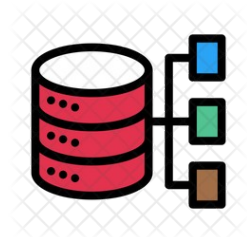


MongoDB Basics

Tushar B. Kute,
<http://tusharkute.com>



MongoDB

- 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 a collection, drop collection, etc.
- There are also given MongoDB interview questions to help you better understand the MongoDB database.

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 manufacturer.
- The manufacturing company 10gen has defined MongoDB as:
- "MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen
- MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.

MongoDB: History

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

MongoDB: Purposes

- The primary purpose of building MongoDB is:
 - 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

MongoDB: Document

- 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.
 - FirstName = "Tushar",
 - Address = "Pune",
 - Spouse = [{Name: "Rashmi"}].
 - FirstName = "Tushar",
 - Address = "Kute"

MongoDB: Features

- 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)

MongoDB: Features

- 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.

MongoDB: Features

- 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.

MongoDB: Features

- 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.

MongoDB: Advantages over RDBMS

- 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.

Data Types

- 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.

Data Types

- 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.

Installation on Ubuntu Linux

```
sudo apt-get update
```

```
sudo apt-get install mongodb
```

- To start the mongodb type:
mongo

MongoDB Compass

- MongoDB compass is nothing but a graphical user interface that can be connected to the MongoDB database and used to find, analyze, modify, and visualize the data stored in the database without requiring any knowledge of queries.
- MongoDB acts as an alternative to Mongo Shell. Mongo Shell can also perform all the aforementioned tasks but requires a lot of technical expertise.
- MongoDB is GUI-based whereas Shell is more technicality based i.e it uses specific commands and queries. MongoDB Compass is present in the Github repo since it is an open-source tool.

MongoDB Compass

- All the data stored in the database can be visualized and explored.
- Data stored in the database can be modified, created, updated, or deleted.
- Shows the Server Statistics in Real-time.
- The Visual representations clearly depict the performance issues and recommend plans.
- All the indexes can be managed.
- JSON schema validation rules can be used to validate data.
- A wide range of plugins is available.

MongoDB Compass: Installation

- Directly download the required files using the command mentioned below(using wget command). To open the terminal click on ctrl+alt+T.

```
wget https://downloads.mongodb.com/compass/mongodb-compass_1.30.1_amd64.deb
```

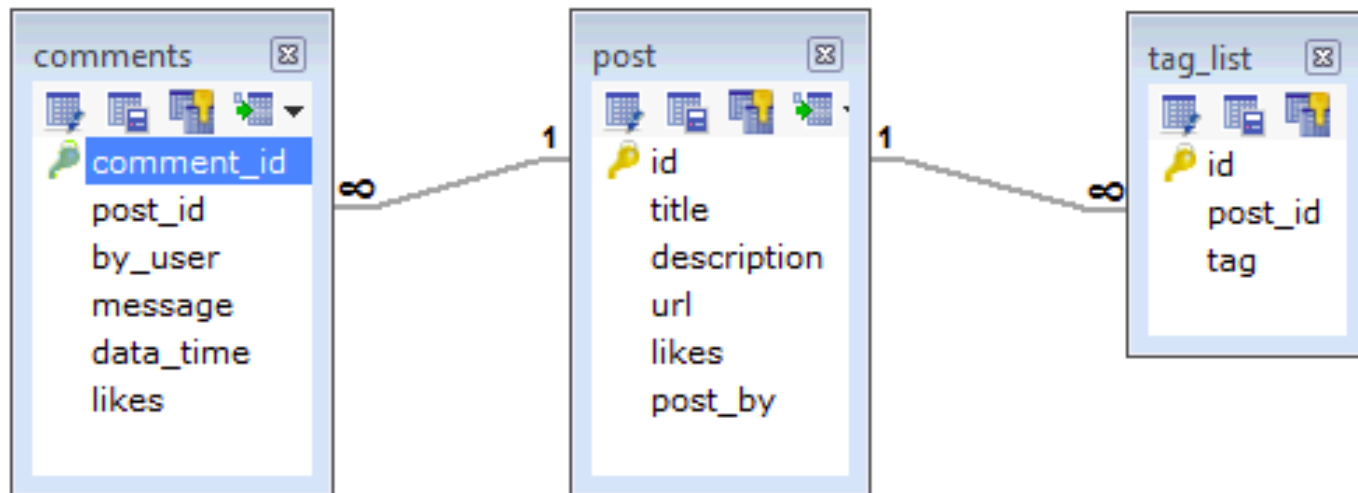
- If you have downloaded the file using the terminal use the following command to install the MongoDB Compass Ubuntu.

```
sudo dpkg -i mongodb-compass_1.30.1_amd64.deb
```

Data Model 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.

Data Model in MongoDB



Create Database

- 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.

Create Database

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

```
use DATABASE_NAME
```
- If the database already exists, it will return the existing database.
- To check the database list, use the command show dbs:

```
> show dbs
```

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:

```
db.dropDatabase()
```
- This syntax will delete the selected database. In the case you have not selected any database, it will delete default "test" database.

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.
- Syntax:

```
db.createCollection(name, options)
```
- Here, Name: is a string type, specifies the name of the collection to be created.

Create Collection: Options

- Capped Boolean (Optional)
 - If it is set to true, enables a capped collection. Capped collection is a fixed size collection 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.

Create Collection: Options

- Size Number (Optional)
 - It specifies a maximum size in bytes for a capped collection. If capped 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.

Create Collection: Example

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

```
>use test
```

```
switched to db test
```

```
>db.createCollection("mydata")
```

```
{ "ok" : 1 }
```

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

```
> show collections
```

Create Collection: Example

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

```
>db.mydata.insert({"name" : "tushar"})
```

```
>show collections
```

```
mydata
```

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

```
db.collection_name.find()
```

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:

`db.COLLECTION_NAME.drop()`

Insert document

- 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.

Insert document

- Syntax

```
>db.COLLECTION_NAME.insert(document)
```

Insert document

```
db.mydata.insert(  
  {  
    course: "BDA",  
    details: {  
      duration: "6 months",  
      Trainer: "Tushar Kute"  
    },  
    Batch: [ { size: "Small", qty: 10 }, { size: "Medium", qty: 30 } ],  
    category: "Data Science"  
  }  
)
```

Check the document

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

```
> db.mydata.find()
```

- You will get the inserted document in return.

Query Document

- The find() Method:
- To query data from MongoDB collection, you need to use MongoDB's find() method.
- Syntax:
 - Basic syntax of find() method is as follows
`db.COLLECTION_NAME.find()`
- find() method will display all the documents in a non structured way.

Query Document

- The pretty() Method:
- To display the results in a formatted way, you can use pretty() method.

- Syntax:

```
db.collection_name.find().pretty()
```

- Apart from find() method there is findOne() method, that reruns only one document.

RDBMS Clauses equivalent to MongoDB

- Equality:
 - Syntax: {<key>:<value>}
 - Example:

```
db.institute.find({"by":"Mitu Research"}).pretty()
```
 - RDBMS equivalent:

```
where by = 'Mitu Research'
```
- Less than:
 - Syntax: {<key>:{\$lt:<value>}}
 - Example:

```
db.institute.find({"likes":{$lt:50}}).pretty()
```
 - RDBMS equivalent:

```
where likes < 50
```

RDBMS Clauses equivalent to MongoDB

- Less than or equal to:
 - Syntax: {<key>:{\$lte:<value>}}
 - Example:
`db.institute.find({"likes":{"$lte":50}}).pretty()`
 - RDBMS equivalent:
`where likes <= 50`
- Greater than:
 - Syntax: {<key>:{\$gt:<value>}}
 - Example:
`db.institute.find({"likes":{"$gt":50}}).pretty()`
 - RDBMS equivalent:
`where likes > 50`

RDBMS Clauses equivalent to MongoDB

- Greater than or equal to:
 - Syntax: {<key>:{\$gte:<value>}}
 - Example:

```
db.institute.find({"likes":{"$gte":50}}).pretty()
```
 - RDBMS equivalent:

```
where likes >= 50
```
- Not equal to:
 - Syntax: {<key>:{\$ne:<value>}}
 - Example:

```
db.institute.find({"likes":{"$ne":50}}).pretty()
```
 - RDBMS equivalent:

```
where likes != 50
```

AND and OR Operations

- In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it AND condition. Basic syntax of AND is shown below:

```
db.mycol.find({key1:value1, key2:value2}).pretty()
```

- To query documents based on the OR condition, you need to use \$or keyword. Basic syntax of OR is shown below:
- ```
db.mycol.find({$or: [{key1: value1}, {key2:value2}]})
```

# AND Operation Example

```
> db.institute.find({"by": "Mitu Research", "title": "NoSQL Database"}).pretty()
{
 "_id" : ObjectId("579b0030329b317d9a0ab0bb"),
 "title" : "NoSQL Database",
 "description" : "NoSQL database doesn't have tables",
 "by" : "Mitu Research",
 "url" : "http://www.mitu.co.in",
 "tags" : [
 "mongodb",
 "database",
 "NoSQL"
],
 "likes" : 20
}
```

