MongoDB

Basic information about NOSQL with MongoDB

Vraj Suratwala

Department of ICT, VNSGU, Surat

Topics Covered

- What is MongoDB ?
- What is NOSQL?
- Advantages and features of NoSQL.
- Types of NoSQL Databases
- SQL VS MongoDB
- MongoDB Terminologies.
- JSON-BSON Overview
- How to install MongoDB ?
- How MongoDB Works ?
- MongoDB Playground - shell
- Simple CRUD Operation - along with all small concepts.

Topics Covered

- in Read Operation. - Specific Topics
  - JSON Importing
  - Comparision Operator
  - Cursors
  - Logical Operators
  - Complex Expressions
  - Conditional Query

  - The Advanced and Other Remaining Concepts will be in Next Notes!

What is MongoDB ?

- MongoDB is an Open-Source, **Document-Oriented** NO-SQL Database Management System.
- Document Oriented : Refers to store data in the document in JSON format!

MongoDB

Humongous

HU **mongo** US

mongo - Means Large Data Which Scalabale and Maintainable

What is NOSQL Database Management System ?

- NoSQL stands for “Not Only SQL.”
- It refers to a new class of non-relational database systems that provide flexible data models and high scalability — especially useful for big data, real-time apps, and cloud-native systems.
- Schema Less
- High Scalable
- Faster
- Distributed
- Flexible Data Models

Two main Advantages of NOSQL!

- 1. High Scalable
- 2. High Availability

- **Limitations of RDBMS**
- 1. ACID Properties
- 2. Performance Overhead

- Where NOSQL is using the JSON or BSON Data to Communication so there is vast difference in the NOSQL Speed and SQL Speed!

Type of NOSQL Databases

- 1. Key-Value Store : Redis, Riak
- 2. Document Store : MongoDB, CouchDB
- 3. GraphDB : Neo4j, ArangoDB
- 4. Column Based : Cassandra, HBase

SQL vs MongoDB

SQL	MongoDB
Relational Database	Non-Relational Database
Stucture Data in Table Formate	Flexible Data Structure
well Defined Schemas and Fixed Data Data Structure	Ideal for Evouling Data
Uses ACID Properties	Uses CAP Thereom.
Joins are Supported	Joins are not directly Supported
Vertical Scalability	Horizontal Scalability
Example : MYSQL, Oracle, POSTGRESQL	Example : MONGODB, Cassandra, Raddis

Features of MongoDB

- flexible Schema Design.
- Scalabilities and Performance
- DOS - Document Oriented Storage
- Dynamic Queries
- Aggregation Framework
- Open Source and Community

MongoDB Terminologies

Term	Description
Database	a container for collections.
Collection	a group releated document.
Document	a single recored JSON or BSON - Binary JSON.
Feild	a key-value pair inside a document
_id	an unique idenfier for object.
BSON	Binary JSON - for internal storage for documents.
Cursor	Point returns the Query set or Resultset.
Embedded Document	Document under Document.
Sharding	Partition

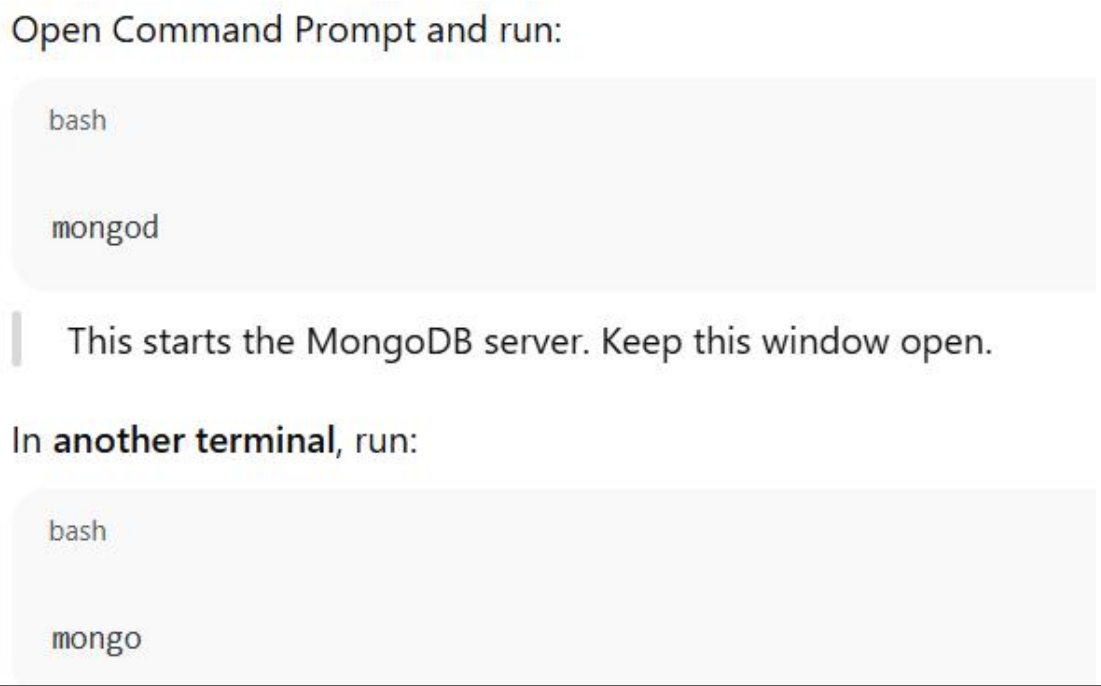
JSON - BSON - internal Process Overview

- in MongoDB, we write in JSON format only but behind the process/scene data is stored in BSON format, a binary representation of JSON.
- Which provides higher operational speed and efficiency.
- it is done by Storage Engine!

How to install MongoDB ? - for windows

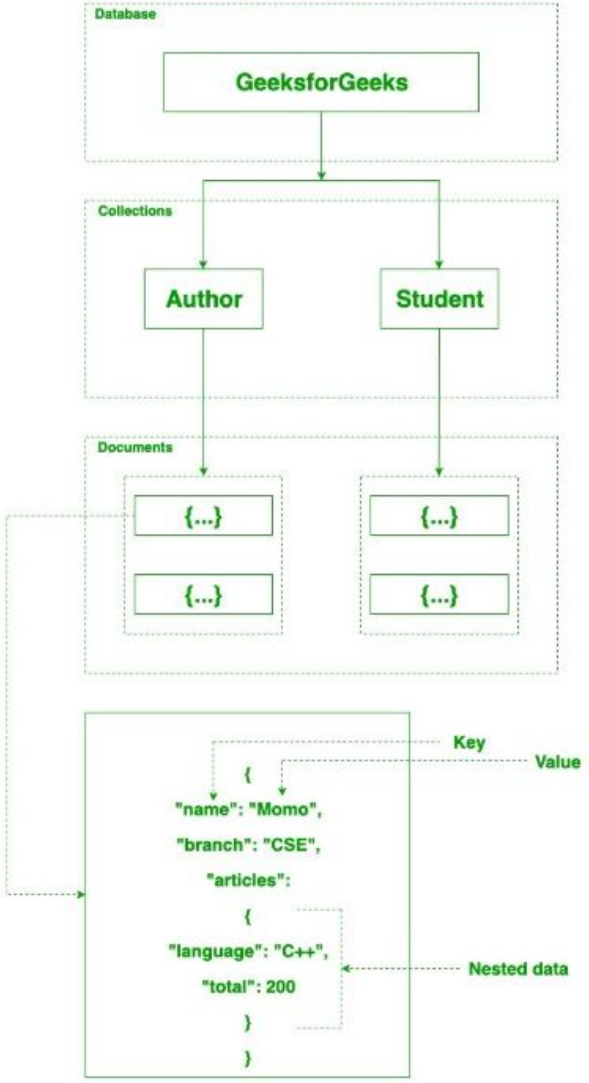
- 1. Install from Official Website : https://www.mongodb.com/try/download/community - download msi version

