**MongoDB Notes:**

MongoDB is a No SQL database. It is an open-source, cross-platform, document-oriented database written in C++.

Our MongoDB tutorial includes all topics of MongoDB database such as insert documents, update documents, delete documents, query documents, projection, sort() and limit() methods, create collection, drop collection etc. There are also given MongoDB interview questions to help you better understand the MongoDB database.

Prerequisite

Before learning MongoDB, you must have the basic knowledge of SQL and OOPs.

# **What is MongoDB**

MongoDB 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 manufacture.

The manufacturing company 10 gen has given the definition of Mongo DB:

"Mongo DB is scalable, open source, high performance, document oriented database." - 10 gen

MongoDB was designed to work with commodity servers. Now it is used by company of all sizes, across all industry.

## Purpose to build MongoDB

This may be a very genuine question that - "what was the need of MongoDB although there were many databases in action?"

This is a very simple answer:

All the modern applications require big data, fast features development, flexible deployment and the older database systems not enough competent, so the MongoDB was obviously needed.

**Main purpose to build MongoDB:**

* Scalability
* Performance
* High Availability
* Scaling from single server deployments to large, complex multi-site architectures.

## Key points of MongoDB

* Develop Faster
* Deploy Easier
* Scale Bigger

# **History of MongoDB**

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to **window azure.**

"Window azure is a cloud computing platform and infrastructure, created by Microsoft, to build, deploy and manage applications and service through a global network."

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform As A Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

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

MongoDB2.4.9 was the latest and stable version which was released on January 10, 2014.

First of all, we should know what is document oriented database?

## Example of document oriented database

MongoDB is a document oriented database. It is a key feature of MongoDB. It offers a document oriented storage. It is very simple you can program it easily.

MongoDB stores data as documents, so it is known as document oriented database.

1. FirstName = "Ajeet",
2. Address = "Laxmi Nagar",
3. Spouse = [{**Name**: "Chaaru"}].
4. FirstName ="Ravi",
5. Address = "Loni"

There are two different documents (separated by ".").

Storing data in this manner is called as document oriented database.

Mongo DB falls into a class of databases that calls Document Oriented Databases. There is also a broad category of database known as No SQL Databases

# **Features of MongoDB**

These are some important features of MongoDB:

**1. Support ad hoc queries**

In MongoDB, you can search by field, range query and it also supports regular expression searches.

**2. Indexing**

You can index any field in a document.

**3. Replication**

MongoDB supports Master Slave replication.

A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes)

**4. Duplication of data**

MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure.

**5. Load balancing**

It has an automatic load balancing configuration because of data placed in shards.

**6. Supports map reduce and aggregation tools.**

**7. Uses JavaScript instead of Procedures.**

**8. It is a schema-less database written in C++.**

**9. Provides high performance.**

**10. Stores files of any size easily without complicating your stack.**

**11. Easy to administer in the case of failures.**

**12. It also supports:**

* JSON data model with dynamic schemas
* Auto-sharding for horizontal scalability
* Built in replication for high availability

Now a day many companies using MongoDB to create new types of applications, improve performance and availability.

# **NoSQL Databases**

We know that MongoDB is a NoSQL Database, so it is very necessary to know about NoSQL Database to understand MongoDB throughly .

## What is NoSQL Database

Databases can be divided in 3 types:

1. RDBMS (Relational Database Management System)
2. OLAP (Online Analytical Processing)
3. NoSQL (recently developed database)

## NoSQL Database

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

It provides a mechanism for storage and retrieval of data other than tabular relations model used in relational databases. NoSQL database doesn't use tables for storing data. It is generally used to store big data and real-time web applications.

## History behind the creation of NoSQL Databases

In the early 1970, Flat File Systems are used. Data were stored in flat files and the biggest problems with flat files are each company implement their own flat files and there are no standards. It is very difficult to store data in the files, retrieve data from files because there is no standard way to store data.

