
MERN
MEAN
mongoDB

```
shradhakhapra@Shradhas-MacBook-Air ~ % sudo brew services start mongodb-community@7.0
Password:
```

shradhakhapra — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — -zsh — 88×31

```
shradhakhapra@Shradhas-MacBook-Air ~ % sudo brew services start mongodb-community@7.0
Password:
Warning: Taking root:admin ownership of some mongodb-community paths:
  /opt/homebrew/Cellar/mongodb-community/7.0.0/bin
  /opt/homebrew/Cellar/mongodb-community/7.0.0/bin/mongod
  /opt/homebrew/opt/mongodb-community
  /opt/homebrew/opt/mongodb-community/bin
  /opt/homebrew/var/homebrew/linked/mongodb-community
This will require manual removal of these paths using `sudo rm` on
brew upgrade/reinstall/uninstall.
Warning: mongodb-community must be run as non-root to start at user login!
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-community)
shradhakhapra@Shradhas-MacBook-Air ~ %
```

```
shradhakhapra@Shradhas-MacBook-Air ~ % mongosh
Current Mongosh Log ID: 64fec3e65ce4f8cf69f7e375
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelection
TimeoutMS=2000&appName=mongosh+2.0.0
Using MongoDB:          7.0.0
Using Mongosh:          2.0.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-09-11T13:08:02.064+05:30: Access control is not enabled for the database. Read a
nd write access to data and configuration is unrestricted
   2023-09-11T13:08:02.064+05:30: You are running this process as the root user, which i
s not recommended
------

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be l
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test>
```

C create / insert
R Read / find
U Update
D Delete

The Shell

mongosh //to start

use college //to create & use a new database called "college"

```
test> help

  Shell Help:

    use                                          Set current database
    show                                         'show databases'/'show dbs':
 of all available databases.
                                                 'show collections'/'show tab
 list of all collections for current database.
                                                 'show profile': Prints syste
ormation.
                                                 'show users': Print a list o
or current database.
                                                 'show roles': Print a list o
or current database.
                                                 'show log <type>': log for c
tion, if type is not set uses 'global'
                                                 'show logs': Print all logs.

    exit                                         Quit the MongoDB shell with
exit
    quit                                         Quit the MongoDB shell with
    Mongo                                        Create a new connection and
ngo object. Usage: new Mongo(URI, options [optional])
    connect                                      Create a new connection an
tabase object. Usage: connect(URI, username [optional], password [optiona
    it                                           result of the last line ev
 further iterate
```

```
For more informati
test> show dbs
admin    40.00 KiB
config   96.00 KiB
local    72.00 KiB
test>
```

shradhakhapra — mongosh mongodb://127.0.0.1:27

```
test> 1+2
3
test> "apnacollege".toUpperCase()
APNACOLLEGE
test
```

temp database

↓

test

```
test> show dbs
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
test> use college
switched to db college
college> s
```

```
local    72.00 Ki
college> db
college
college>
```

BSON Data

Binary JSON

BSON Data

{ key : val }

json

Binary JSON

JSON

1) text based

2) space inefficient

BSON

JSON And BSON | MongoDB

mongodb.com/json-and-bson

Incognito

Event Join us at MongoDB.local Delhi/NCR. Use invite code "MDBPENCIL23" >

MongoDB. Products Solutions Resources Company Pricing Sign In

JSON and BSON

Try MongoDB Atlas Free

JSON vs BSON

	JSON	BSON
Encoding	UTF-8 String	Binary
Data Support	String, Boolean, Number, Array, Object, null	String, Boolean, Number (Integer, Float, Long, Decimal128...), Array, null, Date, BinData
Readability	Human and Machine	Machine Only

Collections

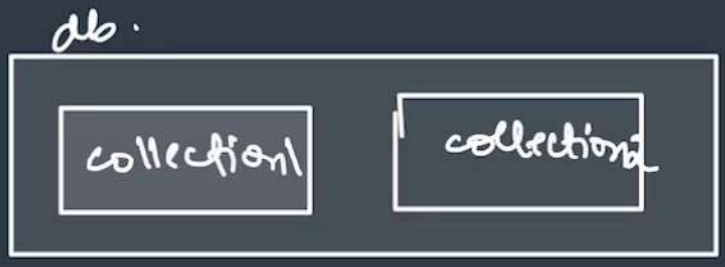
Document : Mongo stores data in form of documents (BSON docs)

