

增+查

use shop 使用collections

db.products.insert()/insertOne()/insertMany([data],[ordered:false]),注意_id不能重复, - 且insertmany报错不会回滚已经插入的数据

ordered:false, 意味着啥

db.products.insertOne({"name":"Ben's book", price:12.99})

db.authors.insertMany([{name: "Ben", age:22}, {name: "Ben", age:22}])

db.products.find({"name":"Ben"}, {_id:0, "-age":1}), 要筛选啥, 要的列扣1, _id是自带的索引

db.products.find({age: {\$gt:30}}), 还有\$eq,\$ne,\$gt,\$gte,\$lt,\$lte,\$eq可以接null

db.products.find({\$expr:{\$gt:["\$volume","\$-target"]}}), volumn和target两个字段, 比大小, 必须用\$expr

db.movies.find({"rating.average":{\$gt:-7}}), embedded需要用.和双引号

#embedded+多重: db.users.find({hobbies:{\$title:'sports',frequency:4}})

db.movies.find({genre: ["Drama"]}), genre-有且仅有Drama, 如果没有方括号就是包括Drama

#就要这两个值, 且有序: db.boxoffice.find({"genre": ['drama','action']})

#好几个值都要, 可以再有多余的, 且无序: db.boxoffice.find({"genre":{\$all:['drama','action']}})

#找到第一条, db.movies.findOne()

find函数in符号: {\$in,\$nin:[30]/[30,40]}相当于sql的in not in

find函数or/and/not: db.movies.find({\$or: [{"rating.average":{\$lt:5}}, {"rating.average":{\$gt:9.3}}]})

find函数, \$and函数会把doc里面所有的doc都遍历一遍, 可能一个doc满足条件A, 一个满足B, 就能被选入

find函数, \$elemMatch 是doc同时满足才能入选

find函数exists: \$exists: true

增+查 (cont)

find函数判断类型: {\$type: "string"/"double"/"number"}

find函数, summary: "musical"可以找到所有summary里面包含musical的document

find函数, summary: {\$regex: /musical/}可以找到所有包含summary里面包含musical的document

#doc里的hobbies里面有两个子doc db.users.find({"hobbies":{\$size:2}})

在find写完后.count()可以统计数量

去重: db.students.distinct("class", {grade: "A" })

区别, 优势

MongoDB允许存储的数据部分, 甚至完全不同, 而sql有固定的列

Mongo一对一的关系 (比如病人对一张病例单)用embedded存储, 而sql会存在不同的表里

Mongo一对一的关系 (一个名字既有薪水也有车辆信息), 如果两个表都单独做分析, 就单独存储. 如果都要分析, 那也可以join

Mongo一对多的关系: 数据里有别的collection的document的id

Mongodb多对多: 一个document里面存储多个_id

db.books.aggregate([{\$lookup:{\$from: "authors", localField:"authors", foreignField:"_id", as: "creators"} }])

把authorjoin进book里面: from是外表, localfield是本表的列, foreignfield是外表的列, as是本表新引进来数据的名字

改

db.customers.update({}, {\$set: {orders: []}})

CRUD 运算都是原子计算, 不会回滚

insertOne()

删

db.patients.deleteMany({"history.disease":"flu"})

聚合函数

db.students.aggregate([{\$group: {_id: "\$class", // 使用 \$group 按班级分组, distinctGrades: { \$addToSet: "\$grade" } } // 使用 \$addToSet 获取每个班级的不同年级 }],

{ \$project: { _id: 0, // 不显示默认的 _id 字段, class: "\$_id", // 重新命名字段, distinctGrades: 1 // 保留 distinctGrades 字段 } }

])

sort/projections/slices

#降序db.movies.find({}).sort({'rating.average':-1,runtime:1})/1是升序

#跳过和限制打印: find().sort().skip(2).limit(5)

#投影 find(),{url:1,name:1,_id:0} id是强制显示的, 除非你取消

#slice db.movtes.find({"rating.average":{\$gt:9}},{genres:{\$slice:[1,2]}, name:1), genre是一个数组, 取第二第三个element, 再取name

数据格式转化: \$convert: { input: <string>, to: "date"} converts a string to ISODate

\$isoWeekYear extracts the year value of a date field

update

#先用find函数查, 再用updateOne, updateMany去修改

db.users.updateOne({name: " Cammy Soh"}, {\$set: [hobbies: [{title: "cooking", frequency: 7}]}]) 把cammy的hobbies 属性给删除, 改成这个新值

db.users.updateOne({name:"Cammy Soh"}, {\$set:[age:20, phoneNo: 97332212]})

db.users.updateMany({"hobbies .title":"sports"}, \$set:fisSport:true))

db.users.updateOne({name:"Jayden Choi"}, {\$inc:{age: 1}})



By **cgееееh**
cheatography.com/cgeeeeh/

Not published yet.
Last updated 30th September, 2023.
Page 1 of 2.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>

update (cont)

报错，因为不能同时修改两个值：db.user.updateOne({ name:"BAA" }, {\$inc:{age:1}, \$set:{age:10}})

```
db.users.updateOne({ name:"abc" }, {$min/$max/$mul: {age:15}})
```

```
db.users.updateMany({isSport: true}, {$unset: {phone: " "}})
```

```
db.users.updateMany({}, {$rename: {age: "-totalage"}})
```

updateMany还有一个{upsert=true}函数，默认为false

```
db.users.updateMany({ hobbies:{$elemMatch: {"frequency":{$gte:5}, title:"sports"}}, {$set: {"hobbies.$": {title:"sports", frequency:15}}})
```

\$是指对那些满足筛选条件的第一个document更改值，\$[]是对所有的更改值

```
db.users.updateMany({"hobbies.frequency":{$gt:3}},{ $set: {"hobbies.$goodFrequency":True}})新建一个列
```

```
db.users.updateMany({},{$set: {"hobbies.$[el].goodFrequency":true}}, {arrayFilters: [{"el.frequency":{$gt:3}]})
```

最后一个参数告诉我们，第一个参数筛选的结果中，也不是所有的都需要改

group

```
db.users.aggregate([{$match:{gender:"female"}},
```

```
 {$group: { _id: {isoWeekYear: { $isoWeekYear: "$date" }},incidentType: "$incidentType", totalAge:{$sum: "$dob.age"}}}
```

```
 {$sort: {totalPersons:-1}}。1是asc，-1desc，可以用之前groupby里面选中的列
```

```
 {$project: { _id:0, gender:1, fullName:{$concat:["hello,world"]}}}
```

{addfields:}首先，层层递进地加列，不要一次性计算很多，第二，如果要计算到之前加进去的列，需要新建一个addfields来算

字符串函数

```
{$toUpper,"$name.first"},{$toLower:"$name.last"}
```

\$substrCP:["\$name.first", 0, 1] extracts starting from the 0 (1st) char for 1 char -> "B"

\$substrCP:["\$name.first", 0, 2] extracts starting from the 0 (1st) char for 2 char -> "Be"

\$subtract:[int1,int2] perform subtraction int1 - int2

\$strLenCP: <string> returns the length (number of char) of a string

projection中的array操作



By cgeeeeh

cheatography.com/cgeeeeh/

Not published yet.

Last updated 30th September, 2023.

Page 2 of 2.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>