

MongoDB教育訓練-04

Array 查詢

Querying Values in Arrays

When the field you are querying is an Array:

You will match if the query **matches any member**

Or if it exactly **matches the whole array, including the order**

```
MongoDB > db.fun.insertOne({ "name": "John", hobbies: ["cars", "robots", "gardens"] })
{"acknowledged":true }

//Find by ANY member of the array
MongoDB> db.fun.find({hobbies:"gardens"})

{"_id":ObjectId("5ca4bbcea2dd94ee58162a68"),"name":"John","hobbies":["cars","robots","gardens"]}

//Find by matching the array itself
MongoDB> db.fun.find({hobbies:["cars","robots","gardens"]})

{"_id":ObjectId("5ca4bbcea2dd94ee58162a68"),"name":"John","hobbies":["cars","robots","gardens"]}

//Not found - order doesn't match
MongoDB> db.fun.find({hobbies:["robots","cars","gardens"]})

//Not found - missing element
MongoDB> db.fun.find({hobbies:["cars","robots"]})
```

Array 查詢 操作

- \$all
- \$size(少用無法使用index)
- \$element 元素滿足

Array specific query operators

MongoDB has operators designed specifically for querying against arrays

- \$all
- \$size
- \$elemMatch

Why is there no \$any operator?

```
MongoDB > db.fun.insertOne({ "name": "John", hobbies: ["cars", "robots", "gardens"] })
{"acknowledged":true }

MongoDB > db.fun.find({ hobbies : { $all : ["robots","cars"] } })
{"_id" : ObjectId("609bc8a6cf0c3aa225ce91e6"), "name" : "John", "hobbies" : [ "cars", "robots", "gardens" ] }

MongoDB > db.fun.find({ hobbies : { $all : ["robots","cars","bikes"] } })
//No result as bikes is not in the array

MongoDB > db.fun.find({ hobbies : { $size : 3 } })
{"_id" : ObjectId("609bc8a6cf0c3aa225ce91e6"), "name" : "John", "hobbies" : [ "cars", "robots", "gardens" ] }

MongoDB > db.fun.find({ hobbies : { $size : 4 } })
//No Result - array is 3 long
```

```
age : { $gt : 18, $lt: 30 }
```

```
#age >= 18 && age <= 30, array 其中一個值滿足所有條件  
age : { $elemMatch { $gt : 18, $lt: 30 } }
```

A surprising array query

If we have a query like this

```
{ age : { $gt : 18 , $lt: 30 } }
```

We instinctively expect it to match if age is between 18 and 30 - but what if age is an array?

```
{ age : [ 40, 10, 5 ] }
```

Would it match this for example - if so why?

Using \$elemMatch

To constrain it to a single element we need a query like this

```
{ age : { $elemMatch: { $gt : 18 , $lt: 30 } } }
```

You can read **\$elemMatch** as saying - has an Element that would match this supplied query - in this case { \$gt : 18 , \$lt: 30 }

Not using \$elemMatch when required is a common source of errors in MongoDB queries.

Exercise \$elemMatch

In "sample_restaurants.restaurants" we have hygiene ratings and dates of inspection.

I'd like to avoid anywhere that has had a rating of C since 2013. Better to eat at consistently clean restaurants.

I might incorrectly try this to find restaurants to avoid, It will find 2,675 :

```
db.restaurants.find({
  "grades.grade": "C",
  "grades.date": {$gt: ISODate("2013-12-31")}
})
```

But it would incorrectly find the record shown - which hasn't had a C since 2011.

What Query should I use and how many restaurants do I need to avoid?

```
MongoDB> db.restaurants.findOne()
{
  "_id" : ObjectId("5eb3d668b31de5d588f4292a"),
  "address" : {
    "building" : "2780",
    "coord" : [
      -73.98241999999999,
      40.579505
    ],
    "street" : "Stillwell Avenue",
    "zipcode" : "11224"
  },
  "borough" : "Brooklyn",
  "cuisine" : "American",
  "grades" : [
    {
      "date" : ISODate("2014-06-10T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2013-06-05T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2012-04-13T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2011-10-12T00:00:00Z"),
      "grade" : "C",
      "score" : 12
    }
  ],
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
```

排序

Sorting Results

Often we want our results to be in a specific order.

