

MongoDB 快速参考

CRUD、查询、聚合、索引、Schema 设计

连接

连接字符串

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

驱动连接 (Node.js)

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

数据库与集合

数据库操作

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

集合操作

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

固定集合

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

CRUD 操作

插入

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

查询

```
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)
```

更新

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

删除

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

替换与 Upsert

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

查询运算符

比较

\$eq / \$ne	等于 / 不等于
\$gt / \$gte	大于 / 大于等于
\$lt / \$lte	小于 / 小于等于
\$in / \$nin	在数组中 / 不在数组中

逻辑

\$and	所有条件都必须匹配
\$or	至少一个条件匹配
\$not	取反条件
\$exists	字段是否存在 (true/false)
\$regex	正则表达式匹配

更新运算符

\$set	设置字段值
\$unset	移除字段
\$inc	数值递增
\$push / \$pull	添加 / 移除数组元素
\$addToSet	不存在时添加到数组
\$rename	重命名字段

聚合

管道阶段

\$match	过滤文档 (类似 WHERE)
\$group	分组并聚合
\$project	重构文档 (类似 SELECT)
\$sort	排序结果
\$limit / \$skip	分页
\$lookup	与另一集合左外连接
\$unwind	将数组拆解为多个文档

聚合示例

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

索引

创建与删除

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

索引类型

Single field	单字段索引 ({name:1})
Compound	多字段索引 ({a:1,b:-1})
Text	全文检索 ({field:'text'})
2dsphere	地理空间查询
TTL	到期后自动删除文档

索引信息

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

Schema 设计

嵌入 vs 引用

Embed	1:1 或 1:少, 数据一起读取
Reference	1:多, 数据独立访问
Embed	子文档很少超过 16 MB
Reference	多对多关系

Schema 验证

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

副本集

副本集概念

Primary	接收所有写入
Secondary	从主节点复制, 可提供读服务
Arbiter	参与选举, 不存储数据

副本集命令

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

常见模式

事务

```
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();
```

批量写入

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

MongoDB 快速参考

变更流

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

mongosh 命令

Shell 辅助命令

show dbs	列出数据库
show collections	列出当前库的集合
db.stats()	数据库统计信息
db.collection.stats()	集合统计信息
db.collection.countDocuments({})	文档计数
db.collection.distinct('field')	字段的唯一值

导出与导入

```
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/
```