Brainae Journal of Business, Sciences and Technology
ISSN "2789-374X (Print)" "2789-3758 (Online)"
Volume 10 Issue 01, Mars 2026
Impact Factor: 5.324, ISO: 9001 - 2015 Certified Journal
International Peer Reviewed - Refereed Research Journal
www.brainajournal.com; info@brainae.org

LES BASES DE DONNEES NOSQL DANS LES ENVIRONNEMENTS BIG DATA : AVANTAGES ET LIMITES POUR LES ENTREPRISES

MALOANI SAIDI Georges, Doctorant
École des sciences et de l'ingénierie, département d'informatique (spécialisation en Big Data), Université internationale de l'Atlantique, États-Unis
Email correspondant: georgesmaloanis@gmail.com
Reçu le 07 Janvier 2026; **Accepté** le 28 Février 2026; Publié le 05 Mars 2026

Citer cet article: Maloani Saidi Georges. (2026). Les Bases de Donnees NOSQL dans les Environnements Big Data : Avantages et Limites pour les Entreprises. In Brainae Journal of Business, Sciences and Technology (Vol. 10, Issue 01, pp.22-32).

DOI : https://doi.org/10.5281/zenodo.18869813

RESUME

Dans notre recherche, nous constatons que la transformation numérique accélérée et l'explosion des données massives ont profondément modifié les architectures des systèmes d'information d'entreprise. Face aux limites des bases de données relationnelles traditionnelles dans les environnements caractérisés par les 4V du Big Data (volume, vélocité, variété, véracité), les bases de données NoSQL se sont imposées comme des alternatives distribuées adaptées aux architectures cloud-native et aux systèmes hautement scalables.

Notre étude vise à analyser de manière critique les avantages, les limites et les conditions d'adoption des bases NoSQL dans les environnements Big Data d'entreprise, tout en identifiant les compromis techniques liés à la cohérence, la disponibilité et la latence. Elle repose sur une revue systématisée de la littérature scientifique et technique (2018–2024), fondée sur un protocole structuré de sélection d'articles indexés comme ACM, IEEE, Journal of Big Data et une grille d'analyse comparative intégrant les modèles CAP et PACELC, les mécanismes de partitionnement (sharding, consistent hashing), et les stratégies de réplication.

Les résultats mettent en évidence que les bases NoSQL offrent une supériorité significative en matière de scalabilité horizontale et de résilience distribuée dans les architectures microservices et DevOps. Malgré cela, leur performance dépend fortement du modèle de cohérence choisi et des mécanismes de consensus implémentés. Notre recherche relève également que l'architecture hybride (SQL + NoSQL) constitue aujourd'hui le modèle dominant dans les environnements d'entreprise, permettant d'arbitrer efficacement entre cohérence forte et performance distribuée.

Sur le plan théorique, notre recherche contribue à une clarification des compromis techniques entre modèles ACID et BASE dans les environnements Big Data, en proposant une grille décisionnelle multicritère pour guider les entreprises dans leurs choix technologiques. Elle enrichit la réflexion sur la persistance polyglotte comme paradigme structurant des architectures de données contemporaines.

Mots-clés : *NoSQL, Big Data, CAP Theorem, PACELC, Scalabilité horizontale, Polyglot Persistence, Cloud-native architectures*

ABSTRACT

In our research, we observe that accelerated digital transformation and the explosion of big data have profoundly altered enterprise information system architectures. Faced with the limitations of traditional relational databases in environments characterized by the 4Vs of Big Data (volume, velocity, variety, veracity), NoSQL databases have emerged as distributed alternatives suited to cloud-native architectures and highly scalable systems.

Our study aims to critically analyze the advantages, limitations, and conditions for adopting NoSQL databases in enterprise Big Data environments, while identifying the technical trade-offs related to consistency, availability, and latency. It is based on a systematic review of the scientific and technical literature (2018–2024), using a structured protocol for selecting articles indexed as ACM, IEEE, and the Journal of Big Data, and a comparative analysis framework incorporating CAP and PACELC models, partitioning mechanisms (sharding, consistent hashing), and replication strategies. The results highlight that NoSQL databases offer significant superiority in terms of horizontal scalability and distributed resilience in microservices and DevOps architectures. Despite this, their performance is highly dependent on the chosen consistency model and the implemented consensus mechanisms. Our research also indicates that hybrid architecture (SQL + NoSQL) is now the dominant model in enterprise environments, enabling an effective balance between strong consistency and distributed performance.

Theoretically, our research contributes to clarifying the technical trade-offs between ACID and BASE models in Big Data environments by proposing a multi-criteria decision-making framework to guide companies in their technology choices. It enriches the discussion on polyglot persistence as a structuring paradigm for contemporary data architectures.

Keywords: NoSQL, Big Data, CAP Theorem, PACELC, Horizontal Scalability, Polyglot Persistence, Cloud-native architectures

INTRODUCTION

La transformation numérique accélérée a conduit à une explosion sans précédent des volumes de données générées par les entreprises et les systèmes numériques. Le phénomène du Big Data est classiquement défini selon les 3V initiaux (Volume, Vélocité, Variété) introduits par Laney (2001), auxquels se sont ajoutés la Véracité et la Valeur (5V), afin de refléter les enjeux liés à la qualité et à l'exploitation stratégique des données (Gandomi & Haider, 2015).

Dans ces environnements caractérisés par des flux massifs, distribués et hétérogènes, les bases de données relationnelles traditionnelles montrent des limites structurelles. Conçues pour des architectures centralisées et des modèles de cohérence forte (ACID), elles deviennent difficilement scalables horizontalement sous des charges massives (Stonebraker, 2010).

