**一、超级用户相关：**

* 进入数据库admin  
  use admin
* 增加或修改用户密码  
  db.addUser(‘name’,’pwd’)
* 查看用户列表  
  db.system.users.find()
* 用户认证  
  db.auth(‘name’,’pwd’)
* 删除用户  
  db.removeUser(‘name’)
* 查看所有用户  
  show users
* 查看所有数据库  
  show dbs
* 查看所有的collection  
  show collections
* 查看各collection的状态  
  db.printCollectionStats()
* 查看主从复制状态  
  db.printReplicationInfo()
* 修复数据库  
  db.repairDatabase()
* 设置记录profiling，0=off 1=slow 2=all  
  db.setProfilingLevel(1)
* 查看profiling  
  show profile
* 拷贝数据库  
  db.copyDatabase(‘mail\_addr’,’mail\_addr\_tmp’)
* 删除collection  
  db.mail\_addr.drop()
* 删除当前的数据库  
  db.dropDatabase()

**二、增删改**

* 存储嵌套的对象  
  db.foo.save({‘name’:’imdst’,’address’:{‘city’:’guangzhou’,’post’:100096},’phone’:[158,155]})
* 存储数组对象  
  db.user\_addr.save({‘Uid’:’leoiceo@163.com’,’Al’:['test-1@163.com','test-2@163.com']})
* 根据query条件修改，如果不存在则插入，允许修改多条记录  
  db.foo.update({‘yy’:5},{‘$set’:{‘xx’:2}},upsert=true,multi=true)
* 删除yy=5的记录  
  db.foo.remove({‘yy’:5})
* 删除所有的记录  
  db.foo.remove()

**三、索引**

* 增加索引 1(ascending),-1(descending)  
  db.foo.ensureIndex({firstname: 1, lastname: 1}, {unique: true});
* 索引子对象  
  db.user\_addr.ensureIndex({‘Al.Em’: 1})
* 查看索引信息  
  db.foo.getIndexes()  
  db.foo.getIndexKeys()
* 根据索引名删除索引  
  db.user\_addr.dropIndex(‘Al.Em\_1′)

**四、查询**

* 查找所有  
  db.foo.find()
* 查找一条记录  
  db.foo.findOne()
* 根据条件检索10条记录  
  db.foo.find({‘msg’:’Hello 1′}).limit(10)
* sort排序  
  db.deliver\_status.find({‘From’:’ixigua@sina.com’}).sort({‘Dt’,-1}) db.deliver\_status.find().sort({‘Ct’:-1}).limit(1)
* count操作  
  db.user\_addr.count()
* distinct操作,查询指定列，去重复  
  db.foo.distinct(‘msg’)
* #”>=”操作  
  db.foo.find({“timestamp”: {“$gte” : 2}})
* 子对象的查找  
  db.foo.find({‘address.city’:’beijing’})

**五、管理**

* 查看collection数据的大小  
  db.deliver\_status.dataSize()
* 查看colleciont状态  
  db.deliver\_status.stats()
* 查询所有索引的大小  
  db.deliver\_status.totalIndexSize()
* advanced queries:高级查询

条件操作符

$gt : >

$lt : <<br> $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′);  
  db.users.find({age : {$mod : [10, 0]}});
* 查询所有name字段是字符类型的  
  db.users.find({name: {$type: 2}});
* 查询所有age字段是整型的  
  db.users.find({age: {$type: 16}});
* 对于字符字段，可以使用正则表达式 查询以字母b或者B带头的所有记录  
  db.users.find({name: /^b.\*/i});
* 匹配所有  
  db.users.find({favorite\_number : {$all : [6, 8]}});
* 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: {$mod : [10, 0]}}});
* 选择返回age和*id字段(*id字段总是会被返回)  
  db.users.find({}, {age:1});  
  db.users.find({}, {age:3});  
  db.users.find({}, {age:true});  
  db.users.find({ name : “bruce” }, {age:1});  
  0为false, 非0为true
* 选择返回age、address和\_id字段  
  db.users.find({ name : “bruce” }, {age:1, address:1});
* 排除返回age、address和\_id字段  
  db.users.find({}, {age:0, address:false});  
  db.users.find({ name : “bruce” }, {age:0, address:false});
* 查询所有存在name字段的记录  
  db.users.find({name: {$exists: true}});
* 排序sort()
  + 查询所有不存在phone字段的记录  
    db.users.find({phone: {$exists: false}});
  + 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:”2015-05-03″, 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: “2015-05″, $lt: “2015-06″}}**

**, 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**

**}**

**]**