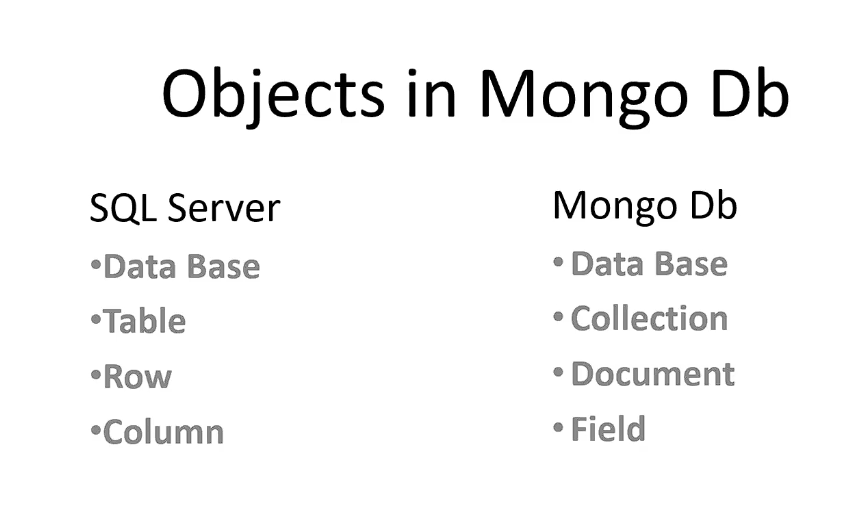
**Mongo DB**

Mongo db is a NOSQLcross-platform, open-source, document oriented database. Which means Not Only sql.

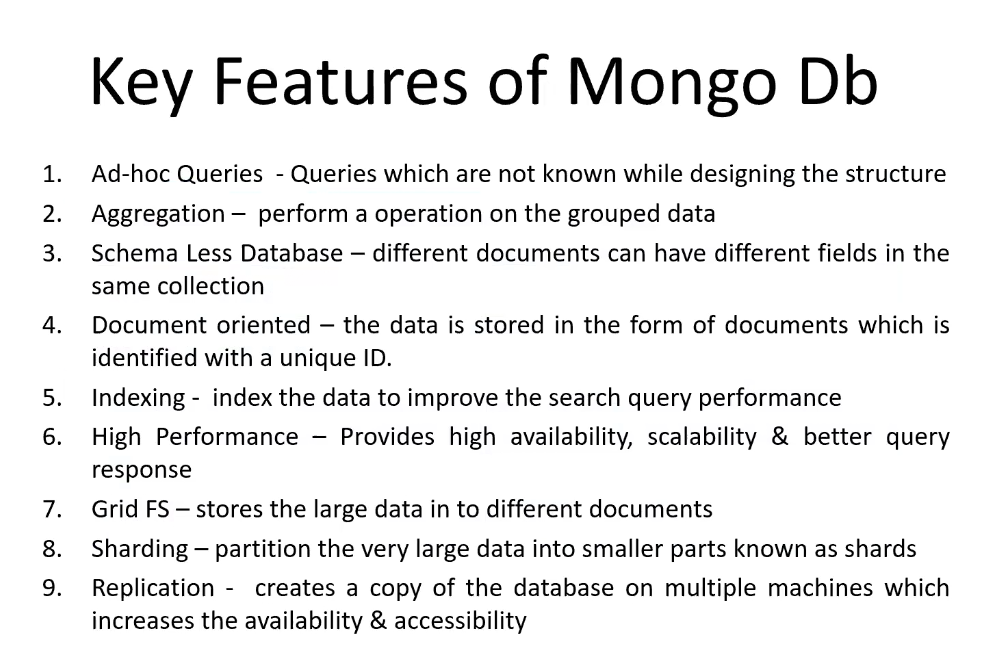
The main purpose of mongo db is to store the data. It is the container for the data which is coming from any of our applications.

It doesn’t use tables and rows to store its data, but instead collections of documents.

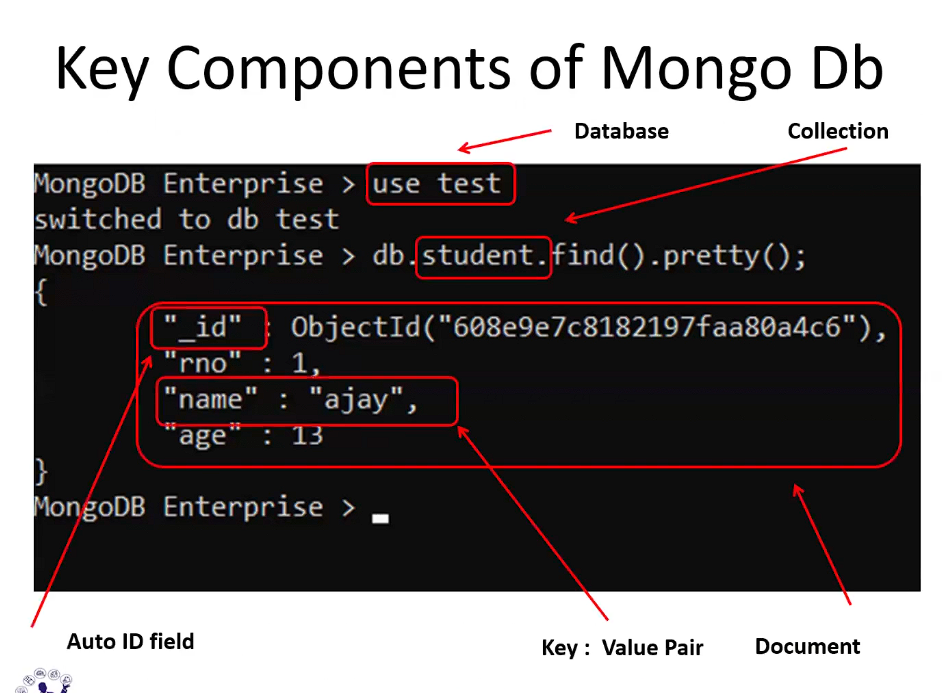
Mongo db is a schema-less database (no particular structure), so we don’t need to specify the number or type of columns before inserting our data.



\*\*Mongo db automatically maintains multiple copies of the data. It is real-time.\*\*

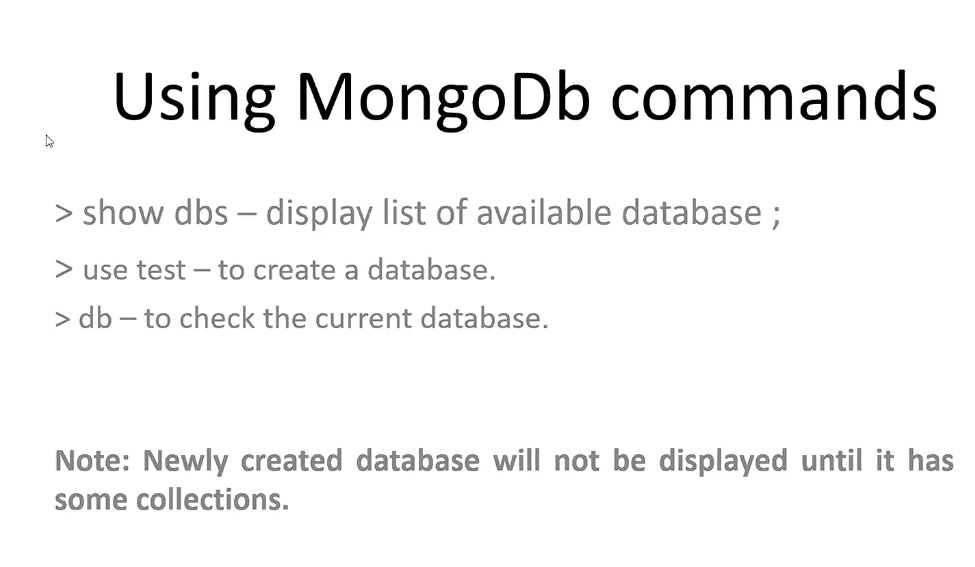


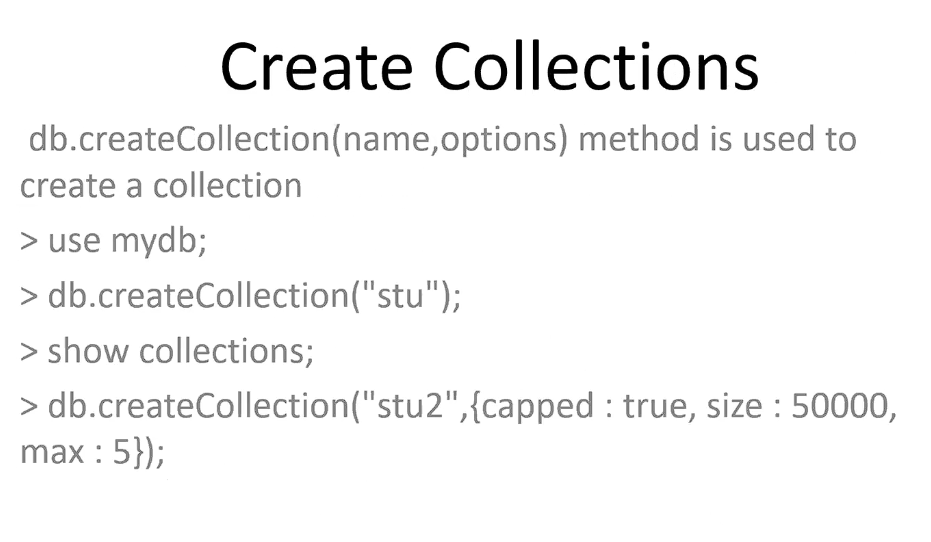






Data is stored in BSON format in Mongo db. Binary form of Json is known as BSON.

