**ASSIGNMENT OF WEEK 2**

**( MONGO DB/EXPRESS JS)**

1. Explain in detail about CRUD operation in MongoDB.

Answer:

CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database.

MongoDB documents are modified by connecting to a server, querying the proper documents, and then changing the setting properties before sending the data back to the database to be updated. CRUD is data-oriented, and it’s standardized according to HTTP action verbs.

When it comes to the individual CRUD operations:

* The Create operation is used to insert new documents in the MongoDB database.
* The Read operation is used to query a document in the database.
* The Update operation is used to modify existing documents in the database.
* The Delete operation is used to remove documents in the database.

Create Operations

For MongoDB CRUD, if the specified collection doesn’t exist, the [create](https://docs.mongodb.com/manual/tutorial/insert-documents/) operation will create the collection when it’s executed. Create operations in MongoDB target a single collection, not multiple collections. Insert operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on a single [document](https://docs.mongodb.com/manual/core/document/) level.

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

* [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne)
* [db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany)

insertOne()

As the namesake, insertOne() allows you to insert one document into the collection. For this example, we’re going to work with a collection called RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on RecordsDB. We then provide the information we want to insert in the form of key-value pairs, establishing the schema.

Db.RecordsDB.insertOne(

{

name: “Marsh”,

age: “6 years”,

species: “Dog”,

ownerAddress: “380 W. Fir Ave”,

chipped: true

})

If the create operation is successful, a new document is created. The function will return an object where “acknowledged” is “true” and “insertID” is the newly created “ObjectId.”

> db.RecordsDB.insertOne({

... name: "Marsh",

... age: "6 years",

... species: "Dog",

... ownerAddress: "380 W. Fir Ave",

... chipped: true

... })

{

"acknowledged" : true,

"insertedId" : ObjectId("5fd989674e6b9ceb8665c57d")

}

insertMany()

It’s possible to insert multiple items at one time by calling the*insertMany()* method on the desired collection. In this case, we pass multiple items into our chosen collection (*RecordsDB*) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

db.RecordsDB.insertMany([{

name: "Marsh",

age: "6 years",

species: "Dog",

ownerAddress: "380 W. Fir Ave",

chipped: true},

{name: "Kitana",

age: "4 years",

species: "Cat",

ownerAddress: "521 E. Cortland",

chipped: true}])

db.RecordsDB.insertMany([{ name: "Marsh", age: "6 years", species: "Dog",

ownerAddress: "380 W. Fir Ave", chipped: true}, {name: "Kitana", age: "4 years",

species: "Cat", ownerAddress: "521 E. Cortland", chipped: true}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("5fd98ea9ce6e8850d88270b4"),

ObjectId("5fd98ea9ce6e8850d88270b5")

]

}

**Read Operations**

The [read](https://docs.mongodb.com/manual/tutorial/query-documents/) operations allow you to supply special query filters and criteria that let you specify which documents you want. The MongoDB documentation contains more information on the available query [filters](https://docs.mongodb.com/manual/core/document/#document-query-filter). Query modifiers may also be used to change how many results are returned.

MongoDB has two methods of reading documents from a collection:

* [db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)
* [db.collection.findOne()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)

### UPDATE OPERATIONS

Like create operations, [update](https://docs.mongodb.com/manual/tutorial/update-documents/) operations operate on a single collection, and they are atomic at a single document level. An update operation takes filters and criteria to select the documents you want to update.

We should be careful when updating documents, as updates are permanent and can’t be rolled back. This applies to delete operations as well.

For MongoDB CRUD, there are three different methods of updating documents:

* [db.collection.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne)
* [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany)
* [db.collection.replaceOne()](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne)

### DELETE OPERATIONS

[Delete](https://docs.mongodb.com/manual/tutorial/remove-documents/) operations operate on a single collection, like update and create operations. Delete operations are also atomic for a single document. You can provide delete operations with filters and criteria in order to specify which documents you would like to delete from a collection. The filter options rely on the same syntax that read operations utilize.

MongoDB has two different methods of deleting records from a collection:

* [db.collection.deleteOne()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteOne/#db.collection.deleteOne)
* [db.collection.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany)

2.What is the namespace of MongoDB?

Answer: MongoDB stores BSON (Binary Interchange and Structure Object Notation) objects in the collection. The concatenation of the collection name and database name is called a namespace.

2.How do I handle 404 responses?

Answer: