一、非关系型数据库

一、*Database简单操作*

1. 显示数据库：show databases / show dbs

   显示当前数据库：db

2. 创建/切换 数据库：use mydb(要存放数据才能show dbs查看)

3. 删除数据库：db.dropDatabase(),

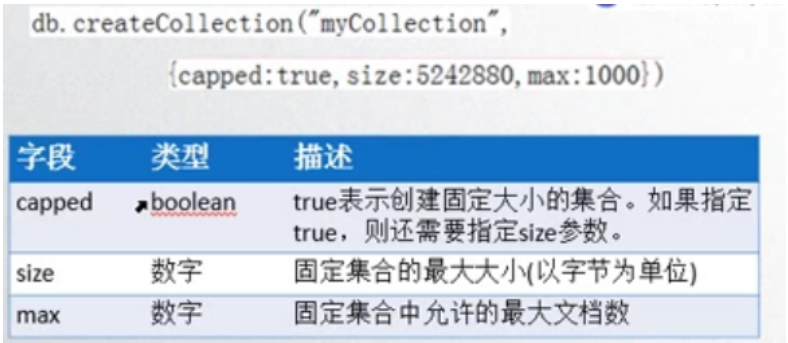
   - 如果没有选择任何数据库，删除系统默认的test数据库
   - 先进入要删除的数据库，再删除

     use mydb ---> db.dropDatabase()

4. 显示集合：show collections / show tables

5. 创建集合：

```
显式创建：
- - 无限大：db.createCollection(集合名称)，eg:db.createCollection("bbs")--->{ "ok" : 1}--->成功
- - 规定大小：db.createCollection("users",{capped:true,size:1048576,max:10})
```



字段	类型	描述
capped	boolean	true表示创建固定大小的集合。如果指定true，则还需要指定size参数。
size	数字	固定集合的最大大小(以字节为单位)
max	数字	固定集合中允许的最大文档数

```
隐式创建：向不存在的集合中插入数据，insert会自动创建  该集合
```

6. 删除集合：db.collection_name.drop()

   删除集合中所有记录：db.collection_name.remove({})

二、数据格式

1. JSON: 布尔、数字、字符串、数组、对象、null（6种数据类型）
2. BSON： 6种数据类型+Data类型+BinData类型

三、文档

1、添加文档

（1）insert : db.collection_name.insert({文档}),

eg: db.student.insert({"name":"张三"})

（2）save：db.collection_name.save({文档}),

eg：旧的：db.student.save({"_id":"01":"name":"李四"})

save后：db.student.save({"_id":"01":"name":"王五"}) ---> 直接更新李四为王五

【注：能存的记录达到上限，最开始插入的数据会被顶掉】

2、删除文档

1、 注意事项

- mongoDB可以通过remove() 方法删除集合中的文档
- 语法： remove(criteria,justOne)
- ---criteria - (可选)删除条件，符合删除条件的集合将被删除
- ---justOne - (可选)如果设置为true或1，则只删除一个文档。
- 删除所有文档remove()的参数要设为{}

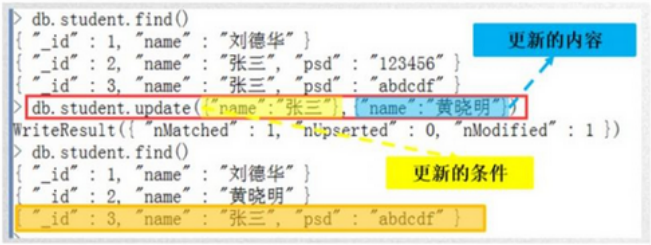
2、 实例操作

```
> db.student.count()
2
> db.student.remove({})
WriteResult({ "nRemoved" : 2 })
> db.student.count()
0
```

3、更新文档

- mongoDB可以通过update() 方法更新集合中现有文档的值
- 语法： db.collection_name.update(criteria, objNew, upsert, multi)
  - criteria。update的查询条件，类似sql update查询内where后面的条件。
  - objNew。update的对象和一些更新的操作符（如$,$inc...）等，类似于sql update查询内set后面的内容
  - upsert。boolean值，true表示如果不存在update的记录，作为新文档插入，false则不插入。默认是false。
  - multi。boolean值，true表示将按条件查出来多条记录全部更新； false表示只更新找到的第一条记录，如果这个参数为。默认是 false。

