[一 重要概念 3](#_Toc381340536)

[1. 文档 3](#_Toc381340537)

[2. 集合 3](#_Toc381340538)

[3. 实例 3](#_Toc381340539)

[4. \_id 3](#_Toc381340540)

[二 服务配置 3](#_Toc381340541)

[1. 下载MongoDB压缩包。 3](#_Toc381340542)

[2. 解压包，并将解压后的bin文件夹拷到MONGODB\_HOME 3](#_Toc381340543)

[3. 在MONGODB\_HOME下建立如下文件夹（默认会建立在C:\data\） 3](#_Toc381340544)

[三 服务启动 4](#_Toc381340545)

[1. 程序启动 4](#_Toc381340546)

[2. window服务 4](#_Toc381340547)

[四 服务停止 4](#_Toc381340548)

[五 客户端访问 4](#_Toc381340549)

[1. 配置环境变量，在path变量中加入MongoDB\_HOME/bin 4](#_Toc381340550)

[2. 创建bat文件启动 4](#_Toc381340551)

[六 创建数据库 4](#_Toc381340552)

[1. use firstdb 创建一个数据库 4](#_Toc381340553)

[2. show dbs 查看所有的数据库 4](#_Toc381340554)

[3. 只有往数据库firstdb的某一个集合中插入一个文档，此时数据库firstdb才真的被创建。 4](#_Toc381340555)

[七 删除数据库 5](#_Toc381340556)

[八 插入文档 5](#_Toc381340557)

[1. insert 5](#_Toc381340558)

[2. save 5](#_Toc381340559)

[3. 区别 5](#_Toc381340560)

[4. 注意 5](#_Toc381340561)

[九 删除文档 5](#_Toc381340562)

[1. remove 5](#_Toc381340563)

[2. drop 5](#_Toc381340564)

[十 更新文档 5](#_Toc381340565)

[1. 强制修改文档 5](#_Toc381340566)

[2. insert or update 5](#_Toc381340567)

[3. 批量更新 5](#_Toc381340568)

[4. $set 6](#_Toc381340569)

[5. $inc 6](#_Toc381340570)

[6. $unset 6](#_Toc381340571)

[7. $push 6](#_Toc381340572)

[8. $pushAll 6](#_Toc381340573)

[9. $addToSet 6](#_Toc381340574)

[10. $pop 6](#_Toc381340575)

[11. $pull 6](#_Toc381340576)

[12. $pullAll 7](#_Toc381340577)

[13. $数组占位符 7](#_Toc381340578)

[14. $addToSet和$each批量更新数组 7](#_Toc381340579)

[十一 查询文档 7](#_Toc381340580)

[1. 显示所有文档的所有键 7](#_Toc381340581)

[2. 显示所有文档的部分键 7](#_Toc381340582)

[3. 查询条件 7](#_Toc381340583)

[4. limit 取前几条数据 9](#_Toc381340584)

[5. skip 跳过前几条数据 9](#_Toc381340585)

[6. sort 排序 9](#_Toc381340586)

[十二 索引 9](#_Toc381340587)

[1. 创建索引 10](#_Toc381340588)

[2. 创建时 同时指定索引名称 10](#_Toc381340589)

[3. 唯一索引 10](#_Toc381340590)

[4. 剔除重复的文档 10](#_Toc381340591)

[5. 查询数据时 强制指定照哪个索引查找 10](#_Toc381340592)

[6. 查看查询数据使用哪个索引 10](#_Toc381340593)

[7. 查看数据库中已有的索引 10](#_Toc381340594)

[8. 后台执行 10](#_Toc381340595)

[9. 删除索引 10](#_Toc381340596)

[十三 2D索引 11](#_Toc381340597)

[1. 创建2D 索引 11](#_Toc381340598)

[2. $near 11](#_Toc381340599)

[3. $within,$box,$center 11](#_Toc381340600)

[十四 count、distinct、group、runCommand 12](#_Toc381340601)

[1. count 12](#_Toc381340602)

[2. distinct 12](#_Toc381340603)

[3. group 12](#_Toc381340604)

[4. runCommand 14](#_Toc381340605)

[十五 固定集合 14](#_Toc381340606)

[1. 创建 14](#_Toc381340607)

[2. 普通集合转为固定集合 14](#_Toc381340608)

[3. $natural 14](#_Toc381340609)

[十六 gridfs文件系统 15](#_Toc381340610)

[1. 主要参数说明 15](#_Toc381340611)

[2. 上传文件 15](#_Toc381340612)

