Large-Scale and Multi-Structured Databases

CRUD operations on MongoDB

Prof Pietro Ducange

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Copyright Issues

Most of the information included this presentation have been extracted from the official documentation of MongoDB.

Specifically, in this classes we used data extracted from:

https://docs.mongodb.com/manual/crud/

Create Operations

Create or insert operations ***add new documents*** to a collection.

If the collection does not currently exist, ***insert*** operations ***will create*** the ***collection***.

In MongoDB, insert operations target a ***single collection***. All write operations in MongoDB ***are atomic*** on the level of a ***single document***.

In MongoDB, each document stored in a collection requires a ***unique _id field*** that acts as a primary key.

If an inserted document omits the _id field, the MongoDB driver ***automatically generates*** an ObjectId for the _id field.

insertOne()

```
db.users.insertOne(          <-- collection
   {
      name: "sue",           <-- field: value
      age: 26,               <-- field: value
      status: "pending"      <-- field: value
   }                                          } document
)
```

The following example inserts a new document into the inventory collection.

```
>>> db.inventory.insertOne(
...     { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5d9ddf06e87b85fdfc31eb19")
}
```

If the document does not specify an _id field, MongoDB adds the _id field with an ObjectId value to the new document.

insertMany()

insertMany() can insert multiple documents into a collection.

To this aim, an array of documents must be passed to the method.

The following example ***inserts three new documents*** into the inventory collection.

```
>>> 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" } }
... ])
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9ddfed7ab5912f2d623a8c"),
        ObjectId("5d9ddfed7ab5912f2d623a8d"),
        ObjectId("5d9ddfed7ab5912f2d623a8e")
    ]
}
```

By default documents are inserted ***in order***.

Query Documents

Let suppose to create the following collection in the DB:

```
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" }
]);
```

Query Documents

In the following, we show an example for querying all documents of a specific collection.

```
>>> 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" }
... ]);
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9de2d501f1b0bfec711681"),
        ObjectId("5d9de2d501f1b0bfec711682"),
        ObjectId("5d9de2d501f1b0bfec711683"),
        ObjectId("5d9de2d501f1b0bfec711684"),
        ObjectId("5d9de2d501f1b0bfec711685")
    ]
}
>>> db.inventory.find( {} )
{ "_id" : ObjectId("5d9de2d501f1b0bfec711681"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9de2d501f1b0bfec711682"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("5d9de2d501f1b0bfec711683"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("5d9de2d501f1b0bfec711684"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("5d9de2d501f1b0bfec711685"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
>>>
```

The query correspond to the ***SELECT * FROM inventory*** in SQL language.

Specify Equality Condition

To specify equality conditions, use ***<field>:<value>*** expressions as parameters for the ***find()*** function.

```
>>> 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" }
... ]);
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9de446f7b6a95d458f9b29"),
        ObjectId("5d9de446f7b6a95d458f9b2a"),
        ObjectId("5d9de446f7b6a95d458f9b2b"),
        ObjectId("5d9de446f7b6a95d458f9b2c"),
        ObjectId("5d9de446f7b6a95d458f9b2d")
    ]
}
>>> db.inventory.find( { status: "D" } )
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b2b"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b2c"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
```

This operation corresponds to the following SQL statement:

*SELECT * FROM inventory WHERE status = "D"*

Specify Conditions Using Query Operators

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

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

The following example retrieves all documents from the inventory collection where status equals either "A" or "D":

```
>>> db.inventory.find( { status: { $in: [ "A", "D" ] } } )
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b29"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b2a"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b2b"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b2c"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("5d9de446f7b6a95d458f9b2d"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
```

The operation corresponds to the following SQL statement:

*SELECT * FROM inventory WHERE status in ("A", "D")*

Specify AND Conditions

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

```
>>> 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" }
... ]);
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9de67ed82723f0aca75b4d"),
        ObjectId("5d9de67ed82723f0aca75b4e"),
        ObjectId("5d9de67ed82723f0aca75b4f"),
        ObjectId("5d9de67ed82723f0aca75b50"),
        ObjectId("5d9de67ed82723f0aca75b51")
    ]
}
>>> db.inventory.find( { status: "A", qty: { $lt: 30 } } )
{ "_id" : ObjectId("5d9de67ed82723f0aca75b4d"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
```

