



# Bases de données NoSQL

---

Alanna DEVLIN GÉNIN

Lundi 9 septembre 2024

IUT Rives de Seine - BUT Science des Données - Parcours VCOD

7 séances de 3 heures

- Séance 1 : CM (lundi 9 septembre)
- Séance 2 : TP (jeudi 12 septembre)
- Séance 3 : TP (vendredi 13 septembre)
- Séance 4 : TP (mardi 24 septembre)
- Séance 5 : TP (vendredi 27 septembre)
- Séance 6 : TP (vendredi 11 octobre)
- Séance 7 : TP (vendredi 8 novembre)

# Programme

---

- Histoire des bases de données
- Caractéristiques principales des bases de données NoSQL
- Typologie des bases de données NoSQL

# Evaluation

QCM avec une ou deux questions ouvertes (30 à 45 minutes)

TP noté (avec rendu 24 ou 48 heures après la séance)

# Mon parcours académique



DUT STID



ENSAI

# Mon parcours professionnel



**LittleBigCode..**  
AI Solution Creator  
Stage fin d'études  
Data Engineer depuis 1 an et demi

# Contact

---

Si vous avez des questions n'hésitez pas à me contacter :

[alannadevlingenin@gmail.com](mailto:alannadevlingenin@gmail.com)

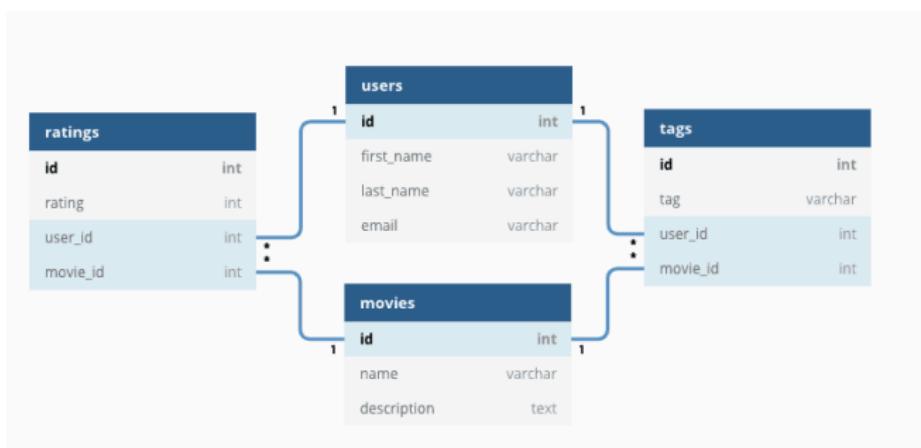
# Time for Kahoot !

## **Historique**

---

# Retour sur les bases de données relationnelles

Les bases de données relationnelles permettent de stocker des **données structurées** dans des **relations** (également appelées tables).

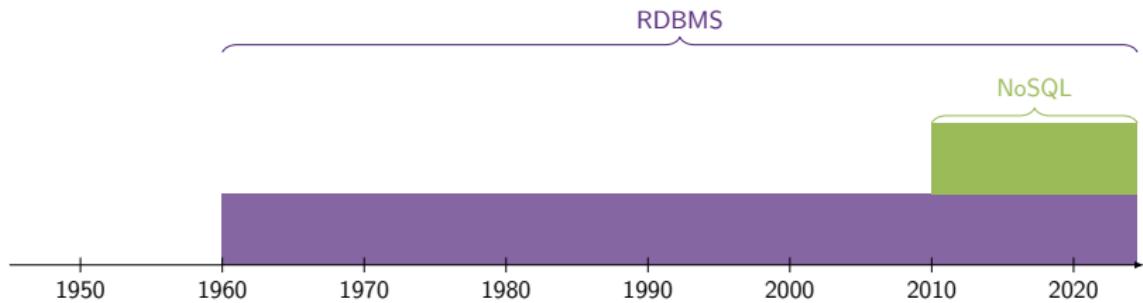


# Changement de paradigme

---

- **Années 2000** : explosion du volume de données
- **Modèle relationnel** : limité en termes de **scalabilité** et de **flexibilité**
- **Nouvelles technologies** : Big Data, IoT, applications web et mobiles

# Du modèle relationnel au NoSQL



# **Introduction au NoSQL**

---

# Pourquoi NoSQL ?

- Stockage et de traitement d'une grande quantité de données
- Scalabilité
- Haute disponibilité
- Réplication
- Partitionnement
- Indexation

# Mais que signifie NoSQL ?

## Définition

NoSQL signifie Not Only SQL ou No SQL.

En 2009, Johan Oskarsson définit NoSQL comme un terme générique pour désigner les bases de données qui ne sont pas des bases de données relationnelles.

NewSQL est un terme générique pour désigner les bases de données qui sont des bases de données relationnelles mais qui sont conçues pour être distribuées.

Examples : Google Spanner, CockroachDB, NuoDB, VoltDB, MemSQL, Clustrix, etc.

## **Concepts fondamentaux**

---

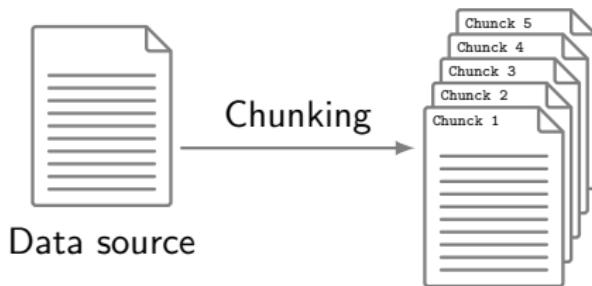
# Caractéristiques principales du NoSQL

- Schéma flexible
- Scalabilité
- Haute disponibilité
- RéPLICATION
- Partitionnement
- Indexation
- Transactions

# Qu'est-ce que le partitionnement ?

## Définition

Le **partitionnement** (ou *partitioning* en anglais) consiste à séparer les données en plusieurs *chunks*.



