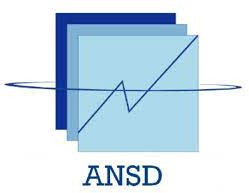
Ministère de l’Economie, du Plan et de la Coopération

République du Sénégal



Agence Nationale de la Statistique et de la Démographie (ANSD)



Un peuple - Un but - Une foi

Ecole Nationale de la Statistique et de l’Analyse Economique Pierre NDIAYE (ENSAE-Pierre NDIAYE)



**Projet de BIG DATA & CLOUD COMPUTING**

**LES BASES DE DONNEES NoSQL**

*Senior Machine Learning Engineer Entrepreneure @Sen-IA*

**OCTOBRE 2024**

*Elèves ingénieurs statisticiens économistes*

KONE CHAKA

MANDO Wendvi Abija Déborah

TRAORE Sié Rachid

TRAORE Mohamed

YOUM Amadou

Mme. DIAW Mously

**Sous l’encadrement de :**

**Rédigé par :**

Sommaire

[Sommaire 2](#_Toc181132177)

[RESUME 4](#_Toc181132178)

[Introduction 5](#_Toc181132179)

[1 Histoire du NoSQL 6](#_Toc181132180)

[2 Les caractéristiques des bases de données NoSQL 7](#_Toc181132181)

[2.1 Modèles de données flexibles 7](#_Toc181132182)

[2.1.1 Clé-Valeur 7](#_Toc181132183)

[2.1.2 Documents 8](#_Toc181132184)

[2.1.3 Colonnes larges 8](#_Toc181132185)

[2.1.4 Graphes 8](#_Toc181132186)

[2.2 Scalabilité horizontale 8](#_Toc181132187)

[2.3 Haute performance 8](#_Toc181132188)

[2.4 Absence de schéma rigide 8](#_Toc181132189)

[2.5 Tolérance aux pannes et haute disponibilité 8](#_Toc181132190)

[2.6 Cohérence éventuelle 9](#_Toc181132191)

[2.7 Absence de transactions ACID strictes 9](#_Toc181132192)

[3 Types de bases de données NoSQL et leurs champs d’application 9](#_Toc181132193)

[3.1 Modèle Clé-Valeur 9](#_Toc181132194)

[3.1.1 Caractéristiques 10](#_Toc181132195)

[3.1.2 Utilisations 10](#_Toc181132196)

[3.1.3 Exemple visuel 10](#_Toc181132197)

[3.2 Modèle Documentaire 10](#_Toc181132198)

[3.2.1 Caractéristiques 11](#_Toc181132199)

[3.2.2 Utilisations 11](#_Toc181132200)

[3.2.3 Exemple visuel 11](#_Toc181132201)

[3.3 Modèle en Colonnes 11](#_Toc181132202)

[3.3.1 Caractéristique 12](#_Toc181132203)

[3.3.2 Utilisation 12](#_Toc181132204)

[3.3.3 Exemple visuel 12](#_Toc181132205)

[3.4 Modèle Graphe 12](#_Toc181132206)

[3.4.1 Caractéristiques 13](#_Toc181132207)

[3.4.2 Utilisations 13](#_Toc181132208)

[3.4.3 Exemple visuel 13](#_Toc181132209)

[4 Présentation de quelques bases de données NoSQL 13](#_Toc181132210)

[4.1 MongoDB 13](#_Toc181132211)

[Caractéristiques et utilisation 13](#_Toc181132212)

[4.2 Redis 14](#_Toc181132213)

[Caractéristiques et utilisation 14](#_Toc181132214)

[4.3 Cassandra 14](#_Toc181132215)

[Caractéristiques et utilisations 15](#_Toc181132216)

[4.4 Neo4j 15](#_Toc181132217)

[Caractéristiques et utilisation 15](#_Toc181132218)

[5 Avantages et inconvénients des bases de données NOSQL 15](#_Toc181132219)

[5.1 Avantages 15](#_Toc181132220)

[5.2 Inconvénients 16](#_Toc181132221)

[6 Bonnes méthodes pour optimiser les bases de données NoSQL 16](#_Toc181132222)

[6.1 Choisir un bon modèle NoSQL 16](#_Toc181132223)

[6.2 Concevoir le schéma avec soin 16](#_Toc181132224)

[6.3 Utiliser des index appropriés 17](#_Toc181132225)

[6.4 Optimiser les requêtes 17](#_Toc181132226)

[6.5 Mise à l’échelle horizontale de la base de données 17](#_Toc181132227)

[6.6 Surveiller et ajuster les performances de votre base de données 17](#_Toc181132228)

[Conclusion 18](#_Toc181132229)

[Table des figures 19](#_Toc181132230)

[Bibliographie 20](#_Toc181132231)

RESUME

Dans ce travail, nous explorons les bases de données NoSQL comme une solution pour gérer les grandes masses de données non structurées, à la suite des limites des bases SQL face aux exigences du big data. Nous commençons par un historique du NoSQL, puis nous détaillons ses principales caractéristiques (modèles flexibles, scalabilité horizontale, haute performance, absence de schéma rigide, tolérance aux pannes, etc.). De plus, différents types de bases NoSQL sont examinés (Clé-Valeur, Document, Colonnes, Graphe), avec leurs champs d’application respectifs.

