Nikit Gokhe

Class – Comp D1

Roll No. 224024

GR No. 21810522

**ASSIGNMENT NO.07**

**Aim:**Design and Develop MongoDB Queries using CRUD operations.

**Theory:**

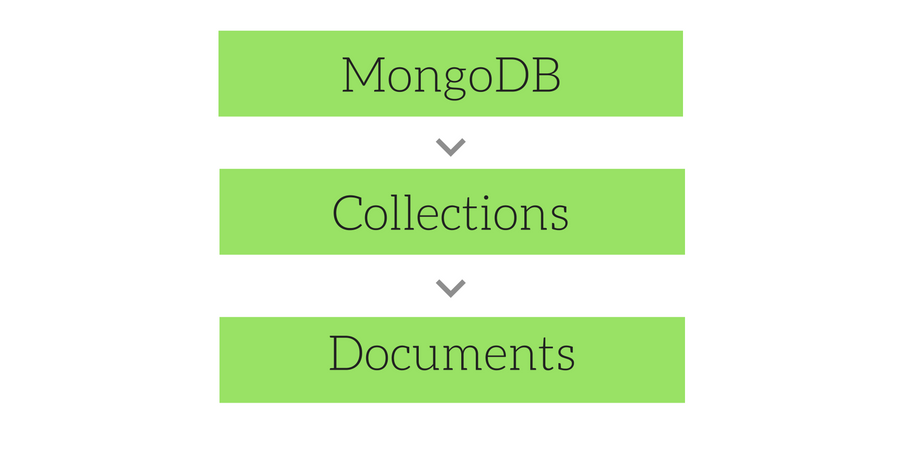
Introduction to MongoDB

MongoDB is a NoSQL database which stores the data in form of key-value pairs. It is an **Open Source**, **Document Database** which provides high performance and scalability along with data modelling and data management of huge sets of data in an enterprise application.

MongoDB also provides the feature of Auto-Scaling. Since, MongoDB is a cross platform database and can be installed across different platforms like Windows, Linux etc.

# Overview of MongoDB

MongoDB consists of a set of databases. Each database again consists of Collections. Data in MongoDB is stored in collections. The below figure depicts the typical database structure in MongoDB.



# Database in MongoDB

Database in MongoDB is nothing but a container for collections. We will learn how to create a new Database, drop a Database and how to use an existing Database in the coming lessons.

# Collections in MongoDB

Collection is nothing but a set of MongoDB documents. These documents are equivalent to the row of data in tables in RDBMS. But, collections in MongoDB do not relate to any set schema as compared to RDBMS. Collections are a way of storing related data. Being schemaless, any type of Document can be saved in a collection, although similarity is recommended for index efficiency.

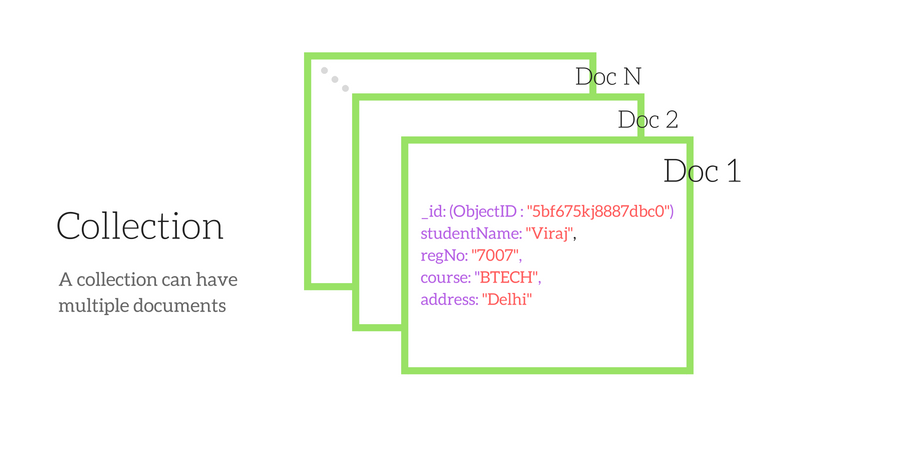
Document's can have a maximum size of 4MB.We can use namespace to logically group and nest collections. A collection is physically created as soon as the first document is created in it.There’s need to create multiple collections with different namespace, when we can keep any form or data in a single collection itself. It's because, MongoDB does not index attributes for totally unrelated documents. So it is advised to keep related data in collections.

# Document in MongoDB

Document in MongoDB is nothing but the set of key-value pairs. These documents will have dynamic schema which means that the documents in the same collection do not need to possess the same set of fields.

Since MongoDB is considered as a schema-less database, each collection can hold different type of objects. Every object in a collection is known as Document, which is represented in a JSON like (JavaScript Object Notation) structure(nothing but a list of key-value pair). Data is stored and queried in BSON, its binary representation of JSON-like data.

# Sample Data in MongoDB



**Code:**

//creating database

**>use currentdb**

//creating collection

**>db.createCollection(“Employee”)**

//Inserting data

**>db.Employee.insert({db.Employee.insert({**

**... name:"rahul",**

**... salary:20000,**

**... age:22**

**... })**

**db.Employee.insert({ name:"saurabh", salary:27000, age:26,phone:080980022 })**

WriteResult({ "nInserted" : 1 })

**>db.Employee.insert({ name:"virendar", salary:19000, age:25})**

WriteResult({ "nInserted" : 1 })

**>db.Employee.insert({ name:"gautam", salary:30000, age:28})**

WriteResult({ "nInserted" : 1 })

// Multiple inserts can be done using array

> **var Allemps = [ {name:"rahul",salary:20000,age:22}, { name:"saurabh", salary:27000, age:26,phone:080980022 }, { name:"virendar", salary:19000, age:25}, { name:"gautam", salary:30000, age:28} ]**

**>db.Employee.insert(Allemps)**

// Reading data

**>db.Employee.find()**

{ "\_id" : ObjectId("5e7511a736bd5168a0bd6a8d"), "name" : "rahul", "salary" : 20000, "age" : 22 }

{ "\_id" : ObjectId("5e7511c736bd5168a0bd6a8e"), "name" : "saurabh", "salary" : 27000, "age" : 26, "phone" : 80980022 }

{ "\_id" : ObjectId("5e7511f936bd5168a0bd6a8f"), "name" : "virendar", "salary" : 19000, "age" : 25 }

{ "\_id" : ObjectId("5e75122036bd5168a0bd6a90"), "name" : "gautam", "salary" : 30000, "age" : 28 }

//find all entries where name starts with "r" and age =22.

**>db.Employee.find({name:/^r/,age:22})**

{ "\_id" : ObjectId("5e7511a736bd5168a0bd6a8d"), "name" : "rahul", "salary" : 20000, "age" : 22 }

//update data

//change salary = 5000 in collection where name == ‘sachin’

**>db.Employee.update({name:"sachin"},{$set:{salary:22000}})**

WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

//Add new field  “profile” to all documents as “player”.

**>db.Employee.update({},{$set:{profile:"player"}},false,true)**

WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 4 })

**>db.Employee.remove({})**

// removes all

**>db.mytable.remove({ salary : 1000 })**

// removes all where salary == 1000

**>db.Employee.drop()**

//removes collection

**>db.dropDatabase()**

//drops current database

**Output:**