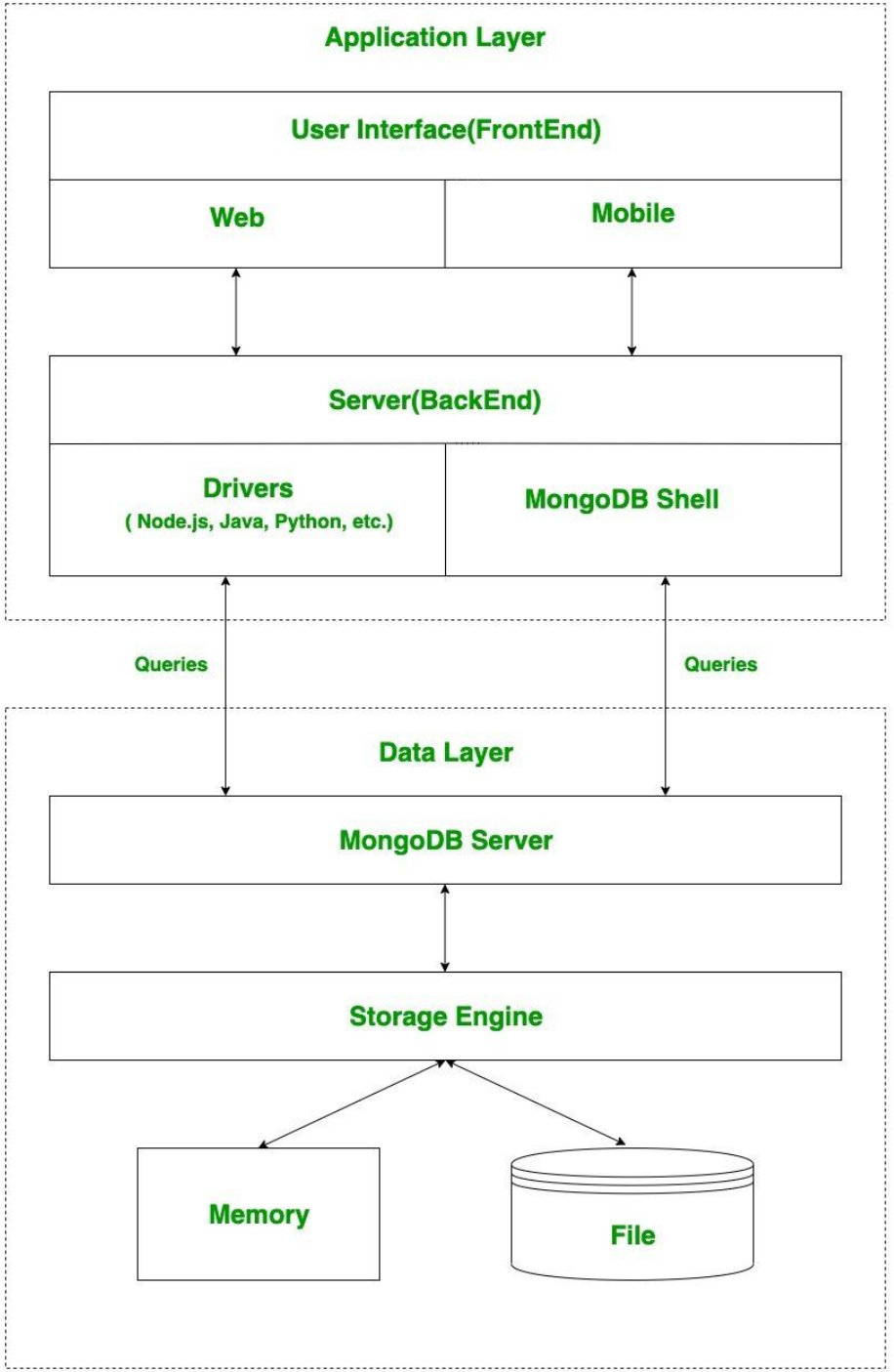
- 2. set up envirnmental variables : C:\Program Files\MongoDB\Server\6.0\bin
- 3. start MongoDB



How MongoDB Works?

Reference :https://www.geeksforgeeks.org/mongodb/what-is-mongodb-working-and-features/





MongoDB shell - Playground

- MongoDB Playground is an interactive environment (like a scratchpad)
  - where you can write and test MongoDB queries using the MongoDB Shell (mongosh) or Playgrounds in VS Code.
- how to enable the Mongo Shell :
  - command : mongosh

Some common commands for Mongo Shell

- show dbs - show all database
- use database_name - use the database with name
- show collections - it will show all the collections for perticular database.
- db.users.find() - find all documents
- db.dropDatabase - it will drop current database.

creating and deleting Collections in playground ?

- Creating a Database : use <database_name>;
- show collections;
- db.createcollections(<'Name'>);

- Deleting Collections : db.dropdatabase();
- Insertion : db.<collection_name>.insertone({
  - field1 : value1
  - …
- });

How to Insert The data in any Collections ?

- Two ways to insert the data :
  - 1. insert()
  - 2. insertone()
  - 3. insertmany()

- 1. insert() :

```
db.students.insertOne({
  name: "Vraj Suratwala",
  branch: "ICT",
  year: 2025
});
```

- 2. insertmany() :

```
db.students.insertMany([
  { name: "ABC", branch: "ICT", year: 2025 },
  { name: "BCD", branch: "ICT", year: 2025 },
  { name: "CDE", branch: "ICT", year: 2025 }
]);
```

- 3. insert() : which is deprecated function and it will not longer supported!
- NOTES ABOUT INSERTION : Insertion can be in two ways.
  - 1. Ordered : Stops insertion when it is getting an error.
  - 2. Unordered : also process next line.

