PRACTICAL NO :03

MONGODB

Unit Structure :

3.0 OBJECTIVE

After going through this unit, you will be able to:

- Have a solid understanding of basics of MongoDB
- Learn to run queries against a MongoDB instance
- Manipulate and retrieve data
- Different techniques and algorithms used in association rules.
- Understand the difference between Data Mining and KDD in Databases.

3.1 INTRODUCTION

MongoDB is a free, open-source document-oriented database that can hold a big amount of data while also allowing you to deal with it quickly. Because MongoDB does not store and retrieve data in the form of tables, it is classified as a NoSQL (Not Only SQL) database.

MongoDB is a database developed and managed by MongoDB.Inc, which was first released in February 2009 under the SSPL (Server Side Public License). It also includes certified driver support for all major programming languages, including C, C++, C#, etc. Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, and Mongoid are examples of programming languages. As a result, any of these languages can be used to develop an application. MongoDB is now used by a large number of firms, including Facebook and Google.

3.1.1 How it works

Now we'll see how things really work behind the scenes. As we all know, MongoDB is a database server that stores data in databases. In other words, the MongoDB environment provides you with a server on which you may install MongoDB and then construct several databases.

The data is saved in collections and documents thanks to the NoSQL database. As a result, the database, collection, and documents are linked as follows:

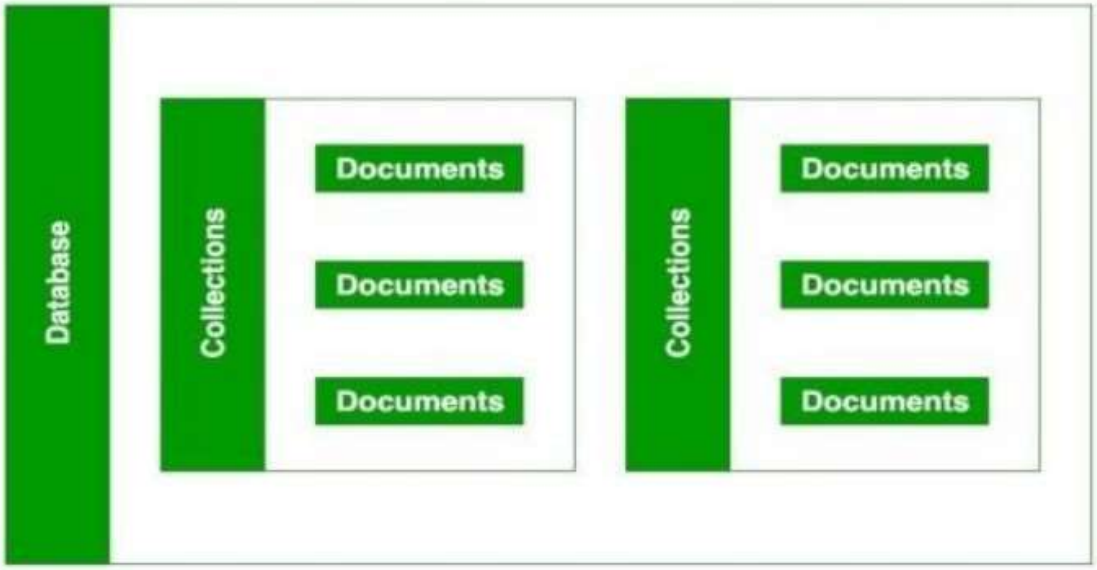


Figure: 4.1 Mongo DB Structure

- Collections exist in the MongoDB database, exactly as tables do in the MYSQL database. You have the option of creating several databases and collections.

- We now have papers within the collection. These documents hold the data we want to store in the MongoDB database, and a single collection can contain several documents. You are schema-less, which means that each document does not have to be identical to the others.

- The fields are used to build the documents. Fields in documents are key-value pairs, similar to columns in a relational database. The fields' values can be of any BSON data type, such as double, string, Boolean, and so on.

- The data in MongoDB is saved in the form of BSON documents. BSON stands for Binary representation of JSON documents in thisccontext. To put it another way, the MongoDB server translates JSON data into a binary format called as BSON, which can then be stored and searched more effectively.

- It is possible to store nested data in MongoDB documents. When opposed to SQL, layering data allows you to construct complicated relationships between data and store them in the same document, making data working and retrieval extremely efficient. To retrieve data from tables 1 and 2, you must use complicated SQL joins. The BSON document can be up to 16MB in size. In MongoDB users are allowed to run multiple databases.

3.1.2 How MongoDB is different from RDBMS

Some major differences between MongoDB and RDBMS are as follows

Mongo DB	RDBMS
It is a non-relational and document-oriented database.	It is a relational database
It is suitable for hierarchical data storage.	It is not suitable for hierarchical data storage.
It has a dynamic schema.	It has a predefined schema.
It centers on the CAP theorem (Consistency, Availability, and Partition tolerance).	It centers on ACID properties (Atomicity, Consistency, Isolation, and Durability).
In terms of performance, it is much faster than RDBMS.	In terms of performance, it is slower than MongoDB.

