mongoDB®

MONGO DATABASE

Rajeev Gupta
Rgupta.mtech@gmail.com
Java Trainer & Consultant

Mongodb NoSQL CONTENTS

- Introduction to NoSQL
- What is MongoDB?
- What is NoSQL?
- How is NoSQL(MongoDB) different
  - from conventional RDBMS?
- Why MongoDB is important?
- Key features of MongoDB
- How to install & run MongoDB?
- What we get in installation package?
- What is JSON
- JSON Example

- Basic SQL to MongoDB Terminology Comparison
- Major Datatypes in MongoDB
- What is ObjectID (_id)?
- Basic CRUD operations in MongoDB
- Important Query methods in MongoDB
- Important operators
- RDBMS Where clause equivalents in
  - MongoDB
- Where MongoDB is best suited?
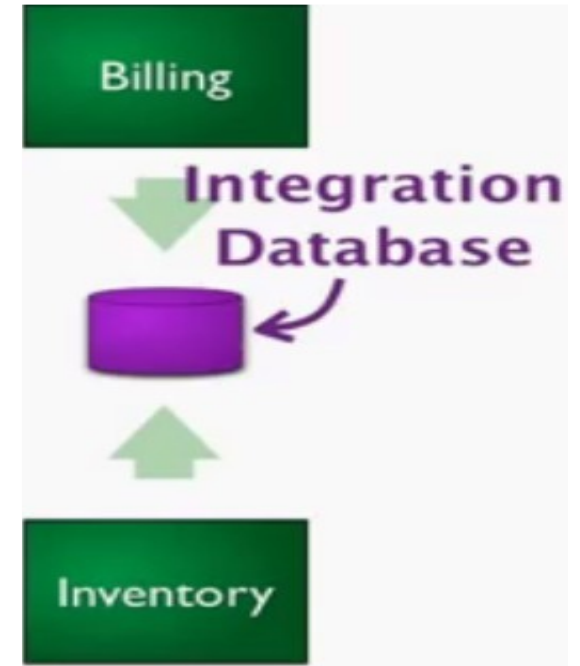- Excellent MongoDB learning references

Background

Rise of RDBMS Why

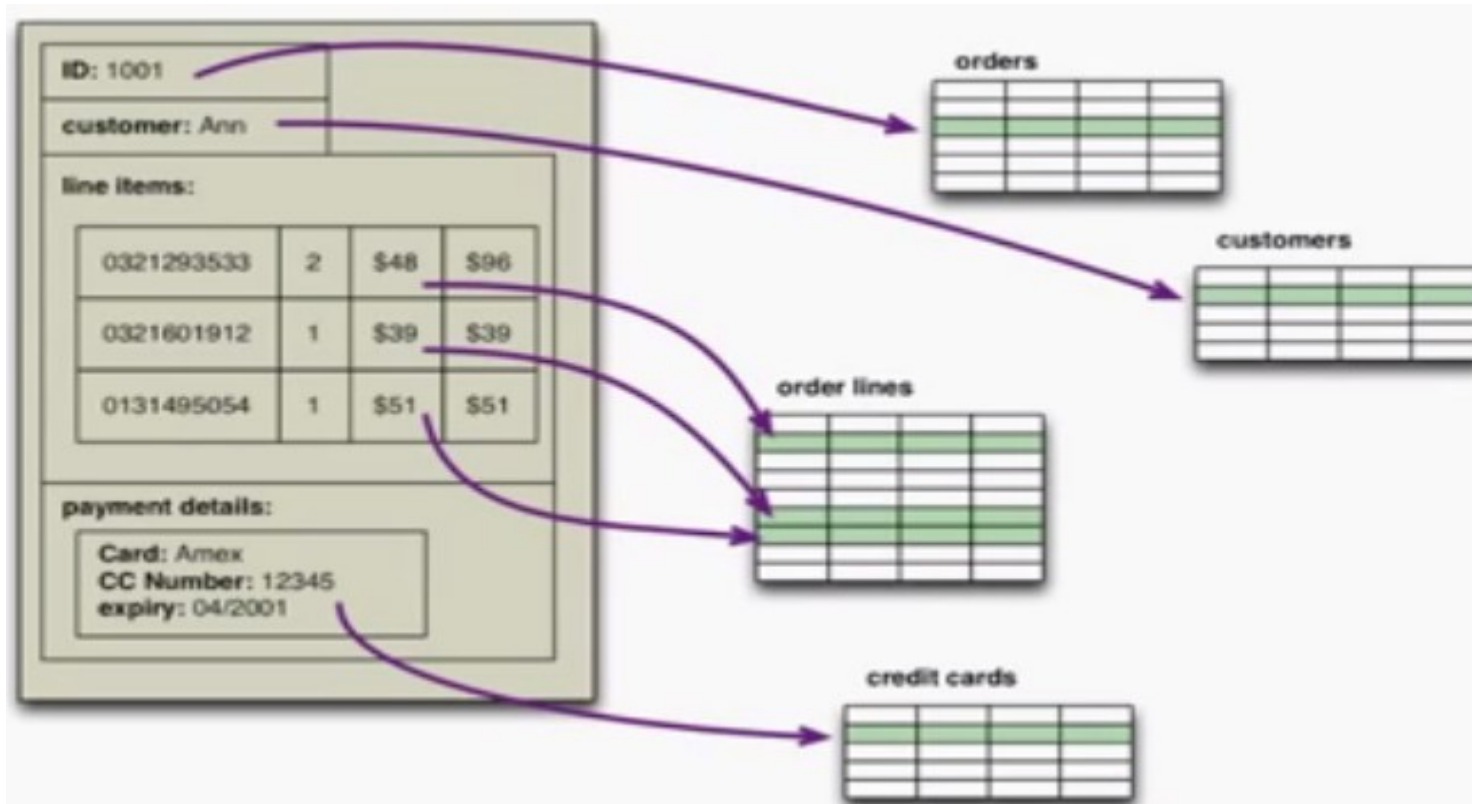Persistence, Integration, SQL, Tx Mgt, Reporting

❖ Rise of OO Database
- Don't really picked...
- RDBMS still dominant technologies

Now What?

Billing

Integration Database

Inventory

Problem with RDBMS?

- ❖ We have to manually map Requirement to different tables..

- ❖ It would be great if order can be stored as one Unit Impedance

- ❖ Mismatch=> ORM tools

Why No SQL

- ❖ Modern web application/ cloud create huge traffic Vertical Scale vs Horizontal scaling

- ❖ Horizontal scaling/ Hadoop is the solution=> RDBMS can not scale
- ❖ RDBMS is designed to run over big server rather then community H/W
- ❖ Running Relational DBMS over cluster is not easy!
- ❖ **Industry comes with new solutions**

Why NoSQL named NoSQL?