Nous poursuivons en décrivant également plusieurs systèmes NoSQL, tels que mongoDB, redis, cassandra et Neo4j, en abordant leurs caractéristiques et cas d’usage. La flexibilité, l’adaptabilité à travers les types de données et la performance élevée sont des avantages que nous expliquerons ainsi que les inconvénients tels que le manque de standardisation, difficulté pour des requêtes complexes, etc.

Pour optimiser l’utilisation des bases de données NoSQL, nous recommandons des stratégies comme le choix du modèle adéquat, la création d’index, l’optimisation des requêtes, et la surveillance des performances.

La conclusion souligne que le NoSQL est un complément utile aux bases de données SQL, adapté aux besoins modernes en matière de gestion de données à grande échelle.

Introduction

Avec l’explosion quantitative des données numériques, la variété des données collectées, l’augmentation de la puissance des machines, des capacités de stockages ainsi que le développement des outils informatiques, les bases de données SQL se sont montrées inappropriées au stockage, au traitement et à l’analyse des données volumineuses et non structurées. Les bases de données NoSQL constituent de ce fait une alternative très puissante pour relever le défi de la gestion des données de grande masse non structurées. De nos jours, les entreprises, les organisations et les chercheurs ont de plus en plus, recours aux bases de données NoSQL pour éclairer leur prise de décision. Pour une meilleure appropriation des bases de données NoSQL, une bonne connaissance des outils, du fonctionnement et des caractéristiques semblent plus que nécessaire. Ce travail vise alors à faire une présentation générale des bases de données NoSQL.

Pour une meilleure appréhension du sujet, nous allons passer en revue l’historique des bases de données NoSQL. Ensuite, nous présenterons les caractéristiques, les différents types de bases de données NoSQL, leurs avantages et inconvénients. Enfin, nous proposerons quelques recommandations pour une optimisation des bases de données.

1. Histoire du NoSQL

Historiquement, l’origine des bases de données remonte aux « années 50 » [[1]](#footnote-1) avec la création des premiers disques dures. Le besoin d'organiser les données, parfois en très grande quantité, afin d’en optimiser la gestion et la restitution, a toujours été central en informatique. Pendant longtemps, les systèmes d’information des entreprises et des organisations structuraient leurs données sous forme de fichiers, chaque application disposant alors de son propre fichier dédié.

Au début des « années 70 »[[2]](#footnote-2), les premiers systèmes de gestion de bases de données SGBD (logiciel de haut niveau permettant de manipuler les informations contenues dans une base de données) ont vu le jour. Le principal défi des bases de données réside dans le stockage et la récupération des données. Toutefois, ces fonctions incluent également des aspects cruciaux tels que la sécurité et la prévention des incohérences. Un système de gestion de données doit ainsi être capable de les conserver sur un support de mémoire de masse et fournir des outils permettant de les retrouver, souvent via un langage dédié.

Il existe plusieurs modèles de SGBD tels que lemodèle hiérarchique, premier modèle de gestion de données sur ordinateur développé dans les « années 50 », le modèle réseau « année 70 » qui permet la création de liaisons de type n-n, sans restriction sur les relations entre objets.

Dans les « années 70 », le modèle relationnel a été développé par **Edgar F. Codd**[[3]](#footnote-3), et il est devenu la norme pour la gestion des bases de données. Les systèmes de gestion de bases de données relationnelles (SGBDR) comme Oracle, MySQL, SQL Server et PostgreSQL ont dominé pendant plusieurs décennies. Ces systèmes utilisent le langage SQL (Structured Query Language) pour manipuler les données, qui sont stockées sous forme de tables (relations). Le modèle relationnel est très structuré, avec des schémas rigides et des relations entre les tables bien définies.

Les évolutions logicielles ont suivi, de manière naturelle, les avancées matérielles. Les premiers (SGBD) étaient conçus pour fonctionner sur des mainframes et dépendaient des capacités de stockage de l’époque. Avec l’interconnexion croissante des réseaux, de nouvelles opportunités ont émergé. Au début du 21e siècle, les volumes de données manipulés par certaines entreprises ou organisations, notamment à travers internet, ont augmenté de façon exponentielle. Que ce soit des données scientifiques, des informations provenant de réseaux sociaux « les insight », des bases de données médicales, ou encore des agences de défense nationale, la quantité de données très souvent hétérogène a explosé. Ce phénomène a posé de nouveaux défis en matière de gestion et de traitement, et les bases de données relationnelles traditionnelles ont montré leurs limites.

