INDEX

1.Introduction of MongoDB

- Mongo is a cloud service.
- MongoDB is a database provided by Mongo Cloud Service.
- MongoDB is designed to meet the demands of modern applications.
- MongoDB is open source.
- MongoDB is cross platform.
- MongoDB is document based database.
- It is known as “Document Data Model”.
- Every row act’s as a table. Every row is a document. [Schema Less]
- It is JavaScript based database.
- It is non-SQL database. [No more SQL queries]
- You can use all JavaScript based commands to manipulate the data.
- Data is like JSON.
- Unified experience – allows to run anywhere [any device, any OS]
- It uses “Distributed Systems Design” – You can index of any field.
- MongoDB is Easy, Fast, Flexible, Versatile.
- High Availability
- Scalability
- Portability

2.Issues with MongoDB

- Non-RDBMS
- Schema Less
- Problems in analysis and reporting.
- Data Predictions will be difficult to handle.

3.Where to Use MongoDB?

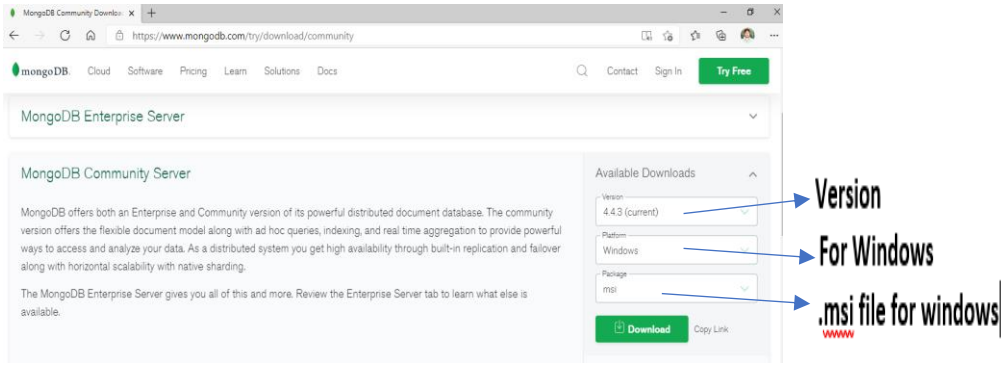
- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

4.Why Use MongoDB?

- Document Oriented Storage: Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

5.Setup and Install MongoDB

- Visit the following
  https://www.mongodb.com/try/download/community



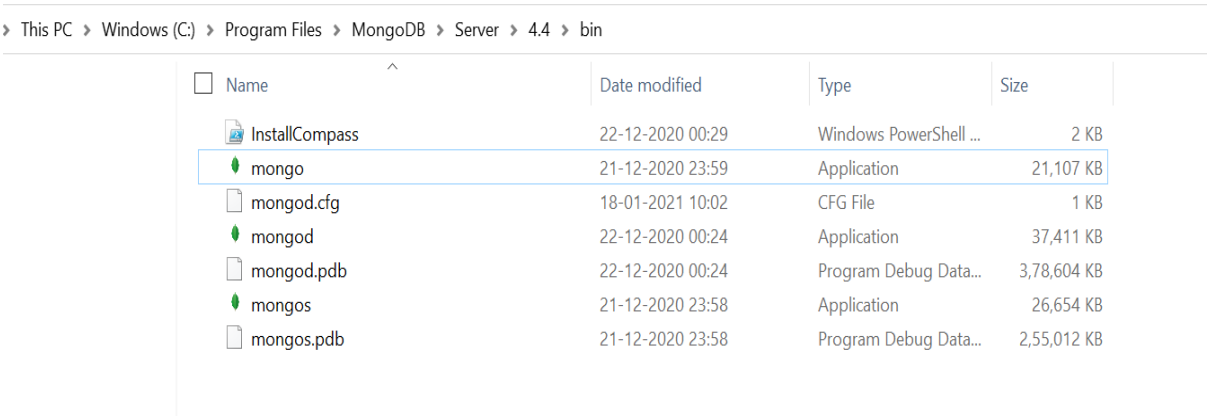
- Download the ".msi" for your windows / any OS
- Download latest available version [4.4]
- Install on your PC
- You can install "compass" GUI tool for MongoDB [Optional]

