MongoDB Complete commands

Objectives

This session will give the knowledge about

- Introduction to MongoDB
- MongoDB Commands

Mongo DB

- A document-oriented, NoSQL database developed by 10gen (Founded Year : 2007)
- Hash-based, schema-less database
- No Data Definition Language - you can store hashes with any keys and values that you choose
- Uses BSON format - Based on JSON – B stands for Binary
- Written in C++
- Supports APIs (drivers) in many computer languages - JavaScript, Python, Ruby, Perl, Java, Java Scala, C#, C++, Haskell, Erlang

Who Uses MongoDB

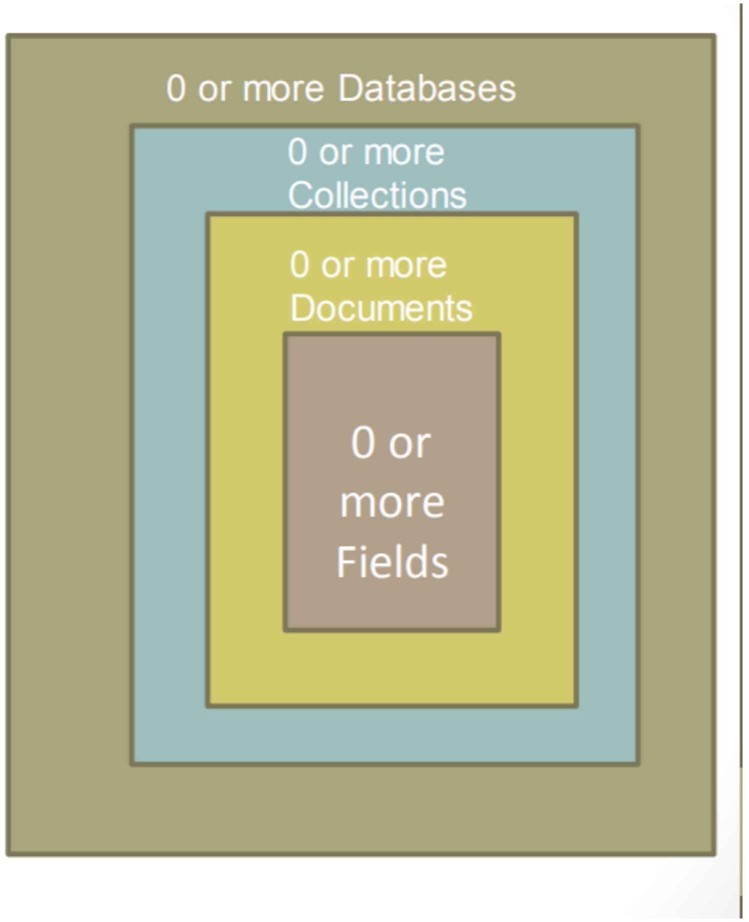
- 3272 companies reportedly use MongoDB in their tech stacks, including Uber, Lyft, and ViaVarejo.
- 28132 developers on StackShare have stated that they use MongoDB.
- Refer: https://www.mongodb.com/who-uses-mongodb
- Google, Facebook, Adobe, EA Sports, Cisco, Ebay
- SAP, Verizon, Paypal
- Jobs: https://www.naukri.com/mongodb-jobs-in-south-india

MongoDB Terminology

RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document (JSON, BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition	Shard

MongoDB: Hierarchical Objects

- A database may have zero or more 'collections'.
- A collection may have zero or more 'documents'.
- A document may have one or more 'fields'.



MongoDB: Installation

- Download at: https://www.mongodb.com/download-center/community

show dbs command

To check your currently selected database, use the command db

```
>db
movies
```

If you want to check your databases list, use the command show dbs.

```
>show dbs
local     0.78125GB
test      0.23012GB
```

To create DB: MongoDB use DATABASE_NAME is used to create database

```
>use student
>switched to db movies
```

createCollection() Method

To create a collection in the selected DB

```
Syntax: db.createCollection(CollectionName,Options)
>db.createCollection("cse")
{ "ok" : 1 }
```

To display the collections available in the selected DB

```
>show collections
cse
```

createCollection() Method

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexId	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

insert() Method

To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.

Syntax

The basic syntax of insert() command is as follows:

```
>db.COLLECTION_NAME.insert(document)
```

To insert a new document in the collections

```
> db.cse.insert ({"stud_id":101, "stud_name":"aadhav"})
WriteResult({ "nInserted" : 1 })
```

Mongo DB Datatypes

- String − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- Integer − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- Boolean − This type is used to store a boolean (true/ false) value.
- Double − This type is used to store floating point values.
- Min/ Max keys − This type is used to compare a value against the lowest and highest BSON elements.

