**MONGO DB**

* One object is called document.
* Group of objects are called collection.
* **Definition:** Mongo db uses JSON like documents with optional schemas.
* Mongo db uses “JSON” like documents e.g. “BSON” document. Not exact “JSON” document.
* **BSON:** Itis knows as Binary JSON. In Mongo db data is stored in BSON in binary form.
* Objects contain key value pairs (field : value)
* We are also having crud operation in mongo db.
* Crud means create, update, delete

**PRACTICAL WORK**

Microsoft Windows [Version 10.0.19045.2604]

(c) Microsoft Corporation. All rights reserved.

C:\Users\Abdul Wahid>mongo

MongoDB shell version v5.0.15

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("0f4f6812-9fca-439a-b130-6e927588a849") }

MongoDB server version: 5.0.15

================

Warning: the "mongo" shell has been superseded by "mongosh",

which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in

an upcoming release.

For installation instructions, see

https://docs.mongodb.com/mongodb-shell/install/

================

---

The server generated these startup warnings when booting:

2023-02-26T06:32:22.339-08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> show collections

> use Abduldb

switched to db Abduldb

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> db.students.insertOne({name:"saeed ali",grade:"A"})

{

"acknowledged" : true,

"insertedId" : ObjectId("63fb7241c97bc8853c7fcdac")

}

> show dbs

Abduldb 0.000GB

admin 0.000GB

config 0.000GB

local 0.000GB

> ^C

bye

C:\Users\Abdul Wahid>mongo

MongoDB shell version v5.0.15

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("39e63b24-dbb7-4f0a-bd5d-c44453b47334") }

MongoDB server version: 5.0.15

================

Warning: the "mongo" shell has been superseded by "mongosh",

which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in

an upcoming release.

For installation instructions, see

https://docs.mongodb.com/mongodb-shell/install/

================

---

The server generated these startup warnings when booting:

2023-02-26T06:32:22.339-08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

> show dbs

Abduldb 0.000GB

admin 0.000GB

config 0.000GB

local 0.000GB

> show collections

> use Abduldb

switched to db Abduldb

> show collections

students

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

> ^C

bye

C:\Users\Abdul Wahid>mongo

MongoDB shell version v5.0.15

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("b4d18cdf-7db3-40c8-be1b-caf50391561c") }

MongoDB server version: 5.0.15

================

Warning: the "mongo" shell has been superseded by "mongosh",

which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in

an upcoming release.

For installation instructions, see

https://docs.mongodb.com/mongodb-shell/install/

================

---

The server generated these startup warnings when booting:

2023-02-26T06:32:22.339-08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

> show dbs

Abduldb 0.000GB

admin 0.000GB

config 0.000GB

local 0.000GB

> use Abduldb

switched to db Abduldb

> show collections

students

> db.personel.insertOne({gender:"male",interest:"cricket"})

{

"acknowledged" : true,

"insertedId" : ObjectId("63fb7384d0ac93d762d06f25")

}

> show dbs

Abduldb 0.000GB

admin 0.000GB

config 0.000GB

local 0.000GB

> show collections

personel

students

> db.personel.find()

{ "\_id" : ObjectId("63fb7384d0ac93d762d06f25"), "gender" : "male", "interest" : "cricket" }

> db.students.insertOne({name:"Abrar Ahmed",grade:"A"})

{

"acknowledged" : true,

"insertedId" : ObjectId("63fb7deed0ac93d762d06f26")

}

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

> db.student.insertMany([{name:"mussafara",grade:"A"},{name:"zehra", grade:"A"},{name:"yumna",grade:"A"}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("63fb7f19d0ac93d762d06f27"),

ObjectId("63fb7f19d0ac93d762d06f28"),

ObjectId("63fb7f19d0ac93d762d06f29")

]

}

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

> db.student.find()

{ "\_id" : ObjectId("63fb7f19d0ac93d762d06f27"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f19d0ac93d762d06f28"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f19d0ac93d762d06f29"), "name" : "yumna", "grade" : "A" }

> db.students.insertMany([{name:"mussafara",grade:"A"},{name:"zehra", grade:"A"},{name:"yumna",grade:"A"}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("63fb7f4ed0ac93d762d06f2a"),

ObjectId("63fb7f4ed0ac93d762d06f2b"),

ObjectId("63fb7f4ed0ac93d762d06f2c")

]

}

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "yumna", "grade" : "A" }

> show collections

personel

student

students

> db.students.find({name:"yumna"})

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "yumna", "grade" : "A" }

> db.students.insertOne({name:"yumna",grade:"B"})

