Introduction to the MongoDB $and operator

The $and is a logical query operator that allows you to carry a logical **AND** operation on an array of one or more expressions.

The following shows the syntax of the $and operator:

$and :[{expression1}, {expression2},...]

Code language: PHP (php)

The $and operator returns true if all expressions evaluate to true.

The $and operator stops evaluating the remaining expressions as soon as it finds an expression that evaluates to false. This feature is called short-circuit evaluation.

Implicit AND operator

When you use a comma-separated list of expressions, MongoDB will carry an implicit AND operation:

{ field: { expression1, expression2, ... }

Code language: CSS (css)

MongoDB $and 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 $and operator example

The following example uses the $and operator to select all documents in the products collection where:

- the value in the price field is equal to 899 **and**
- the value in the color field is either "white" or "black"

```
db.products.find({
   $and: [{
      price: 899
   }, {
      color: {
         $in: ["white", "black"]
      }
   }]
}, {
   name: 1,
   price: 1,
   color: 1
})
```

Code language: PHP (php)

It returned the following document:

```
[
 {
   _id: 2,
   name: 'xTablet',
   price: 899,
```

```
    color: [ 'white', 'black', 'purple' ]
  }
]
```

Code language: JavaScript (javascript)

2) Using MongoDB $and operator with the same field

The following example uses the $and operator to select all documents where:

- the price field value equals 699 and
- the price field value exists

```
db.products.find({
    $and: [{
        price: 699
    }, {
        price: {
            $exists: true
        }
    }]
}, {
    name: 1,
    price: 1,
    color: 1
})
```

Code language: PHP (php)

Output:

```
[
  {
    _id: 4,
    name: 'SmartPad',
    price: 699,
```

```
    color: [ 'white', 'orange', 'gold', 'gray' ]
  }
]
```

Code language: JavaScript (javascript)

The following example uses the implicit AND operator and returns the same result:

```
db.products.find({
    price: {
        $eq: 699,
        $exists: true
    }
}, {
    name: 1,
    price: 1,
    color: 1
})
```

Code language: CSS (css)

Summary

- Use the MongoDB $and operator to perform a logical AND operation.
- MongoDB performs an implicit AND operation if you specify a comma-separated list of expressions.

Introduction to the MongoDB $or operator

The $or is a logical query operator that carries a logical **OR** operation on an array of one or more expressions and selects the documents that satisfy at least one expression.

Here is the syntax of the $or operator:

```
$or:[{expression1}, {expression2},...]
```

Code language: PHP (php)

MongoDB $or 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 $or operator example

The following example uses the $or operator to select all documents in the products collection where the value in the price field equals 799 or 899:

```
db.products.find({
   $or: [{
      price: 799
   }, {
      price: 899
   }]
}, {
   name: 1,
   price: 1
})
```

Code language: PHP (php)

It returned the following documents:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 }
]
```

Code language: JavaScript (javascript)

Since this example checks equality for the same price field, you should use the $in operator instead:

```
db.products.find({
    price: {
        $in: [799, 899]
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

2) Using MongoDB $or operator to select documents where the value of a field is in a range

The following example uses the $or operator to select all documents where the price is less than 699 or greater than 799:

```
db.products.find({
    $or: [
        { price: {$lt: 699} },
        { price: {$gt: 799} }
    ]
}, {
    name: 1,
```

```
    price: 1
})
```

Code language: PHP (php)

Output:

```
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

Code language: JavaScript (javascript)

Summary

- Use the MongoDB $or operator to perform a logical OR operation on a list of expressions and select documents that satisfy at least one expression.

Introduction to the MongoDB $not operator

The following shows the syntax of the $not operator:

```
{ field: { $not: { <expression> } } }
```

Code language: CSS (css)

The $not operator is a logical query operator that performs a logical NOT operation on a specified <expression> and selects documents that do not match the <expression>. This includes the documents that do not contain the field.

MongoDB $not 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 $not operator to select documents

The following example shows how to use the $not operator to find documents where:

- the price field is not greater than 699.
- do not contain the price field.

```
db.products.find({
   price: {
      $not: {
         $gt: 699
      }
   }
}, {
   name: 1,
   price: 1
})
```

Code language: CSS (css)

It returned the following documents:

[

```
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 },
  { _id: 6, name: 'xWidget' }
]
```

Code language: JavaScript (javascript)

Notice that the { $not: { $gt: 699 } } is different from the $lte operator. The { $lte : 699 } returns documents where the price field exists and its value is less than or equal to 699.

The following example uses the $lte operator:

```
db.products.find({
    price: {
        $lte: 699
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

It returned the following documents:

```
[
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

Code language: JavaScript (javascript)

As you can see clearly from the output, the result of the query that uses the $lte operator does not include the documents where the price field does not exist.

2) Using MongoDB $not operator to select documents based on expressions

The following example uses the $not operator to select documents from the products collection where the value of the field does not match the regular expression /^smart+/i:

```
db.products.find({
   name: {
      $not: /^Smart+/
   }
}, {
   name: 1
})
```

Code language: CSS (css)

The regular expression /^Smart+/ matches any string that starts with the string smart and is followed by any number of characters.

The query returns the following documents:

```
[
 { _id: 1, name: 'xPhone' },
 { _id: 2, name: 'xTablet' },
 { _id: 6, name: 'xWidget' }
]
```

Code language: JavaScript (javascript)

Summary

- Use the MongoDB $not operator to perform a logical NOT operation on a specified <expression> and selects documents that do not match the <expression>.

Introduction to the MongoDB $nor operator

The $nor is a logical query operator that allows you to perform a logical NOR operation on a list of one or more query expressions and selects documents that fail all the query expressions.

The syntax of the $nor operator is as follows:

```
{ $nor: [ { <expression1> }, { <expression2> },...] }
```

MongoDB $nor 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)

The following example uses the $nor operator to select documents from the products collection:

```
db.products.find({
   $nor :[
      { price: 899},
      { color: "gold"}
   ]
}, {
   name: 1,
   price: 1,
   color: 1
})
```

Code language: PHP (php)

It returns documents where:

- the value is the price field is not 899
- and the color array does not have any "gold" element.

including the documents that do not contains these fields.

It returned the followig documents:

{ "_id" : 1, "name" : "xPhone", "price" : 799, "color" : ["white", "black"] }

{ "_id" : 6, "name" : "xWidget", "color" : ["black"] }

Code language: JSON / JSON with Comments (json)

Summary

- Use the MongodB $nor operator to perform a logical NOR operation on a list of query expressions and select documents that fail all the query expressions.