What you'll learn

Work with MongoDB with Clarity and Confidence
Working with mongodb tools like robo3T and noSQLBooster
Do Regex, GridFS , Replication , Sharding, Full text search
Basic and Advanced CRUD operations using MongoDB
Import and Export data from MongoDB
Work MapReduce, Embedded Documents,Save&Insert , indexing, capped collections, TTL

Disclaimer: Topics covered during the allotted time period depend upon the learning pace of the participants

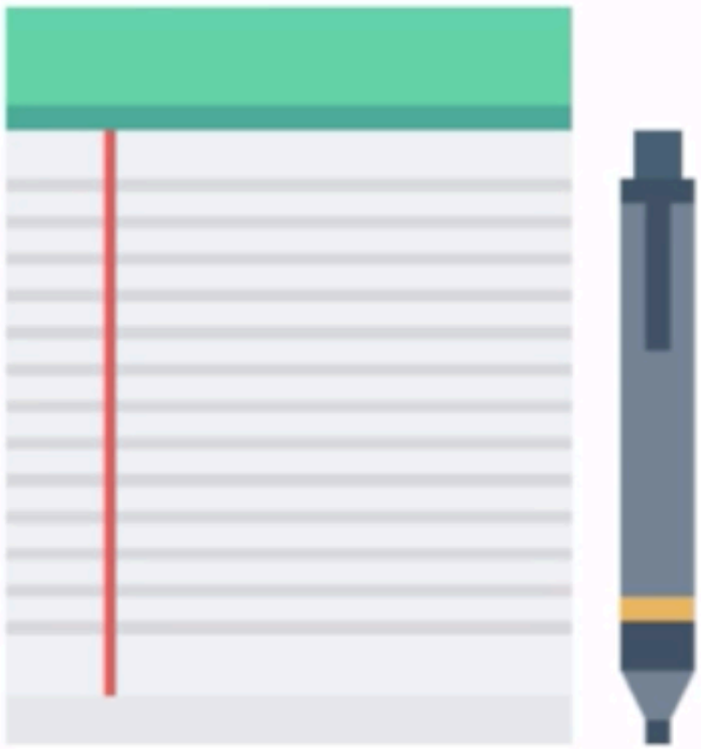
Course outline

- Find/Search Queries
- Update Queries
- Delete Queries
- Working with Arrays
- Full Text search
- Javascript with mongodb
- GridFS
- Aggregation

Course outline

- Sharding
- Replication
- MapReduce
- Indexing
- Regex
- Embedded Docs
- Capped Collections
- TTL Index
- MongodbDump

MongoDB



- What is MongoDB
- Installation
- Basic shell commands

Key Points

Documents

```
{
  "_id" : ObjectId("5501d48fb20c30e0548f"),
  "title" : "Article Title 1",
  "body" : "this is the article body"
}
```

BSON

NoSQL

non-Relational

SQL

Joins

NoSQL

ref = ‘User’

1

Main Advantages

Why Use MongoDB Over Traditional SQL?

Advantages

- Easy Schema Iteration
- Scalability & Performance
- Object-Oriented

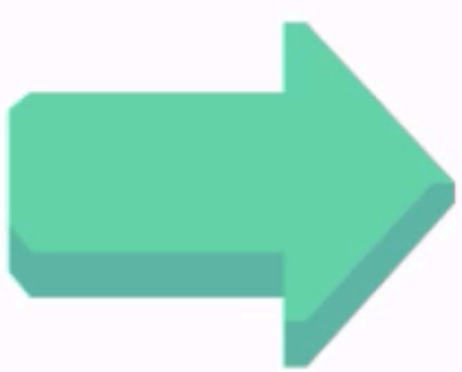
Agile Development

Accomodates large volumes of rapidly changing structured, semi-structured and unstructured data.

Add to Schema

User Schema

```
{
 first: “larry”,
 last: “david”
}
```



User Schema Updated

```
{
  first: “larry”,
  last: “david”,
  number: 555-9090
}
```

LEARN MORE

http://www.mongodb.com/nosql-explained

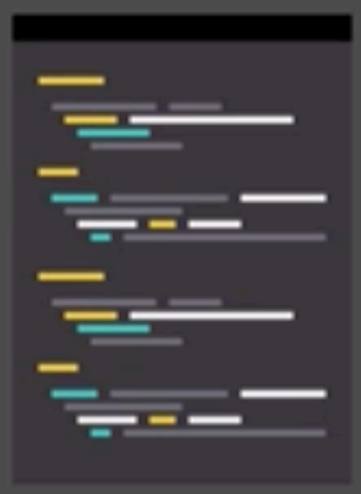
Document vs Collection

“

Document

A record in a MongoDB collection and the basic unit of data in MongoDB. Documents look like JSON objects but exist as BSON.

JSON = JavaScript Object Notation



Document Example

```
{
        "_id" : ObjectId("5591d39fb2933c30e0486f0b"),
        "title" : "Article two",
        "category" : "Education",
        "body" : "this is the body",
        "date" : ISODate("2015-06-29T23:24:15.212Z")
}
```

“

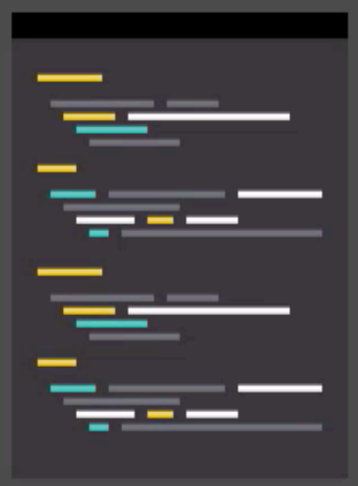
Collection

A grouping of MongoDB documents. Typically, all documents in a collection have a similar or related purpose.

Collection Example

```
{
        "_id" : ObjectId("5591d39fb2933c30e0486f0b"),
        "title" : "Article two",
        "category" : "Education",
        "body" : "this is the body",
        "date" : ISODate("2015-06-29T23:24:15.212Z")
}
{
        "_id" : ObjectId("5591d3ceb2933c30e0486f0c"),
        "title" : "Article three",
        "category" : "Health Care",
        "body" : "this is the body",
        "date" : ISODate("2015-06-29T23:25:02.226Z")
}
```

Schema = Collection



MongoDB Commands

Add, Modify, Find

Mongo Shell

DATA TYPES

- db.createCollection(“cars”)
- $upsert({ cars: ‘honda’ })
- Types of Data

DATA TYPES

STRING

name: String

```
{
  name: “John”
}
```

NUMBER

likes: Number

```
{
  likes: 5
}
```

DATE

timeStamp: Date

```
{
  timeStamp: ISODate(“...”)
}
```

ARRAY

tags: Array

OR

tags: []

```
{
  tags: [“tag1”, “tag2”]
}
```

BOOLEAN

published: Boolean

```
{
  published: true
}
```

ObjectId

_creator: Schema.ObjectId

```
{
  _creator: “41239878”
}
```

OTHER TYPES

Buffer

- Used for Video, Images, and Audio

Mixed

- Combines different types.

QUERY

find()

```
db.student.find({})
db.student.find({'name': 'Rachel'})
db.student.find({units: {$gt: 6}})
```

sort(), limit()

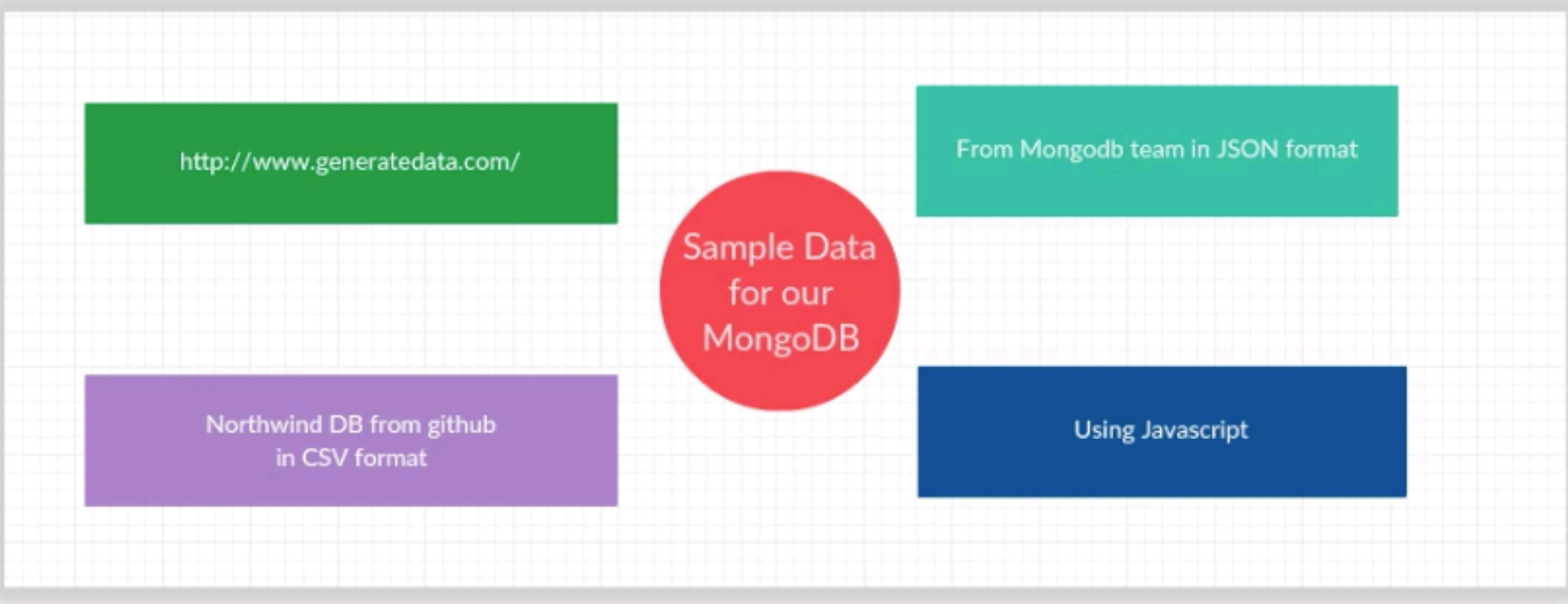
```
db.student.find({}).sort({name: 1}).limit(2)
```

```
db.student.find({})
db.student.find({'name': 'Rachel'})
db.student.find({units: {$gt: 6}})
db.student.find({units: {$lt: 7}})
db.student.find({classes: {$in: ['history']}})
```

```
db.student.find({classes: {$in: ['history']}}).sort({units: -1})    // ascending
db.student.find({}).sort({name: 1})    // descending

db.student.find({}).sort({name: 1}).limit(2)
```