[3. 信息查看 15](#_Toc381340613)

[4. 下载文件 15](#_Toc381340614)

[十七 system.js 15](#_Toc381340615)

[1. 服务端运行eval 15](#_Toc381340616)

[2. js存储 15](#_Toc381340617)

[十八 集合数据导出 16](#_Toc381340618)

[1. 参数说明 16](#_Toc381340619)

[2. 示例 16](#_Toc381340620)

[十九 数据导入集合 16](#_Toc381340621)

[1. 参数说明 16](#_Toc381340622)

[2. 示例 16](#_Toc381340623)

[二十 运行时导出数据库 16](#_Toc381340624)

[1. 参数说明 16](#_Toc381340625)

[2. 示例 17](#_Toc381340626)

[二十一 运行时导入数据库 17](#_Toc381340627)

[1. 参数说明 17](#_Toc381340628)

[2. 示例 17](#_Toc381340629)

[二十二 锁和数据修复 17](#_Toc381340630)

[1. 上锁 17](#_Toc381340631)

[2. 解锁 17](#_Toc381340632)

[3. 数据修复 17](#_Toc381340633)

[二十三 安全认证 17](#_Toc381340634)

[1. 添加用户 17](#_Toc381340635)

[2. 启用用户 17](#_Toc381340636)

[3. 安全检查 17](#_Toc381340637)

[4. 删除用户 18](#_Toc381340638)

# 重要概念

1. 文档

文档类似与关系型数据库中的行。

1. 集合

集合类似与关系型数据库中的没有模式的表。

1. 实例

实例可以容纳多个独立的数据库，每个都有自己的集合和权限。

1. \_id

每个文档中都有一个特殊的键“\_id”,它在文档所处的集合中是唯一的。

# 服务配置

WIN7下安装到D:/Soft/MongoDB（已经用MONGODB\_HOME代替）为例

1. 下载MongoDB压缩包。
2. 解压包，并将解压后的bin文件夹拷到MONGODB\_HOME
3. 在MONGODB\_HOME下建立如下文件夹（默认会建立在C:\data\）

MONGODB\_HOME/data/db

MONGODB\_HOME/data/log

并在MONGODB\_HOME/data/log下建立文件MongoDB.log

# 服务启动

启动方式有如下3种。

1. 程序启动

在cmd中 进入 MONGODB\_HOME的bin目录中，

执行mongod –dbpath “MONGODB\_HOME/data/db”，

在打开MONGODB\_HOME的bin目录，执行mongo.exe文件。

1. window服务

在cmd中 进入 MONGODB\_HOME的bin目录中，

执行mongod –dbpath “MONGODB\_HOME/data/db” –logpath “MONGODB\_HOME/data/log” –install -serviceName “MongoDB”，

启动服务 可以在服务中启动，也可以在cmd执行 net start MongoDB

停止服务 net stop MongoDB

删除 mongod –dbpath “MONGODB\_HOME/data/db” –logpath “MONGODB\_HOME/data/log” –remove -serviceName “MongoDB”

# 服务停止

ctrl+c或者在admin下使用db.shutdownServer()

# 客户端访问

1. 配置环境变量，在path变量中加入MongoDB\_HOME/bin
2. 创建bat文件启动

内容如下

mongo 127.0.0.1:27071/admin

# 创建数据库

1. use firstdb 创建一个数据库

此时，firstdb并没有真正的创建。

1. show dbs 查看所有的数据库
2. 只有往数据库firstdb的某一个集合中插入一个文档，此时数据库firstdb才真的被创建。

# 删除数据库

db.dropDataBase()

# 插入文档

1. insert

默认会加一个\_id的键，也可以直接指定。

db.persons.insert({\_id:”1”,name:”st”})

1. save

db.persons.save({\_id:”1”,name:”st”})

1. 区别

当遇到\_id相同的情况，insert报错，save执行保存。

1. 注意

没有批量插入的概念，必须使用其他方式。

# 删除文档

1. remove

db.persons.remove() 不会删掉集合和索引。

show collections() 查看集合。

db.persons.remove({“name”:”st”})

1. drop

db.persons.drop() 彻底删除集合和索引。

# 更新文档

1. 强制修改文档

db.persons.update({name:”st”},{age:12})

1. insert or update

db.persons.update({name:”st”},{age:12},true)

指定update第三个参数为true ,当集合中存在符合条件的文档，则修改，不存在，则插入。

1. 批量更新

update默认只更新一条数据。

指定update第四个参数为true,批量更新。

db.persons.update({name:”st”},{$set:{name:”john”}},false,true)

