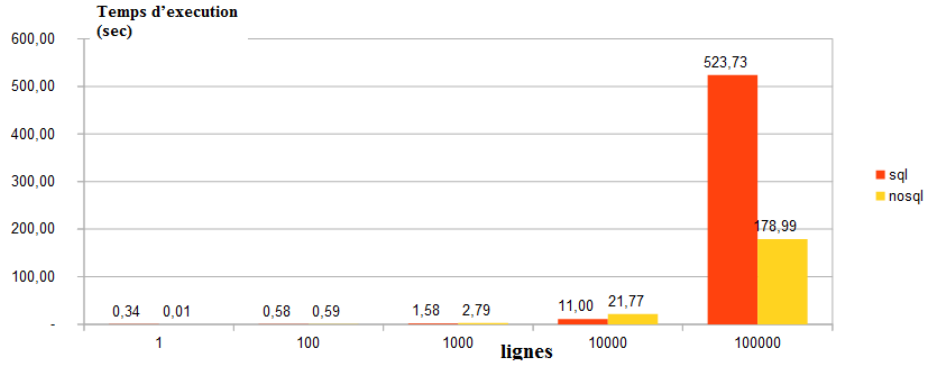


Figure 4.24: le temps d'exécution de l'insertion

Résultat de l'opération Suppression:

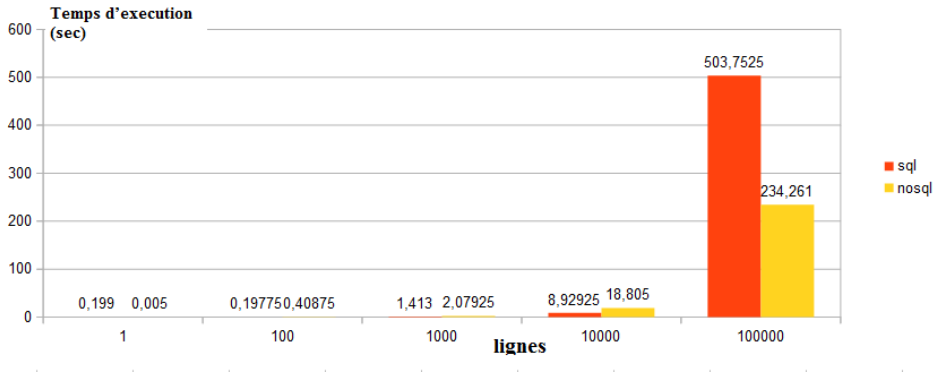


Figure 4.25: le temps d'exécution de la suppression

Résultat de l'opération Select:

