**nosql-day04**

# MONGODB

# 数据类型

# 数据导入导出

## 部署MONGODB

### MONGODB概述

1. MONGODB介绍

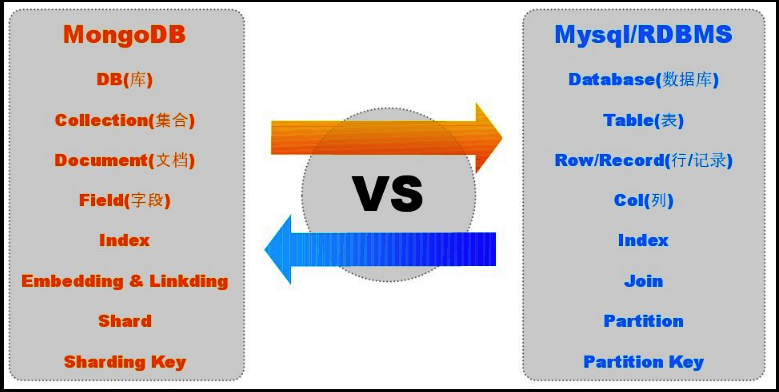
介于关系数据库和非关系数据库之间的产品

— 一款基于分布式文件存储的数据库，旨在为WEB应用提供可扩展的高性能数据存储解决方案

— 将数据存储为一个文档（类似于JSON对象），数据结构由键值（key=>value）对组成

— 支持丰富的查询表达，可以设置任何属性的索引

— 支持副本集，分片



### 搭建MDB服务器

1. **装包**

免安装，解压后即可使用

[root@mysql51 ~]# mkdir /usr/local/mongodb

[root@mysql51 ~]# tar -zxf mongodb-linux-x86\_64-rhel70-3.6.3.tgz

[root@mysql51 ~]# cd mongodb-linux-x86\_64-rhel70-3.6.3/

[root@mysql51 mongodb-linux-x86\_64-rhel70-3.6.3]# cp -r bin/

[root@mysql51 ~]# cd /usr/local/mongodb/

**2. 创建工作目录**

[root@mysql51 mongodb]# mkdir etc

[root@mysql51 mongodb]# mkdir log

[root@mysql51 mongodb]# mkdir -p data/db

**3. 手动创建服务主配置文件**

[root@mysql51 ~]# vim /usr/local/mongodb/etc/mongodb.conf

logpath=/usr/local/mongodb/log/mongodb.log

logappend=true //追加的方式记录日志信息

dbpath=/usr/local/mongodb/data/db //数据库目录

fork=true //守护进程方式运行

**4. 启动服务**

[root@mysql51 ~]# cd /usr/local/mongodb/

[root@mysql51 mongodb]# bin/mongod -f etc/mongodb.conf

**5. 查看服务信息**

[root@mysql51 mongodb]# ss -natpul | grep :27017



[root@mysql51 mongodb]# ps -C mongod



**6. 连接服务**

[root@mysql51 mongodb]# bin/mongo

MongoDB shell version v3.6.3

connecting to: mongodb://127.0.0.1:27017

.........

> show dbs //显示已有的库

admin 0.000GB

local 0.000GB

> exit //断开连接

1. **停止服务**

[root@mysql51 mongodb]# pwd

/usr/local/mongodb

# bin/mongod --shutdown -f etc/mongodb.conf

1. **指定服务使用的IP地址和端口号**

# vim /usr/local/mongodb/etc/mongodb.conf

.........

port=27051

bind\_ip=192.168.4.51

[root@mysql51 mongodb]# bin/mongod -f etc/mongodb.conf

[root@mysql51 mongodb]# ss -natpul | grep :27051



[root@mysql51 mongodb]# bin/mongo --host 192.168.4.51 --port 27051

## MongoDB基本使用

### 常用管理命令

**数据库管理**

1. **查看、创建、切换、删除库**

— show dbs //查看已有的库

— db //显示当前所在的库

— use //切换库，若库不存在延时创建库

— show collections或 show tables //查看库下已有的集合

— db.dropDatabase() //删除当前所在的库

1. **数据库名称规范**

— 不能是空字符串（“”）

— 不的含有’’（空格）、. 、$、/、\和\0（空字符）

— 应全部小写

— 最多64字节

**集合管理**

**1. 查看、创建、删除集合**

— show collections 或 show tables # 查看集合

— db.集合名.drop() # 删除集合

— db.集合名.save({“,”}) # 创建集合，集合不存在时，创建并添加文档

**2. 集合名命名规范**

— 不能是空字符串“”

