## 看看mongodb的基本操作

### mongodb的基本使用包括几个方面

* 增、删、查、改是重点

CRUD

C:insert()

R:find()

U:update()

D:remove()

### 增加数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | [root@blog ~]# hostname  blog.unix178.com  [root@blog ~]# mongo  MongoDB shell version: 2.4.6  connecting to: test  > show dbs  admin0.203125GB  local0.078125GB  test0.203125GB  > use zhuima  switched to db zhuima  > info = {Name:"zhuima",Age:26,Gender:"F",Address:"Beijing China",Work:"Engineer",Other:"DevOps"}  {  "Name" : "zhuima",  "Age" : 26,  "Gender" : "F",  "Address" : "Beijing China",  "Work" : "Engineer",  "Other" : "DevOps"  }  > db.peroson.insert(info)  > db.person.find()  > db.peroson.find()  { "\_id" : ObjectId("53cbc52255ea1fd13e47ad9d"), "Name" : "zhuima", "Age" : 26, "Gender" : "F", "Address" : "Beijing China", "Work" : "Engineer", "Other" : "DevOps" }  > |

### 同样我们可以批量插入数据

### 删除数据

### 批量删除数据

### 指定某个条件查找数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | > db.peroson.findOne()  {  "\_id" : ObjectId("53cbc67855ea1fd13e47ad9e"),  "Name" : "zhuima1",  "Age" : 1,  "Address" : "1.Number Beijing China"  }  > db.peroson.find({Name:"zhuima888"})  { "\_id" : ObjectId("53cbc67855ea1fd13e47b115"), "Name" : "zhuima888", "Age" : 88, "Address" : "888.Number Beijing China" }  > |

### 指定多重条件查找

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | > db.peroson.find({Age:{$gt:33,$lt:44}})  { "\_id" : ObjectId("53cbc67855ea1fd13e47adbf"), "Name" : "zhuima34", "Age" : 34, "Address" : "34.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc0"), "Name" : "zhuima35", "Age" : 35, "Address" : "35.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc1"), "Name" : "zhuima36", "Age" : 36, "Address" : "36.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc2"), "Name" : "zhuima37", "Age" : 37, "Address" : "37.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc3"), "Name" : "zhuima38", "Age" : 38, "Address" : "38.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc4"), "Name" : "zhuima39", "Age" : 39, "Address" : "39.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc5"), "Name" : "zhuima40", "Age" : 40, "Address" : "40.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc6"), "Name" : "zhuima41", "Age" : 41, "Address" : "41.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc7"), "Name" : "zhuima42", "Age" : 42, "Address" : "42.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47adc8"), "Name" : "zhuima43", "Age" : 43, "Address" : "43.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae23"), "Name" : "zhuima134", "Age" : 34, "Address" : "134.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae24"), "Name" : "zhuima135", "Age" : 35, "Address" : "135.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae25"), "Name" : "zhuima136", "Age" : 36, "Address" : "136.Number Beijing China" }  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae26"), "Name" : "zhuima137", "Age" :  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae2c"), "Name" : "zhuima143", "Age" : 43, "Address" : "143.Number Beijing China" }  Type "it" for more  > |

### 更新数据

|  |  |
| --- | --- |
| 1  2  3  4  5 | > db.peroson.find({Name:"zhuima100"})  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae01"), "Name" : "zhuima100", "Age" : 0, "Address" : "100.Number Beijing China" }  > db.peroson.find({Name:"zhuima100"})  { "\_id" : ObjectId("53cbc67855ea1fd13e47ae01"), "Name" : "zhuima100", "Age" : 100, "Address" : "100.Number Beijing China" }  > |

### 批量更新数据

* 数组修改器
* $push  会向已有的数组末尾加入一个元素，要是没有就会创建一个新的数组

## 看了上面的一些例子，我们来点短小的命令

### 命令行接口

|  |  |
| --- | --- |
| 1  2 | mongo  > |

### 查看所有数据库

|  |  |
| --- | --- |
| 1 | > show dbs |

### 查看数据库中所有的表

* 后面的；可有可无，看个人喜好

|  |  |
| --- | --- |
| 1  2  3 | > show dbs;  > use zhuima;  > show collections; |

