# TP - MongoDB : CRUD de Base

## Introduction aux Opérations Fondamentales

**IUT Nord Franche-Comté Durée:** 1h **Objectif:** Maîtriser les opérations CRUD de base dans MongoDB avec le shell `mongosh`

---

## 📋 Contexte

Vous venez d'être embauché dans une startup qui gère une boutique en ligne de livres. Le CTO a décidé d'utiliser MongoDB pour gérer le catalogue de produits, les clients et les commandes.

Votre mission est d'apprendre les opérations de base MongoDB en manipulant directement la base de données via le shell.

---

## 🎯 Objectifs d'Apprentissage

À la fin de ce TP, vous saurez :

- ✅ Créer une base de données MongoDB
- ✅ Créer des collections
- ✅ Insérer des documents (un seul et plusieurs à la fois)
- ✅ Lire des documents avec filtres
- ✅ Mettre à jour des documents (modification, ajout, suppression de champs)
- ✅ Supprimer des documents
- ✅ Utiliser les opérateurs de requête de base

---

# 📦 Prérequis

**Installation de MongoDB :**

```
# Vérifier que MongoDB est installé
mongod --version

# Démarrer MongoDB (dans un terminal)
mongod

# Dans un AUTRE terminal, se connecter au shell
mongosh
```

💡 **Astuce :** Gardez toujours deux terminaux ouverts :

- Terminal 1 : Le serveur MongoDB ( `mongod` )
- Terminal 2 : Le shell MongoDB ( `mongosh` ) pour exécuter vos commandes

---

# 🚀 PARTIE 1 : Créer la Base de Données et les Collections

## Exercice 1.1 : Créer et utiliser une base de données

Dans le shell `mongosh` , tapez les commandes suivantes :

```
// Afficher toutes les bases de données existantes
show dbs

// Créer/basculer vers la base "librairie_en_ligne"
use librairie_en_ligne

// Vérifier quelle base est active
db
// Résultat attendu : librairie_en_ligne
```

❓ **Question 1 :** Exécutez `show dbs`. Est-ce que `librairie_en_ligne` apparaît dans la liste ?

▶ 💡 Réponse

---

### Exercice 1.2 : Créer des collections (implicite)

📝 **Action :**

Nous allons créer trois collections pour notre librairie :

- `livres` : Le catalogue de livres
- `clients` : Les clients inscrits
- `commandes` : Les commandes passées

💡 **Rappel :** Avec MongoDB, pas besoin de "CREATE TABLE". Les collections se créent automatiquement quand on insère des documents.

Vérifions d'abord qu'aucune collection n'existe :

```
// Lister toutes les collections
show collections
// Résultat attendu : (vide)
```

---

## 📚 PARTIE 2 : CREATE - Insérer des Documents

### Exercice 2.1 : Insérer un seul livre (insertOne)

📝 **Action :** Créez votre premier livre dans la collection `livres` :

```
db.livres.insertOne({
  titre: "1984",
  auteur: "George Orwell",
  annee: 1949,
  genre: "Science-fiction dystopique",
  prix: 12.99,
  stock: 25,
  editeur: {
    nom: "Gallimard",
    pays: "France"
  },
  tags: ["dystopie", "totalitarisme", "surveillance"]
})
```

✅ **Résultat attendu :**

```
{
  acknowledged: true,
  insertedId: ObjectId("507f1f77bcf86cd799439011")  // ID généré
automatiquement
}
```

🔍 **Observations importantes :**

1. MongoDB a généré automatiquement un champ `_id` unique
2. Nous avons utilisé un **document imbriqué** pour `editeur` (équivalent d'un objet JSON)
3. Nous avons utilisé un **tableau** pour `tags`
4. Pas besoin de définir un schéma au préalable !

---

❓ **Question 2 :** Vérifiez que la collection `livres` a bien été créée :

```
show collections
// Résultat attendu : livres
```

❓ **Question 3 :** Vérifiez que la base de données apparaît maintenant :

```
show dbs
// Résultat attendu : librairie_en_ligne (avec sa taille)
```

---

## Exercice 2.2 : Insérer plusieurs livres (insertMany)

📝 **Action :** Ajoutez 4 livres d'un coup avec `insertMany()` :

```
db.livres.insertMany([
  {
    titre: "Le Seigneur des Anneaux",
    auteur: "J.R.R. Tolkien",
    annee: 1954,
    genre: "Fantasy",
    prix: 29.99,
    stock: 15,
    editeur: {
      nom: "Christian Bourgois",
      pays: "France"
    },
    tags: ["fantasy", "aventure", "épique"]
  },
  {
    titre: "Harry Potter à l'école des sorciers",
    auteur: "J.K. Rowling",
    annee: 1997,
    genre: "Fantasy jeunesse",
    prix: 18.50,
    stock: 50,
    editeur: {
      nom: "Gallimard Jeunesse",
      pays: "France"
    },
    tags: ["magie", "aventure", "école"]
  },
  {
    titre: "Le Petit Prince",
    auteur: "Antoine de Saint-Exupéry",
    annee: 1943,
    genre: "Conte philosophique",
    prix: 8.90,
    stock: 100,
    editeur: {
      nom: "Gallimard",
      pays: "France"
    },
    tags: ["philosophie", "enfance", "voyage"]
  },
  {
    titre: "Les Misérables",
    auteur: "Victor Hugo",
    annee: 1862,
    genre: "Roman historique",
```

```
    prix: 15.00,
    stock: 8,
    editeur: {
      nom: "Le Livre de Poche",
      pays: "France"
    },
    tags: ["classique", "histoire", "social"]
  }
])
```

✅ **Résultat attendu :**

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("..."),
    '1': ObjectId("..."),
    '2': ObjectId("..."),
    '3': ObjectId("...")
  }
}
```

---

## Exercice 2.3 : Insérer des clients

📝 **À vous de jouer !** Créez la collection `clients` en insérant 3 clients :

```
db.clients.insertMany([
  {
    nom: "Dupont",
    prenom: "Alice",
    email: "alice.dupont@mail.com",
    age: 28,
    ville: "Paris",
    dateInscription: new Date("2024-01-15"),
    achatsTotal: 0
  },
  {
    nom: "Martin",
    prenom: "Bob",
    email: "bob.martin@mail.com",
    age: 35,
    ville: "Lyon",
    dateInscription: new Date("2024-02-20"),
    achatsTotal: 0
  },
  {
    nom: "Dubois",
    prenom: "Charlie",
    email: "charlie.dubois@mail.com",
    age: 42,
    ville: "Marseille",
    dateInscription: new Date("2024-03-10"),
    achatsTotal: 0
  }
])
```

---

# 🔍 PARTIE 3 : READ - Lire des Documents

## Exercice 3.1 : Lire TOUS les documents d'une collection

### 📝 Action :

```
// Lire tous les livres
db.livres.find()

// Version plus lisible (pretty print)
db.livres.find().pretty()
```

💡 **Astuce :** Si vous avez beaucoup de documents, MongoDB affiche seulement les 20 premiers. Tapez `it` (iterate) pour voir les suivants.

---

## Exercice 3.2 : Lire UN SEUL document (findOne)

📝 **Action :**

```
// Récupérer un livre au hasard
db.livres.findOne()

// Récupérer le livre "1984"
db.livres.findOne({ titre: "1984" })

// Récupérer un livre par auteur
db.livres.findOne({ auteur: "J.K. Rowling" })
```

❓ **Question 4 :** Que se passe-t-il si on cherche un livre qui n'existe pas ?

```
db.livres.findOne({ titre: "Livre Inexistant" })
// Résultat : null
```

---

## Exercice 3.3 : Filtrer avec des conditions (opérateurs de comparaison)

📝 **Action :** Trouvez tous les livres qui coûtent **moins de 15€** :

```
db.livres.find({ prix: { $lt: 15 } })
```

🔧 **Opérateurs disponibles :**

| Opérateur | Signification | Exemple |
|---|---|---|
| `$lt` | Less Than (<) | `{ prix: { $lt: 15 } }` |
| `$lte` | Less Than or Equal (<=) | `{ prix: { $lte: 15 } }` |
| `$gt` | Greater Than (>) | `{ annee: { $gt: 2000 } }` |
| `$gte` | Greater Than or Equal (>=) | `{ stock: { $gte: 20 } }` |
| `$ne` | Not Equal (!=) | `{ genre: { $ne: "Fantasy" } }` |
| `$in` | Dans une liste | `{ genre: { $in: ["Fantasy", "Science-fiction dystopique"] } }` |

---

❓ **Question 5 - Exercices pratiques :** Trouvez :