注意：该处第二个参数要使用$。

局部修改

1. $set

db.persons.update({name:”st”},{$set:{age:12}})

如果键存在，就修改，不存在，就添加键。

1. $inc

db.persons.update({name:”st”},{$inc:{count:2}})

如果键存在，就增加原有的值，不存在，就添加键。

1. $unset

删除指定的键

db.persons.update({name:”st”},{$unset:{count:1}})

数组操作

{

name:”st”,

age:12,

colors:

[“red”,”yellow”,”blue”]

}

1. $push

往数组中添加一个值

db.persons.update({name:”st”,{$push:{colors:”green”}})

a.如果指定的键不存在，则创建一个值为数组的键

b.如果指定的键存在，且键值为数组，则往数组中添加该值

c.如果指定的键存在，而键值不为数组，则终止操作

1. $pushAll

往数组中添加多个值（数组形式）

db.persons.update({name:”st”,{$pushAll:{colors:[”green”,”white”]}})

1. $addToSet

往数组中添加一个值，如果数组中存在该值，则不添加，反之，则添加

db.persons.update({name:”st”,{$addToSet:{colors:”red”}})

1. $pop

从数组中移走值 1 移走 最后一个值 -1 移走 第一个值

db.persons.update({name:”st”,{$pop:{colors:1}}})

1. $pull

从数组中移走指定位置的值

db.persons.update({name:”st”,{$push:{colors:”blue”}})

1. $pullAll

往数组中移走多个值（数组形式）

db.persons.update({name:”st”,{$pullAll:{colors:[”green”,”white”]}})

1. $数组占位符

对于如下形式的books的值，数组的值为文档，对它进行修改，得使用数组定位器

{

name:”st”,

age:12,

books:

[

{type:”JS”,name:”JQUERY”},

{type:”JS”,name:”EXJ”},

{type:”DB”,name:”MONGODB”}

]

}

db.persons.update({“books.type”:”JS”},{$set:{“books.$.author”:”unknow”}})

注意，“.” 必须加“”，如“books.type”,占位符修改只更新第一条匹配的数据

1. $addToSet和$each批量更新数组

db.persons.update({name:”st”},{$addToSet:{books:{$each:[“red”,”yellow”,”black”]}}})

# 查询文档

1. 显示所有文档的所有键

db.persons.find()

1. 显示所有文档的部分键

db.persons.find({},{name:1,age:1,\_id:0})

指定find函数的第二个参数，1 显示 0 不显示

\_id,默认情况下是一定显示的,指定为0 不显示

1. 查询条件

var persons=[

{

"\_id" : ObjectId("528dc18cb2dd944d22a5ced5"),

"age" : 25.0,

"books" : ["JS", "C++", "EXTJS", "MONGODB"],

"c" : 89.0,

"country" : "USA",

"e" : 87.0,

"email" : "75431457@qq.com",

"m" : 96.0,

"name" : "jim",

"school" : [{

"school" : "K",

"score" : "A"

}, {

"school" : "L",

"score" : "B"

}, {

"school" : "J",

"score" : "A+"

}],

"size" : 4.0

}

]

$gt >

$lt <

db.persons.find({$age:{$lt:25,$gt:28}},{name:1,age:1,\_id:0})

$gte >=

$lte <=

$ne !=

$in 包含

$nin 不包含

$or 或者

db.persons.find({$or:[{c:{$gt:85}},{e:{$gt:90}}]},{\_id:0,e:1,c:1})

//i 正则表达式

db.persons.find({name:/li/i},{\_id:0,name:1})

$not

db.persons.find({name:{$not:/li/i}},{\_id:0,name:1})

数组

$all 数组查询

db.persons.find({books:{$all:[“MONGODB”,”JS”]}},{\_id:0,books:1})

db.persons.find({“books.1”:”JAVA”},{\_id:0,books:1})

$size 数组长度

db.persons.find({books:{$size:4}},{\_id:0,books:1})

缺点

$size 不能和比较运算符使用

改进

a.每个文档增加一个size的键

db.persons.update({},{$set:{size:4}},false,true)----批量更新

b.往数组中插入数据时,size++

db.persons.update({contry:”China”},{$push:{books:”TEXT”},$inc:{size:1}},faslse,true)

c.执行查询

db.persons.find({size:{$gt:4}},{\_id:0,books:1,size:1})

$slice 列出数组中某段值

db.persons.find({name:”jim”},{books:{$slice:[2,3]}})

db.persons.find({name:”jim”},{books:{$slice:-1}})