# Partitionnement ou sharding ?

# Partitionnement vs sharding

## Définition

Le **partitionnement** est une technique de division des données en plusieurs parties.

## Définition

Le **sharding** est une technique de distribution des données sur plusieurs nœuds.

# Partitionnement vs sharding

## PARTITIONING

ID	Name
1	Alice
2	Bob
3	Camille

ID	Name
4	Diane
5	Elodie
6	Fabrice

VERSUS

## SHARDING

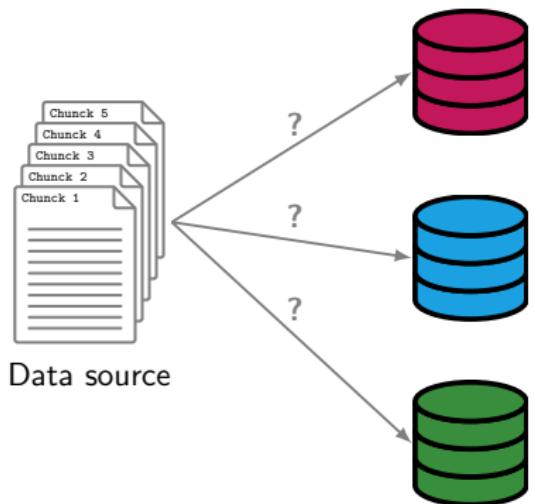
ID	Name
1	Alice
2	Bob
3	Camille

Shard 1

ID	Name
4	Diane
5	Elodie
6	Fabrice

Shard 2

# Différentes stratégies de partitionnement



## Cas d'usage

Camille Lemoine est propriétaire d'une chaîne de trois restaurants situés à Paris, Londres et Dublin. Elle souhaite stocker les données de ses clients dans une base de données en fonction de ses différents restaurants.

Identifiant	Prénom	Nom	Âge	Ville
1000	Alice	Dupont	30	Paris
1001	Bob	Brown	39	Londres
1002	Charlie	O'Brien	26	Dublin
1003	Daniel	Doyle	42	Dublin
1004	Edna	McCarthy	58	Dublin
1005	Fabienne	Lenoir	66	Paris
1006	Gilles	Morvan	71	Paris

Elle vous demande de l'aider à choisir la meilleure stratégie de partitionnement pour stocker les données de ses clients en fonction de leur localisation.

# Partitionnement par tourniquet

## Définition

Le **partitionnement par tourniquet** (*round-robin partitioning* en anglais) est une méthode de distribution des données où chaque enregistrement est successivement assigné à l'une des partitions disponibles de **manière cyclique**, assurant une répartition équilibrée de la charge



### Comment assigner une donnée $i$ à un nœud $n_j$ ?

La donnée située à l'index  $i$  sera assignée au nœud d'index  $j$  noté  $n_j$  avec  $n_j = i \bmod n$  et  $n$  est le nombre de nœuds.

# Partitionnement par tourniquet

Comment partitionner par tourniquet  
selon l'ID de l'utilisateur ?

ID	Prénom	Nom
1000	Alice	Dupont
1001	Bob	Brown
1002	Charlie	O'Brien
1003	Daniel	Doyle
1004	Edna	McCarthy
1005	Fabienne	Lenoir
1006	Gilles	Morvan

On considérera qu'un modulo 0  
(ie  $3 \bmod 3 = 0$ ) correspond au nœud 3.



Nœud 1



Nœud 2



Nœud 3

# Partitionnement par tourniquet

Comment partitionner par tourniquet  
selon l'ID de l'utilisateur ?

ID	Prénom	Nom
1000	Alice	Dupont
1001	Bob	Brown
1002	Charlie	O'Brien
1003	Daniel	Doyle
1004	Edna	McCarthy
1005	Fabienne	Lenoir
1006	Gilles	Morvan



Nœud 1



Nœud 2



Nœud 3

On considérera qu'un modulo 0  
(ie  $3 \bmod 3 = 0$ ) correspond au nœud 3.

Alice : user ID mod n =  $1000 \bmod 3 = 1$

# Partitionnement par tourniquet

Comment partitionner par tourniquet  
selon l'ID de l'utilisateur ?

ID	Prénom	Nom
1000	Alice	Dupont
1001	Bob	Brown
1002	Charlie	O'Brien
1003	Daniel	Doyle
1004	Edna	McCarthy
1005	Fabienne	Lenoir
1006	Gilles	Morvan



Nœud 1



Nœud 2



Nœud 3

ID	Prénom	Nom
1000	Alice	Dupont
1003	Daniel	Doyle
1006	Gilles	Morvan

ID	Prénom	Nom
1001	Bob	Brown
1004	Edna	McCarthy

ID	Prénom	Nom
1002	Charlie	O'Brien
1005	Fabienne	Lenoir

On considérera qu'un modulo 0  
(ie  $3 \bmod 3 = 0$ ) correspond au nœud 3.

# Avantages et inconvénients du partitionnement par tourniquet



**Lecture** : Aucune partition spécifique n'est surchargée, car les données sont uniformément réparties.

**Écriture** : Les écritures sont bien distribuées entre les partitions, évitant ainsi les goulets d'étranglement.



**Lecture** : Les requêtes qui nécessitent des recherches précises peuvent devoir scanner plusieurs partitions, augmentant ainsi le temps de lecture.

**Écriture** : Il n'y a pas de regroupement logique des données, ce qui peut compliquer la gestion et l'accès à des ensembles de données liés.

