

```
1 1. Datatypes
2 MongoDB supports many datatypes.
3 1)String
4 -This is the most commonly used datatype to store the data.
5 -String in MongoDB must be UTF-8 valid.
6 2)Integer
7 -This type is used to store a numerical value.
8 -Integer can be 32 bit or 64 bit depending upon your server.
9 3)Boolean
10 -This type is used to store a boolean (true/ false) value.
11 4)Double
12 -This type is used to store floating point values.
13 5)Min/ Max keys
14 -This type is used to compare a value against the lowest and highest BSON elements.
15 6)Arrays
16 -This type is used to store arrays or list or multiple values into one key.
17 7)Timestamp
18 -timestamp.
19 -This can be handy for recording when a document has been modified or added.
20 8)Object
21 -This datatype is used for embedded documents.
22 9)Null
23 -This type is used to store a Null value.
24 10)Symbol
25 -This datatype is used identically to a string.
26 -However, it's generally reserved for languages that use a specific symbol type.
27 11)Date
28 -This datatype is used to store the current date or time in UNIX time format.
29 -You can specify your own date time by creating object of Date and passing day, month, year into it.
30 12)Object ID
31 -This datatype is used to store the document's ID.
32 13)Binary data
33 -This datatype is used to store binary data.
34 14)Code
35 -This datatype is used to store JavaScript code into the document.
36 15)Regular expression
37 -This datatype is used to store regular expression.
38
39
40 2. Insert Document
41 1)The insert() Method
42 -To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.
43 -Syntax
44 >db.COLLECTION_NAME.insert(document)
45 -Example
46 >db.mycol.insert({
47   _id: ObjectId(7df78ad8902c),
48   title: 'MongoDB Overview',
49   description: 'MongoDB is no sql database',
50   by: 'tutorials point',
51   url: 'http://www.tutorialspoint.com',
52   tags: ['mongodb', 'database', 'NoSQL'],
53   likes: 100
54 })
55 -mycol : Collection name.
56 -If the collection doesn't exist in the database, then MongoDB will create this collection and then insert a document into it.
```

```
57 -In the inserted document, if we don't specify the _id parameter, then MongoDB assigns a unique
58 ObjectId for this document.
59 2)_id
60 -is 12 bytes hexadecimal number unique for every document in a collection.
61 -_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)
62
63 3)To insert multiple documents in a single query, you can pass an array of documents in insert()
64 command.
65 >db.post.insert([
66   {
67     title: 'MongoDB Overview',
68     description: 'MongoDB is no sql database',
69     by: 'tutorials point',
70     url: 'http://www.tutorialspoint.com',
71     tags: ['mongodb', 'database', 'NoSQL'],
72     likes: 100
73   },
74   {
75     title: 'NoSQL Database',
76     description: 'NoSQL database doesn't have tables',
77     by: 'tutorials point',
78     url: 'http://www.tutorialspoint.com',
79     tags: ['mongodb', 'database', 'NoSQL'],
80     likes: 20,
81     comments: [
82       {
83         user:'user1',
84         message: 'My first comment',
85         dateCreated: new Date(2013,11,10,2,35),
86         like: 0
87       }
88     ]
89   }
90 ]
91 ])
92 4)To insert the document you can use db.post.save(document) also.
93 -If you don't specify _id in the document then save() method will work same as insert() method.
94 0If you specify _id then it will replace whole data of document containing _id as specified in save()
95 method.
96
97 3. Query Document
98 1)The find() Method
99 -To query data from MongoDB collection, you need to use MongoDB's find() method.
100 -Syntax
101 >db.COLLECTION_NAME.find()
102
103 -find() method will display all the documents in a non-structured way.
104
105 2)The pretty() Method
106 -To display the results in a formatted way, you can use pretty() method.
107 -Syntax
108 >db.mycol.find().pretty()
109 -Example
110 >db.mycol.find().pretty()
```

```

111     {
112       "_id": ObjectId(7df78ad8902c),
113       "title": "MongoDB Overview",
114       "description": "MongoDB is no sql database",
115       "by": "tutorials point",
116       "url": "http://www.tutorialspoint.com",
117       "tags": ["mongodb", "database", "NoSQL"],
118       "likes": "100"
119     }
120   >

```

3) Apart from find() method, there is findOne() method, that returns only one document.

4) RDBMS Where Clause Equivalents in MongoDB

-To query the document on the basis of some condition, you can use following operations.