```
// A. Tous les livres publiés APRÈS 1950
db.livres.find({ annee: { $gt: 1950 } })

// B. Tous les livres avec un stock SUPÉRIEUR ou ÉGAL à 20
db.livres.find({ stock: { $gte: 20 } })

// C. Tous les livres qui ne sont PAS de genre "Fantasy"
db.livres.find({ genre: { $ne: "Fantasy" } })

// D. Tous les livres de genre "Fantasy" OU "Science-fiction dystopique"
db.livres.find({ genre: { $in: ["Fantasy", "Science-fiction dystopique"] } })
```

---

## Exercice 3.4 : Filtrer avec plusieurs conditions (opérateurs logiques)

📝 **Action :** Trouvez tous les livres qui :

- Coûtent **moins de 20€** ET
- Ont un stock **supérieur à 10**

```
// Méthode 1 : Implicite (ET logique par défaut)
db.livres.find({
  prix: { $lt: 20 },
  stock: { $gt: 10 }
})

// Méthode 2 : Explicite avec $and
db.livres.find({
  $and: [
    { prix: { $lt: 20 } },
    { stock: { $gt: 10 } }
  ]
})
```

💡 **Opérateur $or :** Trouvez tous les livres écrits par “Victor Hugo” **OU** publiés après 1990 :

```
db.livres.find({
  $or: [
    { auteur: "Victor Hugo" },
    { annee: { $gt: 1990 } }
  ]
})
```

---

## Exercice 3.5 : Rechercher dans un champ imbriqué

📝 **Action :** Trouvez tous les livres publiés par “Gallimard” :

```
db.livres.find({ "editeur.nom": "Gallimard" })
```

⚠ **Important :** Utilisez la **notation pointée** entre guillemets : `"editeur.nom"`

---

## Exercice 3.6 : Rechercher dans un tableau

📝 **Action :** Trouvez tous les livres qui ont le tag “aventure” :

```
db.livres.find({ tags: "aventure" })
```

💡 **MongoDB est intelligent !** Même si `tags` est un tableau, vous pouvez chercher une valeur directement.

**Bonus :** Trouvez tous les livres qui ont À LA FOIS les tags “fantasy” ET “épique” :

```
db.livres.find({ tags: { $all: ["fantasy", "épique"] } })
```

---

## Exercice 3.7 : Projection (choisir les champs à retourner)

Par défaut, `find()` retourne TOUS les champs. Parfois, vous voulez seulement certains champs.

📝 **Action :** Retournez seulement le `titre` et le `prix` des livres :

```
db.livres.find(
  {},                          // Filtre vide = tous les documents
  { titre: 1, prix: 1 }        // Projection : 1 = inclure, 0 = exclure
)
```

✅ **Résultat :**

```
[
  { _id: ObjectId("..."), titre: "1984", prix: 12.99 },
  { _id: ObjectId("..."), titre: "Le Seigneur des Anneaux", prix: 29.99 },
  // ...
]
```

💡 **Astuce :** Le champ `_id` est TOUJOURS retourné par défaut. Pour l’exclure :

```
db.livres.find(
  {},
  { _id: 0, titre: 1, prix: 1 }
)
```

---

# ✏️ PARTIE 4 : UPDATE - Mettre à Jour des Documents

## Exercice 4.1 : Modifier un champ avec $set

📝 **Scénario :** Le livre “1984” est en promotion ! Passez son prix à **9.99€**.

```
db.livres.updateOne(
  { titre: "1984" },                // Filtre : quel document modifier ?
  { $set: { prix: 9.99 } }          // Modification : nouveau prix
)
```

### ✅ Résultat :

```
{
  acknowledged: true,
  matchedCount: 1,      // 1 document trouvé
  modifiedCount: 1      // 1 document modifié
}
```

### 🔍 Vérification :

```
db.livres.findOne({ titre: "1984" })
// prix devrait maintenant être 9.99
```

---

## Exercice 4.2 : Ajouter un nouveau champ avec $set

📝 **Scénario :** Ajoutez un champ `enPromotion: true` au livre “1984”.

