**mongodb副本集**

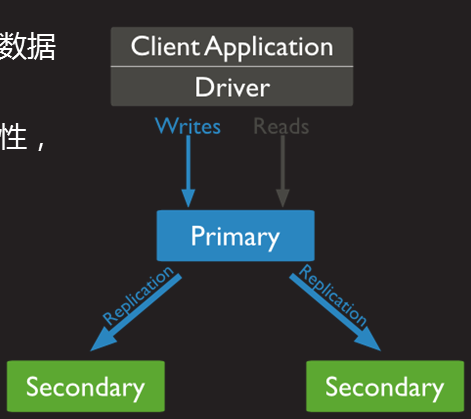
环境准备

副本集介绍

也称为mongodb复制

指在多个服务器上存储数据副本，并实现数据同步

提高数据可用性、安全性，方便数据故障恢复



mongodb复制原理

副本集工作过程

至少需要两个节点。其中一个是主节点，负责处理客户端的请求，其余是从节点，负责复制主节点数据

常见搭配方式：一主一从、一主多从

主节点记录所有操作oplog，从节点定期轮询主节点获取这些操作，然后对自己的数据副本执行这些操作，从而保证从节点的数据与主节点一致

副本集实现方式

master-slave主从复制

启动一台服务器时加上”-master”参数，作为主节点

启动其他服务器时加上”-slave”和”-source”参数，作为从节点

主从复制优点

从节点可以提供数据查询，降低主节点的访问压力

由从节点执行备份，避免锁定主节点数据

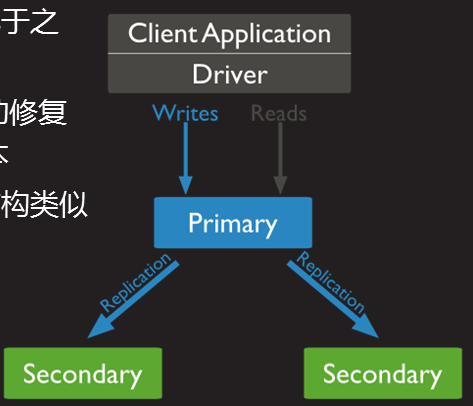
当主节点故障时，可快速切换到从节点，实现高可用

replica sets副本集

从1.6版本开始支持，优于之前的replication

支持故障自动切换、自动修复成员节点，将宕机期间的数据同步过来，降低运维成本

replica sets副本集的结构类似高可用集群



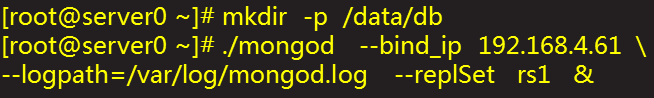
配置replica sets

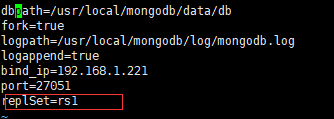
运行mongodb服务

启动服务时，指定主机所在副本集名称

所有副本集成员使用相同的副本集名称

--replSet rs1 //指定副本集名称



所有主机都加

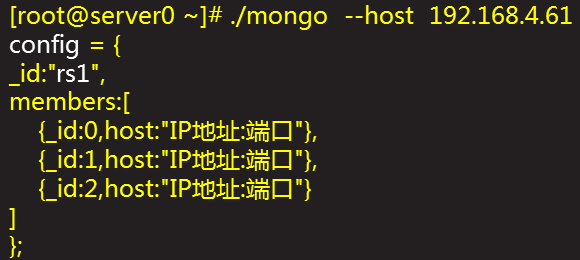


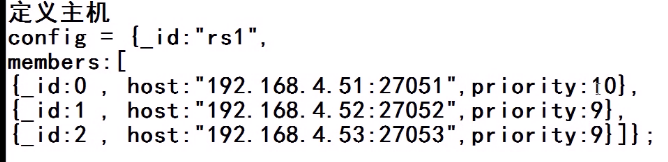




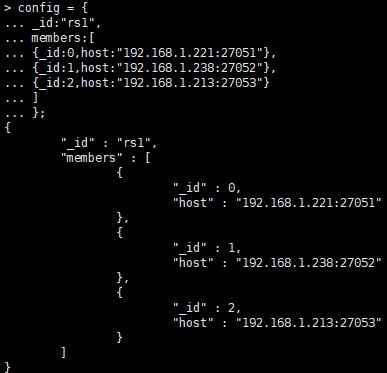
配置节点信息（在谁上创建，谁就是主，优先获取得到IP的就是主）

在任意一台主机连接mongod服务，执行如下操作



值越大,谁就是主



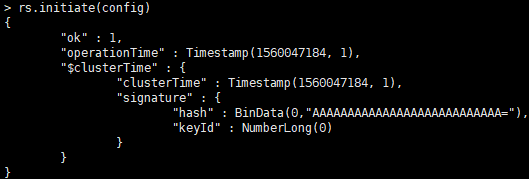


初始化replica sets环境

执行如下命令

>rs.initiate(config)

