# MongoDB

# Data base

- **Databases can be Classified  in 3 types:**
- RDBMS (Relational Database Management System)
- OLAP (Online Analytical Processing)
- NoSQL (recently developed database)

# NOSQL Data base

- 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.

# Mongdb

MongoDB is a cross-platform, document-oriented database that provides

High performance.
High availability.
Easy scalability.

MongoDB works on concept of collection and document.

# RBDMS VS Mongdb

MongoDB advantages over RDBMS

MongoDB is a popularly used database. Based on, 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.

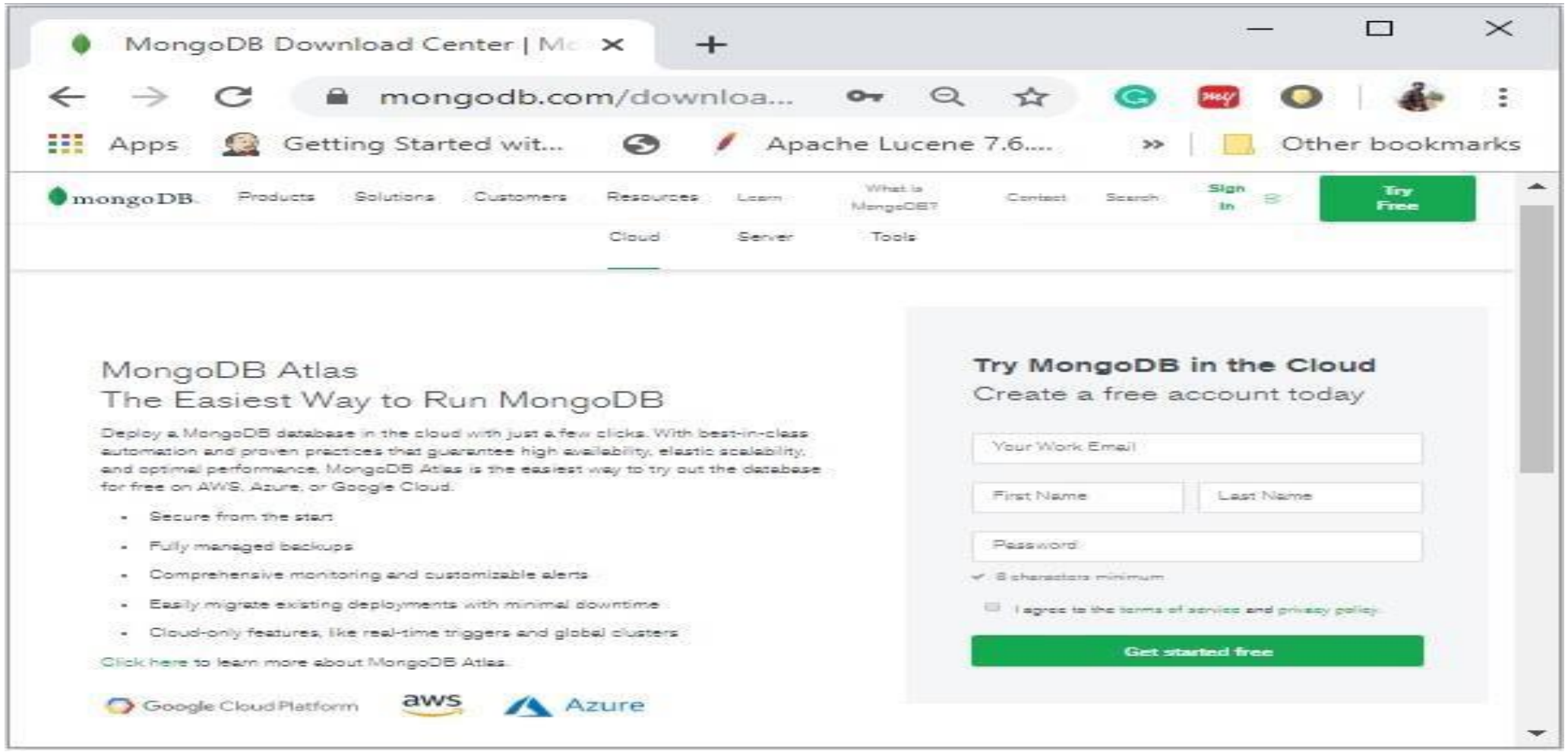
A Relational database has a typical schema design that shows number of tables and the relationship between these tables, while in MongoDB there is no concept of relationship.

