MongoDB CRUD Operations > Query Documents

Query Documents

This page provides examples of query operations using the `db.collection.find()` method in the `mongo` shell. The examples on this page use the `inventory` collection. To populate the `inventory` collection, run the following in the `mongo` shell:

```
db.inventory.insert([
   { 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" }
]);
```

You can run the operation in the web shell below:

MongoDB Web Shell

Click to connect

Full

Reset

Clear

Select All Documents in a Collection

To select all documents in the collection, pass an empty document `{}` as the query filter parameter to the `db.collection.find()` method. The query filter parameter determines the select criteria:

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

Omitting a query filter parameter is equivalent to specifying an empty query document. As such, the following operation is equivalent to the previous operation:

```
db.inventory.find()
```

These operations correspond to the following SQL statement:

```
SELECT * FROM inventory
```

For more information on the syntax of the method, see `db.collection.find()`.

Specify Equality Condition

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

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

The following example selects from the `inventory` collection all documents where the `status` equals `"D"`:

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

Although you can express this query using the `$or` operator, use the `$in` operator rather than the `$or` operator when performing equality checks on the same field.

The operation corresponds to the following SQL statement:

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

Refer to the Query and Projection Operators document for the complete list of MongoDB query operators.

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.

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

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

The operation corresponds to the following SQL statement:

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

See comparison operators for other MongoDB comparison operators.

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`:

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

The operation corresponds to the following SQL statement:

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

NOTE:
Queries which use comparison operators are subject to Type Bracketing.

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

The operation corresponds to the following SQL statement:

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

NOTE:
MongoDB supports regular expressions `$regex` queries to perform string pattern matches.

Additional Query Tutorials

For additional query examples, see:

- Query on Embedded/Nested Documents
- Query an Array
- Query an Array of Embedded Documents
- Project Fields to Return from Query
- Query for Null or Missing Fields

Behavior

Cursor

The `db.collection.find()` method returns a cursor to the matching documents.

Read Isolation

New in version 3.2.

For reads to replica sets and replica set shards, read concern allows clients to choose a level of isolation for their reads. For more information, see Read Concern.

Additional Methods

The following methods can also read documents from a collection:

- `db.collection.findOne` [1]
- In aggregation pipeline, the `$match` pipeline stage provides access to MongoDB queries.

[1] The `db.collection.findOne()` method also performs a read operation to return a single document. Internally, the `db.collection.findOne()` method is the `db.collection.find()` method with a limit of 1.