1. Ordered Insertion

```
db.collection.insertMany(
  [
    { _id: 1, name: "ABCD" },
    { _id: 2, name: "BCDE" },
    { _id: 1, name: "Duplicate" }  // This will cause an error
  ],
  { ordered: true } // This is default
);
```

2. Unordered Insertion

```
db.collection.insertMany(
  [
    { _id: 3, name: "ABCD" },
    { _id: 4, name: "BCDE" },
    { _id: 3, name: "Duplicate" }  //Duplicate again, it will not cause an error.
  ],
  { ordered: false }
);
```

Reading Operation in any collections

- 1. find() - find all the data from database.
- 2. findone() - find only first data.
- Example :

db.students.find();

db.products.find();

-- Find with filter :

db.collections.find({key : “value”})

db.students.find({ name: "Vraj" });

db.collection.findOne({ key: "value" });

db.students.findOne({ rollNo: 11 });

Find with Comparision Operator.

Operator	Description
$gt	Greater Than
$lt	Less Than
$gte	Greater or Equal
$lte	Less or Equal
$eq	Equal to
$ne	No Equal to
$in	In Array
$nin	Not In Array

Example

- db.products.find({ price: { $gt: 100 } });
- db.students.find({ marks: { $gte: 70, $lte: 90 } });

- **limit and skip**
- db.collection.find().limit(5); // Top 5 docs
- db.collection.find().skip(5); // Skip first 5
- db.collection.find().skip(5).limit(5); // Page 2

Sorting in Reading

- db.collection.find().sort({ field: 1 }); // Ascending
- db.collection.find().sort({ field: -1 }); // Descending

- Example :
  - db.products.find().sort({ price: -1 }); // Highest price first

- Note :
  - +1 is responsible for Ascending Order.
  - -1 is responsible for Desending Order.

Importing JSON

- Syntax :
  - mongoimport --db <database_name> --collection <collection_name> --file <file_path>.json --jsonArray
- Example :
  - mongoimport --db school --collection students --file students.json --jsonArray
- Explanation :
  - --db : Name of the Database
  - --collection : Name of the Collection
  - --file : Path to your .json file.
  - --jsonArray: Required only if your file contains a full array ([{}, {}, ...]) of documents.

Importing JSON

- Note :
  - The Maximum Limit to import json is upto 16MB.
  - Make sure your MongoDB server is running before using mongoimport.

Cursors in MongoDB

- a cursor is a pointer to the result set of a read (query) operation.
- in other words, we can say that - Cursor in MongoDB are used to efficiency retrieve large results set from queries, providing cantrol over the data retrieval process.

- Automatic Batching - By Default 101 Batch/docs per at a time.
  - Explanation : it will fetch the 101 Batch/docs or 1 MB at a time from per resultset.
  - If there are more documents, it keeps the cursor open on the server.
  - The client can then request more batches automatically as you iterate (.next(), .forEach() etc.).

Simple Example

```
const cursor = db.students.find({ age: { $gt: 18 } });
cursor.forEach(doc => printjson(doc));
```

Key features :

- Helps to Handle Large Data without loading the whole data in-memory.
- Automatically closes after all tha data read.
- it can also manually closed.

- Note :
  - MongoDB uses lazy loading, meaning documents are fetched in batches as needed — thanks to cursors.

Cursor Methods

- count()
- limit(Number_of_limit)
- skip(Number_of_skips)
- sort(either +1(asc) or -1(des))
- next()
- hasnext()
- toArray()
- forEach()

Disavantages of Cursor

- Cursor Timeout - Cursors can expire after 10 minutes of inactivity (default).
- Memory Overhead on Server - An open cursor occupies memory on the server.
- Security - A cursor left open can leak data unintentionally if accessed insecurely.
- Not good for Real time Applications.
- skip() and sort() can impact on Performance so Use Cursor with Caution.

Logical Operators

- There are main four logical Operators which we can use in MongoDB.
- 1. $and
- 2. $or
- 3. $nor
- 4. $not

1. $and

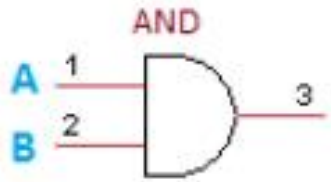
- in $and Condition all the condition must be true.
- Let’s Refer the Truth Table of and condition.