2、 实例操作



db.student.update({" 更新定位"},{" 新写入的值"})--->update是整个文档的替换，只修改部分内容需要用修改器

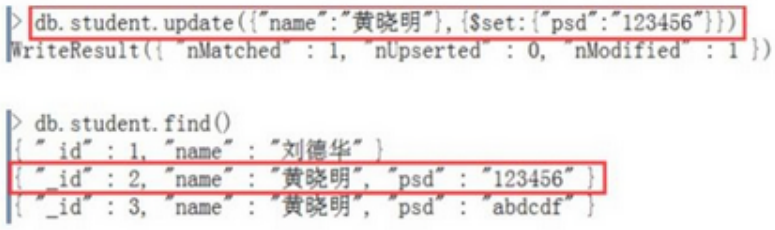
3-1.key修改

$set

1、注意事项

- 用来修改一个指定键的值，如果不存在则创建它
- 语法：db.collection_name.update({ 条件},{$set : { field : value } })

2、实例操作



$unset

1、注意事项

- 从文档中移除指定的键
- 语法：db.collection_name.update({ 条件},{$set : { field : value } })
- $unset只关注key的值，不关注value的值

2、实例操作

```
> db.student.update({"name":"黄晓明"},{$unset:{"psd":"16"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find()                                          }})
{ "_id" : 1, "name" :
{ "_id" : 2, "name" : "黄晓明" }
{ "_id" : 3, "name" : "黄晓明", "psd" : "abdcdf" }
>
```

03.jpg

$inc

04.jpg 意事项

- $inc用来增加已有键的值，或者在键不存在时创建一个键
- $inc就是专门来增加（和减少）数字的
- $inc只能用于整数、长整数或双精度浮点数
- $inc的键值必须为数值
- 语法：db.collection_name.update({ 查找条件文档},{$inc : { field : value } })

```
> db.student.update({"_id":2},{$inc:{"weight":71.3}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find({"_id":2})
{ "_id" : 2, "name" : "黄晓明", "age" : 17, "weight" : 71.3 }
```

04.jpg

```
> db.students.update({"name":"赵六"},{"name":"刘亦菲","age":20})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.find({"name":"刘亦菲"})
{ "_id" : ObjectId("5aa53d8b18696122dce100aa"), "name" : "刘亦菲", "age" : 20 }
```

$rename

1、注意事项

- $rename操作符可以重命名字段名称，新的字段名称不能和文档中现有的字段名相同
- 语法：

db.collection_name.update({ 查找条件文档},{$rename:{old:new,old:new} })

2、实例操作

```
> db.student.find({"_id":4})
{ "_id" : 4, "name" : "聂小倩", "cell" : "13545871234" }
> db.student.update({"_id":4},{$rename:{"cell":"number"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find({"_id":4})
{ "_id" : 4, "name" : "聂小倩", "number" : "13545871234" }
```

3-2.数组

三、更新数组---添加元素

1、 $push

- $push操作符可以在指定字段中添加某个值
- 语法： db.collection_name.update({ 查找条件文档},{$push:{field:value} })
  - 如果该字段是个数组，那么该值将被添加到数组中
  - 如果该字段尚不存在，那么该字段的值将被设置为数组
  - 如果该字段存在，但不是数组，那么将会抛出异常
  - 如果给定的值是个数组，那么该数组被看做是一个元素，添加给定字段中
- 实例操作

```
> db.students.update({"name":"银十"},{$push:{"score":95}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

2、 $pushAll

- 可以一次追加多个内容到数组里面

3、 $each

- 使用$push时如果希望在给定的数组中添加多个值，可以使用可选的 $each

修改操作符

- 实例操作

```
> db.students.find({"name":"银十"})                                    })})
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address
 : "开福区", "score" : [ 87, 79 ] }
> db.students.update({"name":"银十"},{$push:{"score":{$each:[38,59]}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.find({"name":"银十"})                                    })
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address
 : "开福区", "score" : [ 87, 79, 38, 59 ] }
```

4、 $slice

- 在使用$each是还可以使用$slice修改操作符，通过这种方式可以限制 $push操作符中数组内元素的数量
- $slice是正数将保证数组中的前n个元素会被保留
- $slice是负数将保证数组中的最后n个元素会被保留
- $slice是0则表示清空数组
- 实例操作

```
> db.students.update({"name":"银十"},{$push:{"score":{$each:[59,90,87],$slice:2}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.find({"name":"银十"})                                    })
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address
 : "开福区", "score" : [ 59, 90 ] }
```

5、 $addToSet

- 向数组里面增加一个新的内容，只有这个内容不存在的时候才会增加
- 使用$addToSet时，可以使用$each操作符指定的额外的参数
- 实例操作

```
> db.students.update({"name":"银十"},{$addToSet:{"score":{$each:[87,66]}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.find({"name":"银十"})                                    })})
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address
 : "开福区", "score" : [ 95, [ 87, 54 ], 87, 54, 66 ] }
```

四、更新数组---删除元素

1、 $pop

- 删除数组内的数据
- -1表示删除第一个
- 1表示删除最后一个
- 实例操作

