Basics of NoSQL Databases - MongoDB

As per CBSE curriculum
Class 11

Chapter- 20

By-
Neha Tyagi
PGT (CS)
KV 5 Jaipur(II Shift)
Jaipur Region

Introduction

- Till now we have been working on the databases which were based on SQL consisting of table, row, fields ,records etc.
- It is possibe to have database without any structure or record. NoSQL or Not Only SQL Databases are such databases.
- We will learn NoSQL databases in this chapter.

NoSQL Databases

- These are non-relational databases which does not have any strict or rigid structure.
- These does not store records on the basis of conventional tables.
- These runs in clusters and stores data on the basis of web. These are high in scalability. These are also known as bigdata.
- You have worked on several apps/web apps using such databases like Google Mail, Google Earth, Ebay, LinkedIn, facebook, Amazon etc.
- These provides fast response time.
- These can handle data of any kind without any restriction.
- These adopts new features and fast update.
- These does not show down time.

Types of NoSQL Databases

1. Key-value Databases
2. Document Databases
3. Column family stores Databases
4. Graph Databases

Key-Value databases

- Just like python dictionary.
- Very simple and flexible.
- Examples-*Cassandra, Amazon DyanmoDB, ATS (Azure Table Storage, Riak, BerkeleyDB*

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Document Databases

- These are advanced form of key-value databases.
- Here, key-value pair stores in document in structured or semi-structured form.
- Keys are always of string type, values can be of any type.
- It can be in the form of MS office document, PDFs, XML, JSON ,BSON.
- JSON (JavaScript Object Notation) and BSON (Binary JSON)
- JSON is an open, human & machine understandable standard. Its main format to interchange data on modern web is XML.
- We have learnt use of JSON in Python dictionaries.
- Its examples are - MongoDB, Couch DB DocumentDB etc.

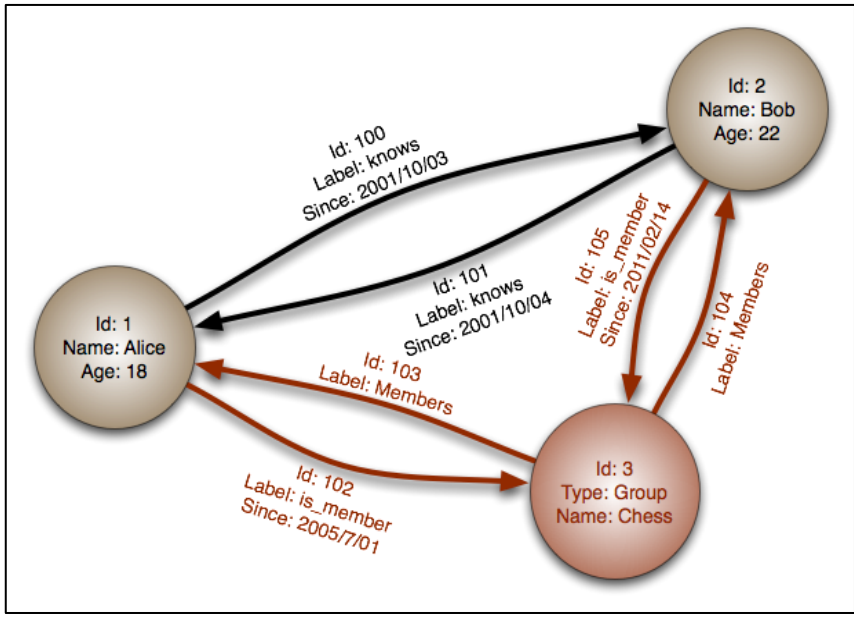
```
{
    "Title": "The Cuckoo's Calling",
    "Author": "Robert Galbraith",
    "Genre": "classic crime novel",
    "Detail": {
        "Publisher": "Little Brown",
        "Publication_Year": 2013,
        "ISBN-13": 9781408704004,
        "Language": "English",
        "Pages": 494
    },
    "Price": [
        {
            "type": "Hardcover",
            "price": 16.65
        },
        {
            "type": "Kidle Edition",
            "price": 7.03
        }
    ]
}
```

