MONGODB 빠른 참조

CRUD, 쿼리, 집계, 인덱스, 스키마 설계

연결

연결 문자열

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

드라이버 연결 (Node.js)

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

데이터베이스 및 컬렉션

데이터베이스 작업

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

컬렉션 작업

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

캡 컬렉션

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

CRUD 작업

삽입

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

조회

```
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)
```

업데이트

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

삭제

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

교체 및 Upsert

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

쿼리 연산자

비교

`$eq` / `$ne`	같음 / 같지 않음
`$gt` / `$gte`	보다 클 / 크거나 같음
`$lt` / `$lte`	보다 작음 / 작거나 같음
`$in` / `$nin`	배열 내 존재 / 배열 내 없음

논리

`$and`	모든 조건이 매칭되어야 함
`$or`	하나 이상의 조건이 매칭
`$not`	조건 부정
`$exists`	필드 존재 여부 (true/false)
`$regex`	정규식 매칭

업데이트 연산자

`$set`	필드 값 설정
`$unset`	필드 제거
`$inc`	숫자 값 증가
`$push` / `$pull`	배열 요소 추가 / 제거
`$addToSet`	없는 경우에만 배열에 추가
`$rename`	필드 이름 변경

집계

파이프라인 단계

`$match`	문서 필터링 (WHERE와 유사)
`$group`	그룹화 및 집계
`$project`	문서 형태 변환 (SELECT와 유사)
`$sort`	결과 정렬
`$limit` / `$skip`	페이지네이션
`$lookup`	다른 컬렉션과 좌측 외부 조인
`$unwind`	배열을 문서로 분해

집계 예시

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

인덱스

생성 및 삭제

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

인덱스 유형

`Single field`	단일 필드 인덱스 ({ name: 1 })
`Compound`	복합 필드 ({ a: 1, b: -1 })
`Text`	전문 검색 ({ field: 'text' })
`2dsphere`	지리공간 쿼리
`TTL`	시간 후 문서 자동 만료

인덱스 정보

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

스키마 설계

임베딩 vs 참조

`임베딩`	1:1 또는 1:소수, 데이터가 함께 읽힘
`참조`	1:다수, 데이터가 독립적으로 접근됨
`임베딩`	서브문서가 16 MB를 초과하는 경우가 드물 때
`참조`	다대다 관계

스키마 유효성 검사

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

복제

레플리카 셋 개념

`Primary`	모든 쓰기를 수신
`Secondary`	프라이머리에서 복제, 읽기 제공 가능
`Arbiter`	선거에서 투표, 데이터 미보유

레플리카 셋 명령어

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

주요 패턴

트랜잭션

```
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();
```

벌크 쓰기

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

Change Stream

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

mongosh 명령어

셸 헬퍼

`show dbs`	데이터베이스 목록
`show collections`	현재 DB의 컬렉션 목록
`db.stats()`	데이터베이스 통계
`db.collection.stats()`	컬렉션 통계
`db.collection.countDocuments({})`	문서 수 세기
`db.collection.distinct('field')`	필드의 고유값

내보내기 및 가져오기

```
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/
```