**What is MongoDB**

[MongoDB](https://www.javatpoint.com/mongodb-tutorial) is an open-source document database that provides high performance, high availability, and automatic scaling.

In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

The manufacturing company 10gen has defined MongoDB as:

"MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen

Example of Document-Oriented Database

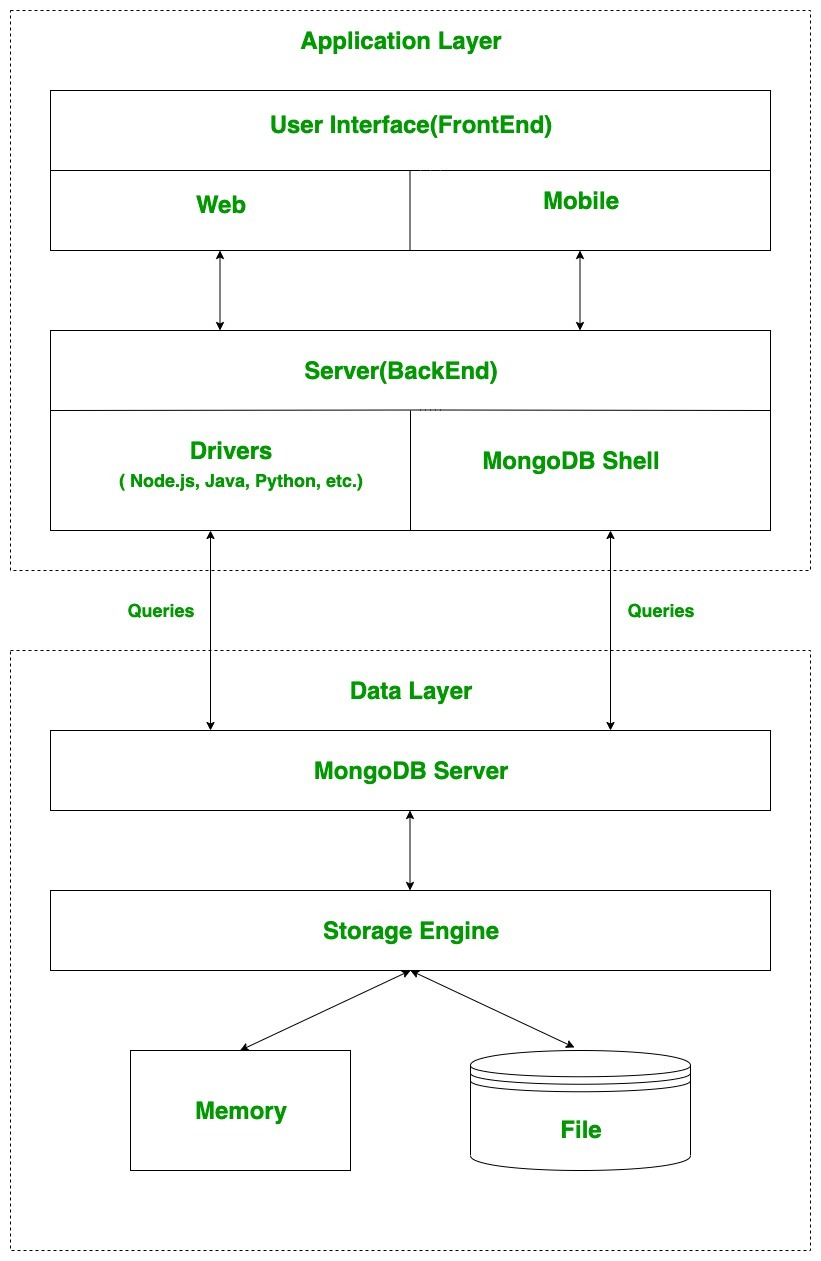
1. FirstName = "John",
2. Address = "Detroit",
3. Spouse = [{**Name**: "Angela"}].
4. FirstName ="John",
5. Address = "Wick"

**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 –**



|  |  |
| --- | --- |
| RDBMS | MongoDB |
| It is a [relational database](https://www.geeksforgeeks.org/relational-model-in-dbms/). | It is a non-relational and document-oriented database. |
| Not suitable for hierarchical data storage. | Suitable for [hierarchical data storage](https://www.geeksforgeeks.org/hierarchical-model-in-dbms/#:~:text=In%20a%20hierarchical%20model%2C%20data,parent%20record%20and%20many%20children.). |
| 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](https://www.geeksforgeeks.org/sql-injection-2/). |
| It centers around [ACID](https://www.geeksforgeeks.org/acid-properties-in-dbms/) properties (Atomicity, Consistency, Isolation, and Durability). | It centers around the [CAP theorem](https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/) (Consistency, Availability, and Partition tolerance). |
| 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](https://www.geeksforgeeks.org/json/) query language along with [SQL](https://www.geeksforgeeks.org/sql-tutorial/). |

**Types of NoSQL Databases**

* Document-based databases
* Key-value stores
* Column-oriented databases
* Graph-based databases

**Document-Based Database:**

The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, BSON, or XML documents

**Key-Value Stores:**

A key-value store is a nonrelational database. The simplest form of a NoSQL database is a key-value store. Every data element in the database is stored in key-value pairs. The data can be retrieved by using a unique key allotted to each element in the database. The values can be simple data types like strings and numbers or complex objects.

**Column Oriented Databases:**

A column-oriented database is a non-relational database that stores the data in columns instead of rows. That means when we want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data.