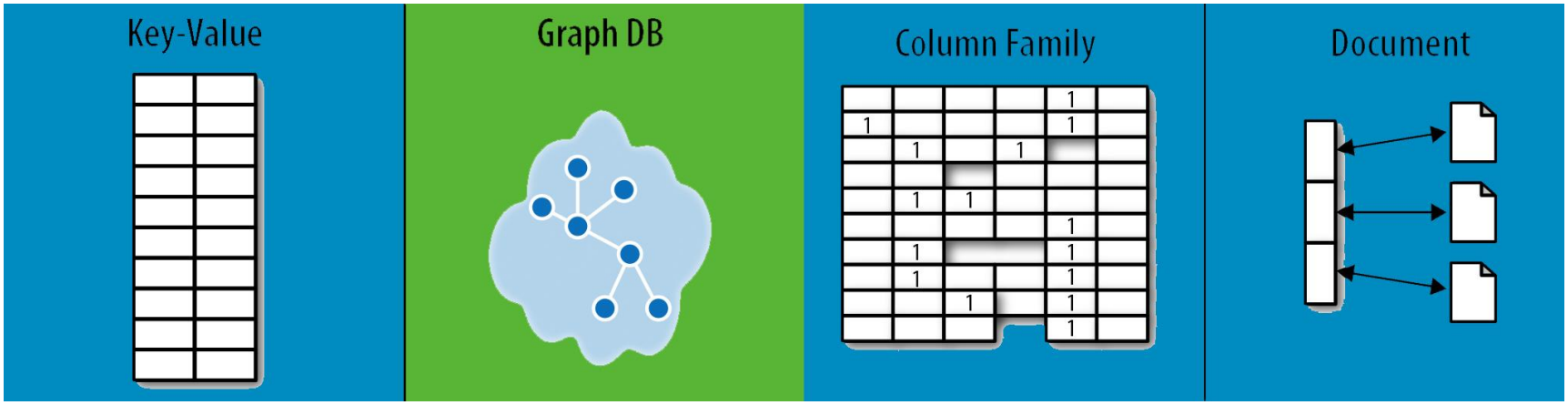
Column Family Store Database

- These are known as column store or column family databases and these are column oriented models.
- Column family is a storage mechanism which can –
  - Have multiple rows.
  - Each row can have multiple columns.
  - In this, there is a row key under which there can be multiple columns as shown in the figure.
  - Its examples are- Hbase, Cassandra, HyperTable etc.

Graph Database

- It uses graphical model to store data.
- Here, **nodes** are used to show object whereas **edges** are used to show relation between those nodes.
- Its examples are- Neo4j, Blazegraph, Titan etc.





Advantages and Disadvantages of NoSQL Databases

•Advantages:

–Flexible Data Model

These are very flexible database which can store any kind of data.

–Evolving Data Model

You can change its schema without downing the system.

–Elastic Scalability

Huge database can be stored on a very less cost.

–High Performance

Time of throughput and latency is very less.

–Open Source

It is available free of cost and you can change it as per yor requirement.

•Disadvantages:

–Lack of Standardization

No standard rules are there for NoSQL database.

–Backup of Database

Main problem with NoSQL databases is of backup. MongoDB provides tool for backup but it is also not up to the mark.

–Consistency

NoSQL database does not think about consistency. Means here, you can have duplicate data very easily.

Working with MongoDB

- MongoDB is a document-oriented NoSQL database.
- It supports dynamic schemas which shws data in JSON format.
- It is a free open source software which gives scalability and high performance.

