SAGIM 2024/2025

Introduction à MongoDB

Définition

MongoDB est une base de données NoSQL orientée documents. Contrairement aux bases de données relationnelles traditionnelles (comme MySQL ...), MongoDB stocke les données sous forme de documents JSON-like (Binary JSON).
Cela permet une grande flexibilité dans la structure des données, car chaque document peut avoir un schéma différent.

Définition

MongoDB a été créé en 2007 par la société 10gen, qui a ensuite changé son nom pour MongoDB Inc.
en 2013 le projet a été développé pour répondre aux besoins de stockage et de gestion de données pour des applications web modernes, qui nécessitaient une plus grande flexibilité et une meilleure scalabilité que ce que les bases de données relationnelles traditionnelles pouvaient offrir.

Pourquoi MongoDB ?

MongoDB est devenu populaire pour plusieurs raisons :

Flexibilité du schéma : Contrairement aux bases de données relationnelles, MongoDB n'impose pas de schéma strict.
Les documents dans une même collection peuvent avoir des structures différentes, ce qui permet une adaptation rapide aux changements de besoins métier.

Pourquoi MongoDB ?

Scalabilité horizontale :

MongoDB est conçu pour être facilement **scalable**.
Il supporte **le sharding**, qui permet de répartir les données sur plusieurs serveurs, ce qui est essentiel pour gérer de grandes quantités de données et des charges de travail importantes.

Pourquoi MongoDB ?

Performances élevées :

MongoDB est optimisé pour les applications modernes qui nécessitent des temps de réponse rapides.
Il utilise des index pour accélérer les requêtes et supporte des opérations en temps réel, ce qui le rend idéal pour les applications web, les analyses en temps réel, et plus encore.

Cas d'utilisation

Applications web :

MongoDB est souvent utilisé pour les applications web modernes qui nécessitent une gestion flexible des données et une mise à l'échelle facile.

IoT (Internet des Objets) :

Les systèmes IoT génèrent d'énormes quantités de données provenant de nombreux capteurs. MongoDB est capable de gérer ces données de manière efficace et scalable.

Cas d'utilisation

Big Data :

MongoDB est utilisé dans les environnements Big Data pour stocker et analyser de grandes quantités de données non structurées ou semi-structurées.

Analyses en temps réel :

MongoDB est capable de gérer des flux de données en temps réel, ce qui le rend adapté aux applications nécessitant des analyses et des prises de décision en temps réel.

Structure

Un document est une structure de données composée de paires **clé-valeur.**
Les valeurs peuvent être de différents types :

- **chaînes de caractères,**
- **nombres,**
- **tableaux,**
- **objets imbriqués, etc.**

Exemple :

```
{
  "_id": ObjectId("64b8f1c7e4b0a1a2b3c4d5e6"),
  "nom": ‘’mohammed",
  "âge": 30,
  "adresse": {  "ville": "Paris", "codePostal": "75001"},
  "hobbies": ["lecture", "voyage"]
}
```

Caractéristiques :

- Chaque document possède un identifiant unique _id (généré automatiquement si non spécifié).
- Les documents peuvent avoir des structures différentes au sein d'une même collection.

Collection

Définition :

Une **collection** est un groupe de documents MongoDB. Elle est l'équivalent d'une table dans une base de données relationnelle.

Caractéristiques :

Les **collections** n'imposent pas de schéma fixe, ce qui signifie que les documents dans une collection peuvent avoir des structures différentes. Les collections sont stockées dans une base de données.

Exemple :

Une collection utilisateurs peut contenir des documents représentant des utilisateurs avec des champs variés.

Base de données

Définition :

Une **base de données** est un conteneur pour les collections.
Elle est l'équivalent d'une base de données dans un système relationnel.

Caractéristiques :

Une **base de données** peut contenir plusieurs **collections**.
Chaque base de données est stockée dans un répertoire distinct sur le système de fichiers.

Exemple :

Une base de données blog peut contenir **des collections** comme articles, commentaires, et utilisateurs.

Installation et configuration

1 . Télécharger **MongoDB** Community Edition depuis le site officiel.

2 . Suivre les instructions d'installation.

3 . Ajouter le répertoire bin à la variable d'environnement PATH.