{

"acknowledged" : true,

"insertedId" : ObjectId("63fb8187d0ac93d762d06f2d")

}

> db.students.find({name:"yumna"})

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "yumna", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "yumna", "grade" : "B" }

> db.students.find().limit(1)

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

> db.students.find({name:"yumna"}).limit(1)

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "yumna", "grade" : "A" }

> db.students.find({name:"yumna"}).limit(2)

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "yumna", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "yumna", "grade" : "B" }

> db.students.updateOne({name:"yumna"},{$set{name:"Yumna Afsar"}})

uncaught exception: SyntaxError: missing : after property id :

@(shell):1:42

> db.students.updateOne({name:"yumna"},{$set:{name:"Yumna Afsar"}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "Yumna Afsar", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "yumna", "grade" : "B" }

> db.students.findOne({name:"yumna"})

{

"\_id" : ObjectId("63fb8187d0ac93d762d06f2d"),

"name" : "yumna",

"grade" : "B"

}

> db.students.insertMany({name:"yumna",grade:"B"},{name:"yumna",grade:"B"})

uncaught exception: TypeError: documents.map is not a function :

DBCollection.prototype.insertMany@src/mongo/shell/crud\_api.js:307:17

@(shell):1:1

> db.students.insertMany([{name:"yumna",grade:"B"},{name:"yumna",grade:"B"}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("63fb8745d0ac93d762d06f2e"),

ObjectId("63fb8745d0ac93d762d06f2f")

]

}

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "Yumna Afsar", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "yumna", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2e"), "name" : "yumna", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2f"), "name" : "yumna", "grade" : "B" }

> db.students.updateOne({name:"Yumna Afsar"},{$set{name:"yumna"}})

uncaught exception: SyntaxError: missing : after property id :

@(shell):1:48

> db.students.updateOne({name:"Yumna Afsar"},{$set:{name:"yumna"}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "yumna", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "yumna", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2e"), "name" : "yumna", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2f"), "name" : "yumna", "grade" : "B" }

> db.students.updateMany({name:"yumna"},{set:{name:"Yumna Afsar"}})

uncaught exception: Error: the update operation document must contain atomic operators :

DBCollection.prototype.updateMany@src/mongo/shell/crud\_api.js:655:19

@(shell):1:1

> db.students.updateMany({name:"yumna"},({set:{name:"Yumna Afsar"}})

...

... ^C

> db.students.updateMany({name:"yumna"},{set:{name:"Yumna Afsar"}})

uncaught exception: Error: the update operation document must contain atomic operators :

DBCollection.prototype.updateMany@src/mongo/shell/crud\_api.js:655:19

@(shell):1:1

> db.students.updateMany({name:"yumna"},({set:{name:"Yumna Afsar"}})

... ^C

> db.students.updateMany({name:"yumna"},{$set:{name:"Yumna Afsar"}})

{ "acknowledged" : true, "matchedCount" : 4, "modifiedCount" : 4 }

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2c"), "name" : "Yumna Afsar", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "Yumna Afsar", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2e"), "name" : "Yumna Afsar", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2f"), "name" : "Yumna Afsar", "grade" : "B" }

> db.students.deleteOne({name:"Yumna Afsar"})

{ "acknowledged" : true, "deletedCount" : 1 }

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

{ "\_id" : ObjectId("63fb8187d0ac93d762d06f2d"), "name" : "Yumna Afsar", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2e"), "name" : "Yumna Afsar", "grade" : "B" }

{ "\_id" : ObjectId("63fb8745d0ac93d762d06f2f"), "name" : "Yumna Afsar", "grade" : "B" }

> db.students.deleteMany({name:})

uncaught exception: SyntaxError: expected expression, got '}' :

@(shell):1:29

> db.students.deleteMany({name:"Yumna Afsar"})

{ "acknowledged" : true, "deletedCount" : 3 }

> db.students.find()

{ "\_id" : ObjectId("63fb7241c97bc8853c7fcdac"), "name" : "saeed ali", "grade" : "A" }

{ "\_id" : ObjectId("63fb7deed0ac93d762d06f26"), "name" : "Abrar Ahmed", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2a"), "name" : "mussafara", "grade" : "A" }

{ "\_id" : ObjectId("63fb7f4ed0ac93d762d06f2b"), "name" : "zehra", "grade" : "A" }

> db.students.delete()

uncaught exception: TypeError: db.students.delete is not a function :

@(shell):1:1

> db.students.deleteMany({})

{ "acknowledged" : true, "deletedCount" : 4 }

> db.students.find()

> show collections

personel

student

students

>