DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Unit 3

INTRODUCTION TO MONGODB AND CASSANDRA

Terms used in RDBMS and Mongo DB- Data Types

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
It is a relational database.	It is a non-relational and document-oriented database.
Not suitable for hierarchical data storage.	Suitable for hierarchical data storage.
It is vertically scalable i.e increasing RAM.	It is horizontally scalable i.e we can add more servers.
It has a predefined schema.	It has a dynamic schema.
It is quite vulnerable to SQL injection.	It is not affected by SQL injection.
It centers around ACID properties (Atomicity, Consistency, Isolation, and Durability).	It centers around the CAP theorem (Consistency, Availability, and Partition tolerance).

RDBMS	MongoDB
It is row-based.	It is document-based.
It is slower in comparison with MongoDB.	It is almost 100 times faster than RDBMS.
Supports complex joins.	No support for complex joins.
It is column-based.	It is field-based.
It does not provide JavaScript client for querying.	It provides a JavaScript client for querying.
It supports SQL query language only.	It supports JSON query language along with SQL.

Data

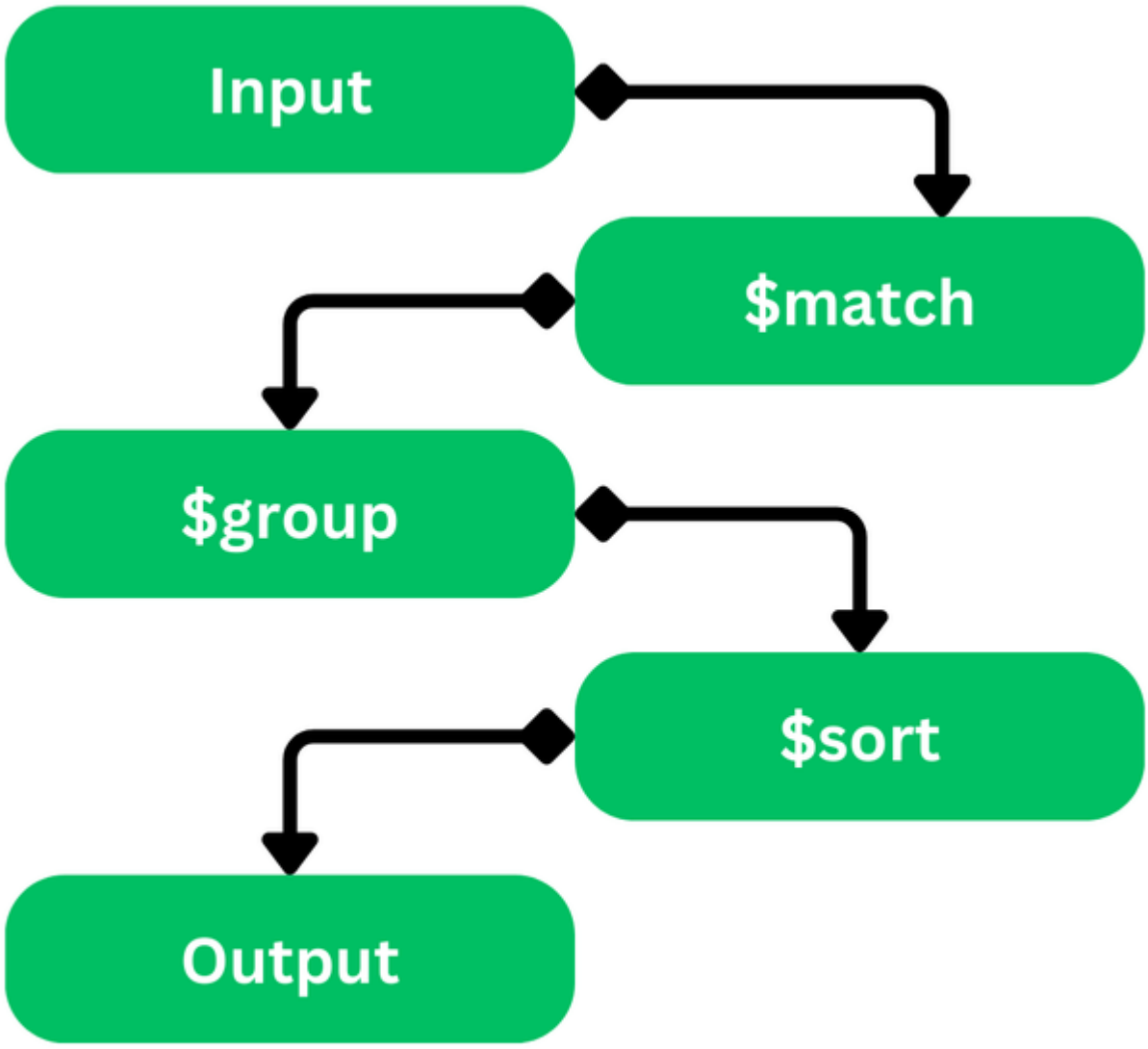
Data is a raw or **unprocessed** information. example: **marks, class**, etc.