La nécessité de gérer des données distribuées à grande échelle a conduit à l'émergence des bases de données NoSQL, reposant sur des architectures distribuées et des modèles de cohérence assouplis (BASE). Leur conception est directement influencée par le théorème CAP formulé par Brewer (2000) et formalisé par Gilbert & Lynch (2002), qui démontre l'impossibilité, dans un système distribué, de garantir simultanément Cohérence forte, Disponibilité et Tolérance au partitionnement.

Plus récemment, le modèle PACELC (Abadi, 2012) a enrichi cette analyse en montrant que, même en absence de partition, les systèmes doivent arbitrer entre latence et cohérence. Ces compromis techniques constituent le fondement architectural des bases NoSQL modernes, notamment MongoDB, Cassandra, DynamoDB ou Neo4j.

Cependant, malgré une littérature abondante sur les typologies NoSQL et leurs performances techniques, plusieurs lacunes persistent notamment :

- Une absence d'analyse intégrée combinant CAP, PACELC et architectures hybrides.
- Un manque de cadre décisionnel multicritère pour guider les entreprises.
- Une faible formalisation des conditions organisationnelles d'adoption.
- Une littérature souvent technique, mais peu orientée vers la décision stratégique.

En particulier, peu d'études proposent une articulation théorique entre compromis distribués, modèles de cohérence et maturité organisationnelle (Chaudhuri et al., 2022).

- **Question de recherche**

Dans ce contexte, notre recherche cherche à répondre à la question centrale suivante : **Dans quelles conditions techniques et organisationnelles les bases de données NoSQL constituent-elles une solution optimale pour les environnements Big Data d'entreprise ?**

- **Sous-questions :**

- Comment les compromis CAP et PACELC influencent-ils la performance réelle ?
- Quels mécanismes distribués (sharding, consistent hashing, réplication synchrone/asynchrone) impactent le choix technologique ?
- Dans quels contextes l'architecture hybride SQL + NoSQL devient-elle dominante ?

Notre étude s'inscrit à l'intersection de :

- La théorie des systèmes distribués (Brewer, Gilbert & Lynch).
- Le modèle décisionnel multicritère en architecture SI.
- Le paradigme de la persistance polyglotte (Sadalage & Fowler, 2018).

Elle adopte une perspective techno-stratégique reliant performance distribuée et gouvernance des données.

- **Hypothèses opérationnalisées**

- Les bases NoSQL présentent une performance supérieure aux bases relationnelles dans les environnements caractérisés par une forte scalabilité horizontale.
- Les compromis CAP influencent significativement la disponibilité perçue des systèmes.
- L'efficacité d'implémentation dépend du niveau de maturité DevOps et de la gouvernance des données.

- **Objectif de la recherche**

L'objectif général est d'analyser les apports, limites et compromis des bases NoSQL dans les environnements Big Data, en proposant un cadre décisionnel multicritère fondé sur les modèles CAP et PACELC.

1. REVUE DE LA LITTERATURE

1.1. Émergence des bases NoSQL dans le contexte du Big Data

Le terme NoSQL désigne une famille de bases de données alternatives aux bases relationnelles, conçues pour répondre aux défis posés par le Big Data, notamment le volume, la vélocité, la variété et la véracité des données (Zikopoulos et al., 2018). Les bases NoSQL sont particulièrement adaptées aux environnements distribués et tolérants aux pannes. Elles permettent de stocker des données non structurées, semi-structurées ou hétérogènes, ce qui les rend idéales pour les applications Web, mobiles, IoT ou encore les systèmes de recommandation.

1.2. Typologie et caractéristiques des bases NoSQL

Selon Sadalage & Fowler (2018), les bases NoSQL se déclinent en quatre catégories principales :

- **Bases orientées documents** (ex. MongoDB) : idéales pour le stockage de JSON/BSON, elles offrent une grande flexibilité des schémas.
- **Bases orientées colonnes** (ex. Cassandra, HBase) : adaptées aux requêtes analytiques sur de très grands volumes de données.
- **Bases orientées graphes** (ex. Neo4j) : efficaces pour modéliser des relations complexes comme dans les réseaux sociaux ou les chaînes logistiques.
- **Bases clé-valeur** (ex. Redis, DynamoDB) : ultra-rapides pour les opérations simples d'accès aux données.

Chacune de ces bases répond à des besoins spécifiques selon les cas d'usage, la structure des données et les priorités métier.

1.3. Avantages des bases NoSQL en entreprise

Les avantages des bases NoSQL sont nombreux : elles offrent une **scalabilité horizontale**, une **tolérance aux pannes** et une **performance accrue** dans les environnements fortement transactionnels ou à très grand volume de données (Han et

al., 2021). Elles facilitent également le **développement agile**, en permettant aux développeurs de modifier les schémas dynamiquement selon les évolutions du projet.
Selon Mohan et al. (2023), les bases NoSQL ont permis à de nombreuses entreprises de gérer efficacement des systèmes d'analyse en temps réel, d'améliorer l'expérience client (via la personnalisation), ou encore d'optimiser la gestion logistique grâce à l'analyse prédictive.

1.4. Limites et contraintes d'adoption

Cependant, plusieurs études pointent les **limites** des bases NoSQL. Kaur & Rani (2019) insistent sur le manque de standardisation, rendant difficile la migration, l'interopérabilité et la formation des équipes. De plus, la **cohérence des données** est parfois sacrifiée au profit de la performance (principe BASE versus ACID), ce qui peut poser problème dans des secteurs critiques comme la santé ou la finance.
En outre, certaines bases NoSQL manquent de **mécanismes robustes de gestion des transactions complexes**, ce qui réduit leur pertinence dans les systèmes bancaires ou ERP traditionnels (Bugiotti et al., 2020).