数组的值为文档

1. 绝对查找

db.persons.find({school:{school:”K”,score:”A”}},{\_id:0,school:1})

缺点

查询school,要带上score;

school和score顺序不能搞反，否则，没有数据。

1. 对象.查找

db.persons.find({“school.school”:”K”,”school.score”:”A”},{\_id:0,school:1})

缺点

只要有一个条件符合，就显示。

1. 元素匹配$elemMatch

Db.person.find({school:{$elemMatch:{school:”K”,score:”A”}}},{\_id:0,school:1})

1. limit 取前几条数据

db.persons.find({},{\_id:0,name:1}).limit(5)

1. skip 跳过前几条数据

db.person.find({},{\_id:0,name:1}).skip(5)

1. sort 排序

db.persons.find({},{\_id:0,name:1,age:1}).sort({age:1}) ---1 升序 -1 降序

# 索引

数据准备

for(var i = 0 ; i<200000 ;i++){

db.books.insert({number:i,name:i+"book"})

}

1. 创建索引

为number键 创建索引

db.books.ensureIndex({number:1})

其中 1 正序创建索引 -1 逆序创建索引

1. 创建时 同时指定索引名称

为name键 创建倒序索引 ，同时指定索引名称 bookname

db.books.ensureIndex({name:-1},{name:”bookname”})

1. 唯一索引

值不能重复

db.books.ensureIndex({name:1},{unique:true})

1. 剔除重复的文档

集合中的该键的值已经重复，不能在该键上建立索引。可以使用dropDups剔除重复的数据，确保索引能够建立。

db.books.ensureIndex({name:1},{unique:true,dropDups:true})

1. 查询数据时 强制指定照哪个索引查找

按照 name 倒序 查找

db.books.find({name:”1book”}).hint({name:-1})

注意 该索引必须存在

1. 查看查询数据使用哪个索引

db.books.find({number:12122}).explain()

结果中：cursor----使用索引

nscanned----查到的文档数

millis----执行时间（毫秒）

1. 查看数据库中已有的索引

db.system.indexes.find()

db.system.namespaces.find()

1. 后台执行

索引创建的时候会暂时锁表，为了查询不受影响，应该后台执行

db.books.ensureIndex({name:1},{background:true})

1. 删除索引

分为精确删除和批量（模糊）删除

删除集合books 中名为number\_1的索引

db.runCommand({dropIndexes:”books”,index:”number\_1”})

删除集合books 中所有的索引（\_id\_不会删除）

db.runCommand({dropIndexes:”books”,index:”\*”})

# 2D索引

map集合 数据如下

[

{gis:{x:50,y:50}},

{gis:{x:80,y:50}},

{gis:{x:50,y:70}},

{gis:{x:100,y:50}},

….

]



1. 创建2D 索引

db.map.ensureIndex({gis:”2d”},{min:-1,max:201})

min,max指定范围，默认[-180,180]

1. $near

查询离(70,180)最近的三个点

db.map.find({gis:{$near:[70,180]},{\_id:0,gis:1}).limit(3)

1. $within,$box,$center

查询[50,50],[190,190]为对角线的正方形内的所有点

db.map.find({gis:{$within:{$box:[[50,50],[190,190]]}}},{\_id:0,gis:1})

查询以(56,80)为中心点，50为半径的圆内的所有点

db.map.find({gis:{$within:{$box:[[56,80],50]}}},{\_id:0,gis:1})

# count、distinct、group、runCommand

1. count

统计persons中中国学生的数量

db.persons.find({country:”China”}).count()

1. distinct

查询persons中一共有多少个国家，分别是什么

db.runCommand({distinct:”persons”,key:”country”}).values

1. group

db.runCommand(

{

group:{

ns---------------------------集合名称

key-------------------------分组的key

initial--------------------初始化累加器

$reduce----------------组分解器

condition--------------条件

finalize-----------------组完成器

}

}

)

会先按照key分组，每组的每一条文档都要执行$reduce的方法。

$reduce接受两个参数：组内的本条记录和累加器数据。

查出persons中每个国家学生数学成绩最好的学生信息（必须90分以上）

db.runCommand({

group:{

ns:”persons”,

key:{country:true},

initial:{m:0},

$reduce:function(doc,prev){

if(doc.m>prev.m){

prev.m=doc.m;

prev.name=doc.name;

prev.country=doc.country;

}

},

condition:{m:{$gt:90}}

}

})