3.1.3 Features of MongoDB

- **Schema-less Database**: MongoDB has a fantastic feature called schema-less database. A schema-less database means that different types of documents can be stored in the same collection. In other words, a single collection in the MongoDB database can hold numerous documents, each of which may have a different amount of fields, content, and size. In contrast to relational databases, it is not required that one document be similar to another. MongoDB gives databases a lot of flexibility because to this amazing feature.
- **Document Oriented**: Unlike RDBMS, MongoDB stores all data in documents rather than tables. Data is kept in fields (key-value pair) rather than rows and columns in these documents, making the data far more flexible than in RDBMS. Each document also has its own unique object id.

- **Indexing**: Every field in the documents in the MongoDB database is indexed with primary and secondary indices, making it easier and faster to get or search data from the pool of data. If the data isn't indexed, the database will have to search each document individually for the query, which takes a long time and is inefficient.
- **Scalability**: Sharding is used by MongoDB to achieve horizontal scalability. Sharding is a method of distributing data across numerous servers. A significant quantity of data is partitioned into data chunks using the shard key, and these data pieces are evenly dispersed across shards located on multiple physical servers. It can also be used to add additional machines to an existing database.
- **Replication**: MongoDB delivers high availability and redundancy through replication, which produces multiple copies of data and sends them to a different server so that if one fails, the data may be retrieved from another.
- **Aggregation**: It enables you to conduct operations on grouped data and obtain a single or computed result. It's analogous to the GROUPBY clause in SQL. Aggregation pipelines map-reduce functions, and single-purpose aggregation methods are among the three types of aggregations available.
- **High Performance**: Due to characteristics such as scalability, indexing, replication, and others, MongoDB has a very high performance and data persistence when compared to other databases.

3.1.4 Advantages of MongoDB

- It's a NoSQL database with no schema. When working with MongoDB, you don't have to worry about designing the database schema.
- The operation of joining is not supported.
- It gives the fields in the papers a lot more flexibility.
- It contains a variety of data.
- It offers excellent availability, scalability, and performance.
- It effectively supports geospatial.
- The data is stored in BSON documents in this document-oriented database.
- It also allows ACID transitions between multiple documents (string from MongoDB 4.0).
- There is no need for SQL injection.
- It is simple to integrate with Hadoop Big Data

3.1.5 Disadvantages of MongoDB

- It stores data in a large amount of memory.
- You may not keep more than 16MB of data in your documents.
- The nesting of data in BSON is likewise limited; you can only nest data up to 100 levels

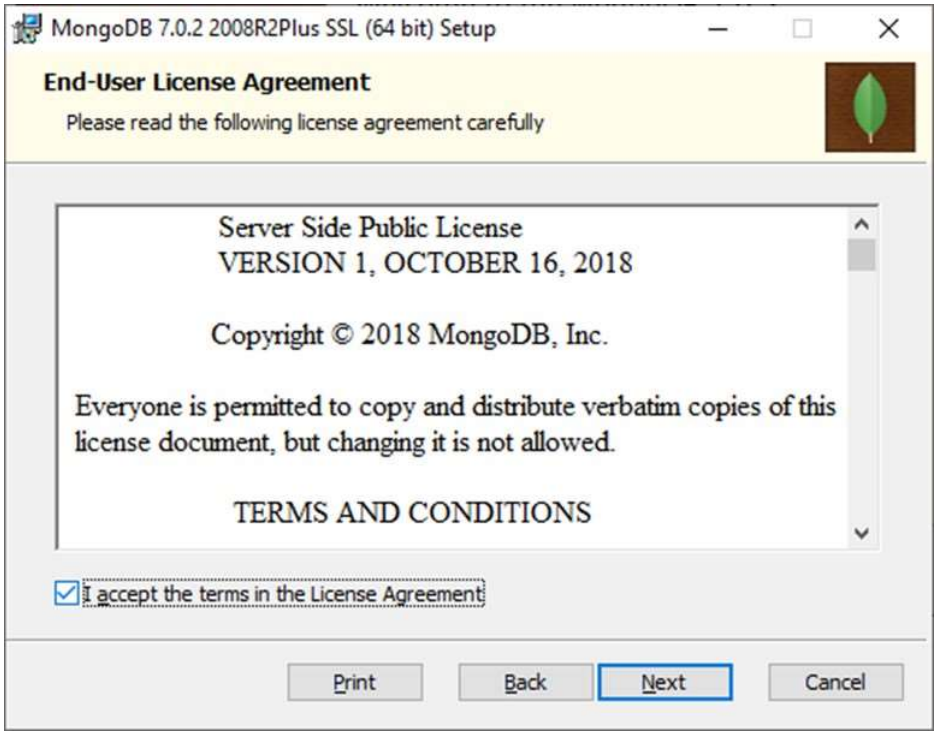
3.2 MONGODB ENVIRONMENT

To get started with MongoDB, you have to install it in your system. You need to find and download the latest version of MongoDB, which will be compatible with your computer system. You can use this (http://www.mongodb.org/downloads) link and follow the instructions to install MongoDB on your PC.

