

Référence rapide MongoDB

CRUD, requêtes, agrégation, index, conception de schéma

Connexion

Chaîne de connexion

```
mongosh "mongodb://localhost:27017"
mongosh "mongodb://user:pass@host:27017/mydb"
mongosh "mongodb+srv://user:pass@cluster.mongodb.net/mydb"
```

Connexion via driver (Node.js)

```
const { MongoClient } = require('mongodb');
const client = new MongoClient(uri);
await client.connect();
const db = client.db('mydb');
```

Bases de données et collections

Opérations sur les bases de données

```
show dbs
use mydb
db.dropDatabase()
```

Opérations sur les collections

```
db.createCollection("users")
show collections
db.users.drop()
```

Collection plafonnée

```
db.createCollection("logs", {
  capped: true, size: 10485760, max: 5000
})
```

Opérations CRUD

Insertion

```
db.users.insertOne({ name: "Alice", age: 30 })
db.users.insertMany([
  { name: "Bob", age: 25 },
  { name: "Carol", age: 28 }
])
```

Recherche

```
db.users.findOne({ name: "Alice" })
db.users.find({ age: { $gte: 25 } })
db.users.find({}, { name: 1, _id: 0 })
db.users.find().sort({ age: -1 }).limit(10)
```

Mise à jour

```
db.users.updateOne(
  { name: "Alice" },
  { $set: { age: 31, city: "Boston" } }
)
db.users.updateMany(
  { active: false },
  { $set: { archived: true } }
)
```

Suppression

```
db.users.deleteOne({ name: "Alice" })
db.users.deleteMany({ active: false })
```

Remplacement et upsert

```
db.users.replaceOne(
  { name: "Alice" },
  { name: "Alice", age: 32, city: "NYC" }
)
db.users.updateOne(
  { email: "a@ex.com" },
  { $set: { name: "Alice" } },
  { upsert: true }
)
```

Opérateurs de requête

Comparaison

\$eq / \$ne	Égal / différent
\$gt / \$gte	Supérieur à / supérieur ou égal
\$lt / \$lte	Inférieur à / inférieur ou égal
\$in / \$nin	Dans le tableau / pas dans le tableau

Logique

\$and	Toutes les conditions doivent correspondre
\$or	Au moins une condition correspond
\$not	Nie une condition
\$exists	Le champ existe (true/false)
\$regex	Correspondance par expression régulière

Opérateurs de mise à jour

\$set	Définir la valeur d'un champ
\$unset	Supprimer un champ
\$inc	Incrémenter une valeur numérique
\$push / \$pull	Ajouter / supprimer un élément de tableau
\$addToSet	Ajouter au tableau si absent
\$rename	Renommer un champ

Agrégation

Étapes du pipeline

\$match	Filter les documents (comme WHERE)
\$group	Grouper et agréger
\$project	Remodeler les documents (comme SELECT)
\$sort	Trier les résultats
\$limit / \$skip	Pagination
\$lookup	Jointure externe gauche avec une autre collection
\$unwind	Décomposer un tableau en documents

Exemple d'agrégation

```
db.orders.aggregate([
  { $match: { status: "completed" } },
  { $group: {
    _id: "$customer_id",
    total: { $sum: "$amount" },
    count: { $sum: 1 }
  }},
  { $sort: { total: -1 } },
  { $limit: 10 }
])
```

Index

Créer et supprimer

```
db.users.createIndex({ email: 1 }, { unique: true })
db.users.createIndex({ name: 1, age: -1 })
db.users.createIndex({ location: "2dsphere" })
db.users.dropIndex("email_1")
```

Types d'index

Single field	Index sur un champ ({ name: 1 })
Compound	Plusieurs champs ({ a: 1, b: -1 })
Text	Recherche plein texte ({ field: 'text' })
2dsphere	Requêtes géospatiales
TTL	Expiration automatique des documents après un délai

Informations sur les index

```
db.users.getIndexes()
db.users.find({ name: "Alice" }).explain()
```

Conception de schéma

Imbrication vs référencement

Embed	1:1 ou 1:peu, données lues ensemble
Reference	1:plusieurs, données accédées indépendamment
Embed	Sous-document dépasse rarement 16 Mo
Reference	Relations plusieurs-à-plusieurs

Validation de schéma

```
db.createCollection("users", {
  validator: { $jsonSchema: {
    bsonType: "object",
    required: ["name", "email"],
    properties: {
      name: { bsonType: "string" },
      email: { bsonType: "string" }
    }
  }
})
```

Réplication

Concepts de replica set

Primary	Reçoit toutes les écritures
Secondary	Réplique depuis le primaire, peut servir les lectures
Arbiter	Vote lors des élections, ne contient pas de données

Commandes de replica set

```
rs.initiate()
rs.add("mongo2:27017")
rs.addArb("mongo3:27017")
rs.status()
rs.conf()
```

Modèles courants

Transactions

```
const session = client.startSession();
session.startTransaction();
await db.collection("accounts").updateOne(
  { _id: 1 }, { $inc: { bal: -100 } }, { session });
await db.collection("accounts").updateOne(
  { _id: 2 }, { $inc: { bal: 100 } }, { session });
await session.commitTransaction();
```

Écriture en masse

```
db.users.bulkWrite([
  { insertOne: { document: { name: "Dan" } } },
  { updateOne: {
    filter: { name: "Alice" },
    update: { $set: { age: 31 } }
  }},
  { deleteOne: { filter: { name: "old" } } }
])
```

Référence rapide MongoDB

Change Streams

```
const stream = db.collection("orders")
  .watch([{$match: { "fullDocument.status": "new" } }]);
stream.on("change", (change) => {
  console.log(change.fullDocument);
});
```

Commandes mongosh

Helpers du shell

show dbs	Lister les bases de données
show collections	Lister les collections de la base courante
db.stats()	Statistiques de la base de données
db.collection.stats()	Statistiques de la collection
db.collection.countDocuments({})	Compter les documents
db.collection.distinct('field')	Valeurs distinctes d'un champ

Export et import

```
mongoexport --db=mydb --collection=users \
  --out=users.json
mongoimport --db=mydb --collection=users \
  --file=users.json
mongodump --db=mydb --out=/backup/
mongorestore --db=mydb /backup/mydb/
```