# RDBMS and MongoDB

| 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) |
| Database Server and Client | |
| mysqld/Oracle | mongod |
| mysql/sqlplus | mongo |

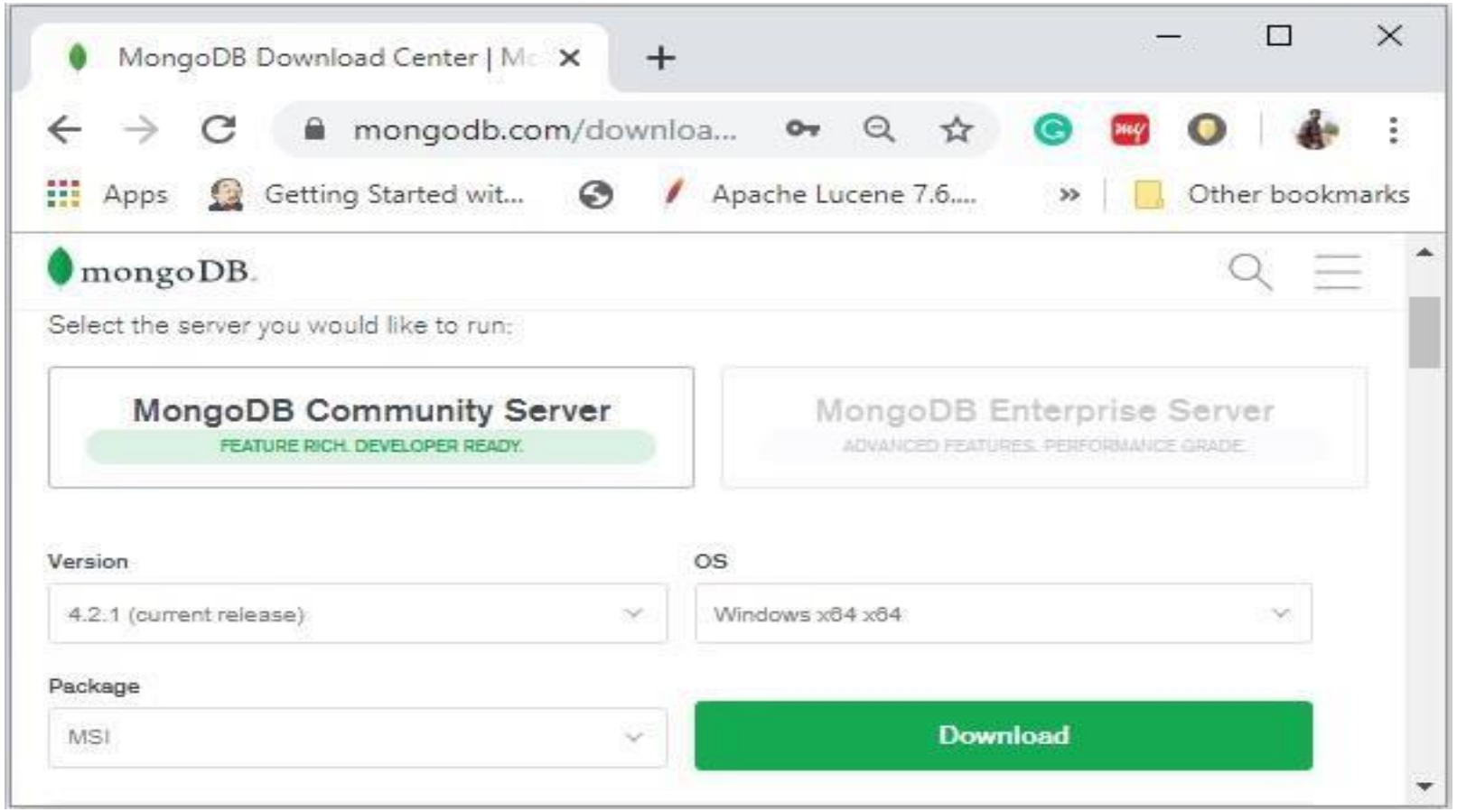
# Install MongoDB On Windows

- To install MongoDB on Windows, first download the latest release of MongoDB from https://www.mongodb.com/download-center.



# Where to use MongoDB

Enter the required details, select the ***Server*** tab, in it you can choose the version of MongoDB, operating system and, packaging as:



# Where to use MongoDB

Now install the downloaded file, by default, it will be installed in the folder **C:\Program Files**.