C’est ainsi que le NoSQL est apparu comme une réponse à ces besoins émergents. Le terme « NoSQL » a été utilisé pour la première fois en 1998 par **Carlo Strozzi** pour désigner les systèmes de gestion de bases de données non relationnelles (SGBDNR). NoSQL signifie « Not Only SQL » (pas seulement SQL) plutôt qu'une opposition totale à SQL. Il regroupe un ensemble de systèmes de gestion de bases de données qui ne sont pas basés sur le modèle relationnel traditionnel, et qui permettent de stocker et de traiter de grandes quantités de données non structurées ou semi-structurées. Ces bases sont particulièrement adaptées à la gestion de gros volumes de données distribuées. L’un des attraits majeurs des bases de données NoSQL réside dans leur approche pragmatique. En effet, de nombreux moteurs NoSQL ont été développés en réponse aux besoins spécifiques des entreprises qui les ont créés.

À partir de l’année 2010, de grandes entreprises ont commencé à adopter des bases NoSQL pour faire face à leurs besoins croissants en matière de traitement de données à grande échelle. Des géants comme Google, Amazon, Facebook, LinkedIn et Twitter ont développé et adopté des bases NoSQL spécifiques à leurs besoins (par exemple, Google a créé Bigtable et Amazon DynamoDB).

Le NoSQL est née de la nécessité de gérer des volumes de données massifs, non structurés et en constante évolution, et d'offrir une scalabilité et des performances supérieures aux systèmes relationnels traditionnels. Cependant, le NoSQL n’a pas remplacé les bases de données relationnelles, mais s'est imposé comme une solution complémentaire pour répondre à des besoins spécifiques dans des environnements de données massifs, distribués et flexibles.

1. Les caractéristiques des bases de données NoSQL

Les bases de données NoSQL se distinguent principalement par leur rupture avec le modèle relationnel traditionnel, évitant l’utilisation de tableaux avec des colonnes fixes. Elles n’exigent ni normalisation des données ni mappage relationnel. Leur interaction est simplifiée, sans recours à des langages de requête complexes. Une autre spécificité est la flexibilité ou l’absence de schéma prédéfini. Il n’est pas nécessaire de définir un schéma préalable, ce qui permet de stocker des données aux structures variées dans un même système.

Ces bases de données se caractérisent aussi par une interface simple pour la gestion du stockage et des requêtes. Elles s’appuient sur des APIs permettant d’effectuer diverses opérations de manipulation des données. Le protocole de communication est souvent basé sur HTTP REST[[4]](#footnote-4), avec un format d’échange en JSON[[5]](#footnote-5), et les requêtes sont généralement formulées en utilisant un langage NoSQL.

De plus, l'une des caractéristiques majeures des bases NoSQL est leur nature distribuée. Elles sont conçues pour fonctionner de manière distribuée, avec des fonctionnalités d’auto-scalabilité et de tolérance aux pannes. Dans ce cadre, les propriétés ACID[[6]](#footnote-6) sont parfois sacrifiées pour privilégier l’élasticité et les performances. Nous pouvons résumer ces principales caractéristiques comme suit :

* 1. Modèles de données flexibles

Les bases NoSQL permettent de stocker des données sous différentes formes, ce qui est une différence majeure par rapport aux bases de données relationnelles, qui imposent des schémas rigides avec des tables et des relations bien définies.

* + 1. Clé-Valeur

Chaque donnée est stockée sous forme de pair clé-valeur, similaire à un dictionnaire. Ce modèle est simple et extrêmement rapide pour les recherches par clé.

* + 1. Documents

Les bases de données documentaires (comme MongoDB) permettent de stocker des documents structurés (JSON, XML). Chaque document peut avoir une structure différente, ce qui est utile pour des données variées ou complexes.

* + 1. Colonnes larges

Ce modèle (comme Cassandra) est conçu pour gérer d'énormes ensembles de données structurées en colonnes, avec la possibilité d’ajouter dynamiquement des colonnes. Il est particulièrement adapté aux grandes bases de données analytiques.

* + 1. Graphes

Les bases NoSQL de graphes (comme Neo4j) sont conçues pour gérer des relations complexes entre des entités, en particulier dans des scénarios comme les réseaux sociaux ou les systèmes de recommandation.

* 1. Scalabilité horizontale

La scalabilité horizontale, également appelée « sharding**»**, permet de répartir les données sur plusieurs serveurs ou nœuds. Cela est particulièrement important lorsque le volume de données ou le nombre d'utilisateurs augmente considérablement. Avec NoSQL, on peut ajouter plus de serveurs pour traiter les charges supplémentaires sans perturber le fonctionnement global et cela améliore les performances et la capacité de stockage.

