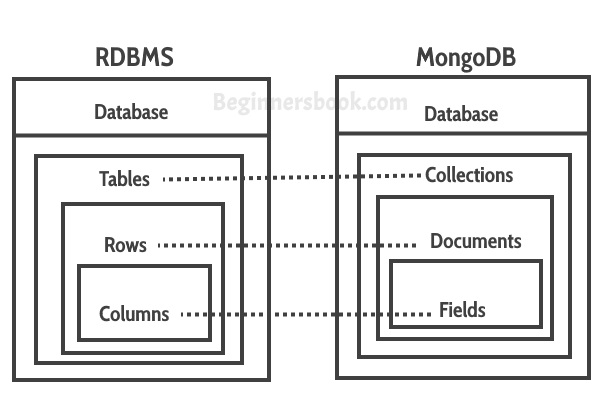
**MongoDB Basics**

****

**Database**

1. **Create database**

* use database\_name

if database is already present then this command will connect you to the database, otherwise it will create a new db.

To check for currently connected database write command,

* db

To display the list of created databases write command,

* show dbs

A database is not created unless we store a document in it.

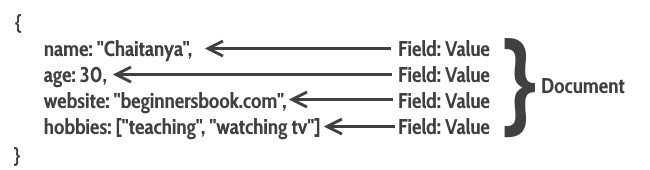
1. **Drop Database**

* db.dropDatabase()

This command will delete the currently selected database.

So, first we have to use the database that we want to delete and then execute the command.

**Collections**

****

1. **Create Collection**

We can create collections on fly just by inserting values directly to a collection, in the form of key-value pairs

* **db.collection\_name.insert({key:value, key:value…})**

**Example:-** db.user.insert({

name: "Samyak",

age: 21,

website: "xyz.com"

})

To display all the documents inside a collection, we can use the command-

* **db.collection\_name.find()**

To display the list of created collections inside a database execute the command,

* **show collections**

Or we can specify the options for a collection before inserting any values in it and basic syntax is,

* db.createCollection(name, options)

where,

**name** is collection name

**options** is an optional parameter which we can use to specify certain things such as size, max number of documents etc.

* db.createCollection("students")

one such option can be,

autoIndexID : type : Boolean

* default is false, if set true then it automatically creates index field \_id for each document.

MongoDB automatically inserts a unique \_id(12-byte field) field in every document, this serves as primary key for each document.

1. **Drop Collection**

* db.collection\_name.drop()

First select the database, then delete the collection that you want.

This will completely delete the collection from the database, if you want to delete the documents present inside a collection only then we can execute the ‘remove’ command,

* db.collection\_name.remove( { } )

The remove method will also retain the indexes created, and to delete all the documents inside the collection we give an empty query option ‘{ }’.

1. **Inserting Documents**

* db.collection\_name.insert()

Example,

* db.user.insert(

{

name: "Samyak",

age: 21,

email: "xyz.com",

course: [ { name: "MongoDB", duration: 7 }, { name: "DotNet", duration: 30 } ]

}

)

Can also insert multiple documents at a time through passing an array variable,

var beginners =

[

{

"StudentId" : 1001,

"StudentName" : "Samyak",

"age": 21

},

{

"StudentId" : 1002,

"StudentName" : "Rajul",

"age": 21

},

{

"StudentId" : 3333,

"StudentName" : “Gopesh",

"age": 22

},

];

db.students.insert(beginners);

1. **Query Document**

To Display the documents inside a collection in JSON format (easy to read),

* db.students.find().forEach(printjson)

Or we can use the ‘pretty’ method also, it does the same thing,

* db.students.find().pretty()

We can define criteria to retrieve specific values from a collection,

Example,

For specific field value,

* db.students.find({StudentName : "Samyak"}).pretty()

**Greater than criteria**

* db.collection\_name.find({"field\_name":{$gt:criteria\_value}}).pretty()

**Less than criteria**

* db.collection\_name.find({"field\_name":{$lt:criteria\_value}}).pretty()

**Not equals Criteria**

* db.collection\_name.find({"field\_name":{$ne:criteria\_value}}).pretty()

Similarly, for **Greater than equal** and **Less than equal** we have “$gte” and “$lte” respectively.

1. **Update Document**

Two ways to update documents in a collection

1. Using update() method, to update existing values
2. Using save() method, to replace the existing document with the document that has been passed in it.

**Update() method**

* db.collection\_name.update(criteria, update\_data)

Example,

* db.got.update({"name":"Jon Snow"},{$set:{"name":"Kit Harington"}})

By default it will only one document in the collection, if there are multiple documents satisfying the criteria then we have to enable the “multi” parameter.

Example, db.got.update({"name":"Jon Snow"},

{$set:{"name":"Kit Harington"}},{multi:true})

**Save() method**

* db.collection\_name.save( {\_id:ObjectId(), new\_document} )

We have to specify the unique id for a particular document to replace it with new document.

If we don’t specify the id in the save method then, the save method will call the insert method and add the document as a new document in the collection

To get the \_id of a particular document we can execute,

var user = db.users.findOne({name:"Kit harington"})

var ID = user.\_id

then execute the save method,

* db.got.save({“\_id”:ID, "name":

"Jon Snow", "age": 30})

1. **Deleting Documents**

* db.collection\_name.remove(delete\_criteria)

To delete the documents with specified criteria

Example,

* db.students.remove({"StudentId": 3333})

If there are multiple documents matching the criteria and we want to delete only one document that first matches the criteria then we can use the “**justOne**” parameter.

* db.collection\_name.remove(delete\_criteria, justOne)

Example,

* db.walkingdead.remove({"age": 32}, 1)