John an big data guys propose meet up
for alternative solution to RDBMS
Need twitter hashtag #nosql

Categories of NoSQL

Document MongoDB, Couch DB

Key value Riak, Redis

Column family Cassandra, Hbase

Graph Neo4j

Characteristics of NoSQL

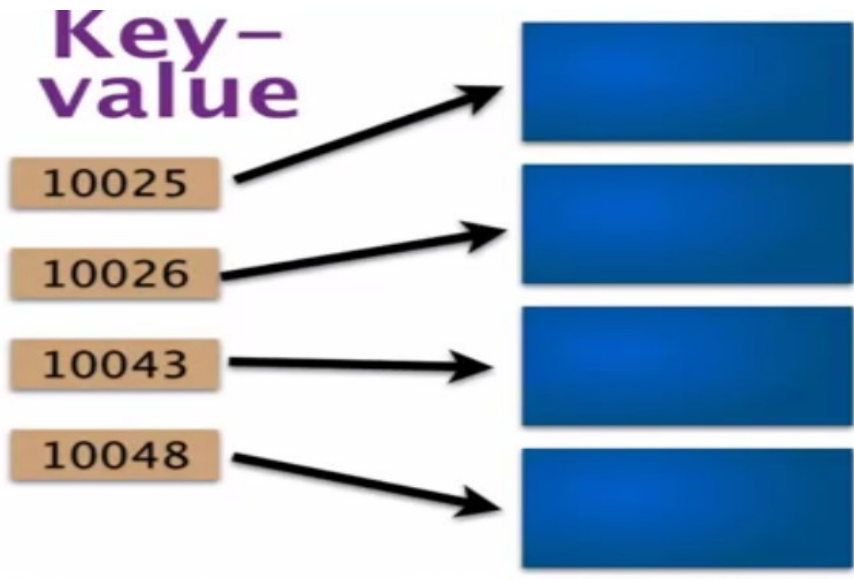Non Relational model
Open Source
Cluster friendly
21st century web
Schema less design

Key-value store

❖ Simplest model
❖ Key have no idea what is actually stored against a key, it could be doc/image/Blob data
❖ Aka hashMap Table==Bucket

Where should not be used?
- Need relationship Need Tx
- ACID
- Quaries based on value eg: price>=200

 Use cases

Document Data Model

Document

```
{"id": 1001,
"customer_id": 7231,
"line-itmes": [
{"product_id": 4555, "quantity": 8},
{"product_id": 7655, "quantity": 4}, {"product_id": 8755,

{"id": 1002,
"customer_id": 9831,
"line-itmes": [
{"product_id": 4555, "quantity": 3},
{"product_id": 2155, "quantity": 4}],
"discount-code": "Y"}
```

no schema

- ❖ Key -> Document (JSON) Flexible
- ❖ Can quarries to doc
- ❖ More transparent then key- value
- ❖ Provide implicit schema

```
anOrder["price"] * anOrder["quantity"]
```

implicit schema
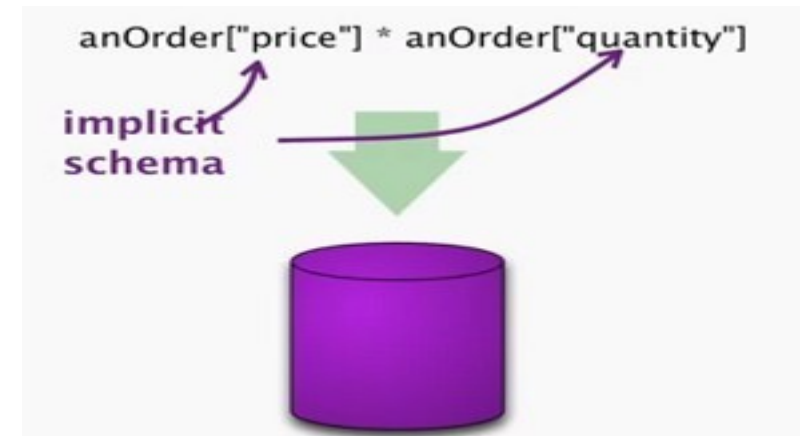
Key value vs. Document
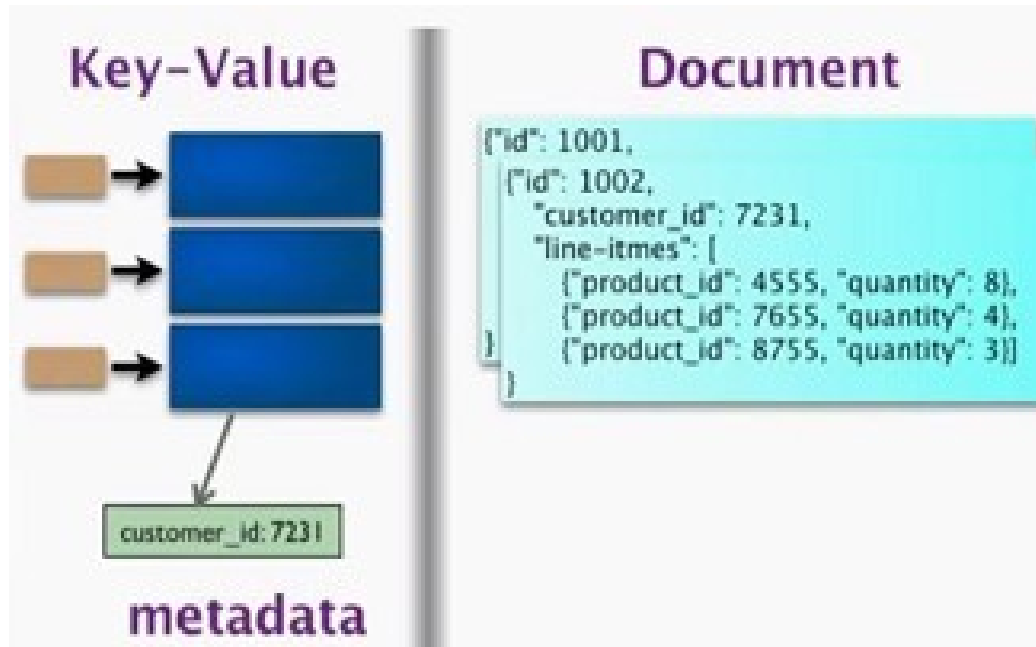
- Difference is very small Many
- key value database allowed use to have meta data like customer_id=7321 Index is allowed
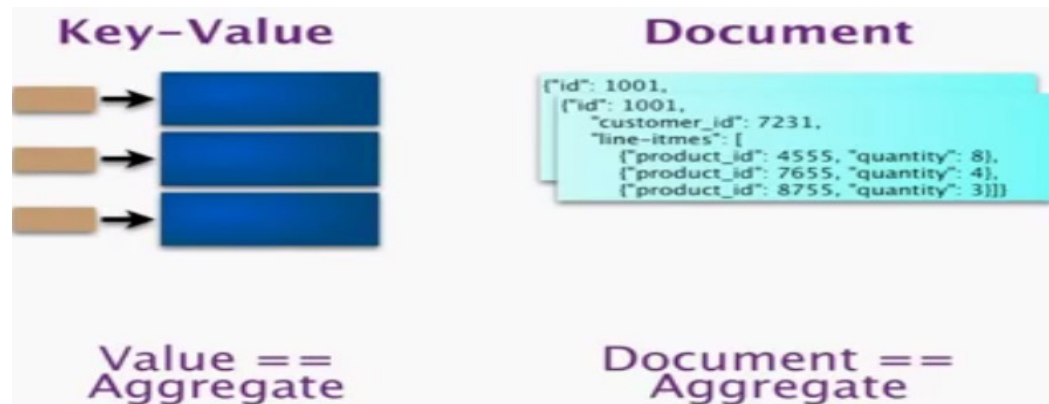
Collectively called Aggregate oriented database

Aggregate database

Order is consider as a single unit that should be store in DB
Very useful in case of distributed cluster: Each aggregate can be stored in an node
without join etc, Only one node need to refer to access one order details.

