MongoDB

What is MongoDB: MongoDB is a cross-platform, document-oriented database.

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

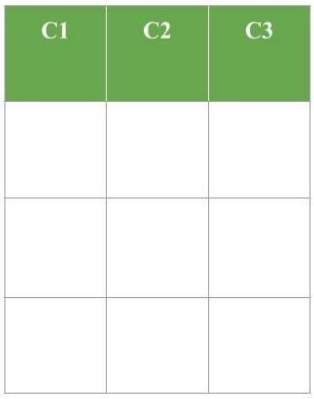
MongoDB is an open-source, document-oriented, and one of the most popular NoSQL database. NoSQL simply means a non-relational database i.e. there is no table-like relational database structure instead there is a totally different mechanism for storing and retrieving data. This format of storage is called **BSON** and is very much similar to JSON.

- ✓ Provides high performance
- ✓ High availability
- ✓ Easy scalability
- ✓ It is no SQL
- ✓ written in C++

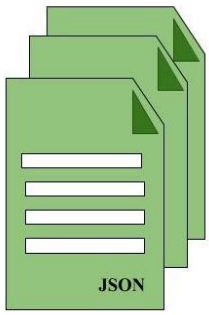
How do Document Databases Work?

A document database has information retrieved or stored in the form of a document or other words semi-structured database. Since they are non-relational, so they are often referred to as NoSQL data.

The document database fetches and accumulates data in forms of key-value pairs but here, the values are called as Documents. A document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML, or any such format. The use of nested documents is also very common. It is very effective as most of the data created is usually in the form of JSON and is unstructured.



Relational Data Model



Document Store Model

Advantages:

Document databases are both natural and flexible for developers to work with.

- ✓ They offer higher productivity and faster evolution for a developer.
- ✓ Document databases are easier to store and query data in a database for a developers by using the same document-model format they use in their application code.
- ✓ For use cases such as catalogs, user profiles, and content management systems where each document is unique and evolves over time, it is much better to use document model.
- ✓ Document databases provide flexible indexing, powerful ad hoc queries, and analytics over collections of documents.

Disadvantages:

- ✓ Handling multiple documents is challenging
- ✓ Aggregation operations may not work accurately.

How MongoDB works?

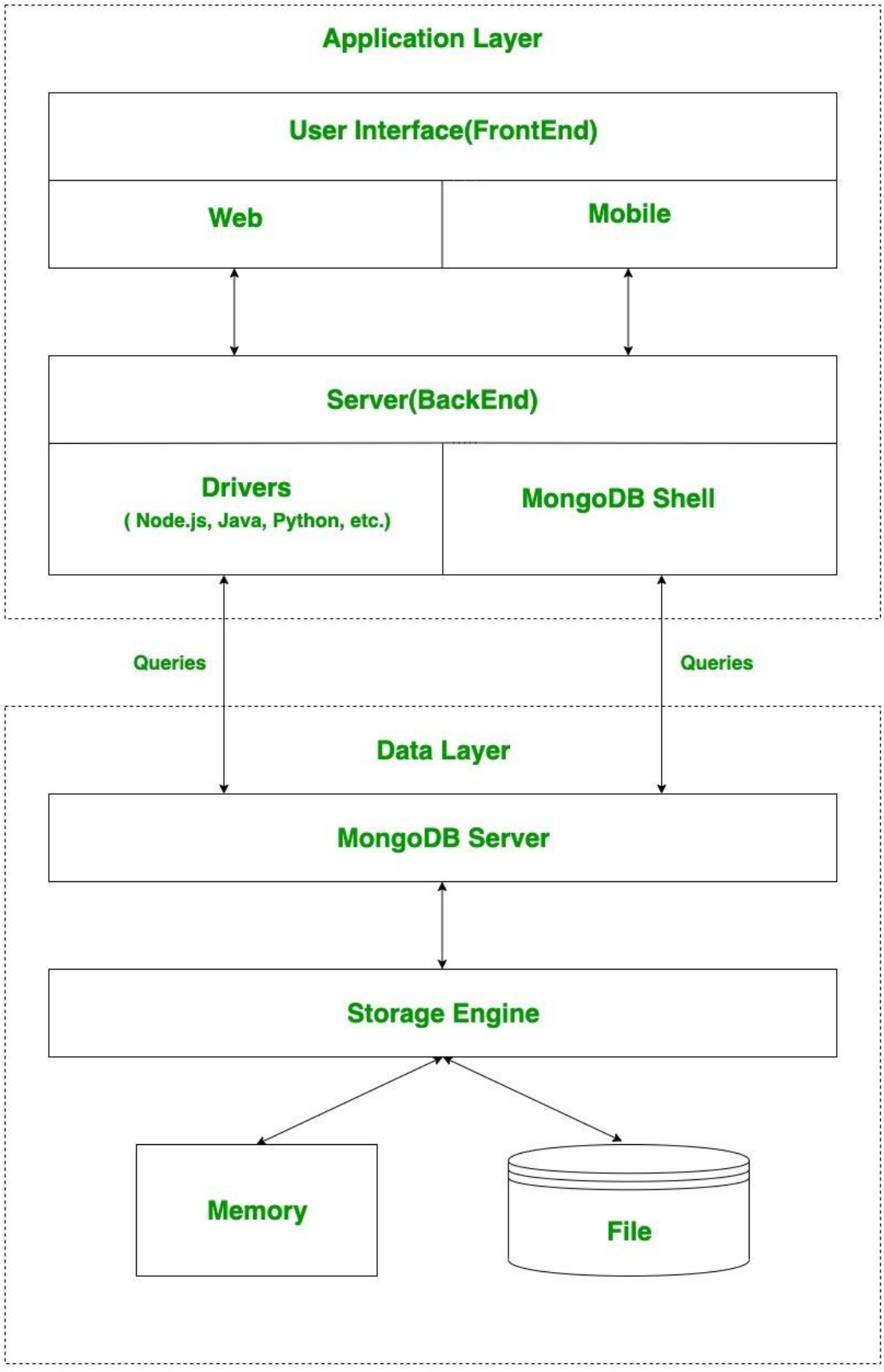
MongoDB is an open-source document-oriented database. It is used to store a larger amount of data and also allows you to work with that data. MongoDB is not based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data, that's why known as NoSQL database. Here, the term 'NoSQL' means 'non-relational'. The format of storage is called BSON (similar to JSON format).