Mongo DB Datatypes

- Arrays − This type is used to store arrays or list or multiple values into one key.
- Timestamp − ctimestamp. This can be handy for recording when a document has been modified or added.
- Object − This datatype is used for embedded documents.
- Null − This type is used to store a Null value.
- Symbol − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.

Mongo DB Datatypes

- Date − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- Object ID − This datatype is used to store the document’s ID.
- Binary data − This datatype is used to store binary data.
- Code − This datatype is used to store JavaScript code into the document.
- Regular expression − This datatype is used to store regular expression.

find() command

To display all documents available in the collection

```
> db.cse.find();
{ "_id" : ObjectId("5ec92a92fef01e2e7c1caf69"), "stud_id" : 101, "stud_name" :
"aadhav" }
```

To display all documents with alignment

```
> db.cse.find().pretty();
{
        "_id" : ObjectId("5ec92a92fef01e2e7c1caf69"),
        "stud_id" : 101,
        "stud_name" : "aadhav"
}
```

Projections

To display all documents without id

```
> db.cse.find({},{"_id":0});
{ "stud_id" : 101, "stud_name" : "aadhav" }
```

To display only one field (Projections)

```
> db.cse.find({},{"stud_id":1,"_id":0});
{ "stud_id" : 101 }
```

_id is 12 bytes hexadecimal number unique for every document in a collection. 12 bytes are divided as follows:

_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)

insertOne() command

To insert documents

```
> db.cse.insertOne ({"stud_id":102, "stud_name":"madhav"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5ec92eaafef01e2e7c1caf6d")
}
```

insert() command

To insert multiple documents

```
> db.cse.insert ([{"stud_id":103, "stud_name":“yadhav"},{"stud_id":104, "mark":98}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]   })
```

insertMany() command

To insert multiple documents

```
> db.cse.insertMany([{"stud_id":105, "stud_name":"sukdev"}, {"stud_id":105, "stud_name":"sukdev"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5ec92f0afef01e2e7c1caf70"),
                ObjectId("5ec92f0afef01e2e7c1caf71")
        ]
}
```

Save() Method

The save() method replaces the existing document with the new document passed in the save() method. For save, if you provide _id, it will update. If you don't, it will insert.

Syntax

```
>db.collectionName.save({_id:ObjectId(),new_document})
```

Example

```
>db.cse.save({"_id":ObjectId("5ec9325ffef01e2e7c1caf72"), "stud_id":105,
"stud_name":"sukdev"})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
```

Selections

To display all the documents in a collection:

```
> db.cse.find();
{ "_id" : ObjectId("5ec92a92fef01e2e7c1caf69"), "stud_id" : 101, "stud_name" : "aadhav" }

> db.cse.find({});
{ "_id" : ObjectId("5ec92a92fef01e2e7c1caf69"), "stud_id" : 101, "stud_name" : "aadhav" }

 db.cse.find({}).pretty();
{
        "_id" : ObjectId("5ec92a92fef01e2e7c1caf69"),
        "stud_id" : 101,
        "stud_name" : "aadhav"
}
```

Querying

Specify Equality Condition

The following example selects from the inventory collection all documents where the status equals "D":

```
> db.cse.find( { stud_id: 102 } )
```

```
{ "_id" : ObjectId("5ec92c14fef01e2e7c1caf6a"), "stud_id" : 102, "stud_name" : "madhav" }

{ "_id" : ObjectId("5ec92c79fef01e2e7c1caf6b"), "stud_id" : 102, "stud_name" : "madhav" }
```

Where Clause in MongoDB

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

Selections

Insert the following documents:

```
db.cse.insert({sid:501,sname:'name1',age:23,mark:68})
db.cse.insert({sid:502,sname:'name2',age:22,mark:98})
db.cse.insert({sid:503,sname:'name3',age:23,mark:88})
db.cse.insert({sid:504,sname:'name4',age:24,mark:78})
db.cse.insert({sid:505,sname:'name5',age:25})
db.cse.insert({sid:506,sname:'name6',mark:75})
```

AND, OR in MongoDB

```
>db.mycol.find(
   { $and: [
         {key1: value1},
{key2:value2}
      ]  } )
```

```
> db.cse.find({ $and: [{sid:
{$gt:500}},{mark:{$gte:70}} ] } )
```

```
>db.mycol.find(
   { $or: [
         {key1: value1},
{key2:value2}
      ]  } )
```

```
> db.cse.find({ $or: [{sid:
{$gt:500}},{mark:{$gte:70}} ] } )
```

