

NSD NOSQL DAY04

1. [案例1：搭建MongoDB服务器](#)
2. [案例2：MongoDB常用管理操作](#)
3. [案例3：数据导入导出/备份/恢复](#)

1 案例1：搭建MongoDB服务器

1.1 问题

- 满足以下要求：
- 在主机192.168.4.51上部署MongoDB服务

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：搭建MongoDB服务器

1) 在主机192.168.4.51上部署MongoDB服务

MongoDB：

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

```
01. [root@mongodb51 ~]# cd mongodb/
02. [root@mongodb51 mongodb]# ls
03. mongodb-linux-x86_64-rhel70-3.6.3.tgz
04. [root@mongodb51 mongodb]# tar -xvf mongodb-linux-x86_64-rhel70-3.6.3.tgz
05. [root@mongodb51 mongodb]# mkdir /usr/local/mongodb
06. [root@mongodb51 mongodb]# cd /usr/local/mongodb/
07. [root@mongodb51 mongodb]# cp -r \
08. /root/mongodb/mongodb-linux-x86_64-rhel70-3.6.3/bin/ .
09. [root@mongodb51 mongodb]# ls
10. bin
11. [root@mongodb51 mongodb]# mkdir etc //创建存放配置文件的目录
12. [root@mongodb51 mongodb]# mkdir log //创建存放日志文件的目录
13. [root@mongodb51 mongodb]# mkdir -p data/db //创建存放数据库的目录db，必须为data
14. [root@mongodb51 mongodb]# vim etc/mongodb.conf
15. dbpath=/usr/local/mongodb/data/db/ //指定数据库目录
16. logpath=/usr/local/mongodb/log/mongodb.log //指定日志文件
```

[Top](#)

17. `logappend=true` //以追加的方式记录日志信息
18. `fork=true` //服务以守护进程的方式运行

2) 设置PATH变量

01. `[root@mongodb51 mongodb] # vim /etc/profile`
02. `export PATH=/usr/local/mongodb/bin: $PATH`
03. `[root@mongodb51 mongodb] # source /etc/profile`

3) 启动服务

01. `[root@mongodb51 mongodb] # mongod -f /usr/local/mongodb/etc/mongodb.conf`
02. about to fork child process, waiting until server is ready for connections.
03. forked process: 28001
04. child process started successfully, parent exiting //启动成功

4) 验证配置, 默认端口为27017

01. `[root@mongodb51 mongodb] # ls /usr/local/mongodb/data/db/`
02. `[root@mongodb51 mongodb] # ps -C mongod`
03. PID TTY TIME CMD
04. 28001 ? 00:00:01 mongod
05. `[root@mongodb51 mongodb] # netstat -antup | grep mongod`
06. tcp 0 0 127.0.0.1:27017 0.0.0.0:* LISTEN 28001/mongod

5) 连接服务

01. `[root@mongodb51 mongodb] # mongo` //默认本地连接, 没有密码
02. MongoDB shell version v3.6.3
03. connecting to: mongodb://127.0.0.1:27017
04. MongoDB server version: 3.6.3
05. ...
06. ...
07. `> show dbs` //显示已有的库
08. admin 0.000GB

[Top](#)

```

09.  config 0.000GB
10.  local 0.000GB
11.  > db      //查看当前所在的库
12.  test
13.  > exit
14.  bye

```

6) 停止服务

```

01.  [root@mongodb51 mongodb] # mongod -- dbpath=/usr/local/mongodb/data/db/ -- shut
02.  killing process with pid: 28001

```

7) 由于启动和停止服务名字太长，可以起一个别名 给停止服务起一个别名

```

01.  [root@mongodb51 mongodb] # alias cmdb='mongod -- dbpath=/usr/local/mongodb/data/

```

给启动服务起一个别名

```

01.  [root@mongodb51 mongodb] # alias smdb='mongod -f /usr/local/mongodb/etc/mongodb