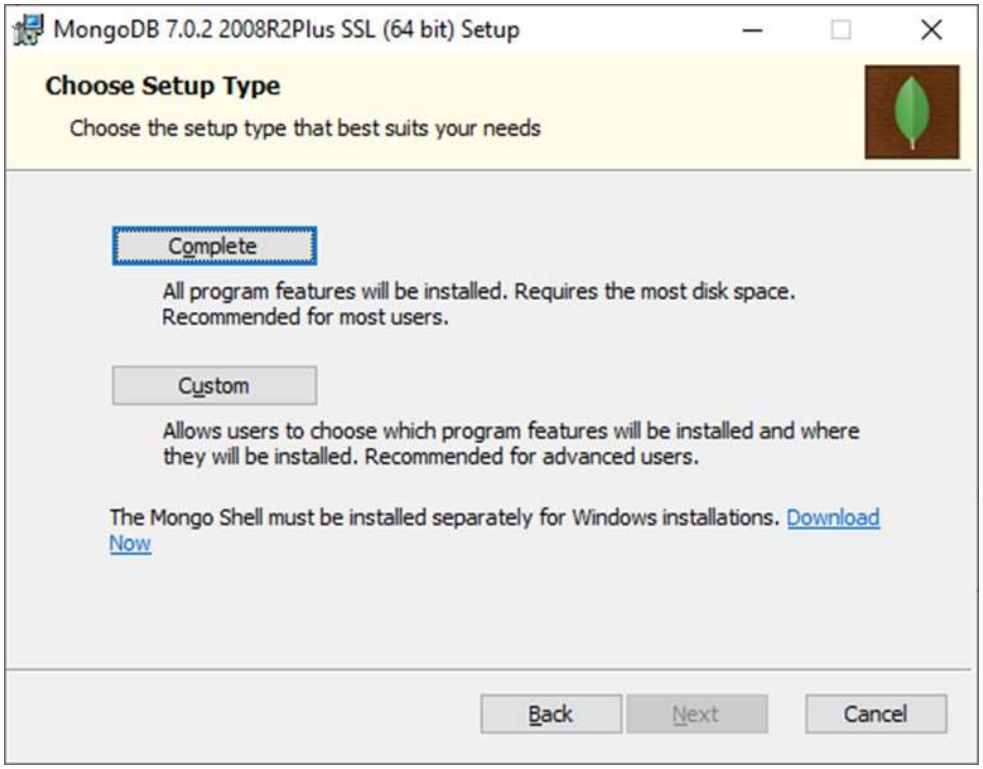
The process of setting up MongoDB in different operating systems is also different, here various installation steps have been mentioned and according to your convenience, you can select it and follow it.

3.2.1 Install MongoDB in Windows The website of MongoDB provides all the installation instructions, and MongoDB is supported by Windows, Linux as well as Mac OS. It is to be noted that, MongoDB will not run in Windows XP; so you need to install higher versions of windows to use this database. Once you visit the link (http://www.mongodb.org/downloads), click the download button. Or use MongoDb Compass

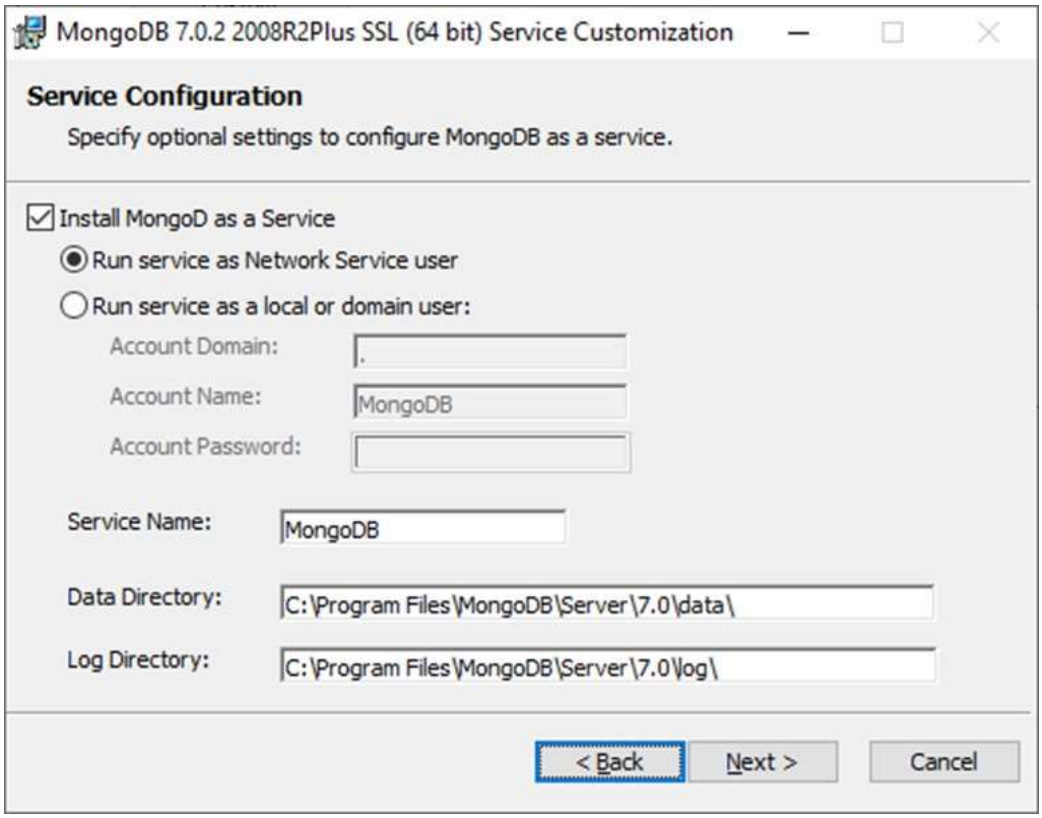
1. Once the download is complete, double click this setup file to install it. Follow the steps:
2. Click on Next



