MongoDB CRUD Operations

CRUD operations are essential for interacting with databases. In MongoDB, CRUD operations allow users to perform various actions like inserting new documents, reading data, updating records and deleting documents from collections.

- **Create**: Add new documents to a collection.
- **Read**: Retrieve documents from a collection.
- **Update**: Modify existing documents.
- **Delete**: Remove documents from a collection.

1. Create Operations

The create or insert operations are used to insert or add new documents in the collection. If a collection does not exist, then it will create a new collection in the database. We can perform, create operations using the following methods provided by the MongoDB:

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.
db.createCollection()	It is used to create an empty collection.

Create Operations Example

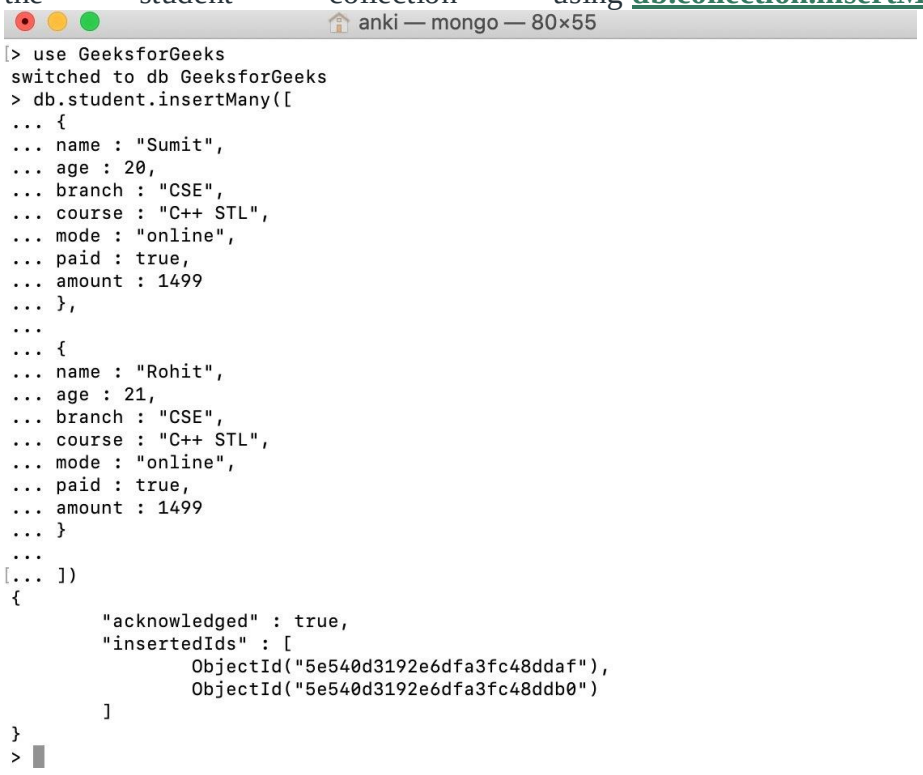
Let's look at some examples of the Create operation from CRUD in MongoDB.

Example 1:

In this example, we are inserting details of a single student in the form of document in the student collection using **db.collection.insertOne() method**.

Example 2:

In this example, we are inserting details of the multiple students in the form of documents in the student collection using **db.collection.insertMany() method**.



2. Read Operations

The Read operations are used to retrieve documents from the collection, or in other words, read operations are used to query a collection for a document. We can perform read operation using the following method provided by the MongoDB:

Method	Description
db.collection.find()	It is used to retrieve documents from the collection.

Method	Description
db.collection.findOne()	Retrieves a single document that matches the query criteria.

Note: *pretty() method is used to decorate the result such that it is easy to read.*

Read Operations Example

In this example, we are retrieving the details of students from the student collection using **db.collection.find()** method.

3. Update Operations

The update operations are used to update or modify the existing document in the collection. We can update a single document or multiple documents that match a given query. We can perform update operations using the following methods provided by the MongoDB:

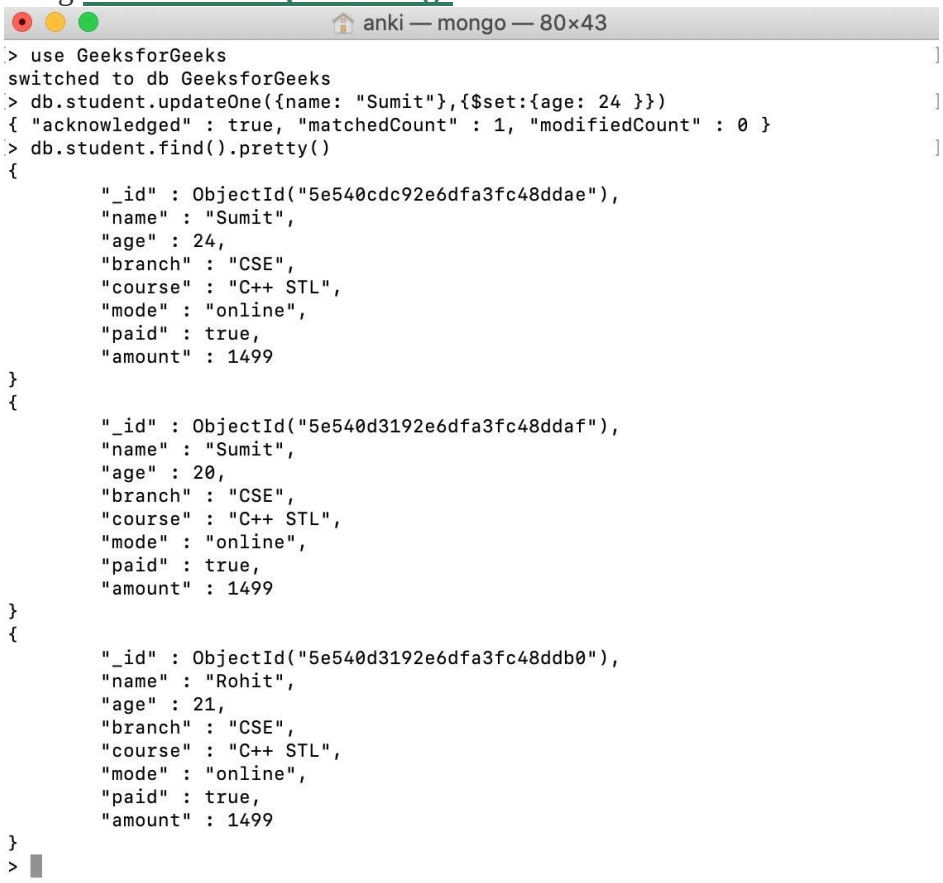
Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfy the given criteria.
db.collection.updateMany()	It is used to update multiple documents in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collection that satisfy the given criteria.

Update Operations Example

Let's look at some examples of the update operation from CRUD in MongoDB.

Example 1:

In this example, we are updating the age of Sumit in the student collection using **db.collection.updateOne()** method.



Example 2:

In this example, we are updating the year of course in all the documents in the student collection using **db.collection.updateMany()** method.

4. Delete Operations

The delete operation are used to delete or remove the documents from a collection. We can delete documents based on specific criteria or remove all documents. We can perform delete operations using the following methods provided by the MongoDB:

Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.

Delete Operations Examples

Let's look at some examples of delete operation from CRUD in MongoDB.

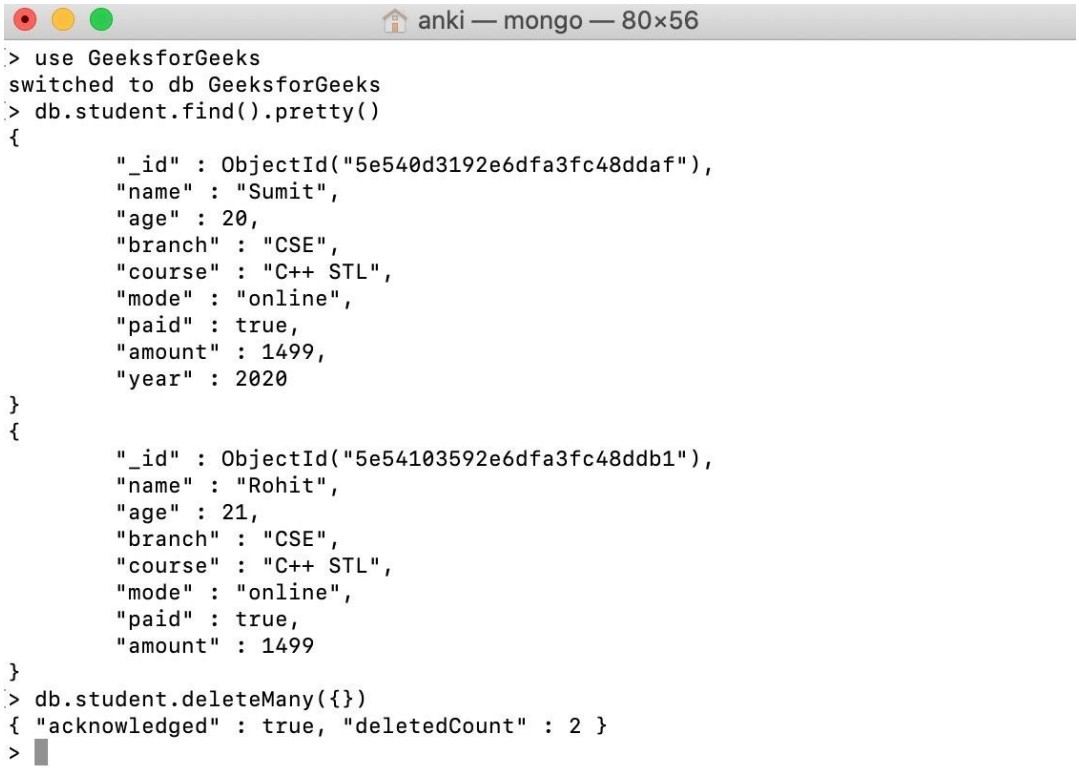
Example 1:

In this example, we are deleting a document from the student collection using **db.collection.deleteOne()** method.

```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> db.student.deleteOne({name: "Sumit"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
```

Example 2:

In this example, we are deleting all the documents from the student collection using **db.collection.deleteMany()** method.



Indexing in MongoDB

Indexing in **MongoDB** is a crucial feature that enhances query processing efficiency. Without indexing, MongoDB must scan every document in a collection to retrieve the **matching** documents and leading to **slower query performance**.
Indexes are special data structures that store information about the documents in a way that makes it easier for MongoDB to quickly locate the right data.

What is Indexing in MongoDB?

Indexing in MongoDB is a technique that improves the speed and efficiency of queries. An index is a special data structure that stores a subset of data in a way that allows MongoDB to quickly locate documents in a collection. When an index is applied, MongoDB doesn't have to scan the entire collection for the query results but instead uses the index to directly find the relevant data.
Indexes in MongoDB are **ordered** by the value of the field that the index is created on. This ordered structure enables faster searching, sorting, and filtering operations. Indexes also help to improve query performance when using conditions, sorting, and aggregation

Why is Indexing Important in MongoDB?

MongoDB provides a method called createIndex() that allows users to create an index. The key determines the field on the basis of which we want to create an index and 1 (or -1) determines the order in which these indexes will be arranged(ascending or descending).
Indexing improves the performance of:

- **Find queries** (db.collection.find()).

- **Range queries** (e.g., queries with <, >, <=, >= operators).
- **Sorting** (e.g., db.collection.find().sort()).
- **Aggregation operations** involving filtering, grouping, and sorting.

Syntax:

db.COLLECTION_NAME.createIndex({KEY:1

How to Create an Index in MongoDB

To create an index, MongoDB provides the createIndex() method. The method requires us to specify the field(s) to index and the order (ascending or descending). We can also specify optional parameters to customize the index creation.

Syntax

db.collection.createIndex({ <field>: <1 or -1> });

Example

```
db.users.createIndex({ username: 1 });
```

Parameters:

- **unique:** Ensures the indexed field contains unique values.
- **background:** Creates the index in the background to avoid blocking other database operations.
- **sparse:** Only indexes documents that contain the indexed field.
- **expireAfterSeconds:** Used for time-to-live (TTL) indexes to automatically remove documents after a certain time.
- **hidden:** Marks the index as hidden, meaning it will not be used for queries but still exists in the system.

How to Drop an Index in MongoDB

We can drop an index using the dropIndex() method. To drop multiple indexes, use dropIndexes(). The dropIndex() methods can only delete one index at a time. In order to delete (or drop) multiple indexes from the collection, MongoDB provides the dropIndexes() method that takes multiple indexes as its parameters.

Syntax (drop a single index):

db.NAME_OF_COLLECTION.dropIndex({KEY:1

Syntax (drop multiple indexes):

db.NAME_OF_COLLECTION.dropIndexes({KEY1:1, KEY2: 1})

The dropIndex() methods can only delete one index at a time. In order to delete (or drop) multiple indexes from the collection, MongoDB provides the dropIndexes() method that takes multiple indexes as its parameters.

How to View all Indexes in MongoDB

The getIndexes() method in MongoDB gives a description of all the indexes that exists in the given collection.

Syntax

db.NAME_OF_COLLECTION.getIndexes()

It will retrieve all the description of the indexes created within the collection.