MongoDB クイックリファレンス

CRUD、クエリ、集計、インデックス、スキーマ設計

接続

接続文字列

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

ドライバー接続（Node.js）

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

データベースとコレクション

データベース操作

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

コレクション操作

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

上限付きコレクション

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

CRUD 操作

挿入

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

検索

```
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 } }
)
```

削除

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

置換と upsert

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

クエリ演算子

比較

$eq / $ne	等価 / 非等価
$gt / $gte	より大きい / 以上
$lt / $lte	より小さい / 以下
$in / $nin	配列内 / 配列外

論理

$and	すべての条件が一致
$or	少なくとも 1 つの条件が一致
$not	条件を否定
$exists	フィールドが存在する（true/false）
$regex	正規表現でマッチ

更新演算子

$set	フィールド値を設定
$unset	フィールドを削除
$inc	数値をインクリメント
$push / $pull	配列要素を追加 / 削除
$addToSet	存在しない場合のみ配列に追加
$rename	フィールドの名前を変更

集計

パイプラインステージ

$match	ドキュメントをフィルター（WHERE に相当）
$group	グループ化と集計
$project	ドキュメントを整形（SELECT に相当）
$sort	結果をソート
$limit / $skip	ページネーション
$lookup	別コレクションへの左外部結合
$unwind	配列をドキュメントに展開

集計の例

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

インデックス

作成と削除

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

インデックス種類

Single field	単一フィールドのインデックス（{ name: 1 }）
Compound	複数フィールド（{ a: 1, b: -1 }）
Text	全文検索（{ field: 'text' }）
2dsphere	地理空間クエリ
TTL	一定時間後にドキュメントを自動期限切れ

インデックス情報

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

スキーマ設計

埋め込み vs 参照

Embed	1:1 または 1:少数、一緒に読まれるデータ
Reference	1:多数、独立してアクセスされるデータ
Embed	サブドキュメントが 16 MB を超えることがほぼない
Reference	多対多の関係

スキーマバリデーション

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

レプリケーション

レプリカセットの概念

Primary	すべての書き込みを受け付ける
Secondary	プライマリからレプリケート、読み取りを提供できる
Arbiter	選挙で投票するが、データを保持しない

レプリカセットコマンド

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

よく使うパターン

トランザクション

```
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();
```

バルク書き込み

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

MongoDB クイックリファレンス

チェンジストリーム

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

mongosh コマンド

シェルヘルパー

show dbs	データベースを一覧表示
show collections	現在の DB のコレクションを一覧表示
db.stats()	データベース統計
db.collection.stats()	コレクション統計
db.collection.countDocuments({})	ドキュメント数をカウント
db.collection.distinct('field')	フィールドの一意な値

エクスポートとインポート

```
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/
```