```
> db.students.update({"name":"银十"},{$pop:{"score":1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.find({"name":"银十"})
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address"
 : "开福区", "score" : [ 95, [ 87, 54 ], 87, 54 ] }
```

2、 $pull

- 从数组中删除所有指定的值或者符合给定条件的值
- 适用于删除多个相同值的元素
- 实例操作

```
> db.students.find({"name":"银十"})
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address"
 : "开福区", "score" : [ 95, [ 87, 54 ], 87, 54 ] }
> db.students.update({"name":"银十"},{$pull:{"score":[87,54]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.find({"name":"银十"})
{ "_id" : ObjectId("5aa4a7cd8c6c56b5cb6c71df"), "name" : "银十", "sex" : "女", "age" : 21, "address"
 : "开福区", "score" : [ 95, 87, 54 ] }
```

3、 $pullAll

- 一次性删除多个内容

```
> db.students.update({"name":"银十"},{$pushAll:{"score":[87,54]}})
```

push只能对数组进行操作，使用push表明是向数组中添加字段。

pushAll追加多个内容到数组

4、查找文档

一、文档的查询find

1、注意事项

- MongoDB可以通过 find()方法查找指定的文档内容
- 语法：db.collection_name.find(criteria,projection)
  - - criteria ：可选，表示查询条件
  - - projection ：可选，使用投影操作符指定返回的键。
- 参数1为{}表示没有查询条件
- 参数2表示指定的键是否要返回。
  - 1表示返回该键
  - 0表示不返回该键
- _id 键默认返回，需要主动指定 _id:0 才会隐藏

查询性别为“女”，所有人的name,age

```
> db.students.find({"sex":"女"},{"name":1,"age":1,"_id":0})
{ "name" : "李四", "age" : 20 }
{ "name" : "王五", "age" : 19 }
{ "name" : "金九", "age" : 20 }
{ "name" : "牛二", "age" : 20 }
```

查找第一条数据：db.student.findOne()

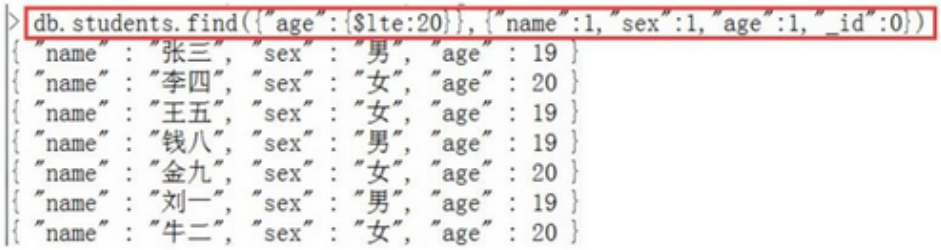
4-1.关系运算符

关系运算符

1、注意事项

序号	运算符	含义	示例
1	$gt	>	年龄大于1：{"age":{$gt:1}}
2	$gte	>=	成绩及格：{“score":{$gte:60}}
3	$lt	<	未成年：{"age":{$lt:16}}
4	$lte	<=	学分小于等于2：{"credit":{$lte:1}}
5	$ne	!=	姓名不是文松：{“name”:{$ne:“张三"}}

2、实例操作




```
$gt -------- greater than  >

$gte --------- gt equal  >=

$lt -------- less than  <

$lte --------- lt equal  <=

$ne ----------- not equal  !=

$eq  --------  equal  =
```

统计数目：.count()--->dbstudent.find("address":"天心区"}).count()

4-2.逻辑运算符

逻辑运算符

1、 注意事项

序号	运算符	含义
1	$and	逻辑与
2	$or	逻辑或
3	$not	逻辑非

2、 $and

- $and等同于SQL中的and，$and所针对的条件被放到一个数组中，每个数组元素表示and的一个条件
- 语法：{ $and: [{ K:V }, {K:V }, ...] }
- 实例操作

```
> db.students.find({$and:[{"age":{$gte:20}},{"age":{$lte:21}}]})
{ "_id" : "1002", "name" : "李四", "sex" : "女", "age" : 20, "addr
{ "_id" : "1004", "name" : "赵六", "sex" : "男", "age" : 21, "addr
{ "_id" : "1006", "name" : "金九", "sex" : "女", "age" : 20, "addr
{ "_id" : "1008", "name" : "牛二", "sex" : "女", "age" : 20, "addr
```

3、 $or

- $or等同于SQL中的or，$or所针对的条件被放到一个数组中，每个数组元素表示or的一个条件
- 语法：{ $or: [{ K:V }, {K:V }, ...] }
- 实例操作

