Université Sultan Moulay Slimane

Département: ITG

Ecole Supérieure de Technologie de Béni Mellal

Module: Big Data & Bases de Données NoSQL (M236)

Filières: DUT IDIA & SIR - S3

Année universitaire: 2025-2026

Support de cours

Big Data & Bases de Données NoSQL

Chapitre III: Bases de données NoSQL

Pr. Siham BAKKOURI

Email: s.bakkouri@usms.ma

Plan

- INTRODUCTION AUX BASES DE DONNÉES NOSQL
- PRINCIPES FONDAMENTAUX DU NOSQL
- TYPOLOGIE DES BASES DE DONNÉES NOSQL
  - Bases clé-valeur
  - Bases orientées document
  - Bases orientées colonnes
  - Bases orientées graphes
- INTÉGRATION DANS L'ÉCOSYSTÈME BIG DATA

Introduction aux bases de données nosql

Au début de l'informatique, la majorité des données étaient structurées, c'est-à-dire organisées dans des tableaux bien définis (clients, produits, ventes…).

Les bases de données relationnelles (SQL) comme Oracle, MySQL ou PostgreSQL dominaient le marché depuis les années 1980.

Mais avec l'explosion du Web, des réseaux sociaux, des applications mobiles et de l'Internet des objets (IoT), le volume, la vitesse et la variété des données ont radicalement changé :

- Données non structurées (textes, images, vidéos, logs, capteurs)
- Données massives et continues (flux en temps réel)
- Besoin d'évolutivité rapide et de traitement distribué

Introduction aux bases de données nosql

Les systèmes SQL classiques ont montré leurs limites face à ces nouveaux besoins :

Difficulté à scaler horizontalement (ajouter de nouveaux serveurs facilement)

Rigidité du schéma de données (colonnes fixes)

Performances limitées sur de très gros volumes ou flux continus

→ Constat : Les entreprises comme Google, Amazon, Facebook ont dû créer de nouvelles architectures de bases pour gérer leurs gigantesques volumes de données. Ainsi est né le mouvement NoSQL vers la fin des années 2000.

Introduction aux bases de données nosql

→ Définition et signification du terme "NoSQL"

Le mot **NoSQL** signifie littéralement **“Not Only SQL”** (*pas seulement du SQL*).

Il ne rejette pas les bases relationnelles, mais les **complète** par de nouveaux modèles plus flexibles et adaptés au Big Data.

→ Une base NoSQL est une **base de données non relationnelle**, conçue pour :

- Gérer de grands volumes de données,
- Offrir de hautes performances,
- Permettre une scalabilité horizontale.

Elle privilégie la **souplesse** et la **performance** plutôt que la rigueur du modèle relationnel.

Exemple

Une base MongoDB peut stocker un document JSON contenant des informations variables d’un enregistrement à l’autre — sans schéma fixe comme dans SQL.

→ Les limites du modèle relationnel (SQL)

Les bases SQL reposent sur un **modèle relationnel** (tables, clés primaires, jointures, contraintes d'intégrité). Ce modèle est très puissant pour les données structurées, mais devient lourd dans les environnements Big Data.

❑ Principales limites (1/2)

1. Scalabilité verticale

Pour augmenter la performance, on doit renforcer le même serveur (CPU, RAM), ce qui coûte cher et a des limites physiques.

→ Le Big Data exige une **scalabilité horizontale** (plusieurs machines qui travaillent ensemble).

→ Les limites du modèle relationnel (SQL)

❑ Principales limites (2/2)

2. **Rigidité du schéma** :

   Dans SQL, chaque table a un schéma fixe. Ajouter une nouvelle colonne implique de modifier toute la structure.

   → Inadapté aux données hétérogènes et évolutives.

3. **Performances limitées pour le Big Data** :

   Les jointures et transactions multiples deviennent très coûteuses sur des milliards d'enregistrements.

4. **Difficulté à gérer la haute disponibilité**

   Le modèle ACID (Atomicité, Cohérence, Isolation, Durabilité) assure la fiabilité, mais limite la performance dans un cluster distribué.

Exemple

Sur un site e-commerce mondial, des millions de clients ajoutent des produits à leur panier en même temps.