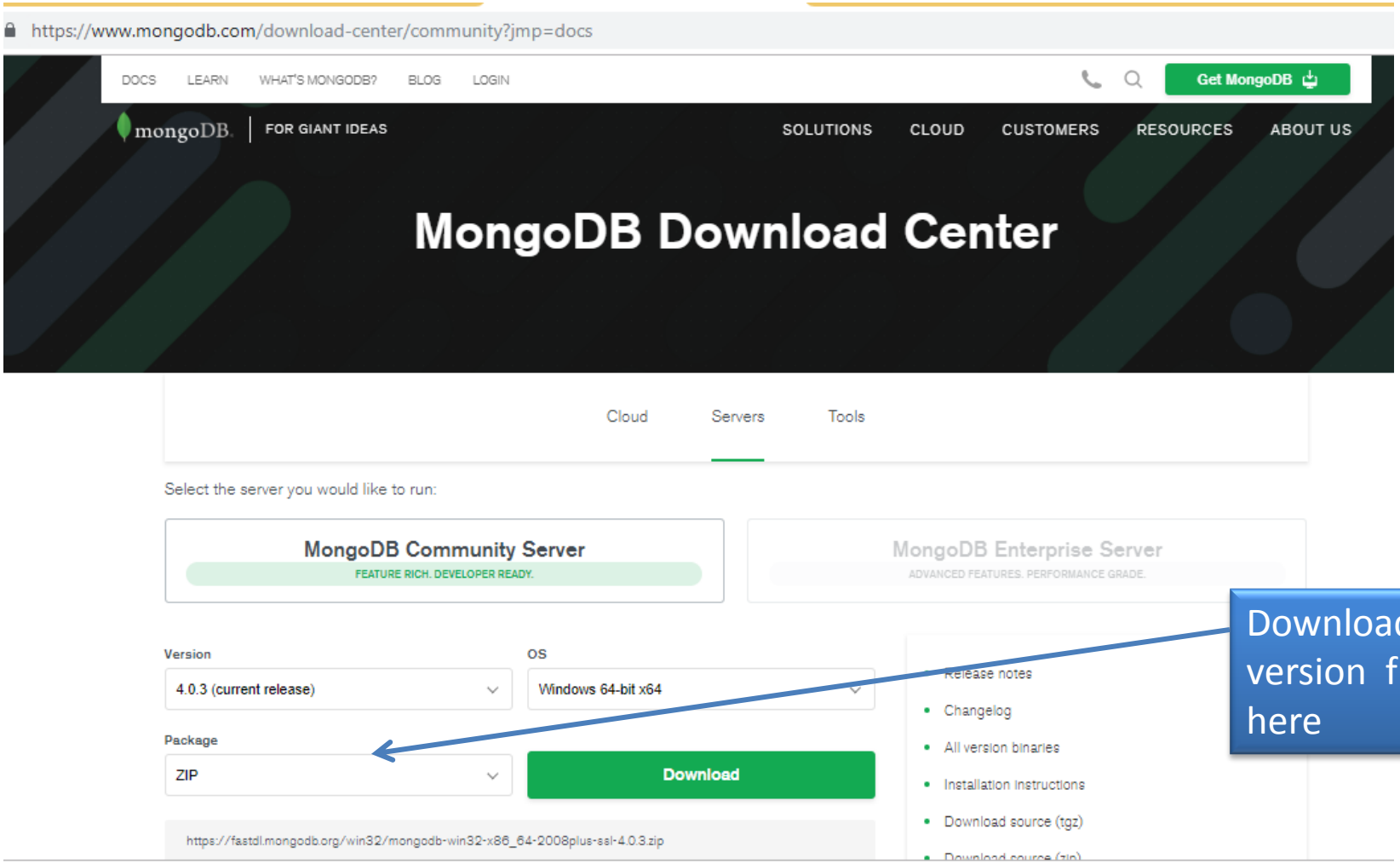
MongoDB Terminology

MongoDB Term	Description	SQL Term
Field	a name-value pair which stores information.	Column
Document	Group of Locally related fields.	Row/record
Collection	Group of Related documents.	Table
Database	A container for Collections. A MongoDB server can have multiple databses.	Database
Primary key	Unique field identifies document.	Primary key

Installing MongoDB

- Copy the following link and paste in browser.