# Partitionnement par intervalle

## Définition

Le **partitionnement par intervalle** ou également appelé partitionnement par plage (*interval partitioning* en anglais) scinde les données en partitions basées sur des plages de valeurs spécifiques (par exemple, des plages de dates, de montants, etc.).



Le partitionnement par intervalle est souvent utilisé pour les données temporelles, où les enregistrements sont répartis en fonction de la date ou de l'heure de création.

# Partitionnement par intervalle

Comment partitionner par intervalle selon l'âge de l'utilisateur avec les catégories d'âge suivantes : 20 à 39 ans, 40 à 69 ans et 70 ans et plus ?

ID	Prénom	Nom	Âge
1000	Alice	Dupont	30
1001	Bob	Brown	39
1002	Charlie	O'Brien	26
1003	Daniel	Doyle	42
1004	Edna	McCarthy	58
1005	Fabienne	Lenoir	66
1006	Gilles	Morvan	71



Nœud 1



Nœud 2



Nœud 3

# Partitionnement par intervalle

Comment partitionner par intervalle selon l'âge de l'utilisateur avec les catégories d'âge suivantes : 20 à 39 ans, 40 à 69 ans et 70 ans et plus ?

ID	Prénom	Nom	Âge
1000	Alice	Dupont	30
1001	Bob	Brown	39
1002	Charlie	O'Brien	26
1003	Daniel	Doyle	42
1004	Edna	McCarthy	58
1005	Fabienne	Lenoir	66
1006	Gilles	Morvan	71



Nœud 1

ID	Prénom	Nom	Âge
1000	Alice	Dupont	30
1001	Bob	Brown	39
1002	Charlie	O'Brien	26



Nœud 2

ID	Prénom	Nom	Âge
1003	Daniel	Doyle	42
1004	Edna	McCarthy	58
1005	Fabienne	Lenoir	66



Nœud 3

ID	Prénom	Nom	Âge
1006	Gilles	Morvan	71

## Avantages et inconvénients du partitionnement par intervalle



**Lecture** : Efficace pour les requêtes qui filtrent sur une plage de valeurs spécifique, car elles accèdent uniquement à la partition pertinente.

**Écriture** : Facile à comprendre et à gérer, particulièrement utile pour les données temporelles ou séquentielles.



**Lecture** : Les requêtes qui ne filtrent pas sur la colonne de partitionnement ou couvrent plusieurs plages peuvent être moins efficaces.

**Écriture** : Les écritures peuvent se concentrer sur certaines partitions (par exemple, la partition la plus récente dans un partitionnement par date), créant un déséquilibre et un risque de surcharge de cette partition.

# Partitionnement par clé

## Définition

Le **partitionnement par clé (ou liste)** (*key partitioning* ou *list partitioning* en anglais) répartit les données en fonction de la valeur d'une ou plusieurs colonnes spécifiques, qui sont appelées les clés de partitionnement.



Le partitionnement par clé est souvent utilisé pour les données qui peuvent être regroupées en catégories distinctes, telles que les données géographiques, les catégories de produits, etc.

# Partitionnement par clé

Comment partitionner par clé selon la ville ?

ID	Prénom	Nom	Ville
1000	Alice	Dupont	Paris
1001	Bob	Brown	Londres
1002	Charlie	O'Brien	Dublin
1003	Daniel	Doyle	Dublin
1004	Edna	McCarthy	Dublin
1005	Fabienne	Lenoir	Paris
1006	Gilles	Morvan	Paris



Nœud 1



Nœud 2



Nœud 3

# Partitionnement par clé

Comment partitionner par clé selon la ville ?

ID	Prénom	Nom	Ville
1000	Alice	Dupont	Paris
1001	Bob	Brown	Londres
1002	Charlie	O'Brien	Dublin
1003	Daniel	Doyle	Dublin
1004	Edna	McCarthy	Dublin
1005	Fabienne	Lenoir	Paris
1006	Gilles	Morvan	Paris



ID	Prénom	Nom
1000	Alice	Dupont
1005	Fabienne	Lenoir
1006	Gilles	Morvan

ID	Prénom	Nom
1001	Bob	Brown

ID	Prénom	Nom
1002	Charlie	O'Brien
1003	Daniel	Doyle
1004	Edna	McCarthy

# Avantages et inconvénients du partitionnement par clé



**Lecture** : Les requêtes qui filtrent sur la clé de partition sont rapides, car elles accèdent directement à la partition appropriée.

**Écriture** : Les opérations d'écriture sont distribuées de manière équilibrée si la distribution des clés est uniforme, évitant ainsi les goulets d'étranglement.



**Lecture** : Si les requêtes ne filtrent pas sur la clé de partition, elles peuvent devoir parcourir plusieurs partitions, ce qui dégrade les performances.

**Écriture** : Si les clés sont mal réparties (skewed distribution), certaines partitions peuvent être surchargées, entraînant des déséquilibres.

# Partitionnement par hachage

## Définition

Le **partitionnement par hachage** (*hash partitioning* en anglais) distribue les données en fonction du résultat d'une fonction de hachage appliquée à une ou plusieurs colonnes.

$$f(x) = y$$

avec  $x$  la clé de partitionnement et  $y$  le nœud de destination.



Le partitionnement par hachage est souvent utilisé pour répartir uniformément les données et éviter les déséquilibres de charge. Les données sont réparties de manière aléatoire, mais déterministe.

# Partitionnement par hachage

Comment partitionner par hachage selon l'ID de l'utilisateur ?

