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.
* MongoDb stores data as documents. So it is a document oriented*.*
* Database: Database is a physical container for collections.
* Collection is a group of MongoDB documents. A collection exists within a single database.
* A document is a set of key-value pairs.

Why should use MongoDB

* Document Oriented Storage : Data is stored in the form of JSON style documents
* Index on any attribute
* Replication & High Availability
* Auto-Sharding
* Rich Queries
* Fast In-Place Updates
* Professional Support By MongoDB

## Where should use MongoDB?

* Big Data
* Content Management and Delivery
* Mobile and Social Infrastructure
* User Data Management
* Data Hub

Using commands

Create: use DATABASE\_NAME

> use db

switched to db db

> use cybage

switched to db cybage

> show dbs

admin (empty)

local 0.078GB

> db.cybage.insert({"emp\_name":"Priya”})

WriteResult({ "nInserted" : 1 })

> show dbs

admin (empty)

cybage 0.078GB

local 0.078GB

## The dropDatabase() Method

db.dropDatabase()

> use mydb

switched to db mydb

> show dbs

admin (empty)

cybage 0.078GB

local 0.078GB

> db.mydb.insert({"emp\_name":"priya"})

WriteResult({ "nInserted" : 1 })

> show dbs

admin (empty)

cybage 0.078GB

local 0.078GB

mydb 0.078GB

> use mydb

switched to db mydb

> db.dropDatabase()

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

> show dbs

admin (empty)

cybage 0.078GB

local 0.078GB

***Create Collection***

> use cybage

switched to db cybage

> db.createCollection("employees")

{ "ok" : 1 }

> show collections

cybage

employees

system.indexes

> show dbs

admin (empty)

cybage 0.078GB

local 0.078GB

> db.employees.drop()

true

> show collections

cybage

system.indexes

## The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()**method.

### Syntax

Basic syntax of **insert()** command is as follows −

>db.COLLECTION\_NAME.insert(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.

## The pretty() Method

To display the results in a formatted way, you can use **pretty()**method.

### Syntax

>db.mycol.find().pretty()

> db.employees.insert({name:'priya lohakare', bloodgroup:'O+', })

WriteResult({ "nInserted" : 1 })

> db.employees.find()

{ "\_id" : ObjectId("566553e80df0dadb881cd887"), "name" : "priya lohakare"

dgroup" : "O+" }

> db.employees.find().pretty()

{

"\_id" : ObjectId("566553e80df0dadb881cd887"),

"name" : "priya lohakare",

"bloodgroup" : "O+"

}

## MongoDB Update() method

The update() method updates values in the existing document.

### Syntax

Basic syntax of **update()** method is as follows

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

## MongoDB Save() Method

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

### Syntax

Basic syntax of mongodb **save()** method is shown below −

>db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA})

## 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:

Basic syntax of **remove()** method is as follows

>db.COLLECTION\_NAME.remove(DELLETION\_CRITTERIA)

> use cybage

switched to db cybage

> db.employees.find().pretty()

{

"\_id" : ObjectId("566553e80df0dadb881cd887"),

"name" : "priya bhatt",

"bloodgroup" : "O+"

}

> db.employees.insert({name:'priya lohakare', bloodgroup:' B+', })

WriteResult({ "nInserted" : 1 })

> db.employees.find().pretty()

{

"\_id" : ObjectId("566553e80df0dadb881cd887"),

"name" : "priya bhatt",

"bloodgroup" : "O+"

}

{

"\_id" : ObjectId("56669efd586f43acd82a529f"),

"name" : "priya lohakare",

"bloodgroup" : " B+"

}

> db.employees.remove({"name":"priya bhatt"})

WriteResult({ "nRemoved" : 1 })

> db.employees.find().pretty()

{

"\_id" : ObjectId("56669efd586f43acd82a529f"),

"name" : "priya lohakare",

"bloodgroup" : " B+"

}

## The sort() Method

To sort documents in MongoDB, you need to use **sort()** method.**sort()** method accepts a document containing 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.

### Syntax:

Basic syntax of **sort()** method is as follows

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

## The aggregate() Method

For the aggregation in mongodb you should use **aggregate()**method.