We use the sort() cursor modifier for this.

It takes an object listing fields in the order to sort and sort direction.

```
MongoDB> let rnd = (x)=>Math.floor(Math.random()*x)
MongoDB> for(let x=0;x<100;x++) {
  db.scores.insertOne({ride:rnd(40),swim:rnd(40),run:rnd(40)})}
//Unsorted
MongoDB> db.scores.find({}, {_id:0})
{ "ride" : 5, "swim" : 11, "run" : 11 }
{ "ride" : 0, "swim" : 17, "run" : 12 }
{ "ride" : 17, "swim" : 2, "run" : 2 }

//Sorted by ride increasing
MongoDB> db.scores.find({}, {_id:0}).sort({ride:1})
{ "ride" : 0, "swim" : 38, "run" : 10 }
{ "ride" : 1, "swim" : 37, "run" : 37 }
{ "ride" : 1, "swim" : 30, "run" : 20 }

//Sorted by swim increasing then ride decreasing
MongoDB> db.scores.find({}, {_id:0}).sort({swim:1, ride:-1})
{ "ride" : 31, "swim" : 0, "run" : 14 }
{ "ride" : 11, "swim" : 0, "run" : 14 }
{ "ride" : 30, "swim" : 1, "run" : 34 }
{ "ride" : 21, "swim" : 1, "run" : 3 }
```

Combined Query Exercise

What is the largest company (by number of employees) in the "sample_training.companies" collection that has fewer than 200 employees?

Write a query that prints out only the company name and number of employees.

For this you will need to use `find()` with a query and projection

You may also need to use some of `sort()` `skip()` and `limit()`

_id: 0 #不顯示id

```
type -- for more
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.companies.find({number_of_employees: {$lt: 200}}, {_id: 0, name: 1, number_of_employees: 1}).sort({number_of_employees: -1})
[
  { name: 'uShip', number_of_employees: 190 },
  { name: 'Squarespace', number_of_employees: 190 },
  { name: 'LearningMata', number_of_employees: 190 },
  { name: 'Cmed', number_of_employees: 188 },
  { name: 'Lyft', number_of_employees: 180 },
  { name: 'Lyft', number_of_employees: 180 },
  { name: 'Sunrun', number_of_employees: 180 },
  { name: 'Ipswitch', number_of_employees: 180 },
  { name: 'NetScaler', number_of_employees: 180 },
  { name: 'Courion Corporation', number_of_employees: 175 },
  { name: 'Vocalocity', number_of_employees: 175 },
  { name: 'Jimdo', number_of_employees: 170 },
  { name: 'Natta', number_of_employees: 170 },
  { name: 'Chelsio Communications', number_of_employees: 170 },
  { name: 'eModeration', number_of_employees: 170 },
  { name: 'Foursquare', number_of_employees: 170 },
  { name: 'Tuenti Technologies', number_of_employees: 170 },
  { name: 'Hifn', number_of_employees: 166 },
  { name: 'Five9', number_of_employees: 165 },
  { name: 'Digital Marketing Solutions', number_of_employees: 165 }
]
```

更新

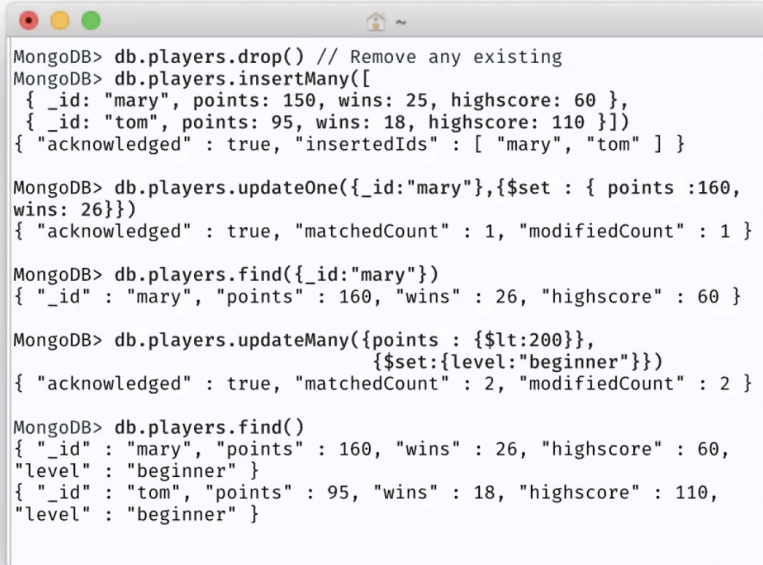
- updateOne : 更新查詢第一筆
- updateMany: 更新所有