Database

Database is a collection of physically stored data into computer. Data is primarily stored on the secondary storage. Data stored in the form of tables. Software that provide a way to manage this data called Database Management System (**DBMS**).

Aggregation

Aggregation is important function of **MongoDB**. Aggregation is the operation which groups the data from multiple destinations and perform some operations onto for produce some result. It is similar to the **count(*)** and **Group By Clause** in **SQL**. Aggregation framework provided by MongoDB is as follow:



Aggregation Framework In MongoDB

- **$match():** This stage filters the data by field (key-value) and passes data to next stage.
- **$group():** This stage groups the data and pass to next stage. The actual computing done here.
- **$sort():** This Function is used to sorts the result produced by the **$group** ascending or descending.

Some Aggregation Functions are Listed Below:

Expression	Description
$sum	Sums the defined values from all the documents in a collection
$avg	Calculates the average values from all the documents in a collection
$min	Return the minimum of all values of documents in a collection
$max	Return the maximum of all values of documents in a collection
$addToSet	Inserts values to an array but no duplicates in the resulting document
$push	Inserts values to an array in the resulting document
$first	Returns the first document from the source document
$last	Returns the last document from the source document

Here, You can find more about the Aggregation in MongoDB

Example of Aggregation: Sum of all marks of students:

```
db.students.aggregate([
  {
    $group:                                                    {
      _id:                                                   null,
```

```
    totalMarks: { $sum: '$marks' },
   },
  },
]);
```

Output:

```
< {
    _id: null,
    totalMarks: 379
  }
```

Indexing

Index is a special type of data structure that stores some information about database so that we can retrieve the information efferently by querying.

Traditionally, without the indexing if we query the database the default index finds the collection based on **_id** field. this method is not efficient and hence to retrieve the information more efficiently and fast we implement Indexing on database. We can Create the index, Drop the index and get description of index.

Creating Index:

```
db.students.ensureIndex({'marks':1});
```

Output:

```
< [ 'marks_1' ]
```

Explanation: marks is the **key** name that you want to make an index and it will be 1 and -1 to make it on ascending or descending order

Drop Index:

```
db.students.dropIndex("marks_1")
```

Output:

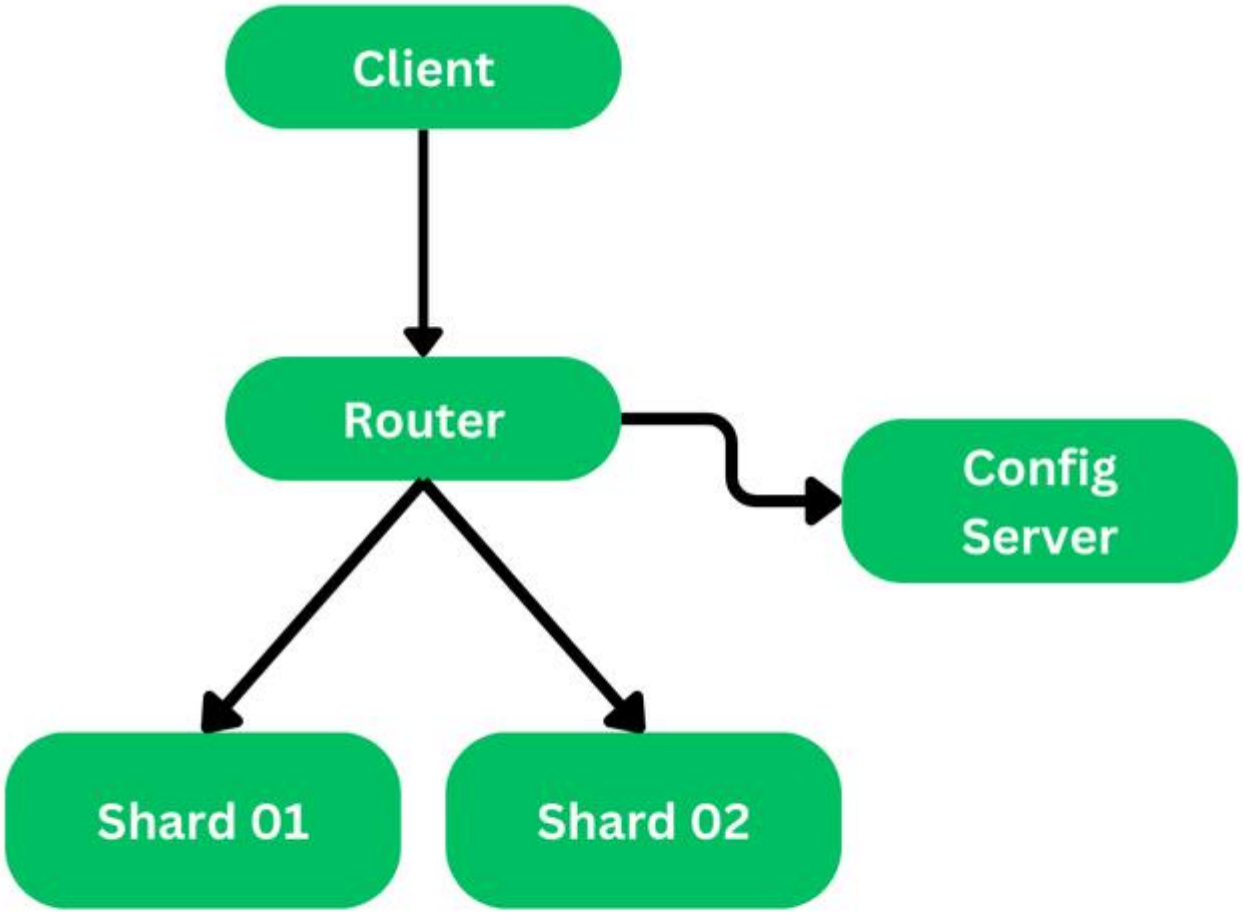
```
< { nIndexesWas: 2, ok: 1 }
```

Here, you can find more about indexing in MongoDB.

Sharding in MongoDB

- **Sharding** is concept in MongoDB, which help achieve distribution of data into different locations. This is a method in which data is allocated across multiple machines.

- It allows horizontal scaling. Sometimes the data required by the request is more larger than the RAM of the server and hence it is mandatory to serve request and provide appropriate data.
- Hence to achieve the Data Availability Database is broken down and stored over multiple servers. By Implementing sharding, we can increase volume and response throughput by just adding servers.



Sharding in MongoDB

Advantages of Sharding

- Allows data distribution.
- Number of request handling by each server gets reduced.

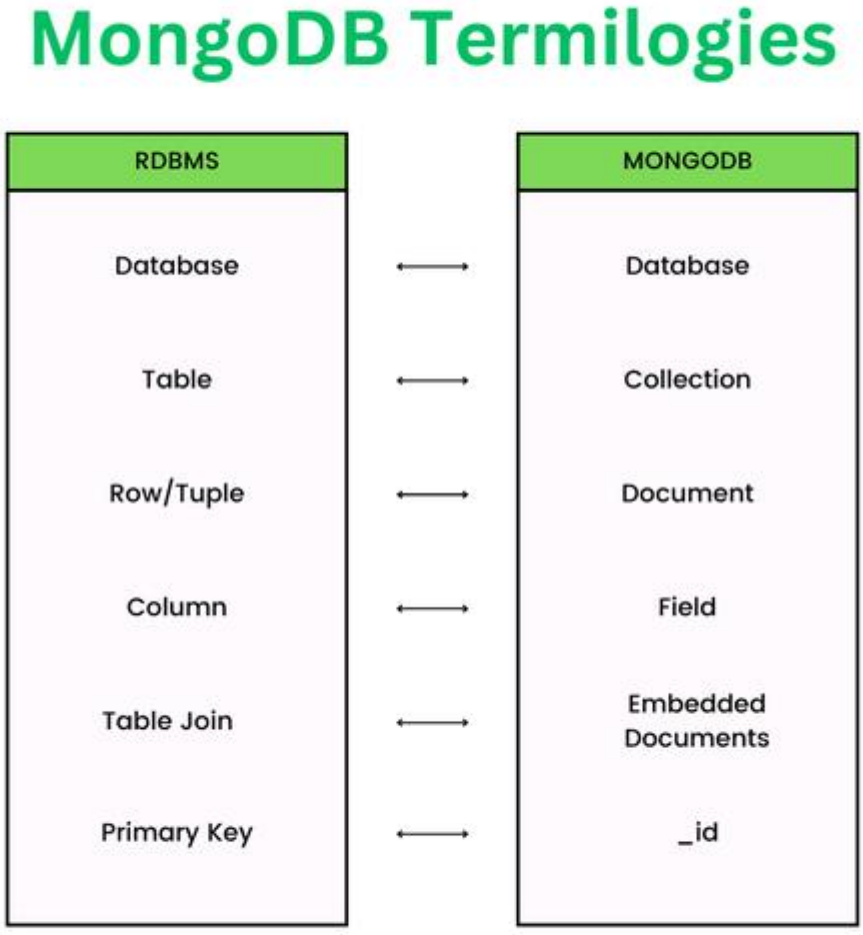
- Makes the system more efficient.
- Capacity of querying and storing data increases.