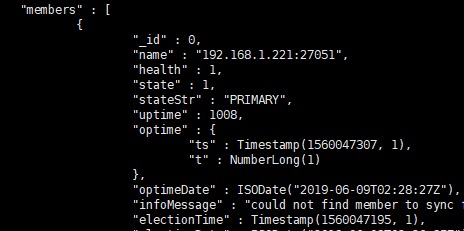




查看副本集信息

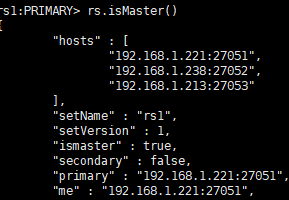
查看状态信息

rs.status()



查看是否是master库

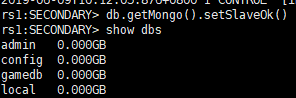
rs.isMaster()



验证副本集配置

同步数据验证，允许从库查看数据，默认从库不允许进行操作

db.getMongo().setSlaveOk()

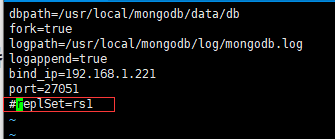


自动切换主库验证

rs.isMaster()

恢复独立数据库服务器

配置文件爱你注释并重新启动服务



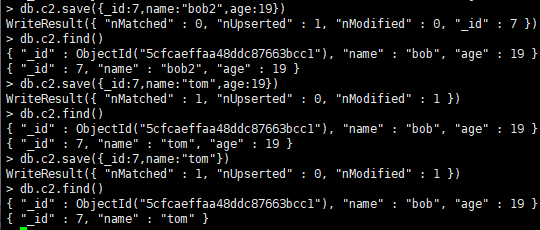
Mongodb文档管理

插入文档

save()

格式

db.集合名.save({key:”值”,key:”值”})



注意

集合不存在时创建集合，然后再插入记录

\_id字段已存在时，修改文档字段值

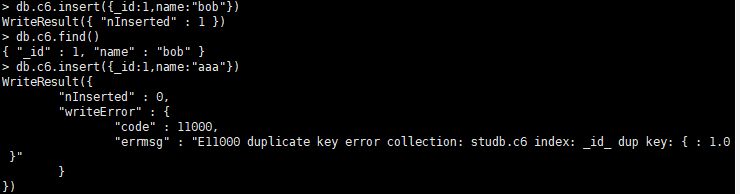
\_id字段值不存在时，插入文档

一次只能写一条文档

insert()

格式

db.集合.insert({key:”值”,key:”值”})





注意

集合不存在时创建集合，然后再插入记录

\_id字段值已存在时，放弃插入

\_id字段值不存在时，插入文档

一次只能写一条文档

插入多条记录

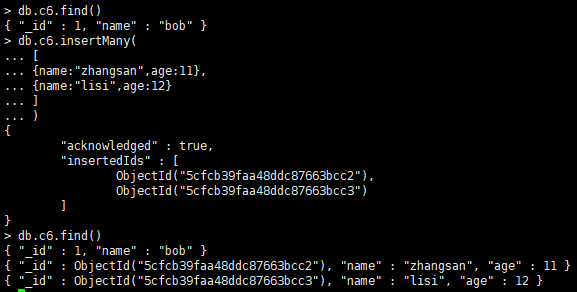
db.集合名.insertMany([

{name:”xiaojiu”,age:19},

{name::”laoshi”,email:”yaya@tedu.cn”}

]

)

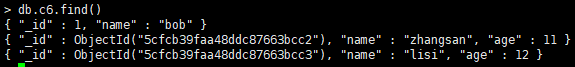


查询文档

查询语法

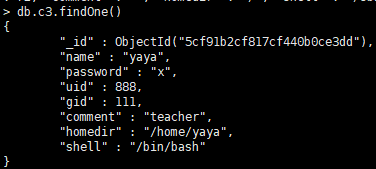
显示所有行（默认输出20行，输入it可显示后续行）

>db.集合名.find()