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/#install-mdb-edition



Installation of MongoDB

- Install MongoDB by opeing MSI file.
- After installation, check the availability of mongodb.exe file and mongo.exe file using following path-

  C:\Program Files\MongoDB\Server\4.0\bin

- After this, create a data folder on c:\ and db folder under data folder i.e. **"c:\data\db".**
- Now, run mongodb by using the location C:\Program Files\MongoDB\Server\4.0\bin from command prompt. Do not close mongodb after run.
- Now, in other command window run mongo using the same path.

Starting MongoDB

mongodb-win32-x86_64-2008plus-ssl-4.0.3-signed	10/30/2018 8:57 AM	Windows Installer ...	191,781 KB

This is mongodb's installer which is of about 190 MB.
In my computer, it has been installed on "C:\Program Files\MongoDB\Server\4.0\bin" path.
You cancheck its path in your computer. We can add it in window's path.

mongod

```
te access to data and configuration is unrestricted.
2018-10-30T09:14:31.414-0700 I CONTROL  [initandlisten]
2018-10-30T09:14:31.415-0700 I CONTROL  [initandlisten] ** WARNING: This server
is bound to localhost.
2018-10-30T09:14:31.416-0700 I CONTROL  [initandlisten] **          Remote syste
ms will be unable to connect to this server.
2018-10-30T09:14:31.417-0700 I CONTROL  [initandlisten] **          Start the se
rver with --bind_ip <address> to specify which IP
2018-10-30T09:14:31.417-0700 I CONTROL  [initandlisten] **          addresses it
 should serve responses from, or with --bind_ip_all to
2018-10-30T09:14:31.418-0700 I CONTROL  [initandlisten] **          bind to all
interfaces. If this behavior is desired, start the
2018-10-30T09:14:31.419-0700 I CONTROL  [initandlisten] **          server with
--bind_ip 127.0.0.1 to disable this warning.
2018-10-30T09:14:31.419-0700 I CONTROL  [initandlisten]
2018-10-30T09:14:31.669-0700 W FTDC     [initandlisten] Failed to initialize Per
formance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with '
The specified object was not found on the computer.' for counter '\Memory\Availa
ble Bytes'
2018-10-30T09:14:31.669-0700 I FTDC     [initandlisten] Initializing full-time d
iagnostic data capture with directory 'C:/data/db/diagnostic.data'
2018-10-30T09:14:31.671-0700 I NETWORK  [initandlisten] waiting for connections
on port 27017
```

mongo

```
Server has startup warnings:
2018-10-30T08:59:52.729-0700 I CONTROL  [initandlisten]
2018-10-30T08:59:52.729-0700 I CONTROL  [initandlisten] ** WARNING: Access contr
ol is not enabled for the database.
2018-10-30T08:59:52.730-0700 I CONTROL  [initandlisten] **          Read and wri
te access to data and configuration is unrestricted.
2018-10-30T08:59:52.730-0700 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive an
d display
metrics about your deployment (disk utilization, CPU, operation statistics, etc)
.

The monitoring data will be available on a MongoDB website with a unique URL acc
essible to you
and anyone you share the URL with. MongoDB may use this information to make prod
uct
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeM
onitoring()
---
>
```

It is required to have mogodb in running state before running Mongo. Now, you are ready to give commmand on mongo.

MongoDB Data Types

S.N.	Data Type	DataType Number	S.N.	Data Type	DataType Number
1.	Double	1	10.	Null	11
2.	String	2	11.	Regular Expression	12
3.	Object	3	12.	JavaScript	13
4.	Array	4	13.	Symbol	14
5.	Binary Data	5	14.	JavaScript with scope	15
6.	Undefined	6	15.	Integer	16 and 18
7.	Object Id	7	16.	Timestamp	10
8.	Boolean	9	17.	Min Key	255
9.	Date	10	18.	Max Key	127

MongoDB - Basic commands

- Creation of Database →

  It is not required to create seperate database in MongoDB. As soon as you insert first information n database, database automaticaly created.