Column family DB
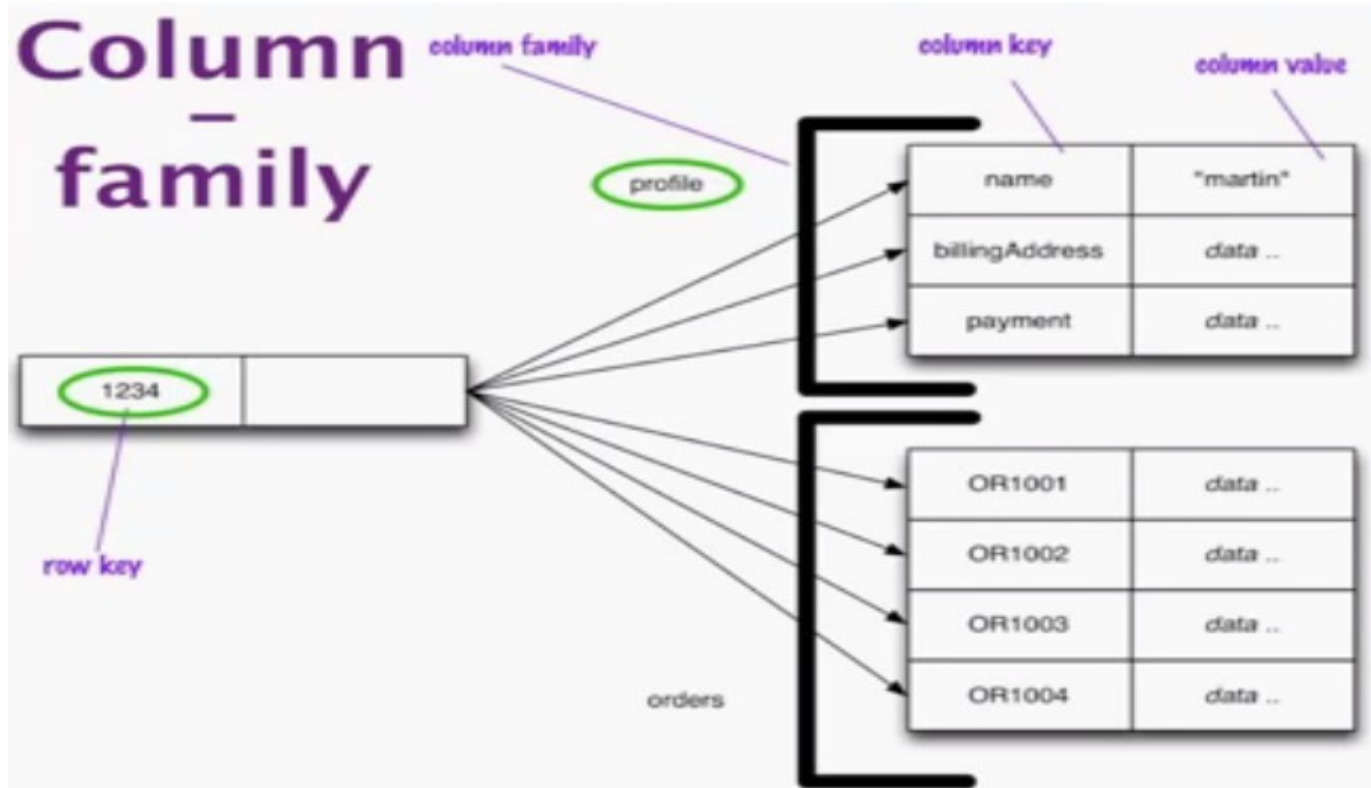
Aka Aggregate oriented DB
More Complicated then earlier model
Within one "Row key" we can store multiple column family, where each column family is combination of column that fit togather
Row Key+ column family name☐ aggregate
Easily retrieval of infidel elements

RDBMS vs Column db

Employee_ID	Job	Dept	City
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Employee_ID	Job	Dept	City
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

INFOBRIGHT

Data stored in rows

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Data stored in columns

1		Shipping		Operations		Toronto
2		Receiving		Operations		Toronto
3		Accounting		Finance		Boston

Employee_ID	Job	Dept	City
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

INFOBRIGHT

Data stored in rows

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Data stored in columns

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

ID	job	dept	city
#			
#			
#			
#			
#			
#			
#			

Row-Based Storage

ID   job   dept   city

Row Oriented works if...

- Transactional processing
- All the columns are required

Column-Based Storage

id   **job**   dept   **city**

Column Oriented works if...

- Analytical reporting
- Only relevant columns are required
- Reports are aggregates (sum, count, average, etc.)

WHAT IS MONGODB?

❖ MongoDB name comes from HuMONGOus data.

❖ According to the official website of MongoDB it is defined in the following way –

❖ *MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need.*

❖ It is a NoSQL Database.

❖ It is Open source and free to use.

❖ It is a cross-platform database which works with almost every platform (Windows, mac, Linux) .

❖ It stores data in flexible, JSON like documents.

WHAT IS NOSQL?

The mechanism of storing and retrieving the data without using the conventional way of storage i.e., inside table.

The storage mechanism adapted here is the non relational form of data.

NoSQL databases are also sometimes referred as "**Not Only SQL**"

Structured Query Language (SQL) adapts the conventional methods of having relations and some proper schema but NoSQL is completely different and it follows no schema.

NoSQL databases are divided in the following parts:

Document databases: used to store semi-structured data as documents (MongoDB, ArangoDB, RethinkDB, CouchDB). Data is represented as a JSON document.

Graph stores: used to store the information about the network of the data (Neo4J, GraphBase,WhiteDB)

Key-value stores: Data is stored in the form of key-value pairs. They are the simplest NoSQL
Databases (Redis, DynamoDB, Riak, Berkeley DB).

Wide-column stores: It uses tables, rows and columns but the names and formats of the columns can vary from row to row in the same table (Cassandra and HBase).

HOW NOSQL IS DIFFERENT FROM CONVENTIONAL RDBMS?