Now, let's see how actually this MongoDB works? But before proceeding to its working, first, let's discuss some important parts of MongoDB –

- ✓ **Drivers:** Drivers are present on your server that are used to communicate with MongoDB. The drivers support by the MongoDB are C, C++, C#, and .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid.
- ✓ **MongoDB Shell:** MongoDB Shell or mongo shell is an interactive JavaScript interface for MongoDB. It is used for queries, data updates, and it also performs administrative operations.
- ✓ **Storage Engine:** It is an important part of MongoDB which is generally used to manage how data is stored in the memory and on the disk. MongoDB can have multiple search engines. You are allowed to use your own search engine and if you don't want to use your own search engine you can use the default search engine, known as *WiredTiger Storage Engine* which is an excellent storage engine, it efficiently works with your data like reading, writing, etc.

Working of MongoDB –

The following image shows how the MongoDB works:


Application Layer
User Interface(FrontEnd)
Web
Mobile
Server(BackEnd)
Drivers
(Node.js, Java, Python, etc.)
MongoDB Shell
Queries
Queries
Data Layer
MongoDB Server
Storage Engine
Memory
File

MongoDB work in two layers –

- ✓ **Application Layer** and
- ✓ **Data layer**

Application Layer is also known as the **Final Abstraction Layer**, it has two-parts, first is a **Frontend (User Interface)** and the second is **Backend (server)**. The frontend is the place where the user uses MongoDB with the help of a Web or Mobile. This web and mobile include web pages, mobile applications, android default applications, IOS applications, etc. The backend contains a server which is used to perform server-side logic and also contain drivers or mongo shell to interact with MongoDB server with the help of queries.

These queries are sent to the MongoDB server present in the **Data Layer**. Now, the MongoDB server receives the queries and passes the received queries to the storage engine. MongoDB server itself does not directly read or write the data to the files or disk or memory. After passing the received queries to the storage engine, the storage engine is responsible to read or write the data in the files or memory basically it manages the data.

Note: The reading and writing from the files are slow as compare to memory.

MongoDB: An introduction

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format).

A simple MongoDB document Structure:

```
{
  title: 'Geeksforgeeks',
  by: 'Harshit Gupta',
  url: 'https://www.geeksforgeeks.org',
  type: 'NoSQL'
}
```

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing

applications. **Modern applications are more networked, social and interactive than ever**. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) i**s not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable**. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

MongoDB can manage:

- ✓ Structured data
- ✓ Semi structured data
- ✓ Un structured data

NoSQL Databases: NoSQL Database is used to refer a non-SQL or non-relational database.

- ✓ No table
- ✓ No row
- ✓ No complex join

NoSQL Behind History:

- ✓ In the early 1970, Flat File Systems are used, that time there was no standard, no format difficult to manage and share.
- ✓ In 1970 Computer scientist Edgar F. Codd proposed a database model to resolve this problem, known as relational database model
- ✓ But later relational database also get a problem that it could not handle big data, due to this problem there was a need of database which can handle every types of problems.
- ✓ The initial development of NoSQL database MongoDB began in 2007 by 10gen, the the company name changed with new name MongoDB Inc.
- ✓ Later in 2009, it is introduced in the market as an open source database server

- ✓ The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.