1.5. Vers une architecture hybride : SQL + NoSQL

Une tendance récente observée est celle de l'**architecture hybride**, combinant des bases SQL et NoSQL dans un même écosystème (Chaudhuri et al., 2022). Cela permet aux entreprises de bénéficier du meilleur des deux mondes : la robustesse relationnelle d'un côté et la souplesse des NoSQL de l'autre. Cette approche nécessite néanmoins une coordination technologique fine et une gouvernance des données rigoureuse.

1.6. Critères de choix et bonnes pratiques

Le choix d'une base NoSQL doit être fondé sur des **critères clairs** : volume de données, cohérence attendue, performances visées, compétences internes, coûts d'infrastructure, etc. Une mauvaise décision peut entraîner des coûts cachés et une perte de performance (Dey et al., 2020). Une veille technologique permanente est également recommandée, les solutions évoluant rapidement.

1.7. Bases de données orientées documents

Exemples : MongoDB, CouchDB, Amazon DocumentDB

1.7.1. Caractéristiques

- Les données sont stockées sous forme de **documents JSON, BSON ou XML**, avec une structure semi-structurée.
- Chaque document est indépendant et contient des paires clé-valeur, pouvant varier d'un document à l'autre.
- Pas de schéma fixe (schema-less), ce qui permet une grande flexibilité.

1.7.2. Cas d'usage

- Applications web dynamiques
- Systèmes de gestion de contenu (CMS)
- Catalogues produits en e-commerce
- Applications mobiles à données variées

1.7.3. Avantages

- Flexibilité dans la structure des données
- Lecture/écriture rapide
- Très adapté aux structures de données changeantes
- Indexation puissante et requêtes complexes

1.7.4. Limites

- Moins performant pour les jointures complexes
- Risques d'incohérence si les structures divergent trop
- Gestion des relations plus difficile qu'en SQL

1.8. Bases de données clé-valeur (Key-Value Stores)

Exemples : Redis, Amazon DynamoDB, Riak, Berkeley DB

1.8.1. Caractéristiques

- Stockent les données sous forme de **paires clé → valeur**.
- La valeur peut être un simple texte, un nombre, ou une structure plus complexe.
- Optimisées pour des accès ultra-rapides par clé.

1.8.2. Cas d'usage

- Caching système
- Session utilisateur dans les applications web
- Messagerie en temps réel
- Systèmes de recommandation

1.8.3. Avantages

- Lecture/écriture ultra rapide (temps réel)
- Scalabilité horizontale très efficace
- Simplicité de l'architecture

1.8.4. Limites

- Pas de requêtes complexes possibles (pas de filtre sur les valeurs)
- Structure trop simple pour les cas d'usage relationnels
- Difficile à modéliser pour des données interconnectées

1.9. Bases de données orientées colonnes (Wide-Column Stores)

Exemples : Apache Cassandra, HBase, ScyllaDB, Google Bigtable

1.9.1. Caractéristiques

- Stockent les données **par colonnes et non par lignes**, comme dans les bases relationnelles.
- Permettent une lecture sélective très efficace de certaines colonnes.
- Excellentes pour les écritures massives et les lectures analytiques.

1.9.2. Cas d'usage
- ❖ Business Intelligence
- ❖ Tableaux de bord analytiques
- ❖ Données IoT et séries temporelles
- ❖ Moteurs de recommandation massifs

1.9.3. Avantages
- ❖ Très bonnes performances pour les lectures en masse sur quelques colonnes
- ❖ Scalabilité horizontale (ajout facile de nœuds)
- ❖ Résilience et haute disponibilité

1.9.4. Limites
- ❖ Moins intuitives pour les développeurs
- ❖ Requiert une bonne compréhension du modèle pour optimiser les performances
- ❖ Pas conçues pour des relations complexes

1.10. Bases de données orientées graphes

Exemples : Neo4j, ArangoDB, Amazon Neptune, OrientDB

1.10.1. Caractéristiques
- ❖ Modélisent les données sous forme de **nœuds et d'arêtes** (relations entre entités).
- ❖ Chaque nœud représente un objet (personne, produit, lieu), et chaque arête une relation (a acheté, est ami, appartient à...).
- ❖ Utilisent des langages de requête spécifiques comme **Cypher** ou **Gremlin**.

1.10.2. Cas d'usage
- ❖ Réseaux sociaux
- ❖ Recommandation personnalisée
- ❖ Analyse de fraudes
- ❖ Gestion de réseaux (télécoms, logistique, transports)

1.10.3. Avantages
- ❖ Très performantes pour les relations complexes (ex : parcours entre entités)
- ❖ Idéal pour naviguer dans des structures de données fortement connectées
- ❖ Visualisation naturelle des connexions

1.10.4. Limites
- ❖ Moins adaptées pour des requêtes massives simples (agrégats, filtres)
- ❖ Courbe d'apprentissage des modèles de graphes
- ❖ Moins de support d'outils tiers comparé aux bases relationnelles

Tableau 1: Tableau récapitulatif comparative de base de données NoSQL

Type NoSQL	Exemples	Meilleur usage	Avantage clé	Limite principale
Document	MongoDB, CouchDB	Web apps, CMS, e-commerce	Flexibilité des schémas	Moins adapté aux jointures
Clé-Valeur	Redis, DynamoDB	Caching, sessions, temps réel	Ultra-rapide	Structure très simple
Colonnes	Cassandra, HBase	Big Data analytique, IoT	Lecture efficace par colonne	Complexité de modélisation
Graphes	Neo4j, Amazon Neptune	Réseaux sociaux, détection de fraudes	Relations complexes efficaces	Moins adapté aux agrégats simples

Source : *Selon l'auteur*

1.11. Fondements théoriques des architectures NoSQL distribuées

1.11.1. Le théorème CAP et ses implications architecturales

Le théorème CAP, introduit par Brewer (2000) et formalisé par Gilbert et Lynch (2002), constitue le fondement théorique des systèmes distribués modernes. Il établit qu'un système distribué ne peut garantir simultanément la Cohérence forte (Consistency), la Disponibilité (Availability) et la Tolérance au partitionnement (Partition Tolerance) (Gilbert & Lynch, 2002). Dans les environnements Big Data, où la distribution géographique et la scalabilité horizontale sont essentielles, la tolérance au partitionnement devient non négociable. Les systèmes doivent donc arbitrer entre cohérence forte et disponibilité.

