# Experiment5-PerformingCRUDOperationsforunstructureddata

For storing data in a MongoDB, you need to create a database first. It will allow you to systematicallyorganize your data so that it can be retrieved as per requirement. If you wish to delete a database,MongoDBalsoallowsyoutodeletethat.

# WhatisCRUDinMongoDB?

CRUD operations describe the conventions of a user-interface that let users view, search, and modifyparts ofthedatabase.

MongoDB documents are modified by connecting to a server, querying the proper documents, andthen changing the setting properties before sending the data back to the database to be updated.CRUDis data-oriented, andit’sstandardizedaccordingtoHTTPactionverbs.

# WhenitcomestotheindividualCRUDoperations:

1. TheCreateoperationisusedto insertnew documentsinthe MongoDBdatabase.
2. TheReadoperationisusedto queryadocumentinthedatabase.
3. TheUpdateoperationis usedtomodifyexistingdocumentsinthedatabase.
4. TheDeleteoperationisusedtoremove documentsinthedatabase.

# CreatingaDatabaseinMongoDB

MongoDB has no "create" command for creating a database. Also, it is essential to note thatMongoDB does not provide any specific command for creating a database. This might seem a bitagitated if you are new to this subject and database tool or in case you have used that conventionalSQL as your database where you are required to create a new database, which will contain table and,youwill thenhaveto usethe INSERT INTOTABLE toinsertvaluesmanuallywithinyourtable.

In this MongoDB tool, you need not have to produce, or it is optional to create a database manually.This is because MongoDBhas the feature of automatically creating it for the first time for you, onceyou save your value in that collection. So, explicitly, you do not need to mention or put a commandto createa database;insteadit willbe createdautomaticallyoncethecollectionisfilledwithvalues.

# The“use”Command\_for\_Creating\_Database\_in\_MongoDB

You can make use of the "use" command followed by the database\_name for creating a database.This command will tell the MongoDB client to create a database by this name if there is no databaseexistsbythisname.Otherwise,thiscommandwillreturnthe existing database thathasthename.

Show list of databases > show dbs; admin0.000GB config0.000GB local0.000GBmyNewDB 0.000GB**>use**

**RetailBikeDB**switchedtodbRetailBikeDB



# CreatingandshowingcollectionsinMongoDB

Collections are like that of tables of RDBMS and are capable enough to store documents of diverse ordissimilartypes.CreationandremovalofcollectionsinMongoDBcanbedoneinspecificways.Inthischapter,youwilllearnabout thecreationofcollections ina databasecreatedusingMongoDB.

Creationofcollectioncanbedoneusingdb.createCollection(name)

* db.createCollection("production.categories")

{ "ok": 1 }

* db.createCollection("production.brands")

{ "ok": 1 }

* db.createCollection("production.products")

{ "ok": 1 }

* db.createCollection("production.stocks")

{ "ok": 1 }

* db.createCollection("sales.stores")

{ "ok": 1 }

* db.createCollection("sales.staffs")

{ "ok": 1 }

* db.createCollection("sales.customers")

{ "ok": 1 }

* db.createCollection("sales.orders")

{ "ok": 1 }

* db.createCollection("sales.order\_items")

{ "ok": 1 }



Wecanshow listofcollectionusing“Showcollections” commandsinMongoDB.

* show collectionsproduction.brandsproduction.categoriesproduction.productsproduction.stockssales.customerssales.order\_itemssales.orderssales.staffssales.stores



# CRUDoperation

CRUD operation is one of the essential concepts of a database system. Inserting data in the databasecomes under one of the CRUD operations. If you do not insert data in your database, you will not beable tocontinuewithotheractivitieswithinyour document.

# CreateOperations

ForMongoDBCRUD,ifthespecifiedcollectiondoesn’texist,thecreateoperationwillcreatethe

collection when it’s executed. Create operations in MongoDB target a single collection, not multiplecollections.Insertoperations in MongoDB areatomicona singledocumentlevel.

MongoDB provides two different create operations that you can use to insert documents into acollection:

* 1. db.collection.insertOne()
  2. db.collection.insertMany()

## TheinsertOne()method

As the namesake, insertOne() allows you to insert one document into the collection. Example -db.movie.insertOne({\_id:2, writername:"StanLee",name:"Aquaman" })

## TheinsertMany()method

It’s possible to insert multiple items at one time by calling the insertMany() method on the desiredcollection. In this case, we pass multiple items into our chosen collection (RetailDB) and separatethem by commas. Within the parentheses, we use brackets to indicate that we are passing in a list ofmultipleentries.This iscommonlyreferred toas a nestedmethod.