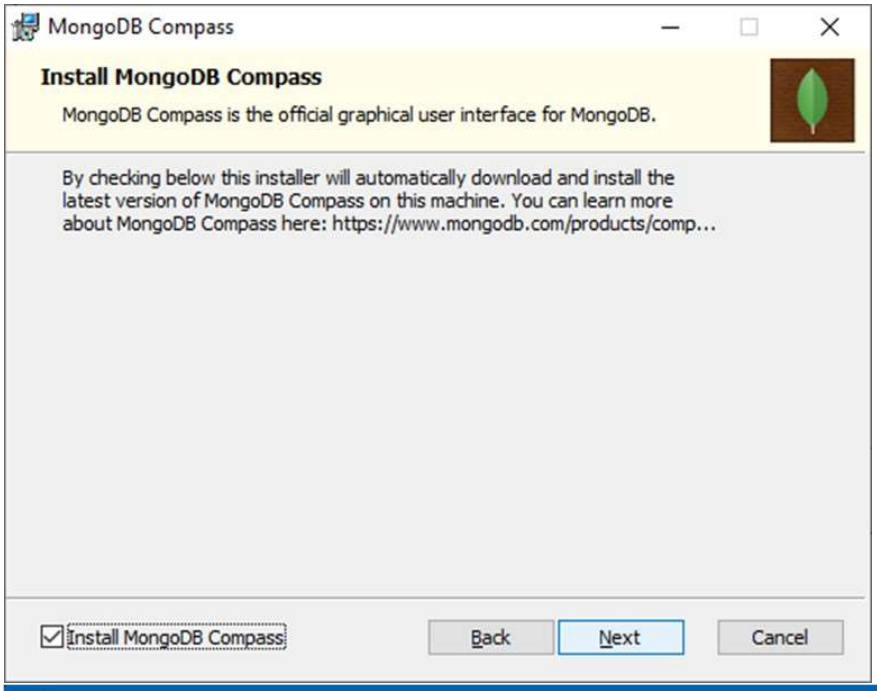
3. Now, choose Complete to install MongoDB completely.