- Displaying Current Database→

  >show dbs — it shows database

  >show collections — it shows collections of current database

- Using Database→

  >use mydb

- CRUD operations →

  The operations are as under -

  Create
  Read
  Update
  Delete

  These are knows as CRUD operation

```
> db
test
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
> db
test
> show collections
```

MongoDB - Basic commands

- Creation of databse using Save operation →

It is not required to create seperate database in MongoDB. As soon as you insert first information n database, database automaticaly created.

- You can input data in collection by save or insert command-

  db.<collection-name>.save({<document details>})

- We can use show collections command to confirm creation of collection.
- >USE <DatabaseName> can also be used to create database
- Following example shows creation of school database and input of 1 collection.

```
> use school
switched to db school
> db.student.save({name:'Pankaj'})
WriteResult({ "nInserted" : 1 })
> show collections
student
>
```

MongoDB - Basic commands

- Creation of databse using Save operation→
  - We can insert multiple document like --

```
> db.student.save([{name:'Suresh'},{name:'Hari',age:23}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Here, 2 documents are inserted.

  - When you insert a document, mongoDB adds a field itself **"_id"** it sets its value in increasing order. This process is not visible to us. If we desire, we can give value of "_id" at the time of insertion.

- If you insert a document using Save or insert and name is not received from given database or collection then mongoDB creates a new database for it.

MongoDB -Basic commands

- Creation of database from Insert operation→
  - You can insert data in collection using insert command-

    db.<collection-name>.insert({<document details>})
  - We can use show collections command to confirm creation of collection.
  - Following example shows creation of school database and input of 1 collection.

```
> db.teachers.insert([{name:'Roop Narayan'}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 1,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

  - We can insert multiple document like
  - >db.teachers.insertMany([{name:'Ratan'},{name:'Krishna',age:45}])

MongoDB -Basic commands

- Document can also be inserted by object creation.

```
> stud={name:'Ojas',age:12,city:'Barabanki'}
{ "name" : "Ojas", "age" : 12, "city" : "Barabanki" }
> db.student.insert(stud)
WriteResult({ "nInserted" : 1 })
>
```

Here stud is a valid mongoDB object.

- An object can have a field which is an object itself.

```
> addr={Hno:113, Vill:'Sangram Kheda',post:'Gulariha'}
{ "Hno" : 113, "Vill" : "Sangram Kheda", "post" : "Gulariha" }
> stud={name:'Mohit',age:24,address:addr}
{
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
> db.student.insert(stud)
WriteResult({ "nInserted" : 1 })
```

address is a field of stud , its value is addr which is an object.

MongoDB -Basic commands

- An Object can have arrays too. For ex –

Name: Himanshu

Class:11

Section: A

Subjects: English, Hindi, Maths, Physics, Chemistry

Here subject is an array

```
> newstud={
... name:'Himanshu',
... class:11,
... Sec:'A',
... Subjects:['English','Hindi','Maths','Physics','Chemistry']
... }
{
        "name" : "Himanshu",
        "class" : 11,
        "Sec" : "A",
        "Subjects" : [
                "English",
                "Hindi",
                "Maths",
                "Physics",
                "Chemistry"
        ]
}
> db.student.save(newstud)
WriteResult({ "nInserted" : 1 })
>
```

MongoDB -Basic commands

•Read Operation:

Read operation is used access documents from collection of database. Syntax is-

>db.<collection-name>.find() will show all documents of collection.

>db.<collection-name>.findOne() will show only one record.

>db.<collection-name>.findOne({<key>:<value>}) it will work like search criteria.