Then the relational database was created by E.F. Codd and these databases answered the question of having no standard way to store data. 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 then NoSQL database was developed.

## Advantages of NoSQL

* It supports query language.
* It provides fast performance.
* It provides horizontal scalability.

# **MongoDB advantages over RDBMS**

In recent days, MongoDB is a new and popularly used database. It is a document based, non relational database provider.

Although it is 100 times faster than the traditional database but it is early to say that it will broadly replace the traditional RDBMS. But it may be very useful in term to gain performance and scalability.

A Relational database has a typical schema design that shows number of tables and the relationship between these tables, while in MongoDB there is no concept of relationship.

## MongoDB Advantages

* **MongoDB is schema less**. It is a document database in which one collection holds different documents.
* There may be **difference between number of fields, content and size of the document** from one to other.
* **Structure of a single object is clear** in MongoDB.
* There are **no complex joins** in MongoDB.
* MongoDB provides the **facility of deep query** because it supports a powerful dynamic query on documents.
* It is very **easy to scale**.
* It **uses internal memory for storing working sets** and this is the reason of its fast access.

## Distinctive features of MongoDB

* Easy to use
* Light Weight
* Extremely faster than RDBMS

## Where MongoDB should be used

* Big and complex data
* Mobile and social infrastructure
* Content management and delivery
* User data management
* Data hub

## Performance analysis of MongoDB and RDBMS

* In relational database (RDBMS) tables are using as storing elements, while in MongoDB collection is used.
* In the RDBMS, we have multiple schema and in each schema we create tables to store data while, MongoDB is a document oriented database in which data is written in BSON format which is a JSON like format.
* MongoDB is almost 100 times faster than traditional database systems.

# **MongoDB Datatypes**

Following is a list of usable data types in MongoDB.

|  |  |
| --- | --- |
| **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. |

# **How to install MongoDB on Windows**

Firstly you will have to download the latest release of MongoDB:

## How to download MongoDB

You can download an appropriate version of MongoDB which your system supports, from the link **"http://www.mongodb.org/downloads"** to install the MongoDB on Windows. You should choose correct version of MongoDB acording to your computer's Window. If you are not sure what Window version are you using, open your command prompt and execute this command:

1. C:\ wmic os get osarchitecture

OSArchitecture

64 bit

C:\ >

**Note:** MongoDB does not support Window XP.

## MongoDB for Windows Server 2008 R2 edition

This version of MongoDB runs only on Window Server 2008 R2, Window7 64 bit, and the newer version of windows. You can't operate it on older version of windows.

## MongoDb for 64 bit Windows

This version of MongoDB runs only on newer version of Windows contains 64 bit operating system.

for example: Window Server 2008 R2, Window 7 64 bit etc.

## MongoDb for 32 bit Windows

This version of MongoDB runs on only 32 bit windows. 32 bit version of MongoDB is generally used in testing and development purposes because it supports databases smaller than 2 GB.

## How to install the downloaded file

In Window explorer, locate the downloaded MongoDB msi file, double click on that file and follow the instructions appears on the screen. These instructions will guide you to complete the installation process.

**Note:** If you want to move the MongoDB folder from default position to another position, it is necessary to issue the move command as an administrator. let us take an example to move the folder to C : \mongodb:

1. **Select** Start Menu > All Programs > Accessories

You can install MongoDB in any folder because it is self contained and does not have any other system dependency.

## How to set up the MongoDB environment

A data directory is required in MongoDB to store all the information. Its by default data directory path is \data\db. you can create this folder by command prompt.

1. md\data\db

**For example:**

If you want to start MongoDB, run mongod.exe

You can do it from command prompt.

1. C:\Program Files\MongoDB\bin\mongod.exe

This will start the mongoDB database process. If you get a message "waiting for connection" in the console output, it indicates that the mongodb.exe process is running successfully.