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So you need to create this folder using the Command Prompt. Execute the following command sequence.

```
C:\>md data
 C:\md data\db
```

# Where to use MongoDB

Setting path for mongod.exe

In the command prompt, navigate to the bin directory current in the MongoDB installation folder. Suppose my installation folder is **C:\Program Files\MongoDB**

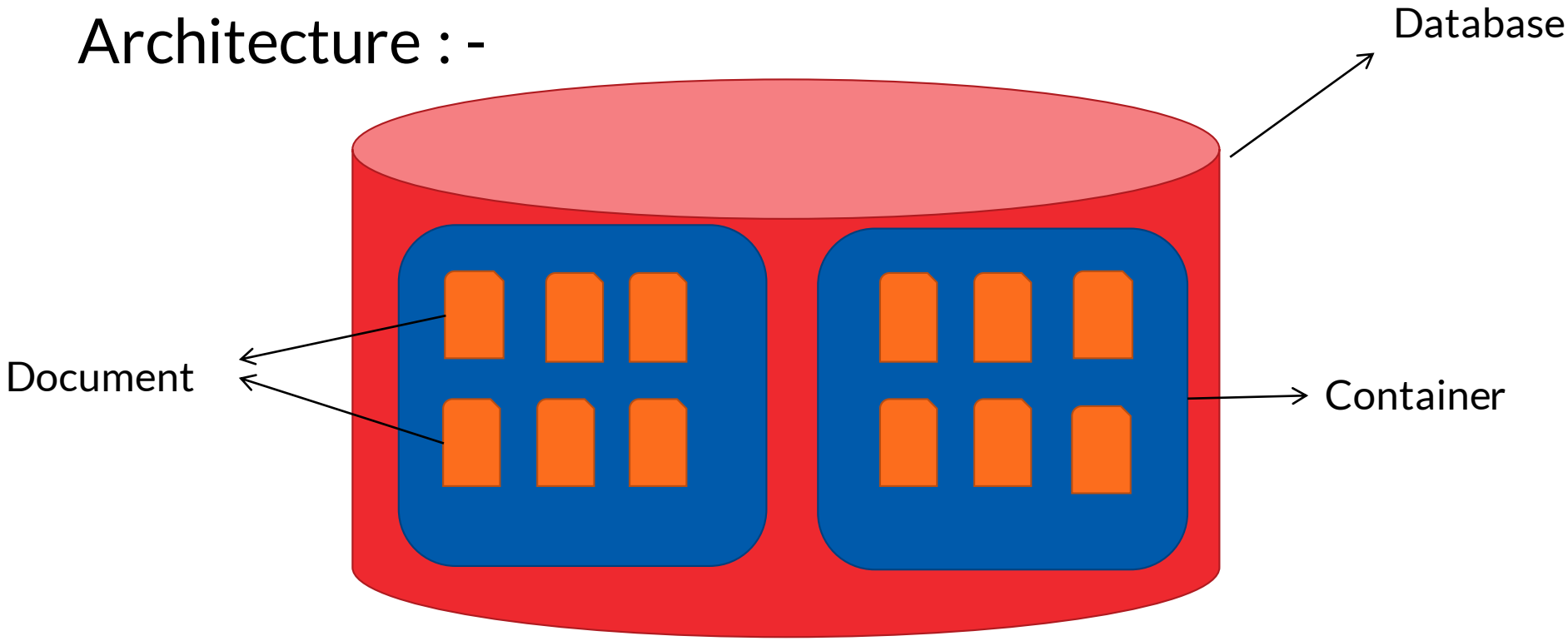
```
C:\Users\XYZ>d:cd C:\Program Files\MongoDB\Server\4.2\bin
 C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath "C:\data"
```

This will show **waiting for connections** message on the console output, which indicates that the mongod.exe process is running successfully.

Now to run the MongoDB, you need to open another command prompt and issue the following command.

# Mongo DB architecture

Architecture : -



# Mongo - Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

# Mongo – Database creation

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

```
use DATABASE_NAME
```

- If there is no existing database, the following command is used to create a new database.
- If the database already exists, it will return the existing database.