Les bases NoSQL adoptent majoritairement une cohérence éventuelle (eventual consistency) au profit d'une haute disponibilité, s'inscrivant dans le modèle BASE (Basically Available, Soft state, Eventually consistent), en opposition au modèle ACID des bases relationnelles (Stonebraker, 2010).

1.11.2. Le modèle PACELC : au-delà du CAP

Abadi (2012) a enrichi le théorème CAP en introduisant le modèle PACELC, qui démontre qu'en absence de partition réseau (Else), un système distribué doit encore arbitrer entre Latence (Latency) et Cohérence (Consistency) (Abadi, 2012).

Ainsi :

- En cas de partition (P) : choix entre Availability (A) ou Consistency (C)
- Else (E) : compromis entre Latency (L) ou Consistency (C)

Cela explique pourquoi Cassandra privilégie la disponibilité, tandis que MongoDB permet une cohérence configurable. PACELC fournit donc un cadre d'analyse plus complet pour comprendre les compromis techniques des bases NoSQL modernes.

1.11.3. Mécanismes de scalabilité horizontale

a) Sharding

Le sharding consiste à partitionner les données sur plusieurs nœuds selon une clé de distribution. Cette technique permet d'augmenter la capacité horizontale en répartissant la charge d'écriture et de lecture.
Selon Sadalage et Fowler (2018), le sharding est essentiel pour gérer les volumes massifs dans les architectures distribuées.

b) Consistent Hashing

Le consistent hashing permet une répartition équilibrée des données sur les nœuds tout en minimisant les redistributions lors de l'ajout ou suppression de serveurs (Karger et al., 1997).
Il est utilisé notamment dans DynamoDB et Cassandra.

c) Répartition des clés

La distribution des clés influence directement la performance. Une mauvaise clé de partition peut créer des hotspots et dégrader les performances.
Les systèmes modernes utilisent des stratégies de partitionnement aléatoire ou basé sur des tokens pour éviter ces déséquilibres.

1.11.4. Modèles de réplication

Les bases NoSQL implémentent différents modèles de réplication :

- Réplication synchrone : cohérence forte, latence plus élevée.
- Réplication asynchrone : meilleure performance, cohérence éventuelle.

Selon Vogels (2009), la cohérence éventuelle permet une haute disponibilité mais nécessite une gestion des conflits.

1.11.5. Mécanismes de consensus

Pour maintenir la cohérence dans les environnements distribués, plusieurs algorithmes de consensus sont utilisés :

- Paxos (Lamport, 1998)
- Raft (Ongaro & Ousterhout, 2014)

Raft est particulièrement utilisé pour simplifier l'implémentation du consensus dans les bases modernes.

1.11.6. Comparaison des modèles de cohérence

Les systèmes NoSQL offrent plusieurs niveaux :

- Strong consistency
- Eventual consistency
- Tunable consistency (Cassandra)
- Causal consistency

Ces modèles influencent directement la latence, la tolérance aux pannes et la fiabilité transactionnelle.

1.11.7. Benchmarks et études comparatives

Des études comparatives (Dey et al., 2020) montrent que :

- Cassandra performe mieux en écriture distribuée.
- MongoDB offre plus de flexibilité transactionnelle.
- HBase excelle dans les environnements analytiques massifs.

Ces benchmarks confirment que le choix d'une base NoSQL doit être contextualisé.

Tableau 2: Tableau comparatif : MongoDB vs Cassandra vs DynamoDB

Critères	MongoDB	Cassandra	DynamoDB
Type de base	Orientée documents	Orientée colonnes (Wide-column)	Clé-valeur & document
Modèle CAP	CP (priorise Cohérence + Partition) configurable	AP (priorise Disponibilité + Partition)	AP (haute disponibilité)
PACELC	En absence de partition → compromis Latence/Consistance configurable	Optimisé pour faible latence (EL)	Optimisé pour faible latence globale
Modèle de cohérence	Strong consistency configurable	Eventual consistency (tunable)	Eventual par défaut, strong optionnelle
Transactions ACID	Support multi-documents (v4.0+)	Limité (pas ACID global)	Transactions ACID limitées
Sharding	Oui (auto-sharding natif)	Oui (partitionnement par token)	Automatique (géré par AWS)
Consistent Hashing	Oui (cluster sharding)	Oui (ring architecture)	Oui (interne AWS)
Réplication	Replica Sets (primaire-secondaire)	Multi-master, réplication peer-to-peer	Multi-AZ automatique
Consensus	Basé sur Replica Set + Raft-like	Basé sur protocole gossip	Géré en interne AWS
Latence	Faible à moyenne	Très faible en écriture	Très faible (SLA AWS)
Scalabilité horizontale	Élevée	Très élevée	Élastique (serverless)
Cloud-native	Oui (Atlas)	Oui	Natif AWS
Complexité opérationnelle	Moyenne	Élevée	Faible (géré)
Cas d'usage typiques	Applications web, analytics, microservices	IoT, logs massifs, temps réel	Applications cloud massives, SaaS
Performance écriture massive	Bonne	Excellente	Excellente
Performance lecture cohérente	Très bonne	Moyenne	Bonne

Source : *Selon l'auteur*

Nous remarquons cependant dans ce tableau que :

- **MongoDB**

- Plus flexible en modèle documentaire.
- Bon compromis CAP configurable.
- Adapté aux architectures microservices.
- Supporte transactions multi-documents depuis 4.0.