**For example:**

When you connect to the MongoDB through the mongo.exe shell, you should follow these steps:

1. Open another command prompt.

2. At the time of connecting, specify the data directory if necessary.

**Note:** If you use the default data directory while MongoDB installation, there is no need to specify the data directory.

**For example:**

1. C:\mongodb\bin\mongo.exe

If you use the different data directory while MongoDB installation, specify the directory when connecting.

**For example:**

1. C:\mongodb\bin\mongod.exe-- dbpath d:\test\mongodb\data

If you have spaces in your path, enclose the entire path in double space.

**For example:**

1. C:\mongodb\bin\mongod.exe-- dbpath  "d: \ test\mongodb\data"

## How to configure directory and files

First to create a configuration file and a directory path for MongoDB log output after that create a specific directory for MongoDB log files.

# **MongoDB Shell**

MongoDB have a JavaScript shell that allows interaction with MongoDB instance from the command line.

If you want to create a table, you should name the table and define its column and each column's data type.

The shell is useful for performing administrative functions and running instances.

## How to run the shell

To start the shell, open command prompt, run it as a administrator then run the mongo executable:

1. $ mongo

MongoDB shell version: 2.4.0

Connecting to: test

You should start mongoDB before starting the shell because shell automatically attempt to connect to a MongoDB server on startup.

The shell is a full-featured JavaScript interpreter. It is capable of running Arbitrary JavaScript program.

**Let us take a simple mathematical program:**

1. >x= 100
2. 100
3. >x/ 5;
4. 20

**You can also use the JavaScript libraries**

1. > "Hello, World!".replace("World", "MongoDB");

Hello, MongoDB!

**You can even define and call JavaScript functions**

1. > **function** factorial (n) {
3. ... if (n <= 1) **return** 1;
5. ... **return** n \* factorial(n - 1);
6. ... }
8. > factorial (5);
10. 120

**Note:** You can create multiple commands.

When you press "Enter", the shell detect whether the JavaScript statement is complete or not.

If the statement is not completed, the shell allows you to continue writing it on the next line. If you press "Enter" three times in a row, it will cancel the half-formed command and get you back to the > - prompt.

# **Data Modeling in MongoDB**

In MongoDB, data has a flexible schema. It is totally different from SQL database where you had to determine and declare a table's schema before inserting data. MongoDB collections do not enforce document structure.

The main challenge in data modeling is balancing the need of the application, the performance characteristics of the database engine, and the data retrieval patterns.

## Consider the following things while designing the schema in MongoDB

* Always design schema according to user requirements.
* Do join on write operations not on read operations.
* Objects which you want to use together, should be combined into one document. Otherwise they should be separated (make sure that there should not be need of joins).
* Optimize your schema for more frequent use cases.
* Do complex aggregation in the schema.
* You should duplicate the data but in a limit, because disc space is cheaper than compute time.

**For example:**

let us take an example of a client who needs a database design for his website. His website has the following requirements:

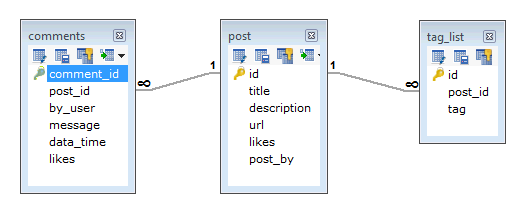
Every post is distinct (contains unique title, description and url).

Every post can have one or more tags.

Every post has the name of its publisher and total number of likes.

Each post can have zero or more comments and the comments must contain user name, message, data-time and likes.

For the above requirement, a minimum of three tables are required in RDBMS.