Une base SQL classique pourrait devenir un goulot d'étranglement à cause du verrouillage des transactions.

→ Philosophie du mouvement NoSQL

Les bases NoSQL ont été conçues selon des principes différents :

- **Flexibilité du modèle de données** : pas besoin de schéma fixe.
- **Scalabilité horizontale** : ajout facile de serveurs pour supporter la charge.
- **Performance élevée** pour les lectures/écritures massives.
- **Disponibilité** avant cohérence stricte (principe BASE : Basically Available, Soft state, Eventually consistent).

→ Philosophie du mouvement NoSQL

❑ BASE vs ACID :

Modèle SQL (ACID)	Modèle NoSQL (BASE)
Atomicité	Basically Available
Cohérence stricte	Soft-state (état temporaire possible)
Isolation	Eventually Consistent (cohérence finale)
Durabilité	—

Dans le Big Data, on privilégie souvent la **disponibilité** et la **vitesse**, même si la cohérence parfaite est atteinte avec un léger délai.

→ Quand utiliser NoSQL ?

Le NoSQL n'est pas une solution magique ni universelle. Il complète les bases relationnelles selon les **besoins spécifiques** du projet.

❑ Situations adaptées au NoSQL

- Données non structurées ou semi-structurées (logs, JSON, images, documents, etc.)
- Besoin de haute disponibilité et de scalabilité horizontale
- Données massives générées à grande vitesse
- Modèle de données évolutif et non figé
- Traitement distribué ou cloud-native

→ Quand utiliser NoSQL ?

Exemples d'usages

Domaine	Exemple d'utilisation NoSQL
Réseaux sociaux	Stockage des profils et interactions utilisateurs
E-commerce	Catalogue produits ou panier client
IoT	Collecte de données de capteurs en temps réel
Big Data Analytics	Prétraitement avant Spark ou Hadoop
Jeux en ligne	Sessions utilisateurs et scores en direct

Principes fondamentaux du NoSQL

→ Une nouvelle approche du stockage de données

Les bases **NoSQL** se distinguent d'abord par leur manière d'organiser et de stocker les données. Elles ne s'appuient pas sur des tables et des relations fixes, mais sur des structures plus souples et distribuées, adaptées aux environnements Big Data.

Les données peuvent être :

- **non structurées** : texte libre, images, vidéos, logs…
- **semi-structurées** : JSON, XML, YAML…
- **très volumineuses** et réparties sur plusieurs serveurs.

NoSQL privilégie la simplicité, la vitesse et la flexibilité du stockage plutôt que la rigueur du modèle relationnel.

Principes fondamentaux du NoSQL

→ Le modèle distribué

Une caractéristique essentielle du NoSQL est son fonctionnement distribué. Cela signifie que la base de données n'est plus stockée sur un seul serveur, mais **répartie** sur un cluster (ensemble de machines).

❑ Avantages du modèle distribué

- **Scalabilité horizontale** : on peut ajouter des serveurs (nœuds) pour augmenter la capacité.
- **Tolérance aux pannes** : si un nœud tombe en panne, les autres continuent de fonctionner.
- **Performance** : plusieurs serveurs peuvent traiter les requêtes en parallèle.

Exemple

Dans un cluster MongoDB, les données sont automatiquement réparties entre plusieurs serveurs selon une clé de partition (ex. identifiant client).

Principes fondamentaux du NoSQL

→ Scalabilité horizontale vs verticale

La scalabilité (extensibilité) est la capacité d'un système à gérer plus de charge sans perdre en performance.

Type de scalabilité	Description	Exemple
Verticale	Ajouter plus de ressources (CPU, RAM) à un seul serveur.	Agrandir la machine SQL existante.
Horizontale	Ajouter de nouveaux serveurs au cluster.	Ajouter un nœud supplémentaire dans un cluster MongoDB.

- Les systèmes **SQL** sont généralement verticaux.
- Les systèmes **NoSQL** sont conçus pour la scalabilité horizontale, donc plus adaptés au Big Data.

Principes fondamentaux du NoSQL

→ Tolérance aux pannes et réplication

Dans un environnement distribué, la panne d'un serveur ne doit pas interrompre le service.

