# Learn MongoDB

MongoDB → NoSQL → Not only Structured Query Langage

Document → Group of field-value pairs. represents object

collection → one or more documents.

database → group of collections.

mongodb shell → mongosh.

## Chapters

1. Databases
2. Insert
3. Datatypes
4. Sorting & limiting
5. find
6. update
7. delete
8. comparision operators
9. indexes
10. collections.

show dbs → used to show all the databases

use ~~school~~ <db_name> → creates database if no database with <db_name> exists, and switches to that database.

db.createCollection(collection_name) → creates a new collection with the name in the database used.

db.dropDatabase() → deletes the particular database that the shell presently exist.

db.collection.insertOne() → The data with no particular structure can be added or inserted into the collection. The structure used is in json: {x:" ", y:" "}.

db.collection.find() → displays all the documents within the particular collection.

db.collection.insertMany() → inserts more than one document at once. The format used is as below.
[{ }, { }, { }, ... { }]

Datatypes in Mongodb.

String: series of text within quotes.
Integer: whole numbers.

Double: number with a decimal portion.

boolean: True or false

Date: any date

null: no data.

arrays: array of any datatype.

Nested documents: documents within documents (objects)

---

db.collection.find().sort({field:1}) → sort the documents in ascending/alphabetical order & show.

db.collection.find().sort({field:-1}) → sort the documents in descending/reverse alphabetical order & show.

db.collection.limit(parameter) → displays the parameter number of documents from the collection.

We can use both sort & limit.

db.collection.find({field:value}) → displays the documents with the condition met. we can also have more than one field.

db.collection.find({}, {field: true/false}) → displays only the fields in the whole collection.

The format for find() is → db.collection.find({query},{projection})

query → condition

projection → output fields needed based on boolean.

db.collection.updateOne(filter, update) → format used for update the documents.

db.collection.updateOne({field:value}, {$set:{field:value}})

The above command first checks the documents that check for the field=value condition then the first document is updated with the $set, that is value is altered or added.

db.collection.updateOne({field:value}, {$unset:{field:""}})

The above command removes the field from the first document matching the condition

db.collection.updateMany({}, {$set:{field:value}})

The above command changes and updates all the documents matching the condition.

db.collection.updateMany({field:{$exists:value}}, {})

The above command checks for the condition and updates the fields with the condition where the field checked does the field does not exist.

db.collection.deleteOne({field:value}) → deletes the document with the condition matching the field= value.

db.collection.deleteMany({condition}) → deletes multiple documents

comparision Query Operators: comparision operators return data based on value comparisons.

db.collection.find({field:{$ne: value}})

finds and returns the documents with the condition of $ne → not equal to.

similarly other operators with the same format,

$ne : equal to
$lt : lesser than
$lte : lesser than or equal to
$gt : Greater than
$gte : Greater than or equal to
$in : checks for range used in [value1, value2].
$nin : Not in, negate of in [value1, value2].

---

Logical Query Operators: logical operators return data based on expressions that evaluate to true or false.

$and → all documents that match the conditions of both clauses
$not → do not match the query expression
$nor → all documents that fail to match both clause
$or → all documents that match the conditions of either clause.

db.collection.find({$and:[{condition1}, {condition2}]})

Indexes: Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a collection scan i.e scan every document in a collection to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

---

db.collection.find({condition}).explain("executionStats")

The above command displays the documents matching the above condition first and then displays the statistics for execution.

---

The find() function performs linear search.

---

db.collection.createIndex({field:1/-1})

The above command creates a new index with the particular matching condition and returns a name of the index. Now when the search will be performed again, the search takes place comparitively much faster.

---

db.collection.getIndexes()

returns the array of existing & associated indexes for the particular collection.

db.collection.dropIndex(indexName) → deletes the created & associated index. with the particular indexName.

---

show collections;

displays all the collections in the particular database

---

db.createCollection("collection", {capped:true, size:10000000})

The above command creates a new collection for which a maximum size is specified.

---

db.createCollection("collection", {capped:true, max:xyz}, {autoIndexId:false}).

The above command creates a collection which can contain a maximum of xyz number of documents for which the auto indexing is turned out to be false.

---

db.collection.drop() → is used to delete a collection

---