But in MongoDB, schema design will have one collection post and has the following structure:

{

\_id: POST\_ID

title: TITLE\_OF\_POST,

description: POST\_DESCRIPTION,

by: POST\_BY,

url: URL\_OF\_POST,

tags: [TAG1, TAG2, TAG3],

likes: TOTAL\_LIKES,

comments: [

{

user: 'COMMENT\_BY',

message: TEXT,

datecreated: DATE\_TIME,

like: LIKES

},

{

user: 'COMMENT\_BY',

message: TEST,

dateCreated: DATE\_TIME,

like: LIKES

}}}

**Database:**

# **MongoDB Create Database**

**Use Database method:**

There is no create database command in MongoDB. Actually, MongoDB do not provide any command to create database.

It may be look like a weird concept, if you are from traditional SQL background where you need to create a database, table and insert values in the table manually.

Here, in MongoDB you don't need to create a database manually because MongoDB will create it automatically when you save the value into the defined collection at first time.

You also don't need to mention what you want to create, it will be automatically created at the time you save the value into the defined collection.

## How and when to create database

If there is no existing database, the following command is used to create a new database.

**Syntax:**

1. use DATABASE\_NAME

If the database already exists, it will return the existing database.

Let' take an example to demonstrate how a database is created in MongoDB. In the following example, we are going to create a database "cts".

**See this example**

1. >use cts

Swithched to db cts

To **check the currently selected database**, use the command db:

1. >db

cts

To **check the database list**, use the command show dbs:

1. >show dbs

local 0.078GB

Here, your created database "javatpointdb" is not present in the list, **insert at least one document** into it to display database:

1. >db.cts.**insert**({"name":"cts123"})

WriteResult({ "nInserted": 1})

1. >show dbs

cts 0.078GB

local 0.078GB

# **MongoDB Drop Database**

The dropDatabase command is used to drop a database. It also deletes the associated data files. It operates on the current database.

**Syntax:**

1. db.dropDatabase()

This syntax will delete the selected database. In the case you have not selected any database, it will delete default "test" database.

To **check the database list**, use the command show dbs:

1. >show dbs

cts 0.078GB

local 0.078GB

If you want to **delete the database "cts"**, use the dropDatabase() command as follows:

1. >use cts

switched to the db cts

1. >db.dropDatabase()

{ "dropped": "cts", "ok": 1}

Now check the list of databases:

1. >show dbs

local 0.078GB

**Collection:**

# **MongoDB Create Collection**

In MongoDB, db.createCollection(name, options) is used to create collection. But usually you don?t need to create collection. MongoDB creates collection automatically when you insert some documents. It will be explained later. First see how to create collection:

**Syntax:**

1. db.createCollection(**name**, options)

Here,

**Name:** is a string type, specifies the name of the collection to be created.

**Options:** is a document type, specifies the memory size and indexing of the collection. It is an optional parameter.

Following is the list of options that can be used.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Capped | Boolean | (Optional) If it is set to true, enables a capped collection. Capped collection is a fixed size collecction that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify size parameter also. |
| AutoIndexID | Boolean | (Optional) If it is set to true, automatically create index on ID field. Its default value is false. |
| Size | Number | (Optional) It specifies a maximum size in bytes for a capped collection. Ifcapped is true, then you need to specify this field also. |
| Max | Number | (Optional) It specifies the maximum number of documents allowed in the capped collection. |

Let's take an **example to create collection**. In this example, we are going to create a collection name SSSIT.

1. >use test

switched to db test

1. >db.createCollection("SSSIT")

{ "ok" : 1 }

To **check the created collection**, use the command "show collections".

1. >show collections

SSSIT

## How does MongoDB create collection automatically

MongoDB creates collections automatically when you insert some documents. For example: Insert a document named seomount into a collection named SSSIT. The operation will create the collection if the collection does not currently exist.

1. >db.SSSIT.**insert**({"name" : "seomount"})
2. >show collections
3. SSSIT

If you want to see the inserted document, use the find() command.

Syntax:

db.collection\_name.find()

# **MongoDB Drop collection**

In MongoDB, db.collection.drop() method is used to drop a collection from a database. It completely removes a collection from the database and does not leave any indexes associated with the dropped collections.

