**MongoDB**

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a collection of data. Each database gets its own set of files on the file system.

Database is a physical container for collections.

## Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database.

* Collection does not have any Schema.
* The Collection do not have any concept of join.

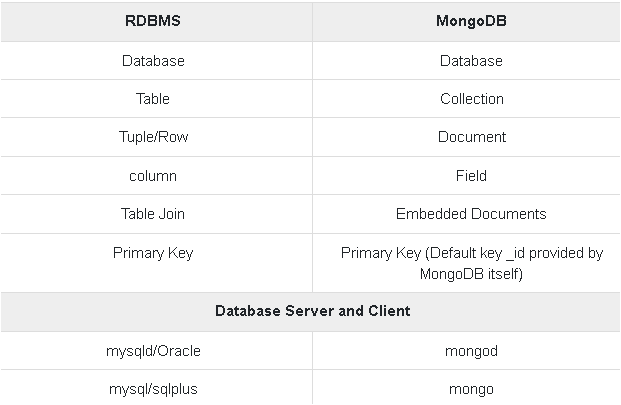
## Document

A document is a set of key-value pairs. Documents have dynamic schema.

Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure,

{"greeting" : "Hello world!"}

Complex documents will contain multiple key/value pairs:  
{"greeting" : "Hello world!", "views" : 3}



db.srimani.insert([

{

title: "MongoDB",

description: "MongoDB is no SQL database",

by: "Subrata Srimani",

url: "www.subratasrimani.com",

tags: ["mongodb", "database", "NoSQL"],

likes: 10

},

{

title: "NoSQL Database",

description: "NoSQL database doesn't have tables",

by: "Subrata Srimani",

url: "www.subratasrimani.com",

tags: ["mongodb", "database", "NoSQL"],

likes: 20,

comments: [

{

user:"user1",

message: "My first comment",

dateCreated: new Date(2018,11,10,2,35),

like: 0

}

]

}

])

**Queries:**

Show Data base: show dbs;

Select Database: use databaseName

Show collections: show collections or db.getCollectionNames()

**Create Collection:** db.createcollection("democollection")

Show Records: db.collectionName.find()

**Show in pretty () format**: db.COLLECTION\_NAME.find().pretty()

Find one record; db.COLLECTIONNAME.findOne()

**Rename CollectionName**: db.student.renameCollection("StName")

**Rename Columns:** db.emp.updateMany({},{$rename:{"naam”: name"}})

**Drop Database**: First, use Database then db.dropDatabase ()

**Pattern Matching:** db.restaurants.find( { “State”: /Delhi/ } ) .. db.matm.find({"State": /Mah/})

**[**Case-insensitive match] **db.collection.find({name: {$regex : /string/i}})**

[Start with String] **db.collection.find({name: {$regex : /^string/i}})**

# **How to Query with a Date Range**

**db.collection.find({ day: { $gt: ISODate("2020-01-21"), $lt: ISODate("2020-01-24")}})**

**Projection:Desired Colulmns:** db.matm.find({"TerminalType":1,"SerialNumber":1,"Mdn":1})

Ex-db.matm.find({"State": /Mah/},{"TerminalType":1,"SerialNumber":1,"Mdn":1,"State":1}).pretty()

**--Add / Delete Columns---**

>db.demo.updateMany({},{$set:{"DOJ": new date()}}) --Add Columns

>db.demo.updateMany({},{$unset:{"DOJ":""}}) --Delete Columns

db.collection.updateMany({}, {$unset: {"field1":1, "field2":1}})

**----Delete Record-----**

Delete multip[le record>> db.emp1.deleteMany({Department:"IT"}) - All record will delete (Department=IT)

Delete Single record>> db.emp1.deleteMany({ID:"1"})

**---Min and Max--**

db.OxiEmp.aggregate([{"$group":{"\_id":null,"maximum\_Salary":{"$max":"$Salary"}}}])

db.Class.count({age:{$gt:"30"}})

db.Class.find({age:{$gt:"30"}}).sort({age:1})

db.Class.find({age:{$lt:"30"}}).sort({age:1})

