MongoDB Notes

>> It is a NoSQL Database

>> It is a non relational Document oriented Dbms and works on document based Db.

>> It stores data in form of documents.

```
user document
{
_id:<Objid1>,
username:"124xyx"
}
```

>> Uses Bson to query Database

>> JSON and BSON are cousins

Binary Script Object Notation... To include data values like : Date & binary value which cannot be done by

JSON and with BSON read and write transactions will be more faster and flexible....

We will use JSON (which is readable for human and device) to store records to database-document but MongoDB

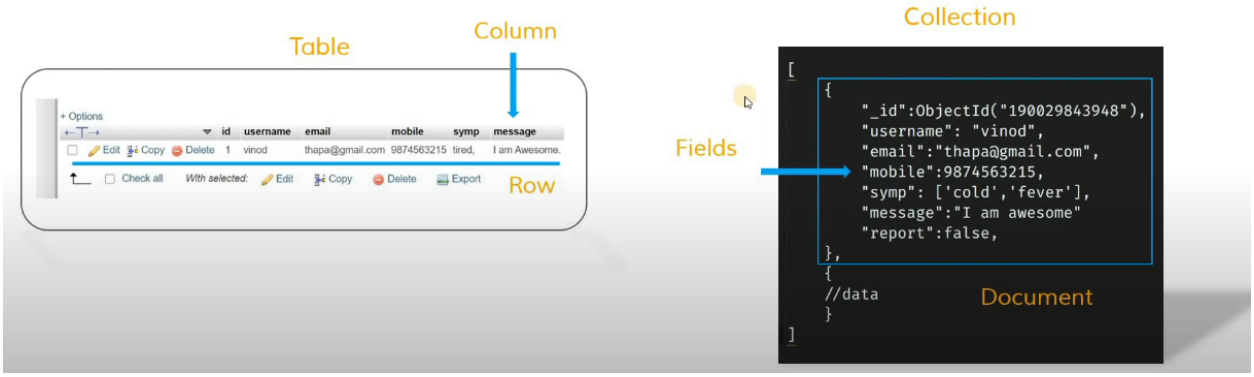
will automatically convert it into BSON data (which is only readable for devices)

>> Bson is more faster and flexible & Scalable.

>> Rows in sql = Document ({…}) in MongoDb

>> Columns in Sql = Fields in Mongo (Key,value pairs)

>> This combined is called collection in MongoDb



>Commands (C.R.U.D)

Creating- Db

- `use (database name) – To create a new db`
- `db.Project1.insertOne({name:" reactjs",type:"Front end",videos:80,active:true}) –> To insert data in db`

```
Project1> db.Project1.insertOne({name:" reactjs",type:"Front end",videos:80,active:true})
{
  acknowledged: true,
  insertedId: ObjectId("64d77501cbae5dfa1ad56c41")
```

- `We can also use insertMany to insert multiple documents in db inside ([{},{},{}…..and so on])`

```
Project1> db.Project1.insertMany([{name:"node"},{age:45},{job:"chaprasi"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64d7787bcd757233412086d8"),
    '1': ObjectId("64d7787bcd757233412086d9"),
    '2': ObjectId("64d7787bcd757233412086da")
  }
}
```

- `db – To check the current Db`
- `show collections – to show collections`

```
Project1> show collections
Project1
```

- db.(collection name).find()– to get the documents in db

```
Project1> db.Project1.find()
[
  { _id: ObjectId("64d774c0cbae5dfa1ad56c40") },
  {
    _id: ObjectId("64d77501cbae5dfa1ad56c41"),
    name: ' reactjs',
    type: 'Front end',
    videos: 80,
    active: true
  }
]
Project1>
```

Reading - Db

- `db.collection.find(query,projection)`
- Projection -**a special feature allowing you to select only the necessary data rather than selecting the whole set of data from the document.**
- To query a particular data we use- db.Project1.find({videos:80})

```
Project1> db.Project1.find({videos:80})
[
  {
    _id: ObjectId("64d77501cbae5dfa1ad56c41"),
    name: ' reactjs',
    type: 'Front end',
    videos: 80,
    active: true
  }
]
```

- To exactly get the particular field we use : db.Project1.find({videos:80},{videos:1})
- By default we get the id also so to get rid of that we can us :`db.Project1.find({videos:80},{_id:0,videos:1})`

```
Project1> db.Project1.find({videos:80},{videos:1})
[ { _id: ObjectId("64d77501cbae5dfa1ad56c41"), videos: 80 } ]
```

```
Project1> db.Project1.find({videos:80},{_id:0,videos:1})
[ { videos: 80 } ]
Project1>
```

- If in the documents if we have same data we and we only want to show 1 we can use limit method

  db.Project1.find({repeated query}).limit(1)

  OR

  We can use findOne

  db.Project1.findOne({repeated query})

  - If in case we want to show only the 2nd document among the repeated Docs we can use skip(1) method

    db.Project1.findOne({repeated query}).skip(1)

Update - Db

- `db.collection_name.updateOne(<filter>,<update>)`
- `db.collection_name.updateMany(<filter>,<update>)`
- `Update the value of the field using $set:{} method`
- db.Project.updateOne({name:" reactjs"},{$set: {videos:100}}) –initially which was 85

```
Project1> db.Project.updateOne({name:" reactjs"},{$set:{videos:100}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

same with updateMany

Delete- Db

- `db.collection.deleteOne(<filter>)`
- `db.collection.deleteMany(<filter>)`
- db.Project1.deleteOne({videos:80})

```
Project1> db.Project1.deleteOne({videos:80})
{ acknowledged: true, deletedCount: 1 }
```

- We can also use db.colletion.remove but it is depcreated as per docs

  the diff b/w the delte and remove is that it dosent give the ackw

- To Delete all the documents we simply leave the filter empty.

```
Project1> db.Project1.deleteMany({})
{ acknowledged: true, deletedCount: 6 }
Project1>
```

  - db.Project1.deleteMany({})
- To simple Delete the entire DataBase we can use:

`db.dropDatabase()`