https://docs.mongodb.com/manual/tutorial/query-documents/

Load Test Data in Mongodb



Mongodb Test Datasets

https://github.com/ozlerhakan/mongodb-json-files

https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

http://media.mongodb.org/zips.json

https://github.com/cjlee/northwind

https://github.com/tmcnab/northwind-mongo

Northwind Database

Products

	Column Name	Condensed...
PK	ProductID	int
	ProductName	nvarchar(40)
	SupplierID	int
	CategoryID	int
	QuantityPerUnit	nvarchar(20)
	UnitPrice	money
	UnitsInStock	smallint
	UnitsOnOrder	smallint
	ReorderLevel	smallint
	Discontinued	bit

Order Details

	Column ...	Conde...
PK	OrderID	int
PK	ProductID	int
	UnitPrice	money
	Quantity	smallint
	Discount	real

EmployeeTerritories

	Column Name	Condensed T...
PK	EmployeeID	int
PK	TerritoryID	nvarchar(20)

Orders

	Column Name	Condensed...
PK	OrderID	int
	CustomerID	nchar(5)
	EmployeeID	int
	OrderDate	datetime
	RequiredDate	datetime
	ShippedDate	datetime
	ShipVia	int
	Freight	money
	ShipName	nvarchar(40)
	ShipAddress	nvarchar(60)
	ShipCity	nvarchar(15)
	ShipRegion	nvarchar(15)
	ShipPostalCode	nvarchar(10)
	ShipCountry	nvarchar(15)

Customers

	Column Name	Condensed...
PK	CustomerID	nchar(5)
	CompanyName	nvarchar(40)
	ContactName	nvarchar(30)
	ContactTitle	nvarchar(30)
	Address	nvarchar(60)
	City	nvarchar(15)
	Region	nvarchar(15)
	PostalCode	nvarchar(10)
	Country	nvarchar(15)
	Phone	nvarchar(24)
	Fax	nvarchar(24)

Categories

	Column Name	Condensed...
PK	CategoryID	int
	CategoryName	nvarchar(15)
	Description	ntext
	Picture	image

Employees

	Column Name	Condensed ...
PK	EmployeeID	int
	LastName	nvarchar(20)
	FirstName	nvarchar(10)
	Title	nvarchar(30)
	TitleOfCourtesy	nvarchar(25)
	BirthDate	datetime
	HireDate	datetime
	Address	nvarchar(60)
	City	nvarchar(15)
	Region	nvarchar(15)
	PostalCode	nvarchar(10)
	Country	nvarchar(15)
	HomePhone	nvarchar(24)
	Extension	nvarchar(4)
	Photo	image
	Notes	ntext
	ReportsTo	int
	PhotoPath	nvarchar(255)

CustomerCustomerDemo

	Column Name	Condensed T...
PK	CustomerID	nchar(5)
PK	Customer...	nchar(10)

Suppliers

	Column Name	Condensed...
PK	SupplierID	int
	CompanyName	nvarchar(40)
	ContactName	nvarchar(30)
	ContactTitle	nvarchar(30)
	Address	nvarchar(60)
	City	nvarchar(15)
	Region	nvarchar(15)
	PostalCode	nvarchar(10)
	Country	nvarchar(15)
	Phone	nvarchar(24)
	Fax	nvarchar(24)
	HomePage	ntext

Territories

	Column Name	Condensed...
PK	TerritoryID	nvarchar(20)
	TerritoryDescription	nchar(50)
	RegionID	int

Region

	Column Name	Conden...
PK	RegionID	int
	RegionDescription	nchar(50)

Shippers

	Column Name	Condensed...
PK	ShipperID	int
	CompanyName	nvarchar(40)
	Phone	nvarchar(24)

CustomerDemographics

	Column Name	Conden...
PK	CustomerTypeID	nchar(10)
	CustomerDesc	ntext

mongoimport command line tool

https://docs.mongodb.com/manual/reference/program/mongoimport/

Import JSON File

```
mongoimport --db users --collection contacts --file contacts.json
```

Import CSV File

```
mongoimport --db users --collection contacts --type csv --headerline --
file /opt/backups/contacts.csv
```

Generate Dummy Data

https://mockaroo.com/

http://generatedata.com/

Assignment

- Use the testdatagenerator available with MongoBooster to generate and Load sample data(for eg) Product details) to MongoDB. Refer here for More info https://mongobooster.com/features#TestDataGenerator

- Use https://www.mockaroo.com/ to generate user profile data a JSON output and load the data to MongoDB using MongoBooster and then using Mongoimport utility

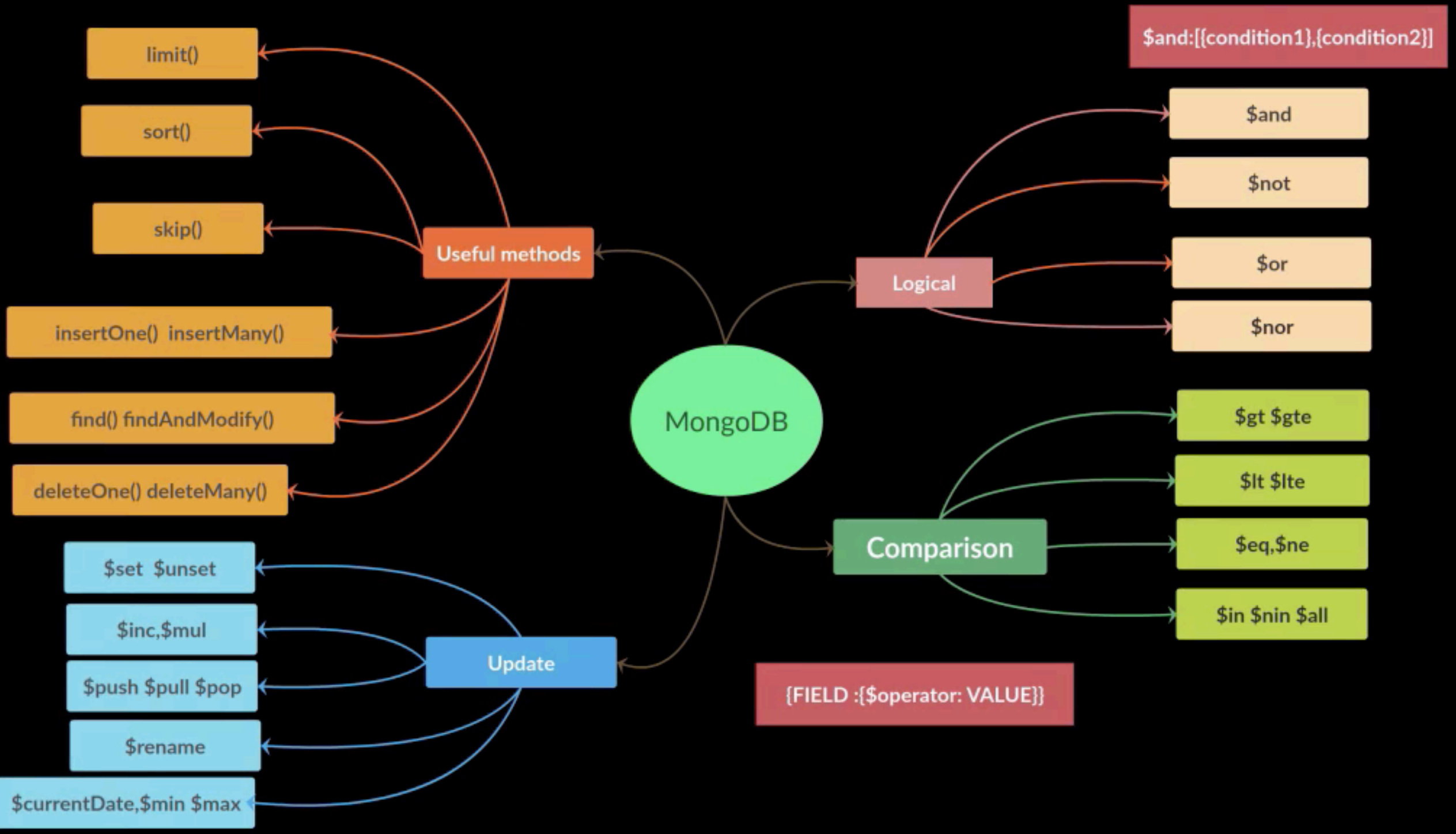
- Repeat the same Exercise using CSV format output from mockaroo.