**------Distinct----**

db.Class.distinct ("salary")

**----Increment------**

Single Increment:

db.Incre.update ({EmpID:11},{$inc:{salary:5000}})

Multiple Increment:

db.emp.updateMany ({salary:{$lt:30000}},{$inc:{salary:4000}})

db.Incre.update ({EmpID: 11},{$inc:{salary:500,"Bonus":500}})

"EmpID" : 11, "Name" : "Deepak", "age" : 24, "salary" : 10500, "Bonus" : 500 }

**UpdateOne() method:**

db.empDetails.updateOne({First\_Name: 'Radhika'},

{ $set: { Age: '30',e\_mail: 'radhika\_newemail@gmail.com'}})

**UpdateOne Method**

db.empDetails.findOneAndUpdate(

{First\_Name: 'Radhika'},

{ $set: { Age: '30',e\_mail: 'radhika\_newemail@gmail.com'}}

)

db.newemp.update( { empname:"admin" } , {$set:{"empname" : "adminnew"} } )

db.newemp.update({empname:"manager"},{$set:{"empname" : "new\_manager"}})

**Define Variable:**

var myemp = [ { empid:1, empname:'admin'},{empid:2, empname:'manager'},{empid:3, empname:"QA" } ]

db.newemp.insert(myemp)

db.srimani.find({"by":"Subrata Srimani"}).pretty()

db.srimani.find({"by":"susmita"}).pretty()

db.srimani.find({"likes":{$lt:50}}).pretty()

db.srimani.find({"likes":{$lte:50}}).pretty()

db.srimani.find({"likes":{$gt:50}}).pretty()

db.srimani.find({"name":{$in:["Raj", "Ram", "Raghu"]}}).pretty()

db.stuinfo.find({"name":{$in:["Student1", "Ram", "Student2"]}}).pretty() 🡪In Arrey

db.emp.find({city:{$in:["mumbai","pune"]}})

db.stuinfo.find({"name":{$nin:["Student1", "Raghav"]}}).pretty() 🡪 Not in Arrey

db.stuinfo.find({$and:[{"name":"Student1"},{"duration": "3yrs"}]}).pretty()

db.stuinfo.find({$or:[{"name":"Student1"},{"duration": "3yrs"}]}).pretty()

db.stuinfo.find({"marks": {$gt:10}, $or: [{"name":"Student1"},{"duration": "3yrs"}]}).pretty()

Aggregate:

db.companies.aggregate( [

{ $match:{"year":2018}},

{$sort:{name:-1}},

{$project:{id:0,name:1}}

}

])

db.companies.aggregate(

[

{ $match:{"year":{$gte:2018}}},

{$group:{\_id:"$year",companies:{$push:"$name"}}}

{$sort:{name:-1}}

}

])

**Group by and count:**

**db.collection.aggregate([{$group : {\_id:"$field\_name", count:{$sum:1}}}])**

**db.collection.aggregate([{$group : {\_id:"$field\_name", count:{$sum:1}}},** **{$sort: {count:1}}**

**])**

# **Group By and Sum**

> db.emp.aggregate([ {$group:{\_id:"$city",count:{$sum:"$salary"}}}])

Ex.

use subrata

db.createCollection("srimani")

db.srimani.insert([

{

title: "MongoDB",

description: "MongoDB is no SQL database",

by: "Subrata Srimani",

url: "www.subratasrimani.com",

tags: ["mongodb", "database", "NoSQL"],

likes: 10

},

{

title: "NoSQL Database",

description: "NoSQL database doesn't have tables",

by: "Subrata Srimani",

url: "www.subratasrimani.com",

tags: ["mongodb", "database", "NoSQL"],

likes: 20,

comments: [

{

user:"user1",

message: "My first comment",

dateCreated: new Date(2018,11,10,2,35),

like: 0

}

]

}

])

**Ex-**

db.empdata.insertMany(

[

{

First\_Name: "Sumana",

Last\_Name: "Ray",

Age: "19",

e\_mail: "sumana@gmail.com",

phone: "9064512345"