Collection : MongoDB stores documents in collections.

```
{
  name: "al",
  age: 18,
  status: "D",
  groups: [ "politics", "news" ]
}
```

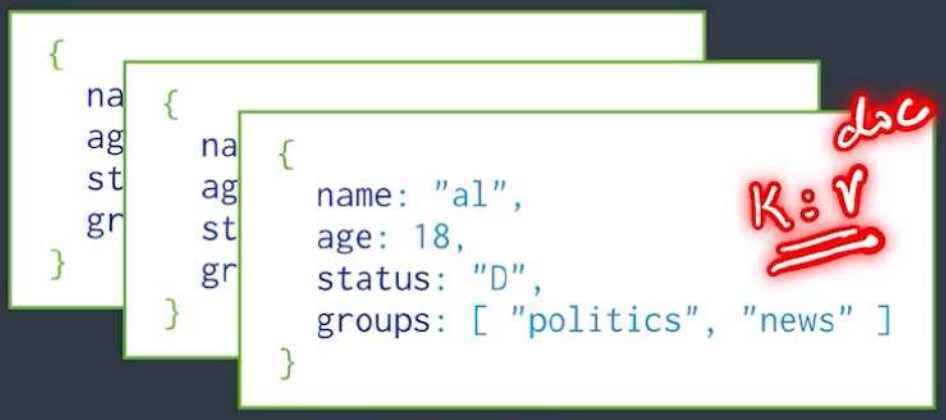
Collection

Collections



Document : Mongo stores data in form of documents (BSON docs)

Collection : MongoDB stores documents in collections.



Collection

INSERT in DB

insertOne

`db.collection.insertOne()`	Inserts a single document into a collection.

show collections

db.student.insertOne({ name: “adam”, marks: 79 })

db.student.find( )

If a collection does not exist, MongoDB creates the collection when you first store data for that collection.

Insert Methods

MongoDB provides the following methods for inserting documents into a collection:

`db.collection.insertOne()`	**Inserts a single document into a** collection.
`db.collection.insertMany()`	Inserts multiple documents into a collection.

student

{

'js objects

}

document

```
test> db
test
test> use college
switched to db college
college> show collections

college> db.student.insertOne( {name: "adam", age: 19, marks: 88} )
{
  acknowledged: true,
  insertedId: ObjectId("64fec88c4d37352da5df8407")
}
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
  }
]
college>
```

INSERT in DB

insertMany (array of documents)

```
db.student.insertMany( [ { name: "bob", marks: 65 }, { name: "eve", city: "Delhi" } ] )
```


`db.collection.insertMany()`	Inserts multiple documents into a collection.

```
college> db.student.insertMany([ {name: "catlyn", marks: 64, city: "Delhi"}, {name: "don
ald", marks: 58, city: "Mumbai"} ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64fec9f54d37352da5df8409"),
    '1': ObjectId("64fec9f54d37352da5df840a")
  }
}
college>
```

```
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'Delhi'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df840a"),
    name: 'donald',
    marks: 58,
    city: 'Mumbai'
  }
]
```

FIND in DB

db.collection.find( ) //returns everything

for specific queries

db.collection.find({ key: value })

db.collection.findOne({ key: value })

```
college> db.student.find( {city: "Delhi"} )
[
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'Delhi'
  }
]
college>
```

```
college> db.student.findOne( {city: "Delhi"} )
{
  _id: ObjectId("64fec9714d37352da5df8408"),
  name: 'bob',
  city: 'Delhi',
  marks: 75
}
college>
```

```
college> db.student.find( {name: "adam"} )
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
```

```
college> db.student.findOne( {name: "adam"} )
{
  _id: ObjectId("64fec88c4d37352da5df8407"),
  name: 'adam',
  age: 19,
  marks: 88
}
college> db.student.find( {name: "adam"} )
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
```

FIND in DB

db.collection.find( ) //returns everything

for specific queries

db.collection.find({ key: value }) → cursor (reference to original)

db.collection.findOne({ key: value }) → actual document

```
college> db.student.find({city: "Delhi", marks: 75})
[
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  }
]
college>
```

FIND in DB

Query Operators

Q. Find students where marks > 75

```
db.student.find( {marks: {$gt: 75}} )
```

Q. Find students who live in Delhi or Mumbai

```
db.student.find( {city: {$in: ["Delhi", "Mumbai"]}} )
```

Q. Find students who scored > 75 or live in Delhi

```
db.student.find( {$or: [ {marks: {$gt: 75}}, {city: "Delhi"}]} )
```

Name	Description
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.

