Insert, Update and Delete Documents in MongoDB

In this session, you will learn

- MongoDB Data Types
- Inserting documents
- Updating existing documents
- Removing documents from a collection

MongoDB Data Types

String	Most commonly used data type to store the data
Integer	This type is used to store a numerical value.
Boolean	This type is used to store a boolean (true/ false) value.
Double	This type is used to store floating point values.
Object	This datatype is used for embedded documents.
Null	This type is used to store a Null value.
Date	This datatype is used to store the current date or time in UNIX time format.
Object ID	This datatype is used to store the document's ID.
Binary Data	This datatype is used to store binary data.
Regular Expression	This datatype is used to store regular expression.

MongoDB : Inserting Document

- Inserting a document with a field name new to the collection is inherently supported by the BSON model.

Syntax to insert one document

```
db.<collection>.insert({<field>:<value>})
```

Example:

```
C:\> db.Student.insert({"Student_id":9,"Name":"Charlie",
"Dob":ISODate("1995-11-05"),"Address" : [
{City:"Mangalore",Country:"India"}, {City:"cbe",Country:"India"}],
"Contact_no":["9632587410","7896541230"],"Gender":"M"})
                                                    // WriteResult({ "nInserted" : 1 })
```

MongoDB : Inserting Document

- Syntax To insert one document using insertOne() method

db.collection.insertOne(<document>, { writeConcern: <document>})

Example:

```
C:\> db.Student.insertOne({
"Student_id": 1,"Name": "John", "Dob":ISODate("1995-10-03"),
"Address":{City:"Jaipur",Country:"India"}, "Contact_no":"9874234681",
"Gender":"M"
})
                                        // WriteResult({ "nInserted" : 1 })
```

MongoDB : Inserting Document

- To insert multiple documents into a collection using insertMany() method.

Syntax:

```
db.<collection>.insertMany(
  [ <document 1> , <document 2>, ... ],
  {
    writeConcern: <document>,
    ordered: <boolean>
  }
)
```

- insertMany() inserts each document in the array into the collection.

MongoDB : Inserting Document

Example:

```
C:\> db.Student.insertMany([
{
"Student_Id": 2,"Name": "Nick",
"Dob":ISODate("1992-10-
02"),"Address":{City:"Durban",Country:"South Africa"},
"Contact_no":"7874234681", "Gender":"M"},
{
"Student_Id": 3,"Name": "Mark",
"Dob":ISODate("1999-11-02"),
"Address":{City:"Ranchi",Country:"India"},
"Contact_no":"8874234681", "Gender":"M"}
])                                    // WriteResult({ "nInserted" : 2 })
```

The Makeup of the _id field

- The ObjectId class is the default primary key for a MongoDB document and is usually found in the _id field in an inserted document.
- An ObjectId is a 12 byte binary BSON type that contain any 12 bytes.
- The 12 byte ObjectId value consists of
  - 4-byte value representing the seconds since the Unix epoch
  - 5-byte random value, and
  - 3-byte counter, starting with a random value

Cont'd...

- _id is the primary key on elements in a collection and using this records can be differentiated by default.

- All the documents in MongoDB must have a populated _id field. If a document has not been assigned an _id value, MongoDB will automatically generate one.

- Lookups specifying { _id: <someval> } refer to the _id index as their guide.

Generate a New ObjectId

- To generate a new ObjectId, use ObjectId() with no argument

Example:

```
x = ObjectId()
```

the value of x would be

```
ObjectId("507f1f77bcf86cd799439011")
```

MongoDB : Updating Document

- Once a document is stored in the database, it can be modified using the update method

- Update method takes two parameters:
  query document
  modifier document

- Updates are atomic

- Updates can safely be sent in rapid-fire succession without any documents being corrupted.

MongoDB : Updating Document

Syntax:

```
db.<collection>.update(
{<field1>:<value1>},                  //all docs in which field = value
{$set: {<field2>:<value2>}},          //set field to value
{multi:true} )                        //update multiple docs
```

- The **$set** operator replaces the value of a field with the specified value.
- If multiple field-value pairs are specified, $set will update or create each field.

Example 1 : Updating Document using $set

Example:

For the document matching the Student_Id equal to 3, the following operation uses the $set operator to update the value of the City field to 'Durban'.

// Before Updation

```
// { "_id" : ObjectId("5c32b8c651683d7a09d1168e"), "Student_Id" : 3, "Name" : "Mark", "Dob" : ISODate("1999-11-02T00:00:00Z"), "Address" : { "City" : "Ranchi", "Country" : "India" }, "Contact_no" : "8874234681", "Gender" : "M" }
```

```
C:\> db.Student.update({"Student_Id": 3 },{ $set:{"Address.City":"Durban"}})
```

//After Updation

```
// { "_id" : ObjectId("5c32b8c651683d7a09d1168e"), "Student_Id" : 3, "Name" : "Mark", "Dob" : ISODate("1999-11-02T00:00:00Z"), "Address" : { "City" : "Durban", "Country" : "India" }, "Contact_no" : "8874234681", "Gender" : "M" }
```

Example 2 : Updating Document using Save

- **db.<collection>.save()** : updates an existing document or inserts a new document, depending on its document parameter.

Syntax:

```
db.collection.save(
  <document>,
  {
   writeConcern: <document>
  }
)
```

MongoDB : Deleting Document

- **remove()** method is used to remove a document from the collection.

- This method accepts two parameters
  deletion_criteria
  justOne flag

- Using remove()

  db.<collection name>.remove(deletion_criteria, justOne)

Example:

C:\> db. Student.remove()

MongoDB : Deleting Document

- The remove function optionally takes a query document as a parameter.

- To removes a single document from a collection using deleteOne()

  db.<collection name>.deleteOne()

- It deletes the first document that matches the filter

- Remove all records where field : value

Syntax:

db.<collection>.remove({<field>:<value>})

THANKS