NIIT

CRUD Operations in MongoDB

Objectives

- By the end of this session, you should be able to:
  - Insert a single document
  - A batch of multiple documents into a MongoDB collection
  - Add or autogenerate an _id field
  - Replace existing documents
  - Update specific fields in the documents
  - Delete all or delete specific documents

Introduction

Now that you know how to correctly find and represent the required documents from a collection, the next step is to learn how to modify the documents in the collection. When working on any database management system, you will be required to modify the underlying data.

Inserting Documents

- MongoDB collections provide a function named insert(), which is used to create a new document in a collection. The function is executed on the collection and takes the document to be inserted as an argument.
- The syntax of this function is shown in the next command:
  - db.collection.insert(<Document To Be Inserted>)
- When multiple documents need to be inserted into a collection, you can call the insert() function multiple times.
- MongoDB collections also provide the insertMany() function, which is a function specifically meant for inserting multiple documents into a collection.
  - db.movies.insertMany(< Array of One or More Documents>)
  - The result of the preceding operation contains two things i.e.

    {“acknowledged”: true, “InsertedIds”:[2,3,4,5]}
  - The first field is acknowledged with the value of true. This confirms the write operation was successfully performed. The second field of the result lists down all the IDs of the inserted documents.

Inserting Documents

- **Inserting Duplicate Keys**
  - In MongoDB collections, the value expressed by the _id field is a primary key, and so it must be unique.
  - If you try to insert a document whose key is already present in the collection, you will get a Duplicate Key Error.
  - Similarly, the operation of a bulk insert fails when one or more of the documents in the given array has a duplicate _id.
- **Inserting without _id**
  - While creating a new document, MongoDB verifies the presence and uniqueness of a given primary key and, if the primary key is not already present, the database autogenerates it and adds it into the document
  - The autogenerated primary is derived from the ObjectId constructor and it is globally unique. The same is true for bulk inserts

Deleting Documents

- **Deleting Using deleteOne():**
  - It accepts a document representing a query condition. Upon successful execution, it returns a document containing the total number of documents deleted (represented by the field deletedCount) and whether the operation was confirmed (given by the field acknowledged).
  - If the given query condition matches more than one document in the collection, only the first document will be deleted
- **Deleting Multiple Documents Using deleteMany():**
  - MongoDB collections provide the function deleteMany() to delete multiple documents in a single command.
  - The deleteMany() function must be provided with a query condition, and all the documents that match the given query will be removed:
  - Passing an empty query document is equivalent to not passing any filter; thus, all the documents are matched. In the case of deleteOne() function will delete the document that is found first.

Deleting Documents

- **Deleting Using findOneAndDelete():**
  - it behaves similarly to the deleteOne() function, it provides a few more options:
    - It finds one document and deletes it.
    - If more than one document is found, only the first one will be deleted.
    - Once deleted, it returns the deleted document as a response.
    - In the case of multiple document matches, the sort option can be used to influence which document gets deleted.
    - Projection can be used to include or exclude fields from the document in response.

Replacing Documents

- To replace a single document in a collection, MongoDB provides the method replaceOne(), which accepts a query filter and a replacement document.
  - The Syntax: db.collections.replaceOne({criteria},{new document to be replaced})
- If there is more than one document that matches the query, then only the first one will be replaced
- **_id Fields Are Immutable:**
  - Immutable fields are like normal fields; however, once assigned a value, their value cannot be changed again.
  - If the replacement document has a new value for the _id field, then the command will be failed. Because we cannot replace the value of _id as it's immutable. Hence, the update will be rolled back, and no change will happen to the record.
- **Upsert Using Replace :**
  - This operation is called an update (if found) or insert (if not found), which is further shortened to upsert.
  - We can pass an additional argument in replaceOne() to run it as the upsert command.
  - The Syntax: db.collections.replaceOne({criteria},{new document to be replaced},{upsert:true})

Replacing Documents

- **Replacing Using findOneAndReplace()**
  - We have seen the replaceOne() function, which, after successful execution, returns the counts of matched and updated documents. MongoDB provides another operation, findOneAndReplace(), to perform the same operations. However, it provides more options. Its main features are as follows:
    - As the name indicates, it finds one document and replaces it.
    - If more than one document is found matching the query, the first one will be replaced.
    - A sort option can be used to influence which document gets replaced if more than one document is matched.
    - By default, it returns the original document.
    - If the option of {returnNewDocument: true} is set, the newly added document will be returned.
    - Field projection can be used to include only specific fields in the document returned in the response.

