

REFERENCIA RÁPIDA DE MONGODB

CRUD, consultas, agregación, índices, diseño de esquema

Conexión

Cadena de conexión

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

Conexión con driver (Node.js)

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

Bases de datos y colecciones

Operaciones de base de datos

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

Operaciones de colección

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

Colección con límite (Capped)

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

Operaciones CRUD

Insertar

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

Buscar

```
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)
```

Actualizar

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

Eliminar

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

Reemplazar y upsert

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

Operadores de consulta

Comparación

\$eq / \$ne Igual / no igual
\$gt / \$gte Mayor que / mayor o igual
\$lt / \$lte Menor que / menor o igual
\$in / \$nin En el array / no en el array

Lógicos

\$and Todas las condiciones deben coincidir
\$or Al menos una condición coincide
\$not Niega una condición
\$exists El campo existe (true/false)
\$regex Coincidencia con expresión regular

Operadores de actualización

\$set Establecer valor de campo
\$unset Eliminar campo
\$inc Incrementar valor numérico
\$push / \$pull Agregar / eliminar elemento del array
\$addToSet Agregar al array si no está presente
\$rename Renombrar un campo

Agregación

Etapas del pipeline

\$match Filtrar documentos (como WHERE)
\$group Agrupar y agregar
\$project Reformar documentos (como SELECT)
\$sort Ordenar resultados
\$limit / \$skip Paginación
\$lookup Left outer join con otra colección
\$unwind Deconstruir array en documentos

Ejemplo de agregación

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

Índices

Crear y eliminar

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

Tipos de índice

Single field Índice en un campo ({ name: 1 })
Compound Múltiples campos ({ a: 1, b: -1 })
Text Búsqueda de texto completo ({ field: 'text' })
2dsphere Consultas geoespaciales
TTL Expirar documentos automáticamente tras un tiempo

Información de índices

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

Diseño de esquema

Embebido vs referencia

Embed 1:1 o 1:pocos, datos leídos juntos
Reference 1:muchos, datos accedidos independientemente
Embed Sub-documento raramente supera 16 MB
Reference Relaciones muchos a muchos

Validación de esquema

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

Replicación

Conceptos del conjunto de réplicas

Primary Recibe todas las escrituras
Secondary Replica desde el primario, puede servir lecturas
Arbiter Vota en elecciones, no almacena datos

Comandos del conjunto de réplicas

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

Patrones comunes

Transacciones

```
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();
```

Escritura masiva

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

Change Streams

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

Comandos de mongosh

Helpers del shell

show dbs Listar bases de datos
show collections Listar colecciones en la BD actual
db.stats() Estadísticas de la base de datos
db.collection.stats() Estadísticas de la colección
db.collection.countDocuments({}) Contar documentos
db.collection.distinct('field') Valores distintos para un campo

Exportar e importar

```
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/
```