The db.collection.drop() method does not take any argument and produce an error when it is called with an argument. This method removes all the indexes associated with the dropped collection.

**Syntax:**

1. db.COLLECTION\_NAME.**drop**()

## MongoDB Drop collection example

Let's take an example to drop collection in MongoDB.

First **check the already existing collections** in your database.

1. >use mydb

Switched to db mydb

1. > show collections

SSSIT

system.indexes

**Note:** Here we have a collection named SSSIT in our database.

Now **drop the collection** with the name SSSIT:

1. >db.SSSIT.**drop**()

True

Now **check the collections** in the database:

1. >show collections

System.indexes

Now, there are no existing collections in your database.

**CRUD documents:**

# **MongoDB insert documents**

In MongoDB, the**db.collection.insert()** method is used to add or insert new documents into a collection in your database.

**Upsert**

There are also two methods "db.collection.update()" method and "db.collection.save()" method used for the same purpose. These methods add new documents through an operation called upsert.

Upsert is an operation that performs either an update of existing document or an insert of new document if the document to modify does not exist.

**Syntax**

1. >db.COLLECTION\_NAME.**insert**(document)

Let?s take an example to demonstrate how to insert a document into a collection. In this example we insert a document into a collection named javatpoint. This operation will automatically create a collection if the collection does not currently exist.

## Example

1. db.javatpoint.**insert**(
2. {
3. course: "java",
4. details: {
5. duration: "6 months",
6. Trainer: "Sonoo jaiswal"
7. },
8. Batch: [ { **size**: "Small", qty: 15 }, { **size**: "Medium", qty: 25 } ],
9. category: "Programming language"
10. }
11. )

After the successful insertion of the document, the operation will return a WriteResult object with its status.

**Output:**

WriteResult({ "nInserted" : 1 })

Here the **nInserted** field specifies the number of documents inserted. If an error is occurred then the **WriteResult** will specify the error information.

## Check the inserted documents

If the insertion is successful, you can view the inserted document by the following query.

1. >db.javatpoint.find()

You will get the inserted document in return.

**Output:**

{ "\_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" : "java", "details" :

{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" }, "Batch" :

[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty" : 25 } ],

"category" : "Programming language" }

**Note:** Here, the ObjectId value is generated by MongoDB itself. It may differ from the one shown.

## MongoDB insert multiple documents

If you want to insert multiple documents in a collection, you have to pass an array of documents to the db.collection.insert() method.

## Create an array of documents

Define a variable named Allcourses that hold an array of documents to insert.

1. var Allcourses =
2. [
3. {
4. Course: "Java",
5. details: { Duration: "6 months", Trainer: "Sonoo Jaiswal" },
6. Batch: [ { **size**: "Medium", qty: 25 } ],
7. category: "Programming Language"
8. },
9. {
10. Course: ".Net",
11. details: { Duration: "6 months", Trainer: "Prashant Verma" },
12. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Medium", qty: 10 }, ],
13. category: "Programming Language"
14. },
15. {
16. Course: "Web Designing",
17. details: { Duration: "3 months", Trainer: "Rashmi Desai" },
18. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Large", qty: 10 } ],
19. category: "Programming Language"
20. }
21. ];

## Inserts the documents

Pass this Allcourses array to the db.collection.insert() method to perform a bulk insert.

1. > db.javatpoint.**insert**( Allcourses );

After the successful insertion of the documents, this will return a BulkWriteResult object with the status.

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 3,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

**Note:**Here the nInserted field specifies the number of documents inserted. In the case of any error during the operation, the **BulkWriteResult**will specify that error.

You can check the inserted documents by using the following query:

1. >db.javatpoint.find()

## Insert multiple documents with Bulk

In its latest version of MongoDB (MongoDB 2.6) provides a Bulk() API that can be used to perform multiple write operations in bulk.

You should follow these steps to insert a group of documents into a MongoDB collection.