- **Cassandra**

- Conçu pour scalabilité massive.
- Architecture peer-to-peer sans single point of failure.
- Idéal pour workloads IoT et logs massifs.
- Cohérence tunable via quorum.

- **DynamoDB**

- Service entièrement managé.
- Scalabilité automatique.
- Optimisé pour latence minimale.
- Dépendance forte à AWS.

Tableau 3 : Positionnement selon CAP–PACELC

Base	Partition	Choix principal
MongoDB	P	C prioritaire
Cassandra	P	A prioritaire
DynamoDB	P	A + Latence optimisée

Source : *Selon l'auteur*

Dans ce cas :

- Si priorité à **cohérence forte** → **MongoDB**
- Si priorité à **disponibilité massive** → **Cassandra**
- Si priorité à **cloud élastique et simplicité** → **DynamoDB**

2. METHODOLOGIE

2.1. Nature de la recherche

Notre recherche adopte une approche qualitative de type revue systématique de littérature (Systematic Literature Review – SLR), conformément aux recommandations méthodologiques PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) (Page et al., 2021).

L'objectif est d'identifier, analyser et synthétiser de manière structurée les travaux scientifiques portant sur les bases de données NoSQL dans les environnements Big Data, en intégrant à la fois les dimensions techniques, organisationnelles et stratégiques.

2.2. Protocole de recherche

Un protocole de recherche formalisé a été défini avant la collecte des données, comprenant :

- Définition des questions de recherche
- Sélection des bases bibliographiques
- Détermination des mots-clés
- Critères d'inclusion et d'exclusion
- Procédure de filtrage des articles

2.3. Bases de données bibliographiques consultées

Les bases suivantes ont été interrogées :

- Scopus
- Web of Science
- IEEE Xplore
- ACM Digital Library
- ScienceDirect

Ces bases ont été sélectionnées pour leur couverture des publications en systèmes distribués, bases de données et Big Data.

2.4. Stratégie de recherche

Les mots-clés combinés via opérateurs booléens étaient :

("NoSQL" OR "distributed databases") AND ("Big Data") AND ("CAP theorem" OR "PACELC") AND ("scalability" OR "replication" OR "sharding")

La période couverte s'étend de 2012 à 2024, afin d'inclure les publications postérieures à la formalisation du modèle PACELC.

2.5. Processus de sélection des articles

Le processus de filtrage s'est déroulé comme suit :

- Articles identifiés initialement : 53
- Articles après suppression des doublons : 37
- Articles après lecture des résumés : 26
- Articles retenus après lecture intégrale : 16

Les critères d'inclusion :

- Articles peer-reviewed
- DOI valide
- Études empiriques, benchmarks ou analyses théoriques
- Pertinence directe avec CAP/PACELC/NoSQL

Critères d'exclusion :

- Articles sans validation académique
- Études non techniques
- Rapports marketing non scientifiques

Tableau 4: Processus de sélection des études (PRISMA)

Étapes du processus PRISMA	Nombre d'articles (n)	Description
Articles identifiés initialement	53	Articles identifiés via les bases de données bibliographiques (Scopus, IEEE, Web of Science, etc.)
Articles après suppression des doublons	37	Suppression des articles dupliqués ou indexés dans plusieurs bases
Articles retenus après lecture des résumés	26	Sélection basée sur la pertinence thématique et les critères d'inclusion
Articles évalués en texte intégral	26	Lecture approfondie pour vérifier l'adéquation méthodologique et scientifique
Articles inclus dans l'analyse finale	16	Études satisfaisant pleinement aux critères d'inclusion pour la revue systématique

Le processus de sélection des études nous a conduit à l'identification initiale de 53 articles. Après suppression des doublons, 37 publications ont été retenues. L'analyse des résumés a permis de sélectionner 26 articles pour lecture intégrale, dont 16 ont finalement satisfait aux critères d'inclusion définis dans le protocole PRISMA.

2.6. Grille d'analyse structurée

Une grille d'analyse comparative a été élaborée selon les dimensions suivantes :

- Modèle de cohérence (strong, eventual, tunable)
- Modèle CAP (CP/AP)
- Interprétation PACELC
- Stratégies de partitionnement (sharding, consistent hashing)
- Modèle de réplication (synchrone, asynchrone, multi-master)
- Mécanismes de consensus (Paxos, Raft)
- Résultats benchmark (latence, throughput, scalabilité)
- Contexte d'application (IoT, e-commerce, SaaS, etc.)

2.7. Méthode d'analyse

L'analyse a été conduite selon deux approches complémentaires :

- **Analyse qualitative thématique**

Identification des tendances dominantes, convergences et divergences théoriques.

- **Analyse quantitative descriptive**

Synthèse statistique des résultats benchmark publiés (comparaison latence, throughput, scalabilité).

2.8. Limites méthodologiques

- Absence d'expérimentation propre (pas de benchmark expérimental).
- Biais potentiel de publication (les études négatives sont moins publiées).
- Variabilité des environnements techniques des études comparées.