```
> db.students.find({"score":{$gt:80}},{"name":1,"age":1,"score":1})
{ "_id" : "1003", "name" : "王五", "age" : 19, "score" : 88 }
{ "_id" : "1004", "name" : "赵六", "age" : 21, "score" : 82 }
{ "_id" : "1006", "name" : "金九", "age" : 20, "score" : 96 }
```

4、 $not

- $not表示取反，等同于SQL中的not。
- 语法：{ field: { $not: { <expression1> } } }
- $not操作符不能独立使用，必须跟其他操作条件一起使用
- 实例操作

```
> db.students.find({"age":{$not:{$eq:19}}})
{ "_id" : "1002", "name" : "李四", "sex" : "女", "age" : 20, "address" : "岳麓区", "score" : 75 }
{ "_id" : "1004", "name" : "赵六", "sex" : "男", "age" : 21, "address" : "天心区", "score" : 82 }
{ "_id" : "1006", "name" : "金九", "sex" : "女", "age" : 20, "address" : "天心区", "score" : 96 }
{ "_id" : "1008", "name" : "牛二", "sex" : "女", "age" : 20, "address" : "天心区", "score" : 47 }
```

模糊查询

查询 title 包含"教"字的文档:

```
db.col.find({title:/教/})
```

查询 title 字段以"教"字开头的文档:

```
db.col.find({title:/^教/})
```

查询 titl e字段以"教"字结尾的文档:

```
db.col.find({title:/教$/})
```

4-3.排序和分页

排序和分页

1、排序

- 在MongoDB里面数据的排序操作使用“sort()”函数
- 在进行排序的时候可以有两个顺序：升序（1）、降序（-1）
- $natural表示按照数据保存的先后顺序排序
- 排序是对查询的结果进行排序
- 实例操作：

```
> db.students.find({},{"parents":0}).sort({$natural:-1})
{ "_id" : "1017", "age" : 19, "name" : "银十", "address" : "开福区" }
{ "_id" : "1016", "age" : 18, "name" : "金九", "address" : "天心区" }
{ "_id" : "1015", "age" : 21, "name" : "钱八", "address" : "开福区" }
{ "_id" : "1014", "age" : 19, "name" : "赵六", "address" : "天心区" }
{ "_id" : "1013", "age" : 18, "name" : "王五", "address" : "岳麓区" }
{ "_id" : "1012", "age" : 20, "name" : "李四", "address" : "岳麓区" }
{ "_id" : "1011", "age" : 19, "name" : "张三", "address" : "天心区" }
```

2、分页

- 数据分页可以用skip()和limit()函数实现
- skip(n)表示略过前n条数据
- limit(n)表示总共取n条数据
- 分页是在查询结果集上分页
- 如果要返回限制之后的记录数量，要使用 count(true)或者count(非0)
- 实例操作

```
> db.students.find({},{"parents":0}).skip(3).limit(2)
{ "_id" : "1014", "age" : 19, "name" : "赵六", "address" : "天心区" }
{ "_id" : "1015", "age" : 21, "name" : "钱八", "address" : "开福区" }
```

三、$where查询

1、 注意事项

- $where查询是MongoDB的高级查询部分
- 可以接受一个javascript函数或字符串作为查询条件
- 以迭代的方式匹配当前集合里面的所有文档
- 如果满足函数条件，则返回这个文档
- 使用$where效率比较低，因为mongodb要将BOSN数据转成javascript数据，然后一个一个遍历操作

2、 实例操作

```
> db.students.find({"$where":"this.age>20"},{"parents":0})
{ "_id" : "1015", "age" : 21, "name" : "钱八", "address" : "开福区" }
```

```
db.test.insert({
    "class":"1",
    "student":[
        {"name":"mac","age":18,"sex":true},
        {"name":"jack","age":19,"sex":true},
        {"name":"rose","age":20,"sex":false}
    ]
})
```

❤ 查询名字叫mac，年龄20岁的同学的班级信息

```
db.test.find({
"student.name":"mac",
"student.age":20
})
```

结果：班级1被查询出来，结果错误

处理：

❤ 引入elemMatch操作符

```
db.test.find({
"student":{$elemMatch:{"name":"mac","age":20}}
})
```

结果：班级1不会被查询出来

4-4.范围查询

一、范围查询

1、$in

- “$in”用来匹配键值等于指定数组中任意值的文档
- 语法：{field:{$in:[value1, value2, value3,…]}}
- 实例操作：