Truth table

Two Input AND gate		
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

theorycircuit.com

Symbol



$Y = A.B$ (or) AB (or) $\overline{\overline{A} + \overline{B}}$

1. $and

- Syntax :

```
db.collection.find({
  $and: [
    { age: { $gt: 18 } },
    { city: "Surat" }
  ]
});
```

- Example : // Shortcut: You can also use this without $and:

```
db.collection.find({
  age: { $gt: 18 },
  city: "Surat"
});
```

2. $or

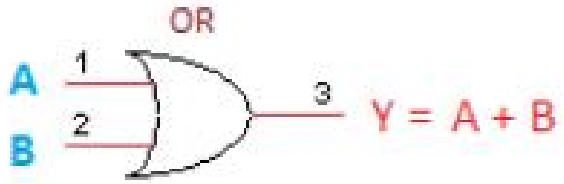
- Any One Condition is Enough
- Truth Table :

Truth table

Two Input OR gate		
A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

theorycircuit.com

Symbol



2. $or

- Example/Syntax :

```
db.collection.find({
  $or: [
    { status: "active" },
    { score: { $gt: 90 } }
  ]
});
```

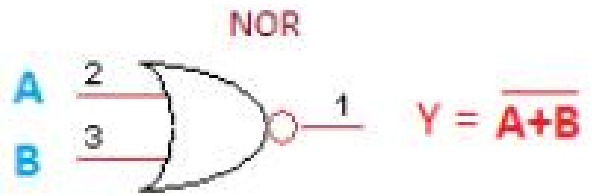
3. $nor

- None of These Conditions Should Be True
- Truth table :

Truth table

Two Input NOR gate		
A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Symbol



3. $nor

- Example/Syntax:

```
db.collection.find({
  $nor: [
    { price: { $gt: 1000 } },
    { category: "luxury" }
  ]
});
```

4. $not

- Negate a Single Condition
- Truth table :

NOT Gate truth table

Input	Output
A	Y
0	1
1	0

Example/Syntax :

```
db.collection.find({
  rating: { $not: { $gte: 4.5 } }
});
```

Complex Expressions

- The $expr Operaator allows us to using aggregation expressions within a query.
- useful when you need to compare fields the same document in more complex manner.
- Use Cases :
  - Filtering documents with multiple conditions.
  - Performing math or string operations during query.
  - Embedding logic within $project, $match, $expr, etc.

Complex Expressions

- Example with $math Operator.

```
db.orders.find({
  $expr: { $gt: ["$quantity", 100] }
});
```

Conditional Query

- A Conditional Query in MongoDB is a query that retrieves documents based on specific conditions or criteria — meaning you only get the documents that match certain logical, comparison, or range rules.

```
db.products.find({
  $and: [
    { price: { $gt: 100 } },
    { quantity: { $lt: 50 } }
  ]
});
```

Conditional Query

- $all => Matches array that Contains all the Specified Elements.
- $elemMatch => Matches documents when at least one array element with Multiple Condition.

Update Operation on any Collection

- it can be done by two main methods :
  - 1. updateOne()
  - 2. updateMany()

- Syntax :

db.collectionName.updateOne({filter},{$set :
{$existingFeild:newvalue},...

});

$unset and $rename can be used rather then $set to Remove and Rename the Feild - It is also one kind of Updation.

Update Operation

- can be done by
  - 1. $push
  - 2. $pop
  - 3. $set
  - 4. $unset
  - 5. $rename etc...

Delete Operation on any Collection

- Deletion is other important Operation which is Quite Easy in Comparision of Other Operation.
- It can also done by two methods.
  - 1. deleteOne()
  - 2. deleteMany()

- NOTE : remove() - is deprecated function which will no longer accessible in future’s version.

1. deleteOne()

- Can Delete One Object!

- Syntax :

db.collection.deleteOne({ key: "value" });

- Example :

db.products.deleteOne({age : 15})

2. deleteMany()

- Can Delete Many Object as One Object also!

- Syntax :

db.collection.deleteMany({ key: "value" });

- Example :

db.products.deleteMany({age : 15})

Thank You

- The Next Notes will Cover : Index, Aggregation, MongoDB ATLAS, MongoDB Compas etc...
- Would Like to Understand to CAP Theorem : CAP Notes