MongoDB $eq

Summary: in this tutorial, you'll learn how to use the MongoDB `$eq` operator to specify an equality condition.

Introduction to the MongoDB $eq operator

The `$eq` operator is a comparison query operator that allows you to match documents where the value of a field equals a specified value.

The following shows the syntax of `$eq` operator:

```
{ <field>: { $eq: <value> } }
```

The query is equivalent to the following:

```
{<field>: <value>}
```

MongoDB $eq operator examples

We'll use the following `products` collection for the demonstration:

```
db.products.insertMany([
    { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-0
    { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-
    { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2
    { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020
```

```
    { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("202
])
```

1) Using $eq operator to check if a field equals a specified value

The following example uses the `$eq` operator to query the `products` collection to select all documents where the value of the `price` field equals `899` :

```
db.products.find({
    price: {
        $eq: 899
    }
}, {
    name: 1,
    price: 1
})
```

The query is equivalent to the following:

```
db.products.find({
    price: 899
}, {
    name: 1,
    price: 1
})
```

They both match the following documents:

```
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 }
]
```

2) Using the $eq operator to check if a field in an embedded document equals a value

The following example uses the `$eq` operator to search for documents where the value of the `ram` field in the `spec` document equals `4` :

```
db.products.find({
    "spec.ram": {
        $eq: 4
    }
}, {
    name: 1,
    "spec.ram": 1
})
```

It is equivalent to the following:

```
db.products.find({
    "spec.ram": 4

}, {
    name: 1,
    "spec.ram": 1
})
```

Both of these queries returns the following documents:

```
[
  { _id: 1, name: 'xPhone', spec: { ram: 4 } },
  { _id: 5, name: 'SmartPhone', spec: { ram: 4 } }
]
```

3) Using $eq operator to check if an array element equals a value

The following example uses the `$eq` operator to query the `products` collection to find all documents where the array `color` contains an element with the value `"black"` :

```
db.products.find({
    color: {
        $eq: "black"
    }
}, {
    name: 1,
    color: 1
})
```

It's equivalent to:

```
db.products.find({
    color: "black"

}, {
    name: 1,
    color: 1
})
```

Both queries return the following matching documents:

```
[
  { _id: 1, name: 'xPhone', color: [ 'white', 'black' ] },
  { _id: 2, name: 'xTablet', color: [ 'white', 'black', 'purple' ] }
]
```

4) Using $eq operator to check if a field equals a date

The following example uses the `$eq` operator to select documents in the `widget` collection with the published date is `2020-05-14` :

```
db.products.find({
    releaseDate: {
        $eq: new ISODate("2020-05-14")
    }
}, {
    name: 1,
    releaseDate: 1
})
```

It returned the following document:

```
[
  {
    _id: 4,
    name: 'SmartPad',
    releaseDate: ISODate("2020-05-14T00:00:00.000Z")
  }
]
```

Summary

- Use the `$eq` operator to specify an equality condition.