MONGODB RIFERIMENTO RAPIDO

CRUD, query, aggregazione, indici, progettazione schema

Connessione

Stringa di Connessione

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

Connessione con Driver (Node.js)

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

Database e Collection

Operazioni sui Database

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

Operazioni sulle Collection

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

Collection Capped

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

Operazioni CRUD

Inserimento

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

Ricerca

```
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)
```

Aggiornamento

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

Eliminazione

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

Replace e Upsert

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

Operatori di Query

Confronto

`$eq` / `$ne`	Uguale / diverso
`$gt` / `$gte`	Maggiore di / maggiore o uguale
`$lt` / `$lte`	Minore di / minore o uguale
`$in` / `$nin`	Presente nell'array / non presente

Logici

`$and`	Tutte le condizioni devono essere soddisfatte
`$or`	Almeno una condizione soddisfatta
`$not`	Nega una condizione
`$exists`	Il campo esiste (true/false)
`$regex`	Corrispondenza con espressione regolare

Operatori di Aggiornamento

`$set`	Imposta il valore di un campo
`$unset`	Rimuove un campo
`$inc`	Incrementa un valore numerico
`$push` / `$pull`	Aggiunge / rimuove un elemento dall'array
`$addToSet`	Aggiunge all'array se non presente
`$rename`	Rinomina un campo

Aggregazione

Stadi della Pipeline

`$match`	Filtra i documenti (come WHERE)
`$group`	Raggruppa e aggrega
`$project`	Ridisegna i documenti (come SELECT)
`$sort`	Ordina i risultati
`$limit` / `$skip`	Paginazione
`$lookup`	Join esterno sinistro con un'altra collection
`$unwind`	Decostruisce un array in documenti

Esempio di Aggregazione

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

Indici

Creazione e Rimozione

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

Tipi di Indice

`Single field`	Indice su un campo ({ name: 1 })
`Compound`	Campi multipli ({ a: 1, b: -1 })
`Text`	Ricerca full-text ({ field: 'text' })
`2dsphere`	Query geospaziali
`TTL`	Scadenza automatica dei documenti

Informazioni sugli Indici

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

Progettazione dello Schema

Embedding vs Riferimento

`Embed`	1:1 o 1:pochi, dati letti insieme
`Reference`	1:molti, dati acceduti indipendentemente
`Embed`	Sotto-documento raramente supera 16 MB
`Reference`	Relazioni molti-a-molti

Validazione dello Schema

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

Replica

Concetti del Replica Set

`Primary`	Riceve tutte le scritture
`Secondary`	Replica dal primary, può servire letture
`Arbiter`	Vota nelle elezioni, non conserva dati

Comandi Replica Set

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

Pattern Comuni

Transazioni

```
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();
```

Bulk Write

```
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);
});
```

Comandi mongosh

Helper della Shell

`show dbs`	Elenca i database
`show collections`	Elenca le collection nel db corrente
`db.stats()`	Statistiche del database
`db.collection.stats()`	Statistiche della collection
`db.collection.countDocuments({})`	Conta i documenti
`db.collection.distinct('field')`	Valori distinti di un campo

Export e 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/
```