### 为数据库下某个表插入数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | > j = { name : "zhuima"}  > k = { age : 23}  > m = {sex : "fmale"}  > db.test.insert(j)  > db.test.insert(k)  > db.test.insert(m)  > |

### 查看某个数据库中的数据

|  |  |
| --- | --- |
| 1  2 | > db.test.find() # 默认显示前20行 可以再>后接着输入it显示下一页的20行数据~  > |

### 查看数据库中的某个表的数据第一行

|  |  |
| --- | --- |
| 1  2 | > db.test.findOne()  > |

### 查看数据库中的某个表的前几个

* limit限制输出信息量

|  |  |
| --- | --- |
| 1  2 | > db.test.find().limit(3) 只列出前几行  > |

### 如何显示数据库中所有信息(突破只能显示20行的限制)

|  |  |
| --- | --- |
| 1  2  3 | > var c = db.test.find()  > while (c.hasNext()) printjson(c.next())  > |

### 对数据进行筛选

|  |  |
| --- | --- |
| 1  2 | > db.text.find({x:3}) #筛选出x对应值为3的匹配的行  > |

### 批量插入数据

|  |  |
| --- | --- |
| 1  2 | > for (i =1; i<=25;i++) db.test.insert({x:i})  > |

### 只列出从第几行开始的行（有规律的）

* sort进行排序操作
* $gt是mongodb内置的比较运算的内置变量

|  |  |
| --- | --- |
| 1  2 | > db.test.find({x:{$gt:5}}).sort({x:3})  #列出从第6行开始的行并进行排序  > |

### 只列出某个范围内的行

|  |  |
| --- | --- |
| 1  2 | > db.test.find({x:{$gt:5}}).limit(5) # 只列出第6行到第10行  > |

### 过滤包含某些关键字的筛选

* $in也是mongodb内置的比较运算的内置变量

|  |  |
| --- | --- |
| 1  2  3  4 | > db.inventory.find( { type: { $in: [ 'food', 'snacks' ] } } )  > db.zhuima.find({name:{$in:["zhuima"]}})  { "\_id" : ObjectId("53b272e9c74fabe96af3a054"), "name" : "zhuima" }  > |

### 数组定义

|  |  |
| --- | --- |
| 1 | > i={"date":new Date()} |

### 可以嵌套数组的

|  |  |
| --- | --- |
| 1  2  3 | > name={"name":{"gender":"male"}}  { "name" : { "gender" : "male" } }  > |

### 定义一个名为追马的人的相关信息

|  |  |
| --- | --- |
| 1  2  3 | > info={"name":"zhuima","other":["male","single"],"age":22}  { "name" : "zhuima", "other" : [ "male", "single" ], "age" : 22 }  > |

# 三.创建、更新及删除文档

1创建

>var joe = {"name":"joe","friends":32,"enemies":2}

>db.user.insert(joe)

WriteResult({ "nInserted" : 1 })

2更新

> var joe\_bak = db.user.findOne({"name":"joe"})

> joe\_bak.relationships = {"friends":joe.friends,"enemies":joe.enemies}

{ "friends" : 32, "enemies" : 2 }

> joe.username = joe.name

Joe

> delete joe.friends

true

> delete joe.enemies

true

> delete joe.name

True

> db.user.update({"username":"joe"},joe\_bak)

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.user.findOne()

{

"\_id" : ObjectId("56d2a103d66803eeed7a475b"),

"name" : "joe",

"friends" : 32,

"enemies" : 2,

"relationships" : {

"friends" : 32,

"enemies" : 2

}

}

## 3.1使用更新修改器

### set/unset

> db.page.update({"\_id" : ObjectId("56d2a7b0d66803eeed7a475c")},

... {"$set":{"favorite book":"war and peace"}})

2.1.1用$set修改键的数据类型

> db.page.update({"\_id" : ObjectId("56d2a7b0d66803eeed7a475c")},

... {"$set":{"favorite book":["cat`s cradle","foundation trilogy","ender`s game"

]}})

> db.page.findOne()

{

"\_id" : ObjectId("56d2a7b0d66803eeed7a475c"),

"url" : "www.example.com",

"pageview" : 54,

"favorite book" : [

"cat`s cradle",

"foundation trilogy",

"ender`s game"