The operation corresponds to the following SQL statement:

*SELECT * FROM inventory WHERE status = "A" AND qty < 30*

Specify OR Conditions

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

```
>>> 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" }
... ]);
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9de67ed82723f0aca75b4d"),
        ObjectId("5d9de67ed82723f0aca75b4e"),
        ObjectId("5d9de67ed82723f0aca75b4f"),
        ObjectId("5d9de67ed82723f0aca75b50"),
        ObjectId("5d9de67ed82723f0aca75b51")
    ]
}
>>> db.inventory.find( { status: "A", qty: { $lt: 30 } } )
{ "_id" : ObjectId("5d9de67ed82723f0aca75b4d"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
>>> db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
{ "_id" : ObjectId("5d9de67ed82723f0aca75b4d"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9de67ed82723f0aca75b4e"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("5d9de67ed82723f0aca75b51"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
>>>
```

The operation corresponds to the following SQL statement:

*SELECT * FROM inventory WHERE status = "A" OR qty < 30*

Specify AND as well as OR Conditions

In the following 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.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" }
... ]);
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9de954e0d5e8eaf04a1997"),
        ObjectId("5d9de954e0d5e8eaf04a1998"),
        ObjectId("5d9de954e0d5e8eaf04a1999"),
        ObjectId("5d9de954e0d5e8eaf04a199a"),
        ObjectId("5d9de954e0d5e8eaf04a199b")
    ]
}
>>> db.inventory.find( {
...      status: "A",
...      $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
... } )
{ "_id" : ObjectId("5d9de954e0d5e8eaf04a1997"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9de954e0d5e8eaf04a199b"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
>>>
```

The operation corresponds to the following SQL statement:

*SELECT * FROM inventory WHERE status = "A" AND (qty < 30 OR item LIKE "p%")*

Comparison Query Operators

Name	Description
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.

Details on ***Comparison Query operators*** can be found here:

https://docs.mongodb.com/manual/reference/operator/query-comparison/#query-selectors-comparison

Update Documents

We will discuss the following functions:

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

- The *<filter>* document is defined as a query that we previously analyzed and specify the set of document to update.

- The *<update>* document specify the modification to apply.

- The *<options>* document specify a set of parameters for the modifications.

The <update> document

To use the update operators, we need to pass to the update methods an ***update document*** as follows:

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

A list with details of update operators can be found here:
https://docs.mongodb.com/manual/reference/operator/update/

Some update operators, such as *$set*, ***will create the field*** if the field does not exist.

The Update Operators

Name	Description
`$currentDate`	Sets the value of a field to current date, either as a Date or a Timestamp.
`$inc`	Increments the value of the field by the specified amount.
`$min`	Only updates the field if the specified value is less than the existing field value.
`$max`	Only updates the field if the specified value is greater than the existing field value.
`$mul`	Multiplies the value of the field by the specified amount.
`$rename`	Renames a field.
`$set`	Sets the value of a field in a document.
`$setOnInsert`	Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.
`$unset`	Removes the specified field from a document.

Creating the DB for Updating Operators

Let suppose to create the following collection in the DB:

```
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" }
] );
```

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 }
  }
)
```

- ***the $set operator*** updates the value of the *size.uom* field to "cm" and the value of the *status* field to "P",

- ***the $currentDate operator*** updates 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

