Introduction to MongoDB $size operator

The $size is an array query operator that allows you to select documents that have an array containing a specified number of elements.

The $size operator has the following syntax:

{ array_field: {$size: element_count} }

Code language: CSS (css)

In this syntax, you specify the element_count after the $size operator to match all documents where the array_field has exact element_count elements.

MongoDB $size operator examples

We'll use the following products collection:

db.products.insertMany([

{ "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate" : ISODate("2011-05-14T00:00:00Z"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 }, "color" : ["white", "black"], "storage" : [64, 128, 256] },

{ "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate" : ISODate("2011-09-01T00:00:00Z"), "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 }, "color" : ["white", "black", "purple"], "storage" : [128, 256, 512] },

{ "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate" : ISODate("2015-01-14T00:00:00Z"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 }, "color" : ["blue"], "storage" : [16, 64, 128] },

{ "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate" : ISODate("2020-05-14T00:00:00Z"), "spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 }, "color" : ["white", "orange", "gold", "gray"], "storage" : [128, 256, 1024] },

{ "_id" : 5, "name" : "SmartPhone", "price" : 599, "releaseDate" : ISODate("2022-09-14T00:00:00Z"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 }, "color" : ["white", "orange", "gold", "gray"], "storage" : [128, 256] },

{ "_id" : 6, "name" : "xWidget", "spec" : { "ram" : 64, "screen" : 9.7, "cpu" : 3.66 }, "color" : ["black"], "storage" : [1024] }

])

Code language: JavaScript (javascript)

1) Using MongoDB $size operator to select documents that have an array containing a number of elements

The following example uses the $size operator to select documents whose color array has two elements:

```
db.products.find({
   color: {
      $size: 2
   }
}, {
   name: 1,
   color: 1
})
```

Code language: CSS (css)

It returned the following document:

```
{ "_id" : 1, "color" : [ "white", "black" ], "name" : "xPhone" }
```

Code language: JSON / JSON with Comments (json)

2) Using MongoDB $size operator with the $or operator example

The following example shows how to use $size operator with the $or operator to select documents whose the color array has one or two elements:

```
db.products.find({
   $or: [{
         color: {
            $size: 1
         }
      },
      {
         color: {
            $size: 2
         }
      }
   ]
```

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

Code language: PHP (php)

It returned the following documents:

```
{ "_id" : 1, "color" : [ "white", "black" ], "name" : "xPhone" }
{ "_id" : 3, "color" : [ "blue" ], "name" : "SmartTablet" }
```

Code language: JSON / JSON with Comments (json)

Summary

- Use the $size operator to select documents that contains an array with a specified number of elements.

Introduction to the MongoDB $all operator

The $all is an array query operator that allows you to find the documents where the value of a field is an array that contains all the specified elements.

The $all operator has the following syntax:

```
{ <arrayField>: { $all: [element1, element2, ...]} }
```

Code language: JavaScript (javascript)

If the array followed the $all operator is empty, then the $all operator matches no documents.

When the array followed the $all operator contains a single element, you should use the contain expression instead:

```
{ <arrayField>: element1 }
```

Code language: HTML, XML (xml)

$all and $and

The following expression that uses the $all operator:

```
{ arrayField: {$all: [element1, element2]} }
```

Code language: JavaScript (javascript)

is equivalent to the following expression that use the $and operator:

{ $and: [{ arrayField: element1}, {arrayField: element2}]}

Code language: JavaScript (javascript)

MongoDB $all operator examples

We'll use the following products collection:

```
db.products.insertMany([
    { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate" : ISODate("2011-05-14T00:00:00Z"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 }, "color" : [ "white", "black" ], "storage" : [ 64, 128, 256 ] },
    { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate" : ISODate("2011-09-01T00:00:00Z"), "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 }, "color" : [ "white", "black", "purple" ], "storage" : [ 128, 256, 512 ] },
    { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate" : ISODate("2015-01-14T00:00:00Z"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 }, "color" : [ "blue" ], "storage" : [ 16, 64, 128 ] },
    { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate" : ISODate("2020-05-14T00:00:00Z"), "spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 }, "color" : [ "white", "orange", "gold", "gray" ], "storage" : [ 128, 256, 1024 ] },
    { "_id" : 5, "name" : "SmartPhone", "price" : 599, "releaseDate" : ISODate("2022-09-14T00:00:00Z"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 }, "color" : [ "white", "orange", "gold", "gray" ], "storage" : [ 128, 256 ] }
])
```

Code language: JavaScript (javascript)

1) Using MongoDB $all operator to match values

The following example uses the $all operator to query the products collection for documents where the value of the color field is an array that includes "black" and "white":

```
db.products.find({
   color: {
      $all: ["black", "white"]
   }
}, {
```

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

Code language: JavaScript (javascript)

It returned the following documents. Notice that the order of elements in the array is not important.

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

Code language: JSON / JSON with Comments (json)

Functionally speaking, the above query is equivalent to the following query that uses the $and operator:

```
db.products.find({
    $and: [
        {color: "black"},
        {color: "white"}
    ]
}, {
    name: 1,
    color: 1
})
```

Code language: JavaScript (javascript)

Summary

- Use the $all operator to select the documents where the value of a field is an array that contains all the specified elements.

Introduction to the MongoDB $elemMatch operator

The $elemMatch is an array query operator that matches documents that contain an array field and the array field has at least one element that satisfies all the specified queries.

The $elemMatch has the following syntax:

{ <arrayField>: {$elemMatch: { <query1>, <query2>, ...} } }

Code language: HTML, XML (xml)

In this syntax:

- First, specify the name of the array field.
- Second, specify a list of queries that you want at least one element in the <arrayField> to meet the query criteria.

Notice that you cannot specify a $where expression or a $text query expression in an $elemMatch.

MongoDB $elemMatch operator examples

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

```
db.products.insertMany([

    { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate" : ISODate("2011-05-14T00:00:00Z"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 }, "color" : [ "white", "black" ], "storage" : [ 64, 128, 256 ] },

    { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate" : ISODate("2011-09-01T00:00:00Z"), "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 }, "color" : [ "white", "black", "purple" ], "storage" : [ 128, 256, 512 ] },

    { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate" : ISODate("2015-01-14T00:00:00Z"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 }, "color" : [ "blue" ], "storage" : [ 16, 64, 128 ] },

    { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate" : ISODate("2020-05-14T00:00:00Z"), "spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 }, "color" : [ "white", "orange", "gold", "gray" ], "storage" : [ 128, 256, 1024 ] },

    { "_id" : 5, "name" : "SmartPhone", "price" : 599, "releaseDate" : ISODate("2022-09-14T00:00:00Z"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 }, "color" : [ "white", "orange", "gold", "gray" ], "storage" : [ 128, 256 ] }

])
```

Code language: JavaScript (javascript)

1) Using the MongDB $elemMatch opeator example

The following example uses the $elemMatch operator to query documents from the products collection:

```
db.products.find({
```

```
    storage: {
        $elemMatch: {
            $lt: 128
        }
    }
}, {
    name: 1,
    storage: 1
});
```

Code language: CSS (css)

It matches the documents where the storage is an array that contains at least one element less than 128:

```
[
  { _id: 1, name: 'xPhone', storage: [ 64, 128, 256 ] },
  { _id: 3, name: 'SmartTablet', storage: [ 16, 64, 128 ] }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $elemMatch operator to select documents that have an array field. And the array field has at least one element that satisfies specified query criteria.