]

$unset可以将键完全删除

> db.page.update({"\_id" : ObjectId("56d2a7b0d66803eeed7a475c")},

... {"$unset":{"favorite book":1}})

结果

> db.page.findOne()

{

"\_id" : ObjectId("56d2a7b0d66803eeed7a475c"),

"url" : "www.example.com",

"pageview" : 54

}

使用$set修改内嵌文档，当前，文档user中relationships" : { "friends" : 32, "enemies" : 2}

> db.user.findOne()

{

"\_id" : ObjectId("56d2a103d66803eeed7a475b"),

"name" : "joe",

"friends" : 32,

"enemies" : 2,

"relationships" : {

"friends" : 32,

"enemies" : 2

}

}

> db.user.update({ "\_id" : ObjectId("56d2a103d66803eeed7a475b")},

... {"$set":{"relationships.friends":33}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.user.findOne()

{

"\_id" : ObjectId("56d2a103d66803eeed7a475b"),

"name" : "joe",

"friends" : 32,

"enemies" : 2,

"relationships" : {

"friends" : 33,

"enemies" : 2

}

}

小结：$inc与$set功能相似，但$inc只能用来增加数字:整数、长整形、双精度浮点。用在其他类型会失败。

另外，$inc键的类型必须是数字，不能使其他的字符串、数组或其他类型。

## 3.2使用数组修改器

### push/pops

数组操作，顾名思义是用来应用在值为数组的键上，例如不能对整数进行push和pop。如果指定的键已经存在，则”$push”会在数组末尾加一个元素，要是没有则创建新的数组。例：

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z")

}

>

>

> db.blog.update({"title" : "My Blog Post"},{"$push":{"conmments"

... :

... {"name":"joe","email":"joe@example.com","content":"nice post." }

... }})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

插入后，查询得到：

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z"),

"conmments" : [

{

"name" : "joe",

"email" : "joe@example.com",

"content" : "nice post."

}

]

}

> db.blog.update({ "title" : "My Blog Post"},{"$push":{"name":"bob","email":"bob

@example.com","content":"good post."}})

### addToSet/$each

在键值为数组的情况下，想要在数组中添加新的值，则可以用addToSet，如果某元素不再列表中，则可以用其添加进去，如果已经在列表中，则添加不进去。如**blog**:

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z"),

"conmments" : [

{

"name" : "joe",

"email" : "joe@example.com",

"content" : "nice post."

},

{

"name" : "bob",

"email" : "bob@example.com",

"content" : "good post."

}

],

"visitors" : [

"joe",

"john"

]

}

用addToSet添加

db.blog.update({"title" : "My Blog Post"}, {"$addToSet":{"visitors":"johndan"}

结果：

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z"),

"conmments" : [

{

"name" : "joe",

"email" : "joe@example.com",

"content" : "nice post."

},

{

"name" : "bob",

"email" : "bob@example.com",

"content" : "good post."

}

],

"visitors" : [

"joe",

"john",

"johndan"

]

}

若再次进行添加 "johndan"，则结果不变,**避免了重复数据**。

也可以和“$each”结合起来使用，可一次添加某个键的多个值。

例：> db.blog.update({"title" : "My Blog Post"},{"$addToSet":{"visitors":{"$each":["joxi","joyi","john"]}}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z"),

"conmments" : [

{

"name" : "joe",

"email" : "joe@example.com",

"content" : "nice post."

},

{

"name" : "bob",

"email" : "bob@example.com",

"content" : "good post."

}

],

"visitors" : [

"joe",

"john",

"johndan",

"joxi",

"joyi"

]

}

## 3.3数组定位修改器

若是数组有多个值，而我们只想对其中的一部分进行操作，这里就需要用一部分技巧。有两种方法操作数组中的值：通过位置和定位操作符$。

* 通过位置：

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z"),

"conmments" : [

{

"name" : "joe",

"email" : "joe@example.com",

"content" : "nice post."

},

{

"name" : "bob",

"email" : "bob@example.com",

"content" : "good post."

