Basic CRUD operations in MONGODB

The following section shows you how to create (C), read (R), update (U), **and delete (D)** a document. These operations are often referred to as CRUD operations.

Note that this section only covers the basic CRUD operations, you will learn about them in detail in the CRUD tutorial.

Create

To add a document to a collection, you use the insertOne() method of the collection.

The following command adds a new document (or a new book) to the books collection:

```
db.books.insertOne({
  title: "MongoDB Tutorial",
  published_year: 2020
})
```

Code language: CSS (css)

Output:

```
{
    "acknowledged" : true,
    "insertedId" :
ObjectId("5f2f39fb82f5c7bd6c9375a8")
}
```

Code language: JSON / JSON with Comments (json)

Once you press enter, the mongo shell sends the command to the MongoDB server.

If the command is valid, MongoDB will insert the document and return the result. In this case, it returns an object that has two keys acknowledged and insertedId.

The value of the insertedId is the value of the _id field in the document.

When you add a document to a collection without specifying the _id field, MongoDB automatically assigns a unique ObjectId value to the _id field and add it to the document.

MongoDB uses the _id field to uniquely identify the document in the collection.

Read

To select documents in a collection, you use the findOne() method:

```
db.books.findOne()
```
Code language: CSS (css)

Output:

```
{ "_id" : ObjectId("5f2f3d8882f5c7bd6c9375ab"), "title" : "MongoDB Tutorial", "published_year" : 2020 }
```
Code language: JSON / JSON with Comments (json)

To format the output, you use the pretty() method like this:

```
db.books.find().pretty()
```
Code language: CSS (css)

Output:

```
{
        "_id" : ObjectId("5f2f3d8882f5c7bd6c9375ab"),
        "title" : "MongoDB Tutorial",
        "published_year" : 2020
}
```
Code language: JSON / JSON with Comments (json)

As you can see clearly from the output, MongoDB added the _id field together with other field-and-value pairs to the document.

Update

To update a single document, you use the updateOne() method.

The updateOne() method takes at least two arguments:

- The first argument identifies the document to update.
- The second argument represents the updates that you want to make.

The following shows how to update the published_year of the document whose title is "MongoDB Tutorial":

```
db.books.updateOne(
   {
 title: "MongoDB Tutorial"
},
   {
 $set: {
       published_year: 2019
   }
}
)
```

Code language: PHP (php)

Output:

```
{
   "acknowledged": true,
   "matchedCount": 1,
   "modifiedCount": 1
}
```

Code language: JSON / JSON with Comments (json)

How it works.

The first argument identifies which document to update. In this case, it will update the first document that has the title "MongoDB tutorial":

```
{title: "MongoDB Tutorial"}
```

Code language: CSS (css)

The second argument is an object that specifies which fields in the document to update:

```
{
    $set: {
        published_year: 2019
    }
}
```

Code language: PHP (php)

The $set is an operator that replaces a field value with a specified value. In this example, it updates the published_year of the document to 2019.

Delete

To delete a document from a collection, you use the deleteOne() method. The deleteOne() method takes one argument that identifies the document that you want to delete.

The following example uses the deleteOne() method to delete a document in the books collection:

```
db.books.deleteOne({title: "MongoDB Tutorial"});
```

Code language: CSS (css)

Output:

```
{
    "acknowledged": true,
```

```
  "deletedCount": 1
}
```

Code language: JSON / JSON with Comments (json)

The output shows that one document has been deleted successfully via the deletedCount field.

Update Documents

This page uses the following `mongosh` methods:

- `db.collection.updateOne(<filter>, <update>, <options>)`
- `db.collection.updateMany(<filter>, <update>, <options>)`
- `db.collection.replaceOne(<filter>, <update>, <options>)`

The examples on this page use the

`inventory`

collection. To create and/or populate the

`inventory`

collection, run the following:

```
db.inventory.insertMany( [
{ item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },
{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
{ item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },
```

```
{ item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },
{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },
{ 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" },
{ item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
{ item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }
] );
```

MongoDB Shell

You can run the operation in the web shell below:

Update Documents in a Collection

To update a document, MongoDB provides update operators, such as `$set`, to modify field values.

To use the update operators, pass to the update methods an update document of the form:

```
{
<update operator>: { <field1>: <value1>, ... },
<update operator>: { <field2>: <value2>, ... },
...
}
```

MongoDB Shell

Update a Single Document

The following example uses the `db.collection.updateOne()` method on the `inventory` collection to update the *first* document where `item` equals `"paper"`:

```
db.inventory.updateOne(
{ item: "paper" },
{
$set: { "size.uom": "cm", status: "P" },
$currentDate: { lastModified: true }
}
)
```

MongoDB Shell

The update operation:

- uses the `$set` operator to update the value of the `size.uom` field to `"cm"` and the value of the `status` field to `"P"`,
- uses the `$currentDate` operator to update the value of the `lastModified` field to the current date. If `lastModified` field does not exist, `$currentDate` will create the field.
  See `$currentDate` for details.

```
db.inventory.updateMany(
{ "qty": { $lt: 50 } },
{
$set: { "size.uom": "in", status: "P" },
$currentDate: { lastModified: true }
}
)
```

MongoDB Shell

The update operation:

- uses the `$set` operator to update the value of the `size.uom` field to `"in"` and the value of the `status` field to `"P"`,

- uses the `$currentDate` operator to update the value of the `lastModified` field to the current date. If `lastModified` field does not exist, `$currentDate` will create the field.
  See `$currentDate` for details.

```
db.inventory.replaceOne(
{ item: "paper" },
{ item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }
)
```