复习：查找

find(query,field)

findOne(query,field)

查找操作符:

比较：$eq $lt $gt $gte $lte $in $nin $ne

逻辑：$and $or $not $nor

数组：$all $size $slice

其他：$exists $mod $type

查找函数：

pretty() limit() skip() sort() count() dictinct()

删除操作：remove(query,justOne)

修改操作：update(query,update,upsert,multi)

##################################################################

补充：通过find查找结果，可以使用序列号获取具体某一项

e.g. 获取第三条文档：

> db.class0.find({},{\_id:0})[2]

修改操作符（修改器）

$set

修改一个域的值

修改Lily的年龄

> db.class0.update({name:'Lily'},{$set:{age:27}})

增加一个域

e.g为jarry增加sex域

> db.class0.update({name:'Jarry'},{$set:{sex:'m'}})

$unset

删除一个域

e.g. 删除Abby的sex域

> db.class0.update({name:'Abby'},{$unset:{sex:''}})

$rename

修改域的名称

e.g. 将 sex域名改为gender

> db.class0.update({},{$rename:{sex:'gender'}},false,true)

$setOnInsert

如果使用update插入了文档，则将该修改器内容作为插入文档的一部分

e.g.如果插入了新文档则setoninsert内容也会作为新的文档一部分

> db.class0.update({name:'idom'},{$set:{age:18},$setOnInsert:{gender:'m',tel:'123456'}},true)

$inc

加法修改器

e.g. 所有人年龄加1

> db.class0.update({},{$inc:{age:1}},false,true)

\*参数可以是正数负数整数小数

$mul

乘法修改器

e.g. > db.class0.update({},{$mul:{age:2}},false,true)

\*参数可以是正数负数整数小数

$min

如果筛选文档的指定域值小于min不修改，大于则修改为这个指定min值

e.g. 如果age大于20则改为20

> db.class0.update({name:'bai'},{$min:{age:20}})

$max

如果筛选文档的指定域值大于max值则不修改，小于max值则改为max值

e.g. 如果age小于22则改22

> db.class0.update({name:'bai'},{$max:{age:22}})

数组修改器：

$push

向数组中添加一项

e.g.给小红score数组添加一项91

> db.class1.update({name:'小红'},{$push:{score:91}})

$pathAll

向数组中添加多项

e.g.

> db.class1.update({name:'小乔'},{$pushAll:{score:[91,10]}})

$pull

从数组中删除一项

e.g.从数组中删除一项

> db.class1.update({name:'小红'},{$pull:{score:78}})

$pullAll

> db.class1.update({name:'小乔'},{$pullAll:{score:[91,10]}})

$each

对多个值逐个进行操作

分别插入99，10

> db.class1.update({name:'小乔'},{$push:{score:{$each:[99,10]}}})

$position

指定插入位置

e.g.将67插入数组1号位

> db.class1.update({name:'小名'},{$push:{score:{$each:[67],$position:1}}})

$sort

数组排序

e.g.将所有score的数降序排序

> db.class1.update({},{$push:{score:{$each:[],$sort:-1}}},false,true)

$pop

弹出一项

1表示弹出最后一项，-1弹出第一项

e.g.删除小乔的最后一项

> db.class1.update({name:'小乔'},{$pop:{score:1}})

$addToSet

向数组中添加一项，但是不能添加 重复的内容

e.g. 如果数组没有99，则添加

> db.class1.update({name:'小刚'},{$addToSet:{score:99}})

时间数据类型

mongo中存储时间大多为ISODate

存储当前时间方法

1. new Date() 自动生成当前时间

e.g.

> db.class2.insert({book:'python入门',date:new Date()})

2. ISODate() 自动生成当前时间

e.g.

> db.class2.insert({book:'python精通',date:ISODate()})

3. Date() 将系统时间转换为字符串

e.g.

> db.class2.insert({book:'python疯狂',date:Date()})

===》字符串类型

指定时间

ISODate()

功能：生成mongo标准时间类型数据

参数：如果不传参默认为当前时间

传参表示指定时间

“2018-01-01 12：12：12”

“20180101 12：12：12”

“20180101”

e.g.

> db.class2.insert({book:'python崩溃',date:ISODate('2018-09-20 11:50:00')})

时间戳：

时间节点

valueOf()

获取摸个时间的时间戳

e.g.

> db.class2.insert({book:'python涅槃',date:ISODate().valueOf()})

Null类型

值null

1. 如果某个域没有值可以赋值为null

e.g.

> db.class2.insert({book:'python死去活来',price:null})

1. 可以查找某个域不存在的情况

e.g.如果date域 不存在也能find到

> db.class2.find({date:null})

Object(内部文档)

文档内部某个域的值还是一个文档数据则这个文档就是内部文档类型数据

e.g.