* 1. Haute performance

Les bases de données NoSQL sont conçues pour offrir des performances élevées, notamment pour les opérations en lecture et en écriture. Elles n'ont pas à exécuter des requêtes complexes avec des jointures comme dans les bases de données relationnelles, ce qui accélère l'accès aux données. Elles sont idéales pour les environnements où les données sont massivement produites ou consultées en temps réel, comme les applications mobiles, les réseaux sociaux ou les systèmes de gestion de contenu. Le modèle dénormalisé (sans relations complexes) permet d’éviter les coûts de performances liés aux jointures, augmentant la rapidité des requêtes.

* 1. Absence de schéma rigide

Contrairement aux bases SQL, où un schéma doit être défini avant de commencer à stocker les données (définissant les colonnes et leurs types), les bases NoSQL permettent d’ajouter et de modifier des données librement sans avoir à se conformer à un schéma prédéfini.

Cela permet de modifier la structure des données à la volée. Par exemple, dans une base de documents comme MongoDB, chaque document peut avoir une structure différente (des champs supplémentaires ou différents types de données). Cette absence de contrainte est bénéfique pour les applications où les données évoluent rapidement ou sont imprévisibles, comme dans les applications web modernes où de nouvelles fonctionnalités ajoutent des types de données supplémentaires.

* 1. Tolérance aux pannes et haute disponibilité

Les bases de données NoSQL sont souvent réparties sur plusieurs serveurs ou datacenters pour garantir une réplication des données et assurer une tolérance aux pannes. Si un serveur tombe en panne, les données sont toujours accessibles à partir d'autres serveurs. Cela assure une disponibilité continue des données, même en cas de panne matérielle ou de problème réseau.

Des systèmes comme Cassandra ou Couchbase assurent une réplication automatique des données, ce qui permet de maintenir une haute disponibilité sans intervention manuelle. Cette caractéristique est particulièrement importante pour les applications critiques qui doivent fonctionner 24H/24, comme les services bancaires ou de commerce en ligne.

* 1. Cohérence éventuelle

Dans de nombreux systèmes NoSQL, en particulier ceux qui sont distribués, les bases de données favorisent ce que l'on appelle la cohérence éventuelle. Cela signifie que les données ne sont pas immédiatement synchronisées sur tous les serveurs/nœuds, mais elles le seront au bout d’un certain temps. Ce compromis est fait pour garantir la disponibilité et les performances, notamment dans les systèmes distribués. Par exemple, lorsqu'une donnée est mise à jour sur un serveur, elle peut ne pas être immédiatement visible sur d'autres serveurs, mais elle finira par l'être.

Cette approche est utile pour des applications comme les réseaux sociaux, où l’on privilégie l'accès rapide aux données même si cela implique un léger délai dans la synchronisation des mises à jour.

* 1. Absence de transactions ACID strictes

Dans les bases NoSQL, pour améliorer les performances et la scalabilité, certaines bases ne respectent pas toutes les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité). Elles adoptent souvent des modèles comme BASE (Basically Available, Soft state, Eventually consistent), où la disponibilité est priorisée sur la cohérence stricte.

Certaines bases NoSQL modernes, comme MongoDB, proposent néanmoins des transactions ACID pour certains types de requêtes (comme des transactions multi-documents), tout en offrant la flexibilité et les performances des modèles NoSQL.

1. Types de bases de données NoSQL et leurs champs d’application
   1. Modèle Clé-Valeur

Le modèle clé-valeur est très simple, chaque élément de données est stocké en tant que clé associée à une valeur. Il est souvent utilisé pour des données qui doivent être accédées rapidement, comme des sessions utilisateurs ou des caches.

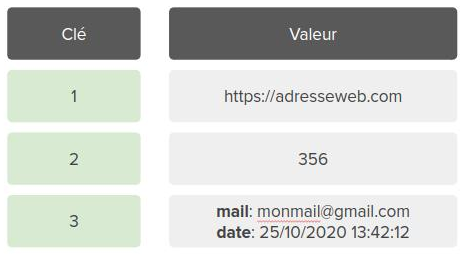


Figure 1:Le modèle clé-valeur

* + 1. Caractéristiques

C’est le modèle NoSQL le plus simple, qui stocke chaque élément sous la forme d’une clé unique associée à une valeur. Les valeurs peuvent être des chaînes de caractères, des listes, ou même des objets plus complexes. Très rapide en lecture et écriture, ce modèle est souvent utilisé pour des accès rapides en mémoire.

