---

link: null
title: 珠峰架构师成长计划
description: 这命令是保存成了文件格式
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=95 sentences=128, words=738

---

1. 通过配置项启动数据库

参数 含义 --dbpath 指定数据库文件的目录 --port 端口 默认是27017 28017 --fork 以后台守护的方式进行启动 --logpath 指定日志文件输出路径 --config 指定一个配置文件 --auth 以安全方式启动数据库，默认不验证

1.1 mongo.conf

```
dbpath=E:\mongo\data
logpath=E:\mongo\log
port=50000
```

1.2 启动服务器

```
mongod --config mongo.conf
```

1.3 启动客户端

```
mongo --port 50000
```

2. 导入导出数据

这命令是保存成了文件格式

- mongoimport 导出数据
- mongoexport 导入数据

参数 含义 -h [--host] 连接的数据库 --port 端口号 -u 用户名 -p 密码 -d 导出的数据库 -d 导出的数据库 -c 指定导出的集合 -o 导出的文件存储路径 -q 进行过滤

2.1 准备数据

```
use school;
var students = [];
for(var i=1;i10;i++){
    students.push({name:'zfpx'+i,age:i});
}
db.students.insert(students);
db.students.find();
```

2.2 备份记录

```
mongoexport -h 127.0.0.1 --port 50000  -d school -c students -o stu.bak
```

2.3 删除记录

```
> db.students.remove({});
WriteResult({ "nRemoved" : 10 })
```

2.4 导入记录

```
mongoimport --port 50000 --db school --collection students --file
stu.bak
```

3. 备份与恢复

3.1 mongodump

在Mongodb中我们使用mongodump命令来备份MongoDB数据。该命令可以导出所有数据到指定目录中。

> mongodump -h dbhost -d dbname -o dbdirectory

- -h MongDB所在服务器地址，例如：127.0.0.1，当然也可以指定端口号：127.0.0.1:27017
- -d 需要备份的数据库实例，例如：test
- -o 备份的数据存放位置

```
mongodump  -d school -o data.dmp
```

3.2 mongorestore

mongodb使用 mongorestore 命令来恢复备份的数据。

- --host MongoDB所在服务器地址
- --db -d 需要恢复的数据库实例
- 最后的一个参数，设置备份数据所在位置

```
mongorestore  data.dmp
mongorestore -d school data.bmp/school
```

> Mongodump可以backup整个数据库，而mongoexport要对每个collection进行操作，最主要的区别也是选择的标准是mongoexport输出的JSON比Mongodump的BSON可读性更高，进而可以直接对JSON文件进行操作然后还原数据（BSON转换JSON存在潜在兼容问题）。

4. 直接拷贝数据

5. 锁定和解锁数据库

为了数据的完整性和一致性，导出前要先锁定写入，导出后再解锁。

```
> use admin;
switched to db admin
> db.runCommand({fsync:1,lock:1});
{
        "info" : "now locked against writes, use db.fsyncUnlock() to unlock",
        "seeAlso" : "http://dochub.mongodb.org/core/fsynccommand",
        "ok" : 1
}
> db.fsyncUnlock();
{ "ok" : 1, "info" : "unlock completed" }
```

6. 安全措施

- 物理隔离
- 网络隔离
- 防火墙(IP/IP段/白名单/黑名单)
- 用户名和密码验证

6.1 用户管理

6.1.1 查看角色

```
show roles;
```

内置角色

- 数据库用户角色：read、readWrite；
- 数据库管理角色：dbAdmin、dbOwner、userAdmin;
- 集群管理角色：clusterAdmin、clusterManager、clusterMonitor、hostManage；
- 备份恢复角色：backup、restore；
- 所有数据库角色：readAnyDatabase、readWriteAnyDatabase、userAdminAnyDatabase、dbAdminAnyDatabase
- 超级用户角色：root
- 内部角色：__system

6.1.2 老的创建用户的方法

```
> db.addUser('zfpx','123456');
WARNING: The 'addUser' shell helper is DEPRECATED. Please use 'createUser' inste
ad
Successfully added user: { "user" : "zfpx", "roles" : [ "root" ] }
show roles;
```

6.1.3 新的创建用户的方法

```
db.createUser({
    user:"zfpx2",
    pwd:"123456",
    roles:[
        {
            role:"readWrite",
            db:"school"
        },
        'read'
   ]
})
```

```
> db.createUser({user:'zfpx2',pwd:'123456',roles:[{role:'read',db:'school'}]});
Successfully added user: {
        "user" : "zfpx2",
        "roles" : [
                {
                        "role" : "read",
                        "db" : "school"
                }
        ]
}
```

6.1.4 查看用户的权限

```
> db.runCommand({usersInfo:'zfpx2',showPrivileges:true});
{
        "users" : [
                {
                        "_id" : "admin.zfpx2",
                        "user" : "zfpx2",
                        "db" : "admin",
                        "roles" : [
                                {
                                        "role" : "read",
                                        "db" : "school"
                                }
                ]
}
```

6.1.5 服务器启动权限认证

```
dbpath=E:\mongo\data
logpath=E:\mongo\log
port=50000
auth=true
```

6.1.6 用户登录和修改密码

```
> use admin;
> use admin;
switched to db admin
> db.auth('zfpx','123456')
1
> db.changeUserPassword('zfpx','123');
> db.auth('zfpx','123')
1
```