6.MongoDB Server

- Server is the location where MongoDB database is stored.
- You have to start "MongoDB server" on your PC.
  - Open "Services.msc" from windows run option
  - Right Click on "MongoDB Database Server"
  - Select "Start"
  - In Properties keep the start type as "Automatic"
- The default server location of MongoDB is
  mongodb://127.0.0.1:27017
  mongodb://localhost:27017
- You have to connect with MongoDB database Server from "MongoDB Client"

7.MongoDB Client

- Client provides a platform from where you can connect with database server and handle interaction with the database.
- Client provides a "shell" from where you have to interact with database server.
- MongoDB Client shell provides commands to interact with database.
- You have to switch to MongoDB client Shell.
- Shell is present in "Installation Folder".
- Double click on "mongo" application [.exe] to open the shell.



You can also login from command prompt:

C:\>

C:\> cd Program Files

C:\Program Files\> cd MongoDB

C:\Program Files\MongoDB> cd Server

C:\Program Files\MongoDB\Server> cd 4.4

C:\Program Files\MongoDB\Server\4.4> cd bin

C:\Program Files\MongoDB\Server\4.4\bin> mongo.exe

> shell commands here.

> exit

You can change the MongoDB default Port:

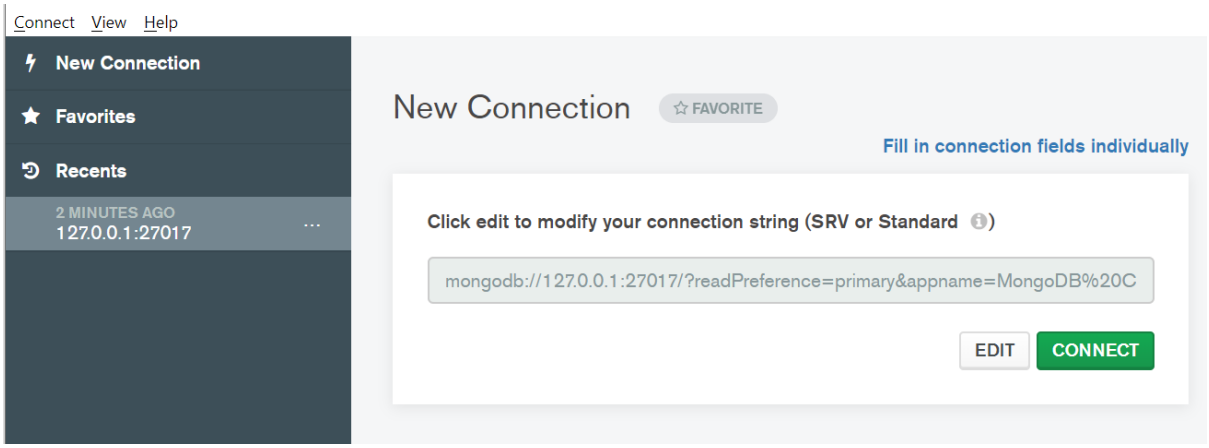
> mongo --port 2450

You can change the MongoDB default host:

> mongo --host mongodb.yoursite.com:2450

8.MongoDB Compass

- It provides an IDE [Integrated Development Environment]
- You can connect with server.
- Configure and Create Database
- Manipulate and Query data etc.
- **Open MongoDB Compass on your PC**



- **Define the connection string**
  mongodb://127.0.0.1:27017
- **Click "Connect"**

9.MongoDB Terminology

RDBMS	**MongoDB**
Database	Database
Tables	Collection
Record / Row	Document
Field/Column	Field
Join	Embedded Document

10.MongoDB Shell Commands

- MongoDB shell is case sensitive.
- MongoDB uses all JavaScript based commands.