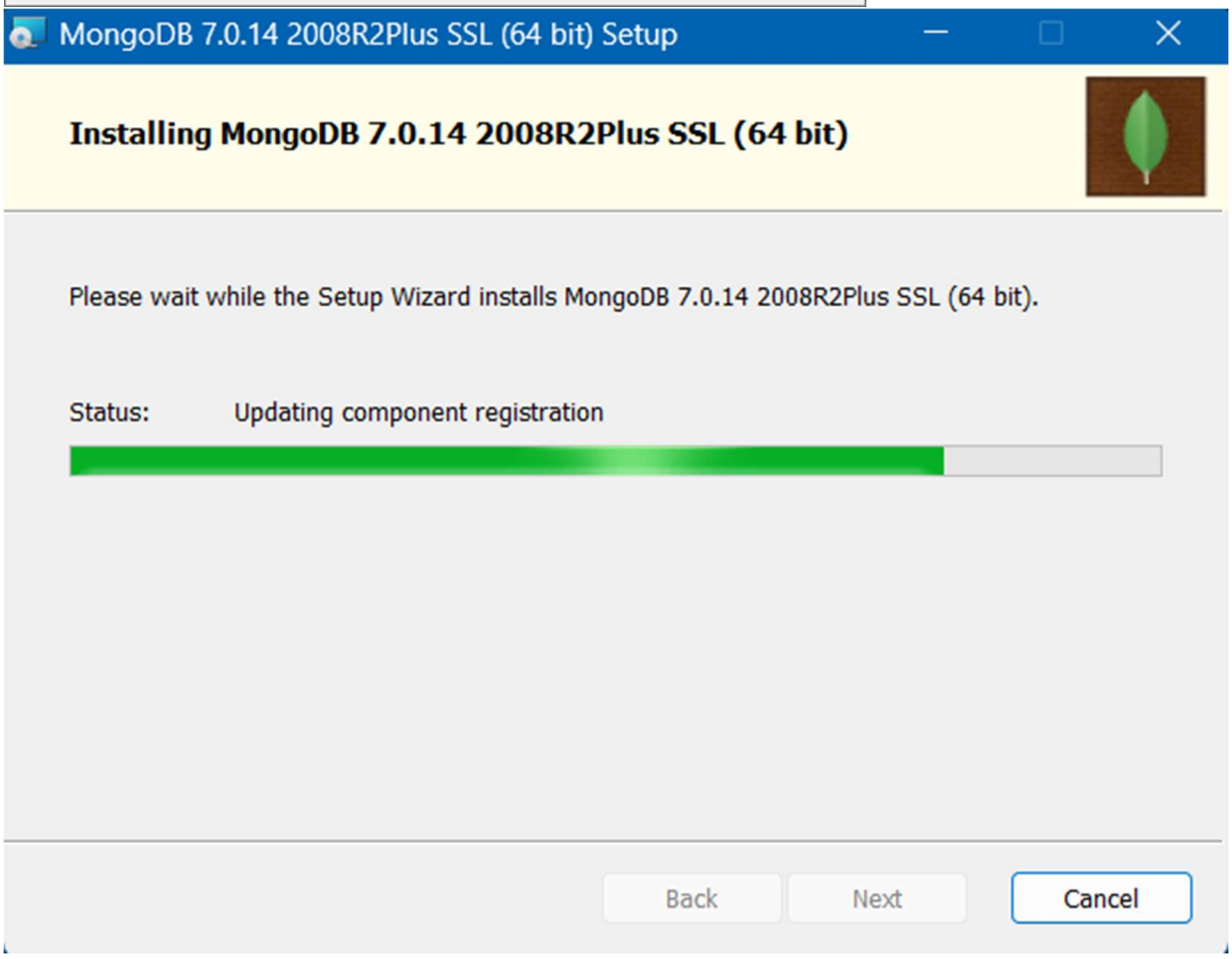


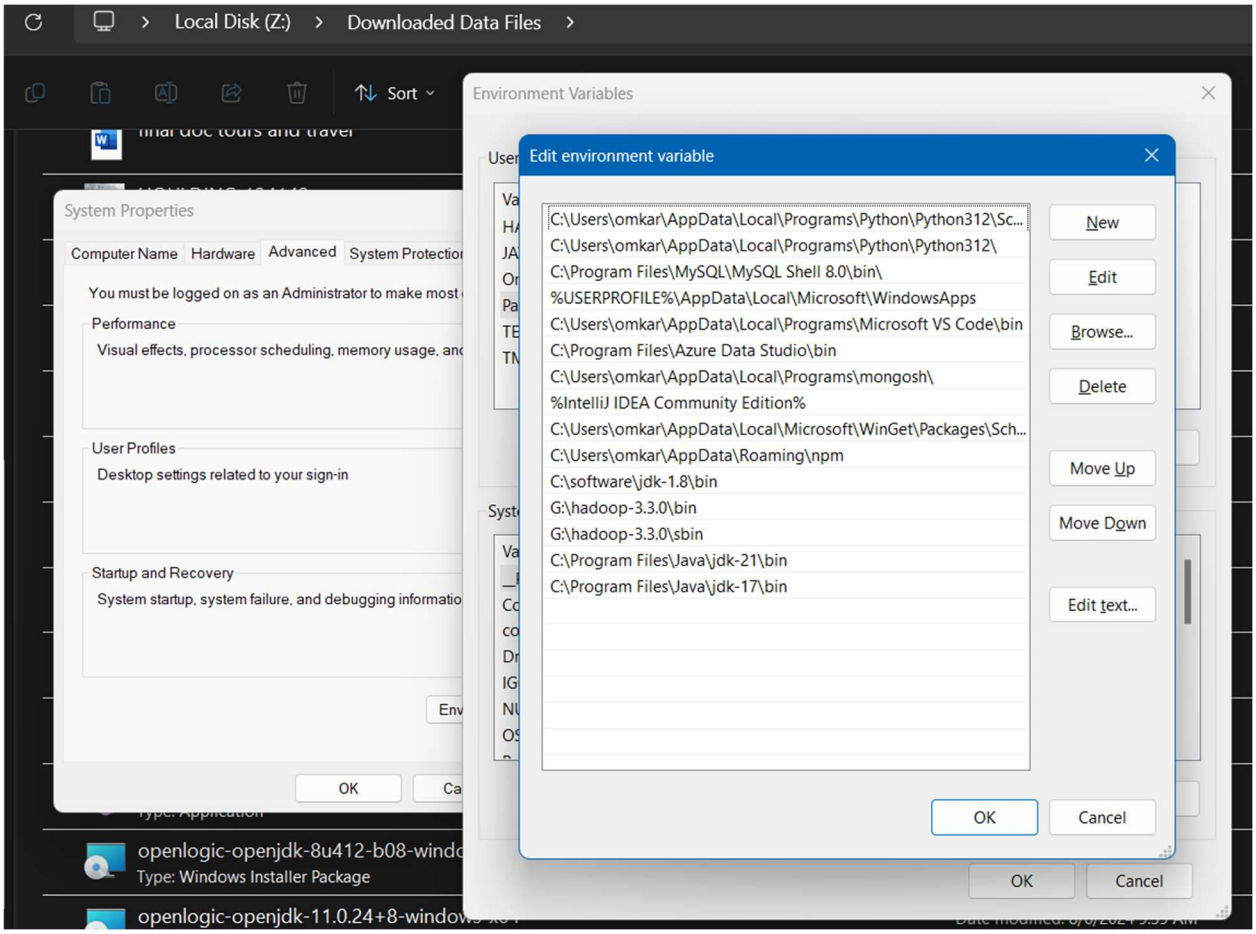
4. Then, select the radio button "Run services as Network service user."