}

],

"visitors" : [

"joe",

"john",

"johndan",

"joxi"

],

> **db.blog.update({"title" : "My Blog Post"}, {"$set":{"conmments.0.name":"joj"}}**

但是在很多情况下，不通过事先查询一边就不知道要修改的数组的下标。为了克服这个困难，M哦你goDB提供了定位操作符“$”用来定位查询出来已经匹配的文档。

* 通过定位符

**> db.blog.update({"conmments.name":"joj"}, {"$set":{"conmments.$.name":"jjj"}})**

注意，定位符只能更新第一个匹配的元素，所以对于符合{"conmments.name":"joj"}条件的所有文档来说，只有第一个被更新。

结果都可以实现：

> db.blog.findOne()

{

"\_id" : ObjectId("56d1518ea487e246283c7a89"),

"title" : "My Blog Post",

"content" : "Here is my blog post",

"date" : ISODate("2016-02-27T07:34:20.950Z"),

"conmments" : [

{

**"name" : "jjj",**

"email" : "joe@example.com",

"content" : "nice post."

},

{

"name" : "bob",

"email" : "bob@example.com",

"content" : "good post."

}

],

"visitors" : [

"joe",

"john",

"johndan",

"joxi"

]

}

## 3.4修改器速度

## 3.5 upsert/更新多个文档

* upsert也是起到更新的作用，但比较特殊。如果没有找到符合条件的文档，则以这个条件为基础 创建一个新的文档；如果找到了一个文档则类似update进行正常更新。它非常方便，不用预置集合，同一套代码即可实现创建又可以实现更新。

> db.math.update({"count":28},{"$inc":{"count":3}},true)

这里主要是通过修改update的第三个参数来实现的upsert的功能，把这个参数设为true则可以实现。

* 当要更新多个文档时，就要用到update的第四个参数，这个参数默认为false，是针对的更新一个符合条件的文档，现在把它设置为true则可以实现更新多个。

> db.math.update({ "count" : 31},{"$inc":{"count":3}},false, true)

## 3.6返回已经更新的文档

有时候，对修改后的文档还要进行再客户端进行下一步的操作，这就需要，我们在服务端处理完相应文档时返回已经更新的文档，这个时候就要用到findAndModify,它有三部分组成：query，sort、update(或remove)

> db.runCommand({"findAndModify":"math", "query":{"count" : 34}, "sort":{"count"

: -1}, "update":{"$set":{"m":56}}})

通过”.value”的方式可以查看到已修改的文档本身。

|  |  |  |
| --- | --- | --- |
| **参数** | **详解** | **默认值** |
| *query* | 查询过滤条件 | {} |
| *sort* | 如果多个文档符合查询过滤条件，将以该参数指定的排列方式选择出排在首位的对象，该对象将被操作 | {} |
| *remove* | 若为true，被选中对象将在返回前被删除 | N/A |
| *update* | 一个 [修改器对象](http://www.mongodb.org/display/DOCS/Updating) | N/A |
| *new* | 若为true，将返回修改后的对象而不是原始对象。在删除操作中，该参数被忽略。 | false |
| *fields* | 参见[Retrieving a Subset of Fields](http://www.mongodb.org/display/DOCS/Retrieving+a+Subset+of+Fields) (1.5.0+) | All fields |
| *upsert* | 创建新对象若查询结果为空。 [示例](http://github.com/mongodb/mongo/blob/master/jstests/find_and_modify4.js) (1.5.4+) | false |

# 四、查询

* 有时候通过find/findOne查询出来的结果键太多，此时就可以通过修改其第二个参数来使结果只显示/不显示某个或某些字段。

> db.blog.update({},{"$unset":{ "comments":1,"commments":0}})

查询条件：

* $lt, $lte, $gt, $gte分别对应<,<=,>,>=，用法和上述一样。
* $in, $nin, $or
* $not , $mod(相除取余)
* null可以匹配null和“不存在的“
* 对于正则表达式，则需要如：> db.blog.find({"title1": /Msy.\*/})
* 查询数组，主要是匹配元素个数的问题

> db.blog.findOne()

{ "\_id" : ObjectId("56d155a1d66803eeed7a4759"), "num" : [ 1, 2, 3 ] }

* 几个条件同时限制

> db.blog.update({"\_id" : ObjectId("56d155a1d66803eeed7a4759")},

{**"$push":{"num":4}**,  **"$inc":{"n":1}**  } )