Basic Commands:

show dbs	It can display the list of databases.
db	It shows the current active database.
use	It is used to switch into existing database or create a new database. Syntax: > use databaseName
show collections	To view the list of tables in database.

Ex:

> use demodb

Note:

- The database you create will not be displayed in list until or unless it is configured with a table. [collection].
- MongoDB commands are case sensitive.

11.Creating a new Database

- You can create and switch into database by using “use” command.
  Syntax:
  > use newDatabaseName
  Ex:
  > use demodb

```
---
> use demodb
switched to db demodb
>
```

12.Remove Existing Database

- You can remove existing database by using the method “dropDatabase()”.
  Syntax:
  > use demodb
  > db.dropDatabase()

```
> db.dropDatabase()
{ "dropped" : "demodb", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

13.Adding Collection to Database

- The database table is known as collection.
- To create a collection [table] you have use the method “createCollection()”
  Syntax:
  >db.createCollection(“tableName”, options)

```
> db.createCollection("tblcategories")
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
demodb  0.000GB
local   0.000GB
```

Note: Options for table are not mandatory.

Ex:

```
> db.createCollection("tblcategories")
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
demodb  0.000GB
local   0.000GB
> db.dropDatabase()
{ "dropped" : "demodb", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

- To create a collection, you can define a set of options provided by MongoDB
- Options are defined as JavaScript Object.
  Syntax:
  > db.createCollection(“tableName”, {options})

- The options used for creating a collection are:

Option	Type	Description
size	number	It defines the size in bytes, which are allocated for table in memory.
capped	boolean	It defines the capped collection for table, if you set to true then the old entries are overwritten when it reaches the max size.
max	number	The maximum number of documents allowed in a capped collection.
autoIndexId	boolean	Every collection in MongoDB is generated with "_id", which is an auto generated field for every table. You can ignore it by configuring "autoIndexId" to false. Note: It is deprecated from version 3.2.

```
> db.createCollection("tblcategories",{capped:true, size:2097152, max:10})

{ "ok" : 1 }
```

To View the list of collections in database you can use the command "show collections"

> show collections

14.MongoDB Data Types

Type	Alias
Double	"double"
String	"string"
Object	"object"
Array	"array"
Binary Data	"binData"
Undefined	"undefined"
ObjectId	"objectId"
Boolean	"bool"
Date	"date"
Null	"null"
Regular Expression	"regex"
Symbol	"symbol"
32-bit Integer	"int"
64-bit Integer	"long"
Decimal128	"decimal"

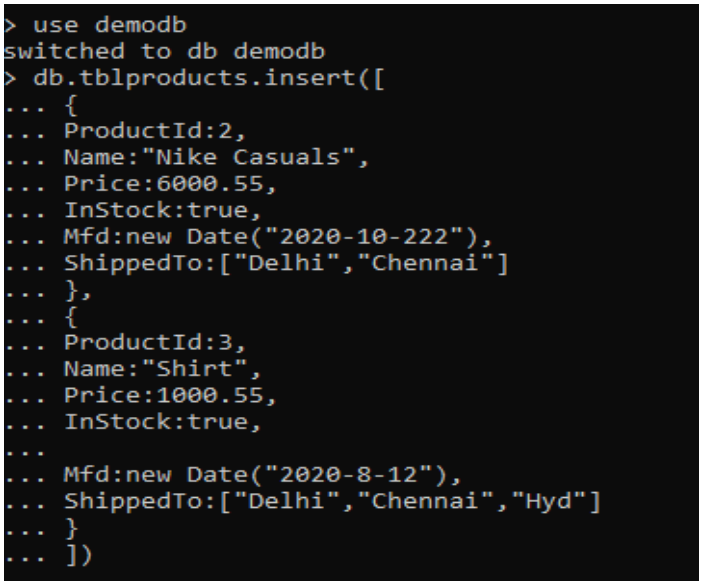
The data types are defined by using "$type". The types are required while querying the data.