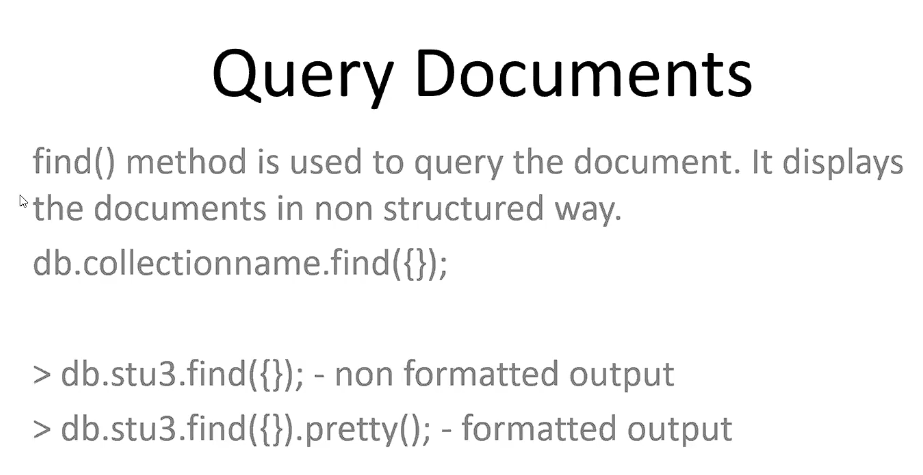




\*\*Data Insertion in Mongo db works on the principle of LIFO.

db.dropDatabase() deletes the database in which we are currently in.

The output we get after using db.stu.find().pretty() is called a Formatted Output.



\*\*In 1 page, only 20 documents can be displayed by Mongo DB.\*\*



***Interview Question:***

Difference between updateOne() and replaceOne() method.

\*\*In case of the replaceOne() method, the entire document will be replaced. But in the case of updateOne(), only the specified fields will be updated.\*\*

db.stu.findOne() – Fetch the first record in the formatted manner.

db.stu.deleteOne() – the first document with the matching criteria will be deleted.

db.stu.deleteMany() – All the documents with the matching criteria will be deleted.

db.stu.save () – another way of inserting the record. It works like insert().

\*\* It is a combination of insert() and replace(). \*\*

**Logical Operators in Mongo DB:-**

Logical operators are used to combine the two operations in mongo db.

‘AND’ is the default logical operator.

For displaying the records with either of the condition to be true, we use OR.

The syntax for NOT operator is slightly different,

db.emp.find({'dept' : {$not : {$eq : 'OPS'}}})

$eq - is the comparision operator.

NOR operator – Neither the first condition should be True, Nor the second.

db.emp.find({$nor : [{‘dept’ : ‘admin’}, {‘desi’ : ‘manager’}] })

it is a combination of NOT and OR.

**Comparision Operators in Mongo DB:-**

EQ – db.emp.find({‘dept’ : {$eq : ‘temp’}})

It is called Equality Operator.

EQ is the default comparison operator.

NE – not equal to operator.

d.emp.find({‘dept’ : {$ne : ‘HR’}})

$lt – less than operator

$lte – Less than equal to

$gt – Greater than Operator

$gte – Greater than equal to operator

Q. Display all the documents where dept = mis, it, admin

1st option –

db.emp.find({$or : [{‘dept’ : ‘mis’}, {‘dept’ : ‘it’}, {‘dept’ : ‘admin’}] })

2nd option –

db.emp.find({‘dept’ : {$in : [‘mis’, ‘it’, ‘admin’]} })

IN is used for list.

$nin – Not in comparision operator

db.emp.find({‘dept’ : {$nin : [‘mis’, ‘it’, ‘admin’]} })

**Importing Data:-**

Command to import data in Mongo DB(should be executed in command prompt)

mongoimport -d tmp(name of the database) -c sal(name of the collection) --type csv(type of the file which we are importing) --file EMP\_SAL.csv(name of the file) –headerline(whatever is in the first line should be treated as the name of the fields or heading).

If the database is present it will be used. If it is not present, a new database will be created.

If the collection is there, it will be used. Otherwise a new collection will be created.

If we execute this command again, since both the database and the collection are present, the data will be appended to the added data. To avoid this, we use the command,

Mongoimport – d tmp -c sal --type csv --file EMP\_SAL.csv --headerline --drop

This will drop the existing collection and the new collection is created and the data is added.

Only 3 kinds of files can be imported in Mongo DB. Csv, tsv and json.

db.emp.remove({‘dept’ : ‘HR’}, JustOne) – By Default the value for just one is False or 0. But if we specify any other value, it will delete the first document matching the criteria.

db.emp.remove({'dept' : 'HR'}) – It works like deleteMany().

db.emp.remove({‘dept’ : ‘HR’},1) – It works like deleteOne().

db.emp.remove({‘dept’ : ‘HR’},9) – It works like deleteOne().

\*\* It is a combination of deleteOne() and deleteMany(). \*\*

db.emp.remove({}) – it will remove all the documents from the collection.

**Projections:-**

db.emp.find({}, {‘eid’ : 1, ‘salary’ : 1}) – From the collection, it shows only the eid and salary for all the documents.

It can be read as db.emp.find({condition}, {projection})

1 = show

**Limit:-**

db.emp.find({}, {‘desi’ : 0}) – from the collection display all the documents but not their ‘desi’ field.

0 = Not show

db.emp.find().limit(10) – Limiting the number of documents to be displayed. Here just the 10 documents will be displayed instead of 20.

**Skipping:-**

db.emp.find().skip(10) – it will skip the first 10 documents and display the rest.

Using limit and skip both:

db.emp.find().limit(10).skip(5) – display first 10 documents after skipping 5.

**Sorting Documents:-**

db.emp.find().sort({‘salary’ : 1}) – the document will be arranged in the ascending order of the salary.

1 = ascending

db.emp.find().sort({‘salary’ : -1}) – the document will be arranged in the descending order of the salary.

-1 = descending

Q. list of HR team members arranged in the descending order of the salary

db.emp.find({‘dept’ : ‘HR’}).sort({‘salary’ : -1})

**Indexes**:-

db.emp.createIndex({‘city’ : 1}) – it is also called a single field index.

here, 1 = ascending

-1 = descending

db.emp.getIndexes() – to get the list of all indexes.

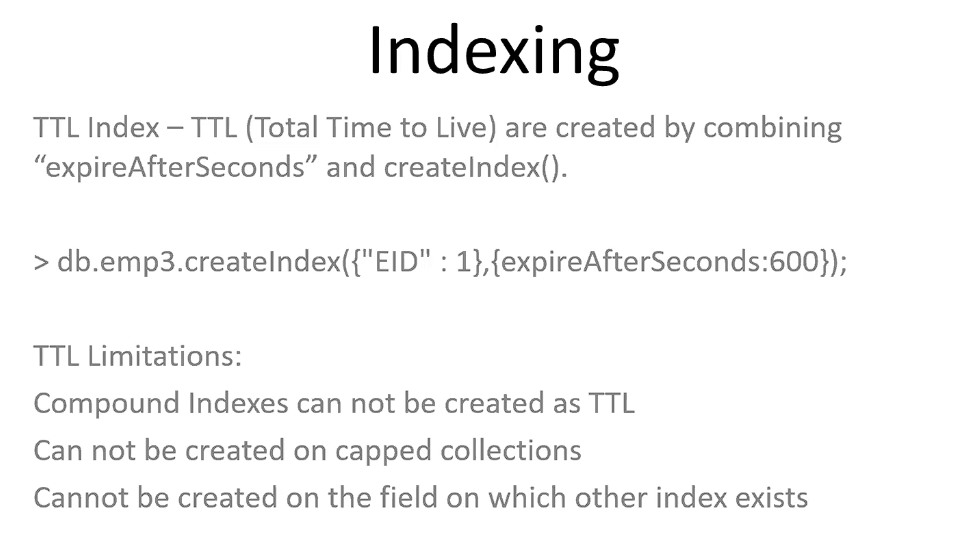
Indexes are the list which makes the data retrieval or search operations faster.