```

8) 修改配置文件，使用ip和端口连接服务

```

01.  [root@mongodb51 mongodb] # vim /usr/local/mongodb/etc/mongodb.conf
02.  bind_ip=192.168.4.51 //在原先的基础上加上这两个，指定ip
03.  port=27077          //指定端口号
04.
05.  [root@mongodb51 mongodb] # smdb      //启动服务，之前设置过别名
06.  about to fork child process, waiting until server is ready for connections.
07.  forked process: 28240
08.  child process started successfully, parent exiting
09.  [root@mongodb51 mongodb] # ps -C mongod
10.  PID TTY          TIME CMD
11.  28240 ?        00:00:01 mongod

```

[Top](#)

```
12. [root@mongodb51 mongodb] # netstat - antup | grep mongod
13. tcp      0      0 192.168.4.51:27077    0.0.0.0:*        LISTEN      28240/mongod
```

9) 连接服务

```
01. [root@mongodb51 mongodb] # mongo -- host 192.168.4.51 -- port 27077 //成功
```

2 案例2：MongoDB常用管理操作

2.1 问题

- 要求如下：
- 练习库的创建、查看、切换、删除
- 练习集合的创建、查看、删除
- 练习文档的查看、插入、删除

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：MongoDB常用管理操作

1) 数据库管理

命令格式：

show dbs 查看已有的库

db 显示当前所在的库

use 库名 切换库，若库不存在延时创建库

show collections 或 show tables 查看库下已有集合

db.dropDatabase() 删除当前所在的库

查看

```
01. > show dbs //查看已有的库
02. admin 0.000GB
03. config 0.000GB
04. local 0.000GB
```

创建，切换

```
01. > db //显示当前所在的库
02. test
03. > use ddsdb //切换库，若库不存在的话 会自动延时创建库
```

[Top](#)

04. switched to db ddsdb
05. > db
06. ddsdb

2) 集合管理

命令格式：

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

01. > db.t1.save({ name: "y ay a", age: 60, addr: "hebei", email: "y ay a@163.com" })
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.save({ name: "y ay a", age: 70 })
04. WriteResult({ "nInserted" : 1 })
05. > show tables
06. t1

查看集合里的所有内容

01. > db.t1.save({ name: "y ay a", age: 70 })
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.find()
04. { "_id" : ObjectId("5badf1b420cdd1574b851f12"), "name" : "y ay a", "age" : 60, "addr" : "
05. { "_id" : ObjectId("5badf21820cdd1574b851f13"), "name" : "y ay a", "age" : 70 }
06. >



查看第一行的内容

01. > db.t1.findOne()
02. {
03. "_id" : ObjectId("5badf1b420cdd1574b851f12"),
04. "name" : "y ay a",
05. "age" : 60,
06. "addr" : "hebei",
07. "email" : "y ay a@163.com"
08. }

[Top](#)

09. >

3) 文档管理

命令格式：

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() 返回查询一条文档

db.集合名.remove({}) 删除所有文档

db.集合名.remove(条件) 删除与条件匹配的所有文档

```
01. > db.t1.save({ name: "xm", age: 18, "addr": "hn", "email": "xm.qq.com", "like": "nicai" })
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.count()
04. 3
05. > db.t1.find({ name: "xm" })
06. { "_id" : ObjectId("5badf4bc20cdd1574b851f14"), "name" : "xm", "age" : 18, "addr" : "hn"
```

删除与条件匹配的所有文档

```
01. > db.t1.remove({ name: "yaya" })
02. WriteResult({ "nRemoved" : 2 })
```

删除所有文档

```
01. > db.t1.remove({})
02. WriteResult({ "nRemoved" : 1 })
```

4) 插入记录

```
01. > db.col.insert(
02. { title: 'MongoDB 教程',
03.   description: 'MongoDB 是一个 NoSQL 数据库',
04.   by: 'MongoDB中文网',
```