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.find();
{ "_id" : ObjectId("5db801d808ca5aab07e32ccb"), "item" : "canvas", "qty" : 100, "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32ccc"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32ccd"), "item" : "mat", "qty" : 85, "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cce"), "item" : "mousepad", "qty" : 25, "size" : { "h" : 19, "w" : 22.85, "uom" : "cm" }, "status" : "P" }
{ "_id" : ObjectId("5db801d808ca5aab07e32ccf"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "P" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd0"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "cm" }, "status" : "P", "lastModified" : ISODate("2019-10-29T09:09:51.400Z") }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd1"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd2"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd3"), "item" : "sketchbook", "qty" : 80, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd4"), "item" : "sketch pad", "qty" : 95, "size" : { "h" : 22.85, "w" : 30.5, "uom" : "cm" }, "status" : "A" }
>>> db.inventory.updateMany(
...    { "qty": { $lt: 50 } },
...    {
...      $set: { "size.uom": "in", status: "P" },
...      $currentDate: { lastModified: true }
...    }
... )
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
>>> db.inventory.find();
{ "_id" : ObjectId("5db801d808ca5aab07e32ccb"), "item" : "canvas", "qty" : 100, "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32ccc"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "in" }, "status" : "P", "lastModified" : ISODate("2019-10-29T09:11:19.014Z") }
{ "_id" : ObjectId("5db801d808ca5aab07e32ccd"), "item" : "mat", "qty" : 85, "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cce"), "item" : "mousepad", "qty" : 25, "size" : { "h" : 19, "w" : 22.85, "uom" : "in" }, "status" : "P", "lastModified" : ISODate("2019-10-29T09:11:19.014Z") }
{ "_id" : ObjectId("5db801d808ca5aab07e32ccf"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "P" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd0"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "cm" }, "status" : "P", "lastModified" : ISODate("2019-10-29T09:09:51.400Z") }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd1"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd2"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "in" }, "status" : "P", "lastModified" : ISODate("2019-10-29T09:11:19.014Z") }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd3"), "item" : "sketchbook", "qty" : 80, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5db801d808ca5aab07e32cd4"), "item" : "sketch pad", "qty" : 95, "size" : { "h" : 22.85, "w" : 30.5, "uom" : "cm" }, "status" : "A" }
```

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***.

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 } ] }
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>>> db.inventory.find();
{ "_id" : ObjectId("5d9ee57579681126ee0de564"), "item" : "canvas", "qty" : 100, "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9ee57579681126ee0de565"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9ee57579681126ee0de566"), "item" : "mat", "qty" : 85, "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9ee57579681126ee0de567"), "item" : "mousepad", "qty" : 25, "size" : { "h" : 19, "w" : 22.85, "uom" : "cm" }, "status" : "P" }
{ "_id" : ObjectId("5d9ee57579681126ee0de568"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "P" }
{ "_id" : ObjectId("5d9ee57579681126ee0de569"), "item" : "paper", "instock" : [ { "warehouse" : "A", "qty" : 60 }, { "warehouse" : "B", "qty" : 40 } ] }
{ "_id" : ObjectId("5d9ee57579681126ee0de56a"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("5d9ee57579681126ee0de56b"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9ee57579681126ee0de56c"), "item" : "sketchbook", "qty" : 80, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9ee57579681126ee0de56d"), "item" : "sketch pad", "qty" : 95, "size" : { "h" : 22.85, "w" : 30.5, "uom" : "cm" }, "status" : "A" }
>>> db.inventory.find({});
```

Delete Documents

To delete ***all documents*** from a collection, pass an empty filter document {} to the *db.collection.deleteMany()* method.

The following example deletes ***all documents*** from the inventory collection:

```
type "help" for help
>>> 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: "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" },
... ] );
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9ee714b92588845f04cbf5"),
        ObjectId("5d9ee714b92588845f04cbf6"),
        ObjectId("5d9ee714b92588845f04cbf7"),
        ObjectId("5d9ee714b92588845f04cbf8"),
        ObjectId("5d9ee714b92588845f04cbf9")
    ]
}
>>> db.inventory.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 5 }
>>> db.inventory.find({});
```

Delete Documents

We can specify ***criteria***, or filters, that identify the documents to delete. The ***filters*** use the same syntax as read operations.

To specify ***equality conditions***, use *<field>:<value>* expressions in the query filter document:

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

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

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

Delete All Documents that Match a Condition

The following example ***removes all documents*** from the inventory collection where the status field equals "A":

```
>>> 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: "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" },
... ] );
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9ee86bebb59618c1bde538"),
        ObjectId("5d9ee86bebb59618c1bde539"),
        ObjectId("5d9ee86bebb59618c1bde53a"),
        ObjectId("5d9ee86bebb59618c1bde53b"),
        ObjectId("5d9ee86bebb59618c1bde53c")
    ]
}
>>> db.inventory.deleteMany({ status : "A" })
{ "acknowledged" : true, "deletedCount" : 2 }
>>> db.inventory.find();
{ "_id" : ObjectId("5d9ee86bebb59618c1bde539"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "P" }
{ "_id" : ObjectId("5d9ee86bebb59618c1bde53a"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("5d9ee86bebb59618c1bde53b"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
```