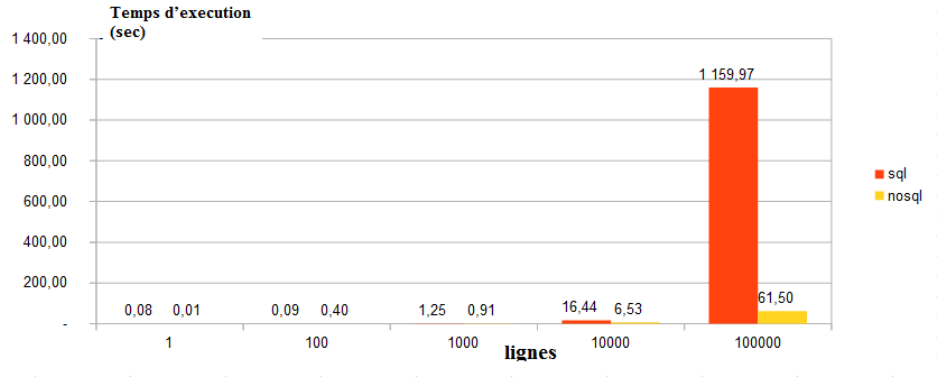


Figure 4.26: le temps d'exécution de select simple

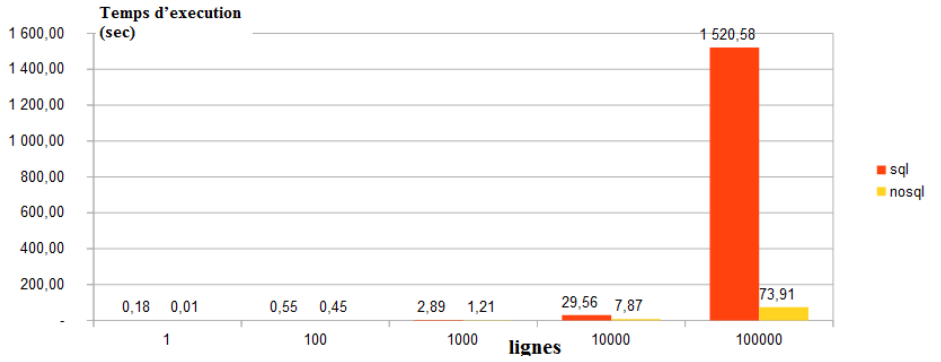


Figure 4.27: le temps d'exécution de select complexe

Résultat de l'opération mise à jour:

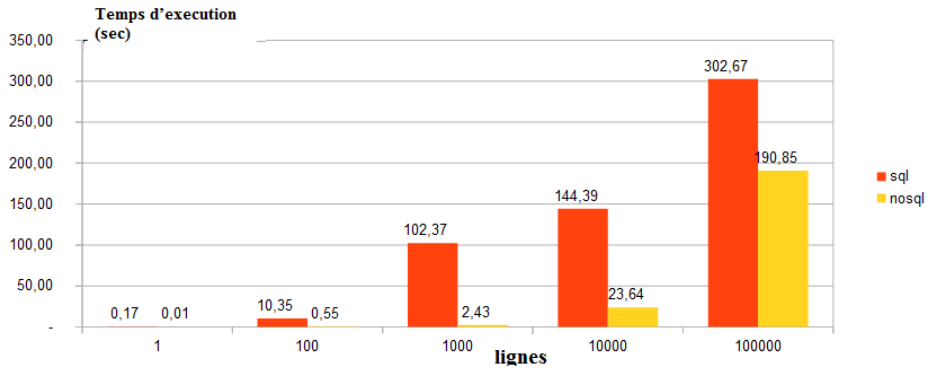


Figure 4.28: le temps d'exécution de mise à jour simple

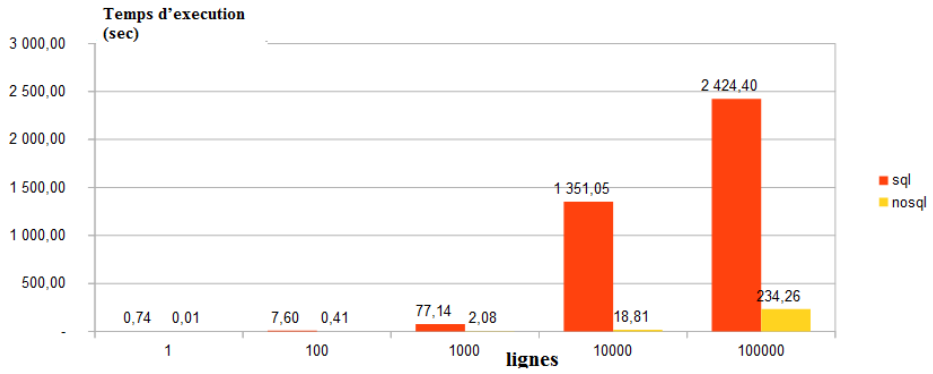


Figure 4.29: le temps d'exécution de select mise à jour complexe

Les résultats de la troisième base de données confirment les analyses et les conclusions que nous avons précédemment tirées des bases de données N°1 et N°2. Nous avons observé que MongoDB est généralement plus rapide que SQL Server. Cette différence de performance s'explique par le fait que MongoDB excelle particulièrement lorsque la taille des données augmente. Ainsi, plus la base de données est volumineuse, meilleures sont les performances de MongoDB.

Cette chapitre nous a conduit à la conclusion que MongoDB présente de meilleures performances en termes d'insertion, de sélection et de mise à jour des données, notamment pour les bases de données de grande taille. En revanche, SQL Server se montre plus rapide que MongoDB pour les requêtes de suppression, ainsi que pour les petites bases de données.