— 不能含有\0字符串（空字符），次字符表示集合的结尾

— 不能以“system.”开头，这是为系统集合保留的前缀

— 用户创建的集合名字不能含有保留字符

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> db

test

> use gamedb

switched to db gamedb

> show tables

> db.t1.save({name:"bob",age:19})

WriteResult({ "nInserted" : 1 })

> show tables

t1

> db.t1.save({name:"tom",age:19,mail:"tom@163.com"})

WriteResult({ "nInserted" : 1 })

**文档基本管理**

文档：类似于MYSQL表里的记录

**查看、统计、添加、删除文档**

— db.集合名.find()

— db.集合名.count()

— db.集合名.insert({“name” :” jim”})

— db.集合名.find({条件})

— db.集合名.findOne() //返回一条文档

— db.集合名.remove({}) //删除所有文档

— db.集合名.remove({条件}) //删除匹配的所有文档

> db.t1.find()

{ "\_id" : ObjectId("5bdbc4f2b5d3829c13dac2ce"), "name" : "bob", "age" : 19 }

{ "\_id" : ObjectId("5bdbc51fb5d3829c13dac2cf"), "name" : "tom", "age" : 19, "mail" : "tom@163.com" }

> db.t1.find({name:"bob"})

{ "\_id" : ObjectId("5bdbc4f2b5d3829c13dac2ce"), "name" : "bob", "age" : 19 }

> db.t1.find({name:"tom"})

{ "\_id" : ObjectId("5bdbc51fb5d3829c13dac2cf"), "name" : "tom", "age" : 19, "mail" : "tom@163.com" }

> db.t1.save({name:"tom",class:"1807",addr:"nanjing"})

WriteResult({ "nInserted" : 1 })

> db.t1.insert({name:"bob",like:"a,b,c"})

WriteResult({ "nInserted" : 1 })

> db.t1.remove({name:"bob"})

WriteResult({ "nRemoved" : 2 })

> db.t1.remove({})

WriteResult({ "nRemoved" : 2 })

> db.t1.find()

> show tables

t1

> db.t1.drop()

true

> show tables

## 基本数据类型

### 字符string/布尔bool/空null

1. 字符串string

— UTF-8字符串都可以表示为字符串类型的数据

— {name:”张三”}或{school:”tarena”}

> use userdb

switched to db userdb

> show tables

> db.t2.save({name:"xiaosix",card:null})

WriteResult({ "nInserted" : 1 })

1. 布尔bool

— 布尔类型有两个值true和false,(x:true)

1. 空null

— 用于表示空值或者不存在的字段,{x:null}

### 数值/数组array

1. 数值

— shell默认使用64为浮点性数值。{x:3.14}或{x:3}

— NumberInt（4字节整数）{x:NumberInt(3)}

— NumberLong（8字节整数）{x:NumberLong(3)}

1. 数组array

— 数据列表或数据集可以表示为数组

— {x: [“a” , ”b” , ”c”]}

> db.t2.save({name:"xiao6",gz:3.14})

WriteResult({ "nInserted" : 1 })

> db.t2.save({name:"xiao8",gz:3})

WriteResult({ "nInserted" : 1 })

> db.t2.save({pay:NumberInt(3)})

WriteResult({ "nInserted" : 1 })

> db.t2.save({pay:NumberInt(3.999)})

WriteResult({ "nInserted" : 1 })

> db.t2.save({pay:NumberLong(3.999)})

2018-11-02T12:02:36.015+0800 E QUERY [thread1] Error: number passed to NumberLong must be representable as an int64\_t :

@(shell):1:17

> db.t2.save({pay:NumberLong(3888)})

WriteResult({ "nInserted" : 1 })

> db.t2.save({name:"lucy",like:["boy","eat","sleep","game"]})

WriteResult({ "nInserted" : 1 })

### 代码/日期/对象

1. 代码

— 查询和文档中可以包括任何JaveScript代码

— {x:function(){/\* 代码 \*/}}

> db.t2.save({la:"php",code:function(){/\*<?php echo "hello boy"; ?> \*/} })

WriteResult({ "nInserted" : 1 })

> db.t2.find({la:"php"})

{ "\_id" : ObjectId("5bdbe93ab5d3829c13dac2d9"), "la" : "php", "code" : { "code" : "function (){/\*<?php echo \"hello boy\"; ?> \*/}" } }

1. 日期

— 日期被存储为自新纪元以来经过的毫秒数，不含时区

— {x:new Date()}

> db.t2.save({name:"jerry",day:new Date() })

WriteResult({ "nInserted" : 1 })

> db.t2.find({name:"jerry"})

{ "\_id" : ObjectId("5bdbe9adb5d3829c13dac2da"), "name" : "jerry", "day" : ISODate("2018-11-02T06:07:41.019Z") }

