

MONGO DB

CLASS 2 : DATABASE, TYPES , COLLECTIONS

WHAT IS MONGO COMPASS?

MongoDB Compass is a graphical user interface (GUI) for MongoDB. It is used in a variety of applications, e.g.

1. Visual Data Exploration: Enables users to visually explore and interact with their data without having to use MongoDB Query Language (MQL).

2. Schematic visualization: Provides a description of the data, showing the structure and characteristics of the data, which helps to understand the structure of the data

3. Query Building: Using visual query builders helps users build complex queries, which simplifies the database query process.

4. Performance Insights: Provides performance insights through transparent query usage statistics, helping users identify and optimize slow queries.

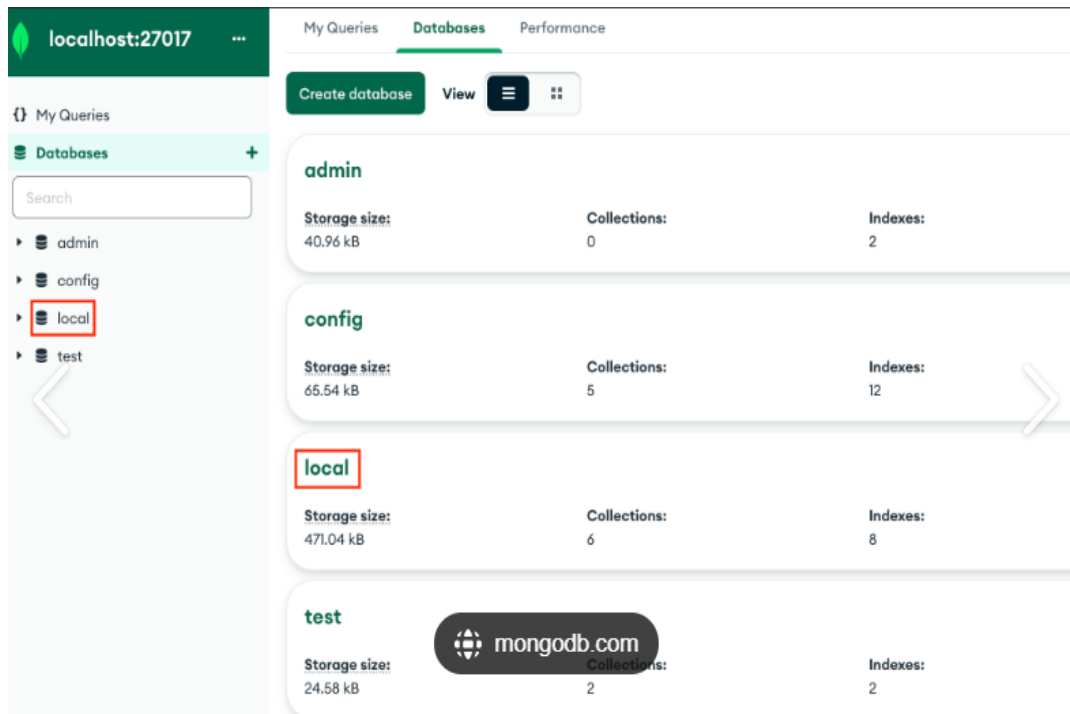
5. Index Management: Enables users to create, modify, and delete indexes, improving query performance.

6. Data Management: Facilitates CRUD (Create, Read, Update, Delete) operations on data in collections and databases.

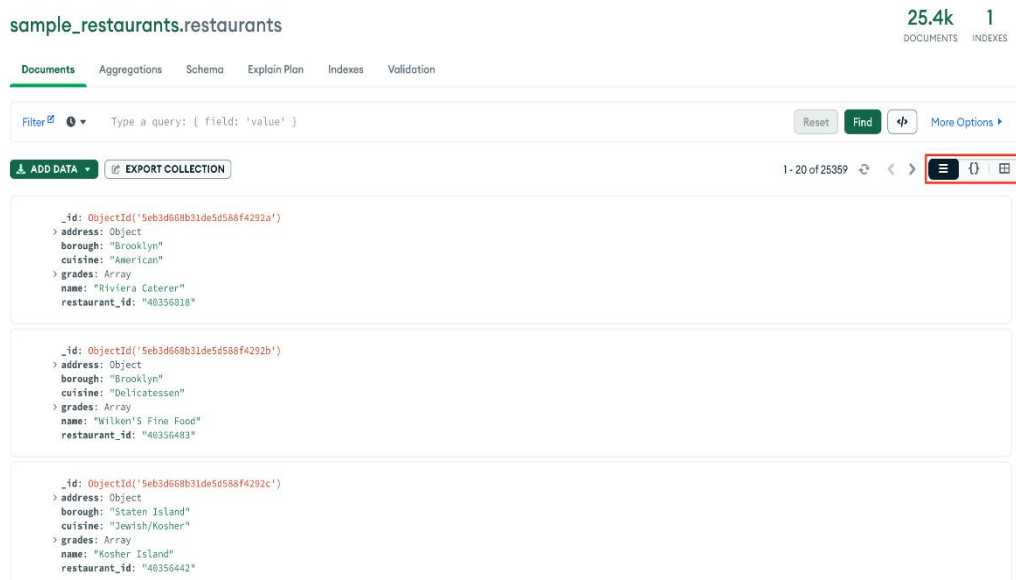
7. Aggregation: Provides tools for creating and visualizing aggregates, which are important for data processing and analysis.