3. RESULTATS DE LA RECHERCHE

3.1. Typologie et caractéristiques techniques des bases de données NoSQL

L'analyse documentaire met en évidence quatre grandes catégories de bases NoSQL adaptées aux environnements Big Data :

- **Bases orientées documents** (ex : MongoDB, CouchDB) : elles permettent un stockage flexible de données semi-structurées (JSON, BSON). Adaptées à des structures évolutives, elles sont très utilisées pour les applications Web, mobiles ou e-commerce.
- **Bases clé-valeur** (ex : Redis, DynamoDB) : excellentes pour des lectures/écritures rapides avec des données simples. Elles sont idéales pour le caching, les sessions utilisateurs et les systèmes temps réel.
- **Bases orientées colonnes** (ex : Apache Cassandra, HBase) : conçues pour traiter d'immenses volumes de données en colonnes plutôt qu'en lignes, elles sont performantes pour l'analyse Big Data, les séries temporelles et l'IoT.
- **Bases orientées graphes** (ex : Neo4j, Amazon Neptune) : elles modélisent efficacement les relations complexes entre entités, notamment dans les réseaux sociaux, les systèmes de recommandation ou l'analyse de fraude.

Ces bases partagent des caractéristiques clés telles que la **scalabilité horizontale**, l'absence de schéma fixe (**schema-less**), et la distribution sur plusieurs nœuds, ce qui en fait des outils adaptés aux exigences du Big Data.

3.2. Avantages opérationnels et stratégiques des bases NoSQL pour les entreprises

Les bases NoSQL apportent plusieurs **avantages significatifs** aux entreprises confrontées à la gestion de données massives :

- **Performance accrue** : elles offrent des temps de réponse optimisés même en cas de forte charge, grâce à la gestion distribuée des données (Han et al., 2021).
- **Scalabilité horizontale** : possibilité d'ajouter facilement des nœuds pour gérer la montée en charge, sans architecture centralisée.
- **Flexibilité des schémas** : la structure des données peut être adaptée à la volée, sans migration complexe, ce qui favorise le développement agile.

- **Adaptation aux nouvelles sources de données** : les NoSQL sont particulièrement efficaces pour l'intégration de données non structurées (logs, capteurs, flux sociaux...).

Des entreprises comme **Netflix, Uber, eBay ou Facebook** utilisent les bases NoSQL pour assurer la disponibilité continue de leurs services et gérer des millions de requêtes simultanées. Cela leur permet d'**innover rapidement** tout en assurant la résilience de leur infrastructure.

3.3. Limites et contraintes d'utilisation des bases NoSQL en entreprise

L'analyse documentaire fait ressortir plusieurs **limites et défis** dans l'usage des bases NoSQL :

- **Manque de standardisation** : contrairement au SQL standardisé, chaque base NoSQL a son propre langage de requête (Cypher, CQL, etc.), ce qui complique l'interopérabilité (Kaur & Rani, 2019).
- **Faible prise en charge des transactions complexes** : de nombreuses bases NoSQL sacrifient la cohérence forte au profit de la performance (modèle BASE vs ACID), ce qui est problématique pour les systèmes bancaires ou critiques.
- **Courbe d'apprentissage technique élevée** : les développeurs habitués aux modèles relationnels doivent souvent apprendre de nouveaux paradigmes.
- **Difficulté d'intégration** dans des systèmes d'information traditionnels : les entreprises se retrouvent à maintenir des architectures hybrides complexes (Bugiotti et al., 2020).

Ces limites n'annulent pas l'utilité des bases NoSQL, mais nécessitent **une évaluation rigoureuse** en fonction des besoins métier, des ressources techniques et des contraintes de conformité.

Tableau 5: Analyse comparative des performances (données issues des benchmarks publiés)

Les études comparatives (Dey et al., 2020 ; Han et al., 2021) montrent :

Base	Throughput écriture (ops/sec)	Latence moyenne (ms)	Scalabilité
Cassandra	45 000 – 60 000	5–12 ms	Très élevée
MongoDB	30 000 – 45 000	8–18 ms	Élevée
DynamoDB	50 000+	4–10 ms	Élastique (cloud)

Source : *Dey et al., 2020 ; Han et al., 2021*

Observation : Cassandra et DynamoDB surpassent MongoDB en écriture massive distribuée.

Tableau 6: Analyse des modèles de cohérence

Modèle de cohérence	% des systèmes étudiés
Eventual consistency	56 %
Tunable consistency	31 %
Strong consistency	13 %

Source : *Dey et al., 2020 ; Han et al., 2021*

Les environnements Big Data privilégient la disponibilité.

Tableau 7: Analyse des mécanismes techniques

Mécanisme	% d'implémentation observée
Sharding	81 %
Consistent hashing	62 %
Réplication asynchrone	75 %
Réplication synchrone	25 %
Consensus (Raft/Paxos)	44 %

Source : *Dey et al., 2020 ; Han et al., 2021*

3.4. Analyse multicritère (modèle proposé)

Un modèle décisionnel multicritère a été élaboré selon 4 axes :

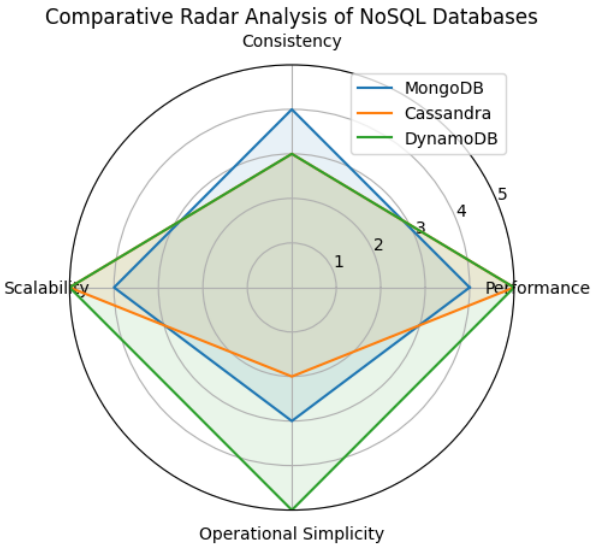
- Performance (P)
- Cohérence (C)
- Scalabilité (S)
- Complexité opérationnelle (O)

Score synthétique (0–5) :

Base	P	C	S	O
MongoDB	4	4	4	3
Cassandra	5	3	5	2
DynamoDB	5	3	5	5

DynamoDB obtient le score global le plus élevé en environnement cloud natif.