1. 对象

— 对象id是一个12字节的字符串，是文档的唯一标识

— {x:ObjectId()}

> db.t2.save({name:"lucy",stuid:ObjectID()}) //ObjectID不能被定义

2018-11-02T14:10:37.706+0800 E QUERY [thread1] ReferenceError: ObjectID is not defined :

@(shell):1:25

> db.t2.find({name:"lucy"}) //默认会创建一个\_id字段，每一个都不同

{ "\_id" : ObjectId("5bdbccd4b5d3829c13dac2d8"), "name" : "lucy", "like" : [ "boy", "eat", "sleep", "game" ] }

> db.t2.save({name:"lucy",stuid:ObjectId()}) //严格区分大小写即可创建

WriteResult({ "nInserted" : 1 })

> db.t2.find({name:"lucy"}) //再次查看会有默认一条和插入的一条

{ "\_id" : ObjectId("5bdbccd4b5d3829c13dac2d8"), "name" : "lucy", "like" : [ "boy", "eat", "sleep", "game" ] }

{ "\_id" : ObjectId("5bdbeacdb5d3829c13dac2dc"), "name" : "lucy", "stuid" : ObjectId("5bdbeacdb5d3829c13dac2db") }

### 内嵌/正则表达式

1. 内嵌

— 文档可以嵌套其他文档，被嵌套的文档作为值来处理

— {tarena:{address:”Beijing”,tel:”888888”,person:”hansy”}}

> db.t2.save({ tarena:{addre:"bj",tel:"40018888",person:"hsy"} })

WriteResult({ "nInserted" : 1 })

> db.t2.find()



1. 正则表达式

— 查询时，使用正则表达式作为限定条件

— {x:/正则表达式/}

> db.t2.save({comment:"match null line",bds:/^$/})

WriteResult({ "nInserted" : 1 })

> db.t2.find()



## 数据导入导出

### 数据导出

**语法格式1**

#mongoexport [--host IP地址 --port 端口] -d 库名 -c 集合名 -f 字段1，字段2 --type=csv > 目录名/文件名.csv

[root@mysql51 ~]# mkdir /mdb

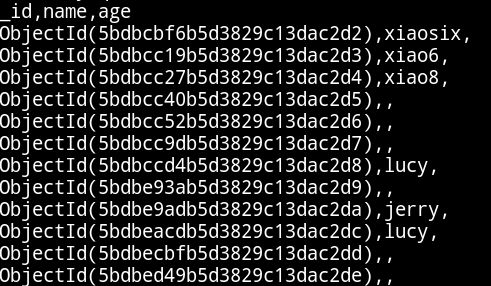
[root@mysql51 mdb]# /usr/local/mongodb/bin/mongoexport \

> --host 192.168.4.51 --port 27051 \

> -d userdb -c t2 -f \_id,name,age --type=csv > /mdb/userdb.t2.csv



[root@mysql51 ~]# cat /mdb/userdb.t2.csv



**语法格式2**

#mongoexport --host IP 地址 --port 端口 -库名 -c 集合名 -q ‘{条件}’ -f 字段名1，字段名2 --type=csv > 目录名/文件名.csv

注意：导出为csv格式必须使用-f指定字段名列表！！！

[root@mysql51 ~]# /usr/local/mongodb/bin/mongoexport \

> --host 192.168.4.51 --port 27051 \

> -d userdb -c t2 -q '{name:"xiaosix"}' -f \_id,name,age \

> --type=csv > /mdb/t2a.csv

[root@mysql51 ~]# cat /mdb/t2a.csv

\_id,name,age

ObjectId(5bdbcbf6b5d3829c13dac2d2),xiaosix,

**语法格式3**

#mongoexport [--host IP地址 --port 端口] -d 库名 -c 集合名 [-q ‘条件’ -f 字段列表] --type=json > 目录名/文件名.json

[root@mysql51 ~]# /usr/local/mongodb/bin/mongoexport \

> --host 192.168.4.51 --port 27051 \

> -d userdb -c t2 --type=json > /mdb/t2.json



[root@mysql51 ~]# cat /mdb/t2.json



### 数据导入

**语法格式1**

# mongoimport --host IP地址 -port 端口 -d 库名 -c 集合名

--type=json 目录名/文件名.json

**语法格式2**

# mongoimport --host IP地址 --port 端口 -d 库名 -c 集合名

--type=csv [--headerline] [--drop] 目录名/文件名.csv

[--headerline]

1. **导入数据时，若库和集合不存在，则先创建库和集合后再导入数据；**
2. **若库和集合以存在，则以追加的方式导入到集合里**
3. **使用--drop选项可以删除原数据后导入新数据，--headerline 忽略标题**