* + 1. Utilisations
* Ce modèle est utilisé pour des applications où la rapidité d’accès est cruciale, comme les systèmes de cache ou les sessions.
* Application : caching, sessions d’utilisateurs, gestion d’états temporaires.
* Gestion de sessions : Stockage temporaire des sessions utilisateurs (identifiants, paniers, informations temporaires).
* Cache**:** Utilisé pour stocker des résultats de calculs coûteux ou des requêtes de bases de données afin d’améliorer les performances.
* File d’attente : Les valeurs peuvent être utilisées pour gérer des files d’attente de tâches ou des listes d’actions à traiter.
  + 1. Exemple visuel

Clé : "session\_user\_12345"  
valeur : {  
 "nom" : "Dupont",  
 "connexion\_dernière": "2023-10-19",  
 "panier": ["produit\_1", "produit\_2"]

}

* 1. Modèle Documentaire

Les bases de données documentaires stockent des données sous forme de documents généralement encodés en JSON, BSON[[7]](#footnote-7) ou XML[[8]](#footnote-8). Chaque document peut avoir une structure différente et permet de stocker des objets complexes.



Figure 2: Le modèle documentaire

* + 1. Caractéristiques

Les bases de données documentaires stockent des données sous forme de documents, généralement au format JSON, BSON ou XML. Chaque document contient des paires clé-valeur et peut avoir une structure flexible, ce qui permet de stocker des objets complexes et variés. C’est un modèle orienté vers les objets et très utilisé pour des données semi-structurées.

* + 1. Utilisations
* Ce modèle est utilisé pour stocker des objets complexes comme les utilisateurs avec des détails tels que des adresses multiples et un historique d’achats.
* Applications : E-commerce, catalogues de produits, systèmes de gestion de contenu.
* Catalogues de produits : Les documents peuvent inclure des descriptions de produits avec des champs variés (nom, prix, catégories, etc.) sans avoir besoin d’une structure rigide.
* Systèmes de gestion de contenu : Gestion de pages Web, de posts de blogs, ou d’articles où chaque document représente une page avec son propre contenu.
* Applications mobiles : Les informations utilisateurs peuvent être stockées sous forme de documents (profils d’utilisateurs, historique d’activités).
  + 1. Exemple visuel

{  
 "nom" : "Article 123",  
 "catégorie" : "Électronique",  
 "prix" : 299.99,  
 "caractéristiques" : {  
 "marque" : "XYZ",  
 "poids" : "1.2kg",  
 "couleur" : "Noir" }  
}

* 1. Modèle en Colonnes

Le modèle de colonnes stocke des données dans des tables, mais contrairement aux bases de données relationnelles, chaque ligne peut avoir un nombre différent de colonnes. Ce modèle est efficace pour la gestion de larges volumes de données qui doivent être distribuées sur plusieurs serveurs.



Figure 3: Le modèle colonnes

* + 1. Caractéristique

Ce modèle stocke des données dans des tables, mais contrairement aux bases de données relationnelles, chaque ligne peut avoir un nombre différent de colonnes. Les données sont organisées par clé primaire, et chaque clé peut être associée à plusieurs colonnes. Ce modèle est distribué, avec une tolérance aux pannes, et est optimisé pour écrire des volumes massifs de données.

* + 1. Utilisation
* Le modèle en colonnes est utilisé pour des applications qui doivent stocker de grandes quantités de données structurées et semi-structurées.
* Applications : Systèmes de logs, systèmes de gestion des données transactionnelles.
* Systèmes de logs : Gestion d’événements continus où chaque entrée est enregistrée comme une ligne avec des colonnes différentes (heure, action, utilisateur, etc.).
* Analyse de données massives : Traitement de larges volumes de données où chaque ligne peut avoir des colonnes spécifiques à l’enregistrement.
* Big Data : Utilisé pour des applications nécessitant des écritures rapides et efficaces sur un cluster distribué.
  + 1. Exemple visuel

| Utilisateur | Action | Heure |
| --- | --- | --- |
| ID\_12345 | Connexion | 2023-10-19 10:00:00 |
| ID\_12345 | Ajout au panier | 2023-10-19 10:05:00 |
| ID\_67890 | Déconnexion | 2023-10-19 10:10:00 |

* 1. Modèle Graphe

Le modèle graphe est parfait pour représenter les relations complexes entre des entités. Les données sont stockées sous forme de nœuds (entités) et d’arêtes (relations), ce qui permet des requêtes très efficaces sur les relations entre les objets.

