**CRUD Operations**

Insert:

db.collection.insert()

db.collection.insertOne({})

db.collection.insertMany([{}, {}, {}])

In MongoDB, each document stored in a collection requires a unique \_id field that acts as a primary key. If an inserted document omits the \_id field, the MongoDB driver automatically generates an ObjectId for the \_id field.

Query:

db.collection.find(filter, projection)

Select all documents:

pass empty document as filter: find({})

Specify conditions:

1. One condition - Equality: {<field>: <value>}

If field’s value is a nested document/array, then specify <value> as doc/array.

Also validate element order.

1. One condition - Using Operators: {<field>: {<operator>: <value>}}

e.g. {status: {$in: [“A”, “D”]}}

1. Compound conditions - Using AND: {field1: value1, field2: value2}
2. Compound conditions - Using OR: {$or: [{}, {}]}
3. Compound conditions - Using AND/OR: {

status: “A”,

$or: [{qty: {$lt: 30}}, {item: “B”}]

}

1. Reference to nested field: dot notation. <field.nestedField>
2. Query an array:
3. Exact match: see a.
4. Contain elements that meet at least one of the conditions (value can also be document): {arrField: {<field1>: <value1>, <field2>: <value2>}}
5. Contain elements that meets all conditions: {arrField: {$elemMatch: {<field1>: <value1>, <field2>: <value2>}}}

Find() method returns a cursor.

Aggregation Pipeline

MongoDB aggregation pipeline consists of stages. Each stage transforms the documents as they pass through the pipeline.

Early filtering:

If your aggregation operation requires only a subset of the data in a collection, use the $match, $limit, and $skip stages to restrict the documents that enter at the beginning of the pipeline.

Group stage:

$group stage.

Index

MongoDB creates a unique index on the \_id field during the creation of a collection.

Create an index (B+ Tree):

db.colleciton.createIndex(<key and spec>, <options>)

**Command**

show databases: show dbs

switch to database (will create upon insert if not exists): use <database>

show current database: db

show collections for current database: show collections

show users for current database: show users

exit shell: quit()

quick reference: <https://docs.mongodb.com/manual/reference/mongo-shell/>

**Connection**

Server

By default, mongod process uses the /data/db directory. Make sure the permission of this directory.

Run without path (without access control ): mongod

Specify path of data directory: mongod --dbpath <path>

Enable Authentication: mongod --auth --dbpath <path>

Client

* Authentication during connection:
  + mongo (--port 27017) -u "myUserAdmin" -p "abc123" --authenticationDatabase "admin"
* Authentication after connection:
  + connect to instance: mongo --port 27017
  + switch to authentication database, and use db.auth(<username>, <pwd>):
    - use admin
    - db.auth(“XXX”, “XXX”)

Create user administrator:

See <https://docs.mongodb.com/manual/tutorial/enable-authentication/>

* start MongoDB instance without access control: mongod
* connect to the instance: mongo --port 27017
* add a user with the userAdminAnyDatabase role in a database (the user can have other roles in a different databse):

use admin

db.createUser(

{

user: "myUserAdmin",

pwd: "abc123",

roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]

}

)

Create additional user:

Once authenticated as the user administrator, use db.createUser() to create additional users.

use test

db.createUser(

{

user: "myTester",

pwd: "xyz123",

roles: [ { role: "readWrite", db: "test" },

{ role: "read", db: "reporting" } ]

}

)

Manage users and roles:

See <https://docs.mongodb.com/manual/tutorial/manage-users-and-roles/>