# Adding and Querying the data into MongoDB

## What is MongoDB?

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++.It is a cross-platform, document Oriented database that provides, high performance, high availability, and easy scalability. It works on concept of collection and document. MongoDB uses json-like documents with schema. MongoDB is developed by MongoDB Inc and licensed under the Server Side Public License (SSPL). Any relational database has a typical schema design that shows number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship.

## Adding the data into MongoDB:

## The Insert() Method

To insert data into MongoDB collection, you need to use MongoDB's Insert() or save() method.

## Syntax

The basic syntax of Insert () command is as follows-
>db.COLLECTION_NAME.insert(document)

Example:

```
>db.student.insert({

_id:ObjectId(110517818056),

name:'Ravi Ranjan Kumar',

department:'Computer Science and Engineering',

Age:20,

Email:'bmrs2398@gmail.com'

});
```

## Output:

**student**

```
{"_id":110517818056,"name":"Ravi Ranjan kumar","department":"Computer Science and Engineering","Age":20,"email":"bmrs2398@gmail.com"}
```

Here student is our collection name.if collection doesn't exist in the database,then MongoDB will create this collection and then insert a data into it.

In the inserted data,if we don’t specify the _id parameter ,then MangoDB assigns a unique ObjectId for this data.

_id is 12 bytes hexadecimal number unique for every document in a collection .12 bytes divided as follows:

_id:ObjectId(4 bytes college’s registration,3 bytes department code ,2 bytes enrollement year,3 bytes registration number)

## Example:

>db.college.insert({

Name:’IIIT,Trichy’,

Url:’www.iiitt.ac.in’,

Email:’iiitt@gmail.com’

});

## Output:

**college**

{"_id":"5d749c4ae188590010d4f068","Name":"IIIT,Trichy","Url":"www.iiitt.ac.in","Email":"iiitt@gmail.com"}

To insert multiple data in a single query, you can pass an array of documents in insert() command:

The insertMany() method has follwing syntax:

db.collection.insertMany( [ <document 1> , <document 2>, ... ], { writeConcern: <document>, ordered: <**boolean**> })

Example:

>db.college_details.insert([

{_id:ObjectId(9501),

Name:’IIIT,Trichy’,

Url:’www.iiitt.ac.in’,

email:’iiitt@gmail.com’,

},

{

_id:ObjectId(1205),

Name:’NIT,Trichy’,

Url:’www.nitt.ac.in’,

Email:’nitt@gmail.com’,

Comments:[{user:’IIITT_students’}]}]);

## Output:

**college_details**

{"_id":9501,"Name":"IIIT,Trichy","Url":"www.iiitt.ac.in","email":"iiitt@gmail.com"}

{"_id":1205,"Name":"NIT,Trichy","Url":"www.nitt.ac.in","Email":"nitt@gmail.com","Comments":[{"user":"IIITT_students"}]}

To insert the document you can use dp.college_details.save(data) also.if you don’t specify id in the document then save() method will work same as insert() method.if you specify _id then it will replace whole data of document containing id as specified save() method.

## Example:

```
db.inventory.insertMany([   { item: "journal",

 qty: 25,

tags: ["blank", "red"],

size: { h: 14, w: 21, uom: "cm" } },

   { item: "mat",

qty: 85,

tags: ["gray"],

size: { h: 27.9, w: 35.5, uom: "cm" } },

   { item: "mousepad",

qty: 25,

 tags: ["gel", "blue"],

size: { h: 19, w: 22.85, uom: "cm" } }]);
```

## Output:

**inventory**

```
{"_id":"5d74a273e188590010d4f069","item":"journal","qty":25,"tags":["blank","red"],"size":{"h":14,"w":21,"uom":"cm"}}{"_id":"5d74a273e188590010d4f06a","item":"mat","qty":85,"tags":["gray"],"size":{"h":27.9,"w":35.5,"uom":"cm"}}{"_id":"5d74a273e188590010d4f06b","item":"mousepad","qty":25,"tags":["gel","blue"],"size":{"h":19,"w":22.85,"uom":"cm"}}
```

Additional Methods for Inserts

The following methods can also add new documents to a collection:

- **db.collection.update()** when used with the upsert:true option.
- **db.collection.updateOne()** when used with the upsert:true option.
- **db.collection.updateMany()** when used with upsert :true option.
- **db.collection.findAndModify()** when used with upsert:true option.
- **db.collection.findOneAndUpdate()** when used with upsert:true option.
- **db.collection.findoneAndReplace()** when used with upsert :true option.
- **db.collection.save()**.
- **db.collection.bulkWrite().**

## Quering data into MongoDB:

The **find() Method**

To query data from MongoDB collection, you need to use MongoDB's find() method.