Basic terminology:

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

Sample Document:

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'value',
   tags: ['value', 'value', 'value'],
   likes: 100,
   comments: [
      {
         user:'value',
         message: 'value',
         dateCreated: new Date(2011,1,20,2,15),
         like: 0
      },
      {
         user:'value',
         message: 'value',
         dateCreated: new Date(2011,1,25,7,45),
         like: 5
      }
   ]
}
```

Default key _id Definition:

_id is a 12 bytes hexadecimal number

7df7	8ad	89	02c

- ✓ First 4 bytes = current timestamp
- ✓ Next 3 bytes = machine id
- ✓ Next 2 bytes = process id
- ✓ Last 3 bytes = incremental VALUE

Advantages:

- ✓ Schema less
- ✓ single object

- ✓ No complex joins
- ✓ Deep query-ability
- ✓ Tuning
- ✓ Ease of scale-out
- ✓ Uses internal memory for storing

Where to Use MongoDB?

- ✓ Big Data
- ✓ Content Management and Delivery
- ✓ Mobile and Social Infrastructure
- ✓ User Data Management
- ✓ Data Hub

MongoDB edition:

- ✓ MongoDB Community Server
- ✓ MongoDB Enterprise Server
- ✓ MongoDB Atlas

Install MongoDB:

- ✓ Download MongoDB Community Server
- ✓ Install MongoDB Community Server on the local machine
- ✓ Download MongoDB Compass
- ✓ Install MongoDB Compass
- ✓ Connect MongoDB with Compass Application

MongoDB - Data Modeling: MongoDB provides two types of data models

- ✓ Embedded Data Model

- ✓ Normalized Data Model

Embedded Data Model: In this model, you can have (embed) all the related data in a single document, it is also known as de-normalized data model.

```
{
  _id: ,
  Emp_ID: "10025AE336"
  Personal_details:{
    First_Name: "Radhika",
    Last_Name: "Sharma",
    Date_Of_Birth: "1995-09-26"
  },
  Contact: {
    e-mail: "radhika_sharma.123@gmail.com",
    phone: "9848022338"
  },
  Address: {
    city: "Hyderabad",
    Area: "Madapur",
    State: "Telangana"
  }
}
```

Normalized Data Model: In this model, you can refer the sub documents in the original document.

```
{
  _id: <ObjectId101>,
  Emp_ID: "10025AE336",
}
```

```
{
  _id: <ObjectId104>,
  empDocID: " ObjectId101",
  city: "Hyderabad",
  Area: "Madapur",
  State: "Telangana"
}
```

```
{
  _id: <ObjectId103>,
  empDocID: " ObjectId101",
  e-mail: "radhika_sharma.123@gmail.com",
  phone: "9848022338",

}
```

```
{
  _id: <ObjectId102>,
  empDocID: " ObjectId101",
  First_Name: "Radhika",
  Last_Name: "Sharma",
  Date_Of_Birth: "1995-09-26",

}
```

Datatypes:

Data Types	Description
String	String is the most commonly used datatype. It is used to store data. A string must be UTF 8 valid in mongodb.
Integer	Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
Boolean	This datatype is used to store boolean values. It just shows YES/NO values.
Double	Double datatype stores floating point values.
Min/Max Keys	This datatype compare a value against the lowest and highest bson elements.
Arrays	This datatype is used to store a list or multiple values into a single key.
Object	Object datatype is used for embedded documents.
Null	It is used to store null values.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object of date and pass the value of date, month, year into it.

MongoDB Shell:

- ✓ The mongo shell is similar to the mysql in MySQL, psql in PostgreSQL, and SQL*Plus in Oracle Database.
- ✓ The mongo shell is an interactive JavaScript interface to MongoDB.
- ✓ The mongo shell is included in the MongoDB installation by default.

```
> Math.max(10,20,30,40,50)
50
>
```

```
> function AddTwo(num1,num2){
... return num1+num2
... }
> AddTwo(100,500)
600
>
```

MongoDB Help:

- ✓ Type **db.help()** in MongoDB client. This will give you a list of commands as shown in the following screenshot.

MongoDB Statistics:

- ✓ To get stats about MongoDB server, type the command db.stats() in MongoDB client. This will show the database name, number of collection and documents in the database.

Create Database:

```
> use MyFirstMongoLists
switched to db MyFirstMongoLists
> db.userList.insert({"name":"Rabbil","city":"dhaka"})
WriteResult({ "nInserted" : 1 })
> show dbs
MyFirstMongoLists  0.000GB
admin              0.000GB
config             0.000GB
local              0.000GB
>
```

Drop Database:

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

Create Collection:

```
> show dbs
School  0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
> use School
switched to db School
> db.createCollection("userList")
{ "ok" : 1 }
>
```

Drop Collection:

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

The insertOne() method allows you to insert a single document into a collection.