SQL databases are table based databases whereas NoSQL databases are document based, key-value pairs, graph databases or wide-column stores.

SQL databases have predefined schema whereas NoSQL databases have dynamic schema for   unstructured data.

SQL databases are vertically scalable whereas the NoSQL databases are horizontally scalable.

SQL databases are not best suited for hierarchal data storage whereas NoSQL is best suited for it.

	RDBMS (SQL)	NoSQL
Base	Table (rows and columns)	Document, key-value pairs, graph and wide-column stores
Schema	Predefined	Dynamic
Scalability	Vertical	Horizontal
Category	Relational	Non-relational or Distributed
Big Data Preference	Preferred	Not preferred
Complex Query Preference	Preferred	Not preferred
Hierarchical Data Storage Preference	Not preferred	Preferred

WHY MONGODB IS IMPORTANT ?

MongoDB is important in many aspects and some of those major points are as follows:

It is document oriented and does not need the row and column format of the data.

It gives high performance.

It is dynamic in nature where you don't need to predefine a schema like in conventional RDBMS.

There is no concept of Joins in MongoDB instead it has **$lookup** aggregation operator in versions >=3.2. There is a mongoose alternative as well which is called as **populate()** that is used to reference documents in other collections.

MongoDB stores data in JSON format which allows to send the data in whatever form you want.

KEY FEATURES OF MONGODB ?

Data is stored in **BSON** and presented in **JSON**.

Server-side JavaScript is supported (JavaScript expressions and functions).

Document oriented database where there are no tables and no row-based data.

NoSQL, where there is no schema. Documents can have different structures. Documents can be embedded. This is specifically designed for horizontal scaling.
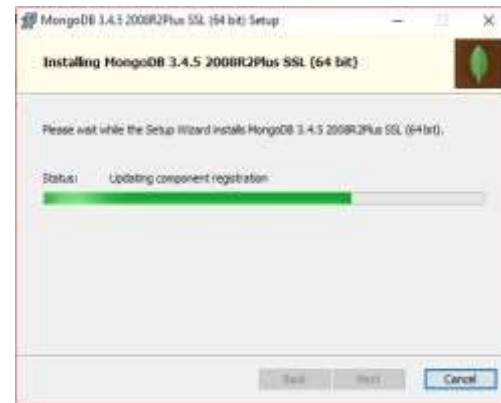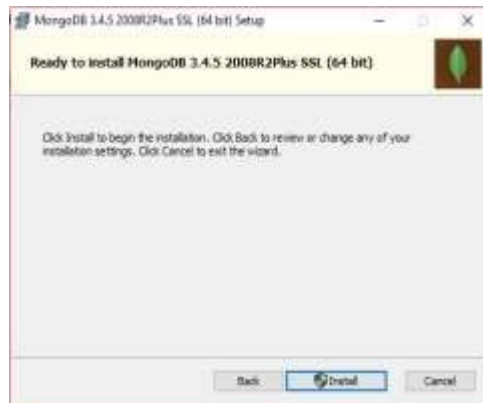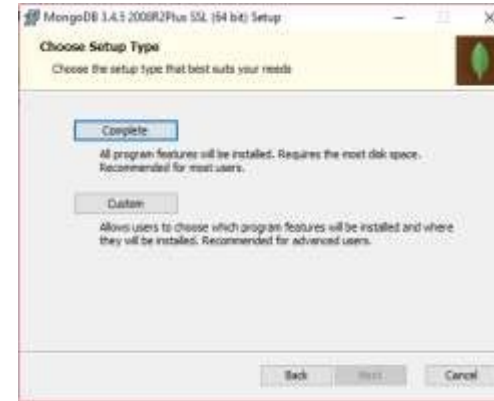
By default, there is a **primary key (_id)**, which is an auto-generated field for every document.

Sharding is supported which is very much essential for horizontal scaling and replication.

It has automatic **load balancing** configurations.

Fault Tolerance is natively built in MongoDB.

HOW TO INSTALL & RUN MONGODB ?

HOW TO INSTALL & RUN MONGODB ?

To install the MongoDB you need to visit the download section of the official website. Visit the following link:https ://www.mongodb.com/download-center/community.

Select the version, OS and the package there and then click download.

After download is completed double click the file (e.g.:- mongodb-win32-x86_64- 2008plus-ssl-3.4.5-signed.msi).
This will start the installation.

Go through all the steps. Simply click next on each step and at the last step click on

Finish button. This will install the MongoDB on your system.
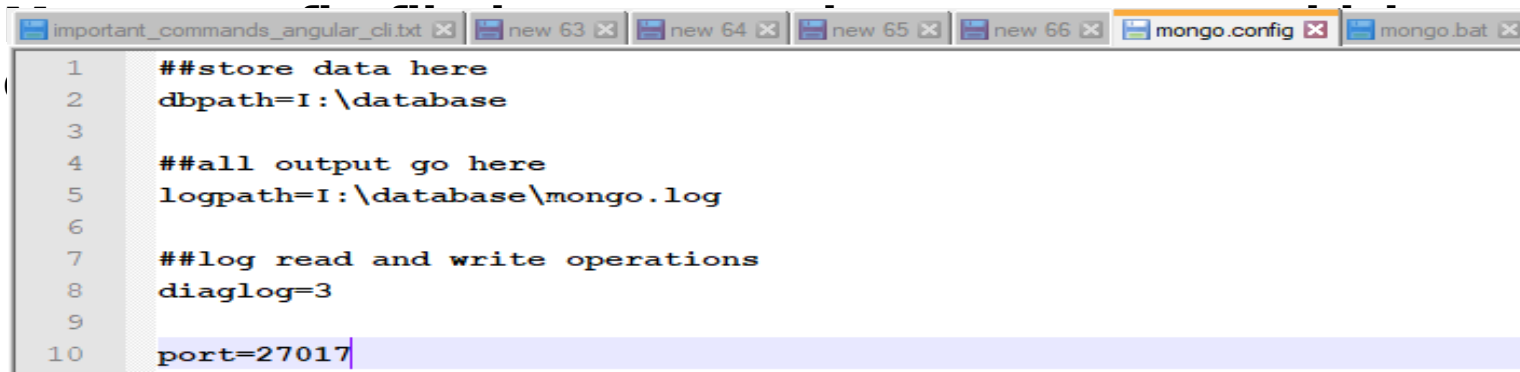
RUNNING THE MONGODB IN SYSTEM:

Create the folder 🡪 C:\data\db (In this case after creating the folder, you need to double click the mongod.exe file or just need to start the command line terminal and run the command mongod).

HOW TO INSTALL & RUN MONGODB ?

