Database

Till now we are not using any database to store our data, but now we will use **MongoDB**

There are 2 popular type of data base

* **SQL (Structured Query Language):** MySQL, PostgreSQL (most popular)
* **NoSQL (Not Structured Query Language):** MongoDB, FireStore
* In **NoSql** we have different Database based on our requirement
* Document-Based (general): MongoDB ()
* Column-Family: Cassandra (heavy number of write)
* Vectors: qdrant, pinecone
* Graph: neo4j
* Key/value: redies (caching, heavy number of reads)

|  |  |
| --- | --- |
| **Sql** | **MongoDB** |
| Uses tables to store data in rows and columns. | Uses collections and documents (BSON format) to store data. |
| Schema-based, meaning the structure of the data is predefined and rigid. | Schema-less, meaning the structure of the data can be flexible and dynamic. |
| Uses SQL (Structured Query Language) for defining and manipulating data. | Uses its own query language that is less standardized compared to SQL. |
| Generally, scales vertically (adding more power to the existing machine).  Horizontal scaling (sharding) is possible but more complex. | Designed for horizontal scaling and can easily distribute data across multiple servers.  Uses sharding to scale out horizontally. |
| Performance can be high for complex queries and transactions but may degrade with massive data. | Often faster for simple queries and large-scale reads/writes due to its document-based model.  Better suited for big data applications. |

**Example:**

**Sql: Table Name – Users**

**Row-2**

**Row-1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UserId** | **Orders** | **Fname** | **Mname** | **Lname** |
| 001 | [phone, shoes] | Yash | Kumar | Nayak |
| 002 | [pencil, pen, bottle] | Gaur |  | Nitai |

**Table**

**MongoDB: Collection Name – Users**

[

{

userId: 001,

Orders: [“phone”, “shoes”],

**Document-1**

Fname: “Yash”,

Mname: “Kumar”,

Lname: “Nayak”

**Collection**

}

{

userId: 002,

Orders: [“pencil”, “pen”, “bottle”]

**Document-2**

Fname: “Gaur”,

Lname: “Nitai”

}

]

We can say, Table (row & column) = Collection (array) & Row (entries) = Document (object)

**Here we will learn MongoDB only**

**Document’s id in MongoDB:**

* In MongoDB, when we add new document inside a collection then MongoDB automatic generate and add unique **id** for each document.
* **id’s** Keywill look like “**\_id**” and id’s value will be **hexadecimal**

**Format of document in MongoDB:**

* The data of document looked like it is stored in JSON format and we can perform the object’s query to traverse the data of document.

* But in backend MongoDB stored data of document in BSON (Binary JSON) format, this format converts the JSON to machine-code and also give us option of more datatype like Date.
* Due to this, in BSON format traversing speed of data increased in MongoDB database compared to JSON.

**Example:**

{

     "\_id": ObjectId("507f1f77bcf86cd799439011"),

     "username": "viveknigam3003",

     "roles": ["ADMIN", "USER"],

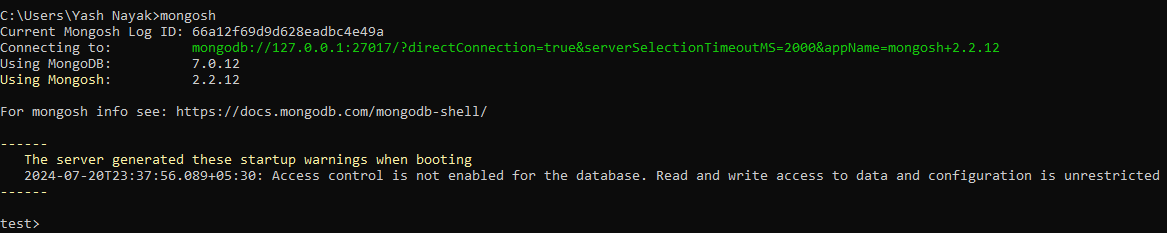
     "email": "vivek@resuminator.in"

}

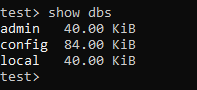
**Using MongoDB (please refer to documentation for commands):**

Cheat-sheet: <https://www.mongodb.com/developer/products/mongodb/cheat-sheet/>

* Write **mongosh** on cmd to start **MongoDB** Server, by default we will be inside **test** database(DB) which is created by MongoDB.



* **show dbs** : To See all the databases and their sizes that are present in MongoDB. **admin**, **config**, **local** database is present by default in MongoDB



* **use <db-name>** :To switch between the databases, in place of <db-name> we have to write database’s name where you want to get switched.



* **db.dropDatabase()** : To delete the database form the MongoDB, before delete any database makes sure that you are inside that database.



