EXPERIMENT 2

NAME: Vishal Shashikant Salvi **UID: 2019230069**

CLASS: TE COMPS **BATCH: C**

DATE:

AIM: To study NoSQL database and perform CRUD operations.

Theory:

MongoDB

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Functionality of MongoDB

- Dynamic schema
- No DDL
- Document-based database
- Secondary indexes
- Query language via an API
- Atomic writes and fully-consistent reads
- If system configured that way
- Master-slave replication with automated failover (replica sets)
- Built-in horizontal scaling via automated range-based
- partitioning of data (sharding)
- No joins nor transactions

Why we used mongodb?

- Simple queries
- Functionality provided applicable to most web applications
- Easy and fast integration of data
- No ERD diagram
- Not well suited for heavy and complex transactions systems

Types of NoSQL Databases

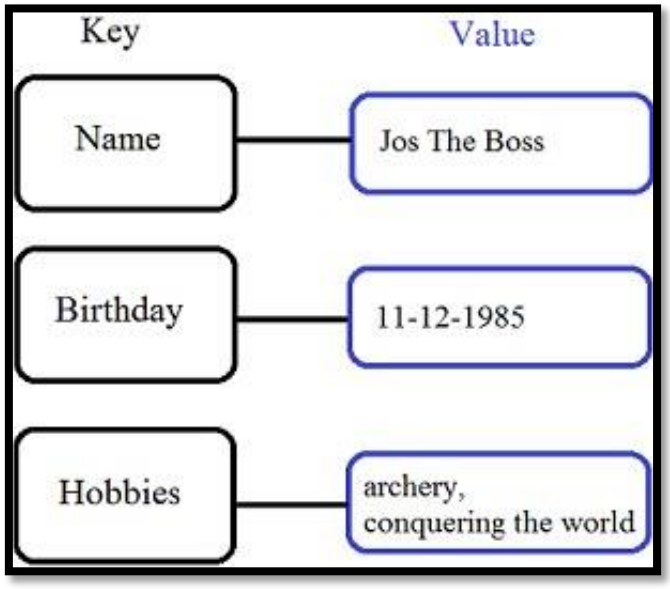
There are four big NoSQL types:

- **key-value store.**
- **document store.**
- **column-oriented database.**
- **graph database**.

Key-Value Pair Based

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.

Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB (Binary Large Objects), string, etc.



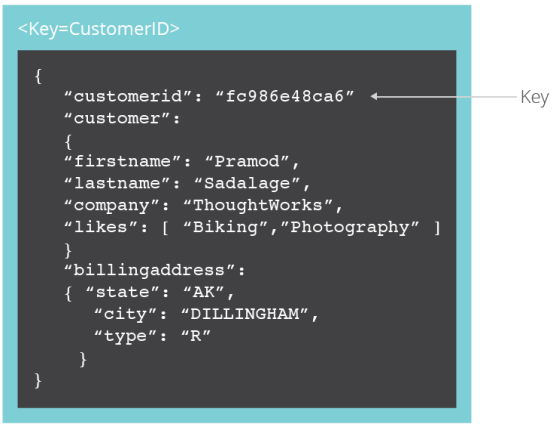
- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object
- Essentially, actual data becomes “value” and an unique id is generated which becomes “key”
- Whole database is then just one big table with these two columns
- Becomes schema-less
- Can be distributed and is, thus, highly scalable
- In essence, a big distributed hash table
- All queries are on keys
- Keys are necessarily indexed

- Example: Cassandra, CouchDB, HBase, Tokyo Cabinet, Redis

The value in a key-value store can be anything: a string, a number, but also an entirely new set of key-value pairs encapsulated in an object. Figure 6 shows a slightly more complex key-value structure.

Document-Oriented:

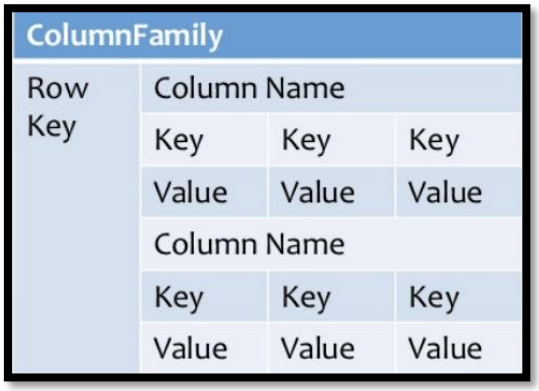
Document stores are one step up in complexity from **key-value stores**: a document store does assume a certain document structure that can be specified with a schema.