Disadvantages of Sharding

- Maintenance cost increases.
- Infrastructure cost increases due to expenses of building different servers and storage.
- Slows down the response time due to query overhead.

Here, You can find more on Sharding in MongoDB.

Following chart shows the relationship between SQL terminologies and MongoDB terminologies



DBMS VS MONGODB Terminologies

MongoDB Collection

A **collection** is equivalent to an **RDBMS** table. It store number of documents inside it. Since MongoDB is schema less, collection do not have schemas. We can store any number of documents in single collection and all of them have related purpose.

You can create a collection using the **createCollection()** database method.

Syntax :

```
db.createCollection("collection_name")
```

Output:

```
> db.createCollection("teachers")
< { ok: 1 }
```

Create Collection Using createCollection() Method

You can also create a collection while inserting document:

```
db.collection_name.insertOne(document_object)
```

```
> db.instructors.insertOne({name:"DR. Geek",class:"X"})
< {
    acknowledged: true,
    insertedId: ObjectId("658498b67de3dc96745d21d0")
  }
```

Create Collection by Inserting Record Into It.

It will create if not already exists.

To show all the collections use **show collections** comman

```
> show collections
< instructors
  students
  teachers
```

Get all the Collections

MongoDB Document

It is a basic unit of data in MongoDB. document is set of key value pairs. It is a record which is being inserted in the collection. It may be already existing data in the collection. we insert documents in the form of JSON. document can have flexible schema that means different documents in collection do not need to have same set of fields or structure.

We can insert **document** in the collection using two methods:

- **insertOne()**: This method inserts a single object into the collection.
- **insertMany():** This method inserts an array of objects into the collection.

Example for insertOne():

```
db.students.insertOne({name:"Jayesh", age:21, rollno:77, marks:97})
```

Qutput:

Using insertOne() Method to add document into collectio

Example for insertMany()

```
> db.students.insertMany([
    { name: "Jayesh", age: 21, rollno: 77, marks: 97 },
    { name: "Aisha", age: 22, rollno: 88, marks: 95 },
    { name: "Raj", age: 20, rollno: 65, marks: 90 }]
  )
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("65849aae7de3dc96745d21d2"),
      '1': ObjectId("65849aae7de3dc96745d21d3"),
      '2': ObjectId("65849aae7de3dc96745d21d4")
    }
  }
```

Using insertMany() Method to add document into collection.

findOne(): To show all the documents use **findOne**() method in which you can pass the key value pair of the desired document.

```
> db.students.findOne({name:"Jayesh"})
< {
    _id: ObjectId("658499e47de3dc96745d21d1"),
    name: 'Jayesh',
    age: 21,
    rollno: 77,
    marks: 97
  }
```

getting a single record using findOne()

MongoDB Field

Field is key-value pair which is base of inserting data into documents. documents can have any number of fields. it can have zero or more fields. it is same as columns in RDBMS.

Syntax:

{key:value}

Let’s understand concept of fields with help of example, given is the output of **findOne()** method.

```
> db.students.findOne({name:"Raj"})
< {
    _id: ObjectId("65849aae7de3dc96745d21d4"),
    name: 'Raj',
    age: 20,
    rollno: 65,
    marks: 90
  }
```

Fields in MongoDb

Explanation: In this document there are different fields present such as **name, age , rollno, marks**. **_id** is the special field which is assigned automatically by MongoDB server. Hence, the fields are the key value pairs in the documents.

MongoDB _id

_id is a special key present in the document. it is used to uniquely identify the document. **_id** is automatically generated by the MongoDB. we can’t change the value to this field.

Demonstration of _id

Explanation: Here as we can see **_id: 65849d677de3dc96745d21d5** is automatically generated. we haven't included **_id** field while inserting document.

Conclusion

Overall, We have also seen that the need of MongoDB and some basic terms which are frequently used in MongoDB to perform operations. We sum each and every terminology with the example also compared the MongoDB terminologies with the RDBMS terminologies. Mongodb is a NoSQL database which is used to handle complex, large and uncomplicated data in very easy way.