在以上的基础上，把这个人的信息链接起来写一个描述赋值给m

db.runCommand({

group:{

ns:”persons”,

key:{country:true},

initial:{m:0},

$reduce:function(doc,prev){

if(doc.m>prev.m){

prev.m=doc.m;

prev.name=doc.name;

prev.country=doc.country;

}

},

condition:{m:{$gt:90}},

finalize:function(prev){

prev.m=prev.name+” Math scores ”+prev.m;

}

}

})

如果同时存在country和counTry键

db.runCommand({

group:{

ns:”persons”,

$keyf:function(doc){

if(doc.counTry){

return {country:doc.counTry}

}else{

return {country:doc.country}

}

},

initial:{m:0},

$reduce:function(doc,prev){

if(doc.m>prev.m){

prev.m=doc.m;

prev.name=doc.name;

if(doc.country){

prev.country=doc.country;

}else{

prev.country=doc.counTry;

}

}

},

condition:{m:{$gt:90}}

}

})

1. runCommand

删除map集合

db.runCommand({drop:”map”})

查看提供的命令

db.listCommands()

也可以访问 <http://localhost:28017/_commands>

注意:可能访问不到该网址，服务启动时，记得加上 –rest 配置

查询服务器的版本号和主机的操作系统

db.runCommand({buildinfo:1})

查看集合persons的详细信息

db.runCommand({collstats:”persons”})

查看集合 persons操作最后一次错误的信息

db.runCommand({getLastError:”persons”})

# 固定集合

1. 创建

创建一个100字节，可以存储10个文档的固定集合mycoll

db.createCollection(“mycoll”,{size:100,max:10,capped:true})

1. 普通集合转为固定集合

将persons转为固定集合

db.runCommand({convertToCapped:”persons”,size:100000})

1. $natural

mycoll结果反向排序

db.mycoll.find().sort({$natural:-1})

# gridfs文件系统

利用mongodb自带的mongofiles以二进制的形式存储文件

直接在cmd 中执行mongofiles命令

1. 主要参数说明

-h 主机

-port 端口号

-d 数据库

-l 本地文件

-put 远程文件

1. 上传文件

将本地d盘下的aa.txt文件上传到mongodb中，文件名为a.txt

mongofiles –h 220.189.246.90 –port 27017 –d firstdb –l “d:\aa.txt” put “a.txt”

1. 信息查看

首次上传文件后，自动会在数据库中建立fs.chunk和fs.files两个集合

db.fs.files.find()

db.fs.chunks.find()

1. 下载文件

下载mongodb中的a.txt文件，保存到本地d盘的a.txt文件中

mongofiles –h 220.189.246.90 –port 27017 –d firstdb –l “d:\a.txt” get “a.txt”

1. 删除文件

mongofiles –h 220.189.246.90 –port 27017 –d firstdb delete “a.txt”

# system.js

1. 服务端运行eval

db.eval(“function(name){return name}”,”st”)

1. js存储

把变量存储到system.js中

db.system.js.insert({\_id:”name”,value:”st”})

可以全局调用

db.eval(“return name;”);

system.js类似oracle中的存储过程，value不仅仅可以存储变量，也可以存储函数体（javascript代码）

# 集合数据导出

在cmd下运行mongoexport导出数据（会终止其他操作）

1. 参数说明

-h 主机

-port 端口

-d 数据库

-c 集合

-o 导出的文件名

1. 示例

导出firstdb中persons集合，保存到本地d盘下的persons.json文件中

mongoexport –h 220.189.246.90 –port –d 27017 firstdb –c persons –o “d:\persons.json”

# 数据导入集合

在cmd下运行mongoimport导入数据（会终止其他操作）

1. 参数说明

-h 主机

-port 端口

-d 数据库

-c 集合

-file 待导入文件

1. 示例

将本地d盘下的persons.json文件导入到远程的firstdb的persons集合中

mongoimport –h 220.189.246.90 –port –d 27017 firstdb –c persons –file “d:\persons.json”

# 运行时导出数据库

由于mongoexport执行会中断数据库其它操作，可以使用mongodump运行时导出

1. 参数说明

-h 主机

-port 端口

-d 数据库

-c 集合

-o 导出的文件名或文件夹

1. 示例

将firstdb数据库导出到本地d盘的firstdb文件夹中

mongodump –h 220.189.246.90 –port –d 27017 firstdb –o “d:”