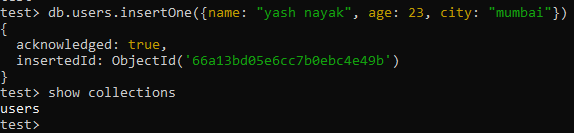
* **show collections** : To see the collections inside the DB.



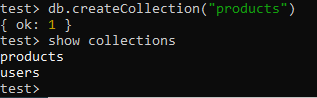
* **db** : To see you are inside which DB



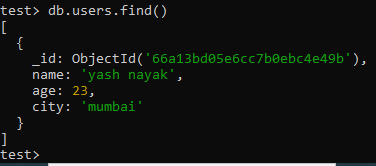
* **db.<collection-name>.insertOne({key: ”value”})** : If collection-name is not present inside the DB we are on then it will create collection based on given collection-name and insert one document inside this collection.



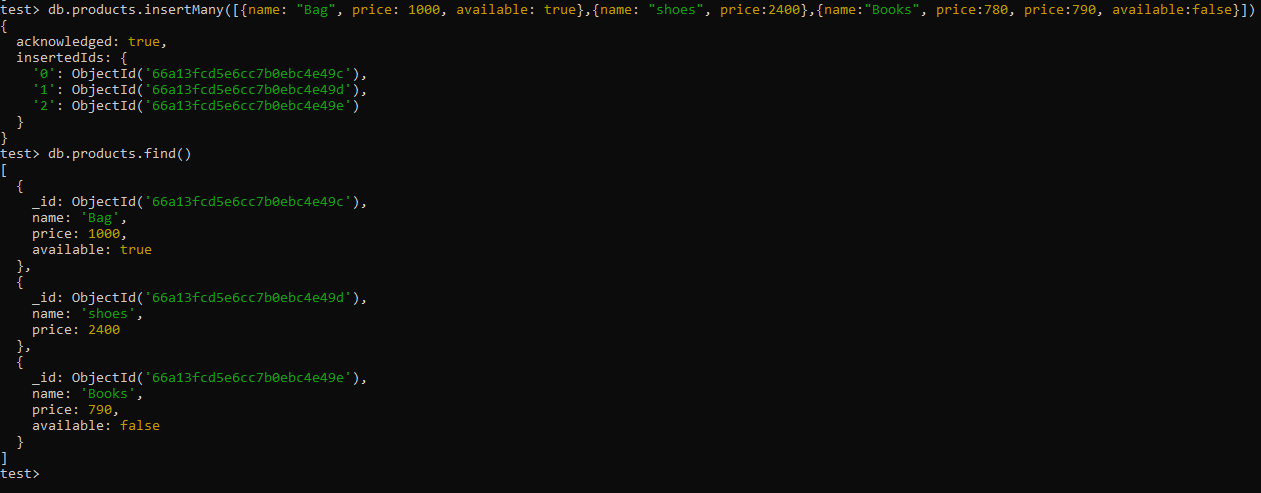
* **db.createCollection(“<collection-name>”)** : To create only collection inside the DB



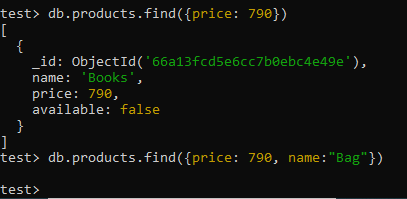
* **db.<collection-name>.find()** : To see all the documents inside the given collection. If the collection don’t have any document then it will return nothing (empty-space).



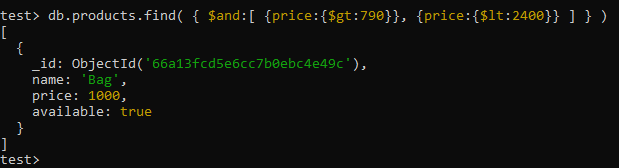
* **db.<collection-name>.insertMany([{key1: value}, {key2: value}, {key3:value}, …])** : We can add multiple document inside the collection at one time.



* **db.<collection-name>.find({key: value, key: value, …})** : we can find the specify object we want, by passing the filter-object inside the find(), if the given filter-object doesn’t matched any document inside the collection then it will return the empty.



* we can add more complex filter like this also using $and / $or / $lt / $gt, here we are asking give the object from products collection were price > 790 and < 2400



* If you want to see more MongoDB commands then go to the cheat-sheet or documentation.

Note: In production we are not going to execute these commands manually. Client will send request to the server and server will execute this command automatically based on the client request.

So, we have to access the MongoDB through the node.js / express server, to do this we have to use ODM (Object Data Model) library.

We will see what is ODM and why we using ODM is next document.