# Mongo – Database creation

- Use SampleDB
- DB  (DB command will give return SampleDB )
- Show databases / show dbs  (will list all data bases in the server)

Created database "Sampledb" will be in the list only after inserting a document in it.

```
db.emp.insert({name:"uma"})
WriteResult({ "nInserted" : 1 })
```

# Mongo –Droping Database

```
> db.dropdatabase
test.dropdatabase
command will delete the selected database.
 In the case we have not selected any database, it will delete default "test"
database.

> use sample1
switched to db sample1

> db.dropdatabase
sample1.dropdatabase
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
sample1 0.000GB
```

# Mongo – Database- Collection

Collection is a group of MongoDB documents.

Collection is equivalent of an RDBMS table.

A collection exists within a single database.

Documents within a collection can have different fields.

Typically, all documents in a collection are of similar or related purpose.

# Mongo – Create Collection

In MongoDB, db.createCollection(name, options) is used to create collection.

MongoDB creates collection automatically when you insert some documents.

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

**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.

# Mongo – Create Collection-options

| Field | Type | Description |
|---|---|---|
| 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. |
| 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. |

# Mongo – Create Collection-automatically

In Mongo Collections will create automatically when insert the data into it.

➔db.sample23.insert({“name”:”uma”})

Insert command will insert the data if collection exist else it will create a new Collection and insert the data in  it.

# MongoDB 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. |

# Mongo – View Document

```
> db.emp.insert({"name":"Uma"})
WriteResult({ "nInserted" : 1 })
> db.emp.find()
{ "_id" : ObjectId("5f9aff2ddb99a0cb0f6678cf"), "name" : "Uma" }
```

# Mongo – 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.

```
>use mydb
>show collections
>db.collection_name.drop()
>show collections
```

# MongoDB insert documents

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

```
db.emp.insert(
... {
...     name: "Chaaru",
...     details: {
...       age: "19 years",
...       standard: "BE first year"
...     },
...     Batch: [ { size: "Small", qty: 15 }, { size: "Medium", qty: 25 } ],
...     category: "Computer Science"
...   }
... )
WriteResult({ "nInserted" : 1 })
```

# MongoDB insert multiple documents

To Insert multiple documents in a collection,have to pass an array of documents to the db.collection.insert() method.

Create an array of documents

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

# MongoDB insert Many – inserting multiple documents

```
db.emp.insertMany(
                [
                        {
                                FirstName: "Radhika",
                                Last_Name: "Sharma",
                                Date_Of_Birth: "1995-09-26",
                                e_mail: "radhika_sharma.123@gmail.com",
                                phone: "9000012345"
                        },
                        {
                                First_Name: "Rachel",
                                Last_Name: "Christopher",
                                Date_Of_Birth: "1990-02-16",
                                e_mail: "Rachel_Christopher.123@gmail.com",
                                phone: "9000054321"
                        },
                                ]
)
```

# MongoDB insert multiple documents

```
> var Allemp =
...   [
...    {
...      name: "Sai",
...      details: { age: "26", standard: "BE" },
...      Batch: [ { size: "Medium", qty: 25 } ],
...      category: "EEE"
...    },
...    {
...      name: "Devi",
...      details: { age: "36", standard: "ME" },
...      Batch: [ { size: "Small", qty: 5 }, { size: "Medium", qty: 10 }, ],
...      category: "EEE"
...    },
...    {
...      name: "Rashmi Desai",
...      details: { Duration: "3 months", standard: "MBBS" },
...      Batch: [ { size: "Small", qty: 5 }, { size: "Large", qty: 10 } ],
...      category: "Medicine"
...    }
...   ];
>
```

# MongoDB insert multiple documents

```
> db.emp.insert(Allemp);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
> db.emp.find();
{ "_id" : ObjectId("5f9af629db99a0cb0f6678ce"), "name" : "uma" }
{ "_id" : ObjectId("5f9bc33b58538447aace7001"), "name" : "Sai", "details" : { "age" : "26", "standard" : "BE" }, "Batch" : [ { "size"
: "Medium", "qty" : 25 } ], "category" : "EEE" }
{ "_id" : ObjectId("5f9bc33b58538447aace7002"), "name" : "Devi", "details" : { "age" : "36", "standard" : "ME" }, "Batch" : [ {
"size" : "Small", "qty" : 5 }, { "size" : "Medium", "qty" : 10 } ], "category" : "EEE" }
{ "_id" : ObjectId("5f9bc33b58538447aace7003"), "name" : "Rashmi Desai", "details" : { "Duration" : "3 months", "standard" :
"MBBS" }, "Batch" : [ { "size" : "Small", "qty" : 5 }, { "size" : "Large", "qty" : 10 } ], "category" : "Medicine" }
>
```