- Load demo data provided by mongodb team using MongoBooster/MongoImport Tool. Demo Data URL: https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

- Load again an useful demo dataset provided by mongodb team. The information is available here :https://docs.mongodb.com/manual/tutorial/aggregation-zip-code-data-set/

- The collection is available : http://media.mongodb.org/zips.json?_ga=2.261334925.1881078164.1503396731-778080937.1486117063

Queries


limit()
sort()
skip()
Useful methods
insertOne() insertMany()
find() findAndModify()
deleteOne() deleteMany()
$set $unset
$inc,$mul
$push $pull $pop
$rename
$currentDate,$min $max
Update
MongoDB
$and:[{condition1},{condition2}]
$and
$not
$or
$nor
Logical
$gt $gte
$lt $lte
$eq,$ne
$in $nin $all
Comparison
{FIELD :{$operator: VALUE}}

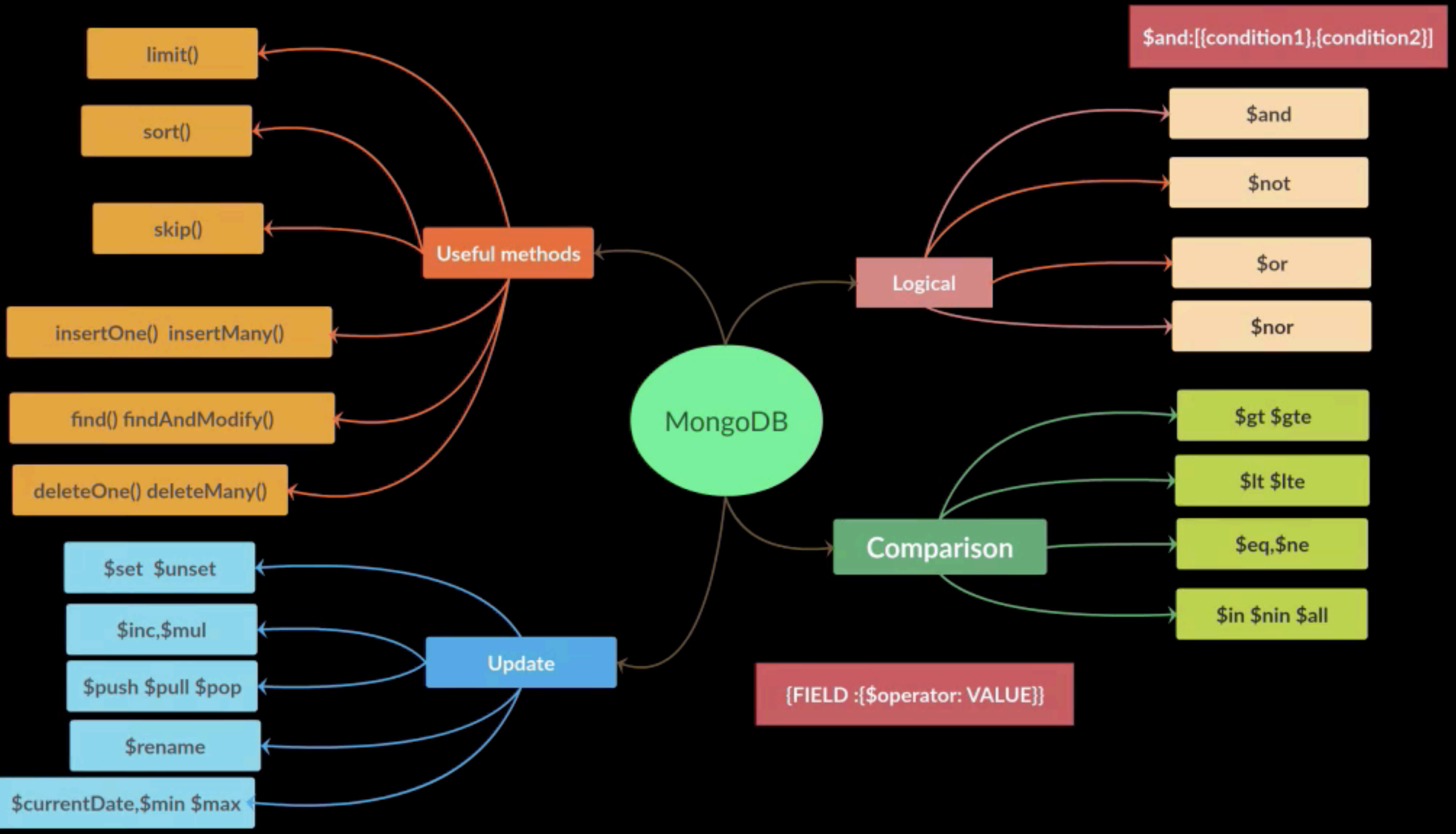
```
find(QUERY, FIELDS TO SELECT)
```

```
find(QUERY with Conditions, FIELDS TO SELECT)
```

```
db.getCollection('regions').find({QUERY},{FIELDS TO SELECT})

db.getCollection('regions').find({}).limit(2)

db.getCollection('regions').find({}).skip(2)

db.getCollection('regions').find().count()

db.getCollection('regions').find({RegionDescription:"Northern"})

db.getCollection('regions').find({RegionDescription:"Northern"})

db.getCollection('regions').find({RegionDescription:"Northern"},{RegionID:1})

db.getCollection('regions').find({RegionDescription:"Northern"},{RegionID:1,_id:0})

db.getCollection('regions').find({"RegionID" : 1,"RegionDescription" : "Eastern"})
```

count()
limit(<Number of documents>)
skip(<Number of documents>)
sort(1 for asc -1 for Desc order)

Now Logical and Comparison



Queries Quiz

Question 1:

This Query db.inventory.find({ qty: { $in: [5, 15] } }) finds the documents in the collection inventory, having qty as 5 and 15.

Is this correct?

Queries Quiz

Question 1:

This Query db.inventory.find({ qty: { $in: [5, 15] } }) finds the documents in the collection inventory, having qty as 5 and 15.

Is this correct?

Ans: No . $in is similar to $or operator. So it selects either 5 or 15

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/in/

Queries Quiz

Question 2:

When performing equality checks on the same field,$in operator is more preferable than $or operator. Is this True?

Is this correct?

Queries Quiz

Question 2:

When performing equality checks on the same field,$in operator is more preferable than $or operator. Is this True?

Is this correct?

Ans: Yes

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/in/

Queries Quiz

Question 3:

In this Query using $not operator, the result will have two types of documents

1. Documents with the price field value is less than or equal to 1.99

2. Documents without price field

```
db.inventory.find( { price: { $not: { $gt: 1.99 } } } )
```

Is this correct?

Queries Quiz

Question 3:

In this Query using $not operator, the result will have two types of documents

1. Documents with the price field value is less than or equal to 1.99

2. Documents without price field

db.inventory.find({ price: { $not: { $gt: 1.99 } } })

Is this correct? Ans: Yes

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/not/

Queries Quiz

Question 4:

In this Query using $lte operator, the result will have two types of documents

1. Documents with the qty field value is less than or equal to 20
2. Documents without qty field

db.inventory.find({ qty: { $lte: 20 } })

Is this correct?

Queries Quiz

Question 4:

In this Query using $lte operator, the result will have two types of documents

1. Documents with the qty field value is less than or equal to 20

2. Documents without qty field

```
db.inventory.find( { qty: { $lte: 20 } } )
```

Is this correct?

Ans: No. It wont return documents without qty field, contrary to the $not operator.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/lte/

Queries Quiz

Question 5:

This Query will produce the _id field in the result. Is this True?

db.products.find({ qty: { $gt: 25 } }, { item: 1, qty: 1 })

Is this correct?

Queries Quiz

Question 5:

This Query will produce the _id field in the result. Is this True?

db.products.find({ qty: { $gt: 25 } }, { item: 1, qty: 1 })

Is this correct?

Ans: Yes, _id is not specifically filtered out in the above Query.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/method/db.collection.find/

Queries Quiz

Question 6:

In the Query result of the below query all the fields will appear in the result except the _id and the Qty Field.

db.products.find({ qty: { $gt: 25 } }, { _id: 0, qty: 0 })

Is this correct?

Queries Quiz

Question 6:

In the Query result of the below query all the fields will appear in the result except the _id and the Qty Field.

db.products.find({ qty: { $gt: 25 } }, { _id: 0, qty: 0 })

Is this correct?

Ans: Yes, Because _id and Qty are explicitly filtered out.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/method/db.collection.find/

Queries Quiz

Question 7:

The two statements produce equal results.

```
db.bios.find().sort( { name: 1 } ).limit( 5 )
```

```
db.bios.find().limit( 5 ).sort( { name: 1 } )
```

Is this correct?

Queries Quiz

Question 7:

The two statements produce equal results.

db.bios.find().sort({ name: 1 }).limit(5)

db.bios.find().limit(5).sort({ name: 1 })

Is this correct?