显示第一行

>db.集合名.findOne()



指定查询条件并指定显示的字段

>db.集合名.find({条件},{定义显示的字段})



>db.user.find({\_id:0,name:1,shell:1})//0不显示，1显示

仅显示name字段

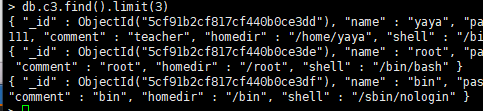


除了id不显示其他都显示

行数显示限制

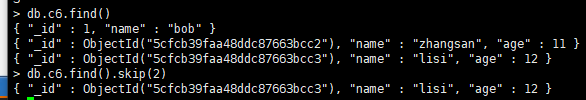
limit(数字) //显示前几行

>db.集合名.find().limit(3)



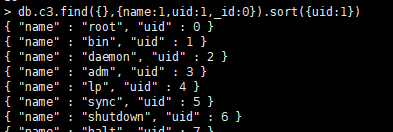
skip(数字)//跳过前几行

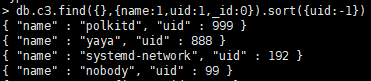
>db.集合名.find().skip(2)



sort(字段名)//1升序，-1降序

>db.集合名.find().sort(age:1|-1)





匹配条件

简单条件

>db.集合名.find({key:”值”})



>db.集合名.find({lkey:”值”,keyname:”值”})

两个条件同时成立时显示

>db.user.find({shell:”/bin/bash”})

>db.user.find({sheell:”/bin/bash”,name:”root”})

范围比较

$in 在...里

$nin 不在...里

$or 或

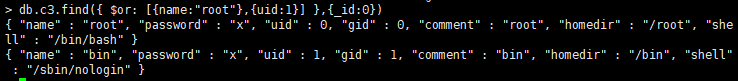
>db.user.find({uid:{$in:[1,6,9]}})



>db.user.find({uid:{$nin:[1,6,9]}})



>db.user.find({$or:[{name:”root”},{uid:1}]})



数值比较

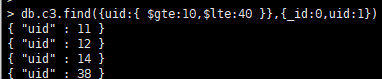
$It $Ite $gt $gte $ne

< <= > > !=

db.user.find({uid:{$gte:10,$lte40},{\_id:0,name:1,uid:1}})

db.user.find({uid:{$lte:5,}})

等于

uid大于10且小于40的

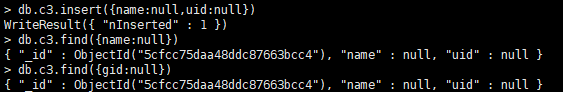
正则匹配

db.user.find({name:/^a/})



匹配空，也可以匹配没有的字段

db.user.save({name:null,uid:uid})

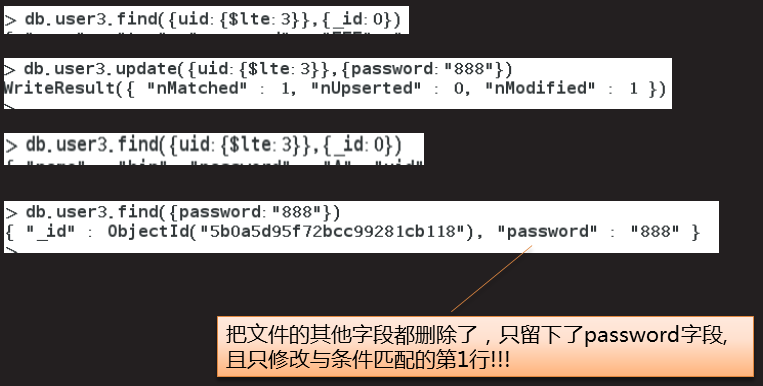


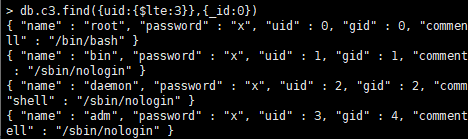
更新文档（修改列的值）

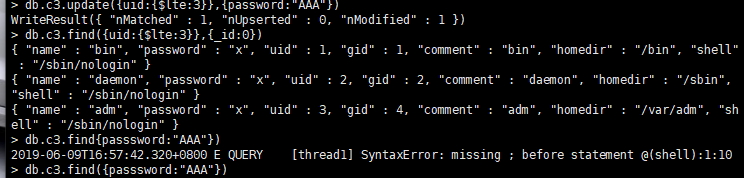
update()