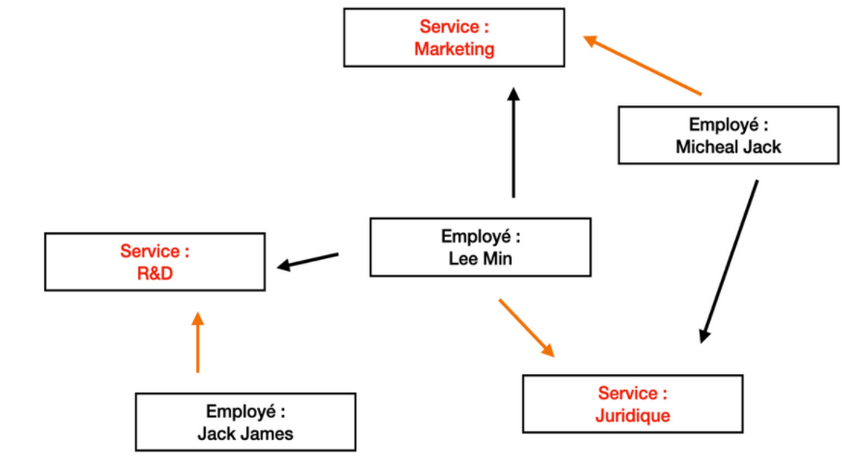


Figure 4: Le modèle graphe

* + 1. Caractéristiques

Les bases de données graphes sont conçues pour modéliser des relations complexes entre entités (nœuds) et les liens entre ces entités (arêtes). Chaque nœud représente un objet (par exemple, une personne) et chaque arête représente une relation (par exemple, une amitié entre deux personnes). Ce modèle est optimisé pour parcourir rapidement des relations complexes et trouver des chemins entre entités connectées.

* + 1. Utilisations
* Ce modèle est utilisé dans des applications qui nécessitent des analyses complexes de réseaux ou des relations entre des entités.
* Applications : Réseaux sociaux, moteurs de recommandations, détection de fraudes.
* Réseaux sociaux : Modélisation des utilisateurs et de leurs relations d’amitié ou de suivi.
* Moteurs de recommandations : Analyser les connexions entre les utilisateurs et les produits pour fournir des recommandations basées sur des achats ou des intérêts partagés.
* Détection de fraudes : Suivre les relations entre différentes transactions pour repérer des comportements suspects.
  + 1. Exemple visuel

(Alice)-[:AMI]->(Bob)  
(Bob)-[:ACHETÉ]->(Produit\_X)  
(Alice)-[:ACHETÉ]->(Produit\_Y)

1. Présentation de quelques bases de données NoSQL
   1. MongoDB



Figure 5:MongoDB

Caractéristiques et utilisation

Base documentaire, flexible, orientée JSON, scalable, utilisation idéale pour les objets complexes (documentaires). Elle est adaptée aux applications web et mobiles, stockage de données semi-structurées.

* 1. Redis



Figure 6: redis

Caractéristiques et utilisation

Base clé-valeur, mémoire rapide, supporte les structures de données complexes, modèle clé-valeur parfait pour les applications à haute performance (cache, sessions). Elle est utilisée comme cache en mémoire pour améliorer les performances des applications.

* 1. Cassandra



Figure 7: cassandra

Caractéristiques et utilisations

Base en colonnes, distribuée et conçue pour gérer des volumes massifs de données, gestion massive de données structurées (logs, big data). Elle est utilisée pour le traitement de grandes quantités de données sur plusieurs serveurs, idéal pour les applications critiques.

* 1. Neo4j



Figure 8:Neo4j

Caractéristiques et utilisation

Base orientée graphes, utilisée pour la gestion des relations complexes, gestion des relations complexes entre entités (réseaux sociaux, recommandations). Elle est utilisée pour les réseaux sociaux, systèmes de recommandation, gestion des fraudes.

1. Avantages et inconvénients des bases de données NOSQL
   1. Avantages

Par rapport aux bases de données traditionnelles, les NoSQL se distinguent par de nombreux bénéfices.

* Les bases de données NOSQL offrent une plus grande flexibilité, c’est-à-dire qu’elles s’adaptent à tous les types de données et elles n’exigent pas de structure fixes.
* Les bases de données NOSQL peuvent être stockées sur des serveurs différents en local ou en CLOUD ce qui permet de réduire considérablement la latence et la charge de travail des data centers pour les utilisateurs.
* Les NOSQL offrent une haute performance à l’utilisateur.
* Les bases de données SQL se sont avérées insuffisantes en termes de performance et d’évolutivité face à l’augmentation croissante de la masse de données. Les bases de données NOSQL s’adaptent à ces types de données.
* Les bases de données NOSQL n’appliquent aucune restriction sur les données. Elles permettent de stocker et de gérer des données structurées, semi-structurées et non structurées.
* Les modèles des bases NOSQL sont flexibles. Ils peuvent être modifiés de manière agile.
* Les NOSQL permettent d’éviter les goulets d’étranglement en répartissant la charge de travail sur plusieurs nœuds du cluster (scalabilité)
* Les NOSQL offrent une plus grande rentabilité aux entreprises avec l’élimination de certaines dépenses comme l’achat de serveur et les coûts d’installation et de maintenance informatiques.
  1. Inconvénients

Malgré leurs avantages, les bases de données NoSQL comportent quelques limites notables. Parmi ses limites nous avons :

* L’absence de règles de standardisation des données des bases de données NOSQL.
* Les bases de données NOSQL ne sont pas adaptées lorsqu’un utilisateur doit exécuter des requêtes complexes.
* Moins de ressources documentaires disponibles pour les NOSQL comparées au SQL.
* Moins de cohérences des données dans les NOSQL. La cohérence relâchée dans les NOSQL peut entrainer des complexités dans la gestion de l'intégrité des données ou nécessiter une logique d'application supplémentaire pour maintenir les niveaux de cohérence souhaités.
* Technologies utilisées dans les NOSQL sont moins mature par rapport aux bases de données SQL. Les recherches sont toujours en cours.