IhavetakenexampleofRetail Bikestore databasetodemonstrate insert crudoperations.

# production.categories

db.production.categories.insertMany([

{

"category\_id": 1,"category\_name":"RoadBike"

},

{

"category\_id": 2,"category\_name":"MountainBike"

},

{

"category\_id": 3,"category\_name":"HybridBike"

},

{

"category\_id": 4,"category\_name":"FoldingBike"

},

{

"category\_id": 5,"category\_name":"TouringBike"

},

{

"category\_id": 6,"category\_name":"CruiserBike"

},

{

"category\_id": 7,"category\_name":"WomenBike"

}

]

)



# production.products

db.production.products.insertMany([

{

"product\_id":1,

"product\_name": "Honda Superfast","brand\_id":1,

"category\_id":1,

"model\_year":1994,

"list\_price":25000

},

{

"product\_id": 2,"product\_name":"6KUBikes",

"brand\_id":2,

"category\_id":4,

"model\_year":2000,

"list\_price":30000

},

{

"product\_id": 3,"product\_name": "Bianchi","brand\_id":3,

"category\_id":2,

"model\_year":2002,

"list\_price":30000

},

{

"product\_id":4,

"product\_name": "BMC Hybrid Bike","brand\_id":4,

"category\_id":3,

"model\_year":2009,

"list\_price":45000

},

{

"product\_id":5,

"product\_name": "Huffy Women Bike","brand\_id":5,

"category\_id":7,

"model\_year":2019,

"list\_price":50000

}

]

)



# production.stocks

db.production.stocks.insertMany([

{

"store\_id":1,

"product\_id":1,

"quantity":15

},

{

"store\_id":2,

"product\_id":4,

"quantity":20

},

{

"store\_id":3,

"product\_id":5,

"quantity":30

},

{

"store\_id":1,

"product\_id":5,

"quantity":100

},

{

"store\_id":2,

"product\_id":2,

"quantity":4

},

{

"store\_id":3,

"product\_id":3,

"quantity":9

}

]

)



# sales.customers

db.sales.customers.insertMany([

{

"customer\_id": "Cus001","first\_name": "Jay","last\_name": "Mehta","phone":1234567890,

"email": ["jay@gmail.com",](mailto:jay@gmail.com)"street":"T.PRoad",

"city": "Mumbai","state": "Maharashtra","zip\_code":400072

},

{

"customer\_id": "Cus002","first\_name": "Ruhi","last\_name": "Singh","phone":7894561230,

"email": ["ruhi@yahoo.com",](mailto:ruhi@yahoo.com)"street":"M.GChauk",

"city": "Mumbai","state": "Maharashtra","zip\_code":400072

},

{

"customer\_id": "Cus003","first\_name": "Aria","last\_name": "Josh","phone":1245789630,

"email": ["aria.josh@gmail.com",](mailto:aria.josh@gmail.com)"street":"JVM",

"city": "Gandhi Nagar","state": "Gujrat","zip\_code":401235

},

{

"customer\_id": "Cus004","first\_name": "Mahi","last\_name": "Kaur","phone":4567890123,

"email": ["kaur.mahi@hotmail.com",](mailto:kaur.mahi@hotmail.com)"street":"J.V.L.R",

"city": "Mumbai","state": "Maharashtra","zip\_code":400072

},

{

"customer\_id": "Cus005","first\_name": "Aditya","last\_name": "Yadav","phone":9638527410,

"email": ["aditya@gmail.com",](mailto:aditya@gmail.com)"street":"Koliwada","city":"Pune",

"state": "Maharashtra","zip\_code":300075

}

]

)





# sales.order\_items

db.sales.order\_items.insertMany([

{

"order\_id": "ORD001","product\_id":1,

"quantity":2,

"list\_price":50000

},

{

"order\_id": "ORD002","product\_id":2,

"quantity":3,

"list\_price":90000

},

{

"order\_id": "ORD003","product\_id":3,

"quantity":1,

"list\_price":30000

},

{

"order\_id": "ORD004","product\_id":4,

"quantity":8,

"list\_price":360000

},

{

"order\_id": "ORD005","product\_id":5,

"quantity":2,

"list\_price":100000

}

]

