**------------MONGO DB--------------**

**Database**

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

**Collection**

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

**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, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

|  |  |
| --- | --- |
| **RDBMS** | **MongoDB** |
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (Default key \_id provided by mongodb itself) |

**Advantages of MongoDB over RDBMS:**

**1.Schema less** − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.

2.Structure of a single object is clear.

3.No complex joins.

Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

Tuning.

Ease of scale-out − MongoDB is easy to scale.

Conversion/mapping of application objects to database objects not needed.

Uses internal memory for storing the (windowed) working set, enabling faster access of data.

**Why Use MongoDB?**

1.Document Oriented Storage − Data is stored in the form of JSON style documents.

2.Index on any attribute

3.Replication and high availability

4.Auto-sharding

5.Rich queries

6.Fast in-place updates

7.Professional support by MongoDB

**Where to Use MongoDB?**

1. Big Data

2. Content Management and Delivery

3. Mobile and Social Infrastructure

4. User Data Management

5. Data Hub

1. installation (for 32 Bit):
2. Crate folder in C:\data

**“data”** is folder name in C drive

3. To Start MONGODB Server:

\* C:\Program Files\MongoDB\Server\3.0\bin>mongod.exe --dbpath C:\data -- journal

**“C:\Program Files\MongoDB\Server\3.0\bin>” ->** The path installed of mongodb

**“mongod.exe --dbpath C:\data -- journal” ->** To run mongo Db server

1. C:\Program Files\MongoDB\Server\3.0\bin

Open  **“Mongo” ->new** window is opened

**Create Database:**

MongoDB **use DATABASE\_NAME** is used to create database.

**Syntax:** use Database-name

**Eg**:

**1**. use jobs // create database

**2.** db.jobseekers.insert({name:”kumar”}) // creating database and insert row

**> Show dbs**

Jobseekers 0.78 GB

admin 0.078GB

config 0.078GB

holidaysway 0.078GB

local 0.078GB

Se 0.9 GB (like this)

> **use holidaysway**

switched to db holidaysway

> **show collections**

advertises

airports

citybreaks

contacts

csvs

enquiries

handpickeds

settings

sponsors

users

> **db.airports.find()** (for checking collections)

**CREATE Collection(Table):**

Basic syntax of **createCollection()** method without options is as follows −

>use test

switched to db test

>db.createCollection("mycollection") // table/collection name

{ "ok" : 1 }

>

MYSQL Insert file Through CMD:

**C:\Users\adroit\Desktop**

**C:\xampp\mysql\bin>mysql -u root -p node\_adverts <C:\Users\adroit\Desktop\node\_adverts.sql**

**Storage Engines: 1.Wider tiger(** Defualt For MongoDB versions After 3.2.**)**

**2. MMAPV1(**Defualt For MongoDB versions before 3.2.**)**

**3.In –Memory**

**1.** Mongo db is a Open source Document Database that provides high perfomance , high availability and Automatic Scaling

\* A Record in a MongoDB is a Document.

**2**.The data structure composed of “FIELD AND VALUES”

Ex: { name:”kumar”, age:26, groups:[‘A’,’B’]}

Name 🡪 Field

“kumar” 🡪 Value

The advantages of using documents are:

* Documents (i.e. objects) correspond to native data types in many programming languages.
* Embedded documents and arrays reduce need for expensive joins.
* Dynamic schema supports fluent polymorphism.

**To insert one Document:**

db.collection.insert ( {name:”kumar”, age:25, group:[‘a’,’b’]} )

Collection 🡪 Tablename (users)

**To Insert More Documents At a time:**

db. users.insert ( [{name:”kumar”, age:25, group:[‘a’,’b’]} ,{name:”hari”,age:25}])

**Get Data:**

**To Get All information about Document:**

db.users,find ({})

**To get user name who is kumar only**

db.users.find ({name: ”kumar” })

**Update:**

**To update one value:**

db.users.update( {name:”kumar”}, {$set: {name:”swamy”} } ).