},

{

First\_Name: "Rahul",

Last\_Name: "Dey",

Age: "17",

e\_mail: "rahuldey@gmail.com",

phone: "9123454321"

},

{

First\_Name: "Ayan",

Last\_Name: "Gouri",

Age: "24",

e\_mail: "gouri@gmail.com",

phone: "9345654321"

}

]

)

db.empdata.find({$nor:[{"First\_Name": "Ayan"},{"Last\_Name": "Ray"}]}).pretty()

**Ex-**

db.testing.insert(

{

Stu\_ID:"ST004",

StuInfo:{

Name:"Amit",

Lname:"Singh",

Father Name:"Shyam Lal",

Mother Name:"Meena singh"

},

Contact:{

Mobile:7827166262,

Email: "sandeepkumarv7@gmail.com",

Alternate Mobile:7701815788

},

Address:{

City:"Delhi",

State:"New Delhi",

Pin:700052

}

}

)

**Ex-**

var bc =

[

{

Course: "Core Java",

details: { Duration: "6 months", Trainer: "Subrata Srimani" },

Batch: [ { size: "Medium", qty: 25 } ],

category: "Programming Language"

}

],

{

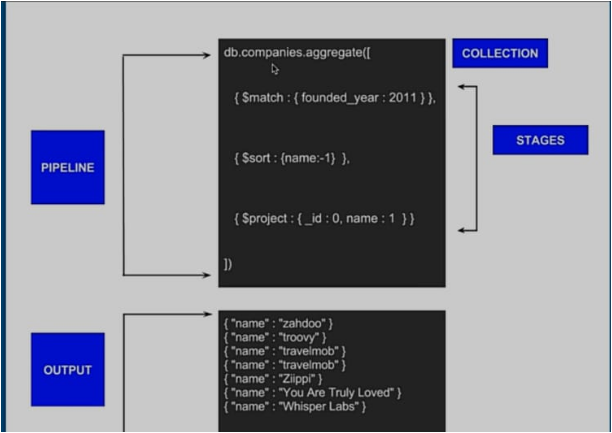
Course: ".Net",

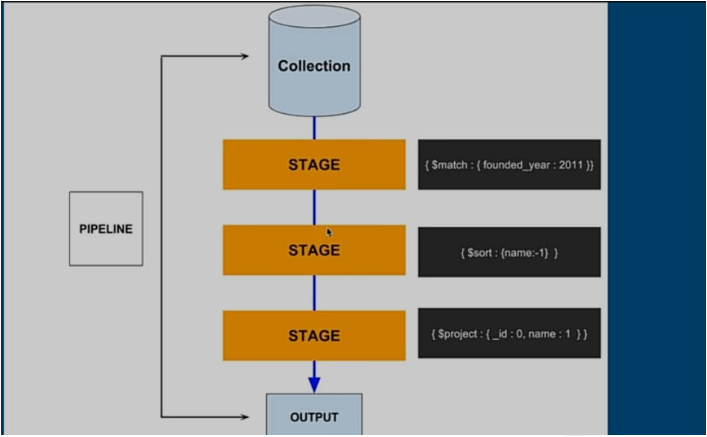
details: { Duration: "6 months", Trainer: "Subrata Srimani" },

Batch: [ { size: "Small", qty: 5 }, { size: "Medium", qty: 10 }, ],

category: "Programming Language"

},





### **What do you understand by NoSQL databases? Is MongoDB a NoSQL database?**

NoSQL stands for "Not Only SQL". NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.

Yes. MongoDB is a NoSQL database.

### **What are the different types of NoSQL databases? Give some example.**

NoSQL database can be classified as 4 basic types:

1. Key value store NoSQL database
2. Document store NoSQL database
3. Column store NoSQL database
4. Graph base NoSQL databse

There are many NoSQL databases. MongoDB, Cassandra, CouchBD, Hypertable, Redis, Riak, Neo4j, HBASE, Couchbase, MemcacheDB, Voldemort, RevenDB etc. are the examples of NoSQL databases.

