BASES DE DONNÉES

MongoDB

Base de données orientée documents

Cours complet — modèle de documents, CRUD, agrégation, réplication, sharding et cas d'utilisation pratiques

Sommaire

1 Qu'est-ce que MongoDB ?

Base de données NoSQL orientée documents, open source

MongoDB stocke des documents au format BSON (Binary JSON), regroupés dans des collections. Un document peut avoir des champs imbriqués (objets, tableaux), et deux documents d'une même collection n'ont pas besoin d'avoir exactement la même structure : c'est le schéma flexible.

```
{
  "_id": ObjectId("665f1a2b..."),
  "nom": "Dupont",
  "age": 28,
  "adresse": { "ville": "Orléans" },
  "tags": ["client", "premium"]
}
```

Schéma flexible

Pas de migration de schéma bloquante, structure évolutive.

Documents imbriqués

Un objet complet en un seul document, sans jointure.

Sharding natif

Montée en charge horizontale intégrée.

Agrégation puissante

Pipeline pour l'analytique sur données semi-structurées.

1 Terminologie : SQL vs MongoDB

Mêmes concepts, vocabulaire différent

SQL	MongoDB
Base de données	`Base de données`
Table	`Collection`
Ligne (row)	`Document`
Colonne	`Champ (field)`
Clé primaire	`_id (généré automatiquement si absent)`
Jointure (JOIN)	`$lookup (agrégation) ou embedding`

1 Pourquoi utiliser MongoDB ?

Un besoin métier → un apport concret

Besoin	Apport de MongoDB
Données à structure variable ou évolutive	Schéma flexible, pas de migration bloquante
Objets naturellement imbriqués (profil, commande)	Un document = un objet complet
Montée en charge horizontale	Sharding natif
Prototypage rapide	Pas de schéma à définir à l'avance
Analytique sur données semi-structurées	Framework d'agrégation

2 Installation et prise en main

Démarrer MongoDB en quelques secondes

Docker

```
docker run -d --name mongo-test \
  -p 27017:27017 mongo:7
```

Se connecter avec mongosh

```
docker exec -it mongo-test mongosh
test> use boutique
switched to db boutique
```

Premier document

```
boutique> db.produits.insertOne({
  nom: "Clavier mécanique",
  prix: 79.90,
  stock: 24
})
boutique> db.produits.find()
```

db.produits est créée implicitement au premier insert, sans déclaration préalable.

3 Le modèle de documents (BSON)

Types de données et identifiant _id

Type	Exemple
String	`"Marie"`
Number	`28, 79.90`
Boolean	`true`
Date	`ISODate("2026-08-18")`
Array	`["client", "premium"]`
Object (imbriqué)	`{ "ville": "Orléans" }`
ObjectId	`ObjectId("665f1a2b...")`

_id : identifiant unique par document, généré automatiquement (ObjectId) si non fourni.

4 CRUD — Créer et lire

insertOne, insertMany, find, findOne

Créer

```
db.produits.insertOne({
  nom: "Écran 27 pouces", prix: 249.00
})
db.produits.insertMany([
  { nom: "Casque", prix: 59.90 },
  { nom: "Webcam", prix: 39.90 }
])
```

Lire

```
db.produits.find()
db.produits.findOne({ nom: "Écran 27 pouces" })

// projection + tri + pagination
db.produits.find({}, { nom: 1, _id: 0 })
  .sort({ prix: -1 }).limit(5)
```

Cas d'usage : catalogue produit, profils utilisateurs, contenus éditoriaux.

5 CRUD — Modifier et supprimer

updateOne/Many, opérateurs, upsert, deleteOne/Many

Mettre à jour

```
db.produits.updateOne(
  { nom: "Clavier" },
  { $set: { prix: 74.90 },
    $inc: { stock: -1 } }
)
```

upsert : créer si absent

```
db.compteurs.updateOne(
  { _id: "visites" },
  { $inc: { total: 1 } },
  { upsert: true }
)
```

Supprimer

```
db.produits.deleteOne({ nom: "Webcam" })

db.produits.deleteMany({
  categorie: "obsolete"
})

// ⚠️ deleteMany({}) supprime TOUT
```

6 Requêtes avancées et opérateurs

Comparaison, logique, tableaux

Comparaison

```
find({ prix: { $gt: 50 } })
find({ prix: { $gte: 50, $lte: 100 } })
find({ categorie: { $ne: "audio" } })
find({ categorie: { $in: ["audio","info"] } })
```

Logique

```
find({ $or: [
  { categorie: "audio" },
  { prix: { $lt: 20 } }
] })
```

Tableaux & texte

```
find({ tags: "bestseller" })
find({ tags: { $all: ["promo","new"] } })
find({ "articles.produit": "Clavier" })

find({ nom: { $regex: "^Casque",
  $options: "i" } })
```

7 Index et performance

Accélérer les requêtes, éviter le collection scan

```
db.produits.createIndex({ nom: 1 })
db.produits.createIndex({ categorie: 1, prix: -1 })   // composé
db.utilisateurs.createIndex({ email: 1 }, { unique: true })
db.sessions.createIndex({ creee_le: 1 }, { expireAfterSeconds: 3600 })  // TTL
```

Vérifier l'utilisation

db.produits.find({...}).explain("executionStats")

IXSCAN = index utilisé (rapide). COLLSCAN = parcours complet (lent).