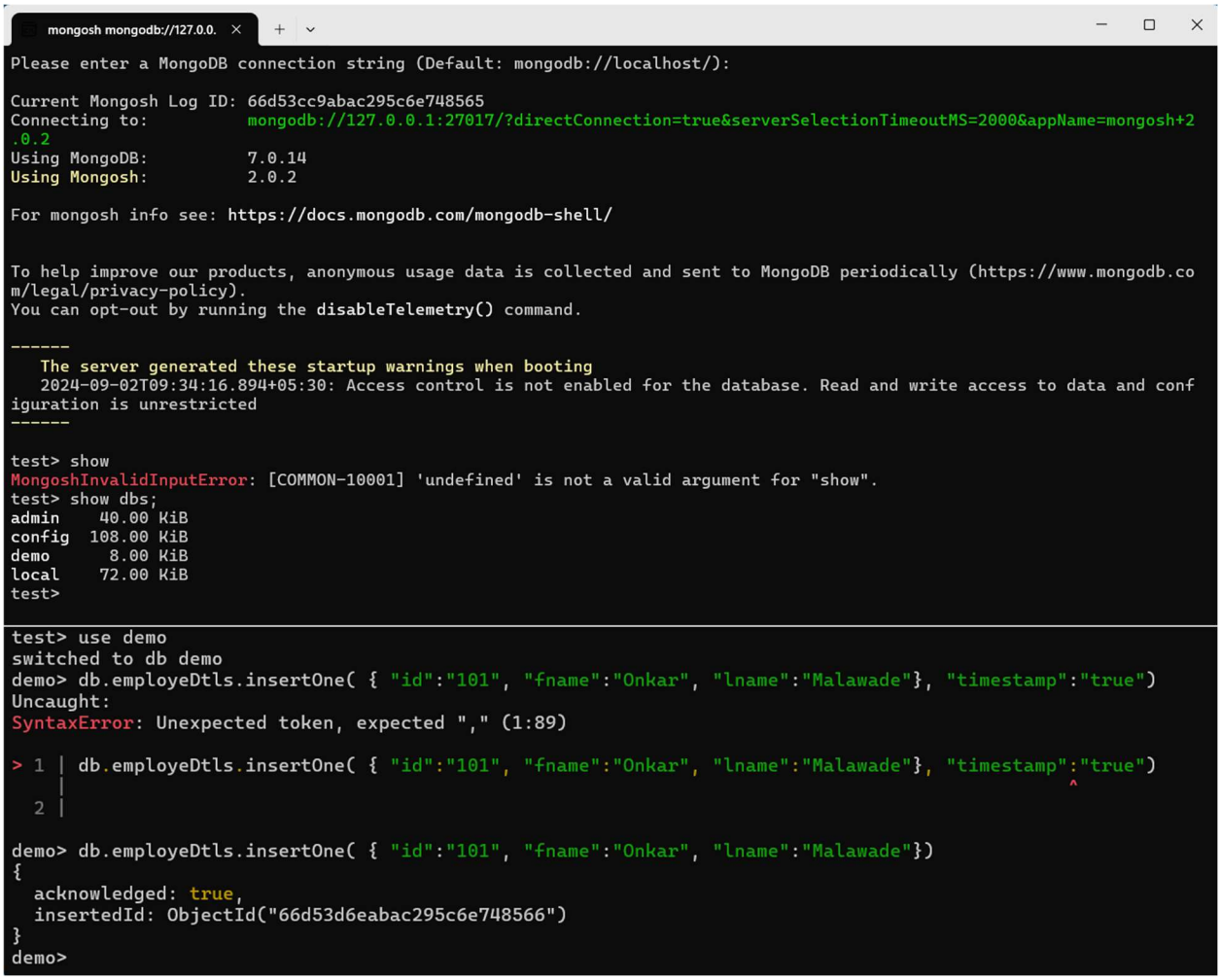
5. The setup system will also prompt you to install MongoDB Compass, which is MongoDB's official graphical user interface (GUI). You can tick the checkbox to install that as well.