We always have one index already present in our collection by default, as we cannot do anything with that index.

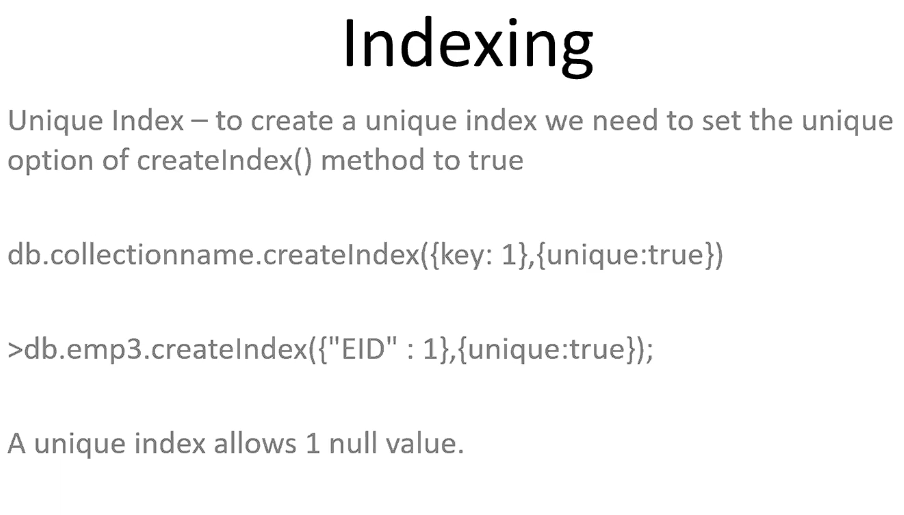
\*\* the index will only be used by Mongo, if we are using the field as a condition on which we have applied index on. \*\*

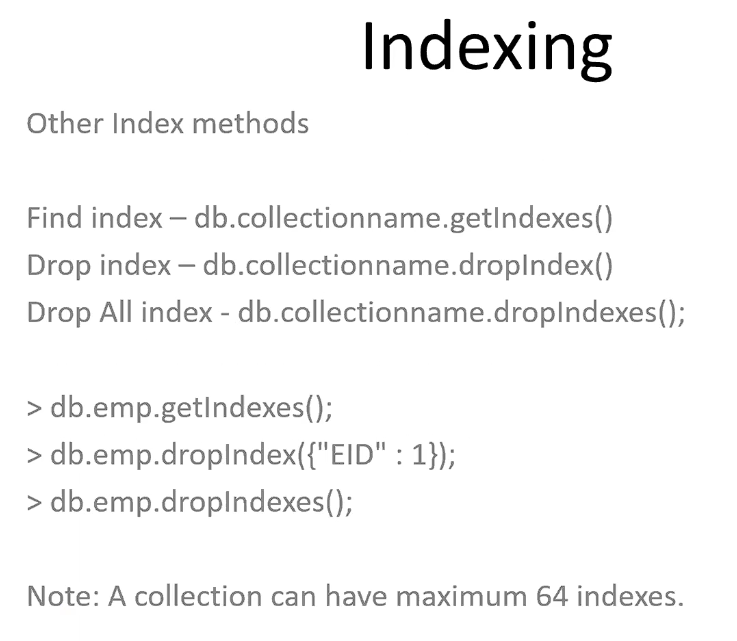
db.emp.createIndex({‘city’ : 1, ‘addr2’ : -1}) – compound index – index applied on 2 fields.

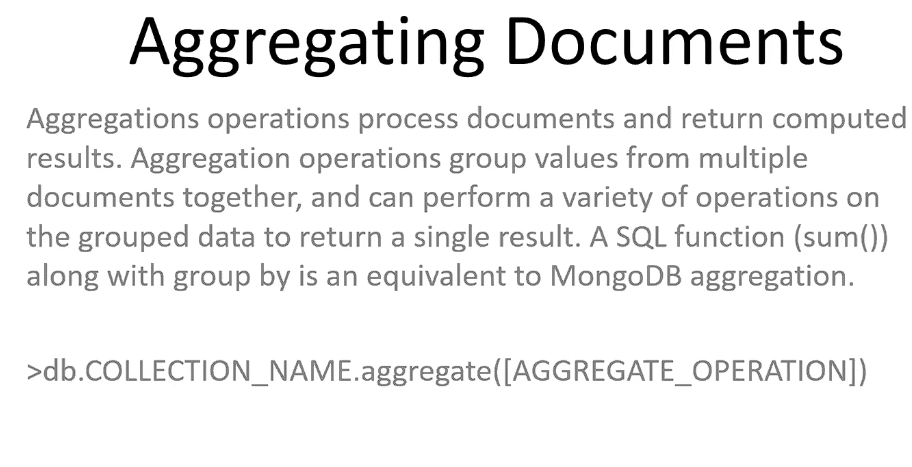
TTL Index – (time to live) :-

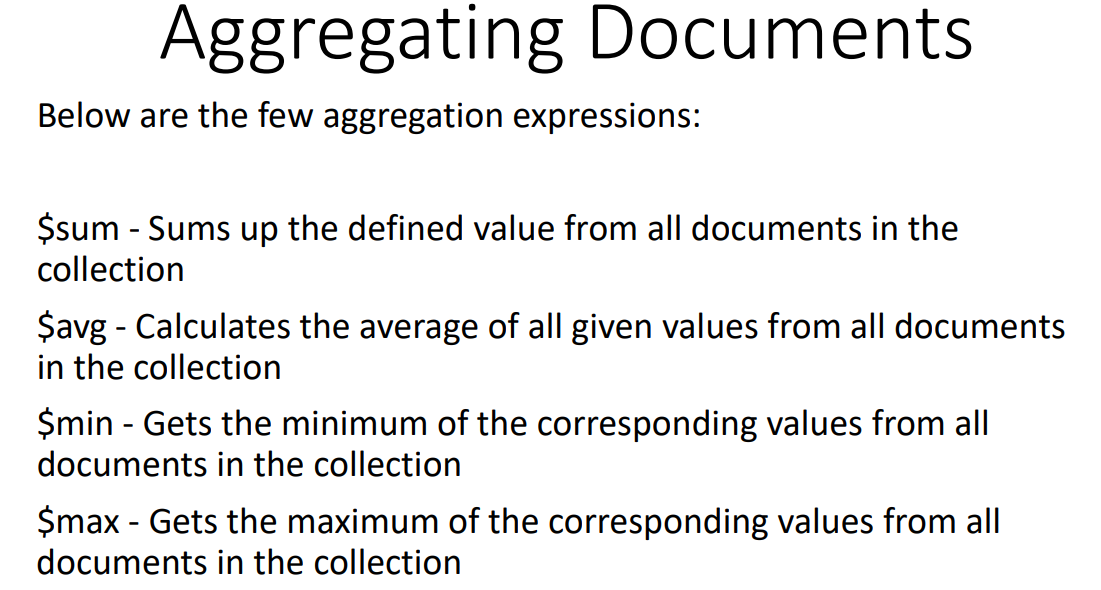


Unique Index – Can be used to enable a unique field.