Logical

Name	Description
`$and`	Joins query clauses with a logical `AND` returns all documents that match the conditions of both clauses.
`$not`	Inverts the effect of a query expression and returns documents that do *not* matc query expression.
`$nor`	Joins query clauses with a logical `NOR` returns all documents that fail to match b clauses.
`$or`	Joins query clauses with a logical `OR` returns all documents that match the co of either clause.

Geospatial

Name	Description
`$geoIntersects`	Selects geometries that intersect with a GeoJSON geometr supports `$geoIntersects`.
`$geoWithin`	Selects geometries within a bounding GeoJSON geometry. indexes support `$geoWithin`.
`$near`	Returns geospatial objects in proximity to a point. Requires 2dsphere and 2d indexes support `$near`.
`$nearSphere`	Returns geospatial objects in proximity to a point on a sphe index. The 2dsphere and 2d indexes support `$nearSphere`.

```
college> db.student.find( {marks: {$gt: 75}} )
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
  }
]
```

```
college> db.student.find( {marks: {$gt: 60}} )
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'Delhi'
  }
```

```
college> db.student.find( {marks: {$gte: 75}} )
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  }
```

```
college> db.student.find({ city: {$in: ["Delhi", "Mumbai"]} })
[
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'Delhi'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df840a"),
    name: 'donald',
    marks: 58,
    city: 'Mumbai'
  }
]
college>
```

```
college> db.student.find({ city: {$in: ["delhi", "mumbai"]} })
```

```
college> db.stduent.find( {$or: [{marks: {$gt: 75}}, {city: "Delhi"}] } )

college> db.student.find( {$or: [{marks: {$gt: 75}}, {city: "Delhi"}] } )
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 88
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'Delhi'
  }
]
college>
```

Update Methods

MongoDB provides the following methods for updating documents in a collection:

`db.collection.updateOne()`	Updates at most a single document that match a specified filter even though multiple documents may match the specified filter.
`db.collection.updateMany()`	Update all documents that match a specified filter.
`db.collection.replaceOne()`	Replaces at most a single document that match a specified filter even though multiple documents may match the specified filter.

UPDATE in DB

updateOne

`db.collection.updateOne()`	Updates at most a single document that match a specified filter even though multiple documents may match the specified filter.

db.collection.updateOne(<filter> , <update>, <options>)

```
db.student.updateOne( {name:"adam"}, {$set: {marks:99}} )
```

UPDATE in DB

Update Operators

- $addFields
- $set
- $project
- $unset
- $replaceRoot
- $replaceWith

`$addFields`

Adds new fields to documents. `$addFields` outputs documents that contain all existing fields from the input documents and newly added fields.

The `$addFields` stage is equivalent to a `$project` stage that explicitly specifies all existing fields in the input documents and adds the new fields.

`$set`

New in version 4.2.

Adds new fields to documents. `$set` outputs documents that contain all existing fields from the input documents and newly added fields.

The `$set` stage is an alias for `$addFields`.

Both stages are equivalent to a `$project` stage that explicitly specifies all existing fields in the input documents and adds the new fields.

```
college> db.student.updateOne( {name:"adam"}, {$set: {marks: 99}} )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
college>
```

```
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 99
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'Delhi'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df840a"),
    name: 'donald',
    marks: 58,
    city: 'Mumbai'
  }
```

UPDATE in DB

`db.collection.updateMany()`	Update all documents that match a specified filter.
`db.collection.replaceOne()`	Replaces at most a single document that match a specified filter even though multiple documents may match the specified filter.

```
college> db.student.updateMany( {city: "Delhi"}, {$set: {city: "New Delhi"}} )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

```
}
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 99
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'bob',
    city: 'New Delhi',
    marks: 75
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'New Delhi'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df840a"),
    name: 'donald',
    marks: 58,
    city: 'Mumbai'
  }
]
college>
```

```
college> db.student.replaceOne( {name: "bob"}, {name: "shradha", marks: 94, state: "Haryana"}  )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
college>
```