```
db.livres.updateOne(
  { titre: "1984" },
  { $set: { enPromotion: true } }
)
```

💡 **Observation :** `$set` peut à la fois **modifier** un champ existant ET **créer** un nouveau champ !

---

## Exercice 4.3 : Supprimer un champ avec $unset

📝 **Scénario :** Finalement, la promotion est terminée. Supprimez le champ `enPromotion`.

```
db.livres.updateOne(
  { titre: "1984" },
  { $unset: { enPromotion: "" } }     // Valeur vide, le champ sera supprimé
)
```

🔍 **Vérification :**

```
db.livres.findOne({ titre: "1984" })
// Le champ enPromotion n'existe plus
```

---

## Exercice 4.4 : Incrémenter/Décrémenter avec $inc

📝 **Scénario :** Un client achète 3 exemplaires du “Petit Prince”. Diminuez le stock de 3.

```
db.livres.updateOne(
  { titre: "Le Petit Prince" },
  { $inc: { stock: -3 } }              // -3 pour décrémenter
)
```

🔍 **Vérification :**

```
db.livres.findOne({ titre: "Le Petit Prince" }, { titre: 1, stock: 1 })
// stock devrait être 97 (était 100, maintenant 100 - 3)
```

💡 **Bonus :** Pour INCRÉMENTER, utilisez une valeur positive :

```
db.livres.updateOne(
  { titre: "Le Petit Prince" },
  { $inc: { stock: 5 } }              // +5 pour augmenter le stock
)
```

---

## Exercice 4.5 : Ajouter un élément à un tableau avec $push

📝 **Scénario :** Ajoutez le tag “bestseller” au livre “Harry Potter”.

```
db.livres.updateOne(
  { titre: "Harry Potter à l'école des sorciers" },
  { $push: { tags: "bestseller" } }
)
```

🔍 **Vérification :**

```
db.livres.findOne({ titre: "Harry Potter à l'école des sorciers" }, { tags:
1 })
// tags devrait maintenant contenir "bestseller"
```

---

## Exercice 4.6 : Retirer un élément d’un tableau avec $pull

📝 **Scénario :** Retirez le tag “école” du livre “Harry Potter”.

```
db.livres.updateOne(
  { titre: "Harry Potter à l'école des sorciers" },
  { $pull: { tags: "école" } }
)
```

🔍 **Vérification :**

```
db.livres.findOne({ titre: "Harry Potter à l'école des sorciers" }, { tags: 1 })
// "école" ne devrait plus être dans tags
```

---

## Exercice 4.7 : Mettre à jour PLUSIEURS documents (updateMany)

📝 **Scénario :** Tous les livres de l'éditeur "Gallimard" passent en promotion : réduisez leur prix de 10%.

```
// 1. Trouvez d'abord tous les livres Gallimard pour voir les prix actuels
db.livres.find({ "editeur.nom": "Gallimard" }, { titre: 1, prix: 1 })

// 2. Réduisez tous les prix de 10% (multipliez par 0.9)
db.livres.updateMany(
  { "editeur.nom": "Gallimard" },
  { $mul: { prix: 0.9 } }              // $mul = multiply
)
```

✅ **Résultat :**

```
{
  acknowledged: true,
  matchedCount: 3,      // 3 livres Gallimard trouvés
  modifiedCount: 3      // 3 livres modifiés
}
```

---

### Exercice 4.8 : Upsert (Update ou Insert si n'existe pas)

📝 **Scénario :** Vous recevez une nouvelle commande pour le livre "Dune". S'il existe, augmentez son stock de 10. S'il n'existe pas, créez-le.

```
db.livres.updateOne(
  { titre: "Dune" },
  {
    $set: {
      titre: "Dune",
      auteur: "Frank Herbert",
      annee: 1965,
      genre: "Science-fiction",
      prix: 22.50,
      editeur: { nom: "Robert Laffont", pays: "France" },
      tags: ["SF", "désert", "politique"]
    },
    $inc: { stock: 10 }
  },
  { upsert: true }                    // ⭐ Option magique !
)
```

💡 **Explication :**

- Si "Dune" existe déjà : incrémente le stock de 10
- Si "Dune" n'existe PAS : crée le document avec stock = 10

🔍 **Vérification :**

```
db.livres.findOne({ titre: "Dune" })
```

---

## 🗑️ PARTIE 5 : DELETE - Supprimer des Documents

### Exercice 5.1 : Supprimer UN document (deleteOne)

📝 **Scénario :** Le livre "Dune" ne se vend pas bien. Supprimez-le du catalogue.