Bonnes pratiques

Indexer les champs filtrés/triés fréquemment. Ne pas sur-indexer : chaque index ralentit les écritures et occupe de l'espace disque.

8 Le framework d'agrégation

Un pipeline d'étapes successives

```
db.commandes.aggregate([
  { $match: { statut: "livree" } },
  { $group: { _id: "$client",
      total_depense: { $sum: "$montant" } } },
  { $sort: { total_depense: -1 } }
])
```

$match

Filtre les documents

$group

Regroupe et agrège (somme, moyenne…)

$lookup

Jointure avec une autre collection

$unwind

Déplie un tableau en plusieurs documents

Cas d'usage : tableaux de bord, statistiques de vente, rapports agrégés sans traitement côté application.

9 Modélisation — embedding vs référence

Choisir la bonne approche selon la relation

Embedding (imbriqué)

{ nom: "Marie", adresses: [{ ville: "Orléans" }] }

+ Une seule requête pour tout récupérer.
− Risque de duplication ou de document trop volumineux (limite 16 Mo).

Référencement

{ _id: 101, utilisateur_id: 1, montant: 129.90 }

+ Évite la duplication, adapté aux relations volumineuses.
− Nécessite $lookup pour reconstituer les données.

Relation one-to-few, lues ensemble	**Embedding**
Relation one-to-many volumineuse	**Référencement**
Donnée partagée entre plusieurs parents	**Référencement**

10 Réplication (Replica Set)

Haute disponibilité avec bascule automatique

```
rs.initiate({
  _id: "monReplicaSet",
  members: [
    { _id: 0, host: "mongo1:27017" },
    { _id: 1, host: "mongo2:27017" },
    { _id: 2, host: "mongo3:27017" }
  ]
})
```

Primaire / Secondaires

Un primaire accepte les écritures ; les secondaires répliquent en continu (lecture seule par défaut).

Élection automatique

Si le primaire tombe, les secondaires élisent un nouveau primaire — nombre impair de membres recommandé (3 minimum en production).

11 Sharding

Partitionnement horizontal pour dépasser une seule machine

```
sh.enableSharding("boutique")
sh.shardCollection("boutique.commandes", { client_id: 1 })
```

Shard

Un Replica Set contenant une portion des données.

mongos

Routeur qui aiguille chaque requête vers le(s) bon(s) shard(s).

Config servers

Stockent les métadonnées de répartition des données.

Une clé de shard mal choisie concentre les écritures sur un seul shard (hotspot) — préférer une clé à forte cardinalité et bien distribuée.

12 Sécurité

Cinq réflexes indispensables en production

Authentification

Activer --auth, créer des utilisateurs avec des rôles (readWrite, read, dbAdmin…).

Chiffrement en transit

TLS/SSL entre les clients et le serveur.

Isolation réseau

Ne jamais exposer mongod directement sur Internet sans pare-feu.

Moindre privilège

Donner à chaque utilisateur applicatif uniquement les rôles nécessaires.

13 Cas d'utilisation détaillés

Où MongoDB excelle concrètement

Catalogue e-commerce

Produits avec variantes et caractéristiques hétérogènes selon la catégorie — schéma flexible.

CMS / contenus

Articles, pages, commentaires : structure naturellement documentaire.

Logs applicatifs

Index TTL pour purge automatique après X jours.

Données géospatiales

Index 2dsphere, recherche par proximité ($near).

Profils utilisateurs

Préférences variables selon l'utilisateur, sans colonne dédiée par option.

14 MongoDB face aux alternatives

Choisir le bon outil selon le besoin

Critère	MongoDB	PostgreSQL	Redis
Modèle	**Documents (BSON)**	Relationnel (+JSONB)	Clé-valeur en mémoire
Schéma	**Flexible**	Strict	Sans schéma
Cas d'usage typique	**Catalogues, contenus**	Données transactionnelles	Cache, temps réel
Montée en charge	**Sharding natif**	Partitionnement manuel	Cluster natif

MongoDB se positionne entre le relationnel classique et les magasins clé-valeur — souvent utilisé en complément de Redis (source de vérité + cache).

15 Projet pratique de synthèse

Application de gestion de bibliothèque

1	**Modéliser les livres**	titre, auteur, genre, année, exemplaires_disponibles
2	**Modéliser les membres**	nom, email (index unique), date d'inscription
3	**Modéliser les emprunts**	référence livre_id + membre_id (référencement)
4	**Emprunter un livre**	$inc sur exemplaires_disponibles + insertion emprunt
5	**Statistiques d'emprunt**	agrégation $match + $group sur 30 jours
6	**Réservations temporaires**	index TTL, expiration après 24h

Ce projet couvre CRUD, index, référencement et agrégation.

Pour aller plus loin

- **Documentation officielle** — mongodb.com/docs
- **Shell interactif** — mongosh — exploration et administration
- **Interface graphique** — MongoDB Compass
- **Suite de la formation** — TP CRUD pour débutants, puis Replica Set + Sharding

Merci — chaque module peut être approfondi en TP séparé.