- Uses documents as the main storage format of data
- Popular document formats are XML, JSON, BSON, YAML
- Document itself is the key while the content is the value
- Document can be indexed by id or simply its location (e.g., URI)
- Content needs to be parsed to make sense
- Content can be organised further
- Extremely useful for insert-once read-many scenarios
- Can use map-reduce framework to compute
- Example: MongoDB, CouchDB

Column-based

Column-oriented databases work on columns and are based on Bigtable paper by Google. Every column is treated separately. The values of single column databases are stored contiguously.



They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN, etc. as the data is readily available in a column.

- Instead of rows being stored together, columns are stored consecutively
- A single disk block (or a set of consecutive blocks) stores a single column family
- A column family may consist of one or multiple columns

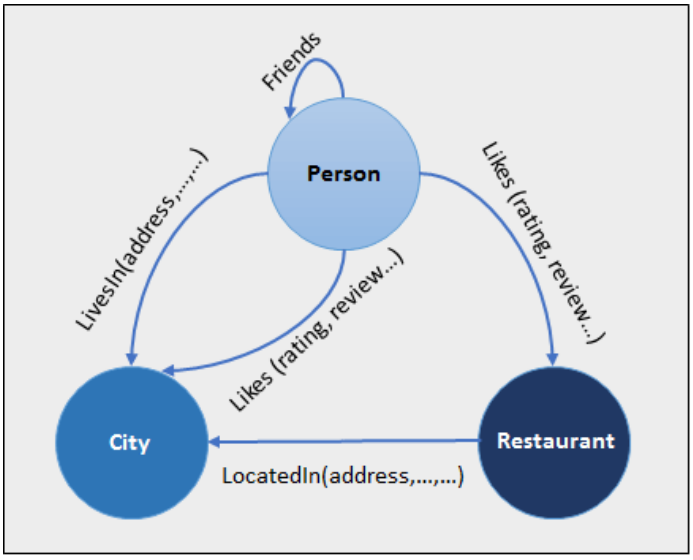
- This set of columns is called a super column
- Two main types Columnar relational models Key-value stores and/or big tables

Column-based NoSQL databases are widely used to manage data warehouses**,** business intelligence**,** CRM**,** Library card catalogs**,** HBase**,** Cassandra**,** HBase**,** Hyper table are examples of a column-based database**.**

Graph-Based

A graph type database stores entity as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge have a unique identifier.

- Nodes represent entities or objects
- Edges encode relationships between nodes Can be directed
- Can have hyper-edges as well
- Easier to find distances and neighbours
- Example: Neo4J, Hypergraph, Infinite Graph, Titan, FlockDB



Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationships as fast as they are already captured into the DB, and there is no need to calculate them.

Crud Operation:

- **Create**
  - db.collection.insert(<document>)
  - db.collection.save(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- **Read**
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- **Update**
  - db.collection.update(<query>, <update>, <options>)
- **Delete**
  - db.collection.remove(<query>, <justOne>)

Create Operations –

The create or insert operations are used to insert or add new documents in the collection. If a collection does not exist, then it will create a new collection in the database.

Db.collection specifies the collection or the 'table' to store the document

- db.collection_name.insert(<document>)
  - Omit the _id field to have MongoDB generate a unique key
  - Example db.**parts**.insert({{type: "screwdriver", quantity: 15 })
  - db.**parts**.insert({_id: 10, type: "hammer", quantity: 1 })
- db.collection_name.update(<query>, <update>, { upsert: true })
  - Will update 1 or more records in a collection satisfying query
- db.collection_name.save(<document>)
  - Updates an existing record or creates a new record

Read Operations –

The Read operations are used to retrieve documents from the collection, or in other words, read operations are used to query a collection for a document.

- db.collection.find(<query>, <projection>).cursor modified
    - Provides functionality similar to the SELECT command
        - <query> where condition , <projection> fields in result set
    - Example: var PartsCursor = db.parts.find({parts: "hammer"}).limit(5)
    - Has cursors to handle a result set
    - Can modify the query to impose limits, skips, and sort orders.
    - Can specify to return the 'top' number of records from the result set
- db.collection.findOne(<query>, <projection>)

Update Operations –

The update operations are used to update or modify the existing document in the collection.

Command:

db.collectionName.update({"key": "value"},{$set:{"key": "update value"}})

Delete Operations –

The delete operation are used to delete or remove the documents from a collection.