> db.class3.find({'book.title':'狂人日记'})

e.g.

> db.class3.update({'book.title':'围城'},{$set:{'book.price':48.8}})

通过数组下标直接操作某一项

e.g. 通过数组下标引用第一项进行查找

> db.class1.find({'score.0':{$gt:90}},{\_id:0})

e.g.

> db.class1.update({name:'小刚'},{$set:{'score.1':80}})

索引：

指建立指定键值及所在文档存储位置的 对照清单，使用索引可以方便我们进行快速查找，减少遍历次数，提高查找效率。

ensureIndex()

功能：创建索引

参数：索引域和索引选项

e.g. 根据name域创建索引

db.class0.ensureIndex({name:1})

\*1表示正序索引，-1表示逆序索引

查看集合中的 索引

> db.class0.getIndexes()

索引选项：

自定义索引名称：

db.collection.ensureIndex({}，{name：名字})

e.g.对age域创建索引命名为ageIndex

> db.class0.ensureIndex({age:1},{name:'ageIndex'})

删除索引：

db.collection.dropIndex(‘index’)

功能：删除索引

参数：要删除的索引名称或者键值对

e.g.

> db.class0.dropIndex('ageIndex')

> db.class0.dropIndex({name:1})

db.collection.dropIndexes()

功能：删除所有索引

\*\_id是系统自动创建的主键索引，不能删除

索引类型：

复合索引：

根据多个域创建一个索引

e.g.

db.class0.ensureIndex({name:1，age:-1},{name:’name\_age’})

数组索引：子文档索引

如果对某个域的值为数组或者子文档的域创建索引，那么通过数组或者子文档中某一项进行查找也是索引查找

e.g.如果对score创建了索引那么该查找就是索引查找

db.class1.find({‘score.1’:88})

唯一索引

创建索引的域要求值不能够重复

e.g.对name创建唯一索引

db.class0.ensureIndex({name:1},{unique:true})

\*当对某个域创建了唯一索引就不能插入重复的值

稀疏索引（间隙索引）

只针对有指定索引域的文档创建索引，没有该域的文档不会插入到索引表

e.g.只对有age域的文档创建索引

> db.class0.ensureIndex({age:1},{sparse:true})

索引约束

\*索引表需要占用一定的数据库磁盘空间

\*当对数据进行增删改等写入操作时索引也需要更新，降低了数据修改的效率

综上：数据量较小时不适合创建索引，当数据库进行频繁的修改操作而不是查找操作时也不适合创建索引。针对一个集合并不是创建索引越多越好。

聚合操作：

对文档的筛选结果进行整理统计

db.collection.aggregate()

功能：完成聚合操作

参数：聚合条件---》聚合操作符

$group分组聚合 需要配合具体的分组统计选项

$sum：求和

e.g.

> db.class0.aggregate({$group:{\_id:'$gender',num:{$sum:1}}})

> db.class0.aggregate({$group:{\_id:'$gender',num:{$sum:'$age'}}})

$avg；求平均数

e.g.

> db.class0.aggregate({$group:{\_id:'$gender',avgage:{$avg:'$age'}}})

$max 求最大值

e.g.

> db.class0.aggregate({$group:{\_id:'$gender',maxage:{$max:'$age'}}})

$max 求最小值

e.g.

> db.class0.aggregate({$group:{\_id:'$gender',maxage:{$min:'$age'}}})

$project

修改文档的显示效果

e.g.

> db.class0.aggregate({$project:{\_id:0,name:1,age:1}})

> db.class0.aggregate({$project:{\_id:0,Name:'$name',Age:'$age'}})

$match

数据筛选

$match值的用法同query一致

e.g.过滤年龄大于18的数据文档

> db.class0.aggregate({$match:{age:{$gt:18}}})

$limit

筛选前几条文档：

e.g.筛选前三条数据文档

> db.class0.aggregate({$limit:3})

$skip

跳过几条文档显示

e.g. 跳过前三项

> db.class0.aggregate({$skip:3})

$sort

将数据排序

e.g.按照正序来排序

> db.class0.aggregate({$sort:{age:1}})

聚合管道

聚合管道指的是将上一个聚合的操作结果给下一个聚合继续操作

db.collection.aggregate({聚合},{},{}…)

e.g. match --->project---> sort

> db.class0.aggregate([{$match:{gender:'m'}},{$project:{\_id:0}},{$sort:{age:1}}])

e.g. group----->match

> db.class0.aggregate([{$group:{\_id:'$name',num:{$sum:1}}},{$match:{num:{$gt:1}}}])