```
> db.students.find({"age":{$in:[18,19]}},{"_id":0,"major":0})
{ "name" : "程明", "sex" : "男", "age" : 19, "credits" : 52, "score" : 92 }
{ "name" : "刘萍", "sex" : "女", "age" : 18, "credits" : 51, "score" : 88 }
{ "name" : "刘文平", "sex" : "女", "age" : 18, "credits" : 50, "score" : 86 }
{ "name" : "王林", "sex" : "男", "age" : 18, "credits" : 47, "score" : 87 }
```

2、$nin

- “$nin”用来匹配键不存在或者键值不等于指定数组的任意值的文档
- 语法：{field:{$nin:[value1, value2, value3,…]}}

- $nin匹配不存在查询条件的文档
- 实例操作

```
> db.students.find({"address":{$nin:["天心区","岳麓区"]}},{"_id":0,"major":0})
{ "name" : "程明", "sex" : "男", "age" : 19, "credits" : 52, "score" : 92 }
{ "name" : "刘萍", "sex" : "女", "age" : 18, "credits" : 51, "score" : 88 }
{ "name" : "张静", "sex" : "女", "age" : 20, "credits" : 48, "score" : 96 }
{ "name" : "陈好", "sex" : "女", "age" : 21, "credits" : 53, "score" : 97 }
{ "name" : "施行", "sex" : "男", "age" : 20, "credits" : 54, "score" : 60 }
{ "name" : "刘文平", "sex" : "女", "age" : 18, "credits" : 50, "score" : 86 }
{ "name" : "李德胜", "sex" : "男", "age" : 21, "credits" : 55, "score" : 95 }
{ "name" : "苏锦州", "sex" : "男", "age" : 21, "credits" : 50, "score" : 70 }
{ "name" : "张文杰", "sex" : "女", "age" : 20, "credits" : 53, "score" : 40 }
{ "name" : "王林", "sex" : "男", "age" : 18, "credits" : 47, "score" : 87 }
```

4-5.空值查询

1、null

- “null”用来匹配值为null或者键不存在的文档
- 语法： {field:null}
- null还能匹配key不存在的记录
- 实例操作

```
> db.students.find({"major":null})
{ "_id" : "1011", "name" : "施行", "sex" : "男", "age" : 19, "credits" : null, "score" : 70 }
{ "_id" : "1012", "name" : "马萌", "sex" : "女", "age" : 19, "major" : null, "score" : null }
{ "_id" : "1013", "name" : "王林", "sex" : "男", "age" : 18, "major" : null, "credits" : 47 }
```

2、$exists

- $exists用于判断某个字段是否存在
- 语法：{field:{$exists:<Boolean>}}
- 如果要真正的做到查询某个字段为null的记录，需要和$exists连用
- 语法：{field:{$in:[null],$exists:true}}
- 实例操作

```
> db.students.find({"score":{$in:[null],$exists:true}})
{ "_id" : "1012", "name" : "马萌", "sex" : "女", "age" : 19, "major" : null, "score" : null }
```

4-6.正则表达式

1、 注意事项

- $regex为模糊查询的字符串提供正则表达式功能， MongoDB使用Perl兼容正则表达式。
- 语法：

{ <field>: { $regex: /pattern/, $options: '<options>' } }

{ <field>: { $regex: 'pattern', $options: '<options>' } }

{ <field>: { $regex: /pattern/<options> } }

- options:
  - ----i：表示忽略大小写
  - ----m：多行查找；
  - ----x：空白字符串除了被转义的或在字符类中意外的完全被忽略；
  - ----s：匹配所有的字符（圆点、“.”），包括换行内容。---justOne -(可选)如果设置为true或1，则只删除一个文档。
- | 表示或
- ^表示以…开头
- $表示以…结尾实例操作

2、 实例操作

```
db.students.find({"name":/刘|张/},{"_id":0,"major":0})
{ "name" : "刘萍", "sex" : "女", "age" : 18, "credits" : 51, "score" : 88 }
{ "name" : "张静", "sex" : "女", "age" : 20, "credits" : 48, "score" : 96 }
{ "name" : "刘文平", "sex" : "女", "age" : 18, "credits" : 50, "score" : 86 }
{ "name" : "张文杰", "sex" : "女", "age" : 20, "credits" : 53, "score" : 40 }
{ "name" : "杨刘", "sex" : "男", "age" : 19 }
{ "name" : "王张", "sex" : "女", "age" : 18 }
```

4-7.数组查询

1、 $all

06.jpg

- $all用来匹配那些指定键的键值中包含数组的所有元素的文档，数组中元素顺序不影响查询结果
- 语法： {field:{$all:[value1,value2,…]}}
- 实例操作