C'est pourquoi les bases NoSQL intègrent des mécanismes de **réplication** des données.

- Chaque donnée est copiée sur plusieurs nœuds.
- Si un nœud échoue, les autres peuvent fournir les copies.
- Un processus automatique restaure la cohérence dès que le nœud revient en ligne.

→ Consistance, disponibilité et partitionnement

Lorsqu'on répartit les données sur plusieurs serveurs, on rencontre un **problème fondamental** : comment garantir la cohérence quand tout est distribué ?

→ Pour répondre à cela, on introduit le **théorème CAP**.

Principes fondamentaux du NoSQL

→ Le théorème CAP

Le **théorème CAP** (Consistency, Availability, Partition tolerance) a été formulé par Eric Brewer en 2000. Il affirme qu'un système distribué **ne peut pas garantir simultanément les trois propriétés suivantes** :

Propriété	Description
C – Consistency (Cohérence)	Tous les nœuds voient les mêmes données au même moment.
A – Availability (Disponibilité)	Chaque requête reçoit une réponse, même si certains nœuds échouent.
P – Partition tolerance (Tolérance aux partitions)	Le système continue de fonctionner malgré la coupure de communication entre certains nœuds.

En pratique, un système doit **choisir deux propriétés sur trois**.

→ Le théorème CAP

→ En pratique, un système doit **choisir deux propriétés sur trois**.

Type de système	Choix privilégié	Exemple
CA	Cohérence + Disponibilité (systèmes centralisés)	Bases SQL classiques
AP	Disponibilité + Tolérance aux partitions	Cassandra, DynamoDB
CP	Cohérence + Tolérance aux partitions	MongoDB (dans certains modes)

Exemple

Si un serveur est coupé, faut-il bloquer l'accès pour garder les données cohérentes (CP) ou laisser les utilisateurs accéder à des copies non encore synchronisées (AP) ? Le choix dépend des besoins métiers.

Principes fondamentaux du NoSQL

→ Du modèle ACID au modèle BASE

Les bases relationnelles suivent le modèle **ACID**, qui garantit la fiabilité stricte des transactions :

- **A** : Atomicité
- **C** : Cohérence
- **I** : Isolation
- **D** : Durabilité

Mais dans le Big Data, ce modèle est souvent trop rigide et lent. Les bases NoSQL suivent plutôt le modèle **BASE**

Cela signifie que le système NoSQL **accepte une cohérence différée** pour garantir la rapidité et la disponibilité.

Principes fondamentaux du NoSQL

→ Principe BASE : cohérence finale

La cohérence finale (***eventual consistency***) est un concept clé du NoSQL. Cela signifie que : ***après une mise à jour, tous les nœuds ne sont pas immédiatement synchronisés, mais ils le deviennent après un court délai.***

Exemple

Un utilisateur met à jour son profil. Le changement apparaît immédiatement sur un serveur, et quelques secondes plus tard sur les autres.

→ **Ce compromis garantit des performances élevées sans bloquer les opérations.**

→ Structure flexible et sans schéma

Contrairement à SQL, les bases NoSQL ne nécessitent pas de schéma fixe. Chaque enregistrement peut avoir des champs différents, ce qui rend l'évolution très simple.

Exemple avec un document JSON :

```
{
   "nom": « Amina",
   "ville": «Beni-mellal",
   "téléphone": "0666666666"
}
```

Et un autre document dans la même collection :

```
{
   "nom": « Amine",
   "email": « amine@mail.com",
   "age": 19
}
```

Aucune erreur, car la structure est libre et extensible.

Principes fondamentaux du NoSQL

→ **Synthèse**

Caractéristique	SQL	NoSQL
Modèle de données	Tables et relations	Documents, clés-valeurs, colonnes, graphes
Schéma	Fixe	Flexible
Scalabilité	Verticale	Horizontale
Transactions	ACID	BASE
Cohérence	Forte	Éventuelle
Cas d'usage	Données structurées	Big Data, temps réel, non structurées

Le **NoSQL** repose sur trois piliers : la distribution, la flexibilité et la scalabilité horizontale. Il cherche un équilibre entre cohérence, disponibilité et performance, ce qui en fait une architecture idéale pour les systèmes modernes à grande échelle.