```
db.livres.deleteOne({ titre: "Dune" })
```

✅ **Résultat :**

```
{
  acknowledged: true,
  deletedCount: 1        // 1 document supprimé
}
```

🔍 **Vérification :**

```
db.livres.findOne({ titre: "Dune" })
// Résultat : null (n'existe plus)
```

---

## Exercice 5.2 : Supprimer PLUSIEURS documents (deleteMany)

📝 **Scénario :** Supprimez tous les livres qui ont un stock inférieur à 10 (rupture de stock).

```
// 1. Voyez d'abord quels livres seront supprimés
db.livres.find({ stock: { $lt: 10 } }, { titre: 1, stock: 1 })

// 2. Supprimez-les
db.livres.deleteMany({ stock: { $lt: 10 } })
```

✅ **Résultat :**

```
{
  acknowledged: true,
  deletedCount: 1        // "Les Misérables" avait un stock de 8
}
```

---

### Exercice 5.3 : Supprimer TOUS les documents d'une collection

#### ⚠️ ATTENTION : Opération dangereuse !

```
// Supprimer tous les clients (pour repartir de zéro)
db.clients.deleteMany({})
```

#### 💡 Différence avec drop() :

```
// deleteMany({}) : Supprime tous les documents MAIS garde la collection
db.clients.deleteMany({})

// drop() : Supprime la collection entière (structure + données)
db.clients.drop()
```

---

## 📊 PARTIE 6 : Exercices Pratiques Complets

### Exercice 6.1 : Gestion d'une commande client

#### 📝 Scénario complet :

Alice Dupont commande 2 exemplaires de "Harry Potter à l'école des sorciers".

#### Étapes à réaliser :

```
// 1. Vérifier le stock disponible
db.livres.findOne(
  { titre: "Harry Potter à l'école des sorciers" },
  { titre: 1, stock: 1, prix: 1 }
)

// 2. Décrémenter le stock de 2
db.livres.updateOne(
  { titre: "Harry Potter à l'école des sorciers" },
  { $inc: { stock: -2 } }
)

// 3. Mettre à jour le montant total des achats d'Alice
// (supposons que le livre coûte 18.50€, donc 2 × 18.50 = 37€)
db.clients.updateOne(
  { email: "alice.dupont@mail.com" },
  { $inc: { achatsTotal: 37.00 } }
)

// 4. Créer la commande
db.commandes.insertOne({
  clientEmail: "alice.dupont@mail.com",
  livres: [
    {
      titre: "Harry Potter à l'école des sorciers",
      quantite: 2,
      prixUnitaire: 18.50
    }
  ],
  montantTotal: 37.00,
  dateCommande: new Date(),
  statut: "confirmée"
})
```

---

## Exercice 6.2 : Statistiques du catalogue

📝 **À vous de jouer !** Répondez à ces questions avec des requêtes MongoDB :

**Question A :** Combien y a-t-il de livres dans le catalogue ?

```
db.livres.countDocuments()
```

**Question B :** Quel est le livre le plus cher ?

```
db.livres.find().sort({ prix: -1 }).limit(1)
```

**Question C :** Quel est le livre le moins cher ?

```
db.livres.find().sort({ prix: 1 }).limit(1)
```

**Question D :** Combien de livres de genre “Fantasy” ?

```
db.livres.countDocuments({ genre: "Fantasy" })
// OU
db.livres.countDocuments({ genre: /Fantasy/i })  // Regex insensible à la casse
```

**Question E :** Listez tous les livres par ordre de prix décroissant :

```
db.livres.find({}, { titre: 1, prix: 1 }).sort({ prix: -1 })
```

---

# 🧹 Nettoyage et Gestion

## Supprimer une collection complète

```
db.commandes.drop()
// Résultat : true
```

### Supprimer la base de données entière