)



# sales.orders

db.sales.orders.insertMany([

{

"order\_id": "ORD001","customer\_id": "Cus001","order\_status": "Completed","order\_date":43992,

"shipped\_date":43994,

"store\_id":1,

"staff\_id":1

},

{

"order\_id": "ORD002","customer\_id": "Cus002","order\_status": "Completed","order\_date":44221,

"shipped\_date":44227,

"store\_id":2,

"staff\_id":2

},

{

"order\_id": "ORD003","customer\_id": "Cus003","order\_status": "Completed","order\_date":44306,

"shipped\_date":44314,

"store\_id":2,

"staff\_id":2

},

{

"order\_id": "ORD004","customer\_id": "Cus004","order\_status": "Pending","order\_date":44367,

"shipped\_date":44377,

"store\_id":3,

"staff\_id":3

},

{

"order\_id": "ORD005","customer\_id": "Cus005","order\_status": "Pending","order\_date":44367,

"shipped\_date":44377,

"store\_id":1,

"staff\_id":1

}

]

)





# sales.staffs

db.sales.staffs.insertMany([

{

"staff\_id": 1,"first\_name": "Pushpa","last\_name":"Yadav",

"email": ["pushpa@gmail.com",](mailto:pushpa@gmail.com)"phone":9999999999,

"active": "Yes","store\_id":1,

"manager\_id":1

},

{

"staff\_id": 2,"first\_name": "Sadiksha","last\_name":"Singh",

"email": ["sadiksha@gmail.com",](mailto:sadiksha@gmail.com)"phone":8888888888,

"active": "Yes","store\_id":2,

"manager\_id":1

},

{

"staff\_id":3,"first\_name": "Priya","last\_name": "Nadar","email": ["priya@gmail.com",](mailto:priya@gmail.com)"phone":7777777777,

"active": "Yes","store\_id":3,

"manager\_id":1

}

]

)



# sales.stores

db.sales.stores.insertMany([

{

"store\_id":1,

"store\_name": "Ambika Showroom","phone":123456,

"email": ["ambika@gmail.com",](mailto:ambika@gmail.com)"street":"GP",

"city": "Mumbai","state": "Maharashtra","zip\_code":400072

},

{

"store\_id":2,"store\_name": "Yash Bikes","phone":789456,

"email": ["yash.bikes@yahoo.com",](mailto:yash.bikes@yahoo.com)"street":"H.G",

"city":"Pune","state": "Maharashtra","zip\_code":300075

},

{

"store\_id":3,

"store\_name": "Josh Automobiles","phone":456983,

"email": ["josh@gmail.com",](mailto:josh@gmail.com)"street":"M.G",

"city": "Gandhi Nagar","state": "Gujrat","zip\_code":401235

}

]

)



# production.brands

db.production.brands.insertMany([

{

"brand\_id": 1,"brand\_name":"Honda"

},

{

"brand\_id": 2,"brand\_name":"6KUBikes"

},

{

"brand\_id": 3,"brand\_name":"Bianchi"

},

{

"brand\_id": 4,"brand\_name":"BMC"

},

{

"brand\_id": 5,"brand\_name":"Huffy"

}

]

)



# ReadOperations

The read operations allow you to supply special query filters and criteria that let you specify whichdocuments you want. The MongoDB documentation contains more information on the availablequeryfilters. Querymodifiersmayalsobeusedtochange howmany resultsarereturned.

MongoDBhastwomethodsofreadingdocumentsfrom acollection:

* db.collection.find()
* db.collection.findOne()

## find()

In order to get all the documents from a collection, we can simply use the find() method on ourchosen collection. Executing just the find() method with no arguments will return all recordscurrentlyinthecollection.

Example-db.production.brands.find()

Here we canseethateveryrecordhas anassigned“ObjectId”mappedtothe“\_id” key.



If you want to get more specific with a read operation and find a desired subsection of the records,you can use the previously mentioned filtering criteria to choose what results should be returned.Oneofthemostcommonwaysof filteringtheresultsis to searchbyvalue.

Example-db.production.brands.find({"brand\_name":"Honda"})



## findOne()

In order to get one document that satisfies the search criteria, we can simply use the findOne()method on our chosen collection. If multiple documents satisfy the query, this method returns thefirst document according to the natural order which reflects the order of documents on the disk. Ifno documents satisfy the search criteria, the function returns null. The function takes the followingformofsyntax.