8. Data Import/Export: Allows you to import and export data to the database, which is useful for data transfer and storage. Overall, MongoDB Compass simplifies database management and increases productivity with an intuitive user interface that lets you interact with MongoDB.

- First open Mongo DB Compass and then create a new database **db** in the data base section along with a collection name data you want to create.
- Then import the data for the collection and add it to the db using the command **ADD DATA**.



- After importing the data to the database , we will open mongo shell to get the necessary data from the collection.



MONGO SHELL:

The MongoDB Shell (mongosh) command line interface is used to communicate with MongoDB. They are used for several main reasons:

1. Database management: Administrators and developers use the shell to manage databases, collections, and documents, including tasks such as creating, updating, and deleting databases and collections

2. Query Execution: Enables users to execute MongoDB queries, and provides data retrieval and manipulation directly from the command line. 3. Debugging and testing: We often use the shell to test examples.

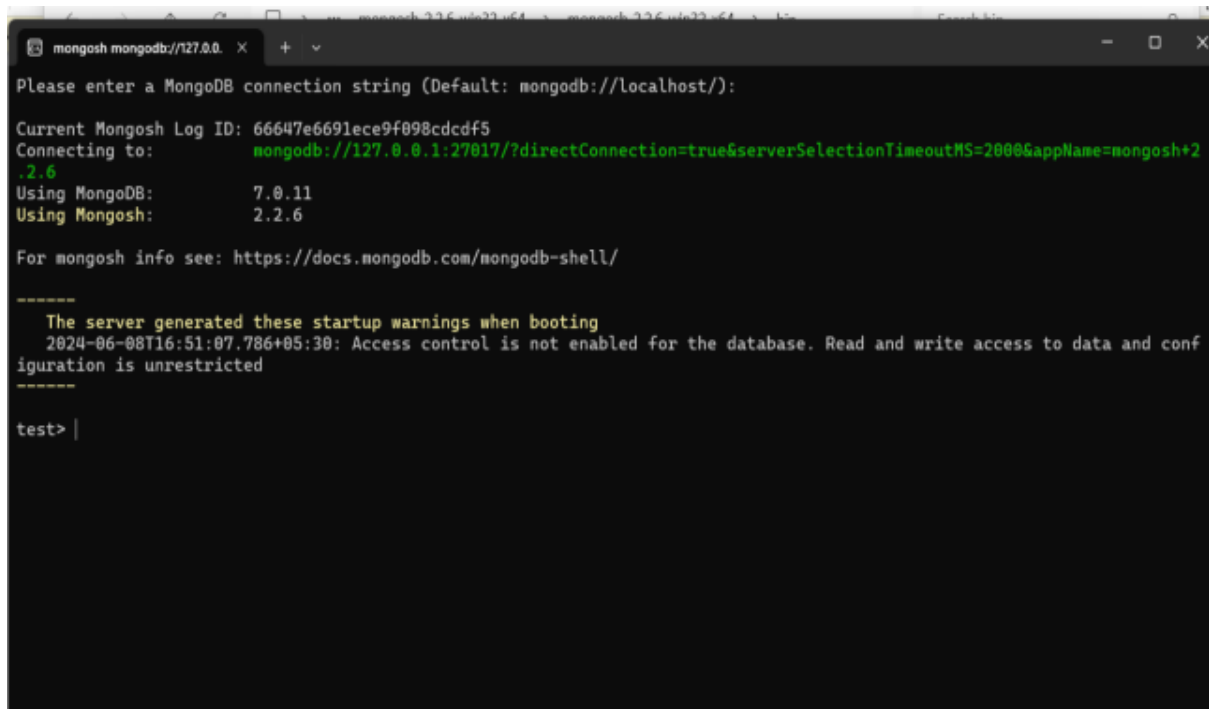
3. Debugging and testing: We often use the shell to test examples.