即：在为“num”添加元素的时候，同时键n的值自动加1，添加“num”的次数就是n的值。

* 排序limit/sort/skip

# 五、索引

**要考虑：**

1. **有可能做到哪些查询？其中哪些键需要索引**
2. **每个键的索引方向是什么样的？**
3. **如何应对扩展？以使不同键的排列可以使常用数据更多的保存在内存中。**

## 5.1ensureIndex/dropIndex

同一个集合，只需创建一个索引：普通文档db.blog.ensureIndex({"title":1},{“background”:”true”})或内嵌文档db.blog.ensureIndex({"conmments.name":1})，并非索引越多越好，因为在执行插入、更新、删除的时候，**数据库不仅要执行这些操作，还要将这些操作在索引中标记**，这会增加开销。

> db.system.indexes.find()

{ "v" : 1, "key" : { "\_id" : 1 }, "name" : "\_id\_", "ns" : "test.blog" }

{ "v" : 1, "key" : { "\_id" : 1 }, "name" : "\_id\_", "ns" : "test.book" }

{ "v" : 1, "key" : { "\_id" : 1 }, "name" : "\_id\_", "ns" : "test.foo" }

{ "v" : 1, "key" : { "title" : 1 }, "name" : "title\_1", "ns" :

> db.runCommand({"dropIndexes":"blog","index":"title\_1"})

{ "nIndexesWas" : 2, "ok" : 1 }

**5.2使用explain和hint**

## 5.3地理空间索引

* $near/geoNear

db.map.ensureIndex({“gps”:**”2d”**},gps的值必须是某种形式的键-值对，如{“gps”:[1,2]},{“gps”:{“x”:30,”y”:-30}}

* 查询距离

db.map.find({“$near”:[1,2]}) .limit(10) 或者db.runCommand({“geoNear”:”map”,near:[3,4],num:10})

* 形状距离查询

# 6.聚合

* count/distinct

db.blog.find().count 表示查询结果的数量

db.runCommand({“distinct”:”blog”,”key”:”title”})

* group/finalize/MapReduce

group作为分组的命令与关系型数据库中的“group by”功能类似，但用法不同。常用在”将集合中的某个或某几个键提取出来，并对其对应的值做一定的修改”的场景下。如：

{"title" : "java sun", "author" : "jk", "day" : "2012-12-14", "tags" : ["java", "nosql", "spring"]},

{"title" : "ssh2的整合", "author" : "cj", "day" : "2012-5-10", "tags" : ["struts 2", "hibernate ", "spring"]},

{"title" : "c#的高级用法", "author" : "zt", "day" : "2012-4-3", "tags" : ["c#", "sql"]},

{"title" : "php mongo", "author" : "lx", "day" : "2012-12-14", "tags" : ["php", "nosql", "mongo"]}

|  |  |
| --- | --- |
| 处理 | 解释 |
| db.blog.group({  ... "key":{"day":true},  ... "initial":{"tags":{}},  ... "reduce":function(doc,prev){  ... for(i in doc.tags){  ... if(doc.tags[i] in prev.tags)  ... {prev.tags[doc.tags[i]]++;}  ... else  ... {prev.tags[doc.tags[i]] = 1 ;}  ... }  }}) | “key”代表要按照分组的键，这个例子是按照“day”这个键进行分组，ture和false没有发现区别  “initial”是要将累加器prev的值进行初始化为”{}”,这个累加器是在每一步的过程中都会有变化的。  对于“"reduce":function(doc,prev)”，每次都会讲doc也就是每一个集合的文档进行拿出来处理，而prev累加器就像一个容器一样积累每一次处理doc的结果。 |

经常，还要和finalize和用。

db.blog.group({

... "key":{"day":true},

... "initial":{"tags":{}},

... "reduce":function(doc,prev){

... for(i in doc.tags){

... if(doc.tags[i] in prev.tags)

... {prev.tags[doc.tags[i]]++;}

... else

... {prev.tags[doc.tags[i]] = 1 ;}

... }

},

"finalize": function (prev) {

var mostpopular = 0;

for (i in prev.tags) {

if (prev.tags[i] > mostpopular) {

prev.tag = i;

mostpopular = prev.tags[i];}

}

delete prev.tags;

}

})