[Top](#)

```
05.     url: 'http://www.mongodb.org.cn',
06.     tags: [ 'mongodb', 'database', 'NoSQL' ],
07.     likes: 100
08.   }
09. )
10. WriteResult({ "nInserted" : 1 })
```

删除记录

```
01. > db.col.remove({ 'title': 'MongoDB 教程' })
02. WriteResult({ "nRemoved" : 1 })
```

步骤二：基本数据类型

1) null：用于表示空值或者不存在的字段，{ "x" : null }

```
01. > db.t1.save({ name: "bob", work: null })
02. WriteResult({ "nInserted" : 1 })
03.
04. > db.t1.find()
05. { "_id" : ObjectId("5badf71520cdd1574b851f16"), "name" : "bob", "work" : null }
06. > db.t1.save({ _id: 9, name: "jerry", work: null })
07. // _id默认自己创建出来，按一定的规律生成
08.
09. WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 9 })
10. > db.t1.find()
11. { "_id" : ObjectId("5badf71520cdd1574b851f16"), "name" : "bob", "work" : null }
12. { "_id" : 9, "name" : "jerry", "work" : null }
```

2) 布尔值：布尔类型有两个值true和false，{ "x" : true }

```
01. > db.t1.save({ name: "zhangsan", addr: "shanghai", marry: "true" })
02. WriteResult({ "nInserted" : 1 })
03.
04. > db.t1.save({ name: "lisi", age: 35, addr: "beijing", marry: "false" })
05. WriteResult({ "nInserted" : 1 })
06.
07. > db.t1.find()
```

[Top](#)

```

08.  { "_id" : ObjectId( "5badf71520cdd1574b851f16" ), "name" : "bob", "work" : null }
09.  { "_id" : 9, "name" : "jerry", "work" : null }
10.  { "_id" : ObjectId( "5badf7b020cdd1574b851f17" ), "name" : "zhangsan", "addr" : "shanghai"
11.  { "_id" : ObjectId( "5badf7b720cdd1574b851f18" ), "name" : "lisi", "age" : 35, "addr" : "beijing"
12.  >

```

3) 数值: shell默认使用64为浮点型数值, { "x" : 3.14 }或{ "x" : 3 }, 对于整型值, 可以使用NumberInt (4字节符号整数) 或NumberLong (8字节符号整数), { "x" : NumberInt("3") } { "x" : NumberLong("3") }

```

01.  > db.t1.save( { name: "alice", woker: "gcs", pay: 28888.88 } )
02.  WriteResult( { "nInserted" : 1 } )
03.  > db.t1.find( { pay: 28888.88 } )
04.  { "_id" : ObjectId( "5badf80f20cdd1574b851f19" ), "name" : "alice", "woker" : "gcs", "pay" : 28888.88 }
05.
06.  > db.t1.save( { name: "lilei", woker: "cxy", pay: 28000 } )
07.  WriteResult( { "nInserted" : 1 } )
08.  > db.t1.find( { pay: 28000 } )
09.  { "_id" : ObjectId( "5badf81f20cdd1574b851f1a" ), "name" : "lilei", "woker" : "cxy", "pay" : 28000 }

```

4) 字符串: UTF-8字符串都可以表示为字符串类型的数据, { "x" : "呵呵" }

```

01.  > db.t1.save( { name: "hehe", woker: null, pay: null } )
02.  WriteResult( { "nInserted" : 1 } )
03.  > db.t1.save( { name: "呵呵", woker: "没有", pay: "没有" } )
04.  WriteResult( { "nInserted" : 1 } )

```

5) 日期: 日期被存储为自新纪元以来经过的毫秒数, 不存储时区, { "x" : new Date() }

```

01.  > db.t1.save( { name: "bobo", wokertime: new Date() } )
02.  WriteResult( { "nInserted" : 1 } )
03.  > db.t1.find( { name: "bobo" } )
04.  { "_id" : ObjectId( "5badf8ff6827555e3fd8680f" ), "name" : "bobo", "wokertime" : ISODate( "2018-11-21T08:55:58.000Z" ) }
05.  >