Syntax: The basic syntax of find() method is as follows-

```
 >db.COLLECTION_NAME.find() find()
```

method will display all the documents in a non-structured way.

**The pretty() Method**

To display the results in a formatted way, we can use pretty() method.

Syntax:

```
>db.COLLECTION_NAME.find().pretty()
```

Example:

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

**Output:**

**inventory**

```
{"_id":"5d74a273e188590010d4f069","item":"journal","qty":25,"tags":["blank","red"],"size":{"
h":14,"w":21,"uom":"cm"}}

{"_id":"5d74a273e188590010d4f06a","item":"mat","qty":85,"tags":["gray"],"size":{"h":27.9,"w
":35.5,"uom":"cm"}}

{"_id":"5d74a273e188590010d4f06b","item":"mousepad","qty":25,"tags":["gel","blue"],"size":{
"h":19,"w":22.85,"uom":"cm"}}

{"_id":"5d74a63be188590010d4f06c","item":"journal","qty":25,"size":{"h":14,"w":21,"uom":"c
m"},"status":"A"}

{"_id":"5d74a63be188590010d4f06d","item":"notebook","qty":50,"size":{"h":8.5,"w":11,"uom"
:"in"},"status":"A"}

{"_id":"5d74a63be188590010d4f06e","item":"paper","qty":100,"size":{"h":8.5,"w":11,"uom":"i
n"},"status":"D"}

{"_id":"5d74a63be188590010d4f06f","item":"planner","qty":75,"size":{"h":22.85,"w":30,"uom"
:"cm"},"status":"D"}

{"_id":"5d74a63be188590010d4f070","item":"postcard","qty":45,"size":{"h":10,"w":15.25,"uo
m":"cm"},"status":"A"}
```

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

**Example:**

```
db.inventory.find( { status: "D" } );
```

**Output:**

```
{"_id":"5d74a63be188590010d4f06e","item":"paper","qty":100,"size":{"h":8.5,"w":11,"uom":"i
n"},"status":"D"}

{"_id":"5d74a63be188590010d4f06f","item":"planner","qty":75,"size":{"h":22.85,"w":30,"uom"
:"cm"},"status":"D"}
```

RDBMS Where Clause Equivalents in MongoDB

To query the data on the basis of some condition , you use the following operations.

| Operation | Syntax | Example | RDBMS Equivalent |
|---|---|---|---|
| Equality | {<key>:<value>} | db.student.find({“email”:”bmrs2398@gmail.com}).pretty() | Where email=”bmrs2398@gmail.com” |
| Less Than | {<key>:{$lt:<value>}} | db.student.find({“Age”:{$lt:20}}).pretty() | Where Age<20 |
| Less Than Equals | {<key>:{$lte:<value>}} | db.student.find({“Age”:{$lte:20}}).pretty() | Where Age<=20 |
| Greater Than | {<key>:{$gt:<value>}} | db.student.find({“Age”:{$gt:<value>}} | Where Age>20 |
| Greater Than Equals | {<key>:{$gte:<value>}} | db.student.find({“Age”:{$gte:<value>}} | Where Age>=20 |
| Not Equals | {<key>:{$ne:<value>}} | db.student.find({“Age”:{$ne:20}}.pretty() | Where Age!=20 |

## Specify Conditions Using Query Operators:

A query filter document can use the query operators to specify conditions in the following form:

```
{ <field1>: { <operator1>: <value1> }, ... }
```

**Example**:

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } );
```

### Output:

```
{"_id":"5d74a63be188590010d4f06c","item":"journal","qty":25,"size":{"h":14,"w":21,"uom":"cm"},"status":"A"}

{"_id":"5d74a63be188590010d4f06d","item":"notebook","qty":50,"size":{"h":8.5,"w":11,"uom":"in"},"status":"A"}

{"_id":"5d74a63be188590010d4f06e","item":"paper","qty":100,"size":{"h":8.5,"w":11,"uom":"in"},"status":"D"}

{"_id":"5d74a63be188590010d4f06f","item":"planner","qty":75,"size":{"h":22.85,"w":30,"uom":"cm"},"status":"D"}

{"_id":"5d74a63be188590010d4f070","item":"postcard","qty":45,"size":{"h":10,"w":15.25,"uom":"cm"},"status":"A"}
```

### Specify AND conditions

A compuund query can specify conditions for more than one field in the collection's documents.Implicitly,a logical AND conjunctions connnects the clauses of a compound query so that query selects the documents in the collection that match all the conditions.

The following example retrieves all documents in the inventory collection where the status equals"A" and qty is less than ($lt)30:

**Example:**

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } );
```

**Output:**

```
{"_id":"5d74a63be188590010d4f06c","item":"journal","qty":25,"size":{"h":14,"w":21,"uom":"cm"},"stat
us":"A"}
```

## Specify OR Conditions

Using the $or operator, you can specify a compound query that joins each clause with a logical OR conjunction so that the query selects the documents in the collection that match at least one condition.