```
db.students.find({"name":{$all:["李四"]}})
{ "_id" : "1002", "name" : "李四", "address" : "岳麓区", "course" : [ "Java", "mongoDB" ] }
```

2、 查找数组中指定元素

- 查询数组指定位置的元素，则需使用key.index语法指定下标
- 语法：{field:{$all:[value1,value2,…]}}
- 实例操作

```
> db.students.find({"course.1":"MySQL"})
{ "_id" : "1001", "name" : "张三", "address" : "天心区", "course" : [ "Java", "MySQL", "mongoDB" ] }

{ "_id" : "1004", "name" : "赵六", "address" : "天心区", "course" : [ "Linux", "MySQL" ] }
{ "_id" : "1006", "name" : "金九", "address" : "天心区", "course" : [ "Java", "MySQL" ] }
```

3、$size

- $size用于查询指定长度的数组
- size必须指定一个定值，不能接受一个范围值，不能与其他查询子句组合
- 实例操作

```
> db.students.find({"course":{$size:3}})
{ "_id" : "1001", "name" : "张三", "address" : "天心区", "course" : [ "Java", "MySQL", "mongoDB" ] }
```

4、$slice

- $slice可以返回数组中的部分数据
- 语法：$slice:[start,count]
- 返回从start开始（包含），共返回count个数据
- 实例操作

```
> db.students.find({"age":19},{"course":{$slice:[1,2]}})
{ "_id" : "1001", "age" : 19, "name" : "张三", "address" : "天心区", "course" : [ "MySQL", "mongoDB"
 ] }
{ "_id" : "1004", "age" : 19, "name" : "赵六", "address" : "天心区", "course" : [ "MySQL" ] }
{ "_id" : "1007", "age" : 19, "name" : "银十", "address" : "开福区" }
```

1.

【注：此处parents是数组】

2. $elemMatch

```
> db.blogs.find({"comment.author":"zhangsan","comment.score":{"$gte":4
}}).pretty()
```

查询结果：存在两个不同的数组中的结果

3. $where:两种查询方式均可

```
> db.students.find({"$where":"this.age>20"},{"parents":0})
{ "_id" : "1015", "age" : 21, "name" : "钱八", "sex" : "男", "address"
 : "开福区" }
```

查询年龄在19-21的男生信息：

```
> db.students.find({"$where":function(){
...  return (this.age>=19 && this.age<=21 && this.sex=='男')
... }},{"parents":0})
```

查询任意两个值相同的文档：

```
> db.foods.insertMany([
... { "_id" : "1002", "apple" :5, "banana" : 2, "watermelon" : 3 },
... { "_id" : "1003", "apple" : 1, "peach" : 4, "banana" : 4 },
... { "_id" : "1004", "apple" : 7, "banana" : 6, "peach" : 3 },
... { "_id" : "1005", "apple" : 1, "spinach" : 4, "watermelon" : 4 },
... { "_id" : "1006", "bar" : "baz", "banana" : "baz" }])
```