# MongoDB find and pretty methods

```
> db.emp.find().pretty()
{ "_id" : ObjectId("5f9bfe8ab662c3a18997f035"), "name" : "Uma" }
{ "_id" : ObjectId("5f9bfec2b662c3a18997f036"), "name" : "Anu", "age" : 12 }
{
	"_id" : ObjectId("5f9bfedbb662c3a18997f037"),
	"name" : "Deepa",
	"age" : 24,
	"sal" : 20000
}
{
	"_id" : ObjectId("5f9c00b0b662c3a18997f03a"),
	"name" : "Chaaru",
	"age" : 35,
	"dept" : {
		"did" : 10,
		"dname" : "HR"
	},
	"sal" : 40000
}
```

# MongoDB - Projection

MongoDB's **find()** method, accepts second optional parameter that is list of fields that you want to retrieve.

In MongoDB, when you execute **find()** method, then it displays all fields of a document.

To limit this, you need to set a list of fields with value 1 or 0. 1 is used to show the field while 0 is used to hide the fields.

```
db.student.find({},{"name":1,"age":1,_id:0})
```

**Note:**

In Mongo the _id field is always displayed while executing find() method, to hide the field set it as 0.

# MongoDB -Limit

To limit the records in MongoDB, you need to use **limit()** method. The method accepts one number type argument, which is the number of documents that you want to be displayed.

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

```
db.emp.find().limit(5);
{ "name" : "Anu", "age" : 12 }
{ "name" : "Deepa", "age" : 24 }
{ "name" : "Chaaru", "age" : 35 }
{ "name" : "Chaaru", "age" : 35 }
{ "name" : "David", "age" : 35 }
```

# MongoDB - Skip

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

db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)

```
 db.emp.find().limit(2).skip(2);
{ "_id" : ObjectId("5f9bfedbb662c3a18997f037"), "name" : "Deepa", "age" : 24, "sal" : 20000 }
{ "_id" : ObjectId("5f9c00b0b662c3a18997f03a"), "name" : "Chaaru", "age" : 35, "dept" : { "did" : 10,
"dname" : "HR" }, "sal" : 40000 }
```

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

# MongoDB -Sort

To sort documents in MongoDB, you need to use **sort()** method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

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

```
db.emp.find().limit(5).sort({"age":-1});
{ "_id" : ObjectId("5f9c00b0b662c3a18997f03a"), "name" : "Chaaru", "age" : 35, "dept" : { "did" : 10, "dname" : "HR" }, "sal" : 40000 }
{ "_id" : ObjectId("5f9c00b0b662c3a18997f03b"), "name" : "Chaaru", "age" : 35, "dept" : { "did" : 10, "dname" : "HR" }, "sal" : 40000 }
{ "_id" : ObjectId("5f9c0113b662c3a18997f03c"), "name" : "David", "age" : 35, "dept" : { "did" : 10, "dname" : "Accounts" }, "address" : [ { "no" : 12, "street" : "ABC Street" }, { "no" : 22, "street" : "XYC Street" }, { "no" : 67, "street" : "KLC Street" } ], "sal" : 40000 }
{ "_id" : ObjectId("5f9bfedbb662c3a18997f037"), "name" : "Deepa", "age" : 24, "sal" : 20000 }
{ "_id" : ObjectId("5f9bfec2b662c3a18997f036"), "name" : "Anu", "age" : 12 }
```

# MongoDB The findOne() method

```
> db.emp.findOne({"First_Name" :"Fathima"});
{
        "_id" : ObjectId("5f9c0d78b662c3a18997f03f"),
        "First_Name" : "Fathima",
        "Last_Name" : "Sheik",
        "Date_Of_Birth" : "1990-02-16",
        "e_mail" : "Fathima_Sheik.123@gmail.com",
        "phone" : "9000054321"
}
```

# MongoDB – Selection Relational Operator

| Operation | Example | RDBMS Equivalent |
|---|---|---|
| Equality | db.student.find({"name":"Bala"}) | where name = 'Bala' |
| Less Than | db.student.find({"id":{$lt:200}}) | where id < 200 |
| Less Than Equals | db.student.find({"id":{$lte:200}}) | where likes <= 200 |
| Greater Than | db.student.find({"id":{$gt:200}}) | where likes > 200 |
| Greater Than Equals | db.student.find({"id":{$gte:200}}) | where likes >= 200 |
| Not Equals | db.student.find({"id":{$ne:200}}) | where likes != 50 |
| Values in an array | db.studentl.find({"name":{$in:[ "Sam","Mahitha",”Haritha”]}}) | Where name matches any of the value in :[ "Sam","Mahitha","Haritha"] |
| Values not in an array | db.student.find({"name":{$nin:[ "Sam","Mahitha","Haritha"]}}) | Where name values is not in the array :[" "Sam","Mahitha","Haritha""] or, doesn’t exist at all |