ID	Prénom	Nom	Hash
1000	Alice	Dupont	$f(1000) = 2$
1001	Bob	Brown	$f(1001) = 1$
1002	Charlie	O'Brien	$f(1002) = 3$
1003	Daniel	Doyle	$f(1003) = 3$
1004	Edna	McCarthy	$f(1004) = 1$
1005	Fabienne	Lenoir	$f(1005) = 2$
1006	Gilles	Morvan	$f(1006) = 1$



Nœud 1



Nœud 2

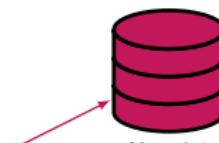


Nœud 3

# Partitionnement par hachage

Comment partitionner par hachage selon l'ID de l'utilisateur ?

ID	Prénom	Nom	Hash
1000	Alice	Dupont	$f(1000) = 2$
1001	Bob	Brown	$f(1001) = 1$
1002	Charlie	O'Brien	$f(1002) = 3$
1003	Daniel	Doyle	$f(1003) = 3$
1004	Edna	McCarthy	$f(1004) = 1$
1005	Fabienne	Lenoir	$f(1005) = 2$
1006	Gilles	Morvan	$f(1006) = 1$



ID	Prénom	Nom
1001	Bob	Brown
1004	Edna	McCarthy
1006	Gilles	Morvan



ID	Prénom	Nom
1000	Alice	Dupont
1005	Fabienne	Lenoir



ID	Prénom	Nom
1002	Charlie	O'Brien
1003	Daniel	Doyle

# Avantages et inconvénients du partitionnement par hachage



**Lecture** : Les requêtes qui utilisent la colonne de hachage sont rapides, car elles accèdent directement à la partition spécifique.

**Écriture** : Les données sont réparties de manière uniforme, minimisant le risque de surcharge d'une partition.



**Lecture** : Les requêtes qui ne filtrent pas sur la colonne de hachage peuvent nécessiter un balayage de plusieurs partitions.

**Écriture** : Le hachage peut rendre le regroupement logique des données plus difficile et compliquer les opérations de maintenance comme le rééquilibrage des partitions.

# Comparatif des différentes stratégies de partitionnement

Identifiant	Prénom	Nom	Âge	Ville
1000	Alice	Dupont	30	Paris
1001	Bob	Brown	39	Londres
1002	Charlie	O'Brien	26	Dublin
1003	Daniel	Doyle	42	Dublin
1004	Edna	McCarthy	58	Dublin
1005	Fabienne	Lenoir	66	Paris
1006	Gilles	Morvan	71	Paris

Partitionnement par tourniquet

Identifiant	Prénom	Nom	Âge	Ville
1000	Alice	Dupont	30	Paris
1001	Bob	Brown	39	Londres
1002	Charlie	O'Brien	26	Dublin
1003	Daniel	Doyle	42	Dublin
1004	Edna	McCarthy	58	Dublin
1005	Fabienne	Lenoir	66	Paris
1006	Gilles	Morvan	71	Paris

Partitionnement par intervalle (âge)

Identifiant	Prénom	Nom	Âge	Ville
1000	Alice	Dupont	30	Paris
1001	Bob	Brown	39	Londres
1002	Charlie	O'Brien	26	Dublin
1003	Daniel	Doyle	42	Dublin
1004	Edna	McCarthy	58	Dublin
1005	Fabienne	Lenoir	66	Paris
1006	Gilles	Morvan	71	Paris

Partitionnement par clé (ville)

Identifiant	Prénom	Nom	Âge	Ville
1000	Alice	Dupont	30	Paris
1001	Bob	Brown	39	Londres
1002	Charlie	O'Brien	26	Dublin
1003	Daniel	Doyle	42	Dublin
1004	Edna	McCarthy	58	Dublin
1005	Fabienne	Lenoir	66	Paris
1006	Gilles	Morvan	71	Paris

Partitionnement par hachage

# Résumé des performances

## Lecture

- **Meilleur** : Partitionnement par intervalle ou par clé (si les requêtes ciblent la clé ou l'intervalle).
- **Moins efficace** : Partitionnement par tourniquet ou par hachage pour des requêtes non ciblées.

## Écriture

- **Meilleur** : Partitionnement par hachage ou par tourniquet (distribution équilibrée des écritures).
- **Moins efficace** : Partitionnement par intervalle (risque de surcharge sur certaines partitions).

Le choix de la stratégie dépend du type de requêtes majoritaires (lecture ou écriture), de la distribution des données, et des besoins spécifiques de l'application en termes de performance.

# Scalabilité



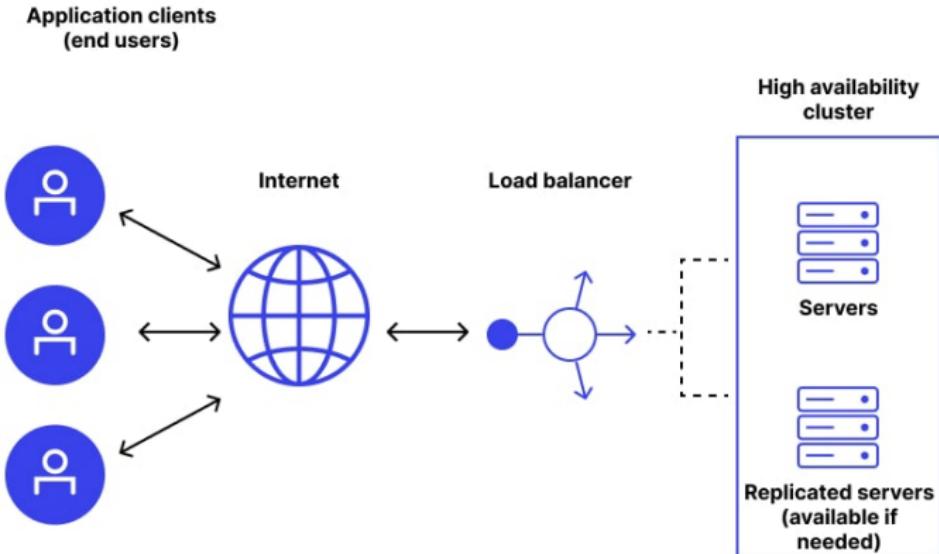
Vertical Scaling  
(Scaling up)

