一　mongodb副本集

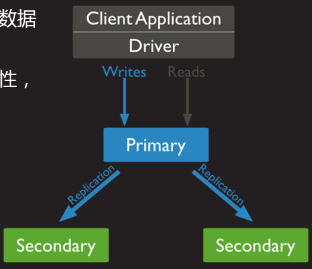
１，前言

--副本集介绍

副本集也称为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



２，配置

--配置Replica Sets

１）指定副本集名称

[root@mysql-52 ~]# vim /etc/mongodb/etc/mongodb.conf

logpath=/etc/mongodb/log/mongodb.log

logappend=true

port=27052

bind\_ip=192.168.4.52

dbpath=/etc/mongodb/data/db

fork=true

replSet=rs1

２）配置节点信息

[root@mysql-51 ~]# /etc/mongodb/bin/mongo --host 192.168.4.51 --port 27051

> config = {

... \_id:"rs1",

... members:[

... {\_id:0,host:"192.168.4.51:27051"},　　　//默认第一行主机作为主

... {\_id:1,host:"192.168.4.52:27052"},

... {\_id:2,host:"192.168.4.53:27053"}

... ]

... };

{\_id:0,host:"192.168.4.51:27051",priority:10} //数值越小，优先级越高

{\_id:1,host:"192.168.4.52:27052",priority:11},

３）初始化replica sets 环境

> rs.initiate(config)

{

"ok" : 1,　　　　#1成功

"operationTime" : Timestamp(1543979280, 1),

"$clusterTime" : {

"clusterTime" : Timestamp(1543979280, 1),

"signature" : {

"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),

"keyId" : NumberLong(0)

}

}

}

rs1:PRIMARY> rs.status()　　#查看状态信息

rs1:PRIMARY> rs.isMaster()　　#查看是否是master库，查看master信息

４）验证

##主库存数据

rs1:PRIMARY> use db5

switched to db db5

rs1:PRIMARY> db.t1.save({name:"jluo",pwd:"123456"})

WriteResult({ "nInserted" : 1 })

##从库验证

rs1:SECONDARY> db.getMongo().setSlaveOk()　　#允许从库查看数据

rs1:SECONDARY> use db5

switched to db db5

rs1:SECONDARY> db.t1.find()

{ "\_id" : ObjectId("5c074aed8b70f6c163d5a78a"), "name" : "jluo", "pwd" : "123456" }

５）验证高可用

##主库51停掉

[root@mysql-51 ~]# /etc/mongodb/bin/mongod -f /etc/mongodb/etc/mongodb.conf --shutdown

##随便登录一台查看主库

[root@mysql-50 mbak]# /etc/mongodb/bin/mongo --host 192.168.4.52 --port 27052

rs1:SECONDARY> rs.isMaster()

{

"ismaster" : false,

"secondary" : true,

"primary" : "192.168.4.53:27053",

"me" : "192.168.4.52:27052",

##登录主库，写入数据

[root@mysql-50 mbak]# /etc/mongodb/bin/mongo --host 192.168.4.53 --port 27053

rs1:PRIMARY> use db5

rs1:PRIMARY> db.t1.save({name:"test1",pwd:"123456"})

##开启51主机

[root@mysql-51 ~]# /etc/mongodb/bin/mongod -f /etc/mongodb/etc/mongodb.conf

rs1:SECONDARY> db.getMongo().setSlaveOk()

rs1:SECONDARY> use db5

rs1:SECONDARY> db.t1.find()

{ "\_id" : ObjectId("5c074aed8b70f6c163d5a78a"), "name" : "jluo", "pwd" : "123456" }

{ "\_id" : ObjectId("5c076f97336ebed61baeb3b8"), "name" : "test1", "pwd" : "123456" }

rs1:SECONDARY> rs.isMaster()

{

"setName" : "rs1",

"setVersion" : 1,

"ismaster" : false,

"secondary" : true,　　　　　　#自动加入集群，成为从库

"primary" : "192.168.4.53:27053",

"me" : "192.168.4.51:27051",

二　mongodb文档管理

１，插入文档

--save　　　　　　#插入单条记录

格式：

db.集合名.save({key:”值”,key:”值”})

--insert　　　　　　#插入单条记录

格式：

db.集合名.insert({key:”值”,key:”值”})

--insertMany　　　#插入多条文档

格式：

db.集合名.insert([{key:”值”,key:”值”},{key:”值”,key:”值”}．．．])