If no record matches then it returns null.

```
> use school
switched to db school
> show collections
student
teachers
> db.student.find()
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3d"), "name" : "Hari", "age" : 23 }
{ "_id" : ObjectId("5be44b54f29d9be663cbee3f"), "name" : "Ojas", "age" : 12, "ci
ty" : "Barabanki" }
{ "_id" : ObjectId("5be44ce9f29d9be663cbee40"), "name" : "Mohit", "age" : 24, "a
ddress" : { "Hno" : 113, "Vill" : "Sangram Kheda", "post" : "Gulariha" } }
{ "_id" : ObjectId("5be44ee2f29d9be663cbee41"), "name" : "Himanshu", "class" : 1
1, "Sec" : "A", "Subjects" : [ "English", "Hindi", "Maths", "Physics", "Chemistr
y" ] }
> db.student.findOne()
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
> db.student.findOne({name:'Suresh'})
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
>
```

MongoDB -Basic commands

- Read Operation:

```
> db.student.find().pretty()
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
{
        "_id" : ObjectId("5be446c3f29d9be663cbee3d"),
        "name" : "Hari",
        "age" : 23
}
{
        "_id" : ObjectId("5be44b54f29d9be663cbee3f"),
        "name" : "Ojas",
        "age" : 12,
        "city" : "Barabanki"
}
{
        "_id" : ObjectId("5be44ce9f29d9be663cbee40"),
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
{
        "_id" : ObjectId("5be44ee2f29d9be663cbee41"),
        "name" : "Himanshu",
        "class" : 11,
        "Sec" : "A",
        "Subjects" : [
                "English",
                "Hindi",
                "Maths",
                "Physics",
                "Chemistry"
        ]
}
>
```

pretty( ) prints documents in JSON format with proper indentation.

MongoDB -Basic commands

- Read Operation:

```
> db.student.find({},{name:1})
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3d"), "name" : "Hari" }
{ "_id" : ObjectId("5be44b54f29d9be663cbee3f"), "name" : "Ojas" }
{ "_id" : ObjectId("5be44ce9f29d9be663cbee40"), "name" : "Mohit" }
{ "_id" : ObjectId("5be44ee2f29d9be663cbee41"), "name" : "Himanshu" }
>
```

With the above given example only name field will be displayed with "_id".

If you don't want to display "_id" then command will be like-

```
> db.student.find({},{name:1,_id:0})
{ "name" : "Pankaj" }
{ "name" : "Suresh" }
{ "name" : "Hari" }
{ "name" : "Ojas" }
{ "name" : "Mohit" }
{ "name" : "Himanshu" }
```

MongoDB -Basic Operators

- Comparison Operator:

Like other databases, mongoDB also provides operators so that we can perform delete, read or update operations.

Operator Name	Meaning
$eq	Equal to
$gt	Greater than
$gte	Greater than or equal to
$lt	Less than
$lte	Less than or equal to
$ne	Not equal to

```
> db.student.find({age:{$gt:23}}).pretty()
{
        "_id" : ObjectId("5be44ce9f29d9be663cbee40"),
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
> db.student.find({age:{$gte:23}}).pretty()
{
        "_id" : ObjectId("5be446c3f29d9be663cbee3d"),
        "name" : "Hari",
        "age" : 23
}
{
        "_id" : ObjectId("5be44ce9f29d9be663cbee40"),
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
```

MongoDB-Basic Operators

- Comparision Operator:

Conditional base or range is to be given as-

{field:{$gte:<lower value>, $lte:<upper value>}}

```
> db.student.find({age:{$gte:23,$lte:24}}).pretty()
{
        "_id" : ObjectId("5be446c3f29d9be663cbee3d"),
        "name" : "Hari",
        "age" : 23
}
{
        "_id" : ObjectId("5be44ce9f29d9be663cbee40"),
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
```

MongoDB -Basic Operators

- Condition based on List/Array

{ field :{ $in : [val1,val2, ...] } }
{ field :{ $nin : [val1,val2, ...] } }

Operator Name	Meaning
$in	In
$nin	Not In

```
> db.student.find({Sec:{$in:['A','C']}}).pretty()
{
        "_id" : ObjectId("5be44ee2f29d9be663cbee41"),
        "name" : "Himanshu",
        "class" : 11,
        "Sec" : "A",
        "Subjects" : [
                "English",
                "Hindi",
                "Maths",
                "Physics",
                "Chemistry"
        ]
}
> db.student.find({Sec:{$in:['B','C']}}).pretty()
>
```

Data shown on Section matching but no data shown on no match.