```
> show dbs
School  0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
> use School
switched to db School
> db.Students.insertOne({name:"Rabbil",city:"Dhaka"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("61a89fa8543c6cf99f20e405")
}
>
```

The insertMany() allows you to insert multiple documents into a collection.

```
> show dbs
School  0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
> use School
switched to db School
> db.Students.insertMany([ {name:"Rupom",city:"Dhaka"},{name:"Rain",city:"Dhaka"},{name:"Rifat",city:"Rangpur"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61a8a086543c6cf99f20e406"),
                ObjectId("61a8a086543c6cf99f20e407"),
                ObjectId("61a8a086543c6cf99f20e408")
        ]
}
>
```

MongoDB find:

The findOne() returns a single document from a collection that satisfies the specified condition.

```
> use School
switched to db School
> db.Students.findOne({"name":"Rifat"})
{
        "_id" : ObjectId("61a8a086543c6cf99f20e408"),
        "name" : "Rifat",
        "city" : "Rangpur"
}
>
```

The find() method finds the documents that satisfy a specified condition and returns a cursor to the matching documents.

```
> db.Students.find()
{ "_id" : ObjectId("61a89fa8543c6cf99f20e405"), "name" : "Rabbil", "city" : "Dhaka" }
{ "_id" : ObjectId("61a8a086543c6cf99f20e406"), "name" : "Rupom", "city" : "Dhaka" }
{ "_id" : ObjectId("61a8a086543c6cf99f20e407"), "name" : "Rain", "city" : "Dhaka" }
{ "_id" : ObjectId("61a8a086543c6cf99f20e408"), "name" : "Rifat", "city" : "Rangpur" }
>
```

MongoDB Projection:

In MongoDB, projection means selecting only the necessary data rather than selecting whole of the data of a document.

```
> db.Students.find({},{name:1})
{ "_id" : ObjectId("61a89fa8543c6cf99f20e405"), "name" : "Rabbil" }
{ "_id" : ObjectId("61a8a086543c6cf99f20e406"), "name" : "Rupom" }
{ "_id" : ObjectId("61a8a086543c6cf99f20e407"), "name" : "Rain" }
{ "_id" : ObjectId("61a8a086543c6cf99f20e408"), "name" : "Rifat" }
>
```

```
> db.Students.find({},{_id:0,name:1})
{ "name" : "Rabbil" }
{ "name" : "Rupom" }
{ "name" : "Rain" }
{ "name" : "Rifat" }
>
```

Query Operators:

- ✓ **$eq:** Equal To Operator
- ✓ **$lt:** Less Than Operator
- ✓ **$lte:** Less Than or Equal To Operator
- ✓ **$gt:** Greater Than Operator
- ✓ **$gte:** Greater Than or Equal To Operator
- ✓ **$ne:** Not Equal To Operator
- ✓ **$in:** In Operator
- ✓ **$nin:** Not In Operator

- ❖ db.Products.find({price:{$eq:"1000"}})
- ❖ db.Products.find({price:{$lt:"1000"}})
- ❖ db.Products.find({price:{$lte:"1000"}})
- ❖ db.Products.find({price:{$gt:"1000"}})
- ❖ db.Products.find({price:{$gte:"1000"}})
- ❖ db.Products.find({price:{$ne:"1000"}})

- db.Products.find({price:{$in:["100","200","3000"]}},{price:1})
- db.Products.find({price:{$nin:["100","200","3000"]}},{price:1})

Logical Operators:

- ✓ **$and:** Logical AND Opeartor
- ✓ **$or:** Logical OR Operator

Limit Records:

- ✓ To limit the records in MongoDB, you need to use **limit()**

```
>db.COLLECTION_NAME.find().limit(NUMBER)
```

Sort Records:

- ✓ To sort documents in MongoDB, you need to use **sort()** method.

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

•db.Products.find({},{price:1}).sort({price:-1})

- ✓ 1 is used for ascending order
- ✓ -1 is used for descending order.

Updating documents:

The update() method updates the values in the existing document.

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
```

```
db.Products.update({id:2868},{$set:{'title':'New MongoDB Tutorial'}})
```

Delete Document

MongoDB's **remove()** method is used to remove a document from the collection.

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```