6.1.7 修改个人信息

```
db.runCommand({updateUser:'zfpx',pwd:'123', customData:{
        name:'珠峰培训',
        email:'zfpx@126.com',
        age:18,
}});
> db.runCommand({usersInfo:'zfpx',showPrivileges:true})
```

- 用户的操作都需要在admin数据库下面进行操作
- 如果在某个数据库下面执行操作，那么只对当前数据库生效
- addUser已经废弃，默认会创建root用户，不安全，不再建议使用

7. 数据库高级命令

7.1 准备数据

```
var students = [
        {name:'zfpx1',home:'北京',age:1},
        {name:'zfpx2',home:'北京',age:2},
        {name:'zfpx3',home:'北京',age:3},
        {name:'zfpx4',home:'广东',age:1},
        {name:'zfpx5',home:'广东',age:2},
        {name:'zfpx6',home:'广东',age:3}
]
db.students.insert(students);
```

7.2 count

查看记录数

```
db.students.find().count();
```

7.2 查找不重复的值 distinct

```
db.runCommand({distinct:'students',key:'home'}).values;

[ "北京", "广东" ]
```

7.3 group 分组

```
db.runCommand({
        group:{
                ns:集合名称,
                key:分组的键,
                initial:初始值,
                $reduce:分解器
                condition:条件,
                finalize:完成时的处理器
        }
});
```

7.3.1 按城市分组，求每个城市里符合条件的人的年龄总和

```
db.runCommand({
        group:{
                ns:'students',
                key:{home:true},
                initial:{total:0},
                $reduce:function(doc,result){
                      result.total += doc.age;
                },
                condition:{age:{$gt:1}},
                finalize:function(result){
                    result.desc = '本城市的总年龄为'+result.total;
                }
        }
});
```

7.4 删除集合

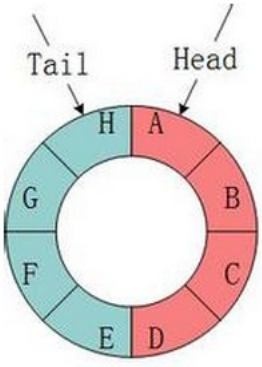
```
db.runCommand({drop:'students'});
```

7.5 runCommand常用命令

```
db.runCommand({buildInfo:1});
db.runCommand({getLastError:"students"});
db.persons.insert({_id:1,name:1});
db.persons.insert({_id:1,name:1});
db.runCommand({getLastError:"students"});
```

8. 什么固定集合

MongoDB 固定集合（Capped Collections）是性能出色且有着固定大小的集合，对于大小固定，我们可以想象其就像一个环形队列，当集合空间用完后，再插入的元素就会覆盖最初始的头部的元素！



8.1 特性

- 1. 没有索引
- 1. 插入和查询速度速度非常快 不需要重新分配空间
- 1. 特别适合存储日志

8.2 创建固定集合

- 我们通过createCollection来创建一个固定集合，且capped选项设置为true：
- 还可以指定文档个数,加上max:1000属性：
- 判断集合是否为固定集合: db.logs.isCapped()
- size 是整个集合空间大小，单位为【KB】
- max 是集合文档个数上线，单位是【个】
- 如果空间大小到达上限，则插入下一个文档时，会覆盖第一个文档；如果文档个数到达上限，同样插入下一个文档时，会覆盖第一个文档。两个参数上限判断取的是【与】的逻辑。
- capped 封顶的

```
db.createCollection('logs',{size:50,max:5,capped:true});
```

8.3 非固定集合转为固定集合

```
db.runCommand({convertToCapped:"logs",size:5});
```

9. gridfs

- gridfs是mongodb自带的文件系统，使用二进制存储文件。
- mongodb可以以BSON格式保存二进制对象。
- 但是BSON对象的体积不能超过4M。所以mongodb提供了 mongofiles。它可以把一个大文件透明地分割成小文件（256K），从而保存大体积的数据。
- GridFS 用于存储和恢复那些超过16M（BSON文件限制）的文件(如：图片、音频、视频等)。
- GridFS 用两个集合来存储一个文件：fs.files与fs.chunks。
- 每个文件的实际内容被存在chunks(二进制数据)中,和文件有关的meta数据(filename,content_type,还有用户自定义的属性)将会被存在files集合中。
- **9.1 上传一个文件#**
- -d 数据库的名称
- -l 源文件的位置
- put 指定文件名

```
mongofiles -d myfiles put test.txt
```

** 9.2 获取并下载文件 #**

```
mongofiles -d myfiles  get 'test.txt'
```

** 9.3 查看所有文件 #**

```
mongofiles -d myfiles  list
>db.fs.files.find()
>db.fs.chunks.find({files_id:ObjectId('')})
```

** 9.4 删除文件 #**

```
mongofiles -d myfiles delete "test.txt"
```

** 9.5 eval 服务器端脚本 #**

- 执行JS语句
- 定义JS全局变量
- 定义函数
- Stored JavaScript

```
db.eval("1+1");
db.eval("return 'hello'");
db.system.js.insert({_id:"x",value:1});
db.eval("return x");
db.system.js.insert({_id:"say",value:function(){return 'hello'}});
db.eval("say()");
```