OPERACIONES CRUD

insertar

db.collection.insertOne()	Inserts a single document into a collection.
db.collection.insertMany()	db.collection.insertMany() inserts *multiple* documents into a collection.
db.collection.insert()	db.collection.insert() inserts a single document or multiple documents into a collection.

```
db.inventory.insertOne(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
```

```
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" } }
])
```

consultar

db.collection.find(query, projection)

Selects documents in a collection or view and returns a cursor to the selected documents.

query: Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
projection: Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter.

Projection
The projection parameter determines which fields are returned in the matching documents. The projection parameter takes a document of the following form:

```
{ field1: <value>, field2: <value> ... }
```

The <value> can be any of the following:

- **1 or true to include the field in the return documents.**
-
- **0 or false to exclude the field.**
-
- **Expression using a Projection Operators.**
  - $
  - $elemMatch
  - $slice

- $meta

For the _id field, you do not have to explicitly specify _id: 1 to return the _id field. The find() method always returns the _id field unless you specify _id: 0 to suppress the field.

Query

Select All Documents in a Collection
To select all documents in the collection, pass an empty document as the query filter parameter to the find method. The query filter parameter determines the select criteria:

```
db.inventory.find( {} )
```

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

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

```
db.inventory.find( { 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> }, ... }
```

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

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

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

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.

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

Specify AND as well as OR Conditions

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

Match an Embedded/Nested Document
To specify an equality condition on a field that is an embedded/nested document, use the query filter document `{ <field>: <value> }` where `<value>` is the document to match.

```
db.inventory.find( { size: { h: 14, w: 21, uom: "cm" } } )
```

Query on Nested Field
To specify a query condition on fields in an embedded/nested document, use dot notation ("field.nestedField").

```
db.inventory.find( { "size.uom": "in" } )
```

```
db.inventory.find( { "size.h": { $lt: 15 } } )
```

```
db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "in", status: "D" } )
```

Array

```
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
   { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },
   { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
   { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
]);
```

Match 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 elements.

The following 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"] } )
```

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

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

Query an Array for an Element
To query if the array field contains at least one element 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 contains the string "red" as one of its elements:

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

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

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

For example, the following operation 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 } } )
```

Query an Array with Compound Filter Conditions on the Array Elements
The following example queries for documents where the dim_cm array contains elements that in some combination 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 } } )
```

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 } } } )
```

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 example queries for all documents where the second element in the array dim_cm is greater than 25:

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

Query an Array by Array Length
Use the $size operator to query 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 } }
```

Query an Array of Embedded Documents

```
db.inventory.insertMany( [
   { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },
   { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },
   { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },
   { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
]);
```

Query for a Document Nested in an Array

The following example selects all documents where an element in the instock array matches the specified document:

```
db.inventory.find( { "instock": { warehouse: "A", qty: 5 } } )
```

Equality matches on the whole embedded/nested document require an exact match of the specified document, including the field order. For example, the following query does not match any documents in the inventory collection:

```
db.inventory.find( { "instock": { qty: 5, warehouse: "A" } } )
```

Specify a Query Condition on a Field Embedded in an Array of Documents
If you do not know the index position of the document nested in the array, concatenate the name of the array field, with a dot (.) and the name of the field in the nested document.

The following example selects all documents where the instock array has at least one embedded document that contains the field qty whose value is less than or equal to 20:

```
db.inventory.find( { 'instock.qty': { $lte: 20 } } )
```

Use the Array Index to Query for a Field in the Embedded Document
Using dot notation, you can specify query conditions for field in a document at a particular index or position of the array. The array uses zero-based indexing.

The following example selects all documents where the instock array has as its first element a document that contains the field qty whose value is less than or equal to 20:

db.inventory.find({ 'instock.0.qty': { $lte: 20 } })

Specify Multiple Conditions for Array of Documents
When specifying conditions on more than one field nested in an array of documents, you can specify the query such that either a single document meets these condition or any combination of documents (including a single document) in the array meets the conditions.

A Single Nested Document Meets Multiple Query Conditions on Nested Fields
Use `$elemMatch` operator to specify multiple criteria on an array of embedded documents such that at least one embedded document satisfies all the specified criteria.

The following example queries for documents where the instock array has at least one embedded document that contains both the field qty equal to 5 and the field warehouse equal to A:

```
db.inventory.find( { "instock": { $elemMatch: { qty: 5, warehouse: "A" } } } )
```

Combination of Elements Satisfies the Criteria
If the compound query conditions on an array field do not use the $elemMatch operator, the query selects those documents whose array contains any combination of elements that satisfies the conditions.

For example, the following query matches documents where any document nested in the instock array has the qty field greater than 10 and any document (but not necessarily the same embedded document) in the array has the qty field less than or equal to 20:

```
db.inventory.find( { "instock.qty": { $gt: 10,  $lte: 20 } } )
```

The following example queries for documents where the instock array has at least one embedded document that contains the field qty equal to 5 and at least one embedded document (but not necessarily the same embedded document) that contains the field warehouse equal to A:

```
db.inventory.find( { "instock.qty": 5, "instock.warehouse": "A" } )
```

Modify the Cursor Behavior

The mongo shell and the drivers provide several cursor methods that call on the cursor returned by the find() method to modify its behavior.

Order Documents in the Result Set

The **sort()** method orders the documents in the result set. The following operation returns documents in the bios collection sorted in ascending order by the name field:

```
db.bios.find().sort( { name: 1 } )
```

sort() corresponds to the ORDER BY statement in SQL.

Limit the Number of Documents to Return

The **limit()** method limits the number of documents in the result set. The following operation returns at most 5 documents in the bios collection:

```
db.bios.find().limit( 5 )
```

limit() corresponds to the LIMIT statement in SQL.

Set the Starting Point of the Result Set

The skip() method controls the starting point of the results set. The following operation skips the first 5 documents in the bios collection and returns all remaining documents:

```
db.bios.find().skip( 5 )
```

Actualizar

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

```
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" }
] );
```

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

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 }
   }
)
```

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

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 } ] }
)
```

Borrar

db.collection.deleteMany()
db.collection.deleteOne()

```
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" },
] );
```

Delete All Documents that Match a Condition
You 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> }, ... }
```

To delete all documents that match a deletion criteria, pass a filter parameter to the deleteMany() method.

```
db.inventory.deleteMany({ status : "A" })
```

Delete Only One Document that Matches a Condition
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.

The following example deletes the first document where status is "D":

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