...in continuation

Alternatively you can create the folder at a different location and give its reference in mongo.config file. Then in mongo.bat file write the reference of the mongo.config file and run it whenever you want to start the mongodb server.


```
1   ##store data here
2   dbpath=I:\database
3
4   ##all output go here
5   logpath=I:\database\mongo.log
6
7   ##log read and write operations
8   diaglog=3
9
10  port=27017
```
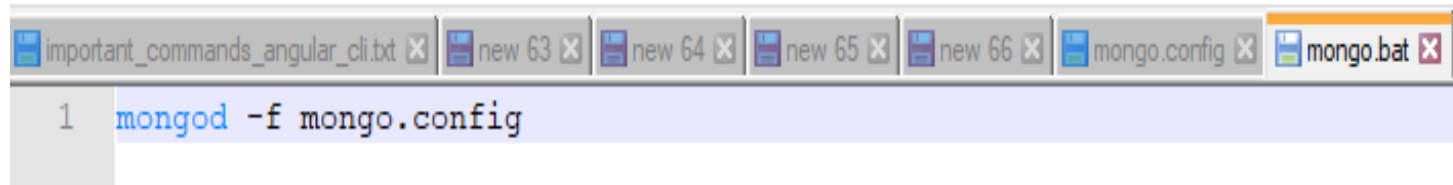
It comprises database path reference, log file path reference and port number reference.

HOW TO INSTALL & RUN MONGODB ?

...in continuation

The mongo.bat file will have a reference to mongo.config file.

The command written inside the mongo.bat file is:

This will allow the server to read the config file and refer the location of the database.

The default port that the mongodb server reads is 27017.

WHAT IS THERE IN INSTALLATION PACKAGE ?

MongoDB comes with handy tools which are bundled together at a single place.
Those tools are as follows:

mongod : Mongo demon which is the server.

mongo : It is a CLI based client.

mongos : It is a mongodb sharing routing service(Used for sharding).

mongostat : It is used for monitoring the Quick Overview of the running instance.

mongotop : It is used for monitoring the Read and Write of an instance.

NOTE:

To start the server we just need the mongod file.

For the ease of learning and exploring MongoDB, you can install any MongoDB GUI tool at this point (Robo 3T, QueryAssist, NoSQLBooster, Studio 3T, MongoDB Compass, Mongo Management Studio, NoSQL Manager, NoSQL Client, Navicat for MongoDB)

WHAT IS THERE IN INSTALLATION PACKAGE ?

…in continuation

mongorestore: It loads data from the Binary DB dump.

bsondump & mongodump: It creates binary dump.

mongooplog: It is used for the purpose of the replication from the oplog.

mongoexport & mongoimport: It is used to export or import files(generally JSON & CSV).

mongoperf: It is used to check the Disk IO.

mongofiles: It is basically used for GridFS functionality.

WHAT IS JSON?

Before proceeding any further you must know what is JSON as MongoDB data is always represented in the JSON format. So, here are some of the essential points related to JSON.

JSON stands for *JavaScript Object Notation*. It is a collection of key-value pairs. Keys must be written inside single or double quotes. Values can have any data type (Array, string, Boolean etc.)

2. JSON file must start and end with a curly brace.

Collection of key-value pair ☐ Various languages recognize it as an object,
record, struct, dictionary, hash table, keyed list, or associative array.

Object is surrounded by the curly braces and key-value pairs are separated by comma.

Array is surrounded by square braces and each value is separated with a comma.

Value can be of any type ☐ string, number, object, array, true, false, null

JSON EXAMPLE

```
{
    "firstName": "John",
    "lastName": "Doe",  "age": 25,
    "address": {
        "streetAddress": "221B Bakers Street",  "city": "London",
        "state": "ENG",  "postalCode": "200001"
    },
}
```

Notice here the key-value pair form of data that is embedded inside the curly braces.

BASIC SQL TO MONGODB TERMINOLOGY COMPARISON

Terminology & Concept Comparison:

SQL	MongoDB
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	$lookup, embedded documents
primary key (need to specify column or combination of column)	primary key (automatically set to _id field)
aggregation	aggregation pipeline

Database Executables

Comparison	MongoDB	MySQL	Oracle	Informix	DB2
Database Server	mongod	mysqld	oracle	IDS	DB2 Server
Database Client	mongo	mysql	sqlplus	DB-Access	DB2 Client

MAJOR DATATYPES IN MONGODB ?

String

Integer

Boolean

Double

Min/Max keys

Arrays

Timestamp

Object

Null

Symbol

Date

ObjectID

Binary data

Code

Regular
expression

WHAT IS objectId (_id) ?

ObjectId in the MongoDB is same as the primary key in the conventional RDBMS.

It is by default set by MongoDB for every document that is created inside any collection.

ObjectIds are small, unique, fast to generate and ordered.

ObjectId values comprises of a 12 bytes hexadecimal number which is unique
for every document.

_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id,
3 bytes counter)

"_id":
ObjectId("5901832c91427cac52e9ea
8f")

BASIC CRUD OPERATIONS

The basic operations in every database comprises of the following four operations:
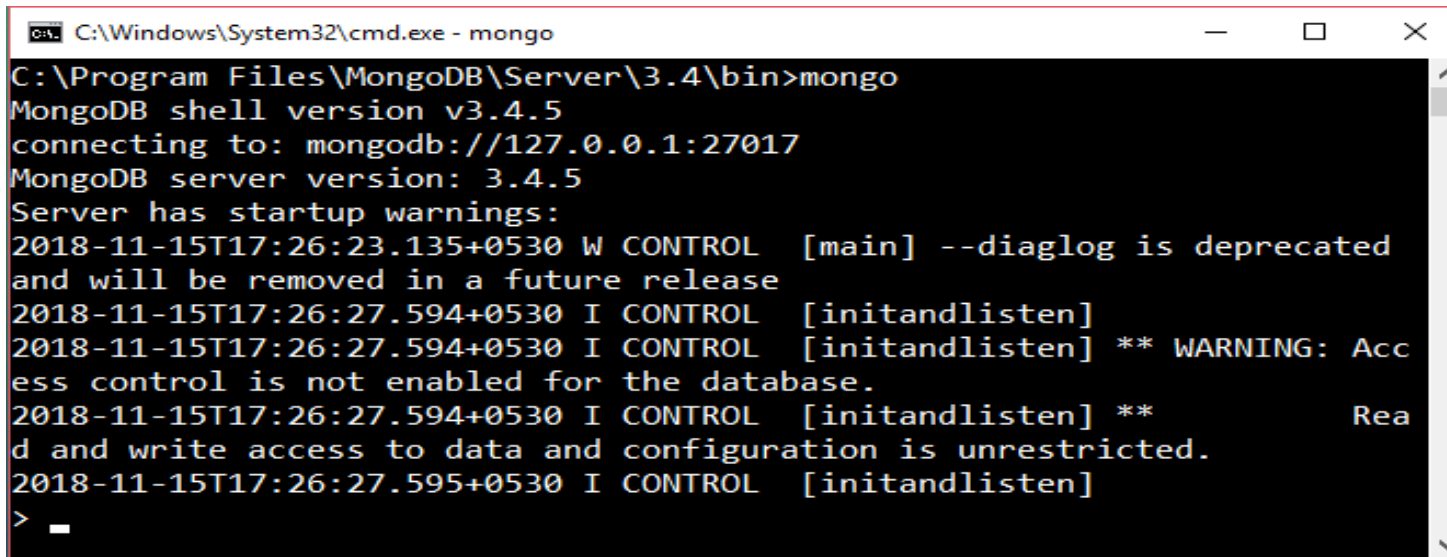
CRUD

In MongoDB these four basic operations are achieved using the unique style of writing. But before actually starting with the data, you must first create a database.