## Initialize a bulk operation builder

First initialize a bulk operation builder for the collection javatpoint.

1. var bulk = db.javatpoint.initializeUnorderedBulkOp();

This operation returns an unorder operations builder which maintains a list of operations to perform .

## Add insert operations to the bulk object

1. bulk.**insert**(
2. {
3. course: "java",
4. details: {
5. duration: "6 months",
6. Trainer: "Sonoo jaiswal"
7. },
8. Batch: [ { **size**: "Small", qty: 15 }, { **size**: "Medium", qty: 25 } ],
9. category: "Programming language"
10. }
11. );

## Execute the bulk operation

Call the execute() method on the bulk object to execute the operations in the list.

1. bulk.**execute**();

After the successful insertion of the documents, this method will return a **BulkWriteResult** object with its status.

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

Here the nInserted field specifies the number of documents inserted. In the case of any error during the operation, the **BulkWriteResult**will specify that error.

# **MongoDB update documents**

In MongoDB, update() method is used to update or modify the existing documents of a collection.

**Syntax:**

1. db.COLLECTION\_NAME.**update**(SELECTIOIN\_CRITERIA, UPDATED\_DATA)

## Example

Consider an example which has a collection name javatpoint. Insert the following documents in collection:

1. db.javatpoint.**insert**(
2. {
3. course: "java",
4. details: {
5. duration: "6 months",
6. Trainer: "Sonoo jaiswal"
7. },
8. Batch: [ { **size**: "Small", qty: 15 }, { **size**: "Medium", qty: 25 } ],
9. category: "Programming language"
10. }
11. )

After successful insertion, check the documents by following query:

1. >db.javatpoint.find()

**Output:**

{ "\_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" : "java", "details" :

{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" }, "Batch" :

[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty" : 25 } ],

"category" : "Programming language" }

**Update the existing course "java" into "android":**

1. >db.javatpoint.**update**({'course':'java'},{$**set**:{'course':'android'}})

### **Check the updated document in the collection:**

1. >db.javatpoint.find()

**Output:**

{ "\_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" : "android", "details" :

{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" }, "Batch" :

[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty" : 25 } ],

"category" : "Programming language" }

# **MongoDB Delete documents**

In MongoDB, the db.colloction.remove() method is used to delete documents from a collection. The remove() method works on two parameters.

**1. Deletion criteria:** With the use of its syntax you can remove the documents from the collection.

**2. JustOne:** It removes only one document when set to true or 1.

**Syntax:**

1. db.collection\_name.remove (DELETION\_CRITERIA)

## Remove all documents

If you want to remove all documents from a collection, pass an empty query document {} to the remove() method. The remove() method does not remove the indexes.

Let's take an example to demonstrate the remove() method. In this example, we remove all documents from the "javatpoint" collection.

1. db.javatpoint.remove({})

## Remove all documents that match a condition

If you want to remove a document that match a specific condition, call the remove() method with the <query> parameter.

The following example will remove all documents from the javatpoint collection where the type field is equal to programming language.

1. db.javatpoint.remove( { type : "programming language" } )

## Remove a single document that match a condition

If you want to remove a single document that match a specific condition, call the remove() method with justOne parameter set to true or 1.

The following example will remove a single document from the javatpoint collection where the type field is equal to programming language.

1. db.javatpoint.remove( { type : "programming language" }, 1 )

# **MongoDB Query documents**

In MongoDB, the **db.collection.find()** method is used to retrieve documents from a collection. This method returns a cursor to the retrieved documents.

The db.collection.find() method reads operations in mongoDB shell and retrieves documents containing all their fields.

#### Note: You can also restrict the fields to return in the retrieved documents by using some specific queries. For example: you can use the db.collection.findOne() method to return a single document. It works same as the db.collection.find() method with a limit of 1.