### **What type of DBMS is MongoDB?**

MongoDB is a document oriented DBMS

### **Why MongoDB is known as best NoSQL database?**

MongoDb is the best NoSQL database because, it is:

Document Oriented

Rich Query language

High Performance

Highly Available

Easily Scalable

### **Does MongoDB support primary-key, foreign-key relationship?**

No. By Default, MongoDB doesn't support primary key-foreign key relationship.

### **Does MongoDB need a lot of RAM?**

No. There is no need a lot of RAM to run MongoDB. It can be run even on a small amount of RAM because it dynamically allocates and de-allocates

### **Explain the structure of ObjectID in MongoDB.**

ObjectID is a 12-byte BSON type. These are:

* 4 bytes value representing seconds
* 3 byte machine identifier
* 2 byte process id
* 3 byte counter

### **By default, which index is created by MongoDB for every collection?**

By default, the\_id collection is created for every collection by MongoDB.

### **What is a Namespace in MongoDB?**

Namespace is a concatenation of the database name and the collection name. Collection, in which MongoDB stores BSON objects.

### **If you remove an object attribute, is it deleted from the database?**

Yes, it be. Remove the attribute and then re-save() the object.

### **Does MongoDB database have tables for storing records?**

No. Instead of tables, MongoDB uses "Collections" to store data.

### **Do the MongoDB databases have schema?**

Yes. MongoDB databases have dynamic schema. There is no need to define the structure to create collections.

### **What is the method to configure the cache size in MongoDB?**

MongoDB's cache is not configurable. Actually MongoDb uses all the free spaces on the system automatically by way of memory mapped files.

### **Why 32 bit version of MongoDB are not preferred ?**

Because MongoDB uses memory mapped files so when you run a 32-bit build of MongoDB, the total storage size of server is 2 GB. But when you run a 64-bit build of MongoDB, this provides virtually unlimited storage size. So 64-bit is preferred over 32-bit.

### **Explain the covered query in MongoDB.**

A query is called covered query if satisfies the following two conditions:

* The fields used in the query are part of an index used in the query.
* The fields returned in the results are in the same index.

### **What is sharding in MongoDB?**

In MongoDB, Sharding is a procedure of storing data records across multiple machines. It is a MongoDB approach to meet the demands of data growth. It creates horizontal partition of data in a database or search engine. Each partition is referred as shard or database shard.

### **32) What is replica set in MongoDB?**

A replica can be specified as a group of mongo instances that host the same data set. In a replica set, one node is primary, and another is secondary. All data is replicated from primary to secondary nodes.

### **33) What is primary and secondary replica set in MongoDB?**

In MongoDB, primary nodes are the node that can accept write. These are also known as master nodes. The replication in MongoDB is single master so, only one node can accept write operations at a time.

Secondary nodes are known as slave nodes. These are read only nodes that replicate from the primary.

### **34) By default, which replica sets are used to write data?**

By default, MongoDB writes data only to the primary replica set.

### **What is CRUD in MongoDB?**

MongoDB supports following CRUD operations:

* Create
* Read
* Update
* Delete

### **Why are MongoDB data files large in size?**

MongoDB doesn't follow file system fragmentation and pre allocates data files to reserve space while setting up the server. That's why MongoDB data files are large in size.

### **39) What is a storage engine in MongoDB?**

A storage engine is the part of a database that is used to manage how data is stored on disk.

### **Which are the storage engines used by MongoDB?**

MMAPv1 and WiredTiger are two storage engine used by MongoDB.

### **What is the difference between MongoDB and Redis database?**

**Difference between MongoDB and Redis:**

* Redis is faster than MongoDB.
* Redis has a key-value storage whereas MongoDB has a document type storage.
* Redis is hard to code but MongoDB is easy.

### **What is the difference between MongoDB and Cassandra?**

**Difference between MongoDB and Cassandra:**

* MongoDB is cross-platform document-oriented database system while Cassandra is high performance distributed database system.
* MongoDB is written in C++ while Cassandra is written in Java.