Horizontal Scaling  
(Scaling out)

Garantir un service disponible en tout temps

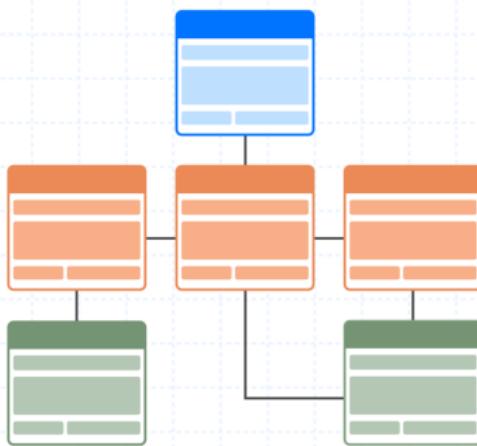
- RéPLICATION
- Partitionnement
- Tolérance aux pannes

# Haute disponibilité



# Quelle flexibilité de schéma ?

## Database schema



# ACID vs BASE

## The PH of a Database



ACID

BASE

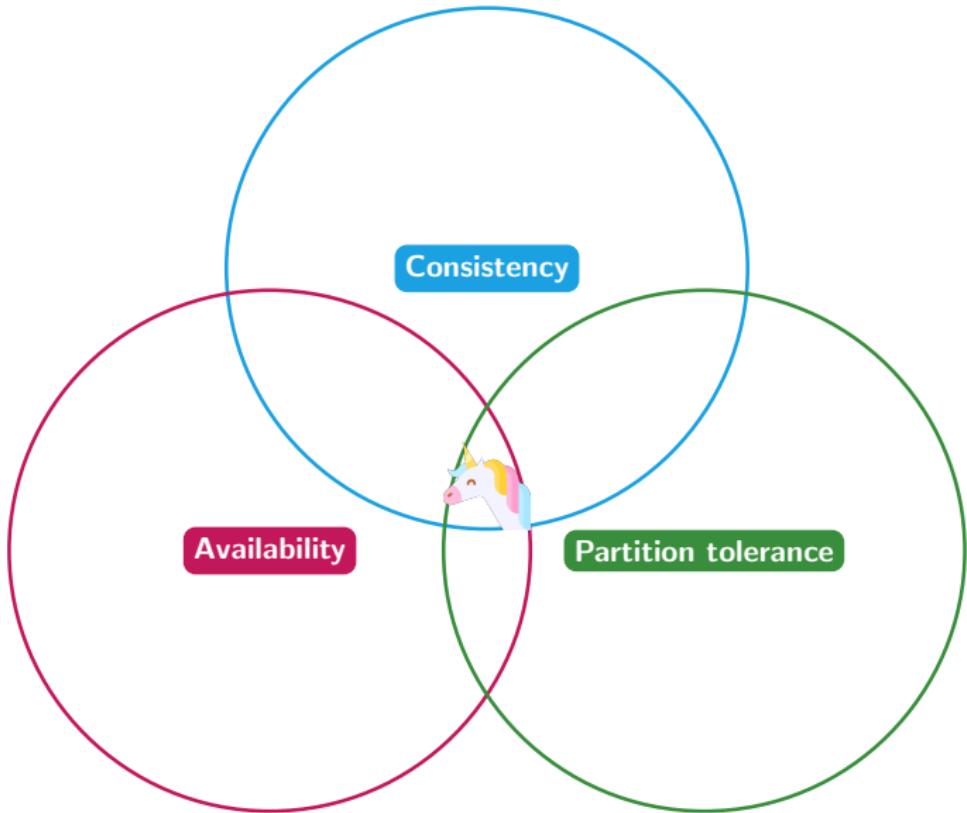
Flexibility	More rigid. Record access is blocked while processing.	More flexible. Multiple applications can simultaneously update the same record.
Performance	Low when processing large data volumes.	Successfully handles large, unstructured data with high throughput.
Scalability	Vertical	Horizontal
System Design	Rigid schema & model changes	Flexible in handling dynamic and evolving models
Synchronization	Yes and implies delays.	No synchronization

- Atomicité : Toutes les opérations d'une transaction sont exécutées ou aucune
- Cohérence : La base de données passe d'un état valide à un autre état valide
- Isolation : Les transactions s'exécutent indépendamment les unes des autres
- Durabilité : Les modifications sont persistantes

# Propriétés BASE

- Basically Available : Le système est toujours disponible
- Soft state : L'état du système peut changer
- Eventually consistent : Le système finit par être cohérent

# Théorème CAP



Selon Eric Brewer, un système informatique ne peut garantir simultanément que trois propriétés :

- Consistency : Toutes les données sont à jour
- Availability : Toutes les requêtes reçoivent une réponse
- Partition tolerance : Le système continue de fonctionner malgré les partitions réseau

## Avantages

- Scalabilité horizontale
- Flexibilité des schémas
- Performances pour certaines charges de travail

## Inconvénients

- Complexité de gestion
- Manque de standardisation
- Consistance éventuelle (CAP Theorem)

## Ce que NoSQL ne permet pas

- Joins
- Group by
- Transactions ACID
- Requêtes complexes
- Agrégations
- SQL
- Intégrations avec des applications basées sur SQL

# Quand utiliser NoSQL

- Grande quantité de données semi-structurées ou non structurées (logs, réseaux sociaux, IoT, time-based data)
- Améliorer les performances d'accès aux données en combinant le traitement de volumes de données plus importants, la réduction des temps de latence et l'amélioration du débit.