Ans: Yes, The order of limit() and the sort() methods is not significant.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/method/db.collection.find/

Update Documents

db.collection.update(query, update, options)	upsert
Replace entire document	$inc $mul
update specific fields using $set	$unset
Multiple Updates	$currentDate
updateOne()	$min $max
updateMany()	update Arrays

Delete Documents

```
deleteOne()
deleteMany()
remove() with option (justOne)
```

Working with Arrays

Types of Array Data

complex / associative

```
"grades" : [
    {
        "grade" : 92,
        "mean" : 88,
        "std" : 8
    },
    {
        "grade" : 78,
        "mean" : 90,
        "std" : 5
    },
    {
        "grade" : 88,
        "mean" : 85,
        "std" : 3
    }
],
```

simple

```
"scores" : [
    78,
    88,
    66
]
```

Working with Arrays

Array Quiz

Question 1:

Using $length array Operator, we can find documents with arrays with the number of elements specified.

Is this correct?

Array Quiz

Question 1:

Using $length array Operator, we can find documents with arrays with the number of elements specified.

Is this correct?

Ans: Nope. Its $size array Operator.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/size/#op._S_size

Array Quiz

Question 2:

$all Array Operator is similar to $or operation .Selects all documents where the array has even a single element matching the values passed in the $all operator.

Is this correct?

Array Quiz

Question 2:

$all Array Operator is similar to $or operation .Selects all documents where the array has even a single element matching the values passed in the $all operator.

Is this correct?

Ans: No, Its the opposite way. Its like $and operation. Selects complete matching.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/all/

Array Quiz

Question 3:

$elemMatch operator supports multiple queries,so documents with array fields, with atleast one element matching the specified Queries will be returned.

Is this correct?

Array Quiz

Question 3:

$elemMatch operator supports multiple queries,so documents with array fields, with atleast one element matching the specified Queries will be returned.

Is this correct?

Ans: yes, You can get more info about $elemMatch

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/elemMatch/#op._S_elemMatch

Array Quiz

Question 4:

We cannot pass multiple Query conditions[Compound Filter Conditions on the Array Elements] to find method. For that we have to use $elemMatch operator .

Is this correct?

Array Quiz

Question 4:

We cannot pass multiple Query conditions[Compound Filter Conditions on the Array Elements] to find method. For that we have to use $elemMatch operator .

Is this correct?

Ans:Nope. You can pass multiple Query criteria to find method.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/tutorial/query-arrays/

Document Validation

In Mongodb Collections, we can validate the documents before they get inserted using the options in createCollection method

https://docs.mongodb.com/manual/reference/method/db.createCollection/

```
db.createCollection(<name>, { capped: <boolean>,
                              autoIndexId: <boolean>,
                              size: <number>,
                              max: <number>,
                              storageEngine: <document>,
                              validator: <document>,
                              validationLevel: <string>,
                              validationAction: <string>,
                              indexOptionDefaults: <document>,
                              viewOn: <string>,
                              pipeline: <pipeline>,
                              collation: <document>,
                              writeConcern: <document>} )
```

```
db.createCollection(name, options)

(validator: <document>,
 validationLevel: <string>,
validationAction: <string>)
```

`validationLevel`	Description
`"off"`	No validation for inserts or updates.
`"strict"`	**Default** Apply validation rules to all inserts and all updates.
`"moderate"`	Apply validation rules to inserts and to updates on existing *valid* documents. Do not apply rules to updates on existing *invalid* documents.

`validationAction`	Description
`"error"`	**Default** Documents must pass validation before the write occurs. Otherwise, the write operation fails.
`"warn"`	Documents do not have to pass validation. If the document fails validation, the write operation logs the validation failure.

Validation Example

Assume a collection - AmazonSpecialOfferCustomers

The Offer is valid for Customers

1. Total purchases >= 10000
2. Customers in Countries
    1. us
    2. uk
    3. brazil

Document validation

Read Links

https://docs.mongodb.com/manual/reference/method/db.createCollection/

https://docs.mongodb.com/manual/reference/command/collMod/

https://docs.mongodb.com/manual/reference/method/db.runCommand/

Should try

add or edit collection validator with noSQLBooster
add or edit collection validator with Robo3T

What validations remain in the app

- User interface
  - Don’t have the database be the first place to detect that an email is poorly formatted
- Any validations that involve comparisons with
  - Other data in the same document
  - Data from other documents
  - External information (e.g. time of day)
- Semantic checks that are designed to fail frequently
  - e.g. user is in wrong country to use this service
  - Database should typically be testing for coding errors rather than implementing your business logic
- Determining why the database rejected a document in order to provide a meaningful error to the user

Full Text Search

MongoDB Full Text Search

Some of the features include

- Soundex or Fuzzy Matching (not supported yet but possible)

  https://www.mongodb.com/blog/post/how-to-perform-fuzzy-matching-with-mongo-connector

- Word breakers or Tokenization

  splitting sentences into meaningful words

- Stemming

searching importance will include word important as well

- Ranking (relevance score)

- Stop words skipping (and, the etc..)

the index size is huge that is why a lot of cloud providers does not support full text search

```
{
  $text:
    {
      $search: <string>,
      $language: <string>,
      $caseSensitive: <boolean>,
      $diacriticSensitive: <boolean>
    }
}
```

https://docs.mongodb.com/manual/text-search/

//https://docs.mongodb.com/manual/reference/operator/query/text/#op._S_text

https://docs.mongodb.com/manual/reference/operator/projection/meta/#proj._S_meta

Text Search features

- Create Text Index
- Exclude content
- Case sensitive search
- Searching Phrases
- Stop words will be skipped
- metascore

Try this

Download a data set and create text search index, then try doing some text searches

https://github.com/google-research-datasets/sentence-compression

https://medium-json-feed.herokuapp.com/the-web-tub/trending

https://medium-json-feed.herokuapp.com/@mikeal/latest

Text Search Quiz

Question 1:

db.collection.createIndex() method is required for creating an Text Index on a Field. Is this True?

Is this correct?

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 1:

db.collection.createIndex() method is required for creating an Text Index on a Field. Is this True?

Is this correct?

Ans: Yes

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 2:

```
db.reviews.createIndex(
    {subject: "text",
      comments: "text"}
  )
```

The above query will fail as it is creating multiple Text indexes on two fields

Is this correct?

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 2:

```
db.reviews.createIndex(
    {subject: "text",
      comments: "text"}
  )
```

The above query will fail as it is creating multiple Text indexes on two fields

Is this correct?

Ans: No, Multiple - Compound Text indexes are allowed and they are very useful.

Please visit the MongoDB Documentation page for more info https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 3:

db.collection.createIndex({ "$**": "text" }) is a Valid Query , as it is creating a index on multiple fields using wildcard.

Is this correct?

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/core/index-text/#wildcard-text-indexes

Text Search Quiz

Question 3:

db.collection.createIndex({ "$**": "text" }) is a Valid Query , as it is creating a index on multiple fields using wildcard.

Is this correct?

Ans: Yes, In MongoDB wildcard text indexes are allowed. Its very handy .

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/core/index-text/#wildcard-text-indexes

Text Search Quiz

Question 4:

For Text search to match on a phrase, enclose the phrase in double quotes (")

"Hello World". Is this Correct ?

Is this correct?

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 4:

For Text search to match on a phrase, enclose the phrase in double quotes (")

"Hello World". Is this Correct ?

Is this correct?

Ans: Yes, Escaped double quotes (\") needs to be used. Example "\"Hello World\""

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 5:

In this search text `hello-world` MongoDB will use the - Negation(Minus) symbol to excludes the documents with the word world. Is this Correct?

Is this correct?

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 5:

In this search text `hello-world` MongoDB will use the - Negation(Minus) symbol to excludes the documents with the word world. Is this Correct?

Is this correct?

Ans: No, It will negate only when the word is like this or else it will be considered as hyphenated. So for this , it has to be written like this hello -world

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 6:

For case sensitive search option ,$caseSensitive: true can be used . Is this correct?

Is this correct?

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/reference/operator/query/text/

Text Search Quiz

Question 6:

For case sensitive search option ,$caseSensitive: true can be used . Is this correct?

Is this correct?

Ans: Yes and its a common feature of most text search engines.

For more info on Text Search Visit this MongoDB Documentation Link https://docs.mongodb.com/manual/reference/operator/query/text/

Javascript With Mongodb

- javascript in shell
- Mongodb jsmethods and commands
- cursor
- Using eval()
- using system.js
- External .js file

Javascript in Shell

REPL stands for Read-Eval-Print-Loop.