```
> db.foods.find({$where:function(){
...  for(var i in this){
...      for(var j in this){
...                  if(i!=j && this[i]==this[j]) return true;
...      }
...  }
...
... }})
{ "_id" : "1003", "apple" : 1, "peach" : 4, "banana" : 4 }
{ "_id" : "1005", "apple" : 1, "spinach" : 4, "watermelon" : 4 }
{ "_id" : "1006", "bar" : "baz", "banana" : "baz" }
```

数组查询

1. 取余符号：mod

eg: 查询 年龄/20 余数为1 的人

db.student.find({"age":{$mod:[20,1] }})

4-8.游标和索引

一、游标的概念

1、注意事项

- mongoDB中find()方法返回的就是一个游标对象，通过该对象可以逐条处理数据，类似于ResultSet
- 游标有两个操作函数：
  - hasNext()：判断是否有下一行
  - next()：取出下一行数据

二、游标的使用

1、注意事项

- 调用find()方法时，shell并不立即查询数据库，而是等待真正开始要求获得结果时才发送查询
- 如果查询结果没有放在变量中，MongoDB shell会自动迭代，自动显示最开始的若干文档
- shell会迭代100条数据或者前4M数据（两种之间取小者），这样下次调用next或者hasNext时就不必再次连接服务器获取结果
- 游标数据取出来之后，实际上每行数据返回的都是一个Object型的内容
- 如果需要将数据按照json的形式出现，则可以使用printjson()函数实现

2、实例操作

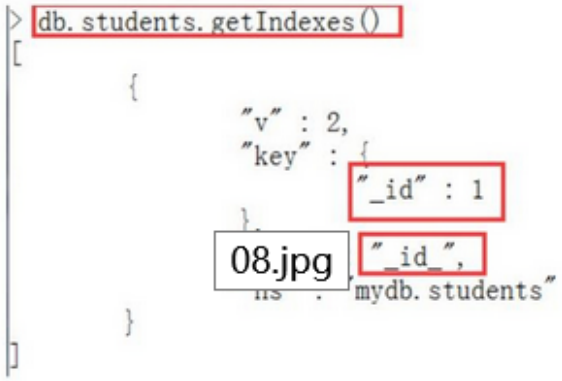
```
> var cursor=db.students.find({"age":20,"sex":"女"})
> while(cursor.hasNext()){
...     printjson(cursor.next().name)
... }
"张静"
"陈好"
"张文杰"
"李四"
```

三、索引的概念

1、 注意事项

- 索引是一种提升数据库检索性能的手段
- MongoDB有两种创建索引的方式：自动创建和手工创建实例操作
- 通过db.name.getIndexes() 命令可以查看集合的索引
- mongoDB默认总是为_id创建索引，并且该索引不能删除。

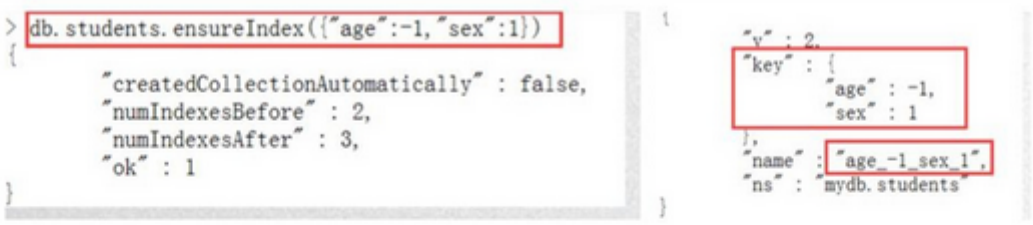
2、 实例操作



四、索引的使用

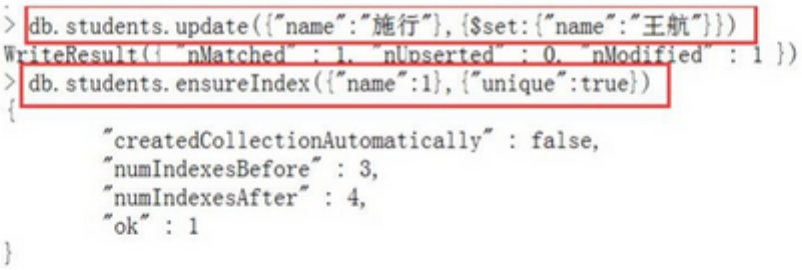
1、 单键索引和复合索引

- 创建索引： db.collection_name.ensureIndex({field:1 或-1})
- 需要用1或-1指明是升序还是降序
- 创建索引时可以通过name字段指定索引的名字
- 实例操作



2、 唯一索引

- 唯一索引的主要目的是用在某一个字段上，使该字段的内容不重复
- 语法：db.collection_name.ensureIndex({field:1 或-1},{ “unique” :true})
- 实例操作



3、 删除索引

- 删除所有索引： db.name.dropIndexes()
- 删除指定的索引： db.name.dropIndex()
- 既可以根据key删除索引，也可以根据索引名删除
- _id_索引不能删除

4-9.聚合操作

菜鸟教程MongDB聚合

地址：https://www.runoob.com/mongodb/mongodb-aggregate.html

二、启动数据库命令

窗口1： cmd ---> mongod --config E:\mongodb\mongodb.conf

窗口2： cmd ---> mongo

检测：窗口2 :show dbs

1.MongoDB安装与配置

1. 官网下载MongoDB，安装MongoDB
2. 配置环境变量：