# Cas d'usage des bases de données NoSQL

- Applications web et mobiles
- Big Data et analytics
- Gestion de contenu et réseaux sociaux
- Internet des objets (IoT)

# Time for Kahoot !

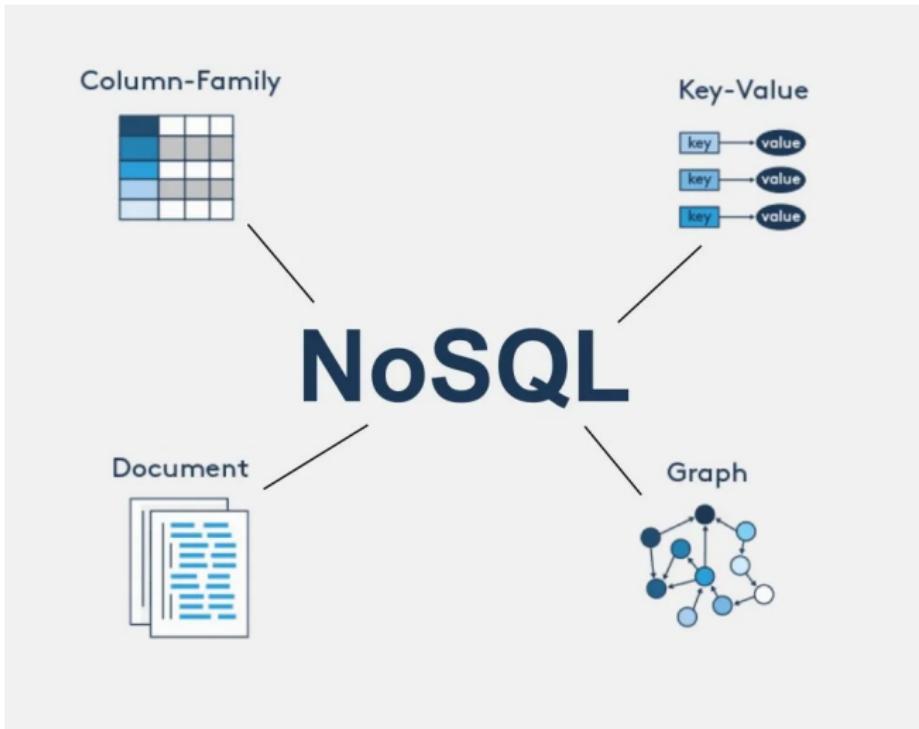
## **Typologie des bases de données NoSQL**

---

# Ecosystème NoSQL



# Typologie des bases de données NoSQL



# Caractéristiques des bases de données clé-valeur

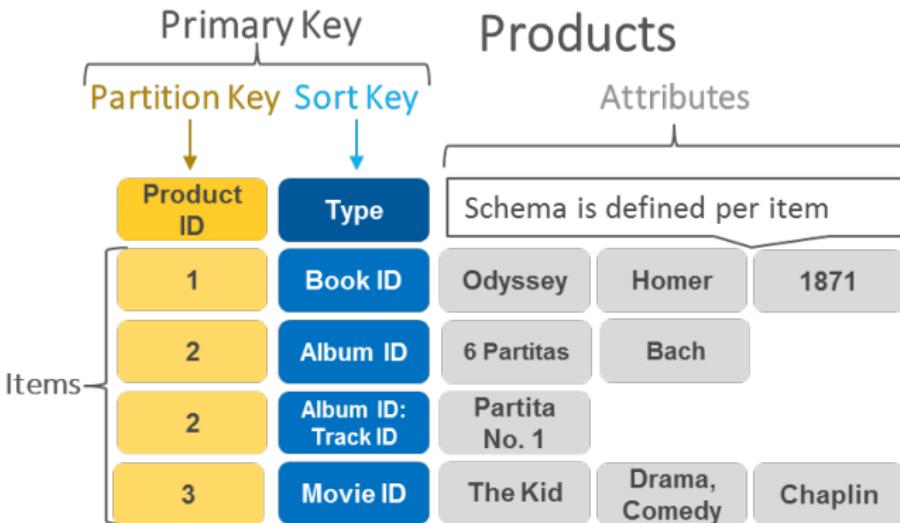
## Définition

Dans une base de données clé-valeur, les données sont stockées sous forme de **paires de clés uniques et de valeurs correspondantes**. La clé agit comme un **identifiant unique**, et la valeur peut être de tout type (chaîne de caractères, JSON, binaire, etc.).



- Très scalable avec une structure simple.
- Pas de schéma ou de structure rigide ; stockage de données flexible.
  - Idéal pour stocker de grandes quantités de données avec une complexité relationnelle minimale.

# Comment fonctionnent les bases de données clé-valeur ?



# Bases de données clé-valeur



**redis**



**MEMCACHED**



Amazon  
DynamoDB

**ORACLE®**  
**NOSQL DATABASE**

- **Stockage de sessions et profils d'utilisateurs**
  - Chaque utilisateur est référencé par une clé qui permet d'accéder à ses informations.
- **Paniers d'achats dans les applications de commerce électronique**
  - Gestion des commandes lors des soldes et de l'évolution de leur statut.
- **Moteur de stockage des métadonnées**
  - Pour une plateforme de jeu, cela permet d'accéder aux données des joueurs, l'historique des sessions et les tableaux de classement pour des millions d'utilisateurs simultanés.
- **Systèmes de cache (ex. : Redis, Memcached)**
  - Les applications de réseaux sociaux peuvent stocker des données fréquemment consultées, telles que le contenu des fils d'actualités.