1. Bonnes méthodes pour optimiser les bases de données NoSQL

Avec l'augmentation exponentielle des données non structurées, des entreprises comme Netflix, Amazon et LinkedIn, ainsi que des organismes tels que la NASA et le CERN, se tournent vers les bases de données NoSQL pour gérer efficacement leurs volumes massifs de données. L'optimisation de ces bases devient alors cruciale pour répondre aux besoins spécifiques de chaque organisation en termes de gestion et d'efficacité.

Plusieurs méthodes peuvent être mises en place permettant de maximiser les avantages offerts par les systèmes NoSQL et de répondre aux exigences de leurs applications. Voici quelques-unes des principales méthodes d'optimisation à considérer :

* 1. Choisir un bon modèle NoSQL

Les bases de données NoSQL sont disponibles dans différents modèles, tels que clé-valeur, document, colonne, graphique ou hybride. Chaque modèle a ses propres forces et faiblesses, et il faudrait choisir celui qui convient le mieux à la structure des données, aux modèles de requête et aux cas d’utilisation.

Par exemple, si vous devez stocker et accéder à des relations complexes entre entités, une base de données de graphes peut être une bonne option. Si vous devez stocker et interroger des données semi-structurées, une base de données de documents peut être plus appropriée. Si vous devez accéder rapidement aux données, une base de données clé valeur est recommandée. Le modèle colonne quant à lui est efficace pour la gestion de larges volumes de données qui doivent être distribuées sur plusieurs serveurs.

* 1. Concevoir le schéma avec soin

Contrairement aux bases de données relationnelles, les bases de données NoSQL n’appliquent pas de schéma rigide, ce qui donne plus de flexibilité pour stocker et mettre à jour vos données. Cependant, cela signifie également que le schéma doit être conçu avec soin, en tenant compte de la façon dont les données seront interrogées, indexées et agrégées.

Par exemple, il est possible de dénormaliser les données et d’incorporer des documents ou des tableaux associés pour réduire le nombre de jointures ou de recherches. De plus, on peut aussi utiliser des conventions d’affectation de noms, des types de données et des formats cohérents pour éviter toute confusion et erreur.

* 1. Utiliser des index appropriés

Les index servent à optimiser les requêtes en réduisant le temps de recherche dans les grands ensembles de données. Au lieu de parcourir toute la base de données pour trouver un élément, l'index permet d'accéder directement aux données souhaitées en se basant sur des champs spécifiques. Ils sont essentiels pour améliorer les performances et l’efficacité des requêtes, mais ils consomment également de l’espace et des ressources. Par conséquent, il est recommandé d’utiliser des index appropriés qui correspondent aux modèles de requête choisis et d’éviter de créer des index inutiles ou redondants.

Par exemple, on peut utiliser des index composés pour prendre en charge plusieurs champs de requête ou utiliser des index hachés pour prendre en charge les requêtes d’égalité sur des champs volumineux. Il est possible également de surveiller et d’analyser l’utilisation et les performances des index afin d’identifier et de supprimer les index inutilisés ou inefficaces.

* 1. Optimiser les requêtes

Les requêtes sont la principale façon par laquelle on interagit avec les données, et elles peuvent avoir un impact significatif sur les performances des bases de données et la consommation des ressources. Par conséquent, il faudrait optimiser les requêtes pour les rendre aussi rapides et efficaces que possible.

Par exemple, on peut utiliser la projection pour limiter les champs que l’on renvoie où utiliser des pipelines d’agrégation pour effectuer des opérations complexes sur les données. Il est également recommandé d’éviter l’utilisation des opérateurs ou des fonctions qui nécessitent des analyses de collection complètes, telles que $where ou regex, et utiliser des plans d’explication pour analyser et optimiser l’exécution des requêtes.

* 1. Mise à l’échelle horizontale de la base de données

L’un des principaux avantages des bases de données NoSQL est leur capacité à évoluer horizontalement, ce qui signifie ajouter plus de nœuds ou de serveurs pour répartir la charge de travail et augmenter la capacité. Cependant, la mise à l’échelle horizontale implique également certains défis et compromis, tels que la cohérence, la disponibilité et le partitionnement des données. Par conséquent, la base de données doit être mise à l’échelle horizontalement en fonction des besoins et des attentes de l’entreprise, et utiliser les outils et techniques appropriés pour gérer et surveiller le cluster.

Par exemple, on peut utiliser le partitionnement pour répartir les données sur plusieurs nœuds ou utiliser la réplication pour augmenter la redondance et la disponibilité de des données.