# 运行时导入数据库

由于mongoimport执行会中断数据库其它操作，可以使用mongorestore运行时导入

1. 参数说明

-h 主机

-port 端口

-d 数据库

-c 集合

-directoryperdb 导入的文件夹

1. 示例

将本地d盘的firstdb文件夹中的数据导入到firstdb数据库中

mongorestore –h 220.189.246.90 –port –d 27017 firstdb –directoryperdb “d:\firstdb”

# 锁和数据修复

Fsync锁，上锁可以让缓冲池里的数据进入数据库，对数据备份有很多大意义。

1. 上锁

db.runCommand({fsync:1,lock:1})

1. 解锁

db.currentOp();

1. 数据修复

停电等因数会造成垃圾数据，这时可以使用以下命令进行修复

db.repairDatabase()

# 安全认证

1. 添加用户

admin数据库增加用户st,密码123;firstdb数据库增加用户test,密码123

use admin

db.addUser(“st”,”123”)

use firstdb

db.addUser(“test”,”123”)

添加的用户，在system.users集合中

1. 启用用户

db.auth(“st”,”123”)

1. 安全检查

启动mongo服务时加上-auth

1. 删除用户

db.system.users.remove({user:”st”})

# 主从复制

1. 概念



主从复制

主从复制是一种简单的数据库同步备份的集群技术。

在数据库集群要明确主服务器是谁，主服务器只有一台。

从服务器要知道主服务器是谁。

1. 案例

三台服务器 端口27017:主服务器；端口37017：从服务器1（直接指定）；端口47017：从服务器2（动态指定）。

主服务器

mongod -dbpath D:\Soft\mongodata\67017\db -logpath D:\Soft\mongodata\67017\log\MongoDB.log -port 27017 –master

从服务器1

mongod -dbpath D:\Soft\mongodata\77017\db -logpath D:\Soft\mongodata\77017\log\MongoDB.log -port 37017 -slave -source 127.0.0.1:27017

从服务器2

mongod -dbpath D:\Soft\mongodata\87017\db -logpath D:\Soft\mongodata\87017\log\MongoDB.log -port 47017 –slave

查看数据源（登录从服务器1，在local下）

use local

db.sources.find()

登录主服务器，插入数据

use test

for(var i=0;i<1000;i++){

db.person.insert({name:"a"+i,age:i});

}

动态配置从服务器2（登录从服务器2,在local下）

use local

db.sources.insert({host:"127.0.0.1:27017"})

# 副本集

1. 概念



副本集

该集群没有特定的主服务器。

如果主服务器故障了，集群中会推选出一个从服务器做为主服务器。

1. 案例

配置

mongod -dbpath D:\Soft\mongodata\A\db -logpath D:\Soft\mongodata\A\log\MongoDB.log -port 27017 -replSet=child/127.0.0.1:37017

mongod -dbpath D:\Soft\mongodata\B\db -logpath D:\Soft\mongodata\B\log\MongoDB.log -port 37017 -replSet=child/127.0.0.1:47017

mongod -dbpath D:\Soft\mongodata\C\db -logpath D:\Soft\mongodata\C\log\MongoDB.log -port 47017 -replSet=child/127.0.0.1:27017

# 分片

1. 概念



分片

1. 步骤

* 创建一个配置服务器（端口37017）

mongod -dbpath D:\Soft\mongodata\37017\db -logpath D:\Soft\mongodata\37017\log\MongoDB.log -port 37017

* 创建路由服务器（端口27017），并且连接配置服务器，路由调用使用mongos

mongos -port 27017 -configdb=127.0.0.1:37017(实际环境中要指定ip，否则会报错)

* 添加分片(暂定2个，端口分别为47017和57017)

mongod -dbpath D:\Soft\mongodata\47017\db -logpath D:\Soft\mongodata\47017\log\MongoDB.log -port 47017

mongod -dbpath D:\Soft\mongodata\57017\db -logpath D:\Soft\mongodata\57017\log\MongoDB.log -port 57017

* 利用路由为集群添加分片

登录路由：

mongo 127.0.0.1:27017/admin

添加分片（允许本地访问）：

db.runCommand({addshard:"127.0.0.1:47017",allowLocal:true})

db.runCommand({addshard:"127.0.0.1:57017",allowLocal:true})

* 打开数据分片功能

db.runCommand({enablesharding:"test"})

* 对集合进行分片

db.runCommand({shardcollection:"test.person",key:{name:1}})

* 插入大量数据测试

use test

for(var i=0;i<100000;i++){

db.person.insert({name:"jack"+i,age:i})

}

1. 查看分片服务器的配置存储

db.printShardingStatus()