   - MongoDB安装路径下的bin(打开bin目录即是)
- 环境变量：编辑系统变量--->Path--->复制bin目录--->加分号隔开

3. 启动MongoDB

两种启动方式前提：在任意盘创建存放数据文件夹"mongodb"--->mongdb文件夹下新建data文件夹

第一种启动方式

(1)启动服务端

- cmd启动MongoDB服务：mongod --dbpath C:\MongoDB\data (即data文件夹目录)

  【注：此时可看到27017服务器端口，等待客户端连接】

(2)通过客户端连接

- cmd--->mongo,出现“>”号说明成功进入数据库
- 测试是否连接成功：显示所有数据库 show dbs

第二种启动方式

- 新建.conf文件“mongodb.conf"，和data在同一目录下
- 新建文件夹存放日志“logs”,在logs文件夹下创建mongo.log

设置mongdb.conf的参数：

- #设置数据目录的路径：dbpath=data文件夹目录（打开data即可）
- #设置日志文件的文件路径: logpath=mongo.log的路径
- #打开日志输出操作:logappend=true
- #端口号 默认为27017：port=27017

通过配置文件启动MongDB

(1)启动服务端

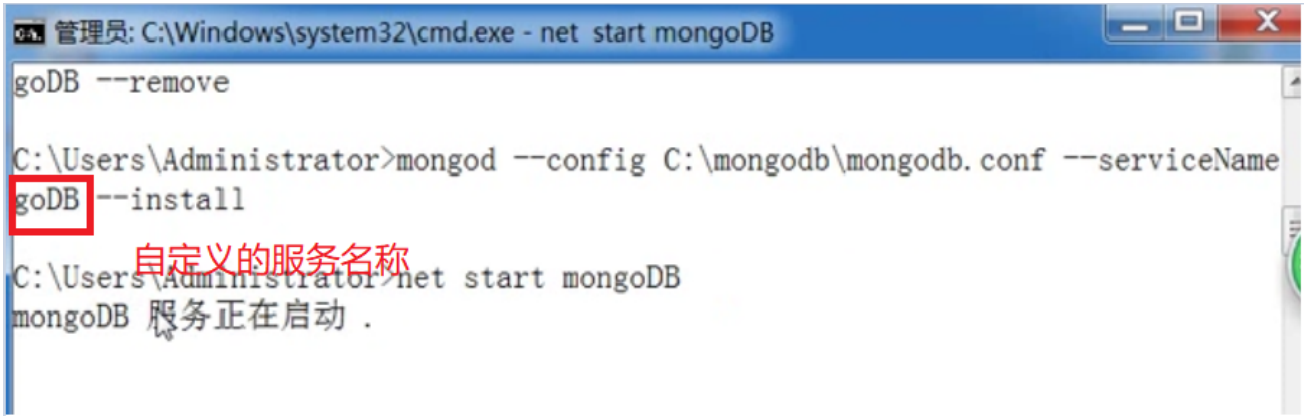
cmd ---> mongod --config E:\mongodb\mongodb.conf（mongodb.conf的地址) 【注：光标闪烁就成功】

(2) 连接客户端：cmd ---> mongo

测试连接是否成功：show dbs

2. 将MongDB作为本地服务启动

（每次启动MongDB都需要启动服务器，连接客户端，一般是第二种启动方式）



NoSql数据库特点

针对web2.0数据量大

高并发需求：如生成动态数据

高扩展性和高可用性

三、DOS命令行

cls：清屏

打开命令行窗口

1. WIN+R
2. 开始+系统+命令提示符
3. 我的电脑+cmd(空格)+X盘，eg: cmd D:\

盘符切换

名称	功能	写法
md	建立文件夹	md 文件名
	进入其他盘	e:
dir	查看当前目录下所有文件	dir
cd	切换目录（e盘换f盘）	e:\ cd /d f:\Android
cd..	#返回上一级	
del	删除文件	删除文件夹前，先删除文件

其他操作

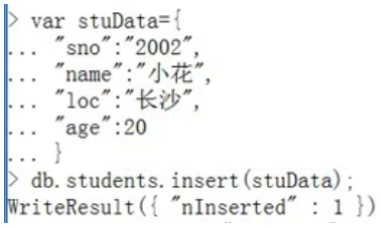
```
#基础操作
清屏：cls
退出：exit
查看电脑 IP:ipcofig

#打开软件
打开计算器：calc
打开画图工具：mspaint
新建记事本：notepad

#ping命令：查看网站IP等作用
ping.www.baidu.com

#文件操作
md  目录名
rd  目录名
cd  文件名
del 文件名
```

四、注意事项

1. $set 只能修改单个key
2. 在mongo中，有一个命令非常的方便，就是upsert，顾名思义就是update+insert的作用，根据条件判断有无记录，有的话就更新记录，没有的话就插入一条记录
3. $inc 只能用于数值型，用于增加或减少
4. 键值对匹配：eg: db . student . find({"parents.age":46}),
5. 格式化显示：db.student.find().pretty()
6. 插入大量数据：

```
db.student.insert({
"name":"sam",
"age":18,
"course":["java","ps"],
"parent":[
            {"name":"mom","age":45},
            {"name":"father","age":49}
    ]
});
```

7. 通过js添加文档



8.一次插入10000条数据：

```
for(var x=1;x<=10000;x++){

db.student.insert({"女朋友": "1"+x});

}
/*
  * Type "it" for more :输入It再显示更多内容
  * 查看文档中有多少条数据：db.test.count()
 */
```