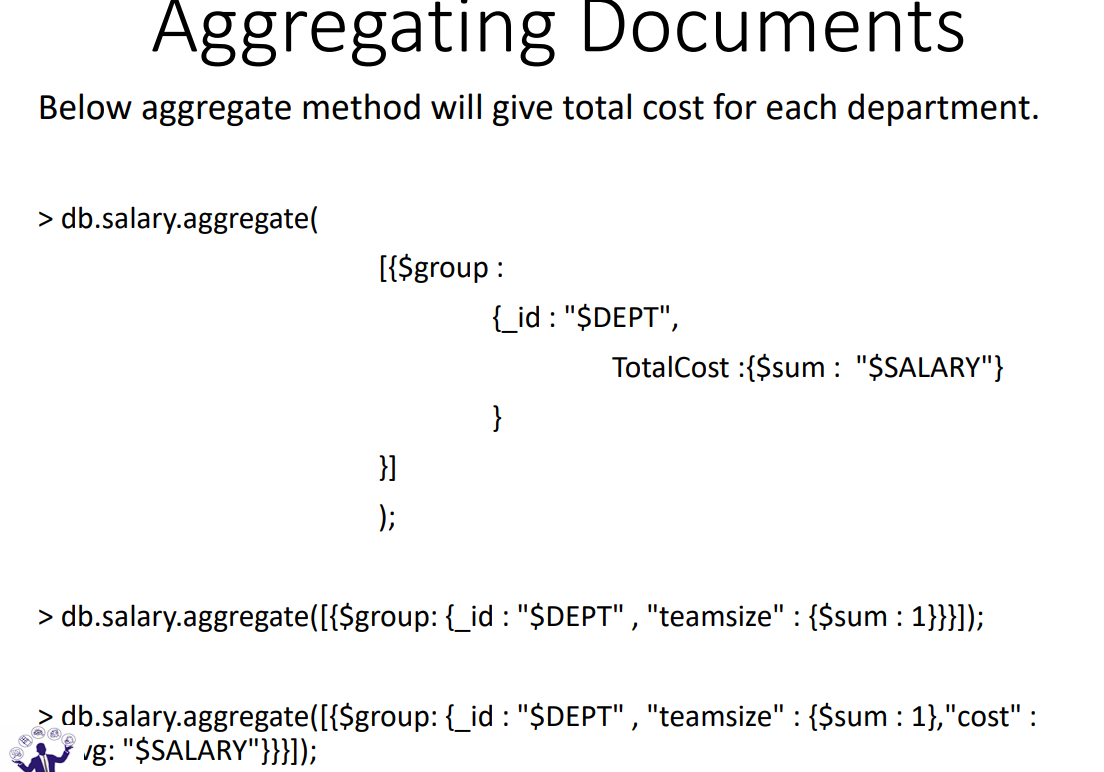








**Group Aggregation:**



**Atomic Opeartions:-**

Doing Multiple operations together is referred to as Atomic Opeartions.

Embedded Documents:

A document within a document is referred to as Embedded Document.

db.order.insert({'\_id' : 'doc1', 'PID' : 'P001', 'PDESC' : 'DELL\_Mouse', 'OSTK' : 10, 'BSTK':8, 'PBY' : [{'CID' : 'C001', 'NAME' : 'AJAY', 'PDATE' : '05-SEPT-2024', }, {'CID' : 'C002', 'NAME' : 'RAKESH', 'PDATE' : '25-DEC-2024'}]})

db.order.find().pretty()

{

"\_id" : "doc1",

"PID" : "P001",

"PDESC" : "DELL\_Mouse",

"OSTK" : 10,

"BSTK" : 8,

"PBY" : [

{

"CID" : "C001",

"NAME" : "AJAY",

"PDATE" : "05-SEPT-2024"

},

{

"CID" : "C002",

"NAME" : "RAKESH",

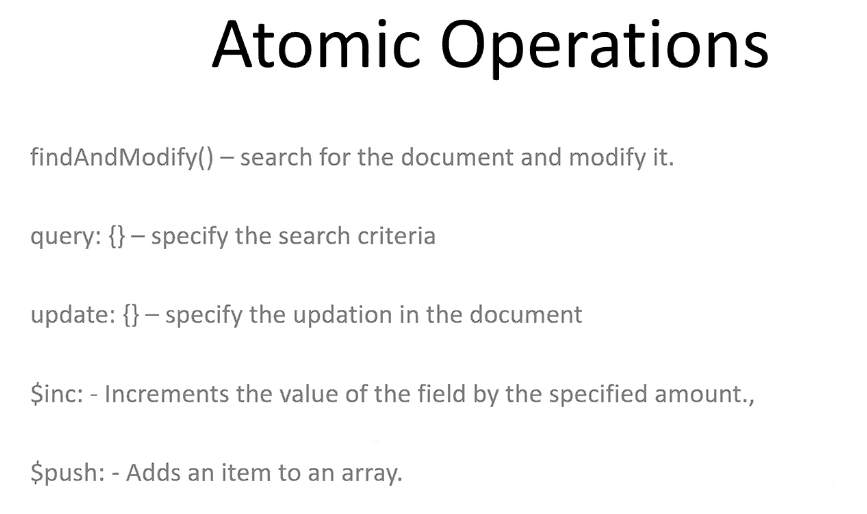
"PDATE" : "25-DEC-2024"

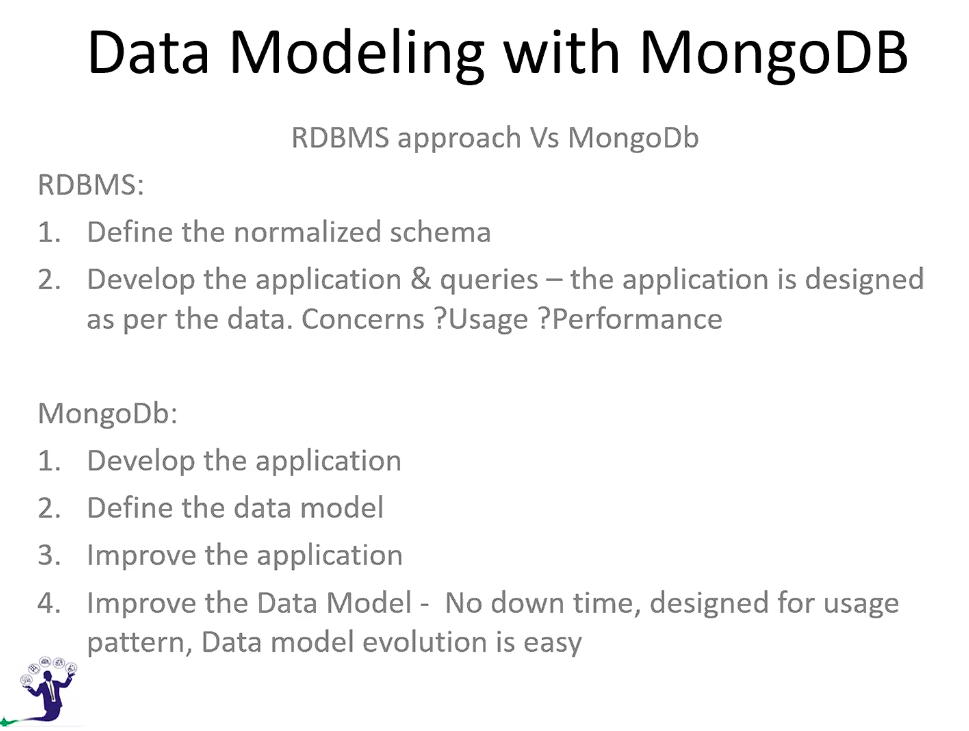
}

]

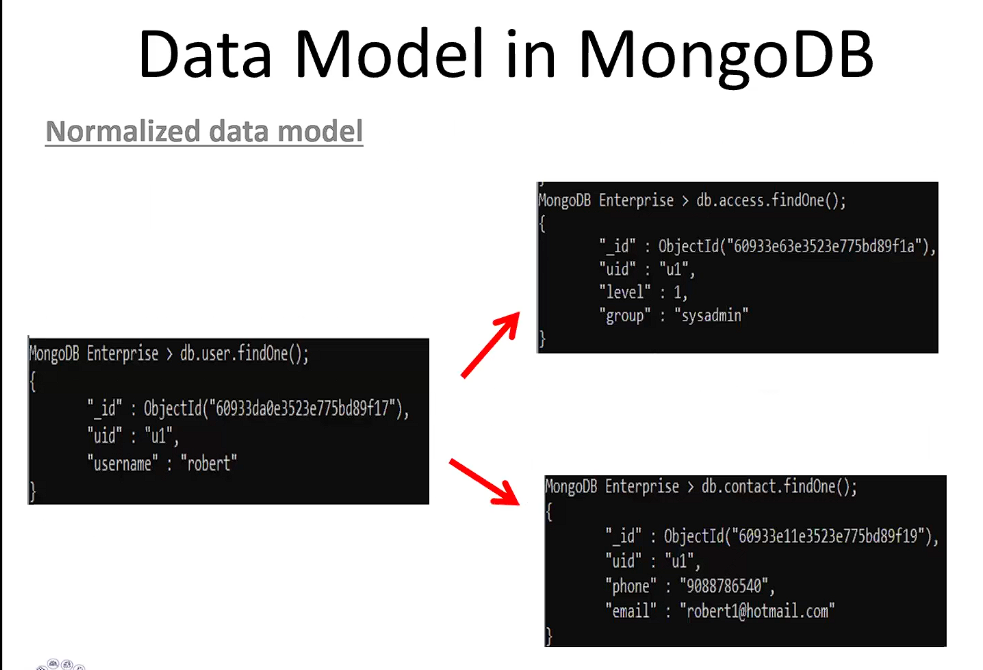
}

db.order.findAndModify({query : {'\_id' : 'doc1', 'BSTK' : {$gt : 0}}, update: {$inc : {'BSTK':-1}, $push: {'PBY': {'CID' : 'C003', 'NAME' : 'RAM', 'PDATE' : '07-SEPT-2024'}}}})