Updating Documents

We modify documents in MongoDB using either `updateOne` or `updateMany`

`updateOne(query, change)`
will change only the first matching document

`updateMany(query, change)`
will change all matching Documents.



```
MongoDB> db.players.drop() // Remove any existing
MongoDB> db.players.insertMany([
  { _id: "mary", points: 150, wins: 25, highscore: 60 },
  { _id: "tom", points: 95, wins: 18, highscore: 110 }])
{ "acknowledged" : true, "insertedIds" : [ "mary", "tom" ] }

MongoDB> db.players.updateOne({_id:"mary"},{$set : { points :160,
wins: 26}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

MongoDB> db.players.find({_id:"mary"})
{ "_id" : "mary", "points" : 160, "wins" : 26, "highscore" : 60 }

MongoDB> db.players.updateMany({points : {$lt:200}},
                                {$set:{level:"beginner"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }

MongoDB> db.players.find()
{ "_id" : "mary", "points" : 160, "wins" : 26, "highscore" : 60,
  "level" : "beginner" }
{ "_id" : "tom", "points" : 95, "wins" : 18, "highscore" : 110,
  "level" : "beginner" }
```

更新欄位

Describing a Mutation

`updateOne(query, mutation)`

Mutation is an Object describing the changes to make to each record. Values can be explicitly set or changed relative to their current value or some external values.

The format is

```
{ operator1 : { field1: value, field2: value},
  operator2 : { field3: value, field4: value } }
```

The \$unset operator

In MongoDB we can remove a field from a document entirely.

This makes it logically equal to null but takes no storage.

\$unset takes an object with the fields to remove and a value of 1 or true.

```
MongoDB> db.bands.insertOne({ _id: "genesis", Singer: "Peter",  
Drums: "Phil",Keyboard:"Tony",Guitar:"Mike"})  
{ "acknowledged" : true, "insertedId" : "genesis" }  
MongoDB> db.bands.findOne()  
{  
  "_id" : "genesis",  
  "Singer" : "Peter",  
  "Drums" : "Phil",  
  "Keyboard" : "Tony",  
  "Guitar" : "Mike"  
}  
MongoDB> db.bands.updateOne({ _id : "genesis" },  
{ $unset: { "Singer": true } })  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
MongoDB> db.bands.findOne()  
{  
  "_id" : "genesis",  
  "Drums" : "Phil",  
  "Keyboard" : "Tony",  
  "Guitar" : "Mike"  
}
```

```
# 加減  
$inc : +  
# 乘  
$mul: *
```

Relative numeric updates \$inc and \$mul

\$inc and \$mul modify numeric value relative to its current value.

\$inc changes it by adding a value to it - the value may be negative.

\$mul changes it by multiplying it by a value, which may be less than 1

```
MongoDB> db.employees.insertOne({name: "Carol", salary: 10000,  
bonus: 500})  
{ "acknowledged" : true, "insertedId" : ObjectId("") }  
  
//Give everyone a 10% payrise  
MongoDB> db.employees.updateMany({},{$mul : {salary: 1.1}})  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
  
MongoDB> db.employees.find({},{_id:0})  
{ "name" : "Carol", "salary" : 11000, "bonus" : 500 }  
  
//Give Carol 1000 more bonus too  
MongoDB> db.employees.updateOne({name:"Carol"},  
{$inc:{bonus:1000}})  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
  
MongoDB> db.employees.find({},{_id:0})  
{ "name" : "Carol", "salary" : 11000, "bonus" : 1500 }
```

```
#大於更新  
$max  
#小於才更新  
$min
```

Relative value operators \$max and \$min

\$max and \$min may or may not modify a field depending on its current value.