```
db.dropDatabase()
// Résultat : { ok: 1, dropped: "librairie_en_ligne" }
```

---

# 📋 Récapitulatif des Commandes

## Commandes de Gestion

| Commande | Description |
|---|---|
| `show dbs` | Liste toutes les bases de données |
| `use nom_db` | Crée/bascule vers une base |
| `db` | Affiche la base active |
| `show collections` | Liste toutes les collections |
| `db.collection.drop()` | Supprime une collection |
| `db.dropDatabase()` | Supprime la base active |

## Commandes CRUD

| Opération | Commande | Exemple |
|---|---|---|
| **CREATE** | `insertOne()` | `db.livres.insertOne({ titre: "..." })` |
| | `insertMany()` | `db.livres.insertMany([{...}, {...}])` |
| **READ** | `find()` | `db.livres.find({ prix: { $lt: 20 } })` |
| | `findOne()` | `db.livres.findOne({ titre: "1984" })` |
| | `countDocuments()` | `db.livres.countDocuments()` |
| **UPDATE** | `updateOne()` | `db.livres.updateOne({...}, { $set: {...} })` |
| | `updateMany()` | `db.livres.updateMany({...}, { $inc: {...} })` |
| **DELETE** | `deleteOne()` | `db.livres.deleteOne({ titre: "..." })` |
| | `deleteMany()` | `db.livres.deleteMany({ stock: { $lt: 5 } })` |

## Opérateurs de Mise à Jour

| Opérateur | Usage | Exemple |
|---|---|---|
| `$set` | Modifier/Ajouter un champ | `{ $set: { prix: 9.99 } }` |
| `$unset` | Supprimer un champ | `{ $unset: { promo: "" } }` |
| `$inc` | Incrémenter/Décrémenter | `{ $inc: { stock: -5 } }` |
| `$mul` | Multiplier | `{ $mul: { prix: 0.9 } }` |
| `$push` | Ajouter à un tableau | `{ $push: { tags: "nouveau" } }` |
| `$pull` | Retirer d’un tableau | `{ $pull: { tags: "ancien" } }` |

### Opérateurs de Requête

| Opérateur | Signification | Exemple |
|---|---|---|
| `$lt` | < | `{ prix: { $lt: 15 } }` |
| `$lte` | <= | `{ prix: { $lte: 15 } }` |
| `$gt` | > | `{ annee: { $gt: 2000 } }` |
| `$gte` | >= | `{ stock: { $gte: 20 } }` |
| `$ne` | != | `{ genre: { $ne: "Fantasy" } }` |
| `$in` | Dans une liste | `{ genre: { $in: ["Fantasy", "SF"] } }` |
| `$or` | OU logique | `{ $or: [{...}, {...}] }` |
| `$and` | ET logique | `{ $and: [{...}, {...}] }` |
| `$all` | Tous les éléments | `{ tags: { $all: ["fantasy", "épique"] } }` |

---

## 🎯 Livrables Attendus

À la fin du TP, vous devez avoir :

- ✅ Une base de données `librairie_en_ligne` créée
- ✅ 3 collections : `livres`, `clients`, `commandes`
- ✅ Au moins 5 livres dans la collection `livres`
- ✅ Au moins 3 clients dans la collection `clients`
- ✅ Au moins 1 commande enregistrée
- ✅ Avoir pratiqué toutes les opérations CRUD
- ✅ Avoir utilisé au moins 5 opérateurs différents ($set$,inc, $push$,pull, $lt$,gt, etc.)

---

## ❓ Questions de Réflexion

1. **Quelle est la principale différence entre SQL et MongoDB ?**
    - SQL : Schéma rigide, tables, relations avec clés étrangères
    - MongoDB : Schéma flexible, collections, documents imbriqués
2. **Quand utiliser `updateOne()` vs `updateMany()` ?**
    - `updateOne()` : Modifier UN SEUL document (même si plusieurs correspondent)
    - `updateMany()` : Modifier TOUS les documents qui correspondent
3. **Pourquoi utiliser `$inc` au lieu de récupérer, calculer, puis mettre à jour ?**
    - `$inc` est **atomique** : évite les problèmes de concurrence (race conditions)
4. **Quelle est la différence entre `deleteMany({})` et `drop()` ?**
    - `deleteMany({})` : Supprime tous les documents, mais garde la collection
    - `drop()` : Supprime la collection entière (structure + données)

---

**Bon courage ! 🎉**