In MongoDB, database is created using the **use** command.

BASIC CRUD OPERATIONS

STARTING THE MONGODB INSTANCE IN THE TERMINAL

1. **Go to the terminal and give the command mongo** there so as to start the MongoDB CLI.

This will start the mongo instance in the terminal from where you can work on mongodb and fire queries.

BASIC CRUD OPERATIONS

…in continuation

CREATING THE DATABASE USING THE use COMMAND

Go to the terminal and write the command *use <database_name>* where database name will be replaced by the name of your database. This will create the database with the name which you want.


```
C:\Windows\System32\cmd.exe - mongo                    —   □   ×

> use habileMongo
switched to db habileMongo
>
```

Alternatively it can be achieved by using any of the GUI tool and from there you can create the new database. But this GUI tool will first need to be connected to the host at the desired port number.

BASIC CRUD OPERATIONS
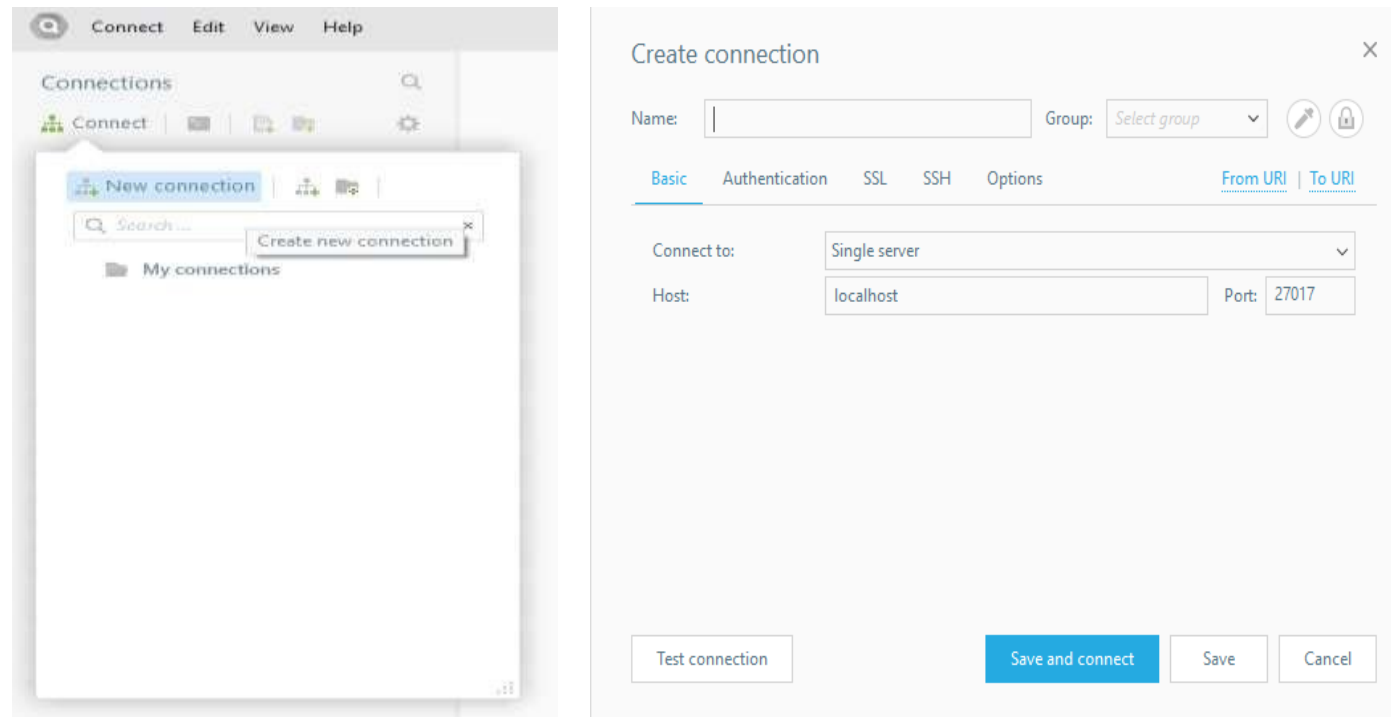
The GUI tool which is shown above is **QueryAssist You can Give a name to the instance from the tool.**

BASIC CRUD OPERATIONS
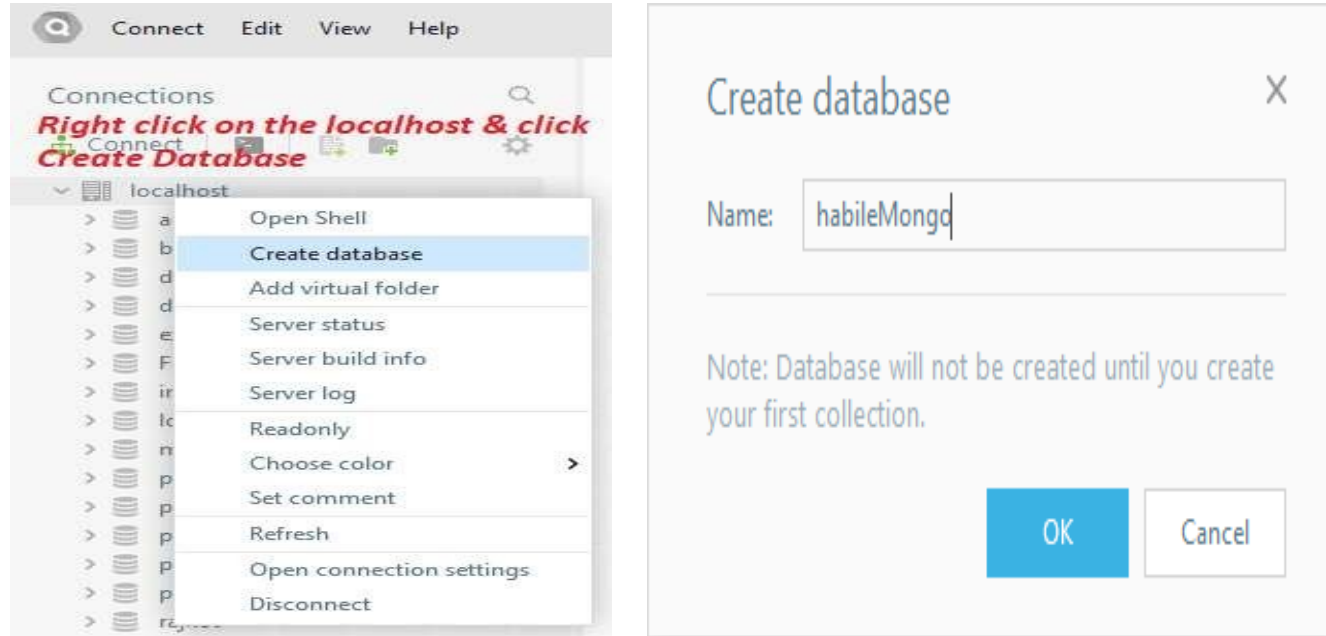
By this way you can create the database in MongoDB.