4 . Télécharger **MongoDB Compass** depuis le site officiel.

Introduction aux collections et documents

Base de données : Contient plusieurs collections.

Collection : Groupe de documents (équivalent à une table).

Document : Donnée structurée en format JSON.

Exemple de base de données :

Base : students_db **Collection** : students
Document :

```
{
  name: "Ali",  age: 22, subjects: ["math", "physics"], graduated: false
}
```

Introduction aux collections et documents

Commandes initiales :

1.Créer/Changer une base de données :

```
use students_db
```

Cela crée la base si elle n'existe pas.

2.Créer une collection :

```
db.createCollection("students")
```

3.Insérer un document dans une collection :

```
db.students.insertOne({ name: "Ali", age: 22,
    subjects: ["math", "physics"],
    graduated: false })
```

Ajouter des documents (INSERTION)

Pour insérer des données dans une collection, **MongoDB** fournit deux principales commandes :

insertOne() : Pour insérer un seul document.
insertMany() : Pour insérer plusieurs documents.

Ajouter des documents (INSERTION)

1. Insérer un seul document :

```
use users_db
db.createCollection("users")
db.users.insertOne({
    name: "Omar",
    age: 30,
    email: "omar@example.com",
    hobbies: ["football", "reading"]
})
```

Insérer plusieurs documents

1.Insérer plusieurs documents :

```
db.users.insertMany([
    { name: "Ali", age: 25, email: "ali@example.com" },
    { name: "Sara", age: 28, email: "sara@example.com" }
])
```

MongoDB ajoutera les deux documents dans la collection.

Lire des documents (READ)

Lire des données est l'une des tâches principales.
La commande pour lire les données est :

find()

Exemples :

1 - Afficher tous les documents :

db.users.find()

Filtrer les documents

Afficher les utilisateurs ayant un âge supérieur ou égal à 28 :

```
db.users.find({ age: { $gte: 28 } })
```

Chercher un utilisateur spécifique par son email

```
db.users.find({ email: "ali@example.com" })
```

Mettre à jour des documents (UPDATE)

MongoDB permet de mettre à jour les documents avec **updateOne(), updateMany()** ou encore **replaceOne().**

Exemples :

Mettre à jour un seul document :

Ajouter un numéro de téléphone pour Omar :

```
db.users.updateOne(
    { name: "Omar" },              // Filtre
    { $set: { phone: "123-456-7890" } } // Modification
)
```

Mettre à jour des documents (UPDATE)

Mettre à jour plusieurs documents :

Ajouter un champ status: "active" à tous les utilisateurs :

```
db.users.updateMany( {}, { $set: { status: "active" } }  )
```

Remplacer entièrement le document d'Ali :

```
db.users.replaceOne(
   { name: "Ali" },              // Filtre
   { name: "Ali", age: 26 }      // Nouveau document
)
```

Supprimer des documents (DELETE)

Pour supprimer des documents, tu peux utiliser **deleteOne()** ou **deleteMany().**

Exemples :

Supprimer un seul document :

Supprimer l'utilisateur avec l'email ali@example.com :

```
db.users.deleteOne({ email: "ali@example.com" })
```

Supprimer des documents (DELETE)

Supprimer plusieurs documents :

Supprimer **tous** les utilisateurs ayant un âge **inférieur** à 30 ans :

```
db.users.deleteMany({ age: { $lt: 30 } })
```

Requêtes avancées avec les opérateurs

MongoDB

Supprimer des documents (DELETE)

MongoDB fournit plusieurs opérateurs pour construire des filtres complexes dans les commandes **find().**

Opérateurs de comparaison

Ces opérateurs permettent de comparer des valeurs :

Opérateur	Signification	Exemple
$eq	Égal	{ age: { $eq: 25 } }
$ne	Différent	{ age: { $ne: 30 } }
$gt	Supérieur	{ age: { $gt: 20 } }
$gte	Supérieur ou égal	{ age: { $gte: 18 } }
$lt	Inférieur	{ age: { $lt: 50 } }
$lte	Inférieur ou égal	{ age: { $lte: 40 } }

Supprimer des documents (DELETE)

Exemple :

Trouver tous les utilisateurs supérieurs à 25 ans :

```
db.users.find({ age: { $gte: 25 } })
```