```

[Top](#)

6) 正则表达式：查询时，使用正则表达式作为限定条件，语法与JavaScript的正则表达式相同，{ "x" :/[abc]/ }

```

01. > db.t1.save({ procname: "php", code: /abc$/ })
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.save({ procname: "php", code: /[a-z]/ })
04. WriteResult({ "nInserted" : 1 })
05. > db.t1.save({ procname: "java", code: /[a-z]/ })
06. WriteResult({ "nInserted" : 1 })
07. > db.t1.find()
08. ...
09. ...
10. { "_id" : ObjectId( "5badf93b6827555e3fd86810"), "procname" : "php", "code" : /abc$/ }
11. { "_id" : ObjectId( "5badf9426827555e3fd86811"), "procname" : "php", "code" : /[a-z]/ }
12. { "_id" : ObjectId( "5badf9496827555e3fd86812"), "procname" : "java", "code" : /[a-z]/ }

```

7) 数组：数据列表或数据集可以表示为数组，{ "x" : ["a ", "b", "c"] }，一个字段有多个值

```

01. > db.t1.save({ name: "jerry", email: [ "plj@163.com", "lij@yahoo.net", "lij@tedu" ] })
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.find({ name: "jerry" })
04. { "_id" : 9, "name" : "jerry", "work" : null }
05. { "_id" : ObjectId( "5badf9976827555e3fd86813"), "name" : "jerry", "email" : [ "plj@163.c
06. >

```

8) 内嵌文档：文档可以嵌套其他文档，被嵌套的文档作为值来处理，{ "x" : { "y" : 3 } }

```

01. > db.t1.save({ book: { zuozhe: "dmy", bname: "yuweizhidao", jiaqian: 69, version: 2.0 } })
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.find()
04. ...
05. ...
06. { "_id" : ObjectId( "5badfa4a6827555e3fd86814"), "book" : { "zuozhe" : "dmy", "bname"

```

[Top](#)

9) 对象id：对象id是一个12字节的字符串，是文档的唯一标识，{ "x" : ObjectId() }

```

01. > db.t1.save({stunum:ObjectId(),name:"yaya",class:"nsd"})
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.find({name:"yaya"})
04. { "_id" : ObjectId("5badfa966827555e3fd86816"), "stunum" : ObjectId("5badfa966827555e3fd86816"), "class" : "nsd", "name" : "yaya" }
05. //自己定义一个id字段，一般情况下都用内置的id字段，相当于mysql里的primary key

```

10) 二进制数据：二进制数据是一个任意字节的字符串。它不能直接在shell中使用。如果要将非utf-字符保存到数据库中，二进制数据是唯一的方式。

11) 代码：查询和文档中可以包括任何JavaScript代码，{ "x" :function(){/*...*/}}

```

01. > db.t1.save({lname:"html",codecript:function(){/*...*/}})
02. WriteResult({ "nInserted" : 1 })
03. > db.t1.save({lname:"html",codecript:function(){/*<html><h1>abc</h1></html>*/}})
04. WriteResult({ "nInserted" : 1 })
05. > db.t1.find()
06. ...
07. ...
08. { "_id" : ObjectId("5badfd626827555e3fd86817"), "lname" : "html", "codecript" : { "code" : "/*<html><h1>abc</h1></html>*/" } }
09. { "_id" : ObjectId("5badfd6a6827555e3fd86818"), "lname" : "html", "codecript" : { "code" : "/*<html><h1>abc</h1></html>*/" } }
10. >

