3、shell命令1-基本操作

1、mongo-》数据库实例和集合基本操作

-----【数据库实例】-----

show dbs; 显示所有的数据库实例

use dbname ; 选择某一个数据库实例，没有也可以选择，当往数据库实例加入数据则创建

db.dropDatabase() ：这将删除当前所选数据库。 如果没有选择任何数据库，那么它将删除默认的' test '数据库。

-----【数据库集合】-----

db.createCollection(name); 在当前的数据库实例里创建集合，**name** 是要创建的集合的名称。

db.collectionName.drop(); 删除当前数据库实例中指定的collectionName的集合

show collections; 检查创建的集合。

2、mongo-》文档基本操作（crud）

a）插入文档：db.集合名称.insert(document)

插入文档时，如果不指定_id参数，MongoDB会为文档分配一个唯一的ObjectId

```
db.stu.insert({name:'gj',gender:1})

s1={_id:'20160101',name:'hr'}
s1.gender=0
db.stu.insert(s1)
```

b）查询所有：db.集合名称.find()

查询指定集合的所有数据

c）更新指定文档（1到多）：db.集合名称.update({query}, {update}, [{multi: boolean}])

参数query:查询条件，类似sql语句update中where部分

参数update:更新操作符，类似sql语句update中set部分

参数multi:可选，默认是false，表示只更新找到的第一条记录，值为true表示把满足条件的文档全部更新

【全文档更新】

```
db.stu.update({name:'hr'},{name:'mnc'})
```

【指定属性更新，通过操作符$set】

```
db.stu.insert({name:'hr',gender:0})
db.stu.update({name:'hr'},{$set:{name:'hys'}})
```

修改多条匹配到的数据

```
db.stu.update({},{$set:{gender:0}},{multi:true})
```

ps：在操作的数据是（数组[]或文档{}）的时候

"属性.下标"---》操作数组对应元素

"属性.属性名"---》操作文档对应元素

【指定属性，自增自减操作】

```
 db.student.update({_id:22},{$inc:{sage:2}})
```

【其他常用操作】

$currentDate 将字段的值设置为当前日期，作为日期或时间戳。

$push 将项目添加到数组。

$pop 删除数组的第一个或最后一个项目。

$addToSet 只有组件中不存在元素才能将数组添加到数组中。

$each 修改$push和$addToSet运算符以追加数组更新的多个项目。

d）保存指定文档（1）：db.集合名称.save(document)

如果文档的_id已经存在则修改，如果文档的_id不存在则添加

```
db.stu.save({_id:'20160102','name':'yk',gender:1})
db.stu.save({_id:'20160102','name':'wyk'})
```

e）删除指定文档（1到多）：db.集合名称.remove({query}, [{justOne: boolean}])

参数query:可选，删除的文档的条件

参数justOne:可选，如果设为true或1，则只删除一条，默认false，表示删除多条

只删除匹配到的第一条

```
db.stu.remove({gender:0},{justOne:true})
```

全部删除

db.stu.remove({})

3、mongo-》文档查询（筛选列select，条件where，排序sort，分页limit，去重distinct，统计个数count）

a）筛选列（投影）类似于sql的select

在查询到的返回结果中，只选择查询部分需要的字段，而不是所有的字段，可以提高查询的性能。

语法：db.集合名称.find({},{字段名1:1,字段名2:0,...})

参数为【字段名与值】，值为1表示显示，值为0不显示，对于普通列，默认为0，_id列默认为1。

```
db.stu.find({},{name:1,gender:1})
显示：_id，name和gender
db.stu.find({},{_id:0,name:1,gender:1})
显示：name和gender
```

b）筛选行：类似于sql的where

在查询到的返回结果中，把需要的（满足条件的）数据行（文档）查询出来。

语法：

```
db.集合名称.find({条件文档}); 查询满足条件的【所有】的文档
db.集合名称.findOne({条件文档}) 查询满足条件的【第一个】的文档
db.集合名称.find({条件文档}).pretty()  查询满足条件的【所有】的文档，并把结果格式化显示
```

一个条件：比较（关系）运算符

【值与值的关系】

```
默认是等于判断，没有运算符
小于$lt
小于或等于$lte
大于$gt
大于或等于$gte
不等于$ne
------------------
查询名称等于'gj'的学生
db.stu.find({name:'gj'})
查询年龄大于或等于18的学生
db.stu.find({age:{$gte:18}})
```

【值与集合的关系】

使用"$in"（在集合），"$nin"（不在集合）判断是否在某个集合内

查询年龄为18或28的学生

```
db.stu.find({age:{$in:[18,28]}})
```

【字符串正则匹配】

$regex或// 编写正则

查询姓黄的学生

```
db.stu.find({name:/^黄/})
db.stu.find({name:{$regex:'^黄'}})
```

多个条件：逻辑与（并且），逻辑或（或者）

查询时可以有多个条件，多个条件之间需要通过逻辑运算符连接，类似于sql的and和or

【逻辑与】：默认是逻辑与的关系

查询年龄大于或等于18，并且性别为1的学生

```
db.stu.find({
age:{$gte:18},
gender:1   })
```

【逻辑或】：使用$or

查询年龄大于18，或性别为0的学生

```
db.stu.find({$or:[
{age:{$gt:18}},
{gender:1}
]})
```

【逻辑与和逻辑或一起】

查询年龄大于18或性别为0的学生，并且学生的姓名为gj

```
db.stu.find({
$or:[{age:{$gte:18}},{gender:1}],
name:'gj'})
```

自定义条件：如果不习惯上面的$关键字，可以编写js函数(利用js语法)来手动进行条件过滤。

$where后面写一个函数，返回满足条件的数据

语法：$where:function(){
　　this：代码当前的文档数据
　　return：true显示当前文档，false不显示
}
　　查询年龄大于30的学生
db.stu.find({$where:function(){return this.age>30}})
　　查询年龄大于30小于70,且名字以x开头的学生
db.student.find({
$where:function(){
var b1=this.sage>30&&this.sage<70;
var b2=/^x/.test(this.sname);
return b1;
}});

c）排序-数据行
语法：
db.集合名称.find().sort({字段1:1,字段2:-1,...})
参数1为升序排列
参数-1为降序排列

根据性别降序，再根据年龄升序
db.stu.find().sort({gender:-1,age:1})

c）分页-查询部分数据行（类似于mysql的limit）
skip：跳过的行数
limit：需要查询的行数
方法limit()和skip()可以一起使用，不分先后顺序
查询第5至8条数据
db.stu.find().limit(4).skip(5)
或
db.stu.find().skip(5).limit(4)

d）统计个数
db.集合名称.find({条件}).count()
db.集合名称.count({条件})

　　统计男生人数
db.stu.find({gender:1}).count()
　　统计年龄大于20的男生人数
db.stu.count({age:{$gt:20},gender:1})
e）去除重复数据（单个-组合需要通过聚合aggregate**）**
语法：
db.集合名称.distinct('去重字段',{条件})

　　查找年龄大于18的性别（去重）
db.stu.distinct('gender',{age:{$gt:18}})