Command:

db.collectionName.remove({"key": "value"})

HBase

- Based on Hadoop, HDFS and BigTable
- Key-value store
- Column stores
- Uses column families
- Requires Zookeeper to maintain distributed coordination,
- configuration and maintenance
- Centralized master that dictates slaves
- Strong consistency
- Versioning can be done
- Scales by adding nodes
- CP

Cassandra

- Column-store based on BigTable
- Decentralized architecture

- Replicated
- Any node can perform any action
- Strong security
- Continuous availability
- Extremely good single-tuple read performance
- Not fully consistent
- Requires quorum reads for consistency
- AP

MongoDB

- Document store
- Data stored in JSON or BSON format
- Supports master-slave replication
- Strong consistency
- Good index support
- Data modeling flexibility
- CP

CouchDB

CouchDB is an open source database developed by Apache software foundation. The focus is on the ease of use, embracing the web. It is a NoSQL document store database.

Why CouchDB?

- CouchDB have an HTTP-based REST API, which helps to communicate with the database easily. And the simple structure of HTTP resources and methods (GET, PUT, DELETE) are easy to understand and use.
- As we store data in the flexible document-based structure, there is no need to worry about the structure of the data.
- Users are provided with powerful data mapping, which allows querying, combining, and filtering the information.
- CouchDB provides easy-to-use replication, using which you can copy, share, and synchronize the data between databases and machines.

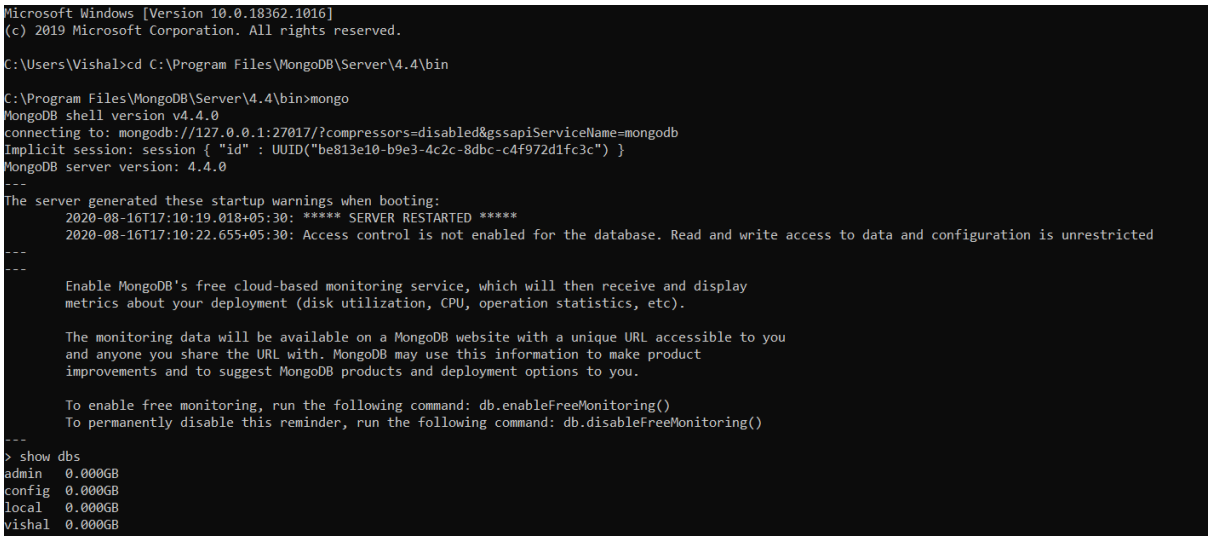
Problem Statement:

To make database for student information system which stored student details who are in college.This information helpful for education establishments used to manage student data. In this system admin can add student details as well as examination details like course, Grade etc.

Key Areas	SQL	NoSQL
Type of database	Relational Database	Non-relational Database
Schema	Pre-defined Schema	Dynamic Schema
Database Categories	Table based Databases	Document-based databases, Key-value stores, graph stores, wide column stores
Complex Queries	Good for complex queries	Not a good fit for complex queries
Hierarchical Data Storage	Not the best fit	Fits better when compared to SQL
Scalability	Vertically Scalable	Horizontally Scalable
Language	Structured Query language	Unstructured Query language
Online Processing	Used for OLTP	Used for OLAP
Base Properties	Based on ACID Properties	Based on CAP Theorem
External Support	Excellent support is provided by all SQL vendors	Rely on community support.