```
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 99
  },
  {
    _id: ObjectId("64fec9714d37352da5df8408"),
    name: 'shradha',
    marks: 94,
    state: 'Haryana'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'New Delhi'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df840a"),
    name: 'donald',
    marks: 58,
    city: 'Mumbai'
  }
]
```

```
college> db.student.insertOne( {name: "farah", performance: {marks: 88, grade: "A"}} )
{
  acknowledged: true,
  insertedId: ObjectId("64fed5714d37352da5df840b")
}
```

Nesting

```
{
  _id: ObjectId("64fdfcad780181c3d03ec623"),
  name: 'farah',
  performance: { marks: 88, grade: 'A' }
}
```

to find

```
db.student.findOne( {"performance.marks": 88} )
```

```
{
  _id: ObjectId("64fec88c4d37352da5df8407"),
  name: 'adam',
  age: 19,
  marks: 99
},
{
  _id: ObjectId("64fec9714d37352da5df8408"),
  name: 'shradha',
  marks: 94,
  state: 'Haryana'
},
{
  _id: ObjectId("64fec9f54d37352da5df8409"),
  name: 'catlyn',
  marks: 64,
  city: 'New Delhi'
},
{
  _id: ObjectId("64fec9f54d37352da5df840a"),
  name: 'donald',
  marks: 58,
  city: 'Mumbai'
},
{
  _id: ObjectId("64fed5714d37352da5df840b"),
  name: 'farah',
  performance: { marks: 88, grade: 'A' }
}
```

```
college> db.student.find( {marks: 88} )

college>
```

```
college> db.student.find( {marks: 88} )

college> db.student.find( {"performance.marks": 88} )
[
  {
    _id: ObjectId("64fed5714d37352da5df840b"),
    name: 'farah',
    performance: { marks: 88, grade: 'A' }
  }
]
college>
```

```
college> db.student.deleteOne( {state: "Haryan"} )
{ acknowledged: true, deletedCount: 0 }
college>
```

```
college> db.student.deleteOne( {state: "Haryan"} )
{ acknowledged: true, deletedCount: 0 }
college> db.student.deleteOne( {state: "Haryana"} )
{ acknowledged: true, deletedCount: 1 }
college> db.student.find()
```

DELETE in DB

deleteOne

```
db.collection.deleteOne( <filter> , <options> )
```

deleteMany

```
db.collection.deleteMany( <filter> , <options> )
```

```
db.dropDatabase( )
```

```
{ acknowledged: true, deletedCount: 1 }
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 99
  },
  {
    _id: ObjectId("64fec9f54d37352da5df8409"),
    name: 'catlyn',
    marks: 64,
    city: 'New Delhi'
  },
  {
    _id: ObjectId("64fec9f54d37352da5df840a"),
    name: 'donald',
    marks: 58,
    city: 'Mumbai'
  },
  {
    _id: ObjectId("64fed5714d37352da5df840b"),
    name: 'farah',
    performance: { marks: 88, grade: 'A' }
  }
]
```

```
college> db.student.deleteMany( {marks: {$lt: 75}} )
{ acknowledged: true, deletedCount: 2 }
college> db.student.find()
```

```
college> db.student.deleteMany( {marks: {$lt: 75}} )
{ acknowledged: true, deletedCount: 2 }
college> db.student.find()
[
  {
    _id: ObjectId("64fec88c4d37352da5df8407"),
    name: 'adam',
    age: 19,
    marks: 99
  },
  {
    _id: ObjectId("64fed5714d37352da5df840b"),
    name: 'farah',
    performance: { marks: 88, grade: 'A' }
  }
]
college>
```

```
  }
]
college> db.student.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
college> db.student.deleteMany({})
```

```
college> db.student.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
college> db.student.find()

college>
```

```
college> db.student.find()

college> db.dropDatabase()
{ ok: 1, dropped: 'college' }
college>
```

```
college> db.student.insertOne( {name: "farah", performance: {marks: 88, gra
{
  acknowledged: true,
  insertedId: ObjectId("64fed6c34d37352da5df840c")
}
college> show dbs
admin     40.00 KiB
college    8.00 KiB
config   108.00 KiB
local     72.00 KiB
college> db.dropDatabase()
{ ok: 1, dropped: 'college' }
college> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
college>
```