Querying with conditions

Specify Conditions Using Query Operators

```
db.cse.find( { sid: { $in: [ 502,504] } } )
```

Specify AND Conditions

```
db.cse.find({sid: {$gt:500},mark:{$gte:70}})
```

Specify AND with OR Conditions

```
db.cse.find( {$and: [
                {sid:{$gt:500}},
                {$or:[
                    {age:23},{mark:98}
                ]}
            ] } );
```

Insert with Array

To insert list of values in a document

```
db.cse.insert({sid:507,sname:'student7',age:24,mark:[78,65,98,76,88]})
WriteResult({ "nInserted" : 1 })
```

To display values from the list of values in a document

```
db.cse.insert({sid:507,sname:'student7',age:24,mark:[78,65,98,76,88]})
WriteResult({ "nInserted" : 1 })
```

Aggregate Functions

To find the sum of values in a document

```
db.cse.aggregate([{$group : {_id : "$by_user", sum_age : {$sum : "$age"}}}]);
{ "_id" : null, "sum_age" : 117 }
```

To find average of values in a document

```
db.cse.aggregate([{$group : {_id : "$by_user", avg_mark : {$avg : "$mark"}}}]);
{ "_id" : null, "avg_mark" : 81.4 }
```

To find min of values in a document

```
db.cse.aggregate([{$group : {_id : "$by_user", min_mark : {$min : "$mark"}}}]);
{ "_id" : null, "min_mark" : 68 }
```

Aggregate Functions

To find min of values in a document

```
 db.cse.aggregate([{$group : {_id : "$by_user", max_mark : {$max : "$mark"}}}]);
{ "_id" : null, "max_mark" : 98 }
```

To count in a document

```
db.cse.aggregate( [ { $match: {mark: {$gt: 80}}}, {$count: "above 80"} ] );
{ "above 80" : 2 }
```

To ascending order in a document

```
db.cse.find({},{_id:0,sname:1,mark:1}).sort({mark:1}) //Ascending order
db.cse.find({},{_id:0,sname:1,mark:1}).sort({mark:1}).limit(1) //topper
```

Update in MongoDB

db.collection.update()

```
db.cse.update({sid:502},{sid:502,sname:'vihan',age:22,mark:98})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

db.collection.updateOne()

```
db.cse.updateOne({sid:502},{sid:502,sname:'vihan',age:22,mark:98})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

db.collection.updateMany()

```
db.cse.updateMany({sid:502},{sid:502,sname:'vihan',age:22,mark:98})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

remove() Method

MongoDB's remove() method is used to remove a document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag.

- **deletion criteria** − (Optional) deletion criteria according to documents will be removed.
- **justOne** − (Optional) if set to true or 1, then remove only one document.

Syntax

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```

remove() Method

Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. This is equivalent of SQL's truncate command.

```
>db.cse.remove({stud_id:105});
WriteResult({ "nRemoved" : 4 })
```

Remove Only One

If there are multiple records and you want to delete only the first record, then set justOne parameter in remove() method.

```
>db.cse.remove({stud_id:102},1);
WriteResult({ "nRemoved" : 1 })
```

Other important methods

- db.collection.findAndModify()
- db.collection.findOneAndUpdate()
- db.collection.findOneAndReplace()
- db.collection.save().
- db.collection.bulkWrite().

drop() Method

MongoDB's db.collection.drop() is used to drop a collection from the database.

Syntax

```
db.COLLECTION_NAME.drop()
```

Example

```
>db.cse.drop()
true
```

db.dropDatabase() command

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax

```
db.dropDatabase()
```

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

```
>use student
switched to db mydb
>db.dropDatabase()
>{ "dropped" : "mydb", "ok" : 1 }
```

Querying with nested documents

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Querying with nested documents

Match an Embedded/Nested Document

To specify an equality condition on a field that is an embedded/nested document, use the query filter document { <field>: <value> } where <value> is the document to match.

For example, the following query selects all documents where the field size equals the document { h: 14, w: 21, uom: "cm" }:

```
> db.inventory.find( { size: { h: 14, w: 21, uom: "cm" } },{_id:0})
```

```
{ "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" },
"status" : "A" }
```

Querying with nested documents

Query on Nested Field

The following example selects all documents where the field uom nested in the size field equals "in":

```
db.inventory.find( { "size.uom": "in" } ,{_id:0})
```

Specify Match using Query Operator

The following query uses the less than operator ($lt) on the field h embedded in the size field:

```
db.inventory.find( { "size.h": { $lt: 15 } } ,{_id:0})
```

Querying with nested documents

Specify AND Condition

The following query selects all documents where the nested field h is less than 15, the nested field uom equals "in", and the status field equals "D":