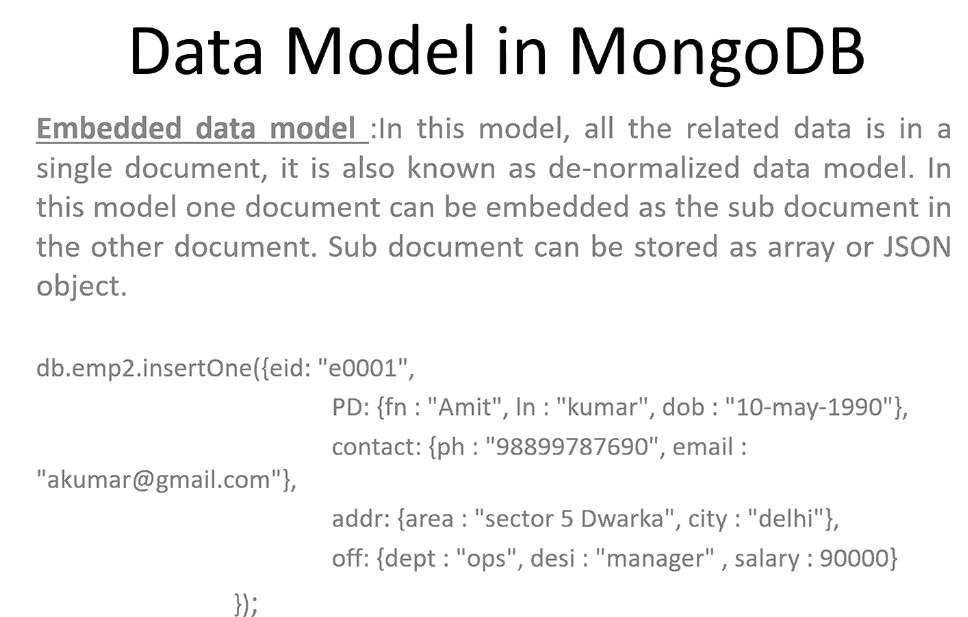




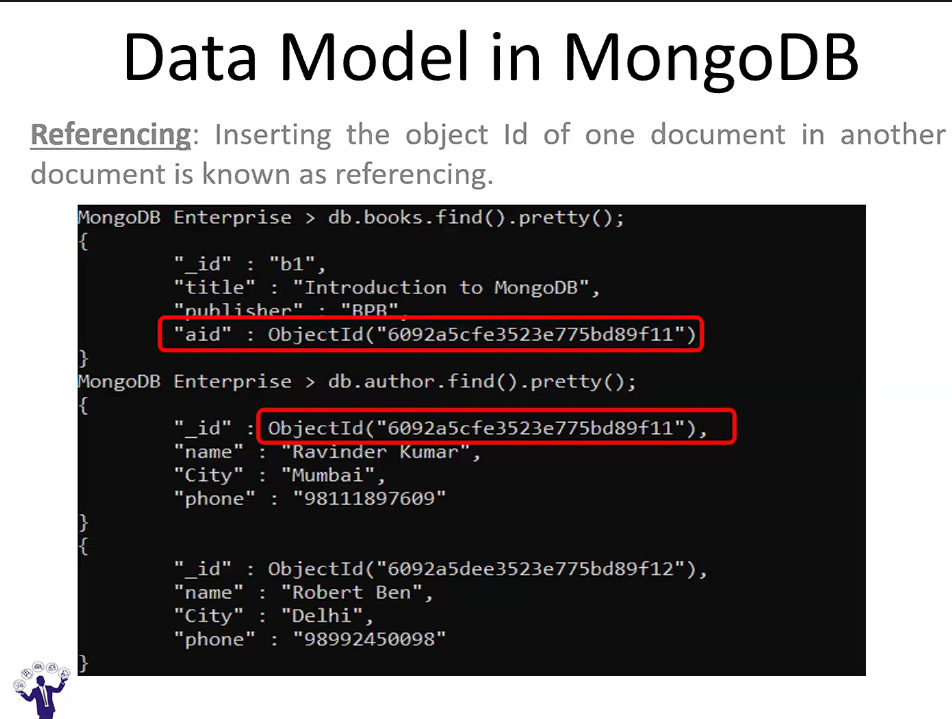




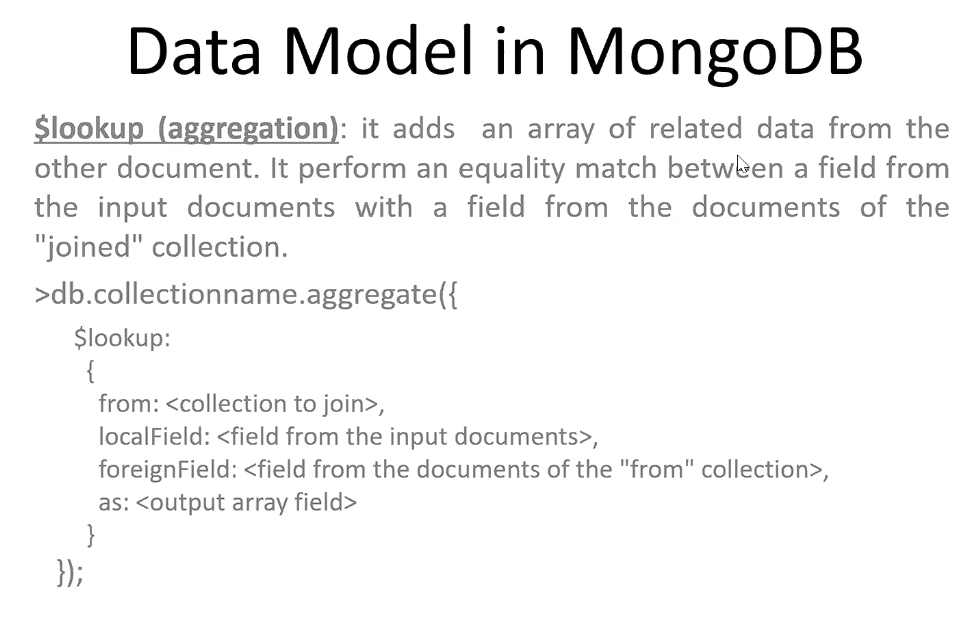
Normalized Data model says that we should not put all our data into a single document, rather put it into multiple documents.



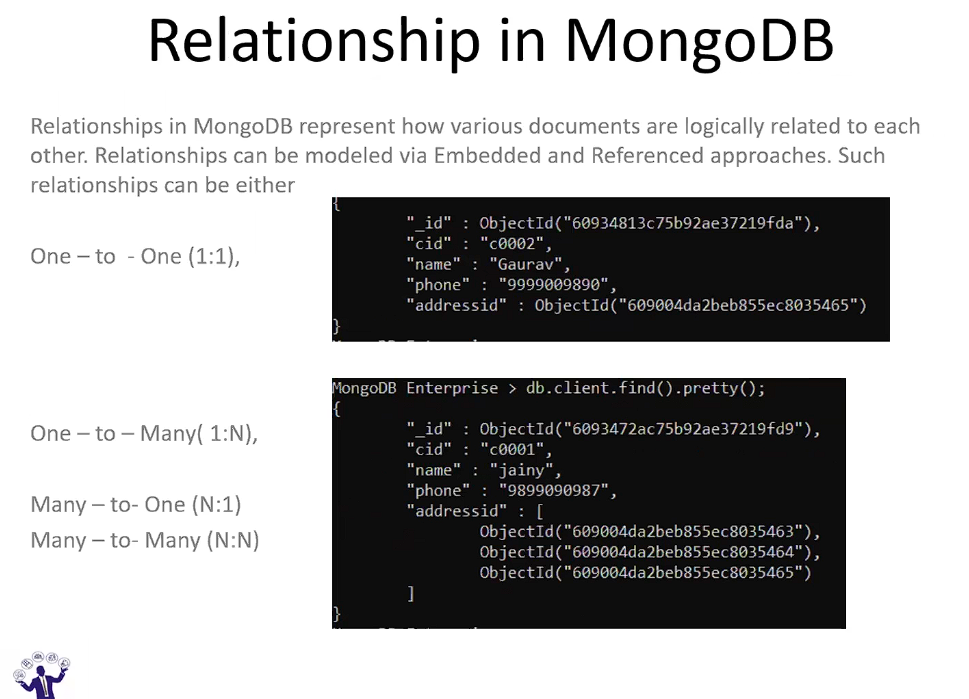
Embedded data model is just the reverse as it focuses on putting all the data into a single document. It is also called DeNormalized data model.



