Cheat Sheet for comprensive MongoDB

MongoDB Comprehensive Cheat Sheet

1. Introduction to MongoDB

- **Document-Oriented**: Stores data in JSON-like documents.
- **Scalability**: Supports horizontal scaling through sharding.
- **Flexibility**: Schema-less design allows flexible data models.
- **High Performance**: Built-in indexing and aggregation capabilities.

---

2. Basic Concepts

2.1 Databases

- **Create Database**: `use myDatabase`
- **List Databases**: `show dbs`
- **Drop Database**: `db.dropDatabase()`

2.2 Collections

- **Create Collection**: `db.createCollection("myCollection")`
- **List Collections**: `show collections`
- **Drop Collection**: `db.myCollection.drop()`

2.3 Documents

- **Insert Document**: `db.myCollection.insertOne({name: "John", age: 30})`
- **Insert Multiple Documents**: `db.myCollection.insertMany([{name: "Jane", age: 25}, {name: "Doe", age: 35}])`

---

3. CRUD Operations

3.1 Create (Insert)

- **Insert One**: `db.myCollection.insertOne({name: "John", age: 30})`

- **Insert Many**: `db.myCollection.insertMany([{name: "Jane", age: 25}, {name: "Doe", age: 35}])`

3.2 Read (Query)

- **Find One**: `db.myCollection.findOne({name: "John"})`
- **Find All**: `db.myCollection.find({})`
- **Query Operators**:

- `$eq`: `db.myCollection.find({age: {$eq: 30}})`
- `$gt`: `db.myCollection.find({age: {$gt: 25}})`
- `$lt`: `db.myCollection.find({age: {$lt: 35}})`
- `$in`: `db.myCollection.find({age: {$in: [25, 30]}})`

3.3 Update

- **Update One**: `db.myCollection.updateOne({name: "John"}, {$set: {age: 31}})`
- **Update Many**: `db.myCollection.updateMany({age: {$gt: 25}}, {$set: {status: "active"}})`
- **Replace One**: `db.myCollection.replaceOne({name: "John"}, {name: "John", age: 32})`

3.4 Delete

- **Delete One**: `db.myCollection.deleteOne({name: "John"})`
- **Delete Many**: `db.myCollection.deleteMany({age: {$gt: 30}})`

---

4. Indexing

4.1 Creating Indexes

- **Single Field Index**: `db.myCollection.createIndex({name: 1})`
- **Compound Index**: `db.myCollection.createIndex({name: 1, age: -1})`
- **Text Index**: `db.myCollection.createIndex({description: "text"})`

4.2 Listing Indexes

- **List Indexes**: `db.myCollection.getIndexes()`

4.3 Dropping Indexes

- **Drop Index**: `db.myCollection.dropIndex("name_1")`
- **Drop All Indexes**: `db.myCollection.dropIndexes()`

---

5. Aggregation Framework

5.1 Basic Aggregation

- **Pipeline**: `db.myCollection.aggregate([{$match: {age: {$gt: 25}}}, {$group: {_id: "$name", totalAge: {$sum: "$age"}}}])`

5.2 Common Stages

- **$match**: Filters documents.
- **$group**: Groups documents by a specified key.
- **$sort**: Sorts documents.
- **$project**: Reshapes documents.
- **$lookup**: Performs a left outer join.

---

6. Transactions

6.1 Starting a Transaction

- **Start Transaction**: `session.startTransaction()`

6.2 Committing a Transaction

- **Commit**: `session.commitTransaction()`

6.3 Aborting a Transaction

- **Abort**: `session.abortTransaction()`

---

7. Data Modeling

7.1 Embedded Documents

- **Example**: `{name: "John", address: {street: "123 Main St", city: "Anytown"}}`

7.2 Referenced Documents

- **Example**: `{name: "John", address_id: ObjectId("507f1f77bcf86cd799439011")}`

7.3 Denormalization

- **Example**: Store frequently accessed data in the same document to reduce joins.

---

8. Geospatial Queries

8.1 Creating Geospatial Indexes

- **2d Index**: `db.myCollection.createIndex({location: "2d"})`

- **2dsphere Index**: `db.myCollection.createIndex({location: "2dsphere"})`

8.2 Querying Geospatial Data

- **Find Nearby**: `db.myCollection.find({location: {$near: [longitude, latitude]}})`

- **Within Polygon**: `db.myCollection.find({location: {$geoWithin: {$geometry: {type: "Polygon", coordinates: [[[0, 0], [3, 6], [6, 1], [0, 0]]]}}}})`

---

9. Text Search

9.1 Creating Text Index

- **Create Text Index**: `db.myCollection.createIndex({description: "text"})`

9.2 Performing Text Search

- **Search**: `db.myCollection.find({$text: {$search: "keyword"}})`

---

10. Security

10.1 Authentication

- **Create User**: `db.createUser({user: "myUser", pwd: "myPassword", roles: ["readWrite"]})`

- **Authenticate**: `db.auth("myUser", "myPassword")`

10.2 Authorization

- **Roles**: `read`, `readWrite`, `dbAdmin`, `userAdmin`, `clusterAdmin`

10.3 Encryption

- **TLS/SSL**: Configure MongoDB to use TLS/SSL for secure connections.

---

11. Backup and Restore

11.1 Backup

- **mongodump**: `mongodump --db myDatabase --out /path/to/backup`

11.2 Restore

- **mongorestore**: `mongorestore --db myDatabase /path/to/backup/myDatabase`

---

12. Monitoring and Performance Tuning

12.1 Monitoring

- **mongostat**: Provides a quick overview of the status of a running mongod instance.

- **mongotop**: Tracks the amount of time a mongod instance spends reading and writing data.

12.2 Performance Tuning

- **Indexing**: Ensure relevant fields are indexed.

- **Sharding**: Distribute data across multiple servers.

- **Query Optimization**: Use `$explain` to analyze query performance.

---

13. Sharding and Replication

13.1 Sharding

- **Add Shard**: `sh.addShard("shard0001/host1:port,host2:port")`

- **Enable Sharding**: `sh.enableSharding("myDatabase")`

13.2 Replication

- **Replica Set**: A group of mongod instances that maintain the same data set.

- **Add Member**: `rs.add("host:port")`

---

14. MongoDB Shell (mongosh)

14.1 Basic Commands

- **Show Databases**: `show dbs`

- **Use Database**: `use myDatabase`

- **Show Collections**: `show collections`

14.2 Advanced Commands

- **Explain**: `db.myCollection.find({}).explain("executionStats")`

- **Profile**: `db.setProfilingLevel(2)`

---

15. Tips and Tricks

15.1 Bulk Operations

- **Bulk Write**: `db.myCollection.bulkWrite([{insertOne: {document: {name: "John", age: 30}}}])`

15.2 Projections

- **Select Fields**: `db.myCollection.find({}, {name: 1, _id: 0})`

15.3 Cursor Methods

- **Limit**: `db.myCollection.find({}).limit(10)`

- **Skip**: `db.myCollection.find({}).skip(5)`

- **Sort**: `db.myCollection.find({}).sort({age: -1})`

15.4 Data Validation

- **Schema Validation**: `db.createCollection("myCollection", {validator: {$jsonSchema: {bsonType: "object", required: ["name", "age"], properties: {name: {bsonType: "string"}, age: {bsonType: "int"}}}}})`

This cheat sheet provides a comprehensive overview of MongoDB, covering essential concepts, operations, and best practices. Use it as a quick reference to enhance your MongoDB skills.

By Ahmed Baheeg Khorshid

ver 1.0