Syntax:

db.tblcategories.find({"CategoryId": {$type: "number"}})

Ex: Inserting Records [Documents]

```
> db.tblproducts.insert([
 {
   ProductId:2,
   Name: "Nike Casuals",
   Price: 6000.55,
   InStock: true,
   Mfd: new Date("2020-10-22"),
   ShippedTo: ["Delhi", "Chennai"]
 },
{
   ProductId:3,
   Name: "Shirt",
   Price: 1000.55,
   InStock: true,
   Mfd: new Date("2020-8-12"),
   ShippedTo: ["Delhi", "Chennai", "Hyd"]
 }
])
```



15.MongoDB CRUD Operations

- Create C
- Read R
- Update U
- Delete D

You use the CRUD operations to manipulate the documents [Records].

Create a new Document / Insert a new Record into Collection

- To insert a new document into collection we can uses the methods
  - insert() - Insert one or many
  - insertOne() - Allows to insert only one document
  - insertMany() - Allows to insert many documents.
- Every document is considered as JSON object.
- It comprises of Property and Value.
- The data type for value will be the same as JavaScript data types.

```
> db.tblcategories.insertOne({CategoryId:1, CategoryName:"Electronics"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6007b288869092ef63af5080")
}
>
```

```
> db.tblcategories.insertMany([{CategoryId:2, CategoryName:"Fashion"},{Cat
egoryId:3, CategoryName:"Footwear"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6007b310869092ef63af5081"),
                ObjectId("6007b310869092ef63af5082")
        ]
}
>
```

Read Operations

- The read operations are used to retrieve the documents from a collection.
- You can query a collection for documents.
- MongoDB provides “find()” method for querying.
  > db.collection.find()

- MongoDB provides “pretty()” method to display the documents in a friendly format.

  > db.collection.find().pretty()

Syntax:

> db.tblproducts.find().pretty()

```
> db.tblproducts.find().pretty()
{
        "_id" : ObjectId("6008fc2c99019b261c6a2b60"),
        "ProductId" : 1,
        "Name" : "JBL Speaker",
        "Price" : 4500.55,
        "InStock" : true,
        "Mfd" : ISODate("2020-02-10T00:00:00Z"),
        "ShippedTo" : [
                "Delhi",
                "Hyd"
        ]
}
{
```

- **"find()" method provides query filters or criteria that identify the documents based on specified condition and return only the documents that match the criteria.**

Syntax:

> db.tblproducts.find({queryFilters})

Ex:

MongoDB: > db.tblproducts.find({})

SQL : Select * from tblproducts

Update Operations

- It includes modifying existing documents in a collection.
- MongoDB provides the following methods for update
  - updateOne()
  - updateMany()
  - replaceOne()

Syntax:

db.collection.updateOne({ UpdateOperator : { Field : value} })

- MongoDB update operator is "$set"
- MongoDB provides various operators for updating

Operator	Description
$currentDate	It is used to set the value as current date.

$inc	Increments the value of the field by specified amount.
$min	It will update only when the filed value is below the specified value.
$max	It will update only when the field value is above the specified value.
$rename	It is used to rename the field.
$set	It is used to change the filed value.
$unset	It is used to remove any specific field from document.

- MongoDB provides operators for modifying the array fields

Operator	Description
$pop	It removes the first or last item of an array
$push	It adds a new item into array.
$pullAll	Removes all matching value from array.
$position	It is used to add a new element at specific position in array.
$slice	It is used to limit the size of updated arrays.
$sort	It is used to arrange documents in order.

Note: MongoDB supports Indexing.