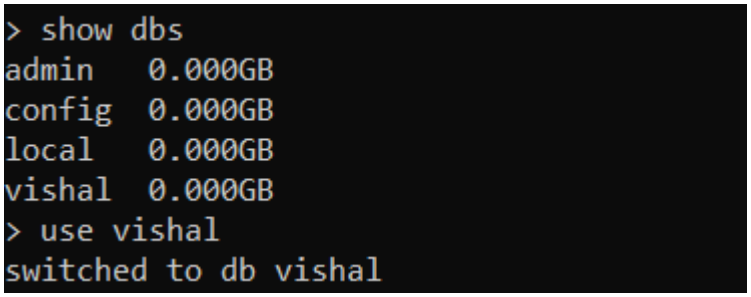
Show Database

Display all database present in system.



Use Database

Select database to perform operations



Drop Database

Drop/remove collection from database

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
vishal  0.000GB
> use vishal
switched to db vishal
> db.student.drop()
true
```

Drop Database and check if it is available or not

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

Create Collection

Create Collection to stored student details

```
> db.dropDatabase()
{ "dropped" : "vishal", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use vishal
switched to db vishal
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.createCollection("Students")
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
vishal  0.000GB
> use vishal
switched to db vishal
```

Insert into Collection

Insert operation to insert key as well as values in database

```
> db.Students.insert({"Name":"Vishal","Roll_No":"52","Age":"19","Gender":"Male","Address":"Mumbai"})
WriteResult({ "nInserted" : 1 })
```

Read Collection

Display/Find the information which is stored in collection.

```
> db.Students.insert({"Name":"Vishal","Roll_No":"52","Age":"19","Gender":"Male","Address":"Mumbai"})
WriteResult({ "nInserted" : 1 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Mumbai"
}
```

```
> db.Students.insert({"Name":"Sandesh","Roll_No":"53","Age":"18","Gender":"Male","Address":"Andheri"})
WriteResult({ "nInserted" : 1 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Mumbai"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
```

Update Operation

Update address of student if student wants to change the address.

```
> db.Students.update({"Address":"Mumbai"},{$set:{"Address":"Borivali"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
```

Insert Operation

Insert student details

```
> db.Students.insert({"Name":"Shivam","Roll_No":"54","Age":"20","Gender":"Male","Address":"Virar"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Tanay","Roll_No":"55","Age":"19","Gender":"Male","Address":"Vasai"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Manish","Roll_No":"56","Age":"18","Gender":"Male","Address":"Bhandup"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Sneha","Roll_No":"57","Age":"19","Gender":"Female","Address":"Jogeshwari"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Harshal","Roll_No":"58","Age":"18","Gender":"Male","Address":"Andheri"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Siddhi","Roll_No":"59","Age":"20","Gender":"Female","Address":"Thane"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Pooja","Roll_No":"60","Age":"19","Gender":"Female","Address":"Andheri"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({"Name":"Rohit","Roll_No":"51","Age":"19","Gender":"Male","Address":"Virar"})
WriteResult({ "nInserted" : 1 })
```

Read Operation

Display / Read student information

```
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
{
        "_id" : ObjectId("5f3922b24ec60c5745a3c406"),
        "Name" : "Manish",
        "Roll_No" : "56",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Bhandup"
}
```

```
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923354ec60c5745a3c408"),
        "Name" : "Harshal",
        "Roll_No" : "58",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923a44ec60c5745a3c40b"),
        "Name" : "Rohit",
        "Roll_No" : "51",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Virar"
}
```

Limit:

Give limit to details that only display/read students detail up to 4

```
> db.Students.find().limit(4).pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
```

```
> db.Students.find().limit(2).pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
```

Skip

Skip Students details therefore user can see remaining details

```
> db.Students.find().skip(2).pretty()
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
{
        "_id" : ObjectId("5f3922b24ec60c5745a3c406"),
        "Name" : "Manish",
        "Roll_No" : "56",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Bhandup"
}
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923354ec60c5745a3c408"),
        "Name" : "Harshal",
        "Roll_No" : "58",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
```

```
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923a44ec60c5745a3c40b"),
        "Name" : "Rohit",
        "Roll_No" : "51",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Virar"
}
```

Skip

```
> db.Students.find().skip(5).pretty()
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923354ec60c5745a3c408"),
        "Name" : "Harshal",
        "Roll_No" : "58",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923a44ec60c5745a3c40b"),
        "Name" : "Rohit",
        "Roll_No" : "51",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Virar"
}
```

Sort

Sort student details in ascending order of age