# MongoDB – Logical Gates

| Operation | Example |
|---|---|
| **And** | db.student.find({$and:[{"course":"Java"},{"age": 14}]}).pretty() |
| **Or** | db.student.find({$or:[{"course":"Java"},{"age": 14}]}).pretty() |
| **Nor** | db.student.find({$nor:[{"course":"Java"},{"age": 14}]}).pretty() |
| **Not** | db.student.find( { "age": { $not: { $gt: 25 } } } ) |

To query documents based on the NOT condition, you need to use $not keyword. Above is the basic syntax of **NOT** −NOR,NOT

# MongoDB  Update

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

db.COLLECTION_NAME.**update**(SELECTIOIN_CRITERIA, UPDATED_DATA)

```
>db.emp.find()
>{ "_id" : ObjectId("5f9af629db99a0cb0f6678ce"), "name" : "uma", "age" : "96" }
>db.emp.update({'name':'uma'},{$set:{'age':'62'}})
>db.emp.find()
{ "_id" : ObjectId("5f9af629db99a0cb0f6678ce"), "name" : "uma", "age" : “62" }
```

# MongoDB Save

To insert the document you can use **db.post.save(document)** also. If you don't specify **_id** in the document then **save()** method will work same as **insert()** method. If you specify _id then it will replace whole data of document containing _id as specified in save() method.

# MongoDB Save

```
db.emp.save(
{
 name: “Harish",
    details: {
      age: “18 years",
      standard: "BE first Year"
    },
    Batch: [ { size: "Small", qty: 15 },
 { size: "Medium", qty: 25 } ],
    category: "Computer Science"
  }
)
```

```
db.emp.find()
{ "_id" : ObjectId("5f9aff2ddb99a0cb0f6678cf"), "name" : "Uma" }
{ "_id" : ObjectId("5f9b0214db99a0cb0f6678d0"), "name" : "Chaaru",
"details" : { "age" : "19 years", "standard" : "BE first year" }, "Batch" : [ {
"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty" : 25 } ], "category" :
"Computer Science" }
{ "_id" : ObjectId("5f9b054adb99a0cb0f6678d1"), "name" : "Harish",
"d          " : { "age" : "19 years", "standard" : "BE fSec year" }, "Batch" : [ {
"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty" : 25 } ], "category" :
"Computer Science" }
```

# MongoDB Save

```
db.emp.save(
{
    _id :
ObjectId("5f9b054adb99a0cb0f6678
d1"),
    name: "Aravindan",
    details: {
      age: "55 years",
      standard: "BEr"
    },
    Batch: [ { size: "Small", qty: 15 },
    {
   size: "Medium", qty: 25 } ],
    category: "Computer Science"
  }
)
```

```
> db.emp.find()
{ "_id" : ObjectId("5f9aff2ddb99a0cb0f6678cf"),
"name" : "Uma" }
{ "_id" : ObjectId("5f9b0214db99a0cb0f6678d0"),
"name" : "Chaaru", "details" : { "age" : "19 years",
"standard" : "BE first year" }, "Batch" : [ { "size" :
"Small", "qty" : 15 }, { "size" : "Medium", "qty" : 25 } ],
"category" : "Computer Science" }
{ "_id" : ObjectId("5f9b054adb99a0cb0f6678d1"),
"name" : "Aravindan", "details" : { "age" : "55 years",
"standard" : "BEr" }, "Batch" : [ { "size" : "Small", "qty" :
15 }, { "size" : "Medium", "qty" : 25 } ], "category" :
"Computer Science" }
```

# 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.

db.collection_name.remove (DELETION_CRITERIA)

```
> db.emp.remove({name:"Uma"})
WriteResult({ "nRemoved" : 1 })
>
```

# MongoDB Delete 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.collection_name.remove()

```
 db.emp.remove({});
WriteResult({ "nRemoved" : 11 })
➢ db.emp.find()
```

# Thank you

*Innovative Services*

*Passionate Employees*

*Delighted Customers*