The `mongo` shell is an interactive JavaScript interface to MongoDB. You can use the `mongo` shell to query and update data as well as perform administrative operations.

we can write js code in shell like this

```
for (let index = 0; index < 100; index++) {
    let doc = {};
    doc.randomNumber = Math.floor(Math.random()*100)+1;
    db.CollectionJS.insert(doc);
}

let greaterThanNumber = 50;
let searchQuery = {
    randomNumber:{$gt:greaterThanNumber}
}
db.CollectionJS.find(searchQuery)
```

https://docs.mongodb.com/manual/mongo/

Mongodb jsmethods and commands

You can write scripts for the `mongo` shell in JavaScript that manipulate data in MongoDB or perform administrative operation.

https://docs.mongodb.com/manual/tutorial/write-scripts-for-the-mongo-shell/

Shell Helpers	JavaScript Equivalents
`show dbs, show databases`	`db.adminCommand('listDatabases')`
`use <db>`	`db = db.getSiblingDB('<db>')`
`show collections`	`db.getCollectionNames()`
`show users`	`db.getUsers()`

Cursor

https://docs.mongodb.com/manual/tutorial/iterate-a-cursor/

https://docs.mongodb.com/manual/reference/method/js-cursor/

```
//!Cursor Copy data to another collection

let cursor = db.getCollection('CollectionJS').find({})
//loop through cursor
while(cursor.hasNext()){
    printjson(cursor.next().randomNumber); //print the value
    db.CollectionJSCopy.insert(cursor.next());
}
```

eval & db.eval()

Depricated in version 3.0

https://docs.mongodb.com/manual/reference/command/eval/

https://docs.mongodb.com/manual/reference/method/js-cursor/

The `eval` command evaluates JavaScript functions on the database server.

```
{
   eval: <function>,
   args: [ <arg1>, <arg2> ... ],
   nolock: <boolean>
}
```

`db.eval()` method does not support the `nolock` option. everything else is same

Using system.js

https://docs.mongodb.com/manual/tutorial/store-javascript-function-on-server/

Do not store application logic in the database. There are performance limitations to running JavaScript inside of MongoDB.

There is a special system collection named `system.js` that can store JavaScript functions for reuse.

To store a function, you can use the `db.collection.save()`, as in the following examples:

```
db.system.js.save(
   {
     _id : "myAddFunction" ,
     value : function (x, y){ return x + y; }
   }
);
```

Execute a JavaScript file

https://docs.mongodb.com/manual/tutorial/write-scripts-for-the-mongo-shell/#execute-a-javascript-file

You can execute a `.js` file from within the `mongo` shell, using the `load()` function, as in the following:

```
load("myjstest.js")
```

The `load()` method accepts relative and absolute paths. If the current working directory of the `mongo` shell is `/data/db`, and the `myjstest.js` resides in the `/data/db/scripts` directory, then the following calls within the `mongo` shell would be equivalent:

```
load("scripts/myjstest.js")
load("/data/db/scripts/myjstest.js")
```

GridFS

GridFS is a specification for storing and retrieving files that exceed the BSON-document size limit of 16 MB.

GridFS divides the file into parts, or chunks, and stores each chunk as a separate document.

GridFS uses a default chunk size of 255 kB

GridFS uses two collections to store files. One collection stores the file chunks, and the other stores file metadata.

You can perform range queries on files stored through GridFS

file chunk - 255KB

file chunk - 255KB

Collection 1 - fs.chunks

file chunk - 255KB

file chunk - 255KB

Can have many buckets in a Database

metadata

Collections 2- fs.files

metadata

mongodb

Mongofiles command

Bucket
[Base name default is fs]

GridFS

GridFS with NoSQLBooster

Create a different db for learning GridFS but this is not necessary

Create a Bucket Collection

upload file

view chunks

view stored file info

view stored file from db

GridFS

https://docs.mongodb.com/manual/core/gridfs/

McAfeeoptimizes delivery of analytics and incremental updates in MongoDB as binary packages for efficiently delivery to customers.

Pearson stores student data in GridFS and leverages MongoDB's replication to distribute data and keep it synchronized across facilities.

Additional Resource Articles

https://www.mongodb.com/blog/post/building-mongodb-applications-binary-files-using-gridfs-part-1?jmp=docs

https://www.mongodb.com/blog/post/building-mongodb-applications-binary-files-using-gridfs-part-2?jmp=docs

GridFS

mongofiles utility

The `mongofiles` utility makes it possible to manipulate files stored in your MongoDB instance in GridFS objects from the command line. It is particularly useful as it provides an interface between objects stored in your file system and GridFS.

```
mongofiles <options> <commands> <filename>
```

To upload a file named `32-corinth.lp` to the GridFS collection in the `records` database

```
mongofiles -d records put 32-corinth.lp
```

To return a list of all files in a GridFS collection in the `records` database

```
mongofiles -d records list
```

https://docs.mongodb.com/manual/reference/program/mongofiles/

GridFS in Mongodb Drivers

http://mongodb.github.io/mongo-csharp-driver/2.7/reference/gridfs/

http://mongodb.github.io/mongo-java-driver/3.9/driver/tutorials/gridfs/

http://mongodb.github.io/node-mongodb-native/3.1/tutorials/gridfs/

Why GridFS

- The resulting application will have a simpler architecture: one system for all types of data;

- Document metadata can be expressed using the rich flexible document structure, and documents can be retrieved using all the flexibility of MongoDB's query language;

- MongoDB's high availability (replica sets) and scalability (sharding) infrastructure can be leveraged for binary data as well as the structured data;

- One consistent security model for authenticating and authorizing access to the metadata and files;

- GridFS doesn't have the limitations of some filesystems, like number of documents per directory, or file naming rules.

Aggregation

3 Types of Aggregation

Aggregation pipeline

Map Reduce

Single Purpose Aggregation Operations
- count, distinct, group

https://docs.mongodb.com/manual/aggregation/

Aggregation pipeline

Collection

```
db.orders.aggregate( [
    $match stage ──►   { $match: { status: "A" } },
    $group stage ──►   { $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
                     ] )
```

orders

```
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
```

$match

```
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
```

$group

Results

```
{ _id: "A123", total: 750 }
{ _id: "B212", total: 200 }
```

https://docs.mongodb.com/manual/aggregation/#aggregation-pipeline

Aggregation pipeline

Aggregation Pipeline Stages

```
db.collection.aggregate( [ { <stage> }, ... ] )
```

https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/#aggregation-pipeline-stages

Aggregation

- $group
- $match
- $sort
- $limit
- $skip
- $project
- $sum
- $avg

Map Reduce

Collection

```
db.orders.mapReduce(
          map      →   function() { emit( this.cust_id, this.amount ); },
          reduce   →   function(key, values) { return Array.sum( values ) },
                       {
          query    →     query: { status: "A" },
          output   →     out: "order_totals"
                       }
                     )
```

orders

```
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
```

query

```
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
```

map

```
{ "A123": [ 500, 250 ] }
{ "B212": 200 }
```

reduce

order_totals

```
{ _id: "A123", value: 750 }
{ _id: "B212", value: 200 }
```

Single Purpose Aggregation Operations

Collection

```
db.orders.distinct( "cust_id" )
```

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
```

```
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
```

```
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

```
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

orders

distinct → ["A123", "B212"]

[db.collection.distinct()](#)
[db.collection.count()](#)
[db.collection.group()](#)

https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/#aggregation-pipeline-stages

Aggregation Pipeline

The MongoDB aggregation pipeline consists of stages.

Each stage transforms the documents as they pass through the pipeline.

Pipeline stages do not need to produce one output document for every input document; e.g., some stages may generate new documents or filter out documents.

Pipeline stages can appear multiple times in the pipeline.

https://docs.mongodb.com/manual/core/aggregation-pipeline/

Aggregation Pipeline Operators

https://docs.mongodb.com/manual/reference/operator/aggregation/

- Arithmetic Expression Operators
- Array Expression Operators
- Boolean Expression Operators
- Comparison Expression Operators
- Conditional Expression Operators
- Date Expression Operators
- Literal Expression Operator
- Object Expression Operators
- Set Expression Operators
- String Expression Operators
- Text Expression Operator
- Type Expression Operators
- Accumulators (`$group`)
- Accumulators (`$project`)
- Variable Expression Operators

Operators

```
{
_id: ....
author: "edouard",
status: "active",
post:"this is brilliant and wise and useful",
tags:["brilliant", "wise", "useful"],
views: 10
}

{
_id: ....
author: "ron",
status: "active",
post:"this is phenomenal and useful",
tags:["phenomenal", "useful"],
views: 15
}
```

```
SELECT author,
      count(*) as postCnt,
      sum(views) as viewCnt
FROM  ....
GROUP BY author
```

```
db.agg_test.aggregate(
{$group :
      {_id: "$author",
      postCnt: {$sum : 1},
      viewCnt: {$sum: "$views"}
      }
}
)
```

$match	$project	$unwind	$group	$match	$sort	$limit
where clause	select clause	pivot an array	group by clause	having clause	order by clause	top clause

Operators

```
{
_id: ....
author: "edouard",
status: "active",
post:"this is brilliant and wise and useful",
tags:["brilliant", "wise", "useful"],
views: 10
}


{
_id:....
author: "ron",
status: "active",
post:"this is phenomenal and useful",
tags:["phenomenal", "useful"],
views: 15
}
```

```
        "result" : [
                {
                        "_id" : "useful",
                        "authors" : [
                                "ron",
                                "edouard"
                        ],
                        "postCnt" : 2,
                        "viewCnt" : 25
                },
                {
                        "_id" : "phenomenal",
                        "authors" : [
                                "ron"
                        ],
                        "postCnt" : 1,
                        "viewCnt" : 15
                }
        ],
        "ok" : 1
}
```

$match where clause

$project select clause

$unwind pivot an array

$group group by clause

$match having clause

$sort order by clause

$limit top clause

Operators

```
{
_id: ....
author: "edouard",
status: "active",
post:"this is brilliant and wise and
      useful",
tags:["brilliant", "wise", "useful"],
views: 10
}

