**SQL vs NOSQL**

SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.

SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like **documents or JSON**.

SQL database can scale only Vertically whereas NOSQL database can scale Vertically as well as horizontally.

Example of Vertical and Horizontal scaling.



SQL database can scale only existing data

NoSQL database can scale existing as well as new data too.

SQL has a fixed or static schema whereas NoSQL have a dynamic schema.

SQL databases display data in form of tables so it is known as table-based database whereas NoSQL databases display data as collection of key-value pair, documents, graph databases or wide-column stores.

**SQL Databases** - MYSQL, MS SQL SERVER, POSTRESQL AND AMAZON AURORA

**NoSQL Databases** - Elasticsearch, DynamoDB, mongoDB, Cassandra

**Mysql Installation:**

sudo apt-get install mysql-server  
mysql\_secure\_installation

mysql -u root -p #**Used to connect to database.**

sudo mysql **#Will take u inside of the database.**

Now start creating databases, tables, insert values into it and so on…

**MongoDB database:**

MongoDB is a document database. It stores data in a type of JSON format called BSON.

A record in MongoDB is a document, which is a data structure composed of key value pairs similar to the structure of JSON objects.

GUI applications of MongoDB

1. **MongoDB shell**
2. **MongoDB compass**

// **Installation un ubuntu**.

**Step 1: Import the public key used by the package management system**

sudo apt-get install gnupg

curl -fsSL https://pgp.mongodb.com/server-6.0.asc | \

sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg \

--dearmor

**Step 2: Create a list file for MongoDB (Ubuntu version 22.04)**

echo **"deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg ] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 multiverse"** | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

sudo apt-get update

**Step 3: Install MongoDB package**

sudo apt-get install -y mongodb-org

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **(Optional)**   |  | | --- | | echo **"mongodb-org hold"** | sudo dpkg --set-selections | | echo **"mongodb-org-database hold"** | sudo dpkg --set-selections | | echo **"mongodb-org-server hold"** | sudo dpkg --set-selections | | echo **"mongodb-mongosh hold"** | sudo dpkg --set-selections | | echo **"mongodb-org-mongos hold"** | sudo dpkg --set-selections | | echo **"mongodb-org-tools hold"** | sudo dpkg --set-selections | |

sudo systemctl start mongod

If received an error message such as

Failed to start mongod.service: Unit mongod.service not found.

Run the following command

sudo systemctl daemon-reload

sudo systemctl status mongod

sudo systemctl enable mongod

sudo systemctl stop mongod

sudo systemctl restart mongod

Now to use the MongDB, use

**mongosh**

**The log file will be stored at** /var/log/mongodb/mongod.log file.

The configuration file stored at /etc/mongod.conf

**it was all about the installation:**

**Demo:**

# mongosh -> will open the mongoDB in shell

**Database commands**

1. show dbs -> list all the databases
2. use demo -> will create new database and switch into it.
3. db.dropDatabase() -> Will delete the database

**Collections**: It is similar to table in SQL.2 ways to create the collections

1. **db.createCollection(“collection\_name”)**
2. **db.collection\_name.insert()** -> In this method of creating collection, we need to pass the collection values also.
3. **db.collection\_name.drop()** -> will delete the collection.
4. **show collections** -> Will list all the collections in the database.

**Insert document into collection**:

There are 3 ways using we can insert document, they are

1. insert()
2. insertOne()
3. insertMany()

Example:

1. db.collection\_name.insert({ Name: “Narendra”})
2. db.collection\_name.insertOne({ Name: “Narendra”,

Age: “28”,

Place: “Humnabad” })

// This will create one document.

1. db.collection\_name.insertMany([ { Name: “smith”,

Age: “34”,

Place: “Melbourne”

},

{ Name: “Gayle”,

Age: “35”,

Place: “West”

},

{ Name: “Dhoni”,

Age: “35”,

Place: “Ranchi”

}

])

// this will create the multiple documents.

***Every entry will create the ObjectID of the document.***

**Find():** It is like ‘select \* from’ in SQL

1. db.collection\_name.find() //will show all insert documents.
2. db.collection\_name.findOne() // will show the first inserted document.
3. db.collection\_name.find({Age: “28” }) // this will find all the entries where Age value is 28.

**Delete():** We can delete documents by using two methods

1. deleteOne()
2. deleteMany()

deleteOne() will delete the first document that matches the query provided.

**db.collection\_name.deleteOne({ Age: “34” })** //it will delete the first document where Age is 34.

deleteMany() will delete all the documents that match the query provided.

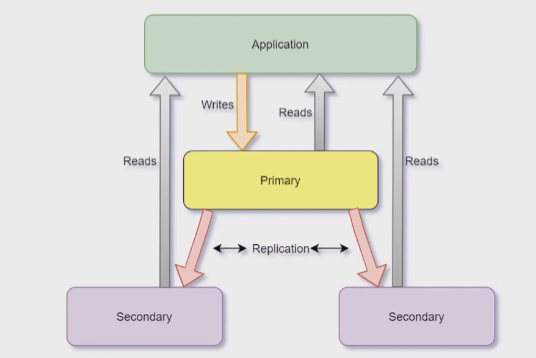
**db.collection\_name.deleteMany({Place: “Humnabad”})** // Will delete all the instances where Place is Humnabad.

**Configuring Replica set in MongoDB**:

Create 3 servers and set up MongoDB in all. in this there will be 1 primary server and 2 secondary servers.

Only from primary server user can read and write the document whereas with secondary servers, user can only read the document.

If primary goes down then one from secondary will become a primary.



This diagram says, from application we will interact with primary MongoDB server and Primary server internally interact with secondary servers.

**Demo:**

Stop MongoDB in all the servers.

sudo systemctl stop mongod

then sudo vim /etc/mongod.conf

|  |
| --- |
| net:  port: 27017  **bindIp: 0.0.0.0**  replication:  **replSetName: “rs1”** |

Do the same in all 3 servers.

Start the mongod db

sudo systemctl start mongod

To start the replication process, choose any one server and other servers in it.

mongosh --- takes into the mongo as a shell.

rs.initiate() --- Initiating replicaset

rs.add (“Private-ip of server-1”)

rs.add(“Private-ip of server-2”)

To verify:

rs.status()

It will show the replicas

if the server ID is 0, it is primary server.

If ID is 1, it is secondary and

If ID is 2, it is also secondary.

Now, go to the other servers

run **mongosh**

It will say it is **secondary…**

Create the **new database in primary server**, this will also be **replicated in secondary servers.**

Now create new collection and insert document into it. This entered document will be replicated in other two servers.

Now to enable the secondary server should read from primary, run the following command:

**Run these commands in Secondary servers.**

|  |
| --- |
| **rs.slaveOK() --- this method is not applicable in MongoDB version 3.2 or later** |

|  |
| --- |
| **Use the following commands if the MongoDB version is 6 or later.**  const db = db.getSiblingDB('myDatabase');  const cursor = db.collection.find().readPref('secondary');  Replace Database with your actual database name and collection with your actual collection name.  **db.collection\_name.find()**  **Example:**  **const db = db.getSiblingDB('testdb');**  **const cursor = db.company.find().readPref('secondary');** |

Now exit from the mongo shell and stop the mongod of primary server,

sudo systemctl stop mongod

Now, enter into the secondary servers and one of them will become primary.

mongosh

If you check the status now, the first server ID will remain 0 but

stateStr: '(not reachable/healthy)'

Start the MongoDB server and it will become secondary server.

sudo systemctl start mongod