shell操作数据库：

1. 超级用户相关：
    1. #进入数据库admin `use admin`
    2. #增加或修改用户密码 `db.addUser('name','pwd')`
    3. #查看用户列表 `db.system.users.find()`
    4. #用户认证 `db.auth('name','pwd')`
    5. #删除用户 `db.removeUser('name')`
    6. #查看所有用户 `show users`
    7. #查看所有数据库 `show dbs`
    8. #查看所有的collection `show collections`
    9. #查看各collection的状态 `db.printCollectionStats()`
    10. #查看主从复制状态 `db.printReplicationInfo()`
    11. #修复数据库 `db.repairDatabase()`
    12. #设置记录profiling，0=off 1=slow 2=all `db.setProfilingLevel(1)`
    13. #查看profiling `show profile`
    14. #拷贝数据库 `db.copyDatabase('mail_addr','mail_addr_tmp')`
    15. #删除collection `db.mail_addr.drop()`
    16. #删除当前的数据库 `db.dropDatabase()`

2. 增删改
    1. #存储嵌套的对象
    1. `db.foo.save({'name':'ysz','address':{'city':'beijing','post':100096},'phone':[138,139]})`
    2. #存储数组对象
    2. `db.user_addr.save({'Uid':'yushunzhi@sohu.com','Al':['test-1@sohu.com','test-2@sohu.com']})`
    3. #根据query条件修改，如果不存在则插入，允许修改多条记录
    3. `db.foo.update({'yy':5},{'$set':{'xx':2}},upsert=true,multi=true)`
    4. #删除yy=5的记录 `db.foo.remove({'yy':5})`
    5. #删除所有的记录 `db.foo.remove()`

3. 索引
    1. #增加索引：1(ascending),-1(descending)
    2. `db.foo.ensureIndex({firstname: 1, lastname: 1}, {unique: true});`
    3. #索引子对象
    4. `db.user_addr.ensureIndex({'Al.Em': 1})`
    5. #查看索引信息
    6. `db.foo.getIndexes()`
    7. `db.foo.getIndexKeys()`
    8. #根据索引名删除索引
    9. `db.user_addr.dropIndex('Al.Em_1')`

4. 查询
    1. #查找所有
    2. `db.foo.find()`
    3. #查找一条记录
    4. `db.foo.findOne()`
    5. #根据条件检索10条记录
    6. `db.foo.find({'msg':'Hello 1'}).limit(10)`
    7. #sort排序
    8. `db.deliver_status.find({'From':'ixigua@sina.com'}).sort({'Dt',-1})`
    9. `db.deliver_status.find().sort({'Ct':-1}).limit(1)`
    10. #count操作
    11. `db.user_addr.count()`
    12. #distinct操作,查询指定列，去重复
    13. `db.foo.distinct('msg')`
    14. #">="操作
    15. `db.foo.find({"timestamp": {"$gte" : 2}})`
    16. #子对象的查找
    17. `db.foo.find({'address.city':'beijing'})`

5. 管理
    1. #查看collection数据的大小
    2. `db.deliver_status.dataSize()`
    3. #查看colleciont状态
    4. `db.deliver_status.stats()`
    5. #查询所有索引的大小
    6. `db.deliver_status.totalIndexSize()`

        1. advanced queries:高级查询

条件操作符

```
$gt : >
$lt : <
$gte: >=
$lte: <=
$ne : !=、<>
$in : in
$nin: not in
$all: all
```

$not: 反匹配(1.3.3及以上版本)

查询 name <> “bruce” and age >= 18 的数据

```
db.users.find({name: {$ne: "bruce"}, age: {$gte: 18}});
```

查询 creation_date > ‘2010-01-01’ and creation_date <= ‘2010-12-31’ 的数据

```
db.users.find({creation_date:{$gt:new Date(2010,0,1), $lte:new Date(2010,11,31)}});
```