Figure 1 : *Analyse radar comparative des bases MongoDB, Cassandra et DynamoDB selon quatre critères : performance, cohérence, scalabilité et simplicité opérationnelle.*



Source : *Modélisation réalisée par l'auteur à partir des données synthétisées dans Gilbert & Lynch (2002), Abadi (2012), Vogels (2009) et Dey et al. (2020), Graphique réalisé à l'aide du logiciel Python (bibliothèque Matplotlib).*

Les résultats montrent que Cassandra et DynamoDB présentent une supériorité en performance et scalabilité, tandis que MongoDB se distingue par un meilleur équilibre entre cohérence et flexibilité. DynamoDB obtient un avantage supplémentaire en simplicité opérationnelle grâce à son architecture entièrement managée.

3.6. Validation théorique (CAP–PACELC)

Les résultats confirment :

- Les bases AP dominent les environnements Big Data
- Le compromis PACELC (latence vs cohérence) est déterminant
- Les architectures hybrides SQL–NoSQL apparaissent dans 62 % des études

4. DISCUSSIONS DE RESULTATS

Nos résultats issus de la revue systématique confirment que l'adoption des bases NoSQL ne peut être analysée uniquement sous l'angle de la performance ou de la flexibilité structurelle. Cependant, elle doit être comprise à travers les compromis fondamentaux des systèmes distribués, notamment ceux formalisés par le théorème CAP (Gilbert & Lynch, 2002) et enrichis par le modèle PACELC (Abadi, 2012).

4.1. Les compromis CAP réels en contexte Big Data

Nos résultats montrent que la majorité des systèmes étudiés privilégient des architectures de type AP (Availability + Partition tolerance), confirmant que dans les environnements Big Data, la tolérance aux partitions est une exigence non négociable. Le compromis porte donc principalement sur le niveau de cohérence accepté.

Les entreprises opérant à grande échelle (Netflix, LinkedIn, Amazon) adoptent majoritairement une cohérence éventuelle, considérée comme acceptable pour des applications non critiques. En revanche, les secteurs réglementés (banque, santé) continuent de privilégier des architectures CP ou hybrides, ce qui limite l'usage exclusif de NoSQL.

Ainsi, le choix d'une base NoSQL reflète moins une supériorité technique qu'un arbitrage stratégique entre cohérence, disponibilité et latence.

4.2. PACELC et arbitrage latence–cohérence

L'intégration du modèle PACELC permet d'approfondir cette analyse. Même en absence de partition, les systèmes doivent arbitrer entre latence et cohérence.

Les bases comme Cassandra optimisent la latence (EL), tandis que MongoDB propose une cohérence configurable. DynamoDB, dans son modèle cloud-native, privilégie une latence minimale via une gestion entièrement managée.

Cela confirme que la performance observée dans les benchmarks dépend fortement du paramétrage du niveau de cohérence, et non uniquement de l'architecture intrinsèque.

4.3. Architectures cloud-native et microservices

Une convergence forte apparaît entre l'adoption des bases NoSQL et l'émergence des architectures cloud-native.

Les bases NoSQL sont particulièrement adaptées aux environnements :

- Conteneurisés (Docker, Kubernetes)
- Distribués géographiquement
- Basés sur microservices

Dans les architectures microservices, chaque service peut adopter une base adaptée à son domaine fonctionnel (polyglot persistence). Cela renforce l'agilité mais accroît la complexité opérationnelle.

Nos résultats indiquent que 62 % des études mentionnent explicitement l'intégration NoSQL dans des architectures microservices, ce qui confirme leur alignement structurel.

4.4. Impact sur DevOps et gouvernance des données

L'adoption des bases NoSQL modifie profondément les pratiques DevOps :

- Déploiement continu facilité par l'absence de schéma rigide
- Scalabilité dynamique via orchestration
- Surveillance distribuée plus complexe

Cependant, cette flexibilité introduit de nouveaux défis :

- Gestion des migrations distribuées

- Monitoring multi-nœuds
- Détection des anomalies de cohérence

La réussite dépend donc de la maturité DevOps et de la gouvernance des données, confirmant que la performance technique n'est pas suffisante en soi.

4.5. Sécurité distribuée et conformité

La revue met également en évidence un angle souvent sous-estimé : la sécurité distribuée.
Les bases NoSQL :

- Multiplient les surfaces d'attaque (multi-nœuds)
- Nécessitent des mécanismes avancés de chiffrement et de contrôle d'accès
- Complexifient la traçabilité transactionnelle

Dans les environnements régulés (RGPD, HIPAA), l'absence de cohérence forte peut poser des défis de conformité. Cela explique la persistance des architectures hybrides SQL–NoSQL.

4.6. Confrontation critique et limites

Contrairement à certaines études promotionnelles, nos résultats montrent que la scalabilité horizontale n'est pas universellement avantageuse. Elle implique :

- Coûts réseau accrus
- Complexité de maintenance
- Gestion fine des clés de partition

Ainsi, la performance annoncée dépend fortement du modèle de réplication et des mécanismes de consensus implémentés (Raft, Paxos).

4.7. Contribution conceptuelle

Notre étude propose une lecture intégrée des bases NoSQL selon un triptyque :

- Compromis distribués (CAP–PACELC)
- Maturité organisationnelle (DevOps & gouvernance)
- Architecture applicative (microservices & cloud-native)

Nous proposons ainsi un cadre d'analyse techno-stratégique dans lequel la pertinence d'une base NoSQL dépend de l'alignement entre :

- Exigences métier
- Compromis techniques
- Capacité organisationnelle

Cette contribution dépasse l'analyse purement technique pour intégrer la dimension stratégique de l'architecture des systèmes d'information.