4. Access to the MongoDB API: The shell provides access to the MongoDB API, allowing users to use all the features and capabilities of MongoDB directly from the command line.

5. Interactive Data Exploration: Users can interactively explore and manipulate data, which is especially useful for querying and data analysis.

Overall, mongosh is a powerful tool for MongoDB users, which provides great power and flexibility to manipulate and interact with MongoDB databases.

- After opening mongo shell command prompt we can find outputs by using commands.



```

mongosh mongodb://127.0.0.1:27017/
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 66647e6691ece9f898cdcdf5
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.6
Using MongoDB:      7.0.11
Using Mongosh:       2.2.6

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2024-06-08T16:51:07.786+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

test> |
  
```

COMMANDS:

A command is an instruction given by a user to a computer or software to perform task. It can be a single word, a line of code or a series of instructions that tell the computer what to do.

Here in Mongo DB we use a command called “show dbs” where it shows all the databases which are imported through mongo compass

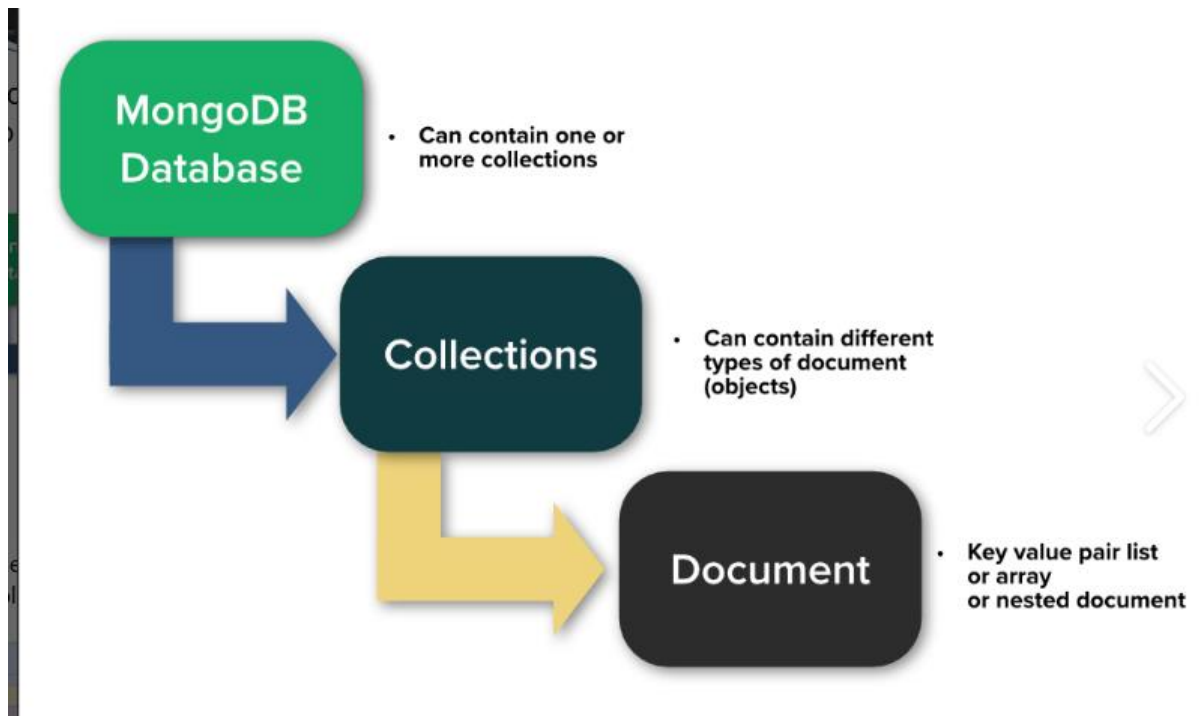
Then, we use a command called “use db” where this connects to the database which is imported.

To check whether the database is switched with a particular collection we use a command “show collections”. It’s a proof of switching a db.

Command	Expected Output	Notes
show dbs	admin 40.00 KiB config 72.00 KiB db 128.00 KiB local 40.00 KiB	All Databases are shown
use db	switched to db db	Connect and use db
show collections	Students	Show all tables
db.foo.insert({"bar" : "baz"})		Insert a record to collection. Create Collection if not exists

Command	Notes
db.foo.batchInsert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])	Insert more than one document
db.foo.find()	Print all rows
db.foo.remove()	Remove foo table

DATABASE, COLLECTION AND DOCUMENTS:



DATA BASE:

In MongoDB Compass, a database serves as a container for collections, providing an organized structure for managing related sets of data. Databases in MongoDB can hold multiple 3 collections, each with its own set of documents, and they enable efficient data segmentation and access control. With Compass, users can easily create, rename, and delete databases, as well as explore their contents through a graphical interface. The intuitive design of Compass allows users to drill down from the database level to individual collections and documents, providing a comprehensive view of the data hierarchy. Additionally, Compass offers functionalities such as monitoring database performance, managing indexes, and executing queries, making it a powerful tool for database administration and data analysis. This seamless interaction with databases in MongoDB Compass enhances productivity and simplifies the management of complex data structures.

