**Concept of MONGODB**

SQL: Structure query language

RDBMS: Relational database management system

Database: collection of data in a structured way

**What is noSql?**

A nosql database is a key value database

It is not is the form of table.

Different types of nosql are: MongoDB, Amazon DocumenrDb, Google Datastore, Amazon DynamoBd ect.

Extremely useful,powerful and high performance database in largr big data applications, large distributed network architure apps ect.

**What is mongoDB:**

Open and free source,cross platform(any platform can run),

It used Json-like document with schema

**What is document oriented database?**

**What Mongodb is not?**

It is not a RDMS

Doesnot have any concept of Joins.

Not tough or complicated.

Languages supports:

PHP,Node Js, Python, java, c#, C++

Why learn MongoDB?

It

IN mongo db

Database

Collections

Documents:

Document can have any data type – long as it is valid mongodb data type

It is simply key-value pair data

Example: {

“firstname”: “Dasari”,

“lastname”:”Samanth”,

“Email”: “Dasari.samanth@gamil.com”

},

{

“firstname”: “siri”,

“lastname”:”sk”,

“Email”: siri@gmail.com

};

**Collections:**

It’s a set of documents

Can have any number of documents

Documents can have any dynamic schema

They can be same or different

No join concept

**Database:**

Sing collection or more collections

**Queries :**

**Creating and dropping data base:**

* Show database; :- it show the data in our database.
* Use databasename(CMR) :use to switch into CMR data base.
* Db: it shows in which database we are in.
* Db.dropDatabase(); :-it used to deleted the database in which we are present.
* To drop a database, first we need to select the DB

---use (database – name>

---db.dropDatabase();

**Creating and dropping collections:**

**Db.createCollection(“< name of collection”);:-creating collections in database.**

**Db.collectionName.drop():- dropped the collection from the database.**

**Let us know about few more documents in this session:**

SQL vs Document Databases

SQL databases are considered relational databases. They store related data in separate tables. When data is needed, it is queried from multiple tables to join the data back together.

MongoDB is a document database which is often referred to as a non-relational database. This does not mean that relational data cannot be stored in document databases. It means that relational data is stored differently. A better way to refer to it is as a non-tabular database.

MongoDB stores data in flexible documents. Instead of having multiple tables you can simply keep all of your related data together. This makes reading your data very fast.

You can still have multiple groups of data too. In MongoDB, instead of tables these are called collections.

Create Database using mongosh

After connecting to your database using mongosh, you can see which database you are using by typing db in your terminal.

If you have used the connection string provided from the MongoDB Atlas dashboard, you should be connected to the myFirstDatabase database.

Show all databases

To see all available databases, in your terminal type show dbs.

Notice that myFirstDatabase is not listed. This is because the database is empty. An empty database is essentially non-existant.

Create Collection using mongosh

There are 2 ways to create a collection.

Method 1

You can create a collection using the createCollection() database method.

Example

db.createCollection("posts")

Method 2

You can also create a collection during the insert process.

Example

We are here assuming object is a valid JavaScript object containing post data:

db.posts.insertOne(object)

Insert Documents

There are 2 methods to insert documents into a MongoDB database.

insertOne()

To insert a single document, use the insertOne() method.

This method inserts a single object into the database.

updateOne()

The updateOne() method will update the first document that is found matching the provided query.

Let's see what the "like" count for the post with the title of "Post Title 1":

Example

db.posts.find( { title: "Post Title 1" } )

Now let's update the "likes" on this post to 2. To do this, we need to use the $set operator.

Example

db.posts.updateOne( { title: "Post Title 1" }, { $set: { likes: 2 } } )

Check the document again and you'll see that the "like" have been updated.

Example

db.posts.find( { title: "Post Title 1" } )

Insert if not found

If you would like to insert the document if it is not found, you can use the upsert option.

Example

Update the document, but if not found insert it:

db.posts.updateOne(

{ title: "Post Title 5" },

{

$set:

{

title: "Post Title 5",

body: "Body of post.",

category: "Event",

likes: 5,

tags: ["news", "events"],

date: Date()

}

},

{ upsert: true }

)

* Insert if not found

If you would like to insert the document if it is not found, you can use the upsert option.

Example

Update the document, but if not found insert it:

db.posts.updateOne(

{ title: "Post Title 5" },

{

$set:

{

title: "Post Title 5",

body: "Body of post.",

category: "Event",

likes: 5,

tags: ["news", "events"],

date: Date()

}

},

{ upsert: true }

)

updateMany()

The updateMany() method will update all documents that match the provided query.

Example

Update likes on all documents by 1. For this we will use the $inc (increment) operator:

db.posts.updateMany({}, { $inc: { likes: 1 } })

Now check the likes in all of the documents and you will see that they have all been incremented by 1

* Delete Documents

We can delete documents by using the methods deleteOne() or deleteMany().

These methods accept a query object. The matching documents will be deleted.

deleteOne()

The deleteOne() method will delete the first document that matches the query provided.

Example

db.posts.deleteOne({ title: "Post Title 5" })

deleteMany()

The deleteMany() method will delete all documents that match the query provided.

Example

db.posts.deleteMany({ category: "Technology" })