查询 age in (20,22,24,26) 的数据

```
db.users.find({age: {$in: [20,22,24,26]}});
```

查询 age取模10等于0 的数据

```
db.users.find('this.age % 10 == 0');
```

或者

```
db.users.find({age : {$mod : [10, 0]}});
```

匹配所有

```
db.users.find({favorite_number : {$all : [6, 8]}});
```

可以查询出`{name: 'David', age: 26, favorite_number: [ 6, 8, 9 ] }`
可以不查询出`{name: 'David', age: 26, favorite_number: [ 6, 7, 9 ] }`

查询不匹配name=B*带头的记录

```
db.users.find({name: {$not: /^B.*/}});
```

查询 age取模10不等于0 的数据

```
db.users.find({age : {$not: {$mod : [10, 0]}}});
```

返回部分字段

选择返回age和_id字段(_id字段总是会被返回)

```
db.users.find({}, {age:1});
db.users.find({}, {age:3});
db.users.find({}, {age:true});
db.users.find({ name : "bruce" }, {age:1});
```

0为false, 非0为true

选择返回age、address和_id字段

```
db.users.find({ name : "bruce" }, {age:1, address:1});
```

排除返回age、address和_id字段

```
db.users.find({}, {age:0, address:false});
db.users.find({ name : "bruce" }, {age:0, address:false});
```

数组元素个数判断
对于{name: 'David', age: 26, favorite_number: [6, 7, 9] }记录
匹配 `db.users.find({favorite_number: {$size: 3}});`
不匹配 `db.users.find({favorite_number: {$size: 2}});`

$exists判断字段是否存在
查询所有存在name字段的记录

```
db.users.find({name: {$exists: true}});
```

查询所有不存在phone字段的记录

```
db.users.find({phone: {$exists: false}});
```

$type判断字段类型
查询所有name字段是字符类型的

```
db.users.find({name: {$type: 2}});
```

查询所有age字段是整型的

```
db.users.find({age: {$type: 16}});
```

对于字符字段，可以使用正则表达式
查询以字母b或者B带头的所有记录

```
db.users.find({name: /^b.*/i});
```

$elemMatch(1.3.1及以上版本)
为数组的字段中匹配其中某个元素

Javascript查询和$where查询
查询 age > 18 的记录，以下查询都一样

```
db.users.find({age: {$gt: 18}});
db.users.find({$where: "this.age > 18"});
db.users.find("this.age > 18");
f = function() {return this.age > 18} db.users.find(f);
```

排序sort()
以年龄升序asc

```
db.users.find().sort({age: 1});
```

以年龄降序desc

```
db.users.find().sort({age: -1});
```

限制返回记录数量limit()
返回5条记录

```
db.users.find().limit(5);
```

返回3条记录并打印信息

```
db.users.find().limit(3).forEach(function(user) {print('my age is ' + user.age)});
```

结果

```
my age is 18
my age is 19
my age is 20
```

限制返回记录的开始点skip()
从第3条记录开始，返回5条记录(limit 3, 5)

```
db.users.find().skip(3).limit(5);
```

查询记录条数count()

```
db.users.find().count();
db.users.find({age:18}).count();
```

以下返回的不是5，而是user表中所有的记录数量

```
db.users.find().skip(10).limit(5).count();
```

如果要返回限制之后的记录数量，要使用count(true)或者count(非0)

```
db.users.find().skip(10).limit(5).count(true);
```

分组group()
假设test表只有以下一条数据

```
{ domain: "www.mongodb.org"
, invoked_at: {d:"2009-11-03", t:"17:14:05"}
, response_time: 0.05
, http_action: "GET /display/DOCS/Aggregation"
}
```

使用group统计test表11月份的数据count:count(*)、total_time:sum(response_time)、avg_time:total_time/count;

```
db.test.group(
{ cond: {"invoked_at.d": {$gt: "2009-11", $lt: "2009-12"}}
, key: {http_action: true}
, initial: {count: 0, total_time:0}
, reduce: function(doc, out){ out.count++; out.total_time+=doc.response_time }
, finalize: function(out){ out.avg_time = out.total_time / out.count }
} );

[
{
"http_action" : "GET /display/DOCS/Aggregation",
"count" : 1,
"total_time" : 0.05,
"avg_time" : 0.05
}
]
```