```
> db.Students.find().sort({"Age":1}).pretty()
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3922b24ec60c5745a3c406"),
        "Name" : "Manish",
        "Roll_No" : "56",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Bhandup"
}
{
        "_id" : ObjectId("5f3923354ec60c5745a3c408"),
        "Name" : "Harshal",
        "Roll_No" : "58",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
```

```
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923a44ec60c5745a3c40b"),
        "Name" : "Rohit",
        "Roll_No" : "51",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
```

Sort -1

Sort student details in descending order of age

```
> db.Students.find().sort({"Age":-1}).pretty()
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
```

```
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3923a44ec60c5745a3c40b"),
        "Name" : "Rohit",
        "Roll_No" : "51",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3922b24ec60c5745a3c406"),
        "Name" : "Manish",
        "Roll_No" : "56",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Bhandup"
}
{
        "_id" : ObjectId("5f3923354ec60c5745a3c408"),
        "Name" : "Harshal",
        "Roll_No" : "58",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
```

```
> db.Student.find({Age:"18"}).explain()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "vishal.Student",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "Age" : {
                                "$eq" : "18"
                        }
                },
                "winningPlan" : {
                        "stage" : "EOF"
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "LAPTOP-9VH1A37S",
                "port" : 27017,
                "version" : "4.4.0",
                "gitVersion" : "563487e100c4215e2dce98d0af2a6a5a2d67c5cf"
        },
        "ok" : 1
}
```

Count

Count the student Record

```
> db.Students.count()
10
```

/^V/

Check the student's details whose name starting from V

```
> db.Students.find({Name:{$regex:/^V/}}).pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
> db.Students.find({Name:{$regex:/^T/}}).pretty()
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
```

```
> db.Students.find({Name:{$regex:/^S/}}).pretty()
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
```

Update Operation

```
> db.Students.update({"Address":"Bhandup"},{$set:{"Address":"Jogeshwari"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3920734ec60c5745a3c402"),
        "Name" : "Vishal",
        "Roll_No" : "52",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Borivali"
}
{
        "_id" : ObjectId("5f3920f94ec60c5745a3c403"),
        "Name" : "Sandesh",
        "Roll_No" : "53",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Andheri"
}
{
        "_id" : ObjectId("5f3922774ec60c5745a3c404"),
        "Name" : "Shivam",
        "Roll_No" : "54",
        "Age" : "20",
        "Gender" : "Male",
        "Address" : "Virar"
}
{
        "_id" : ObjectId("5f39228f4ec60c5745a3c405"),
        "Name" : "Tanay",
        "Roll_No" : "55",
        "Age" : "19",
        "Gender" : "Male",
        "Address" : "Vasai"
}
{
        "_id" : ObjectId("5f3922b24ec60c5745a3c406"),
        "Name" : "Manish",
        "Roll_No" : "56",
        "Age" : "18",
        "Gender" : "Male",
        "Address" : "Jogeshwari"
}
```

Remove Operation

Remove details of male from collection

```
> db.Students.remove({"Gender":"Male"})
WriteResult({ "nRemoved" : 7 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
```

```
> db.Students.remove({"Gender":"Male"})
WriteResult({ "nRemoved" : 7 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
{
        "_id" : ObjectId("5f3923884ec60c5745a3c40a"),
        "Name" : "Pooja",
        "Roll_No" : "60",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Andheri"
}
> db.Students.count()
3
>
```

```
> db.Students.remove({"Name":"Pooja"})
WriteResult({ "nRemoved" : 1 })
> db.Students.find().pretty()
{
        "_id" : ObjectId("5f3922e64ec60c5745a3c407"),
        "Name" : "Sneha",
        "Roll_No" : "57",
        "Age" : "19",
        "Gender" : "Female",
        "Address" : "Jogeshwari"
}
{
        "_id" : ObjectId("5f3923684ec60c5745a3c409"),
        "Name" : "Siddhi",
        "Roll_No" : "59",
        "Age" : "20",
        "Gender" : "Female",
        "Address" : "Thane"
}
> db.Students.count()
2
>
```

Conclusion:

- NoSQL databases are superior to normal SQL databases, they are more reliable with less cost. They handle more data and are geared towards making life easier for the developer.
- NoSQL databases are quickly becoming a major part of the database landscape today, and they are providing to be a real game-changer in the IT.
- They have numerous benefits, including lower cost, open-source availability, and easier scalability, which makes NoSQL an appealing option for anyone thinking about integrating in Big Data.
- Thus, from this experiment I performed different operation on NoSQL database (MongoDB).