COLLECTONS:

Collections in MongoDB Compass are analogous to tables in a relational database but offer much greater flexibility due to MongoDB's schema-less nature. A

collection is a grouping of MongoDB documents, where each document can have a different structure, making it easy to store diverse data within the same collection. In MongoDB Compass, collections are visually represented, allowing users to explore and manage their data seamlessly. Users can create, delete, and rename collections, as well as view detailed statistics about the documents they contain. Compass provides tools to index collections, ensuring efficient query performance, and offers various options to filter, sort, and aggregate data within collections. This user-friendly interface empowers users to perform complex database operations and gain insights into their data without needing extensive MongoDB query language knowledge.

DOCUMENTS:

A document in MongoDB is a record in a collection, akin to a row in a relational database, but it is stored in a flexible, JSON-like format called BSON (Binary JSON). This flexibility allows for the storage of nested structures and varying data types without a predefined schema. Using Compass, users can easily visualize document structures, modify field values, add new fields, and run queries to filter documents, all without needing to write complex code. This makes MongoDB Compass a powerful tool for both developers and database administrators, enabling efficient management of data and simplification of database operations.

DATA TYPES:

1. **String:** This is the most commonly used data type in MongoDB to store data, BSON strings are of UTF-8. So, the drivers for each programming language convert from the string format of the language to UTF-8 while serializing and de-serializing BSON. The string must be a valid UTF-8.
2. **Integer:** In MongoDB, the integer data type is used to store an integer value. We can store integer data type in two forms 32 -bit signed integer and 64 – bit signed integer.
3. **Double:** The double data type is used to store the floating-point values.
4. **Boolean:** The boolean data type is used to store either true or false.
5. **Null:** The null data type is used to store the null value.

MongoDB Data types

S.No	Data type	Explanation
1	String	String in MongoDB must be UTF-8
2	Integer	Integer can be 32 bit or 64 bit depending upon our server
3	Boolean	To store values (true/false)
4	Double	Store floating point values
5	Min/Max keys	Used to compare a value against the lowest and highest BSON elements
6	Arrays	Used to store arrays or list or multiple values
7	Timestamp	For recording when the document has been modified
8	Object	Used for embedded documents
9	Null	Used to store a Null value
10	Symbol	Used to specify symbol type
11	Date	Used to store current date or time in UNIX time format
12	Object ID	Used to store document's ID
13	Binary data	Used to store binary data
14	Code	Used to store javascript code into the document
15	Regular Expression	Used to store regular expression



38

ADD, UPDATE:

The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()** method.

Syntax

The basic syntax of **insert()** command is as follows –

```
>db.COLLECTION_NAME.insert(document)
```

DELETE:

Remove () Method – db.Collection.remove()

The **remove ()** method removes documents from the database. It can remove one or all documents from the collection that matches the given query expression. If you pass an empty document ({}), then it will remove all documents from the specified collection. It takes four parameters and returns an object that contains the status of the operation.

Database: *gfg*

Collections: *student*

Document: *Three documents contains name and the age of the students*

```
> use gfg
switched to db gfg
> db.student.find().pretty()
{
  "_id" : ObjectId("600ee5c60cf217478ba93578"),
  "name" : "Akshay",
  "age" : 19
}
{
  "_id" : ObjectId("600ee5c60cf217478ba93579"),
  "name" : "Bablu",
  "age" : 18
}
{
  "_id" : ObjectId("600ee5c60cf217478ba9357a"),
  "name" : "chetan",
  "age" : 18
}
> █
```

Example 1: Remove all the documents that match the given condition

```
db.student.remove({name: "Akshay"})
```

Here, we remove all the documents from the student collection that matches the given condition, i.e, name: "Akshay".


```
[> db.student.remove({name: "Akshay"})
WriteResult({ "nRemoved" : 1 })
[> db.student.find().pretty()
{
  "_id" : ObjectId("600ee5c60cf217478ba93579"),
  "name" : "Bablu",
  "age" : 18
}
{
  "_id" : ObjectId("600ee5c60cf217478ba9357a"),
  "name" : "chetan",
  "age" : 18
}
> █
```