The Following example retrieves all documents in the collection where the status equals "A" or qty is less than ($lt)30.

**Example:**

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } );
```

**Output:**

```
{"_id":"5d74a273e188590010d4f069","item":"journal","qty":25,"tags":["blank","red"],"size":{"h":14,"w":21,"
uom":"cm"}}

{"_id":"5d74a273e188590010d4f06b","item":"mousepad","qty":25,"tags":["gel","blue"],"size":{"h":19,"w":22
.85,"uom":"cm"}}

{"_id":"5d74a63be188590010d4f06c","item":"journal","qty":25,"size":{"h":14,"w":21,"uom":"cm"},"status":"
A"}

{"_id":"5d74a63be188590010d4f06d","item":"notebook","qty":50,"size":{"h":8.5,"w":11,"uom":"in"},"status"
:"A"}

{"_id":"5d74a63be188590010d4f070","item":"postcard","qty":45,"size":{"h":10,"w":15.25,"uom":"cm"},"stat
us":"A"}
```

## Specify AND as well as OR Conditions

In the above example,the compound query document selects all documents in the collection where the status equals "A" and either qty is less than ($lt)30 or item starts with the character p:

```
db.inventory.find( {
     status: "A",
     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} );
```

**Output**:

```
{"_id":"5d74a63be188590010d4f06c","item":"journal","qty":25,"size":{"h":14,"w":21,"uom":"c
m"},"status":"A"}

{"_id":"5d74a63be188590010d4f070","item":"postcard","qty":45,"size":{"h":10,"w":15.25,"uo
m":"cm"},"status":"A"}
```

## Query an Array

To specify equality condition on an array,use the query document{<field>:<value>}where <value> is the exact array to match ,including the order of the elemnts.

The above example queries for all documents where the field tags value is an array with exactly two elements, ”red” and “blank” , in the specified order:

```
 db.inventory.find( { tags: ["red", "blank"] } );
```

## Output:

```
[]
```

If, instead ,you wish to find an array that contains both the element “red” and “blank”,without regard to order or other elements  in the array,use the $all operator:

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } );
```

## Output:

```
{"_id":"5d74a273e188590010d4f069","item":"journal","qty":25,"tags":["blank","red"],"size":{"
h":14,"w":21,"uom":"cm"}}
```

## Query an Array for an Element

To query if the array field contains at least with the specified value,use the filter{<field>:<value>}where <value> is the element value.

The following example queries for all documents where tags is an array that the string “red” as one of its elements:

```
db.inventory.find( { tags: "red" } );
```

**Output:**

```
{"_id":"5d74a273e188590010d4f069","item":"journal","qty":25,"tags":["blank","red"],"size":{"h":14,"w":21,"uom":"cm"}}
```

To specify conditions on the elements in the array field,use query operators in the filter document:

```
{ <array field>: { <operator1>: <value1>, ... } }
```

For example,the following operations queries for all documents where the array dim_cm contains at least one element whose value is greater than 25.

```
db.inventory.find( { dim_cm: { $gt: 25 } } );
```

## Output:

```
[]
```

## Specify Multiple Conditions for Array Elements

When specifying compound conditions on array elements, you can specify the query such that either a single array element meets these condition or any combination of array elements the conditions.

## Query an Array with Compound Filter Conditions on the Array Elements

The following examples queries for documents where the dim_cm array contains elements that in some combinations satisfy the query conditions;e.g.,one element can satisfy the greater than 15 condition and another element can satisfy the less than 20 condition,or a single element can satisfy both:

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } );
```

## Output:
[]

## Query for an Array Element that Meets Multiple Criteria

Use $elemMatch operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.

The following example queries for documents where the dim_cm array contains at least one element that is both greater than ($gt)22 and less than($lt)30:

db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } );

## Output:

[]

## Query for an element by the Array Index Position

Using dot notation ,you can specify query conditions for an element at a particular index or position of the array.The array uses zero-based indexing.

The following examples queries for all documents where the second element int he array dim_cm is greater than 25:

db.inventory.find( { "dim_cm.1": { $gt: 25 } } );

## Output:

[]

## Query an Array by Array length

Use the $size operator for arrays by number of elements. for example,

The following selects documents where the array tags has 3 elements:

db.inventory.find( { "tags": { $size: 3 } } );

**Output:**

[]