Local Disk (Z:)
Downloaded Data Files
Sort
Environment Variables
Edit environment variable
C:\Users\omkar\AppData\Local\Programs\Python\Python312\Sc...
C:\Users\omkar\AppData\Local\Programs\Python\Python312\
C:\Program Files\MySQL\MySQL Shell 8.0\bin\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\omkar\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\Azure Data Studio\bin
C:\Users\omkar\AppData\Local\Programs\mongosh\
%IntelliJ IDEA Community Edition%
C:\Users\omkar\AppData\Local\Microsoft\WinGet\Packages\Sch...
C:\Users\omkar\AppData\Roaming\npm
C:\software\jdk-1.8\bin
G:\hadoop-3.3.0\bin
G:\hadoop-3.3.0\sbin
C:\Program Files\Java\jdk-21\bin
C:\Program Files\Java\jdk-17\bin
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel
System Properties
Computer Name
Hardware
Advanced
System Protectio
You must be logged on as an Administrator to make most
Performance
Visual effects, processor scheduling, memory usage, an
User Profiles
Desktop settings related to your sign-in
Startup and Recovery
System startup, system failure, and debugging informatio
OK
openlogic-openjdk-8u412-b08-windo
Type: Windows Installer Package
openlogic-openjdk-11.0.24+8-windows

3.3 CREATING A DATABASE USING MONGODB or MONGODB COMPASS



```
demo> db.employeDtls.insertOne( { "id":"101", "fname":"Onkar", "lname":"Malawade"})
{
  acknowledged: true,
  insertedId: ObjectId("66d53d6eabac295c6e748566")
}
demo> db.createCollection("myCollection");
{ ok: 1 }
demo> show collections;
employeDtls
employeeDtls
myCollection
demo> db.myCollection.insertOne({"fname":"demo"});
{
  acknowledged: true,
  insertedId: ObjectId("66d53ec2abac295c6e748567")
}
demo> db.new_collection.drop();
true
demo> db.myCollection.drop();
true
demo> db.employeeDtls.find( {fname: "Onkar"} )

demo> db.employeDtls.find( {fname: "Onkar"} )
[
  {
    _id: ObjectId("66d53d6eabac295c6e748566"),
    id: '101',
    fname: 'Onkar',
    lname: 'Malawade'
  }
]
demo> db.employeeDtls.drop();
true
demo>
```

```
demo> db.employeDtls.find({"id":"101"})
[
  {
    _id: ObjectId("66d53d6eabac295c6e748566"),
    id: '101',
    fname: 'Onkar',
    lname: 'Malawade'
  }
]
demo> db.employeDtls.find({"id":"102"})
[
  {
    _id: ObjectId("66d53f80abac295c6e748568"),
    id: '102',
    fname: 'Atharva',
    lname: 'Bhatye'
  }
]
demo> db.employeDtls.find({"id":"103"})
[
  {
    _id: ObjectId("66d53f80abac295c6e748569"),
    id: '103',
    fname: 'Aafan',
    lname: 'Kotawadekar'
  }
]
demo> db.employeDtls.find({"fname":"Atharva"})
[
  {
    _id: ObjectId("66d53f80abac295c6e748568"),
    id: '102',
    fname: 'Atharva',
    lname: 'Bhatye'
  }
]
demo>
```

```
demo> db.employeeDtls.find( {fname: "Onkar"} )

demo> db.employeDtls.find( {fname: "Onkar"} )
[
  {
    _id: ObjectId("66d53d6eabac295c6e748566"),
    id: '101',
    fname: 'Onkar',
    lname: 'Malawade'
  }
]
demo> db.employeeDtls.drop();
true
demo> db.employeDtls.insertMany( [ { "id":"102", "fname":"Atharva", "lname" : "Bhatye"},{ "id":"103", "fname":"Aafan", "lname" : "Kotawadekar"} ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("66d53f80abac295c6e748568"),
    '1': ObjectId("66d53f80abac295c6e748569")
  }
}
```

```
> db.employeDtls.updateOne({id: "104"}, {$set:{fname:"Bhushan"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
demo>
```

```
> db.employeDtls.deleteOne ({"id":"104"})
< {
    acknowledged: true,
    deletedCount: 1
  }
demo>
```

```
>_MONGOSH

> db.employeDtls.find()
< {
    _id: ObjectId("66d53d6eabac295c6e748566"),
    id: '101',
    fname: 'Onkar',
    lname: 'Malawade'
  }
  {
    _id: ObjectId("66d53f80abac295c6e748568"),
    id: '102',
    fname: 'Atharva',
    lname: 'Bhatye'
  }
  {
    _id: ObjectId("66d53f80abac295c6e748569"),
    id: '103',
    fname: 'Aafan',
    lname: 'Kotawadekar'
  }
  {
    _id: ObjectId("66d54379178981b3977f7cd1"),
    id: '109',
    Aafan: 'Atharv'
  }
```

```
> db.employeDtls.updateOne({"id" : "109"},{$set : {"email" : "omshree@gmail.com"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
{
  _id: ObjectId("66d543791789 81b3977f7cd1"),
  id: '109',
  Aafan: 'Atharv',
  email: 'omshree@gmail.com'
}
```

3.4 CREATING COLLECTIONS IN MONGODB

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Syntax

Basic syntax of createCollection() command is as follows

db.createCollection(name, options)

In the command, name is the name of the collection to be created. Options is a document and is used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexId	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

3.5 CRUD DOCUMENT

As we know, we can use MongoDB for a variety of purposes such as building an application (including web and mobile), data analysis, or as an administrator of a MongoDB database. In all of these cases, we must interact with the MongoDB server to perform specific operations such as entering new data into the application, updating data in the application, deleting data from the application, and reading the data of the application.