Typologie des bases de données NoSQL

Le terme **NoSQL** ne désigne pas un seul type de base de données, mais **une famille de systèmes** conçus pour des besoins variés :

- performances élevées,
- flexibilité de structure,
- stockage de données massives et distribuées.

Chaque type de base NoSQL répond à une problématique précise. On distingue généralement **quatre grandes catégories** :

1. **Bases clé–valeur**
2. **Bases orientées document**
3. **Bases orientées colonnes**
4. **Bases orientées graphes**

Typologie des bases de données NoSQL

→ Bases clé–valeur

Les bases **clé–valeur** stockent les données sous forme de **paires simples** :

clé → valeur

Chaque élément est identifié par une clé unique. C'est le modèle le plus simple, mais aussi le plus performant pour les recherches directes.

Exemple

Clé	Valeur
"session:102"	{"user":"Ali","panier":["PC","souris"],"montant": 5600}
"user:88"	{"nom":"Salma","ville":"Agadir"}

Typologie des bases de données NoSQL

→ Bases clé–valeur

❑ Principaux outils et technologies

- **Redis** : base en mémoire, très rapide, souvent utilisée comme cache distribué ou file de messages.
- **Amazon DynamoDB** : service NoSQL entièrement managé, hautement disponible et extensible.
- **Riak** : orientée haute tolérance aux pannes, utilisée dans les systèmes distribués à large échelle.

Typologie des bases de données NoSQL

→ Bases clé–valeur

❑ Les avantages

- Accès ultra rapide (temps réel)
- Très léger, simple à déployer
- Parfait pour le **cache**, les **sessions**, ou les **configurations temporaires**

❑ Les limites

- Pas de schéma complexe ni de jointures
- Inadapté pour les requêtes analytiques ou structurées

❑ Cas d'usage typique

- Stockage des sessions utilisateurs dans un site web
- Mise en cache d'informations fréquemment consultées (par exemple, le panier e-commerce)

Typologie des bases de données NoSQL

→ Bases orientées document

Ces bases stockent les données sous forme de **documents semi-structurés** (en **JSON**, **BSON** ou **XML**). Chaque document regroupe des informations complètes et autonomes.

Exemple d'un document JSON

```
{
  "client_id": 105,
  "nom": "Bakkouri",
  "ville": "Beni Mellal",
  "commandes": [
    {"produit": "Laptop", "montant": 8000},
    {"produit": "Clavier", "montant": 300}
  ]
}
```

Typologie des bases de données NoSQL

→ Bases orientées document

❑ Principaux outils et technologies

- **MongoDB** : le système NoSQL le plus répandu, avec des requêtes riches et un langage inspiré du JSON.
- **CouchDB** : repose sur le protocole HTTP et un modèle de synchronisation simplifié, idéal pour le travail déconnecté.
- **Elasticsearch** : moteur de recherche distribué basé sur Lucene, utilisé pour la recherche textuelle et analytique.

Typologie des bases de données NoSQL

→ Bases orientées document

❑ Les avantages

- Structure flexible : chaque document peut avoir ses propres champs.
- Facile à mettre à jour et à faire évoluer.
- Parfait pour les applications web modernes (JSON ↔ JavaScript).

❑ Les limites

- Risque d'incohérences entre documents similaires.
- Difficulté à gérer les relations complexes entre documents.

❑ Cas d'usage typique

- Gestion de contenu (CMS, blog, e-commerce).
- Applications mobiles avec synchronisation offline/online.
- Plateformes de recherche ou d'analyse textuelle.

Typologie des bases de données NoSQL

→ Bases orientées colonnes

Inspirées du modèle de Google **BigTable**, ces bases stockent les données **par colonnes** et non par lignes.

Cela permet d'optimiser les **requêtes analytiques massives** et le **stockage compressé**.

