Update Documents

This page provides examples in:

MONGO SHELL COMPASS PYTHON JAVA (SYNC) NODE.JS PHP MOTOR

JAVA (ASYNC) C# PERL RUBY SCALA GO

This page uses the following `mongo` shell 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" }
] );
```

You can run the operation in the web shell below:

```
type "help" for help
>> 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"
},
>>>
```


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>, ... },
  ...
}
```

Some update operators, such as `$set`, will create the field if the field does not exist. See the individual update operator reference for details.

> **NOTE:**
> Starting in MongoDB 4.2, MongoDB can accept an aggregation pipeline to specify the modifications to make instead of an update document. See the method reference page for details.

Update a Single Document

The following example uses the `db.collection` update the *first* document where `item` equals `"pa`

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

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.

Update Multiple Documents

New in version 3.2.

The following example uses the `db.collection.updateMany()` method on the `inventory` collection to update all documents where `qty` is less than `50`:

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

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.

Replace a Document

To replace the entire content of a document except for the `_id` field, pass an entirely new document as the second argument to `db.collection.replaceOne()`.

When replacing a document, the replacement document must consist of only field/value pairs; i.e. do not include update operators expressions.

The replacement document can have different fields from the original document. In the replacement document, you can omit the `_id` field since the `_id` field is immutable; however, if you do include the `_id` field, it must have the same value as the current value.

The following example replaces the *first* document from the `inventory` collection where `item: "paper"`:

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

Behavior

Atomicity

All write operations in MongoDB are atomic on the level of a single document. For more information on MongoDB and atomicity, see Atomicity and Transactions.

`_id` Field

Once set, you cannot update the value of the `_id` replacement document that has a different `_id` fiel

Field Order

MongoDB preserves the order of the document fields following write operations *except* for the following cases:

- The `_id` field is always the first field in the document.
- Updates that include `renaming` of field names may result in the reordering of fields in the document.

Upsert Option

If `updateOne()`, `updateMany()`, or `replaceOne()` includes `upsert : true` **and** no documents match the specified filter, then the operation creates a new document and inserts it. If there are matching documents, then the operation modifies or replaces the matching document or documents.

For details on the new document created, see the individual reference pages for the methods.

Write Acknowledgement

With write concerns, you can specify the level of acknowledgement requested from MongoDB for write operations. For details, see Write Concern.

SEE ALSO:

- Updates with Aggregation Pipeline
- `db.collection.updateOne()`
- `db.collection.updateMany()`
- `db.collection.replaceOne()`
- Additional Methods