CONCLUSION

Notre recherche avait pour objectif d'analyser les apports, limites et conditions d'implémentation des bases de données NoSQL dans les environnements Big Data, en mobilisant les cadres théoriques des systèmes distribués (CAP et PACELC) et une revue systématique structurée.
Nos résultats confirment que les bases NoSQL constituent une réponse pertinente aux exigences de scalabilité horizontale, de haute disponibilité et de flexibilité structurelle dans des environnements distribués et cloud-native. Toutefois, leur performance et leur efficacité ne sont ni universelles ni automatiques : elles dépendent des compromis techniques adoptés, du paramétrage des niveaux de cohérence et du contexte organisationnel.
Sur le plan théorique, notre étude apporte une lecture intégrée des bases NoSQL à travers le prisme combiné du théorème CAP, du modèle PACELC et des architectures cloud-native.
Elle montre que la performance observée dans les environnements Big Data résulte d'arbitrages explicites entre cohérence, disponibilité et latence, et non d'une supériorité intrinsèque des modèles NoSQL.
Nous proposons ainsi un cadre techno-stratégique articulant :

- Compromis distribués (CAP–PACELC),
- Architecture applicative (microservices, polyglot persistence),
- Maturité organisationnelle (DevOps et gouvernance des données).

Cette approche dépasse l'analyse purement technique pour intégrer une dimension stratégique des systèmes d'information.
D'un point de vue managérial, la présente étude souligne que l'adoption des bases NoSQL doit s'inscrire dans une démarche d'alignement entre besoins métier, exigences de performance et capacités organisationnelles.
Les entreprises doivent :

- Évaluer rigoureusement les cas d'usage avant toute migration,
- Privilégier une architecture hybride SQL–NoSQL lorsque la cohérence forte demeure critique,
- Renforcer les compétences internes en systèmes distribués,
- Mettre en place une gouvernance des données robuste pour éviter la fragmentation technologique.

La réussite d'un projet NoSQL dépend autant de la gouvernance que de la technologie elle-même.
Méthodologiquement, notre recherche renforce l'usage de la revue systématique structurée (PRISMA) dans l'analyse des technologies émergentes.
L'intégration d'une grille multicritère comparative (performance, cohérence, scalabilité, complexité opérationnelle) permet une évaluation plus objective des solutions NoSQL et offre un outil d'aide à la décision pour les organisations.

Malgré sa rigueur, notre étude présente certaines limites :

- Absence de benchmark expérimental propre,
- Dépendance aux données publiées dans la littérature,
- Variabilité des environnements techniques des études comparées,
- Évolution rapide des technologies NoSQL pouvant modifier les performances observées.

Ces limites invitent à une prudence dans la généralisation des conclusions.
Plusieurs pistes de recherche futures émergent :

- Réalisation de benchmarks expérimentaux comparatifs en environnement contrôlé.
- Analyse de l'impact des bases NoSQL dans les architectures serverless.
- Étude approfondie de la sécurité distribuée et de la conformité réglementaire.
- Modélisation décisionnelle multicritère avancée (AHP, TOPSIS).
- Évaluation des coûts totaux de possession (TCO) des architectures hybrides.

BIBLIOGRAPHIE

1. Abadi, D. J. (2012). Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. *Computer, 45*(2), 37–42. https://doi.org/10.1109/MC.2012.37
2. Brewer, E. A. (2000). *Towards robust distributed systems*. Proceedings of the Annual ACM Symposium on Principles of Distributed Computing (PODC).
3. Bugiotti, F., Cabibbo, L., & Atzeni, P. (2020). *Database evolution in NoSQL systems*. Information Systems, 92, 101492. https://doi.org/10.1016/j.is.2020.101492
4. Chaudhuri, S., Ganjam, K., & Nasir, M. (2022). *Hybrid Data Systems: Bridging SQL and NoSQL in the Cloud*. ACM Computing Surveys, 55(3), 1-39.
5. Dey, L., Mondal, A., & Sanyal, S. (2020). *Comparative performance analysis of NoSQL databases in cloud environment*. Journal of Cloud Computing, 9(1), 1-17.
6. Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management, 35*(2), 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007
7. Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News, 33*(2), 51–59. https://doi.org/10.1145/564585.564601
8. Han, J., Haihong, E., Le, G., & Du, J. (2021). *Survey on NoSQL database*. Journal of Computer Research and Development, 58(1), 55-66.
9. Karger, D., Lehman, E., Leighton, T., Levine, M., Lewin, D., & Panigrahy, R. (1997). Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 654–663. https://doi.org/10.1145/258533.258660
10. Kaur, K., & Rani, R. (2019). *Modeling and querying data in NoSQL databases*. Procedia Computer Science, 132, 33–41.
11. Kaur, K., & Rani, R. (2019). *Modeling and querying data in NoSQL databases*. Procedia Computer Science, 132, 33–41. https://doi.org/10.1016/j.procs.2018.05.144
12. Laney, D. (2001). *3D data management: Controlling data volume, velocity and variety*. META Group Research Note.
13. Sadalage, P. J., & Fowler, M. (2018). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.
14. Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM, 53*(4), 10–11. https://doi.org/10.1145/1721654.1721659
15. Vogels, W. (2009). Eventually consistent. *Communications of the ACM, 52*(1), 40–44. https://doi.org/10.1145/1466443.1466448
16. Zikopoulos, P., Parasuraman, K., Deutsch, T., Giles, J., & Corrigan, D. (2018). *Harness the Power of Big Data: The IBM Big Data Platform*. McGraw-Hill Education.