**Graph-Based databases:**

Graph-based databases focus on the relationship between the elements. It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships.

**MongoDB Cursor**

The MongoDB cursor is a pointer that references the documents of the collection returned by the find() method.

The cursor is used to access the documents. By default, the cursor iterates automatically, but can also be iterated manually by the user.

**JSON VS BSON**

JSON, short for **JavaScript Object Notation**, makes sharing data simple and straightforward. Created by Douglas Crockford, it’s designed for easy reading and writing by humans, and easy parsing and generating by computers. Its main goal was to make a text format that’s good at showing simple data like lists and text, and really useful for websites.

JSON is special because it’s very clear and easy to use, and it uses a **“.json”** file ending to show that a file is in this format. This makes JSON great for both people and programs to work with.

**JSON Syntax:**

**1. Using ‘Objects’**

Objects in JSON are collections of key/value pairs enclosed in curly braces {}.

Each key is a string (enclosed in double quotes ") followed by a colon :, and the key/value pairs are separated by commas (,).

Example: {"firstName": "John", "lastName": "Doe", "age": 30}

**2. Using ‘Arrays’**

Arrays are ordered lists of values, enclosed in square brackets [].

Values within an array are separated by commas (,).

Example: ["apple", "banana", "cherry"]

**BSON**

BSON stands for Binary JSON. It is a binary file format that is used to store serialized JSON documents in a binary-encoded format. It was developed in 2009 by MongoDB. The MongoDB database had several scalar data formats that were of special interest only for MongoDB, hence they developed the BSON data format to be used while transferring files over the network. Although the format was developed specifically for MongoDB, it can be used anywhere as per business requirements independently.

It has several similarities with JSON for instance BSON too supports nested documents and arrays within other documents, but yet has a lot of striking differences. Refer to this post to read more about the difference between JSON and BSON.

**Sample BSON document**

Consider the following JSON document:

{

"hello" : "world"

}

It’s BSON equivalent will be:

\x16\x00\x00\x00 // Size of the Document

\x02 // 0x02 = type String

hello\x00 // field name

\x06\x00\x00\x00world\x00 // field value

\x00 // Used to represent end of object

**MongoDB Cursor**

The MongoDB cursor is a pointer that references the documents of the collection returned by the find() method.

The cursor is used to access the documents. By default, the cursor iterates automatically, but can also be iterated manually by the user.

**Using the next() Method**

We can also use the next() cursor method to access the next document. Let us discuss with the help of an example:

Count cursor:

In order to get the correct documents, we need to know how many documents are present for that collection. To get that we can use the count() method which returns the total number of documents present in the given collection.

**Cursor Limit:**

The limit() method helps to fetch limited records from a collection. Suppose we have multiple documents, but we want to have the topmost or only 2 documents, then by using the limit() method, we can achieve that.

**Cursor size:**

The cursor.size() method will be helpful to return a count of the number of documents that got as the output from the db.collection.find() query after applying any cursor.skip() and cursor.limit() methods. Hence, it is mentioned as it has applied cursor.skip() and cursor.limit() methods.

**Cursor sort:**

Usually while verifying documents, if the output is in sorted order, either in ascending or descending order, it will be easier. So we use the sort() method to sort the documents. If you want to sort the documents in ascending, then set the value of the field to 1 and in descending, then set -1.

**Cursor.toArray():**

In order to have an array that contains all documents returned by the cursor, we can use the toArray() method.

**MongoDB – Comparison Query Operators**

Comparison Query Operators in MongoDB are used to filter documents based on some specific criteria within their fields.

MongoDB uses various comparison query operators to compare the values of the documents. The following table contains the comparison query operators:

|  |  |
| --- | --- |
| **Operators** | **Description** |
| [**$eq**](https://www.geeksforgeeks.org/mongodb-equality-operator-eq/) | Matches the values of the fields that are equal to a specified value. |
| [**$ne**](https://www.geeksforgeeks.org/mongodb-inequality-operator-ne/) | Matches all values of the field that are not equal to a specified value. |
| [**$gt**](https://www.geeksforgeeks.org/mongodb-greater-than-operator-gt/) | Matches values of the fields that are greater than a specified value. |
| [**$gte**](https://www.geeksforgeeks.org/mongodb-greater-than-equals-to-operator-gte/) | Matches values of the fields that are greater than equal to the specified value. |
| [**$lt**](https://www.geeksforgeeks.org/mongodb-less-than-operator-lt/) | Matches values of the fields that are less than a specified value |
| [**$lte**](https://www.geeksforgeeks.org/mongodb-less-than-equals-to-operator-lte/) | Matches values of the fields that are less than equal to the specified value |
| [**$in**](https://www.geeksforgeeks.org/mongodb-in-operator/) | Matches any of the values specified in an array. |
| [**$nin**](https://www.geeksforgeeks.org/mongodb-nin-operator/) | Matches none of the values specified in an array. |
|  |  |

MongoDB – Logical Query Operators

MongoDB supports logical query operators. These operators are used for filtering the data and getting precise results based on the given conditions. The following table contains the comparison query operators:

|  |  |
| --- | --- |
| Operator | Description |
| $and | It is used to join query clauses with a logical AND and return all documents that match the given conditions of both clauses. |
| $or | It is used to join query clauses with a logical OR and return all documents that match the given conditions of either clause. |
| $not | It is used to invert the effect of the query expressions and return documents that does not match the query expression. |
| $nor | It is used to join query clauses with a logical NOR and return all documents that fail to match both clauses. |