MongoDB provides a set of simple yet fundamental operations known as CRUD operations that will allow you to quickly interact with the MongoDB server.

3.5.1 Create Operations — these operations are used to insert or add new documents to the collection. If a collection does not exist, a new collection will be created in the database. MongoDB provides the following methods for performing and creating operations:

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.

insertOne() As the namesake, insertOne() allows you to insert one document into the collection. For this example, we're going to work with a collection called RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on RecordsDB. We then provide the information we want to insert in the form of key-value pairs, establishing the Schema.

Example:

```
db.RecordsDB.insertOne({
   name: "Marsh",
   age: "6 years",
   species: "Dog",
   ownerAddress: "380 W. Fir Ave",
   chipped: true
})
```

insertMany() It's possible to insert multiple items at one time by calling the insertMany() method on the desired collection. In this case, we pass multiple items into our chosen collection (RecordsDB) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

```
db.RecordsDB.insertMany([{
   name: "Marsh",
   age: "6 years",
   species: "Dog",
   ownerAddress: "380 W. Fir Ave",
   chipped: true},
    {name: "Kitana",
    age: "4 years",
    species: "Cat",
    ownerAddress: "521 E. Cortland",
    chipped: true}])
```

3.5.2 Read Operations

You can give specific query filters and criteria to the read operations to indicate the documents you desire. More information on the possible query filters can be found in the MongoDB manual. Query modifiers can also be used to vary the number of results returned. MongoDB offers two ways to read documents from a collection:

- db.collection.find()
- db.collection.findOne() find()

In order to get all the documents from a collection, we can simply use the find() method on our chosen collection. Executing just the find() method with no arguments will return all records currently in the collection.

db.RecordsDB.find

findOne()

In order to get one document that satisfies the search criteria, we can simply use the findOne() method on our chosen collection. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. If no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax. db.{collection}.findOne({query}, {projection})

3.5.3 Update Operations

Update operations, like create operations, work on a single collection and are atomic at the document level. Filters and criteria are used to choose the documents to be updated during an update procedure. You should be cautious while altering documents since alterations are permanent and cannot be reversed. This also applies to remove operations. There are three techniques for updating documents in MongoDB CRUD:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne() **updateOne()**

With an update procedure, we may edit a single document and update an existing record. To do this, we use the updateOne() function on a specified collection, in this case "RecordsDB." To update a document, we pass two arguments to the method: an update filter and an update action.

The update filter specifies which items should be updated, and the update action specifies how those items should be updated. We start with the update filter. Then we utilise the "$set" key and supply the values for the fields we wish to change. This function updates the first record that matches the specified filter.

updateMany()

updateMany() allows us to update multiple items by passing in a list of items, just as we did when inserting multiple items. This update operation uses the same syntax for updating a single document.

replaceOne()

The replaceOne() method is used to replace a single document in the specified collection. replaceOne() replaces the entire document, meaning fields in the old document not contained in the new will be lost.

3.5.4 Delete Operations

Delete operations, like update and create operations, work on a single collection. For a single document, delete actions are similarly atomic. You can provide delete actions with filters and criteria to indicate which documents from a collection you want to delete. The filter options use the same syntax as the read operations. MongoDB provides two ways for removing records from a collection:

db.collection.deleteOne()

db.collection.deleteMany()

deleteOne()

deleteOne() is used to remove a document from a specified collection on the MongoDB server. A filter criterion is used to specify the item to delete. It deletes the first record that matches the provided filter.

deleteMany()

deleteMany() is a method used to delete multiple documents from a desired collection with a single delete operation. A list is passed into the method and the individual items are defined with filter criteria as in deleteOne().

3.6 SUMMARY

- MongoDB is an open-source database that uses a document-oriented data model and a non-structured query language
- It is one of the most powerful NoSQL systems and databases around, today.
- The data model that MongoDB follows is a highly elastic one that lets you combine and store data of multivariate types without having to compromise on the powerful indexing options, data access, and validation rules.
- A group of database documents can be called a collection. The RDBMS equivalent to a collection is a table. The entire collection exists within a single database. There are no schemas when it comes to collections. Inside the collection, various documents can have varied fields, but mostly the documents within a collection are meant for the same purpose or for serving the same end goal.
- A set of key-value pairs can be designated as a document. Documents are associated with dynamic schemas. The benefit of having dynamic schemas is that a document in a single collection does not have to possess the same structure or fields. Also, the common fields in a collection document can have varied types of data.

3.7 REFERENCES

1. https://www.mongodb.com
2. https://www.tutorialspoint.com
3. https://www.w3schools.com 4. https://www.educba.com

   https://computingforgeeks.com/install-mongodb-on-windows-

   server/#google_vignette