## Avantages

- Lectures et écritures extrêmement rapides grâce à l'accès direct par clé.
- Scalabilité horizontale facile pour de grands ensembles de données.



## Inconvénients

- Absence de requêtes complexes.
- Mauvaise gestion du schéma.

# Caractéristiques des bases de données document

## Définition

Dans une base de données orientée document, les données sont stockées sous forme de **documents** structurés, généralement au format **JSON**, **BSON**, ou **XML**.

Chaque document représente un enregistrement, ce qui permet de stocker des données hiérarchiques et complexes.



- Très flexible : chaque document peut avoir une structure différente.
- Capable de stocker des données complexes et imbriquées.
  - Idéal pour des applications nécessitant un modèle de données flexible et des mises à jour fréquentes.

# Bases de données document

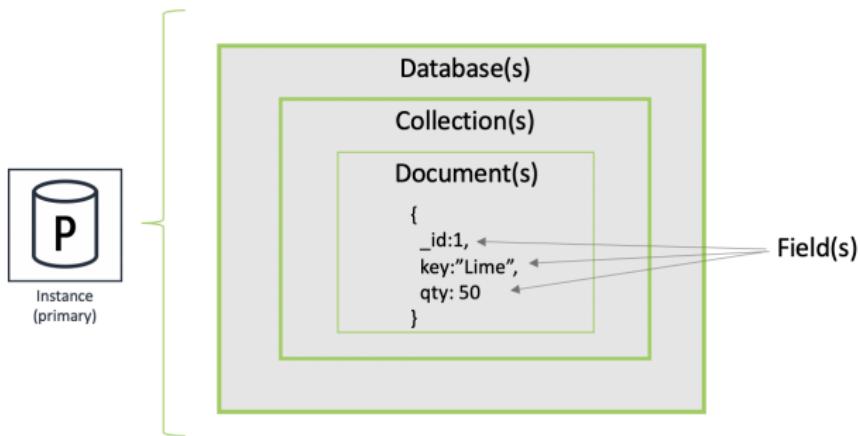


Firestore

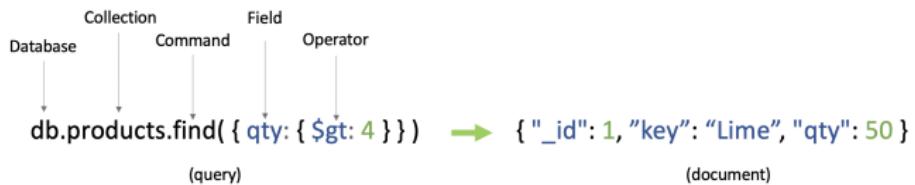
# Exemple de document JSON

```
1  [
2    {
3      "year" : 2013,
4      "title" : "Turn It Down, Or Else!",
5      "info" : {
6        "directors" : [ "Alice Smith", "Bob Jones"],
7        "release_date" : "2013-01-18T00:00:00Z",
8        "rating" : 6.2,
9        "genres" : ["Comedy", "Drama"],
10       "image_url" : "http://ia.media-imdb.com/images/N/09ERWAU7FS797AJ7LU8HN09AMUP908RL",
11       "plot" : "A rock band plays their music at high volumes, annoying the neighbors."
12       "actors" : ["David Matthewman", "Jonathan G. Neff"]
13     }
14   },
15   {
16     "year": 2015,
17     "title": "The Big New Movie",
18     "info": {
19       "plot": "Nothing happens at all.",
20       "rating": 0
21     }
22   }
23 ]
```

# Comment fonctionnent les bases de données document ?



# Comment fonctionnent les bases de données document ?



# Cas d'usages des bases de données document

- **Gestion du contenu sur les réseaux sociaux**

→ L'activité de chaque utilisateur est stockée sous forme d'un document contenant des informations imbriquées telles que les posts, commentaires, et connexions.

- **Plateformes de gestion de contenu (CMS)**

→ Stockage de documents flexibles qui contiennent des informations sur des articles de blog, des images, des métadonnées, etc.

- **Gestion des capteurs**

→ L'Internet des objets (IoT) permet de collecter un grand volume de données brutes à partir de capteurs (données non structurées et évolutives).



## Avantages

- Grande flexibilité dans le schéma, chaque document peut avoir une structure différente.
- Bonne gestion des données hiérarchiques ou imbriquées.
- Scalabilité horizontale facile grâce à la partition des documents.



## Inconvénients

- Performances limitées pour des requêtes complexes ou des jointures entre documents.
- Les mises à jour simultanées de documents imbriqués peuvent être plus complexes à gérer.

## Définition

Dans une base de données orientée colonne, les données sont stockées par **colonnes** plutôt que par lignes.

Chaque ligne correspond à une clé unique, et les colonnes stockent des paires de **nom-colonne : valeur**, ce qui permet de regrouper des colonnes similaires pour une meilleure performance des requêtes en lecture.