Représentation simplifiée

ID	Nom	Ville	Commande1	Commande2
1	Omar	Rabat	300	450
2	Salma	Agadir	800	200

Dans une base orientée colonnes, **chaque colonne est stockée séparément** :

```
colonne_nom → [Omar, Salma]
colonne_ville → [Rabat, Agadir]
```

Typologie des bases de données NoSQL

→ Bases orientées colonnes

❑ Principaux outils et technologies

- **Apache Cassandra** : hautement scalable, sans point unique de défaillance, utilisée par Netflix et Twitter.
- **HBase** : s'exécute au-dessus de HDFS et s'intègre facilement à l'écosystème Hadoop.
- **ScyllaDB** : alternative moderne à Cassandra, avec des performances très élevées.

Typologie des bases de données NoSQL

→ Bases orientées colonnes

❑ les avantages

- Excellentes performances pour les requêtes analytiques.
- Haute scalabilité horizontale (cluster).
- Résistance aux pannes.

❑ Les limites

- Complexe à configurer.
- Pas de support natif pour les relations (comme SQL).
- Moins pratique pour les petites transactions fréquentes.

❑ Cas d'usage typique

- Analyse des logs ou métriques d'un site web.
- Surveillance d'IoT à grande échelle.
- Systèmes de recommandation et statistiques.

Typologie des bases de données NoSQL

→ Bases orientées graphes

Ces bases représentent les données sous forme de **graphe** :

- des **nœuds** **(entités)**,
- des **arêtes** **(relations)**,
- et des **propriétés** sur chacun d'eux.

Elles sont conçues pour explorer les **relations complexes** entre éléments.

Exemple

```
(Imane) —[travaille_avec]→ (Ihssane)
(Imane) —[habite_à]→ (Beni Mellal)
```

Typologie des bases de données NoSQL

→ Bases orientées graphes

❑ Principaux outils et technologies

- **Neo4j** : le leader du marché, avec le langage de requête Cypher.
- **ArangoDB** : base multi-modèle (graphes + documents).
- **Amazon Neptune** : service géré par AWS, compatible Gremlin et SPARQL.

Typologie des bases de données NoSQL

→ Bases orientées graphes

❑ Les avantages

- Idéal pour les réseaux sociaux, les recommandations et la détection de fraudes.
- Requêtes très rapides sur les relations (via Cypher ou Gremlin).

❑ Les limites

- Moins adapté pour des volumes massifs sans lien entre données.
- Courbe d'apprentissage plus élevée.

❑ Cas d'usage typique

- Réseaux sociaux (Facebook, LinkedIn).
- Recommandations ("les clients ayant acheté X ont aussi acheté Y").
- Analyse de relations dans les bases scientifiques ou criminelles.

Typologie des bases de données NoSQL

→ **Comparatif général**

Type de base	Structure	Points forts	Points faibles	Cas d'usage
Clé–valeur	Paire clé/valeur	Rapide, simple	Pas de relations	Cache, sessions
Document	JSON/BSON	Flexible, facile à évoluer	Données redondantes	Web, mobile
Colonnes	Familles de colonnes	Très scalable, analytique	Configuration complexe	Logs, métriques
Graphe	Nœuds + arêtes	Requêtes relationnelles rapides	Moins scalable	Réseaux sociaux

Typologie des bases de données NoSQL

→ Comparatif général

❑ Synthèse et positionnement

Les bases NoSQL ne remplacent pas complètement les bases relationnelles ; elles viennent les **compléter** selon le **type de données** et le **besoin** :

- Pour la **rapidité et la simplicité** → Clé–valeur
- Pour la **souplesse de structure** → Document
- Pour la **performance analytique** → Colonnes
- Pour les **relations complexes** → Graphe

Le choix d'une base NoSQL dépend avant tout du **problème à résoudre**, pas seulement du volume de données.

Intégration dans l'écosystème Big Data

Les bases NoSQL ne fonctionnent pas isolément. Elles s'intègrent dans des pipelines de traitement complets :

- Ingestion via Kafka, Flume ou Spark Streaming.
- Stockage dans Cassandra, MongoDB ou HBase.
- Traitement avec Spark, Hive, ou Presto.
- Visualisation via Superset, Grafana ou PowerBI.

Ainsi, dans un projet Big Data, on combine souvent plusieurs outils : par exemple, Kafka pour le streaming, Spark pour le traitement, Cassandra pour le stockage, et Elasticsearch pour l'analyse et la recherche.