**Delete:**

db.users.remove( {name:”kumar”} ). // delete particular records

db.user.remove( {} ) //to delete all records

**remove/drop collection/table:**

db.collectionname.drop()

**remove/drop database:**

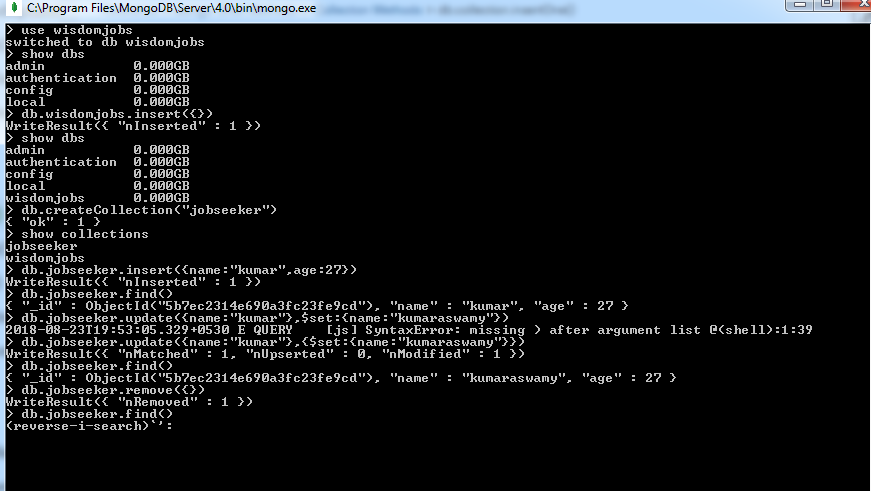
db.dropDatabase()

**Step by step procedure:**

**Database:** wisdomjobs

**Table:** jobseeker, employer

1. use wisdomjobs
2. db.wisdomjobs.insert({})
3. show dbs
4. db.createCollection(“jobseeker”)
5. db.jobseeker.insert({name: “kumar”,age:25 })
6. db.jobseeker.find()
7. db.jobseeker.update({name: “kumar},{$set:{name:”kumarswamy”}})
8. db.jobseeker.remove({})
9. db.jobseeker.drop()



**1. I want to query something as SQL's like query**:

select \*

from users

where name like '%m%'

in mongo db:

db.users.find({"name": /m/})

Example:

db.users.insert({name: 'paulo'})

db.users.insert({name: 'patric'})

db.users.insert({name: 'pedro'})

db.users.find({name: /a/}) //like '%a%'

out: paulo, patric

db.users.find({name: /^pa/}) //like 'pa%'

out: paulo, patric

db.users.find({name: /ro$/}) //like '%ro'

out: pedro

**2. count (displays the total number of records in collection)**

db.users.count() // o/p:9

**3.inserting data using save():**

db.users.save ( {email:”kumar”, password:”12345”} )

**4. sorting:**

A) db.users.find().sort({name:1}) // for ascending order

B) db.users.find().sort({name:-1}) // for descending order

# 5. [Add new field to every document in a MongoDB collection](https://stackoverflow.com/questions/7714216/add-new-field-to-every-document-in-a-mongodb-collection)

db.your\_collection.update({},

{$set : {"new\_field":1}},

{upsert:false, multi:true})

eg: db.users.update.( {},{ $set:{“age”:25}},{ upsert:false, multi:true})

# 6.[How to remove a field completely from a MongoDB document?](https://stackoverflow.com/questions/6851933/how-to-remove-a-field-completely-from-a-mongodb-document)

[Ask Question](https://stackoverflow.com/questions/ask)

db.users.update({}, {$unset: {age:1}} , {multi: true});

**7. where condition**

A) db.user.remove({“email”:”kumar”}) // remove one record where email matches “kumar”

# B) [MongoDB multiple condition in WHERE clause](https://stackoverflow.com/questions/24513766/mongodb-multiple-condition-in-where-clause)

# Mysql:

DELETE FROM tablename WHERE id = 6 OR id =8

# MongoDB: ($or)

db.tablename.remove({ $or: [ { \_id: 6 }, { \_id: 8 } ] })

**8. Update one field:**

db.users.update({“name”:”kumar”},{$set:{“age”:27}})

**9. Find data based on fields:**

db.users.find({“age”:25})