MONGODB

**What is MongoDb?**

* *MongoDb* is a Open Source database written in C++.
* Drivers and client libraries are typically written in their respective languages, although some drivers use C extensions for better performance.
* It can be used to store data for very high performance applications
* MongoDB does not support SQL It supports a rich, ad-hoc query language of its own.
* MongoDb stores data as documents. So it is a *document oriented database*.
* Notice there are two different documents (separated by "."). Storing data in this fashion is called as *document oriented database*. *MongoDb* is a document oriented database.

**Databases :**

MongoDB does not support joins nor transactions. However, it features secondary indexes, an expressive query language, atomic writes on a per-document level, and fully-consistent reads. MongoDB uses [BSON](http://www.w3resource.com/JSON/BSON.php), a binary object format similar to, but more expressive than JSON.

**Schemas :**

We can create collections without defining the structure, i.e. the fields or the types of their values, of the documents. You can change the structure of documents simply by adding new fields or deleting existing ones. Documents in a collection need unique set of fields.

**Tables :**

MongoDB database stores its data in collections not in tables A collection holds one or more documents, which corresponds to a record or a row in a relational database table, and each document has one or more fields, which corresponds to a column in a relational database table.

## Advantages of MongoDB over RDBMS

* Schema less : MongoDB is document database in which one collection holds different different documents. Number of fields, content and size of the document can be differ from one document to another.
* Structure of a single object is clear
* No complex joins
* Ease of scale-out: MongoDB is easy to scale
* Uses internal memory for storing the (windowed) working set, enabling faster access of data

## Why should use MongoDB

* Document Oriented Storage : Data is stored in the form of JSON style documents
* Index on any attribute
* Replication & High Availability
* 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

## use Command

MongoDB **use DATABASE\_NAME** is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database.

### Syntax:

Basic syntax of **use DATABASE** statement is as follows:

use DATABASE\_NAME

## The dropDatabase() Method

MongoDB **db.dropDatabase()** command is used to drop a existing database.

### Syntax:

Basic syntax of **dropDatabase()** command is as follows:

db.dropDatabase()

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

### Syntax:

Basic syntax of **createCollection()** command is as follows

db.createCollection(name, options)

## drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

### Syntax:

Basic syntax of **drop()** command is as follows

db.COLLECTION\_NAME.drop()

## 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)

### Example

>db.mycol.insert({

\_id: ObjectId(7df78ad8902c),

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

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

likes: 100

})

## 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()

## 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)

## 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)

In MongoDB the conditional operators are :

(>) greater than - $gt

(<) less than - $lt

(>=) greater than equal to - $gte

(<= ) less than equal to - $lte

Eg:

db.testtable.find({age : {$gt : 22}}).pretty();

by using rename operator:

db.employee.update( { emp\_id: 1232 }, { $rename: { 'offinfo': 'service\_details', 'emale': 'email' } } );

by using variable:

document=({"user\_id" : "QRSTBWN","password" :"QRSTBWN" ,"date\_of\_join" : "17/10/2010" ,"education" :"M.B.A." , "profession" : "MARKETING","interest" : "MUSIC","community\_name" :["MODERN MUSIC", "CLASSICAL MUSIC","WESTERN MUSIC"],"community\_moder\_id" : ["MR. BBB","MR. JJJ","MR MMM"],"community\_members" : [500,200,1500],"friends\_id" : ["MMM123","NNN123","OOO123"],"ban\_friends\_id" :["BAN123","BAN456","BAN789"]});

db.userdetails.insert(document)

update command:

db.userdetails.update({"user\_id" : "QRSTBWN"},{"user\_id" : "QRSTBWN","password" :"NEWPASSWORD" ,"date\_of\_join" : "17/10/2010" ,"education" :"M.B.A." , "profession" : "MARKETING","interest" : "MUSIC","community\_name" :["MODERN MUSIC", "CLASSICAL MUSIC","WESTERN MUSIC"],"community\_moder\_id" : ["MR. BBB","MR. JJJ","MR MMM"],"community\_members" : [500,200,1500],"friends\_id" : ["MMM123","NNN123","OOO123"],"ban\_friends\_id" :["BAN123","BAN456","BAN789"]});

**ASCENDING**

db.employee.aggregate(

{

$sort : {salary : 1}

}

);

**DESECENDING**

db.employee.aggregate(

{

$sort : {salary : -1}

}

);

**AVERAGE :**

db.employee.aggregate(

{

$group : {

\_id : "all",

avg\_score : { $avg : "$salary" },

}

});

**MINIMUM:**

db.employee.aggregate(

{

$group : {

\_id : "all",

minimum : { $min: "$salary" },

}

});

**MAXIMUM:**

db.employee.aggregate(

{

$group : {

\_id : "all",

maximum : { $max: "$salary" },

}

});

**index**

db.COLLECTION\_NAME.ensureIndex({KEY:1})

db.employee.ensureIndex({“name”:1})

{ $group: { \_id: "$dateofjoining", avgsalPop: { $avg: "$salary" } } }

**Create Index**

**db.students.ensureIndex ({student\_id:1})**  
This creates an index on student\_id in collection students in ascending order

**> db.employees.ensureIndex({employee\_id:1})**  
{  
“createdCollectionAutomatically” : false,  
“numIndexesBefore” : 1,  
“numIndexesAfter” : 2,  
“ok” : 1  
}

**Creating indexes on multiple keys**  
**db.students.ensureIndex ({student\_id:1,age:-1})**  
student\_id is ascending  
age is descending

**Drop Index**

**> db.employees.dropIndex({first\_name:1})**  
{ “nIndexesWas” : 3, “ok” : 1 }

**Discover Index**

**db.system.indexes.find()**

> db.employees.getIndexes()

### Creating a Unique Index and drop duplicates

**db.employees.ensureIndex({employee\_id:1},{unique:true,dropDups:true})**

### Using Hint Option

**db.employees.find().sort({employee\_id:1}).hint({employee\_id:1})**

Important link <http://www.w3resource.com/mongodb/introduction-to-mongodb-index.php>