**Lookup Aggregation:-**



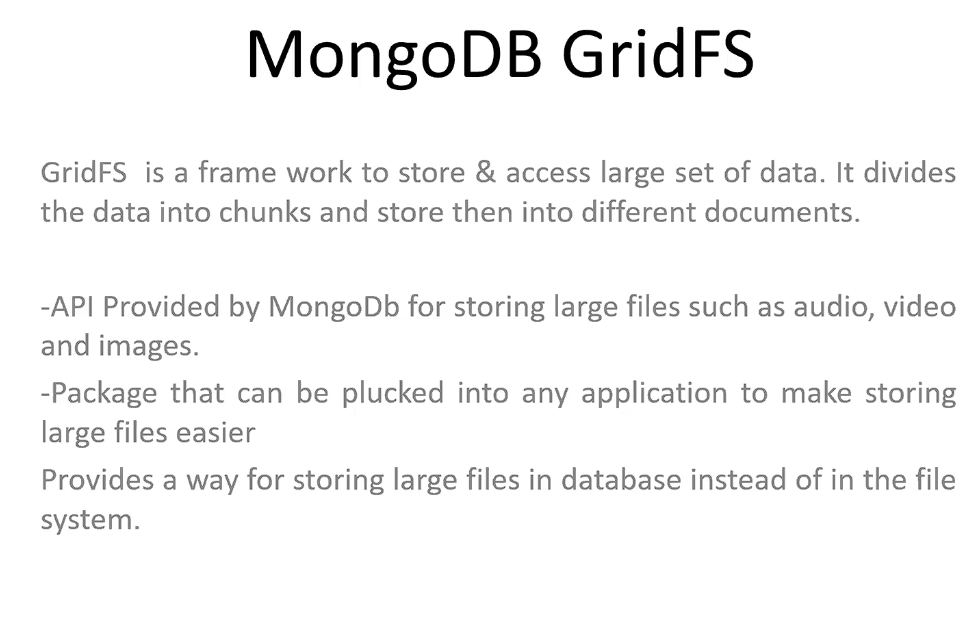


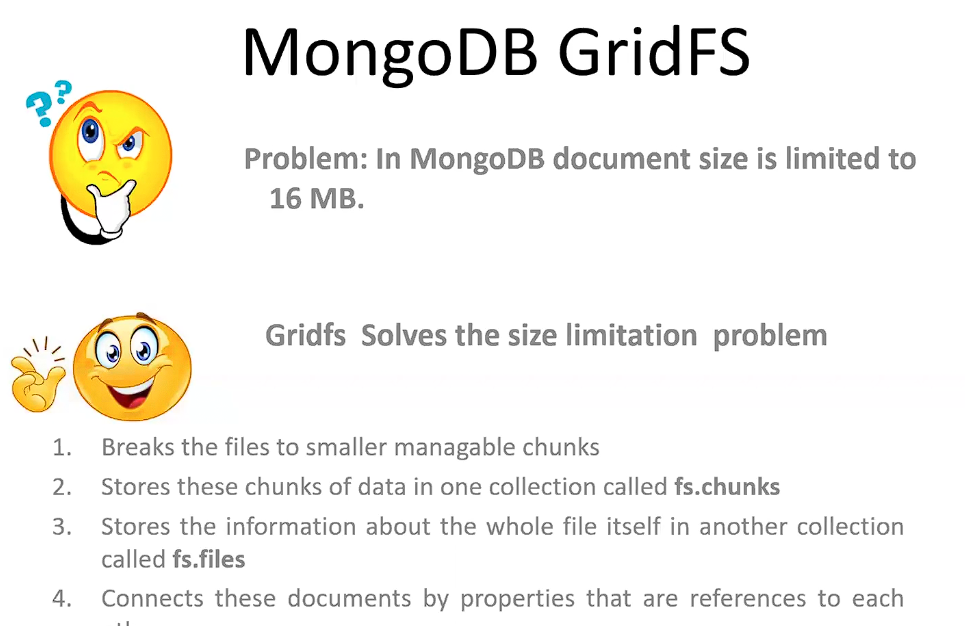


Many -to-One: Many main documents having same sub-documents.

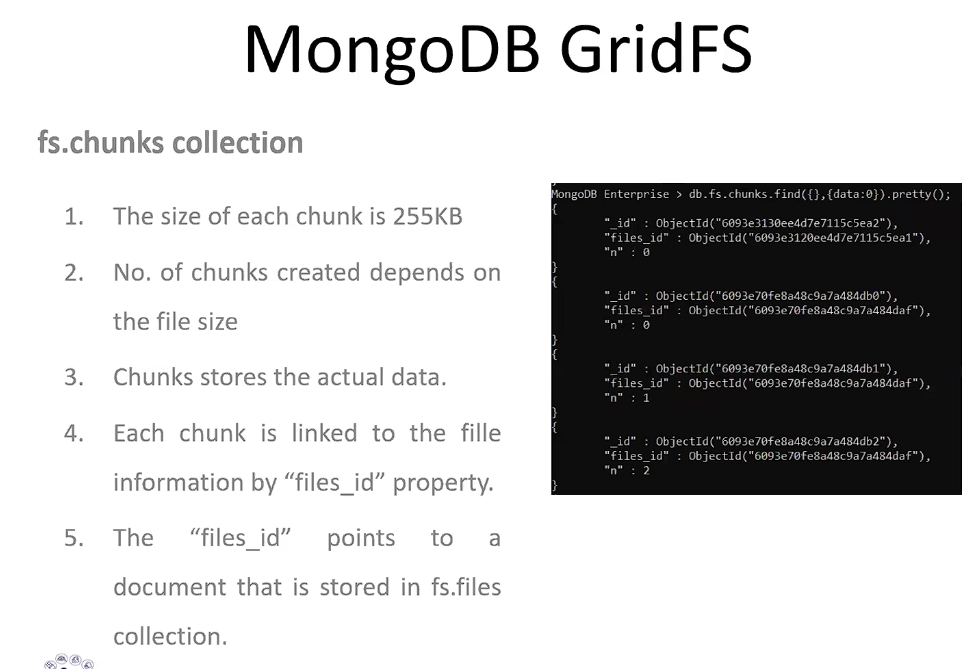
**Data types in Mongo DB:-** Types of data that can be stored in Mongo db.