{
_id:....
author: "ron",
status: "active",
post:"this is phenomenal and
      useful",
tags:["phenomenal", "useful"],
views: 15
}
```

```
db.agg_test.aggregate(

{$match: {status: "active"}},
{$project :
      {author: 1,
      tags: 1,
      views : 1
      }
},
{$unwind : "$tags"},
{$group :
      {_id: "$tags",
      authors: { $addToSet: "$author"},
      postCnt: {$sum : 1},
      viewCnt: {$sum: "$views"}
      }
} ,
{$match : {viewCnt : {$gte: 10}}},
{$sort: {viewCnt : -1}},
{$limit: 2}

)
```

```
{
"result" : [
      {
      "_id" : "useful",
      "authors" : [
                  "ron",
                  "edouard"
                  ],
      "postCnt" : 2,
      "viewCnt" : 25
      },
      {
      "_id" : "phenomenal",
      "authors" : [
                  "ron"
                  ],
      "postCnt" : 1,
      "viewCnt" : 15
      }
],
"ok" : 1
}
```

$match where clause → **$project** select clause → **$unwind** pivot an array → **$group** group by clause → **$match** having clause → **$sort** order by clause → **$limit** top clause

Aggregation Pipeline Limits

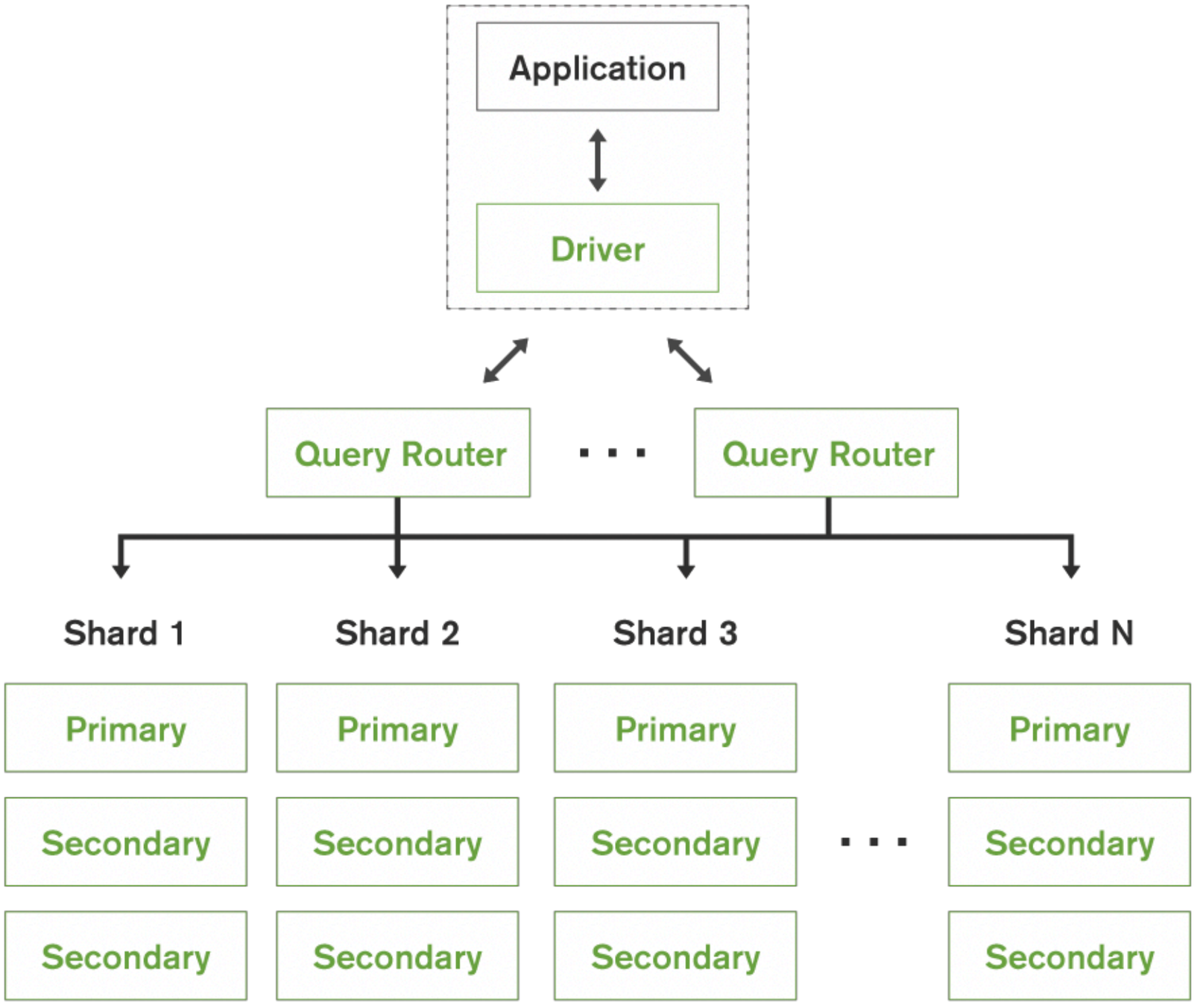
https://docs.mongodb.com/manual/core/aggregation-pipeline-limits/

Sharding

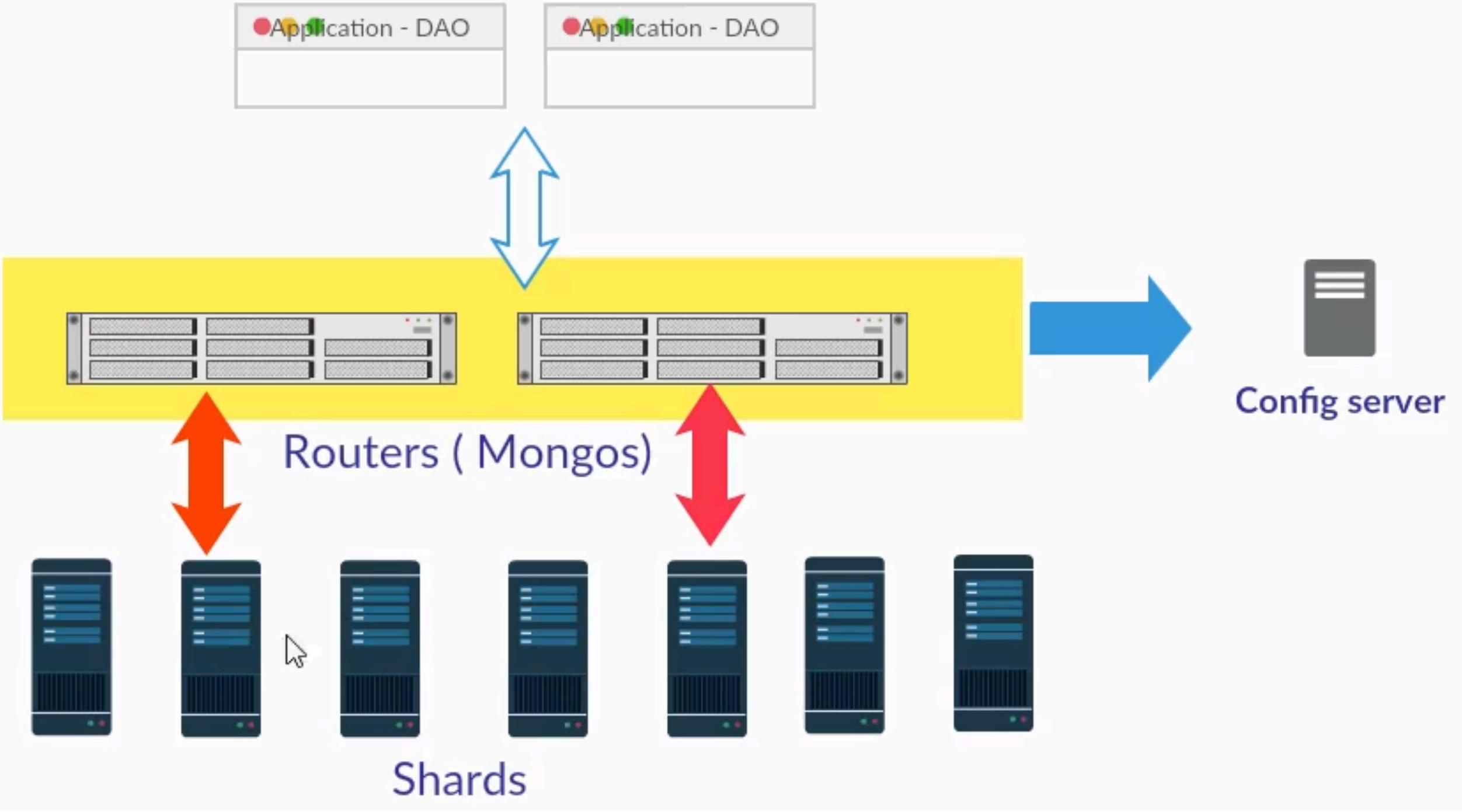
Sharding is a type of database partitioning that separates very large databases the into smaller, faster, more easily managed parts called data shards. The word **shard** means a small part of a whole.

Sharding involves a **shard key** *defined* by a data modeler that describes the partition space of a data set.
Data is partitioned into data **chunks** by the shard key, and these chunks are distributed evenly across **shards** that reside across many physical servers.
MongoDB provides **3 Types** of Sharding Strategies:

- Ranged
- Hashed
- Tag-aware


Application
Driver
Query Router
Query Router
Shard 1
Shard 2
Shard 3
Shard N
Primary
Primary
Primary
Primary
Secondary
Secondary
Secondary
Secondary
Secondary
Secondary
Secondary
Secondary

Sharding


Application - DAO
Application - DAO
Routers (Mongos)
Config server
Shards

Sharding Example Steps

create database directories for the 5 servers involved

1 -config server
1 - Query Router
3 - Sharding instances

1. Create instances of sharding server
2. create config server
3. create router for the config server
4. create sharding instances
5. in the admin database add sharding instances, database and collection through router
6. in the query router-mongos insert data
7. validate the data inserted is distributed among servers