Syntax- db.{collection}.findOne({query}, {projection})Example–db.sales.order\_items.findOne({"quantity":2})



# QueryDocuments

1. **SpecifyEqualityCondition**

* db.production.products.find()-toshowallthedocuments
* db.production.products.find({"list\_price": 30000}) - to show only documents which have“list\_price”as30000



# SpecifyConditionsUsingQueryOperators

* db.sales.customers.find()-toshowallthedocuments
* db.sales.customers.find({city: { $in: [ "Mumbai", "Pune" ] } }) - to show all the documents where"city"iseither"Mumbai"or "Pune"



# SpecifyANDConditions

* db.sales.order\_items.find()-toshowallthedocuments
* db.sales.order\_items.find({quantity: 2, list\_price : { $gt: 70000}}) - Here we are trying to find howsalesorder itemdocumentsforwhichquantity is2 andlist price is greaterthan70000



# SpecifyORConditions

* db.production.products.find()-toshowallthedocuments
* db.production.products.find({ $or: [ { product\_name: "Honda Superfast" }, { model\_year : { $lt:2003 } } ] }) - to show all the document which is either "product name" as "Honda Superfast" or"modelyear"is less thanyear 2003



# UpdateOperations

Like create operations, update operations operate on a single collection, and they are atomic at asingle document level. An update operation takes filters and criteria to select the documents youwant toupdate.

You should be careful when updating documents, as updates are permanent and can’t be rolled back.Thisappliestodeleteoperations aswell.

ForMongoDBCRUD,therearethree differentmethodsofupdatingdocuments:

* db.collection.updateOne()
* db.collection.updateMany()
* db.collection.replaceOne()

## updateOne()

We can update a currently existing record and change a single document with an update operation.To do this, we use the updateOne() method on a chosen collection. To update a document, weprovidethemethodwithtwoarguments: anupdatefilter andan updateaction.

The update filter defines which items we want to update, and the update action defines how toupdate those items. We first pass in the update filter. Then, we use the “$set” key and provide thefields we want to update as a value. This method will update the first record that matches theprovidedfilter.

Example–

* db.sales.staffs.find()-to showcurrectdocumentinthe system
* db.sales.staffs.updateOne({first\_name: "Pushpa"}, {$set:{phone: 999999988}}) -update mobilenumber from 9999999999 to 999999988 for document name where name is " first\_name " incollection



## updateMany()

updateMany() allows us to update multiple items by passing in a list of items, just as we did wheninsertingmultiple items.Thisupdateoperationusesthe same syntaxforupdatingasingledocument.

Example–

db.sales.orders.find()-to showcurrentdocumentinthe system

db.sales.orders.updateMany({shipped\_date:44377}, {$set: {shipped\_date: "1-July-2021"}}) – with thehelp of this command we are updating “shipped\_date” to "1-July-2021" where shipped\_date is44377



## replaceOne()

The replaceOne() method is used to replace a single document in the specified collection.replaceOne() replaces the entire document, meaning fields in the old document not contained in thenewwillbe lost.

# DeleteOperations

Delete operations operate on a single collection, like update and create operations. Deleteoperations are also atomic for a single document. You can provide delete operations with filters andcriteria in order to specify which documents you would like to delete from a collection. The filteroptionsrely onthesamesyntax thatreadoperations utilize.

MongoDBhastwodifferentmethodsofdeletingrecordsfrom acollection:

* db.collection.deleteOne()
* db.collection.deleteMany()

## deleteOne()

deleteOne() is used to remove a document from a specified collection on the MongoDB server. Afilter criteria is used to specify the item to delete. It deletes the first record that matches theprovidedfilter.

Example–

* db.production.brands.find()-toshowcurrentdocumentsincollection
* db.production.brands.deleteOne({brand\_id:6})-deletethedocumentwhere"brand\_id"is6



## deleteMany()

deleteMany() is a method used to delete multiple documents from a desired collection with a singledelete operation. A list is passed into the method and the individual items are defined with filtercriteriaas in deleteOne().

Example–

* db.production.brands.find()-toshowcurrentdocumentsincollection
* db.production.brands.deleteMany({brand\_id: {$gt: 5}}) - delete documents for which brand id isgreaterthan5