语法格式

>db.集合名.update({条件},{修改的字段})





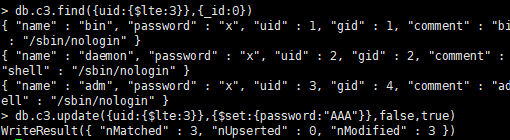


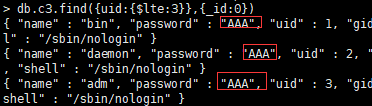
默认的时候，当条件匹配，会去更新第一个值，并且会删除其他列

多文档更新

语法格式：默认是更新与条件匹配的第1行

>db.user.update({条件},{$set:{修改的字段},false,true})





$set/$unset

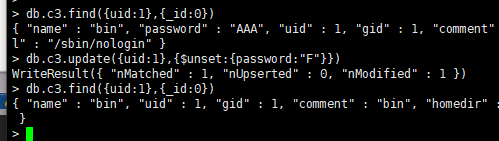
$set 条件匹配时，修改制定字段的值

>db.user.update({条件,$set:{修改的字段}})



$unset删除与条件匹配文档的字段，删除文档指定的列

>db.集合名.update({条件},unset:{key:values})

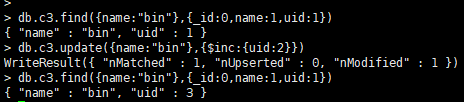


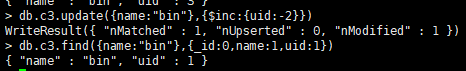
$inc

$inc条件匹配时，字段值自加或自减

>db.集合名.update({条件}，{$inc:{字段名:数字}})

+num自增，-num自减

自增

自减

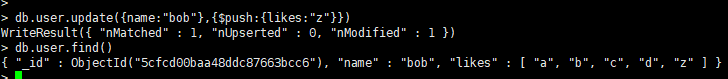
$push/$addToSet

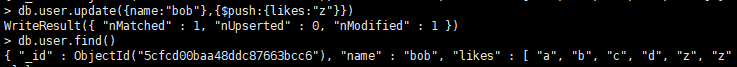




$push向数组中添加新元素

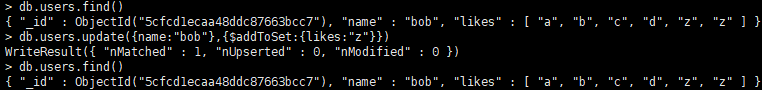
>db.集合名.update({条件},{$push:{数组名:”值”}})





$addToSet避免重复添加

>db.集合名.update({条件},{$addToSet:{数组名:”值”}})

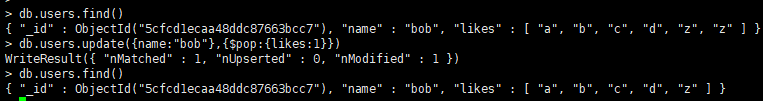


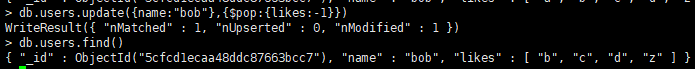
$pop/$pull

$pop从数组头部删除一个元素

>db.集合名.update({条件},{$pop:{数组名:数字}})

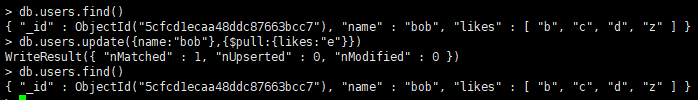
1删除数组尾部元素，-1删除数组头部元素

从后删除

从前删除

$pull删除数组指定元素

>db.集合名.update({条件},{$pull:{数组名:值}})

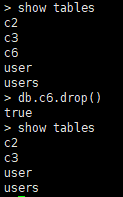


删除集合

$drop/$remove

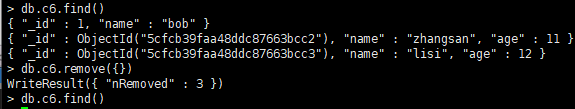
$drop 删除集合的同时删除索引

>db.集合名.drop()



$remove()删除文档时不删除索引

>db.集合名.remove({})//删除所有文档



>db.集合名.remove({条件})//删除与条件匹配的文档