Observe the note written in the Create Database Pop Up

BASIC CRUD OPERATIONS

NOTE: You can ensure that your database has been created or not by using the ***show databases*** or ***show dbs*** command. The database will only appear when you have at least one collection inside it.

To create a collection inside the recently created database, you must use the insert command:

db.employee.insert({'name': 'John Doe', 'empId': 'HL0014'});

Now when you hit the show dbs or show databases command, it will show your database.


```
C:\Windows\System32\cmd.exe - mongo
> show databases;
FMC                 0.203GB
accenture           0.078GB
bartender           0.078GB
demo                0.078GB
demoCURD            0.078GB
examsbook           0.203GB
habileMongo         0.078GB
initialExpress      0.078GB
local               0.078GB
mean-passport       0.078GB
payment             0.078GB
pcat                0.078GB
practiseMongoDB     0.078GB
practiseMongoDB2    0.078GB
practiseMongoDB3    0.078GB
rajkot              0.078GB
register            0.078GB
vendorClient        0.078GB
weddingClient       0.078GB
wordCupDictionary   3.952GB
> _
```

BASIC CRUD OPERATIONS

DROPPING A DATABASE

To drop a database in MongoDB you need to type the following command.

db.dropDatabase()

This will drop the selected database. If database is not selected then it will delete the default 'test' database.

Now you can check through **show dbs** or **show databases** command, that your created database is dropped.

You can clearly see in the picture that the **habileMongo** database is now not listed. It was dropped using the command above.


```
C:\Windows\System32\cmd.exe - mongo

> db.dropDatabase()
{ "dropped" : "habileMongo", "ok" : 1 }
> show dbs;
FMC                  0.203GB
accenture            0.078GB
bartender            0.078GB
demo                 0.078GB
demoCURD             0.078GB
examsbook            0.203GB
initialExpress       0.078GB
local                0.078GB
mean-passport        0.078GB
payment              0.078GB
pcat                 0.078GB
practiseMongoDB      0.078GB
practiseMongoDB2     0.078GB
practiseMongoDB3     0.078GB
rajkot               0.078GB
register             0.078GB
vendorClient         0.078GB
weddingClient        0.078GB
wordCupDictionary    3.952GB
>
```

BASIC CRUD OPERATIONS

For previous operation's result to be visible, one must first create a
collection inside it. So, let's start with CRUD operations.

CREATE

For creating a collection use the following command

db.createCollection(name, options)

Parameter	Type	Description
Name	String	It is the name of the collection that is to be created
Options	Document	Specifies option about the memory size and indexing. This is completely optional.

BASIC CRUD OPERATIONS

Options can have the
following fields:

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify size parameter also.
autoIndexI d	Boolean	(Optional) If true, automatically create index on _id fields. Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. If capped is true, then you need to specify this field also.
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

BASIC CRUD OPERATIONS

EXAMPLE:

1. Without Options parameter –

> *db.createCollection('Employees');*


```
> db.createCollection('Employees');
{ "ok" : 1 }
>
```

2. With Options parameter

db.createCollection('Employees', {capped: true, autoIndexId: true, size: 6142800, max: 10000});


```
> db.createCollection('Employees', {capped: true, autoIndexId: true, size: 6142800, max: 1000});
{
    "note" : "the autoIndexId option is deprecated and will be removed in a future release",
    "ok" : 1
}
>
```

BASIC CRUD OPERATIONS

INSERT DOCUMENT

For inserting the document in MongoDB collection, you need to write the following command:

> ### db.collection_name.insert(document)

Where in place of collection name you can place your collection name.

> ### db.Employees.insert({'name': 'John Doe', 'empId': 'HL0014'})


```
C:\Windows\System32\cmd.exe - mongo
> db.Employees.insert({'name': 'John Doe', 'empId': 'HL0014'});
WriteResult({ "nInserted" : 1 })
>
```

INSERT DOCUMENT

To insert multiple documents inside a single query, you must pass an
array of documents in the insert() command:

db.Employees.insert([{'name': 'John Smith', 'empId': 'HL0015'},{'name': 'Dwane

```
> db.Employees.insert([{'name': 'John Smith', 'empId': 'HL0015'},{'name': 'Dwane Simmons', 'empId': 'HL0016'}]);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

BASIC CRUD OPERATIONS

UPDATE

For updating any document, an **update()** or **save()** method is used:

> ***db.Employees.update({'name': 'John Smith'},***
> ***{$set:{'name': 'John Smith***
> ***Carlos'}});***

> ***NOTE: By default mongodb will update only a single document. To update multiple documents , there is a need to set the 'multi' parameter to true.***

> ***db.Employees.update({'name': 'John Smith'},{$set:***
> ***{'name': 'John Smith***
> ***Carlos'}}, {multi: true});***

BASIC CRUD OPERATIONS

UPDATE

Using save() method:

Save method will replace the existing document with the new document   passed in the save() method.

> **db.Employees.save({"_id" :**
> **ObjectId("5bef7ca23a929140015e491f"),'n**
> **ame': 'John Smith Carlos',**


```
C:\Windows\System32\cmd.exe - mongo
> db.Employees.save({"_id" : ObjectId("5bef7ca23a929140015e491f"),'name': 'John Smith Carlos', 'empId':'HL0017'});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Employees.find({});
{ "_id" : ObjectId("5bef7ca23a929140015e491e"), "name" : "John Doe", "empId" : "HL0014" }
{ "_id" : ObjectId("5bef7ca23a929140015e491f"), "name" : "John Smith Carlos", "empId" : "HL0017" }
{ "_id" : ObjectId("5bef7ca23a929140015e4920"), "name" : "Dwane Simmons", "empId" : "HL0016" }
>
```

NOTE: If Save method is used without _id, then it will update the document else it will insert a document.

BASIC CRUD OPERATIONS
DELETE

To delete a document , MongoDB has **remove()**
method.