**Syntax:**

1. db.COLLECTION\_NAME.find({})

## Select all documents in a collection:

To retrieve all documents from a collection, put the query document ({}) empty. It will be like this:

1. db.COLLECTION\_NAME.find()

**For example:** If you have a collection name "canteen" in your database which has some fields like foods, snacks, beverages, price etc. then you should use the following query to select all documents in the collection "canteen".

1. db.canteen.find()

# **MongoDB limit() Method**

In MongoDB, limit() method is used to limit the fields of document that you want to show. Sometimes, you have a lot of fields in collection of your database and have to retrieve only 1 or 2. In such case, limit() method is used.

The MongoDB limit() method is used with find() method.

**Syntax:**

1. db.COLLECTION\_NAME.find().limit(NUMBER)

## Scenario:

Consider an example which has a collection name javatpoint.

This collection has following fields within it.

1. [
2. {
3. Course: "Java",
4. details: { Duration: "6 months", Trainer: "Sonoo Jaiswal" },
5. Batch: [ { **size**: "Medium", qty: 25 } ],
6. category: "Programming Language"
7. },
8. {
9. Course: ".Net",
10. details: { Duration: "6 months", Trainer: "Prashant Verma" },
11. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Medium", qty: 10 }, ],
12. category: "Programming Language"
13. },
14. {
15. Course: "Web Designing",
16. details: { Duration: "3 months", Trainer: "Rashmi Desai" },
17. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Large", qty: 10 } ],
18. category: "Programming Language"
19. }
20. ];

Here, you have to display only one field by using limit() method.

## Example

1. db.javatpoint.find().limit(1)

After the execution, you will get the following result

Output:

{ "\_id" : ObjectId("564dbced8e2c097d15fbb601"), "Course" : "Java", "details" : {

"Duration" : "6 months", "Trainer" : "Sonoo Jaiswal" }, "Batch" : [ { "size" :

"Medium", "qty" : 25 } ], "category" : "Programming Language" }

## MongoDB skip() method

In MongoDB, skip() method is used to skip the document. It is used with find() and limit() methods.

## Syntax

1. db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

## Scenario:

Consider here also the above discussed example. The collection javatpoint has three documents.

1. [
2. {
3. Course: "Java",
4. details: { Duration: "6 months", Trainer: "Sonoo Jaiswal" },
5. Batch: [ { **size**: "Medium", qty: 25 } ],
6. category: "Programming Language"
7. },
8. {
9. Course: ".Net",
10. details: { Duration: "6 months", Trainer: "Prashant Verma" },
11. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Medium", qty: 10 }, ],
12. category: "Programming Language"
13. },
14. {
15. Course: "Web Designing",
16. details: { Duration: "3 months", Trainer: "Rashmi Desai" },
17. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Large", qty: 10 } ],
18. category: "Programming Language"
19. }
20. ];

Execute the following query to retrieve only one document and skip 2 documents.

## Example

1. db.javatpoint.find().limit(1).skip(2)

After the execution, you will get the following result

Output:

{ "\_id" : ObjectId("564dbced8e2c097d15fbb603"), "Course" : "Web Designing", "det

ails" : { "Duration" : "3 months", "Trainer" : "Rashmi Desai" }, "Batch" : [ { "

size" : "Small", "qty" : 5 }, { "size" : "Large", "qty" : 10 } ], "category" : "

Programming Language" }

As you can see, the skip() method has skipped first and second documents and shows only third document.

# **MongoDB sort() method**

In MongoDB, sort() method is used to sort the documents in the collection. This method accepts a document containing list of fields along with their sorting order.

The sorting order is specified as 1 or -1.

* 1 is used for ascending order sorting.
* -1 is used for descending order sorting.

**Syntax:**

1. db.COLLECTION\_NAME.find().sort({**KEY**:1})

## Scenario

Consider an example which has a collection name javatpoint.

This collection has following fields within it.