### Syntax:

Basic syntax of **aggregate()** method is as follows

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

# > db.records.insert({ title: 'MongoDB Overview', description: 'MongoDB is no

# sql database', by: 'tutorials point', url: 'http://www.tutorialspoint.com

# ', tags: ['mongodb', 'database', 'NoSQL'], likes: 100 })

# WriteResult({ "nInserted" : 1 })

# > db.reocrds.find().pretty()

# > db.records.find().pretty()

# {

# "\_id" : ObjectId("5669282e5112ec2e7c1f0cdb"),

# "title" : "MongoDB Overview",

# "description" : "MongoDB is no sql database",

# "by" : "tutorials point",

# "url" : "http://www.tutorialspoint.com",

# "tags" : [

# "mongodb",

# "database",

# "NoSQL"

# ],

# "likes" : 100

# }

# > db.records.insert({ title: 'tutorial point', description: 'tutorialp point

# gives tutorial on various lang', by: 'tutorials point', url: 'http://www.t

# utorialspoint.com', tags: ['mongodb', 'database', 'NoSQL'], likes: 100 })

# WriteResult({ "nInserted" : 1 })

# > db.records.find().pretty()

# {

# "\_id" : ObjectId("5669282e5112ec2e7c1f0cdb"),

# "title" : "MongoDB Overview",

# "description" : "MongoDB is no sql database",

# "by" : "tutorials point",

# "url" : "http://www.tutorialspoint.com",

# "tags" : [

# "mongodb",

# "database",

# "NoSQL"

# ],

# "likes" : 100

# }

# {

# "\_id" : ObjectId("5669372c5112ec2e7c1f0cdc"),

# "title" : "tutorial point",

# "description" : "tutorialp pointgives tutorial on various lang",

# "by" : "tutorials point",

# "url" : "http://www.tutorialspoint.com",

# "tags" : [

# "mongodb",

# "database",

# "NoSQL"

# ],

# "likes" : 100

# }

# > db.records.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}

# ])

# { "\_id" : null, "num\_tutorial" : 2 }

# Replication:

Replication is the process of synchronizing data across multiple servers.\

Replication also allows you to recover from hardware failure and service interruptions

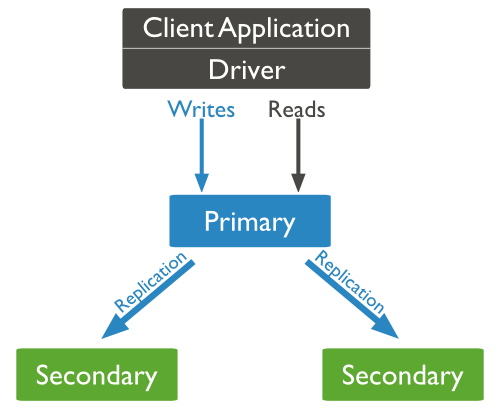
Why Replication?

* To keep your data safe
* High (24\*7) availability of data
* Disaster Recovery
* No downtime for maintenance (like backups, index rebuilds, compaction)
* Read scaling (extra copies to read from)
* Replica set is transparent to the application

How replication works in MongoDB

MongoDB achieves replication by the use of replica set. A replica set is a group of **mongod** instances that host the same data set. In a replica one node is primary node that receives all write operations. All other instances, secondaries, apply operations from the primary so that they have the same data set. Replica set can have only one primary node.

1. Replica set is a group of two or more nodes (generally minimum 3 nodes are required).
2. In a replica set one node is primary node and remaining nodes are secondary.
3. All data replicates from primary to secondary node.
4. At the time of automatic failover or maintenance, election establishes for primary and a new primary node is elected.
5. After the recovery of failed node, it again join the replica set and works as a secondary node.



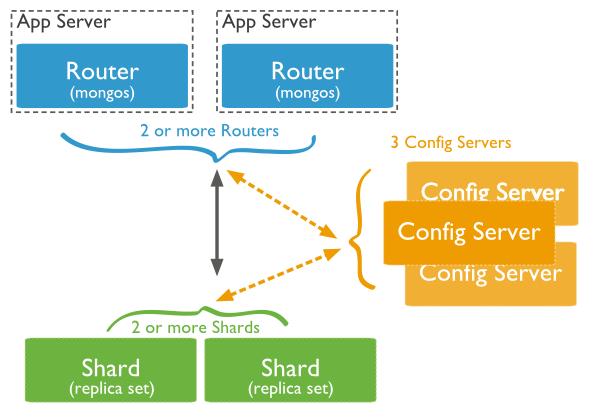
## Sharding

Sharding is the process of storing data records across multiple machines and it is MongoDB's approach to meeting the demands of data growth.

With sharding, you add more machines to support data growth and the demands of read and write operations.

## Why Sharding?

* In replication all writes go to master node
* Latency sensitive queries still go to master
* Single replica set has limitation of 12 nodes
* Memory can't be large enough when active dataset is big
* Local Disk is not big enough
* Vertical scaling is too expensive



## Dump MongoDB Data

To create backup of database in mongodb you should use**mongodump** command. This command will dump all data of your server into dump directory.

### Syntax:

Basic syntax of **mongodump** command is as follows

>mongodump

## Restore data

To restore backup data mongodb's **mongorestore** command is used. This command restore all of the data from the back up directory.

### Syntax

Basic syntax of **mongorestore** command is

>mongorestore