MongoDB-Basic Operators

- Logical Query Operators

```
{ field :{ $not :{<op-Exp>}} }
{ field :{ $and :[{<op-Exp>}, {<op-Exp>},..]} }
{ field :{ $or :[{<op-Exp>}, {<op-Exp>},..]} }
```

Operator Name	Meaning
$not	Logical NOT
$and	Logical AND
$or	Logical OR

```
> db.student.find({age:{$not:{$gt:23}}}).pretty()
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
{
        "_id" : ObjectId("5be446c3f29d9be663cbee3d"),
        "name" : "Hari",
        "age" : 23
}
{
        "_id" : ObjectId("5be44b54f29d9be663cbee3f"),
        "name" : "Ojas",
        "age" : 12,
        "city" : "Barabanki"
}
{
        "_id" : ObjectId("5be44ee2f29d9be663cbee41"),
        "name" : "Himanshu",
        "class" : 11,
        "Sec" : "A",
        "Subjects" : [
                "English",
                "Hindi",
                "Maths",
                "Physics",
                "Chemistry"
        ]
}
> db.student.find({$and:[{age:{$lt:23}},{age:{$gt:11}}]}).pretty()
{
        "_id" : ObjectId("5be44b54f29d9be663cbee3f"),
        "name" : "Ojas",
        "age" : 12,
        "city" : "Barabanki"
}
>
```

•Update Operation:

Update operation can be used in two ways-

>update/updateOne or >updateMany ($set operator is used with it)

```
>db.<CollectionName>.update/updateOne({query-exp},{$set:{<field1>:<val1>, . . . }}
```

```
> db.student.update({age:12},{$set:{name:'Hari Prakash'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find().pretty()
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
{
        "_id" : ObjectId("5be446c3f29d9be663cbee3d"),
        "name" : "Hari",
        "age" : 23
}
{
        "_id" : ObjectId("5be44b54f29d9be663cbee3f"),
        "name" : "Hari Prakash",
        "age" : 12,
        "city" : "Barabanki"
}
{
        "_id" : ObjectId("5be44ce9f29d9be663cbee40"),
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
{
        "_id" : ObjectId("5be44ee2f29d9be663cbee41"),
        "name" : "Himanshu",
        "class" : 11,
        "Sec" : "A",
        "Subjects" : [
                "English",
                "Hindi",
                "Maths",
                "Physics",
                "Chemistry"
        ]
}
```

Only to make it understand this example has taken otherwise updation always to be made with primary key. Here name has modified as Hari Prakash where age was 12.

If you need to update multiple matching records then you should use updateMany( ).

•Delete Operation:

Delete operation can be used in two ways-

>deleteOne or >deleteMany

>db.<CollectionName>.deleteOne({<filter Exp>}) it will delete only one record even on multiple matching.

>db.<CollectionName>.deleteMany({<filter Exp>}) it will delete multiple records on multiple matching.

```
> db.student.deleteOne({name:'Hari Prakash'})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{ "_id" : ObjectId("5be1255c6b46969b847e3f5f"), "name" : "Pankaj" }
{ "_id" : ObjectId("5be446c3f29d9be663cbee3c"), "name" : "Suresh" }
{
        "_id" : ObjectId("5be446c3f29d9be663cbee3d"),
        "name" : "Hari",
        "age" : 23
}
{
        "_id" : ObjectId("5be44ce9f29d9be663cbee40"),
        "name" : "Mohit",
        "age" : 24,
        "address" : {
                "Hno" : 113,
                "Vill" : "Sangram Kheda",
                "post" : "Gulariha"
        }
}
{
        "_id" : ObjectId("5be44ee2f29d9be663cbee41"),
        "name" : "Himanshu",
        "class" : 11,
        "Sec" : "A",
        "Subjects" : [
                "English",
                "Hindi",
                "Maths",
                "Physics",
                "Chemistry"
        ]
}
>
```

"Hari Prakash" record is deleted.

Thank you

Please follow us on our blog-

www.pythontrends.wordpress.com