-Equality

```

--{<key>:<value>}
--db.mycol.find({"by":"tutorials point"}).pretty()
--RDBMS -> where by = 'tutorials point'

```

-Less Than

```

--{<key>:{$lt:<value>}}
--db.mycol.find({"likes":{$lt:50}}).pretty()
--RDBMS -> where likes < 50

```

-Less Than Equals

```

--{<key>:{$lte:<value>}}
--db.mycol.find({"likes":{$lte:50}}).pretty()
--RDBMS -> where likes <= 50

```

-Greater Than

```

--{<key>:{$gt:<value>}}
--db.mycol.find({"likes":{$gt:50}}).pretty()
--RDBMS -> where likes > 50

```

-Greater Than Equals

```

--{<key>:{$gte:<value>}}
--db.mycol.find({"likes":{$gte:50}}).pretty()
--RDBMS -> where likes >= 50

```

-Not Equals

```

--{<key>:{$ne:<value>}}
--db.mycol.find({"likes":{$ne:50}}).pretty()
--RDBMS -> where likes != 50

```

5) AND in MongoDB

-Syntax

-In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it as AND condition.

```

>db.mycol.find(
  {
    $and: [
      {key1: value1}, {key2:value2}
    ]
  }
).pretty()

```

-Example

```

>db.mycol.find({$and:
  [
    {"by":"tutorials point"},
    {"title": "MongoDB Overview"}
  ]
})

```

```

167     }).pretty()
168     {
169         "_id": ObjectId(7df78ad8902c),
170         "title": "MongoDB Overview",
171         "description": "MongoDB is no sql database",
172         "by": "tutorials point",
173         "url": "http://www.tutorialspoint.com",
174         "tags": ["mongodb", "database", "NoSQL"],
175         "likes": "100"
176     }
177 -RDBMS -> ' where by = 'tutorials point' AND title = 'MongoDB Overview' '.
178

```

6)OR in MongoDB

-Syntax

--To query documents based on the OR condition, you need to use \$or keyword.

```

182
183 >db.mycol.find(
184     {
185         $or: [
186             {key1: value1}, {key2:value2}
187         ]
188     }
189 ).pretty()
190 -Example
191 >db.mycol.find({
192     $or:[
193         {"by":"tutorials point"},
194         {"title": "MongoDB Overview"}
195     ]}).pretty()
196     {
197         "_id": ObjectId(7df78ad8902c),
198         "title": "MongoDB Overview",
199         "description": "MongoDB is no sql database",
200         "by": "tutorials point",
201         "url": "http://www.tutorialspoint.com",
202         "tags": ["mongodb", "database", "NoSQL"],
203         "likes": "100"
204     }
205 >
206

```

7)Using AND and OR Together

-Example

-RDBMS -> 'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'

```

210
211 >db.mycol.find(
212     {
213         "likes": {$gt:10},
214         $or: [
215             {"by": "tutorials point"},
216             {"title": "MongoDB Overview"}
217         ]}).pretty()
218     {
219         "_id": ObjectId(7df78ad8902c),
220         "title": "MongoDB Overview",
221         "description": "MongoDB is no sql database",
222         "by": "tutorials point",
223         "url": "http://www.tutorialspoint.com",

```

```

224     "tags": ["mongodb", "database", "NoSQL"],
225     "likes": "100"
226   }
227   >
228
229
230 4. Update Document
231 1)MongoDB's update() and save() methods are used to update document into a collection.
232 2)The update() method updates the values in the existing document while the save() method replaces
the existing document with the document passed in save() method.
233 3)MongoDB Update() Method
234 -The update() method updates the values in the existing document.
235 -Syntax
236   >db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
237 -Example
238   { "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
239   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
240   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
241   Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is
'MongoDB Overview'.
242
243   >db.mycol.update(
244     {'title':'MongoDB Overview'},
245     {$set: {'title':'New MongoDB Tutorial'}}
246   )
247   >db.mycol.find()
248   { "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
249   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
250   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
251   >
252 4)By default, MongoDB will update only a single document.
253 -To update multiple documents, you need to set a parameter 'multi' to true.
254
255   >db.mycol.update(
256     {'title':'MongoDB Overview'},
257     {$set: {'title':'New MongoDB Tutorial'}},
258     {multi:true})
259 5)MongoDB Save() Method
260 -The save() method replaces the existing document with the new document passed in the save()
method.
261 -Syntax
262   >db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})
263 -Example
264   >db.mycol.save(
265     {
266       "_id" : ObjectId(5983548781331adf45ec7),
267       "title":"Tutorials Point New Topic",
268       "by":"Tutorials Point"
269     }
270   )
271   >db.mycol.find()
272   { "_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New Topic",
273     "by":"Tutorials Point"}
274   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
275   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
276   >
277

```

278 5. Delete Document

279 1)The remove() Method

280 -MongoDB's remove() method is used to remove a document from the collection.

281 -remove() method accepts two parameters.

282 -One is deletion criteria and second is justOne flag.

283 --deletion criteria

284 --- (Optional) deletion criteria according to documents will be removed.

285 --justOne

286 --- (Optional) if set to true or 1, then remove only one document.

287 -Syntax

288 >db.COLLECTION_NAME.remove(DELETION_CRITERIA)

289 -Example

290 { "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

291 { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

292 { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

293 Following example will remove all the documents whose title is 'MongoDB Overview'.

294

295 >db.mycol.remove({'title':'MongoDB Overview'})

296 >db.mycol.find()

297 { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

298 { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

299 >

300

301 2)Remove Only One

302 -If there are multiple records and you want to delete only the first record, then set justOne parameter in remove() method.

303

304 >db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)

305

306 3)Remove All Documents

307 -If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection.

308 -This is equivalent of SQL's truncate command.

309

310 >db.mycol.remove()

311 >db.mycol.find()

312 >

313

314

315

316

317

318

319

320

321