```
db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "in", status: "D" }
,{_id:0} )
```

Querying with Arrays

```
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
   { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },
   { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
   { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
]);
```

Querying with Arrays

Match an Array

The following example queries for all documents where the field tags value is an array with exactly two elements, "red" and "blank", in the specified order:

```
db.inventory.find( { tags: ["red", "blank"] } ,{_id:0} )
```

If, instead, you wish to find an array that contains both the elements "red" and "blank", without regard to order or other elements in the array, use the $all operator:

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } ,{_id:0} )
```

Querying with Arrays

Query an Array for an Element

The following example queries for all documents where tags is an array that contains the string "red" as one of its elements:

```
db.inventory.find( { tags: "red" } ,{_id:0} )
```

The following operation queries for all documents where the array dim_cm contains at least one element whose value is greater than 25.

```
db.inventory.find( { dim_cm: { $gt: 25 } } ,{_id:0} )
```

Querying with Arrays

Specify Multiple Conditions for Array Elements

The following example queries for documents where the dim_cm array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than 15 condition and another element can satisfy the less than 20 condition, or a single element can satisfy both:

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } ,{_id:0} )
```

The following example queries for documents where the dim_cm array contains at least one element that is both greater than ($gt) 22 and less than ($lt) 30:

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } ,{_id:0} )
```

Querying with Arrays

Query for an Element by the Array Index Position

The following example queries for all documents where the second element in the array dim_cm is greater than 25:

```
db.inventory.find( { "dim_cm.1": { $gt: 25 } } ,{_id:0} )
```

Query an Array by Array Length

The following selects documents where the array tags has 3 elements.

```
db.inventory.find( { "tags": { $size: 3 } } ,{_id:0} )
```

Querying with Arrays and Nested documents

```
db.inventory.insertMany( [
   { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },
   { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },
   { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },
   { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
]);
```

Querying with Arrays and Nested documents

Query for a Document Nested in an Array

The following example selects all documents where an element in the instock array matches the specified document:

```
db.inventory.find( { "instock": { warehouse: "A", qty: 5 } } ,{_id:0} )
```

The following query does not match any documents in the inventory collection:

```
db.inventory.find( { "instock": { qty: 5, warehouse: "A" } } ,{_id:0} )
```

Querying with Arrays and Nested documents

Specify a Query Condition on a Field Embedded in an Array of Documents

If you do not know the index position of the document nested in the array, concatenate the name of the array field, with a dot (.) and the name of the field in the nested document.

The following example selects all documents where the instock array has at least one embedded document that contains the field qty whose value is less than or equal to 20:

```
db.inventory.find( { 'instock.qty': { $lte: 20 } } ,{_id:0} )
```

Querying with Arrays and Nested documents

Use the Array Index to Query for a Field in the Embedded Document

The following example selects all documents where the instock array has as its first element a document that contains the field qty whose value is less than or equal to 20:

```
db.inventory.find( { 'instock.0.qty': { $lte: 20 } } ,{_id:0} )
```

NOTE

When querying using dot notation, the field and index must be inside quotation marks.

Querying with Arrays and Nested documents

A Single Nested Document Meets Multiple Query Conditions on Nested Fields

The following example queries for documents where the instock array has at least one embedded document that contains both the field qty equal to 5 and the field warehouse equal to A:

```
db.inventory.find( { "instock": { $elemMatch: { qty: 5, warehouse: "A" } } } ,{_id:0} )
```

The following example queries for documents where the instock array has at least one embedded document that contains the field qty that is greater than 10 and less than or equal to 20:

```
db.inventory.find( { "instock": { $elemMatch: { qty: { $gt: 10, $lte: 20 } } } } ,{_id:0} )
```

Querying with Arrays and Nested documents

Query for Null or Missing Fields

Equality Filter

The { item : null } query matches documents that either contain the item field whose value is null or that do not contain the item field.

```
db.inventory.find( { item: null } ,{_id:0} )
```

Type Check

The { item : { $type: 10 } } query matches only documents that contain the item field whose value is null; i.e. the value of the item field is of BSON Type Null (type number 10) :

```
db.inventory.find( { item : { $type: 10 } } ,{_id:0} )
```

Querying with Arrays and Nested documents

Existence Check

The following example queries for documents that do not contain a field.

The { item : { $exists: false } } query matches documents that do not contain the item field:

```
db.inventory.find( { item : { $exists: false } } ,{_id:0} )
```

Summary

This session will give the knowledge about

- Introduction to MongoDB
- MongoDB Commands