Modify Fields

- The replace operation should only be used when all or most of the fields are being modified. To modify one or only a few fields of a document, MongoDB provides the update command.
- **Updating a Document with updateOne()**
  - To update the fields of a single document in a collection.
  - Syntax: db.collection.updateOne(<query conditions>, <update expression>, <options>)
  - The output is a document that contains the following fields :
    - "acknowledged" : true indicates that the update was performed and confirmed.
    - "matchedCount" : 1 shows the number of documents found and chosen for the update (1 in this case.)
    - "modifiedCount" : 1 refers to the number of documents modified (1 in this case.)

Modify Fields

- **Modifying More Than One Field**
  - The $set operator that we used to update a field of a document can also be used to modify multiple fields of a document.
  - To modify more than one field, the update expression can contain more than one field and value pair.
- **Multiple Documents Matching a Condition**
  - If the given query condition matches more than one document, only the first document will be modified.
- **Upsert with updateOne()**
  - When upsert based updates are executed, the document will be updated if it is found; however, if the document is not found, a new document is created inside the collection. Similar to the replace operations, updateOne() also supports upserts with an additional flag in the command.

Updating Documents

- **Updating a Document with findOneAndUpdate():**
  - MongoDB also provides the findOneAndUpdate() function, which is capable of doing everything that updateOne() does with a few additional features.
  - **Syntax:** db.collection.findOneAndUpdate (<query condition> ,<update expression> ,<options>)
  - The findOneAndUpdate() function does not return the query stats, such as how many records were matched and how many records were modified. Instead, it returns the document in its old state. This is the difference it has with the update().
  - For returning the new document in response, need to pass additional boolean parameter returnNewDocument.
- **Sorting to Find a Document**:
  - We have covered two update functions, and both are capable of updating a single document at a time. If more than one document is matched by the given query condition, the first document will be chosen for modification.
  - The findOneAndUpdate() function provides an additional option to sort the matching documents in a specific order.

Updating Documents

- **Updating Multiple Documents with updateMany()**
  - Similar to updateOne(), the updateMany() function takes two mandatory arguments. The first argument is the query condition, and the second is the update expression.
  - **Syntax:** db.collection.updateMany (<query condition> ,<update expression> ,<options>)
  - The following are a few important points about the update operations and are applicable to all three functions:
    - None of the update functions allows you to change the _id field.
    - The order of the fields in a document is always maintained, except when the update includes renaming a field. However, the _id field will always appear first. (We will cover renaming fields in the next section).
    - Update operations are atomic on a single document. A document cannot be modified until another process has finished updating it.
    - All of the update functions support upsert. To execute an upsert command, upsert : true needs to be passed as an option.

Update Operators

- **Set ($set):** The operator takes a document that contains pairs of field names and their new values. If the given field is not already present, it will be created.
- **Increment ($inc):** It is used to increment the value of a numeric field by a specific number. The operator accepts a document containing pairs of a field name and a number. Given a positive number, the value of the field will be incremented and if a negative number is provided, the value will be decremented.
- **Multiply ($mul):** It is used to multiply the value of a numeric field by the given number. The operator accepts a document containing pairs of field names and numbers and can only be used on numeric fields.
- **Rename ($rename):** It is used to rename fields. The operator accepts a document containing pairs of field names and their new names. If the field is not already present in the document, the operator ignores it and does nothing. The provided field and its new name must be different. If they're the same, the operation fails with an error. If a document already contains a field with the provided new name, the existing field will be removed.
- **Current Date ($currentDate):** The operator $currentDate is used to set the value of a given field as the current date or timestamp. If the field is not present already, it will be created with the current date or timestamp value. Providing a field name with a value of true will insert the current date as a Date. Alternatively, a $type operator can be used to explicitly specify the value as a date or timestamp:

Update Operators

- **Removing Fields ($unset):** The $unset operator removes given fields from a document. The operator accepts a document containing pairs of field names and values and removes all the given fields from the matched document. As the provided fields are being removed, their specified values have no impact.
- **Setting When Inserted ($setOnInsert):** The operator $setOnInsert is similar to $set; however, it only sets the given fields when an insert happens during an upsert operation. It has no impact when the upsert operation results in the update of existing documents.

Summary

- In this session, you learned about:
  - The creation of documents in a collection.
  - During an insert operation, MongoDB creates the underlying collection if it does not exist, and autogenerates an _id field if the document does not have one already.
  - Covered various functions provided by MongoDB to delete and replace one or more documents in a collection, as well as the concept of upsert, its benefits, its support in MongoDB, and how an upsert operation differs from delete and insert.
  - Learned how to add, update, rename, or remove fields in MongoDB documents using various functions and operators.
- In the next chapter, we will execute some complex update commands using the aggregation pipeline support that was added in MongoDB 4.2, and learn how to modify the elements in an array field