1. [
2. {
3. Course: "Java",
4. details: { Duration: "6 months", Trainer: "Sonoo Jaiswal" },
5. Batch: [ { **size**: "Medium", qty: 25 } ],
6. category: "Programming Language"
7. },
8. {
9. Course: ".Net",
10. details: { Duration: "6 months", Trainer: "Prashant Verma" },
11. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Medium", qty: 10 }, ],
12. category: "Programming Language"
13. },
14. {
15. Course: "Web Designing",
16. details: { Duration: "3 months", Trainer: "Rashmi Desai" },
17. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Large", qty: 10 } ],
18. category: "Programming Language"
19. }
20. ];

Execute the following query to display the documents in descending order.

1. db.javatpoint.find().sort({"Course":-1})

This will show the documents in descending order.

{ "\_id" : ObjectId("564dbced8e2c097d15fbb603"), "Course" : "Web Designing", "det

ails" : { "Duration" : "3 months", "Trainer" : "Rashmi Desai" }, "Batch" : [ { "

size" : "Small", "qty" : 5 }, { "size" : "Large", "qty" : 10 } ], "category" : "

Programming Language" }

{ "\_id" : ObjectId("564dbced8e2c097d15fbb601"), "Course" : "Java", "details" : {

"Duration" : "6 months", "Trainer" : "Sonoo Jaiswal" }, "Batch" : [ { "size" :

"Medium", "qty" : 25 } ], "category" : "Programming Language" }

{ "\_id" : ObjectId("564dbced8e2c097d15fbb602"), "Course" : ".Net", "details" : {

"Duration" : "6 months", "Trainer" : "Prashant Verma" }, "Batch" : [ { "size" :

"Small", "qty" : 5 }, { "size" : "Medium", "qty" : 10 } ], "category" : "Progra

mming Language" }

**Differences:**

# **Cassandra vs MongoDB**

Cassandra and MongoDB both are types of NoSQL databases. Cassandra is a distributed database system designed to handle large amount of data and known for its high scalability and high performance. While, MongoDB is document oriented database which also provides high scalability, high performance and automatic scaling.

In terms of simplicity, databases can be divided in two types:

* Development simplicity
* Operational simplicity

While MongoDB is known for an easy out-of-the-box experience, Cassandra is known for easy to manage at scale.

Following is a list of important differences between them:

|  |  |  |
| --- | --- | --- |
| **Index** | **Cassandra** | **Mongodb** |
| 1) | Cassandra is high performance distributed database system. | MongoDB is cross-platform document-oriented database system. |
| 2) | Cassandra is written in Java. | MongoDB is written in C++. |
| 3) | Cassandra stores data in tabular form like SQL format. | MongoDB stores data in JSON format. |
| 4) | Cassandra is got license by Apache. | MongoDB is got license by AGPL and drivers by Apache. |
| 5) | Cassandra is mainly designed to handle large amounts of data across many commodity servers. | MongoDB is designed to deal with JSON-like documents and access applications easier and faster. |
| 6) | Cassandra provides high availability with no single point of failure. | MongoDB is easy to administer in the case of failure. |

## Key Points of Apache Cassandra

* Cassandra is highly scalable, high performance, consistent and fault-tolerant database system. Cassandra is a column-oriented database.
* Cassandra provides easy data distribution.
* Cassandra supports ACID properties i.e. Atomicity, Consistency, Isolation, and Durability.
* Cassandra follows the distribution design of Amazon?s dynamo and its data model design is based on Google's Bigtable.
* Cassandra was initially created at Facebook for inbox search and now it is being used by some of the biggest companies like Facebook, Twitter, ebay, Netflix, Cisco, Rackspace etc.

## Key Points of MongoDB

* MongoDB is well suited for Bigdata and mobile & social infrastructure.
* MongoDB provides Replication, High availability and Auto-sharding.
* MongoDB is used by companies like Foursquare, Intuit, Shutterfly, SourceForge, The New York Times, Lexis Nexis Orange Digital etc.