Syntax:

db.collection.updateOne(FilterQuery, UpdateQuery)

```
> db.tblproducts.updateOne({Name:"Shirt"},{ $set: {Price: 1500.44}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.tblproducts.updateMany({Price: {$gt: 5000}}, {$set:{Price:8000.55}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

Delete Operation

- It is the process of removing record from the collection.
- There are two methods for deleting the record
  - deleteOne()
  - deleteMany()

Syntax:

db.collectionName.deleteOne({filter})

db.tblproduct.deleteOne({Product:3})

db.tblproduct.deleteMany({Category:”Electronics”})

```
> db.tblproducts.find().pretty()
{
        "_id" : ObjectId("60170bfef38360cadba7ff7d"),
        "ProductId" : 2,
        "Name" : "Nike Casuals",
        "Price" : 6000.55,
        "InStock" : true,
        "Mfd" : ISODate("1970-01-01T00:00:00Z"),
        "ShippedTo" : [
                "Delhi",
                "Chennai"
        ]
}
{
        "_id" : ObjectId("60170bfef38360cadba7ff7e"),
        "ProductId" : 3,
        "Name" : "Shirt",
        "Price" : 1000.55,
        "InStock" : true,
        "Mfd" : ISODate("2020-08-12T00:00:00Z"),
        "ShippedTo" : [
                "Delhi",
                "Chennai",
                "Hyd"
        ]
}
> db.tblproducts.deleteOne({ProductId:3})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

16.Conditions in MongoDB Query

- Equality Condition
  - It can access and return only the documents that match the given value.
  - It is similar to “==” operator of JavaScript
  - SQL uses : “Select * from tblproducts where Name=”Nike Casuals”

Syntax:

find({FiledName:Value}) - Equal Condition

> db.tblproducts.find({Name:”Nike Casuals”}).pretty()

```
> db.tblproducts.find({Price:6000.55}).pretty()
{
        "_id" : ObjectId("6008fdd899019b261c6a2b61"),
        "ProductId" : 2,
        "Name" : "Nike Casuals",
        "Price" : 6000.55,
        "InStock" : true,
        "Mfd" : ISODate("2020-10-22T00:00:00Z"),
        "ShippedTo" : [
                "Delhi",
                "Chennai"
        ]
}
>
```

- Condition using Query Operators
  - MongoDB allows condition using query operators like
    - $or
    - $in

Syntax:

> db.tblproducts.find({Name:{ $in: ["Shirt", "Nike Casuals"]}}).pretty()

SQL: Select * from tblproducts where Name in (“Shirt”, “Nike Casuals”)

- Conditions are defined by using “literals” not operators
  - $lt Less than
  - $gt Greater than
  - $lte Less than or Equal
  - $gte Greater than or Equal
  - $ne Not Equal

Ex:

```
> db.tblproducts.find({Price:{$gt:5000}}).pretty()
{
        "_id" : ObjectId("6008fdd899019b261c6a2b61"),
        "ProductId" : 2,
        "Name" : "Nike Casuals",
        "Price" : 6000.55,
        "InStock" : true,
        "Mfd" : ISODate("2020-10-22T00:00:00Z"),
        "ShippedTo" : [
                "Delhi",
                "Chennai"
        ]
}
>
```

Specify Query using “OR”

> db.tblproducts.find({$or: [{Name:"Shirt"}, {Price:{$lt:6000}}]}).pretty()

[Select * from tblproducts WHERE Name=”Shirt” OR Price < 6000]

Using LIKE in query

- You have to use regular expression for verifying the value in specific string.
- It is similar to SQL LIKE

> db.tblproducts.find({Name:/^S/}).pretty()

Select * from tblproducts where Name LIKE “S%”

Querying Array Elements

- It requires the following operators
    - $all
    - $elemMatch

Ex:

> db.tblproducts.find({ShippedTo:{$all:["Hyd","Delhi"]}}).pretty()

> db.tblproducts.find({ShippedTo: “Hyd”}).pretty()

TASK:

- Change the field “InStock” to “StockStatus”
- Add a new filed into all records “Category”
- Update Category according to product [Nike Casuals – Category:Footwear, TV – Category-Electronics]