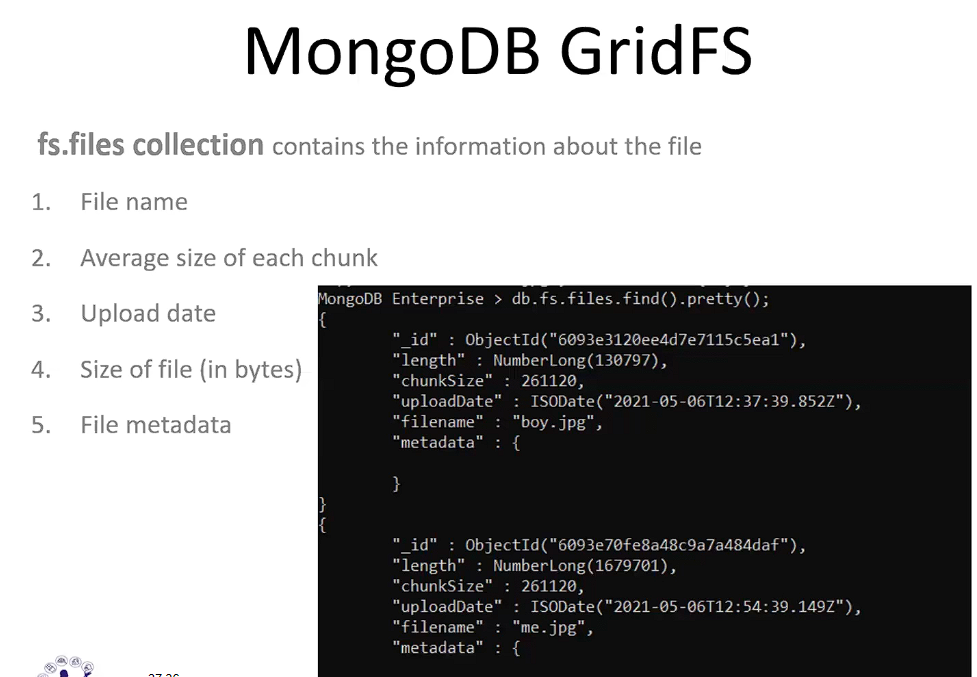
**MongoDB GridFS:**





\*\*Bigger the file size, more the chunks and vice versa.\*\*



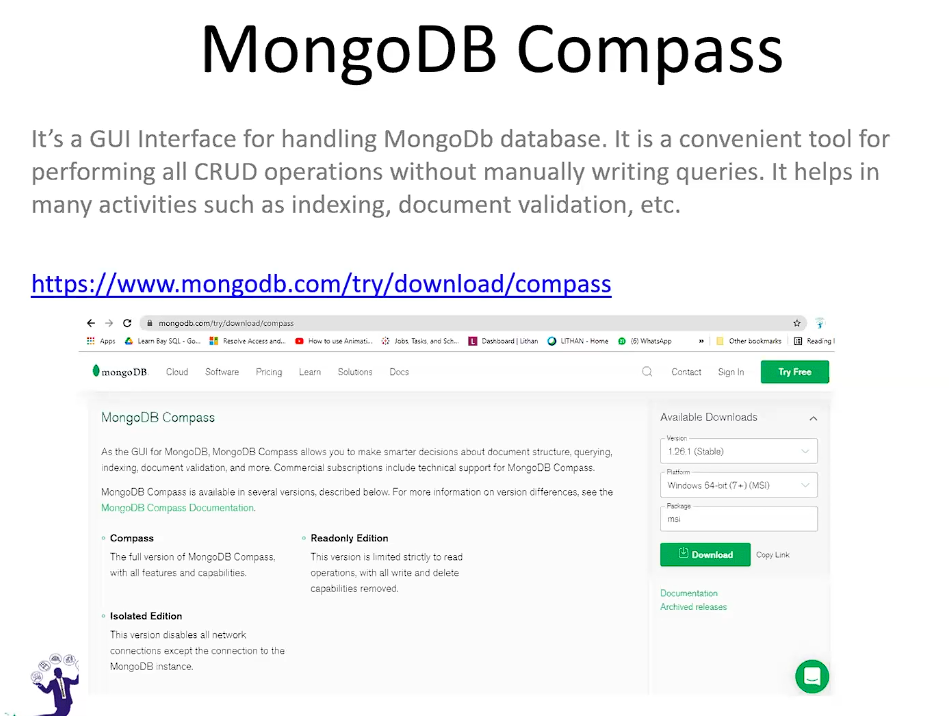


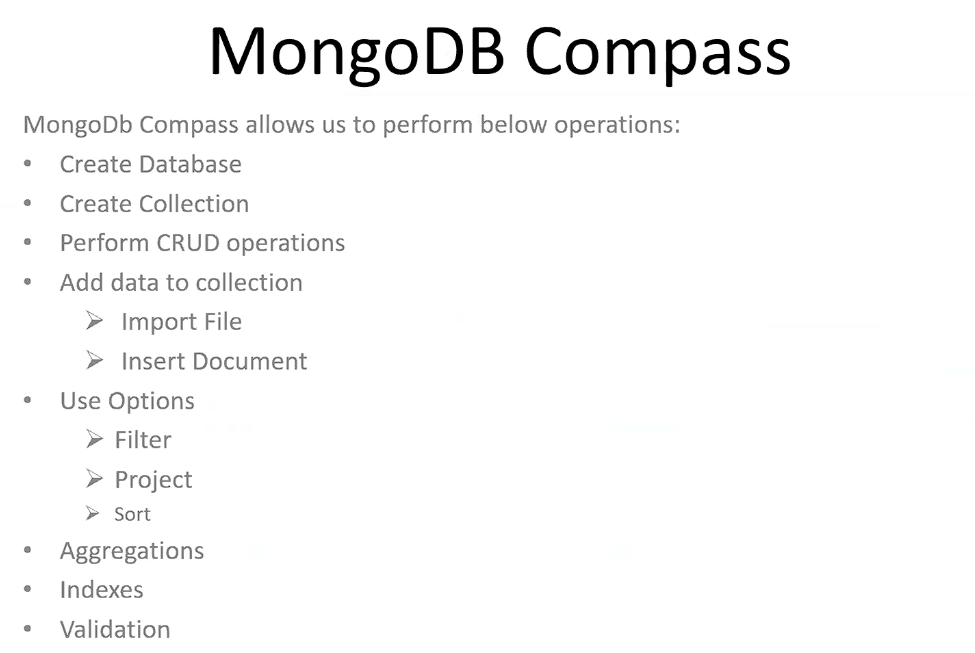
mongofiles put boy.jpg --db=emp

mongofiles get boy.jpg --db=emp

put – used to insert data in Mongo.

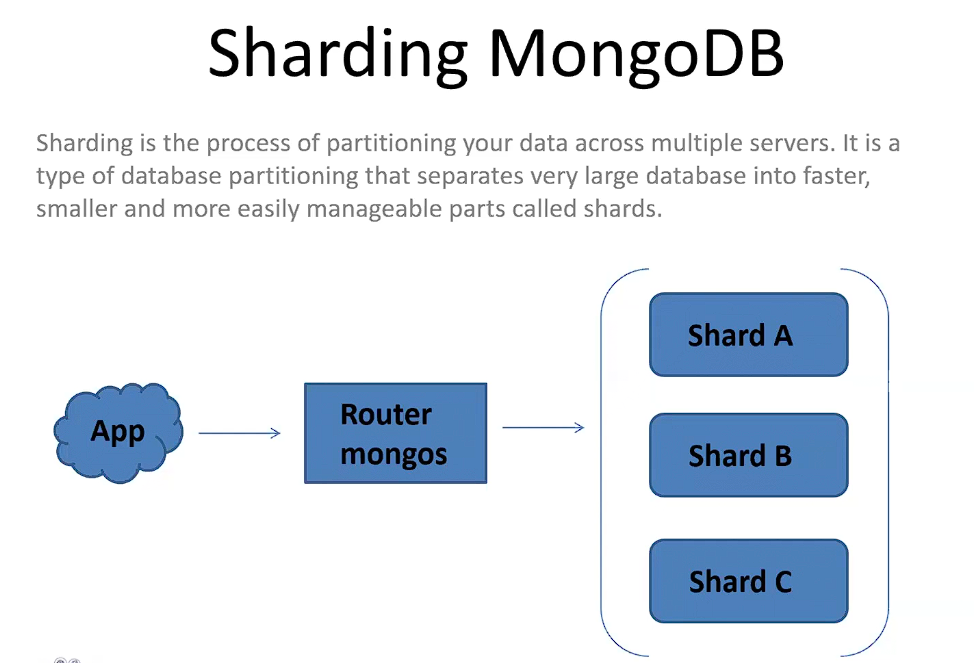
Get – used to get data from mongo.



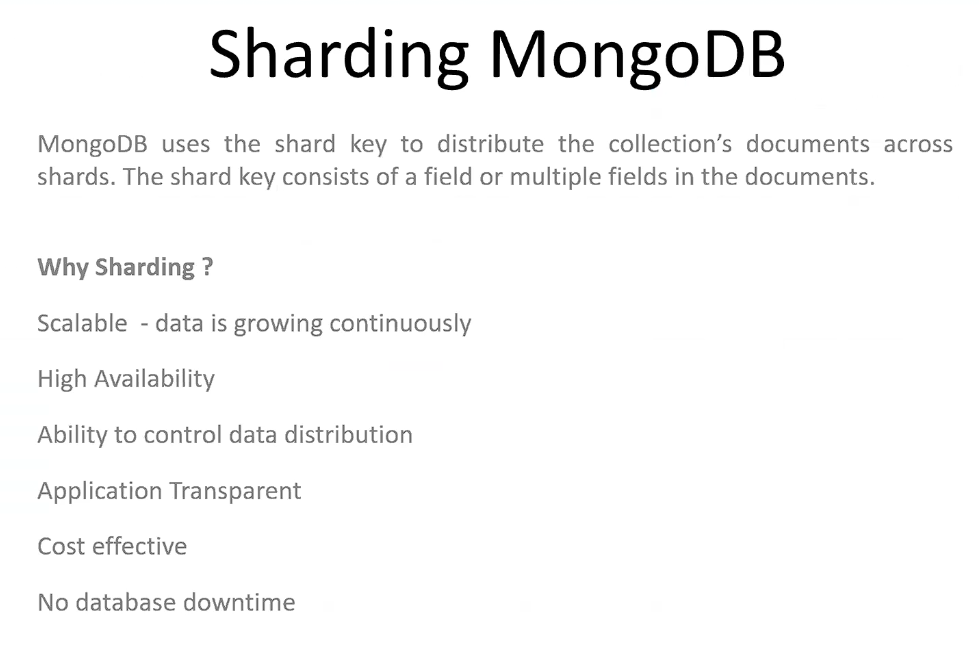


It is a GUI for MongoDB shell. Both the shell and compass are important as per the interviews.