They only change if the value given is larger (or smaller) than the current value.

```
MongoDB > db.gamescores.insertOne({name: "pacman", highscore: 10000 })
{ "acknowledged" : true, "insertedId" : ObjectId("") }

//This finds the record but does not change it as 9000 < 10000
MongoDB > db.gamescores.updateOne({name:"pacman"},{$max: { "highscore": 9000}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }

//This finds and changes highscore as 12000 > 10000
MongoDB > db.gamescores.updateOne({name:"pacman"},{$max: { "highscore": 12000}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

MongoDB Enterprise > db.gamescores.find({})
{ "_id" : ObjectId("609bf0f8cf0c3aa225ce9314"), "name" : "pacman", "highscore" : 12000 }
```

Exercise: Updates

Using the inspections collection (sample_training.inspections) complete the following exercise.

Exercise: Pass Inspections

In the inspections collection in the sample database, let's imagine that we want to do a little data cleaning. We've decided to eliminate the "Completed" inspection result and use only "No Violation Issued" for such inspection cases. Please update all inspections accordingly.

Exercise: Set fine value

For all inspections that fail, set a fine value of 100.

Exercise: Increase fines in ROSEDALE

Update all inspections done in the city of "ROSEDALE", for failed inspections, raise the "fine" value by 150.

```
#q1
db.inspections.updateMany({result:"Completed"},{$set:{result:"No Violation Issued"}})

#q2
db.inspections.updateMany({result:'Fail'},{$set:{fine:100}})

#q3
db.inspections.updateMany({"address.city": "ROSEDALE"}, {$inc: {fine: 150}})
```

Array 新增移除

```
$pop : 1 (最後元素)
$pop : -1 (第一個元素)

# 檢查不存在才新增
$addToSet
```

Basic Array update operators - \$push and \$pop

MongoDB allows us to modify an array that is in a document without having to read and replace the entire array.

This is important to prevent overwriting other users changes.

We can add and remove items in a number of ways, the simplest are \$push and \$pop.

```
MongoDB> db.playlists.insertOne(
{name: "funky",
 tracks : [
  { artist:"queen",track:"Liar"},
  {artist:"abba",track:"Chiquitita"},
 ]})

{ "acknowledged" : true,"insertedId" : ObjectId("")}

MongoDB> db.playlists.updateOne({name:"funky"},
{ $push : {tracks : { artist: "AC/DC", track: "Hells Bells"} }})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

MongoDB> db.playlists.find({}).pretty()
{ "_id" : ObjectId(""),
  "name" : "funky",
  "tracks" : [
    { "artist" : "queen", "track" : "Liar" },
    { "artist" : "abba", "track" : "Chiquitita" },
    { "artist" : "AC/DC", "track" : "Hells Bells" } ]
}
```

Basic Array update operators - \$push and \$pop

\$pop either removes the last item from an array, or the first if we set the value to -1.

```
MongoDB> db.playlists.find({},{_id:0}).pretty()
{ "name" : "funky",
  "tracks" : [ { "artist" : "queen", "track" : "Liar" },
               { "artist" : "abba", "track" : "Chiquitita" },
               { "artist" : "AC/DC", "track" : "Hells Bells" } ]
}

MongoDB> db.playlists.updateOne({name:"funky"},
{ $pop: {tracks:-1}})

MongoDB> db.playlists.find({},{_id:0}).pretty()
{ "name" : "funky",
  "tracks" : [ { "artist" : "queen", "track" : "Liar" },
               { "artist" : "abba", "track" : "Chiquitita" } ]
}

MongoDB> db.playlists.updateOne({name: "funky"},
{ $pop: {tracks: -1}})

MongoDB> db.playlists.find({},{_id:0}).pretty()
{ "name" : "funky",
  "tracks" : [ { "artist" : "abba", "track" : "Chiquitita" } ]
}
```

Basic Array update operators - \$pull and \$pullAll

\$pull and \$pullAll allow us to selectively remove things from a list based on them matching either a given value or a query.