# OR Operation Example

```
> db.institute.find({$or:[{"by":"Mitu Research","title": "NoSQL Database"}]}).pretty()
{
 "_id" : ObjectId("579b0030329b317d9a0ab0bb"),
 "title" : "NoSQL Database",
 "description" : "NoSQL database doesn't have tables",
 "by" : "Mitu Research",
 "url" : "http://www.mitu.co.in",
 "tags" : [
 "mongodb",
 "database",
 "NoSQL"
],
 "likes" : 20
}
```



# 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.

# Insert multiple documents

```
var Allcourses =
[
 {
 Course: "Java",
 details: { Duration: "6 months", Trainer: "Tushar Kute" },
 Batch: [{ size: "Medium", qty: 25 }],
 category: "Programming Language"
 },
 {
 Course: "Python",
 details: { Duration: "3 months", Trainer: "Rashmi Thorave" },
 Batch: [{ size: "Small", qty: 10 }, { size: "Large", qty: 30 }],
 category: "Programming Language"
 }
];
```

# Insert multiple documents

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

```
> db.mydata.insert(Allcourses);
```

# Update Documents

- In MongoDB, update() method is used to update or modify the existing documents of a collection.
- Syntax:  
`db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)`

# Update Documents

- Update the existing course "java" into "python":

```
> db.mydata.update({'course':'java'},{$set:
{'course':'python'}})
```

- Check the updated document in the collection:

```
> db.mydata.find()
```

# Update Documents

- By default mongodb will update only single document, to update multiple you need to set a paramter 'multi' to true.
- `db.institute.update({"by":"Mitu Research"},  
{$set:{"title":"MongoDB Tutorial"}},  
{multi:true})`

```
> db.institute.update({"by":"Mitu Research"},{$set:{"title":"MongoDB Tutorial"}})
> db.institute.find()
{ "_id" : ObjectId("5798d3b6e102bcdf8d057377"), "name" : "tushar kute" }
{ "_id" : ObjectId("579b0030329b317d9a0ab0ba"), "title" : "MongoDB Overview", "description" : "MongoDB is no s
ql database", "by" : "Mitu Skillologies", "url" : "http://www.mitu.co.in", "tags" : ["mongodb", "database",
"NoSQL"], "likes" : 1000 }
{ "_id" : ObjectId("5798d311e102bcdf8d057376"), "name" : "tushar kute", "title" : "MongoDB Tutorial" }
{ "_id" : ObjectId("579b0030329b317d9a0ab0bb"), "by" : "Mitu Research", "description" : "NoSQL database doesn'
t have tables", "likes" : 20, "tags" : ["mongodb", "database", "NoSQL"], "title" : "MongoDB Tutorial", "u
rl" : "http://www.mitu.co.in" }
```

# Update Documents

- MongoDB Save() Method:

The save() method replaces the existing document with the new document passed in save() method

- Syntax:

```
db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})
```

# Update Documents - Examples

- `db.institute.save({"_id":ObjectId("5798d311e102bCDF8d057376"),"name":"rashmi thorave"})`

```
> db.institute.find()
{ "_id" : ObjectId("579b0030329b317d9a0ab0ba"), "title" : "MongoDB Overview", "description" : "MongoDB is no s
ql database", "by" : "Mitu Skillologies", "url" : "http://www.mitu.co.in", "tags" : ["mongodb", "database",
"NoSQL"], "likes" : 1000 }
{ "_id" : ObjectId("5798d311e102bCDF8d057376"), "name" : "tushar kute", "title" : "MongoDB Tutorial" }
{ "_id" : ObjectId("579b0030329b317d9a0ab0bb"), "by" : "Mitu Research", "description" : "NoSQL database doesn'
t have tables", "likes" : 20, "tags" : ["mongodb", "database", "NoSQL"], "title" : "MongoDB Tutorial", "u
rl" : "http://www.mitu.co.in" }
{ "_id" : ObjectId("5798d3b6e102bCDF8d057377"), "name" : "tushar kute", "title" : "MongoDB Tutorial" }
> db.institute.save({"_id":ObjectId("5798d311e102bCDF8d057376"),"name":"rashmi thorave"})
> db.institute.find()
{ "_id" : ObjectId("579b0030329b317d9a0ab0ba"), "title" : "MongoDB Overview", "description" : "MongoDB is no s
ql database", "by" : "Mitu Skillologies", "url" : "http://www.mitu.co.in", "tags" : ["mongodb", "database",
"NoSQL"], "likes" : 1000 }
{ "_id" : ObjectId("5798d311e102bCDF8d057376"), "name" : "rashmi thorave" }
{ "_id" : ObjectId("579b0030329b317d9a0ab0bb"), "by" : "Mitu Research", "description" : "NoSQL database doesn'
t have tables", "likes" : 20, "tags" : ["mongodb", "database", "NoSQL"], "title" : "MongoDB Tutorial", "u
rl" : "http://www.mitu.co.in" }
{ "_id" : ObjectId("5798d3b6e102bCDF8d057377"), "name" : "tushar kute", "title" : "MongoDB Tutorial" }
```