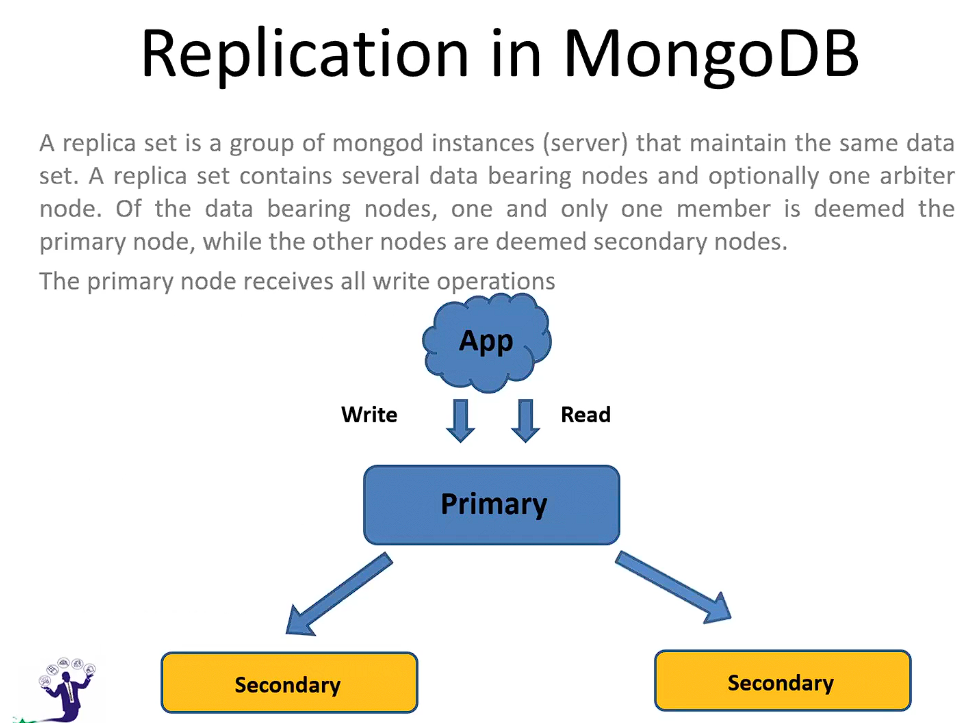
**Sharding in MongoDB (important for interviews):-**



It is a one-time effort which is done while setting up a mongodb data.

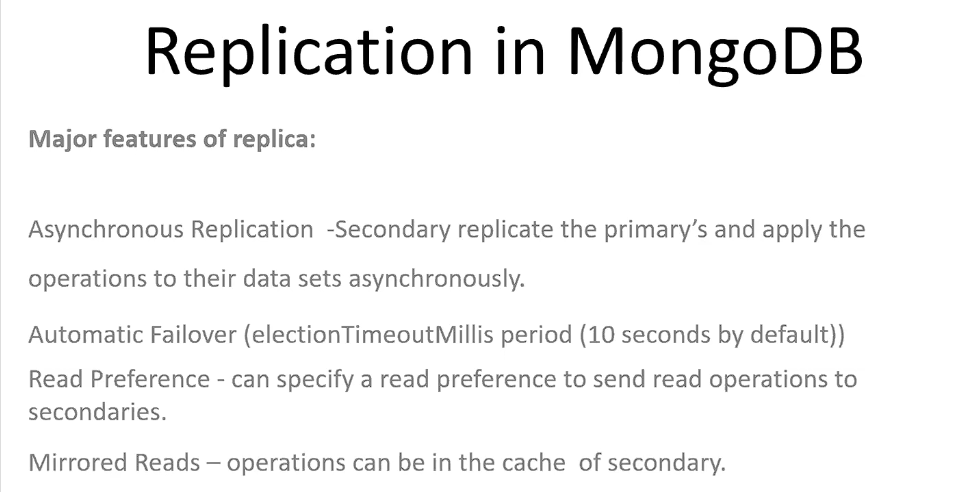


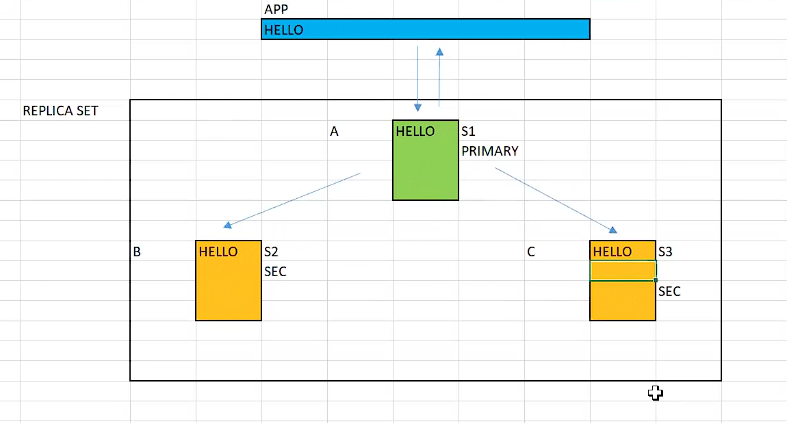
By looking at the shard key, the mongo comes to know that which data is lying in which shard.



A replica set is a group of or a collection of multiple servers. Despite of the number of servers, only 1 will be primary and rest all will be secondary.

The application will talk only to the primary server.





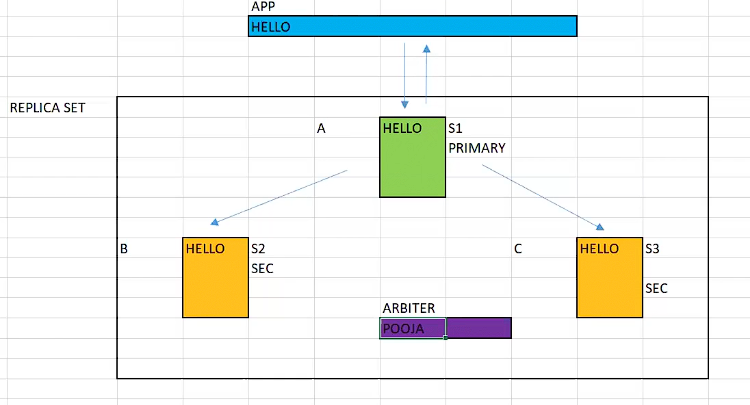
Asynchronous Replication:- The replication happens real-time without any delay.

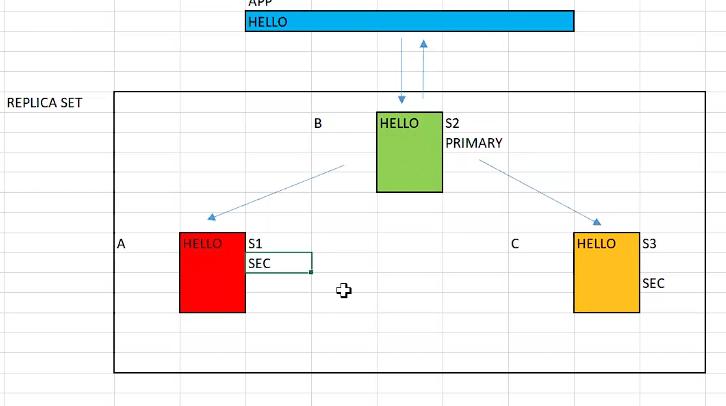
**\*\*Automatic Failover:-**

If for some reason, the primary server is not able to respond within 10 seconds, the secondary server will replace the primary server and will act as primary and the primary will now server as the secondary.

There could also be one other server, known as Arbiter, and the role of the Arbiter is to decide which secondary will act as the primary, in case the primary fails.

The arbiter will not contain any data. It is optional server.





**Read Preference:-** if the primary is busy with something, it redirects some of the incoming requests to the secondary servers.

Some of the frequent requests can be in the cache memory of the secondary.

\*\* Sharding and Replication are the networking concepts or the physical arrangement of servers. \*\*

The application will be transparent to all this arrangement and it will only be concerned with the fetching and writing data.