[root@mysql51 ~]# mongoexport --host 192.168.4.51 --port 27501 \

> -d userdb -c t2 -f name,age,class \

> --type=csv > /mdb/userdb-t2.csv

[root@mysql51 ~]# mongoimport --host 192.168.4.51 --port 27051 \

> -d db2 -c stutab --type=csv /mdb/userdb-t2.csv

[root@mysql51 ~]# mongoimport --host 192.168.4.51 --port 27051 \

> -d db2 -c stutab2 --headerline \

> --type=csv /mdb/userdb-t2.csv

> use db2

switched to db db2

> db.stutab2.find()

> db.stutab2.count()

[root@mysql51 ~]# mongoimport --host 192.168.4.51 --port 27051 \> -d db2 -c stutab3 \

> --headerline --drop --type=csv /mdb/userdb-t2.csv

> db.stutab3.find()

> db.stutab3.count()

[root@mysql51 ~]# mongoimport --host 192.168.4.51 --port 27051 -d db2 -c studb2 --type=json /mdb/t2.json

> use db2

> show tables

> db.studb2.find()

> db.studb2.count()

**将用户信息文件导入mongodb数据库中**

[root@mysql51 ~]# cp /etc/passwd /mdb/

[root@mysql51 ~]# sed -i 's/:/,/g' /mdb/passwd

[root@mysql51 ~]# sed -i '1i name,password,uid,gid,comment,homedir,shell' /mdb/passwd

[root@mysql51 ~]# mongoimport --host 192.168.4.51 --port 27051 \

> -d db4 -c user --headerline --type=csv /mdb/passwd

> use db4

> show tables

> db.user.find()

> db.user.find()-> it //显示下一页

> db.user.count()

> db.user.find({name:"root"})

[root@mysql51 ~]# mongoexport --host 192.168.4.51 --port 27051 -d db4 -c user -f name,password,uid,gid,comment,homedir,shell --type=csv > /mdb/user.csv

[root@mysql51 ~]# cat /mdb/user.csv

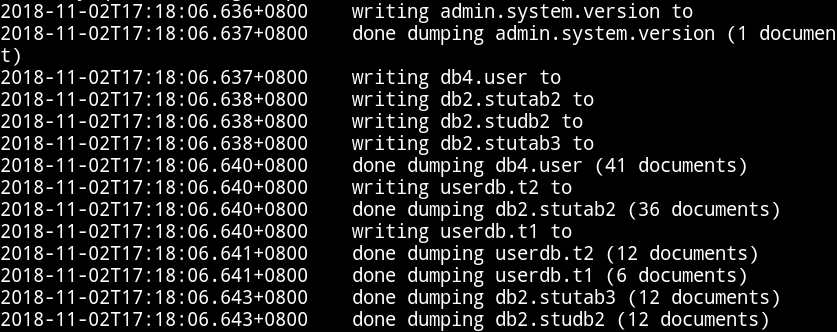
## 数据备份与恢复

### 备份数据

1. **备份数据所有库到当前目录下的dump目录下**

# mongodump [--host ip地址 --port 端口]

[root@mysql51 ~]# mongodump --host 192.168.4.51 --port 27051



1. **备份时指定备份的库和备份目录**

# mongodump [ --host ip地址 --port 端口] -d 数据奎 -c 集合名 -0 目录

[root@mysql51 ~]# mongodump --host 192.168.4.51 --port 27051 -o /mdb/ //备份所有库到 /mdb目录

[root@mysql51 mdb]# mongodump --host 192.168.4.51 --port 27051 -d db4 -c user -o /mdb //备份db4库的user表到/mdb目录下



1. **查看bson文件内容**

# bsondump ./dump/bbs/t1.bson

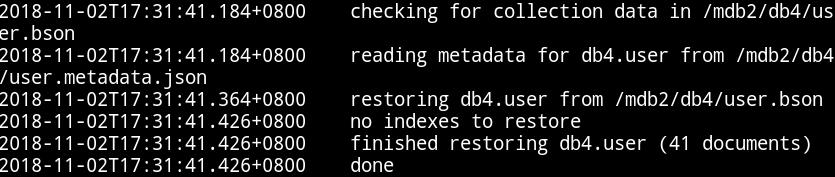
[root@mysql51 ~]# bsondump /mdb/admin/system.version.bson



### 数据恢复

# mongorestore --host ip地址 --port 端口 -d 数据库名 [ -c 集合名] 备份目录名

[root@mysql51 mdb]# mongorestore --host 192.168.4.51 --port 27051 -d db4 -c user /mdb/db4/user.bson



> show tables

> db.user.count()

> db.user.find()