# Delete Documents

- The remove() Method:
- MongoDB's remove() method is used to remove document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag
  1. deletion criteria : (Optional) deletion criteria according to documents will be removed.
  2. justOne : (Optional) if set to true or 1, then remove only one document.
- Syntax:

```
db.COLLECTION_NAME.remove(DELETION_CRITERIA)
```

# Delete Documents

- `db.institute.remove({'title':'MongoDB Overview'})`

```
> db.institute.find()
{ "_id" : ObjectId("579b0030329b317d9a0ab0ba"), "title" : "MongoDB Overview", "description" : "MongoDB is no s
ql database", "by" : "Mitu Skillologies", "url" : "http://www.mitu.co.in", "tags" : ["mongodb", "database",
"NoSQL"], "likes" : 1000 }
{ "_id" : ObjectId("5798d311e102bcdf8d057376"), "name" : "rashmi thorave" }
{ "_id" : ObjectId("579b0030329b317d9a0ab0bb"), "by" : "Mitu Research", "description" : "NoSQL database doesn'
t have tables", "likes" : 20, "tags" : ["mongodb", "database", "NoSQL"], "title" : "MongoDB Tutorial", "u
rl" : "http://www.mitu.co.in" }
{ "_id" : ObjectId("5798d3b6e102bcdf8d057377"), "name" : "tushar kute", "title" : "MongoDB Tutorial" }
> db.institute.remove({'title':'MongoDB Overview'})
> db.institute.find()
{ "_id" : ObjectId("5798d311e102bcdf8d057376"), "name" : "rashmi thorave" }
{ "_id" : ObjectId("579b0030329b317d9a0ab0bb"), "by" : "Mitu Research", "description" : "NoSQL database doesn'
t have tables", "likes" : 20, "tags" : ["mongodb", "database", "NoSQL"], "title" : "MongoDB Tutorial", "u
rl" : "http://www.mitu.co.in" }
{ "_id" : ObjectId("5798d3b6e102bcdf8d057377"), "name" : "tushar kute", "title" : "MongoDB Tutorial" }
```

# Delete Documents

- Remove only one
  - If there are multiple records and you want to delete only first record, then set justOne parameter in remove() method

```
db.COLLECTION_NAME.remove(DELETION_CRITERIA, 1)
```
- Remove All documents
  - If you don't specify deletion criteria, then mongodb will delete whole documents from the collection. This is equivalent of SQL's truncate command.

```
db.mycol.remove()
db.mycol.find()
```

# MongoDB vs. SQL Similarities

- Although we call MongoDB and MySQL a database, both of them are used to create databases.
- We create a lot of tables in MySQL, similarly, in the case of MongoDB we call it a collection, which means the equivalent thing of the table in the world of MongoDB is called a collection.
- In the case of MySQL, you will come to know a lot about the Index, and the same thing is there for MongoDB.

# MongoDB vs. SQL Similarities

- In the MySQL world, we have the primary key, and it consists of a huge impact on linking the tables. Similarly, we have the same when it comes to MongoDB. Here, each document stored in a collection requires a unique `_id` field. It is equivalent to what the primary key is in the case of MongoDB.
- Another key point is that in MySQL we have the concept of rows, we create a lot of rows to enter the data. Similarly the same stands for MongoDB, but it is termed as BSON documents or JSON also.
- MySQL and MongoDB support the programming languages such as Node.JS, Java, and Python.

# MongoDB Language Bindings

- MongoDB loves developers. And we want to make software engineers everywhere successful with building applications—whether that's a serverless web app, a mobile game, an IoT integration, or any other application you can imagine.
- MongoDB is the most popular NoSQL databases and is widely adopted for storing and managing both structured and unstructured data.
- Data administrators, analysts, and programmers can use the programming language of their choice to optimize and manage data, and create highly performant applications.

# MongoDB Language Bindings

- The languages officially supported and compatible with MongoDB are:
  - C
  - C++
  - C#
  - Go
  - Java
  - Node.js
  - PHP
  - Python
  - Ruby
  - Rust
  - Scala
  - Swift

# Thank you

*This presentation is created using LibreOffice Impress 7.4.1.2, can be used freely as per GNU General Public License*



@mitu\_skillologies



@mITuSkillologies



@mitu\_group



@mitu-skillologies



@MITUSkillologies

kaggle

@mituskillologies

## Web Resources

<https://mitu.co.in>

<http://tusharkute.com>



@mituskillologies

**[contact@mitu.co.in](mailto:contact@mitu.co.in)**

**[tushar@tusharkute.com](mailto:tushar@tusharkute.com)**