\$pullAll allows to specify multiple specific values to remove.

```
MongoDB> db.playlists.drop()

MongoDB> db.playlists.insertOne(
  {name: "funky",
   tracks : [
     { artist:"queen",track:"Liar"},
     { artist:"abba",track:"Chiquitita"},
     { artist:"queen",track:"Under Pressure"},
     { artist:"AC/DC",track:"Hells Bells"},
   ]})

MongoDB> db.playlists.updateOne({name:"funky"},
  { $pull : { tracks : { artist: "queen" }}})

MongoDB> db.playlists.find({},{_id:0}).pretty()
{ "name" : "funky",
  "tracks" : [{"artist" : "abba", "track" : "Chiquitita" },
               {"artist" : "AC/DC", "track" : "Hells Bells" }]}
```

Array Operators - \$addToSet

\$addToSet Appends an element to an array only if it is not already present.

```
MongoDB > db.sports.insertOne( {name: "fives",
  players : ["Ravi", "Jon", "Niyati", "John" ]})
{"acknowledged" : true,"insertedId" : ObjectId("")}

MongoDB > db.sports.find({name:"fives"},{_id:0})
{ "name" : "fives", "players" : [ "Ravi", "Jon", "Niyati", "John" ] }

//Ravi is not added, as he is already there
MongoDB > db.sports.updateOne({name:"fives"},{ $addToSet : {
  players: "Ravi"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }

//Kim is added as they are not in the array currently
MongoDB > db.sports.updateOne({name:"fives"},{ $addToSet : {
  players: "Kim"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

MongoDB > db.sports.find({name:"fives"},{_id:0})
{ "name" : "fives", "players" : [ "Ravi", "Jon", "Niyati", "John",
  "Kim" ] }
```

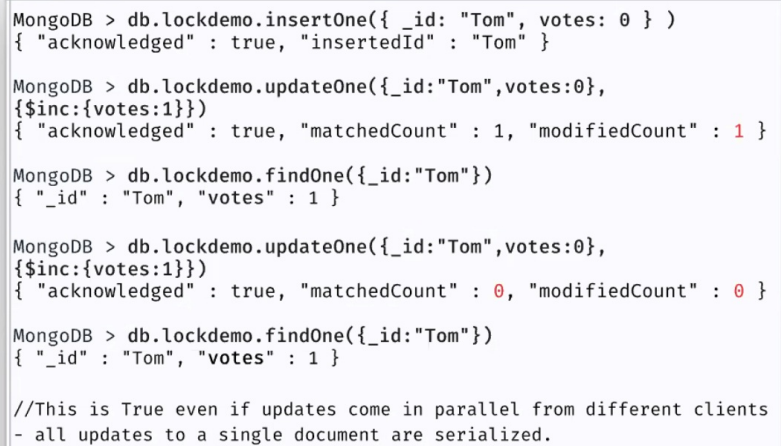
基本mongoDB 基本無lock，每次更新檢更新序號，是否為最新，若不是最新，嘗試合併最新，重新更新

Updating, Locking and Concurrency.

If two processes attempt to update the same document at the same time they are serialised.

The conditions in the query must always match for the update to take place.

In the example - if the two updates take place in parallel - the result is the same.



```
MongoDB > db.lockdemo.insertOne({_id: "Tom", votes: 0 } )
{ "acknowledged" : true, "insertedId" : "Tom" }

MongoDB > db.lockdemo.updateOne({_id:"Tom",votes:0},
{$inc:{votes:1}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

MongoDB > db.lockdemo.findOne({_id:"Tom"})
{ "_id" : "Tom", "votes" : 1 }

MongoDB > db.lockdemo.updateOne({_id:"Tom",votes:0},
{$inc:{votes:1}})
{ "acknowledged" : true, "matchedCount" : 0, "modifiedCount" : 0 }

MongoDB > db.lockdemo.findOne({_id:"Tom"})
{ "_id" : "Tom", "votes" : 1 }

//This is True even if updates come in parallel from different clients
- all updates to a single document are serialized.
```

🕒 修訂版本 #6

★ 由 treeman 建立於 16 🕒 Q🕒 G🕒 2021 22:36:07

🔧 由 treeman 更新於 5 🕒 2023 10:14:59