```
 //Create data directories for 3 Shard Servers & 1 for the config server
  mkdir c:\data\srx\dbxv1
  mkdir c:\data\srx\dbxv2
  mkdir c:\data\srx\dbxv3
  mkdir c:\data\configdbxv_2


//Start Shard Servers on different ports
  mongod --shardsvr --port 30001 --dbpath c:\data\srx\dbxv1
  mongod --shardsvr --port 30002 --dbpath c:\data\srx\dbxv2
  mongod --shardsvr --port 30003 --dbpath c:\data\srx\dbxv3

//Start config Server
  mongod.exe --configsvr --port 40001  --dbpath c:\data\configdbxv_2

//Using mongos start the Query Router for the config server and specify the chunk size
  mongos.exe --configdb localhost:40001 --port 40005 --chunkSize 1

 //Connect to Router
 mongo --port 40005 --host localhost
// Use admin Privileges
 use admin
// Add shards
 db.runCommand( { addshard : "localhost:30001" } );
 db.runCommand( { addshard : "localhost:30002" } );
 db.runCommand( { addshard : "localhost:30003" } );

// Enable Sharding on DB and in Collection
 db.runCommand( { enablesharding  : "person_shard_DB" } );
 db.runCommand( { shardcollection : "person_shard_DB.details_collection", key : { Name : 1 } } );

 // populate data to the DB
 use person_shard_DB
 // Insert Test Data
 for(i=0;i<50000;i++){
   db.details_collection.insert({Name:'Hello'+i, id:i+10, annual_income:100000+i})
 }
```

https://www.mongodb.com/presentations/webinar-everything-you-need-know-about-sharding?jmp=docs

https://www.slideshare.net/mongodb/everything-you-need-to-know-about-sharding

Replication

A *replica set* in MongoDB is a group of `mongod` processes that maintain the same data set.

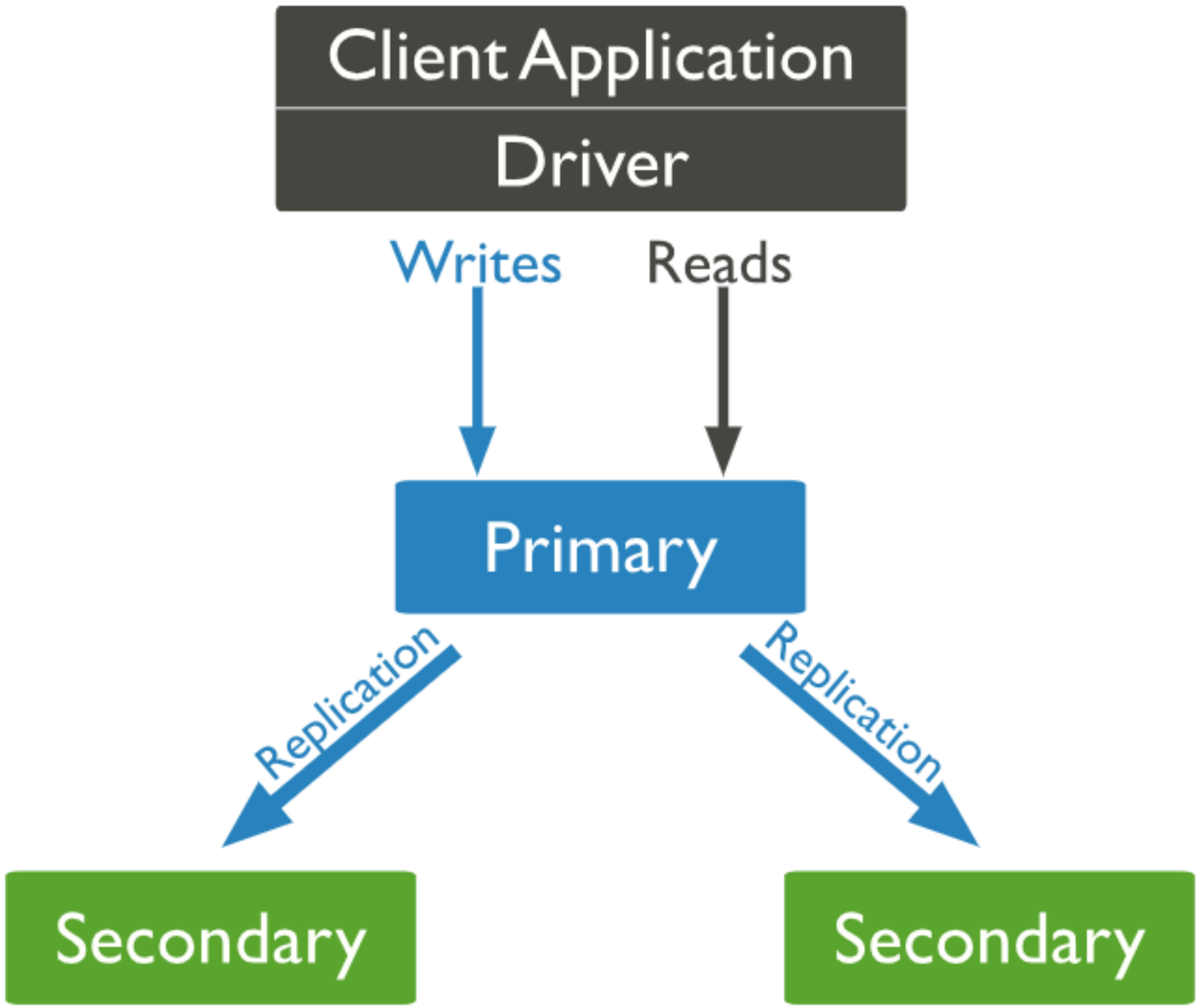
Replication provides redundancy and increases data availability.

replication can provide increased read capacity as clients can send read operations to different servers.

additional copies for dedicated purposes, such as disaster recovery, reporting, or backup.

https://docs.mongodb.com/manual/replication/
https://docs.mongodb.com/manual/administration/replica-set-deployment/

Replication



The secondaries replicate the primary's oplog and apply the operations to their data sets such that the secondaries' data sets reflect the primary's data set.

Replication

The primary is the only member in the replica set that receives write operations

All members of the replica set can accept read operations. However, by default, an application directs its read operations to the primary member.

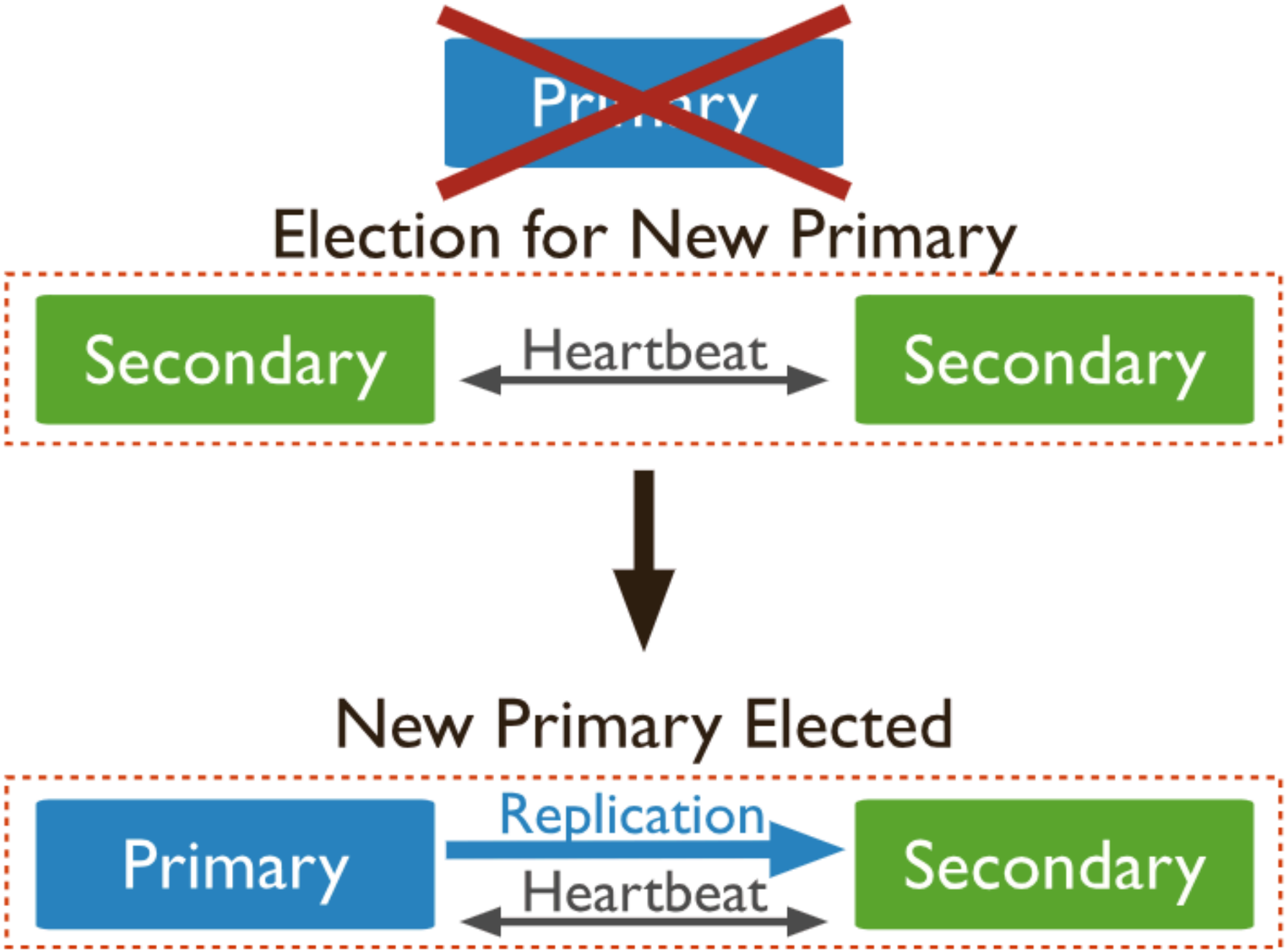
you can read from secondary replica sets but cannot write through them. only primary can write

The replica set can have at most one primary.

If the current primary becomes unavailable, an election determines the new primary.

https://docs.mongodb.com/manual/core/replica-set-primary/

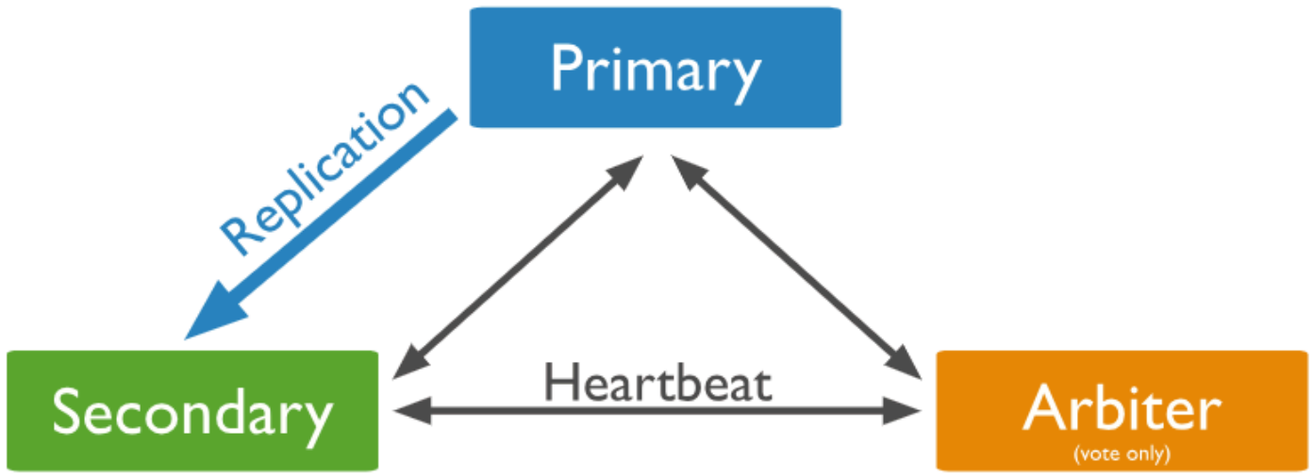
Replication



If the current primary becomes unavailable, an election determines the new primary.

Replication

An arbiter does **not** have a copy of data set and **cannot** become a primary.



Heartbeats : Replica set members send heartbeats (pings) to each other every two seconds. If a heartbeat does not return within 10 seconds, the other members mark the delinquent member as inaccessible.

Arbiters do not maintain a data set

The purpose of an arbiter is to maintain a quorum in a replica set by responding to heartbeat and election requests by other replica set members.

https://docs.mongodb.com/manual/core/replica-set-arbiter/

Replication

Replication Election

https://www.mongodb.com/presentations/replication-election-and-consensus-algorithm-refinements-for-mongodb-3-2

https://www.slideshare.net/mongodb/replication-election-and-consensus-algorithm-refinements-for-mongodb-32

https://docs.mongodb.com/manual/core/replica-set-elections/

Replication

JUST THREE SIMPLE STEPS