* 1. Surveiller et ajuster les performances de votre base de données

Les bases de données NoSQL sont dynamiques et évolutives, tout comme les besoins et objectifs commerciaux. Par conséquent, il est important de surveiller et ajuster régulièrement les performances de la base de données pour s’assurer qu’elle répond à vos attentes et à vos normes.

Par exemple, il est possible d’utiliser des mesures et des alertes pour suivre et dépanner l’intégrité et les performances de la base de données, ou utiliser des bancs d’essai et des tests pour comparer et optimiser les configurations et les paramètres de la base. On doit également utiliser des stratégies de sauvegarde et de restauration pour protéger les données et garantir leur fiabilité et leur capacité de récupération.

Conclusion

En définitive, il ressort de cette étude que les bases de données NoSQL « not only SQL », non pas une contrepartie, mais bien un enrichissement et complément utile des bases de données SQL relationnelles traditionnelles constituent une avancée majeure dans la gestion des données contemporaines, répondant aux exigences d'une époque marquée par l’émergence du big data où rapidité, flexibilité et scalabilité sont essentielles.

Bien qu’elles offrent de nombreux avantages, les bases de données NoSQL révèlent toutefois certaines limites. Il sied alors d’adopter des stratégies adéquates de modélisation et d’optimisation, pour tirer pleinement parti de leurs potentiels.

L'émergence des bases de données NoSQL illustre l'évolution continue des besoins technologiques. Ainsi, une adoption réfléchie de ces technologies peut offrir un avantage significatif à ceux qui savent les exploiter.

Table des figures

[Figure 1:Le modèle clé-valeur 9](#_Toc181131862)

[Figure 2: Le modèle documentaire 10](#_Toc181131863)

[Figure 3: Le modèle colonnes 11](#_Toc181131864)

[Figure 4: Le modèle graphe 12](#_Toc181131865)

[Figure 5:MongoDB 13](#_Toc181131866)

[Figure 6: redis 14](#_Toc181131867)

[Figure 7: cassandra 14](#_Toc181131868)

[Figure 8:Neo4j 15](#_Toc181131869)

Bibliographie

 Bertrand, Y., & Phan, H.T. *Les bases de données NoSQL et le Big Data*. Disponible en ligne : [eni.fr](https://www.eni.fr)

 *NoSQL et Big Data : Comparaison des bases de données MongoDB, Cassandra, Redis* - [Journal du Net](https://www.journaldunet.com)

 Ministère de l’Économie, des Finances et de la Relance, *Le Big Data en France : Enjeux et applications pour les entreprises*. Disponible en ligne : economie.gouv.fr

 *MongoDB : une base NoSQL pour le Big Data et le Cloud Computing*, [OpenClassrooms](https://www.openclassrooms.com)

 *Les bases de données NoSQL : Présentation et cas d’usage*, [Developpez.com](https://www.developpez.com)

 Bertrand, Y., & Phan, H.T. (2018). *Les bases de données NoSQL : Principes et mise en œuvre*. Editions ENI.

 Sarrazin, T., & Benhamou, A. (2019). *Big Data et Machine Learning - Comprendre les bases de données et leur impact sur le traitement des données massives*. Dunod.

 Baude, F., & Beauvallet, G. (2017). *Introduction au Big Data et à l'Internet des objets : Concepts et outils pour les données massives*. Presses des Mines.

 Darmont, J. (2015). *Big Data, nouvelles données et nouveaux usages : L'importance de l’analyse de données massives*. Techniques de l'Ingénieur.

 Hug, H. (2016). *MongoDB - Gestion de Big Data en NoSQL*. Editions ENI.

 Zureik, E., & Fortin, C. (2014). *Le Big Data à découvert*. CNRS Editions.

 Villard, L. (2017). *Bases de données et Big Data : Introduction aux nouvelles architectures de gestion des données*. Editions ENI.

1. « **années 50** » fait référence à la période allant de **1950 à 1959**. [↑](#footnote-ref-1)
2. « **années 70** » fait référence à la période allant de **1970 à 1979**. [↑](#footnote-ref-2)
3. Informaticien britannique (23 août 1923 - 18 avril 2003). Source : Wikipédia [↑](#footnote-ref-3)
4. Fait référence à l'utilisation de l'architecture REST (Representational State Transfer) avec le protocole HTTP pour créer des **API Web**. [↑](#footnote-ref-4)
5. JSON : Java Script Objet Notation [↑](#footnote-ref-5)
6. ACID : Atomicité-Cohérence-Isolation-Durabilité [↑](#footnote-ref-6)
7. BSON : « Binary JSON » est un format binaire conçu pour représenter des données structurées de manière similaire au format JSON. [↑](#footnote-ref-7)
8. XML : « Extensible Markup Language » est un format de texte conçu pour structurer, stocker et transporter des données. [↑](#footnote-ref-8)