```
C:\Windows\System32\cmd.exe - mongo
> db.Employees.remove({'name': 'John Smith Carlos'});
WriteResult({ "nRemoved" : 1 })
> db.Employees.find({});
{ "_id" : ObjectId("5bef7ca23a929140015e491e"), "name" : "John Doe", "empId" : "HL0014" }
{ "_id" : ObjectId("5bef7ca23a929140015e4920"), "name" : "Dwane Simmons", "empId" : "HL0016" }
>
```

This will remove all the documents matching with the
conditions.

To remove only one matching document there is **deleteOne()**
method or
remove() method with **justOne** parameter.

BASIC CRUD OPERATIONS

DELETE

To delete a document , MongoDB has **remove()** method.

> *db.Employees.deleteOne({'name': 'John Smith Carlos'});*


```
C:\Windows\System32\cmd.exe - mongo
> db.Employees.save({'name': 'John Smith Carlos', 'empId':'HL0017'});
WriteResult({ "nInserted" : 1 })
> db.Employees.save({'name': 'John Smith Carlos', 'empId':'HL0018'});
WriteResult({ "nInserted" : 1 })
> db.Employees.save({'name': 'John Smith Carlos', 'empId':'HL0019'});
WriteResult({ "nInserted" : 1 })
> db.Employees.find({});
{ "_id" : ObjectId("5bef7ca23a929140015e491e"), "name" : "John Doe", "empId" : "HL0014" }
{ "_id" : ObjectId("5bef7ca23a929140015e4920"), "name" : "Dwane Simmons", "empId" : "HL0016" }
{ "_id" : ObjectId("5befa1eb3a929140015e4923"), "name" : "John Smith Carlos", "empId" : "HL0017" }
{ "_id" : ObjectId("5befa1f63a929140015e4924"), "name" : "John Smith Carlos", "empId" : "HL0018" }
{ "_id" : ObjectId("5befa2043a929140015e4925"), "name" : "John Smith Carlos", "empId" : "HL0019" }
> db.Employees.deleteOne({'name': 'John Smith Carlos'});
{ "acknowledged" : true, "deletedCount" : 1 }
> db.Employees.find({});
{ "_id" : ObjectId("5bef7ca23a929140015e491e"), "name" : "John Doe", "empId" : "HL0014" }
{ "_id" : ObjectId("5bef7ca23a929140015e4920"), "name" : "Dwane Simmons", "empId" : "HL0016" }
{ "_id" : ObjectId("5befa1f63a929140015e4924"), "name" : "John Smith Carlos", "empId" : "HL0018" }
{ "_id" : ObjectId("5befa2043a929140015e4925"), "name" : "John Smith Carlos", "empId" : "HL0019" }
>
```

The other way to delete single document is : use the findOne() method first to find the matching first document and then use remove() method.

BASIC CRUD OPERATIONS
DELETE

```
C:\Windows\System32\cmd.exe - mongo
> var item = db.Employees.findOne({'name': 'John Smith Carlos'});
> db.Employees.remove({_id: item._id});
WriteResult({ "nRemoved" : 1 })
> db.Employees.find({});
{ "_id" : ObjectId("5bef7ca23a929140015e491e"), "name" : "John Doe", "empId" : "HL0014" }
{ "_id" : ObjectId("5bef7ca23a929140015e4920"), "name" : "Dwane Simmons", "empId" : "HL0016" }
{ "_id" : ObjectId("5befa2043a929140015e4925"), "name" : "John Smith Carlos", "empId" : "HL0019" }
>
```

This way you can remove a particular document from the collection.

findOne() method searches all the data and returns the first matching
document. You can pass the condition inside the **findOne()** method.m

NOTE: If you don't specify the deletion criteria (db.Employees.remove()), then it will delete all the documents inside the collection.

IMPORTANT PROJECTION & QUERY METHODS

find()

find() is the essential projection method which helps in selecting only the necessary data.

db.Employees.find({});

findOne()

This method returns the first matched document from the collection. You need to specify the condition inside the findOne() method.

db.Employees.findOne({'name': 'John Smith Carlos'});

pretty()

This method is used to display the text in the formatted way.

IMPORTANT PROJECTION & QUERY METHODS

distinct()

distinct() is the essential method which helps in returning   an array of documents that have distinct values for a specified field.

db.Employees.distinct({'name'});

isCapped()

This method is used to check whether that collection is capped or not.

db.Employees.isCapped();

It will return a Boolean value. If capped then true other wise false.

limit()

This method is used to limit the records in MongoDB. It accepts the number type
argument.

db.Employees.find({}).limit(2);

IMPORTANT PROJECTION & QUERY METHODS

skip()

This method is used to skip the number of documents.
It accepts the number type argument.

db.Employees.find({}).limit(1).skip(1);

sort()

This method is used to sort the documents in
MongoDB.

*db.Employees.find({}).sort({'name': -
1});*

This will sort the document in descending order on the
field **name**.

IMPORTANT OPERATORS

AND

This is the operator used to query the data with AND relation. It will search the document satisfying the conditions that you pass in the query.

*db.Employees.find({$and: [{'name': 'John Smith Carlos'},
{'EmpId': 'HL0020'}]});*

OR

This operator is used to query the data with OR relation. It will search the document and return the result if any of the specified condition is fulfilled.

db.Employees.find({$or: [{'name': 'John Smith Carlos'}, {'EmpId': 'HL0020'}]});

RDBMS WHERE CLAUSE EQUIVALENTS IN MONGODB

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.Employees.find({"name":"John Doe"}).pretty()	where name = John Doe'
Less Than	{<key>: {$lt:<value>}}	db.Employees.find({"salary": {$lt:20000}}). pretty()	where salary < 20000
Less Than Equals	{<key>: {$lte:<value>}}	db.Employees.find({"salary": {$lte:20000}}).pretty()	where salary <= 20000
Greater Than	{<key>: {$gt:<value>}}	db.Employees.find({"salary": {$gt:20000}}) .pretty()	where salary > 20000
Greater Than Equals	{<key>: {$gte:<value>}}	db.Employees.find({"salary": {$gte:20000}}).pretty()	where likes >= 20000
Not Equals	{<key>: {$ne:<value>}}	db.Employees.find({"salary": {$ne:20000}}) .pretty()	where likes != 20000

WHERE MONGODB IS BEST SUITABLE?

MongoDB is best suited for unstructured and semi-structured data such as:

- Social media posts

- Web pages

- Emails

- Medical records

- Reports

- Raw data from marketing researches

- Scientific data

EXCELLENT MONGODB LEARNING REFERENCES

SQL to MongoDB mapping Chart
https://docs.mongodb.com/manual/reference/sql-comparison/

What is NoSQL?
https://www.mongodb.com/nosql-explained

MongoDB and MySQL Comparison
https://www.mongodb.com/compare/mongodb-mysql

MongoDB Cheat Sheet
https://blog.codecentric.de/files/2012/12/MongoDB-CheatSheet-v1_0.pdf

Organized Reference card – PDF file – Dzone
https://dzone.com/refcardz/mongodb?chapter=1

…to be continued

EXCELLENT MONGODB LEARNING REFERENCES

Online Course Catalogue
https://university.mongodb.com/courses/catalog

MongoDB resources – starter kit
https://resources.mongodb.com/getting-started-with-mongodb

Brief Introduction to MongoDB by Eliot Horowitz
(CTO & Co- founder of MongoDB)
https://www.youtube.com/watch?v=EE8ZTQxa0AM

THANK YOU