```
mongod –port “PORT” –dbpath “db dir” –replSet
“Replica Set Name”

rs.initiate()

rs.add(HOST:PORT)
```

Schema Design

Terminology

RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Index	→	Index
Row	→	Document
Join	→	Embedding & Linking

What is a Document?

```
{
     _id: “123”,
     title: "MongoDB: The Definitive Guide",
     authors: [
        { _id: "kchodorow", name: "Kristina Chodorow“ },
        { _id: "mdirold", name: “Mike Dirolf“ }
     ],
     published_date: ISODate("2010-09-24"),
     pages: 216,
     language: "English",
     thumbnail: BinData(0,"AREhMQ…==")
     publisher: {
          name: "O’Reilly Media",
          founded: 1980,
          locations: ["CA”, "NY” ]
     }
}
```

Documents Map to Language Constructs

```
// Java: maps
DBObject query = new BasicDBObject("publisher.founded", 1980));
Map m = collection.findOne(query);
Date pubDate = (Date)m.get("published_date"); // java.util.Date


// Javascript: objects
m = collection.findOne({"publisher.founded" : 1980});
pubDate = m.published_date;     // ISODate
year = pubDate.getUTCFullYear();


# Python: dictionaries
m = coll.find_one({"publisher.founded" : 1980 });
pubDate = m["pubDate"].year    # datetime.datetime
```

Map Reduce

https://en.wikipedia.org/wiki/MapReduce

https://www.ibm.com/analytics/hadoop/mapreduce

Map reduce is a programming model

Programming framework for processing distributed data

https://docs.mongodb.com/manual/core/map-reduce/

https://docs.mongodb.com/manual/reference/method/db.collection.mapReduce/#db.collection.mapReduce

Map Reduce

https://odetocode.com/blogs/scott/archive/2012/03/19/a-simple-mapreduce-with-mongodb-and-c.aspx

REGEX

```
{ <field>: /pattern/<options> }
```

```
{ <field>: { $regex: /pattern/, $options: '<options>' } }
{ <field>: { $regex: 'pattern', $options: '<options>' } }
{ <field>: { $regex: /pattern/<options> } }
```

https://docs.mongodb.com/manual/reference/operator/query/regex/index.html

Indexing

Indexing

```
Index are the base for high performance find Queries.

Without indexes mongodb queries through all the documents

Index increases storage amount but completely reduces the query execution time.

Indexes will affect insert and update operation speediness.

Default _id Index
MongoDB creates a unique index on the _id field during the creation of a collection.You can delete this.
```

```
There are around 5 types of indexes in mongodb

  1. Single field index
  2. Compound index  -- Multiple field indexes
  3. MultiKey index  -- indexig arrays of values
  4. Text indexes    -- for text search
  5. GeoSpatial indexes
```

```
getindexes()    -- Lists all the indexes
ensureindex() / createindex() --- create new index
dropindex() / dropindexes()   ---  remove index
reindex() --- Rebuild or modify indexes

What I love is Partial Indexes
```

Indexing

List all the Indexes in a Collection

Search without indexing

Search with indexing

Check the default _id index

drop index

https://docs.mongodb.com/manual/tutorial/analyze-query-plan/

TTL Index

TTL Indexes

Time Based Automatic removal of documents

```
db.collectionName.createIndex( { "fieldname": 1 },
{ expireAfterSeconds: 3600 } )
```

Indexed field has to be of date type

Few Restrictions - _id field, compound indexes, capped collections

https://docs.mongodb.com/manual/core/index-ttl/

Capped Collection

https://docs.mongodb.com/manual/core/capped-collections/

Mongodump & Restore

Tools

- Robomongo / Robo3T
- MongodbBooster / NoSQLBooster
- MongoChef / Studio 3T (Not Free)
- NoSql Manager / Mongodbmanager
- Compass

MongoDB Limits and Thresholds

https://docs.mongodb.com/manual/reference/limits/

MongoDB Drivers

https://docs.mongodb.com/ecosystem/drivers/#mongodb-odm-object-document-mapper

C#

https://docs.mongodb.com/ecosystem/drivers/csharp/

http://mongodb.github.io/mongo-csharp-driver/?jmp=docs

http://api.mongodb.com/csharp/current/html/R_Project_CSharpDriverDocs.htm

http://mongodb.github.io/mongo-csharp-driver/2.7/

Java

http://mongodb.github.io/mongo-java-driver/?jmp=docs

Node.js

https://mongodb.github.io/node-mongodb-native/?jmp=docs

Quick Reference Cards

https://www.mongodb.com/collateral/quick-reference-cards

Studio3T

Studio 3T official youtube channel

Using SQL in Mongodb

User Roles

Adding a New User Video

Manage User Roles Video