Chapter 5

Conclusion

Les bases de données sont utilisées pour stocker, récupérer et gérer des informations. Pour longtemps, le modèle relationnel était utilisé pour la conception et le stockage des bases de données.
Le travail réalisé dans le cadre de ce mémoire a été motivé par l'émergence des systèmes NoSQL comme alternative aux systèmes relationnels.
Notre objectif principal est de réaliser une étude comparative entre les deux systèmes afin d'éclairer les choix des utilisateurs des bases de données.
Nous avons présenté dans un premier temps les avantages et les inconvénients des deux systèmes.
Nous nous sommes focalisés par la suite sur les bases orientées documents et plus particulièrement la base MongoDB qui est très largement utilisée.
Nous avons utilisé des outils pour réaliser la conversion d'une base relationnelle en base orientée document et vise versa. L'utilisation de ces outils, bien qu'un peu complexe à prendre en main, s'est avéré très enrichissante.
Nous avons exécuté plusieurs requêtes sur des bases de tailles différentes pour chacun des deux systèmes de stockage des données.
Nos premières expériences confirment un avantage de la modélisation NoSql et laissent penser qu'elles sont plus adaptées pour les grands volumes de données. Ils nous semble également important de souligner que le modèle relationnel gardent ses avantages pour les bases de tailles modérées. Ceci dit, une utilisation hybride sera plus avantageuse.
Une perspective intéressante de notre travail serai, d'une part, de réaliser des expérimentations avec des jeux de données plus importants. D'autre part, utiliser d'autre type de base NoSql telles que les bases orientées graphe.

Bibliography

[1] les avantages des bases de données relationnelles. https://www.codyx.org/1158/les-avantages-des-bases-de-donnees-relationnelles/s-bases-de-donnees-relationnelles/ . Consulté le: 2023-04-25.

[2] Qu'est-ce qu'une base de données relationnelle ? https://www.oracle.com/fr/database/what-is-a-relational-database/. Consulté le:2023-04-27.

[3] Youtube's channels dataset — kaggle. https://www.kaggle.com/datasets/harshithgupta/y channels-dataset. Consulté le:2023-06-29.

[4] Nosql at e. renaux. https://manurenaux.wp.imt.fr/2012/04/18/nosql/, 2012.

[5] Bases de données : Big data et nosql. https://administration-systeme.blogspot.com/2013/10/, 2013.

[6] Cap theorem and distributed database management systems. https://towardsdatascience. com/cap-theorem-and-distributed-databasemanagement-systems-5c2be977950e, 2018.

[7] Mongodb to sql migration. https://studio3t.com/knowledge-base/articles/mongodb-to-sql-migration/, 2019.

[8] Matteo Di Maglie. *Adoption d'une solution NoSQL dans l'entreprise*. PhD thesis, Haute école de gestion de Genève, 2012.

[9] Kouedi Emmanuel. Approche de migration d'une base de données relationnelle vers une base de données nosql orientée colonne. *Mémoire de Master, Université Yaoundé, Cameroun*, 2012.

[10] Christoforos Hadjigeorgiou. Rdbms vs nosql: Performance and scaling comparison. *MSc in High Performance Computing The University of Edinburgh*, August 23, 2013.

[11] BENALLAL Zeyneb TAHRAOUI Hayet. Etude comparative des bases de données nosql : Mongodb, couchbase, cassandra, hbase, redis, orientdb. *En vue de l'obtention du diplome de Master Academique, universitè é Abou Bakr Belkaid– Tlemcen*, 2015-2016.

[12] Bennabi Narimane and Ait Ibrahim Nassima. Conception et réalisation d'une base de données nosql sous hbase et firebase cas d'un tremblement de terre. 2019.

[13] ShankarNayak Bhukya Narsimha Banothu and K Venkatesh Sharma. "big-data: Acid versus base for database transactions. *In: International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT).*, page 3704–3709, 2016.

[14] Mebark OMARI, Mohammed BEN OMAR, El Mamoun Mamouni, et al. *L'utilisation d'une base NoSQL (HBASE) dans un milieu distribué (Hadoop)*. PhD thesis, Université Ahmed Draïa-Adrar, 2017.

[15] BOUDJEMAA SAMIRA. Etude comparative entre des systèmes nosql. *Mémoire de Master, Université Mohamed Khider Biskra*, 15/09/2020.