提示：

集合不存在时创建集合，然后再插入记录

\_id字段值已经存在时,save修改文档字段的值，insert则放弃插入

\_id字段值不存在时，插入文档

例子：

> db.t1.save({name:"test1",mail:"jluo@jluo.com"})

> db.t1.insert({name:"test2",age:"18",class:"jluo2018"})

> db.t1.insertMany(

... [

... {name:"jack",age:"20",love:"game"},

... {name:"rose",age:"15",sex:"女"}

... ]

... )

２，查询文档

--find()

格式1：db.集合名.find()　　#显示所有行（默认输出20行，出入it可显示后续行）

格式2:db.集合名.find({条件}) #显示符合条件的文档

格式3:

db.集合名.find({条件},{字段1:0,字段2:1,....})#显示符合条件并指定显示的字段

--findOne()

格式：db.集合名.findOne() #显示第一行

例子：

> db.user.findOne()

--limit(数字)

格式：db.集合名.find().limit(数字)　　　#指定显示前几行

例子：

> db.user.find({shell:"/sbin/nologin"},{\_id:0,uid:1,name:1}).limit(3)

--skip(数字)

格式：db.集合名.find().skip(数字)　　#跳过前几行

例子：

> db.user.find({shell:"/sbin/nologin"},{\_id:0,uid:1,name:1}).skip(3)

--sort({字段名})

格式：db.集合名.find().sort({字段名:1|-1}) #1升序，-1降序

例子：

> db.user.find({shell:"/sbin/nologin"},{\_id:0,uid:1,name:1}).sort({uid:1})

--条件匹配

##范围比较

$in 在....里

$nin 不在...里

$or 或

例子：

> db.user.find({name:{$in:["root","bin"]}},{\_id:0})

> db.user.find({name:{$nin:["root","bin"]}},{\_id:0})

> db.user.find({$or:[{name:"root"},{name:"bin"}]},{\_id:0})

##正则匹配

格式：db.集合名.find({name:/正则/})

例子：

> db.user.find({name:/t$/},{\_id:0,name:1,uid:1})

> db.user.find({name:/^r.../},{\_id:0,name:1,uid:1})

##数值比较

$lt $lte $gt $gte $ne

例子：

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

##匹配null,也可以匹配没有的字段

３，更新文档

--$update

##默认修改第一行，修改第一行password字段，删除其他字段

> db.user.update({uid:{$lte:2}},{password:"f"})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

--$set

##多文档更新，修改指定字段的值，保留其他字段不变，不加false,true默认修改一行

> db.user.update({uid:{$lte:2}},{$set:{password:"f"}},false,true)

WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })

--$unset

##删除与条件匹配文档的字段

> db.user.update({password:"f"},{$unset:{password:"f"}},false,true)

--$inc

字段值自加或自减　　+num自增　　-num自减

> db.user.update({uid:{$lt:6}},{$inc:{uid:+1}},false,true)

> db.user.update({uid:{$lt:7}},{$inc:{uid:-1}},false,true)

提示：加上false,true会更新所有匹配的行，不加则默认更新第一行

--$push/$addToSet　==>对数组操作

#$push向数组中添加新元素，允许重复

> db.t2.save({name:"bob",likes:["a","b","c"]})

> db.t2.update({name:"bob"},{$push:{likes:"w"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.t2.update({name:"bob"},{$push:{likes:"w"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

#$addToSet向数组中添加新元素，避免重复添加

> db.t2.update({name:"bob"},{$addToSet:{likes:"s"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.t2.update({name:"bob"},{$addToSet:{likes:"s"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

--$pop/$pull ==>操作数组

#$pop从数组头部删除一个元素

格式：db.集合名.update({条件},{$pop:{数组名：数字}})　1 删除尾部，-1删除头部

例子：

> db.t2.update({name:"bob"},{$pop:{likes:1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.t2.update({name:"bob"},{$pop:{likes:-1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

#$pull删除数组指定元素

格式：db.集合名.update({条件},{$pop:{数组名:值}})

例子：

> db.t2.update({name:"bob"},{$pull:{likes:"b"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

４，删除文档

--$drop

#删除集合的同时删除索引

例子：

> db.t2.drop()

true

--$remove

#删除文档时不删除索引

例子：

> db.t1.remove({})

WriteResult({ "nRemoved" : 4 })