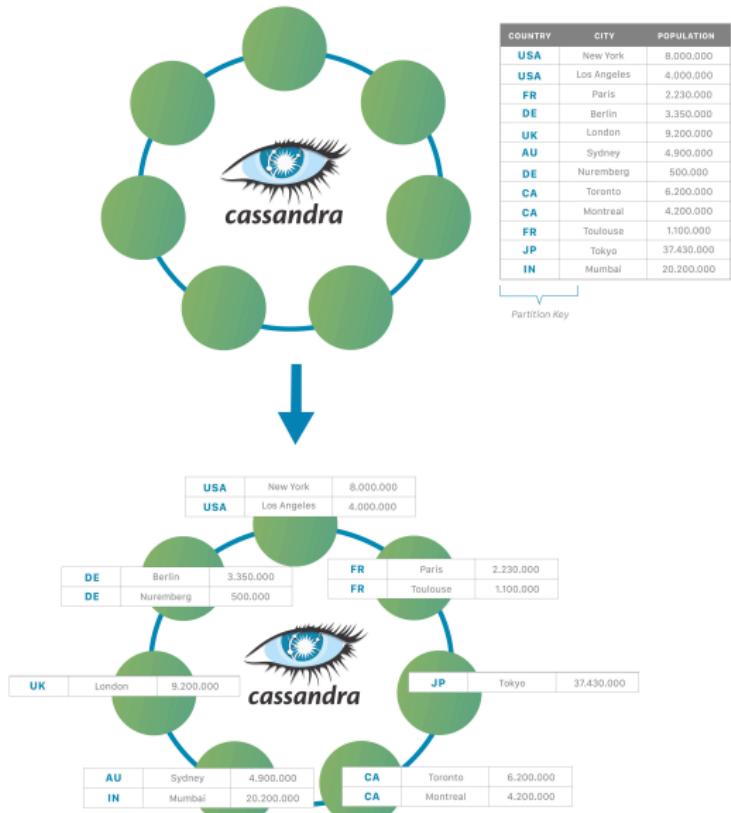


- Optimisé pour les lectures massives sur des colonnes spécifiques.
- Très scalable et performant pour des requêtes analytiques.
  - Idéal pour des applications nécessitant des analyses de grandes quantités de données.

# Bases de données orientées colonne



# Exemple : distribution des données sur Cassandra



- **Systèmes de gestion des données financières**
  - Permet d'extraire rapidement des colonnes spécifiques comme les prix ou les transactions sur de grands ensembles de données.
- **Outils analytiques et de Big Data**
  - Optimisé pour les requêtes analytiques complexes sur des jeux de données massifs.
- **Moteurs de recommandation**
  - Stocke des informations sur les utilisateurs et leurs préférences sous forme de colonnes, permettant une extraction rapide pour les recommandations en temps réel.



## Avantages

- Très performant pour les requêtes en lecture sur des colonnes spécifiques.
- Scalabilité horizontale adaptée pour le traitement de données massives.
- Idéal pour des applications analytiques ou des systèmes OLAP.



## Inconvénients

- Moins efficace pour des écritures fréquentes ou des transactions complexes.
- Complexité accrue pour gérer les relations entre colonnes.

# Caractéristiques des bases de données graph

## Définition

Dans une base de données graphe, les données sont organisées en **nœuds**, **liens** et **propriétés** qui permettent de modéliser des relations complexes et dynamiques entre les entités. Les **nœuds** représentent les entités, les **liens** représentent les relations entre ces nœuds, et les **propriétés** fournissent des informations supplémentaires sur les nœuds et les liens.

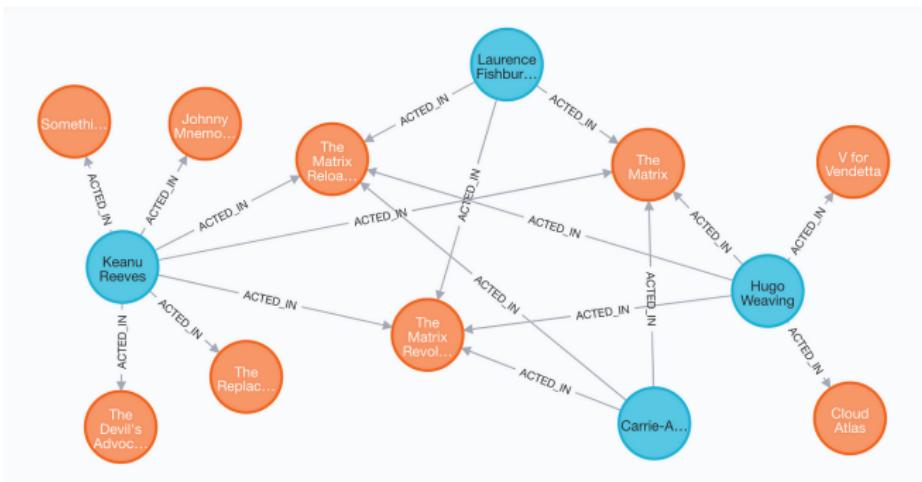


- Représentation des relations complexes entre les données.
  - Optimisé pour les requêtes de lecture et les opérations relationnelles.
- Idéal pour des applications nécessitant des relations complexes et dynamiques entre les données.

# Bases de données graphe



# Comment fonctionnent les bases de données graphe ?



# Cas d'usages des bases de données graphe

- **Réseaux sociaux**
  - Modélisation des relations entre utilisateurs, amis, groupes, et interactions pour des recommandations et des analyses sociales.
- **Détection de fraude**
  - Analyse des connexions entre transactions et entités pour identifier des schémas de fraude ou des comportements suspects.
- **Systèmes de recommandation**
  - Analyse des préférences des utilisateurs et des relations entre produits pour fournir des recommandations personnalisées.
- **Gestion des réseaux de télécommunications**
  - Modélisation des réseaux de télécommunications et des relations entre équipements pour optimiser les performances et la maintenance.



## Avantages

- Excellente performance pour les requêtes impliquant des relations complexes.
- Flexibilité pour modéliser des structures de données dynamiques.



## Inconvénients

- Moins efficace pour les opérations non liées aux relations.
- Complexité accrue pour les utilisateurs non familiers avec la modélisation en graphe.

# Time for Kahoot !