Delete a Single Document

To delete ***at most a single*** document that matches a specified filter (even though multiple documents may match the specified filter) use the *db.collection.deleteOne()* method.

```
>>> 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: "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" },
... ] );
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5d9ee99e2092ea103794b22d"),
        ObjectId("5d9ee99e2092ea103794b22e"),
        ObjectId("5d9ee99e2092ea103794b22f"),
        ObjectId("5d9ee99e2092ea103794b230"),
        ObjectId("5d9ee99e2092ea103794b231")
    ]
}
>>> db.inventory.deleteOne( { status: "D" } )
{ "acknowledged" : true, "deletedCount" : 1 }
>>> db.inventory.find();
{ "_id" : ObjectId("5d9ee99e2092ea103794b22d"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5d9ee99e2092ea103794b22e"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "P" }
{ "_id" : ObjectId("5d9ee99e2092ea103794b230"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("5d9ee99e2092ea103794b231"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
```

Export Documents

We can use the ***mongoexport*** command-line tool to produce a JSON or CSV export of data stored in a MongoDB instance.

Run mongoexport from the system ***command line***, not the mongo shell.

To export a ***specified collection*** to a specified output file from a ***local MongoDB instance*** running on port 27017 use the following command:

mongoexport --collection=<collectionName> --db=<dbName>
--fields=<field1[,field2]> --out=<filename.json(-csv)> --type=<json/csv>

If we want to export from a specific host and port, use the following command

mongoexport --host= <hostAddress> --port=<portNumber>
--collection=<collectionName> --db=<dbName> --fields=<field1[,field2]>
--out=<filename.json(-csv)> --type=<json/csv>

Import Documents

The ***mongoimport*** tool imports content from an Extended JSON, CSV, or TSV export created by ***mongoexport***, or potentially, another ***third-party export tool***.

Run mongoimport from the ***system command line***, not the mongo shell.

Starting in MongoDB 4.2, mongoimport expects import data to be in ***Extended JSON v2.0*** by default.

Details on ***JSON v2.0*** can be found here: ***Extended JSON v2.0***
https://docs.mongodb.com/manual/reference/mongodb-extended-json/

Mongoimport only supports data files that are UTF-8 encoded. Using other encodings will produce errors.

Import a JSON and CSV document

The example in the following shows how to import the document ***restaurants.json***

```
MacBook-Air-di-pietro:Desktop pietroducange$ mongoimport -c=restaurant -d=exer --file=restaurants.json
2019-10-25T18:04:20.366+0200    connected to: mongodb://localhost/
2019-10-25T18:04:20.542+0200    3772 document(s) imported successfully. 0 document(s) failed to import.
MacBook-Air-di-pietro:Desktop pietroducange$
```

The example in the following shows how to import the document ***athlete.csv***

```
MacBook-Air-di-pietro:Desktop pietroducange$ mongoimport -c=athlete -d=exer --file=Dataset/athlete.csv --type=csv --headerline
2019-10-25T17:52:19.422+0200    connected to: mongodb://localhost/
2019-10-25T17:52:22.422+0200    [###############.........] exer.athlete 25.6MB/39.6MB (64.7%)
2019-10-25T17:52:24.054+0200    [########################] exer.athlete 39.6MB/39.6MB (100.0%)
2019-10-25T17:52:24.054+0200    271116 document(s) imported successfully. 0 document(s) failed to import.
```

In order to maintain the insertion order user --maintainInsertionOrder

Suggested Readings

Students are invited to read the official documentation regarding CRUD operations with MongoDB.

The documentation is available at:

https://docs.mongodb.com/manual/tutorial/insert-documents/
https://docs.mongodb.com/manual/tutorial/query-documents/
https://docs.mongodb.com/manual/tutorial/update-documents/
https://docs.mongodb.com/manual/tutorial/remove-documents/
https://docs.mongodb.com/manual/reference/program/mongoimport/
https://docs.mongodb.com/manual/reference/program/mongoexport/

Students are also invited to repeat all the examples on their MongoDB shell.