```

3 案例3：数据导入导出/备份/恢复

3.1 问题

- 要求如下：
- 练习数据导入导出
- 练习数据备份恢复

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：数据备份与恢复

1) 数据备份

```

01. [root@mongodb51 ~]# mongodump -- host 192.168.4.51 -- port 27077
02. //不指定备份哪个库，默认备份所有，不指定目录，自动生成dump目录，备份的数据在
03. 2018-09-28T18:14:12.585+0800 writing admin.system.version to

```

[Top](#)

```

04. 2018-09-28T18:14:12.586+0800 done dumping admin.system.version ( 1 document)
05. 2018-09-28T18:14:12.586+0800 writing ddsdb.t1 to
06. 2018-09-28T18:14:12.586+0800 writing test.t1 to
07. 2018-09-28T18:14:12.586+0800 writing ddsdb.col to
08. 2018-09-28T18:14:12.587+0800 done dumping ddsdb.t1 ( 17 documents)
09. 2018-09-28T18:14:12.588+0800 done dumping test.t1 ( 1 document)
10. 2018-09-28T18:14:12.588+0800 done dumping ddsdb.col ( 0 documents)
11. [root@mongodb51 ~] # ls
12. dump
13. [root@mongodb51 ~] # bsondump dump/ddsdb/t1.bson //查看bson文件内容
14. { "_id": { "$oid": "5badf71520cdd1574b851f16" }, "name": "bob", "work": null }
15. { "_id": 9.0, "name": "jerry", "work": null }
16. ...
17. ...
18. { "_id": { "$oid": "5badfd626827555e3fd86817" }, "lname": "html", "codecrypt": { "$code": "fun
19. { "_id": { "$oid": "5badfd6a6827555e3fd86818" }, "lname": "html", "codecrypt": { "$code": "fun
20. 2018-09-28T18:15:45.948+0800 17 objects found

```

备份时指定备份的库和备份目录

```

01. [root@mongodb51 ~] # mongodump -- host 192.168.4.51 -- port 27077 -d ddsdb -o /root/ddsdb
02. 2018-09-28T18:23:30.389+0800 writing ddsdb.t1 to
03. 2018-09-28T18:23:30.389+0800 writing ddsdb.col to
04. 2018-09-28T18:23:30.391+0800 done dumping ddsdb.t1 ( 17 documents)
05. 2018-09-28T18:23:30.391+0800 done dumping ddsdb.col ( 0 documents)
06. //- d备哪个库，- o指定备份的目录，备份ddsdb库里的所有到/root/ddsdb

```

只备份ddsdb库里的集合t1

```

01. [root@mongodb51 ~] # mongodump -- host 192.168.4.51 -- port 27077 -d ddsdb -c t1 -o /root/ddsdb
02. 2018-09-28T18:19:00.210+0800 writing ddsdb.t1 to
03. 2018-09-28T18:19:00.211+0800 done dumping ddsdb.t1 ( 17 documents)

```

2) 恢复数据

[Top](#)

```

01. [root@mongodb51 ~] # mongo -- host 192.168.4.51 -- port 27077

```

```

02. > show tables;
03. col
04. t1
05. > db.t1.remove({})
06. WriteResult({ "nRemoved" : 17 })
07. >exit
08. [root@mongodb51 ~]# mongorestore -- host 192.168.4.51 -- port 27077 -d ddsdb /root/
09. //-d ddsdb恢复到数据库的目录，从/root/bbsdb.t1/ddsdb/目录恢复
10. 2018-09-28T18:26:16.889+0800 the --db and --collection args should only be used whe
11. 2018-09-28T18:26:16.890+0800 building a list of collections to restore from /root/bbsd
12. 2018-09-28T18:26:16.891+0800 reading metadata for ddsdb.t1 from /root/bbsdb.t/dds
13. 2018-09-28T18:26:16.891+0800 restoring ddsdb.t1 from /root/bbsdb.t/ddsdb/t1.bson
14. 2018-09-28T18:26:16.893+0800 no indexes to restore
15. 2018-09-28T18:26:16.893+0800 finished restoring ddsdb.t1 (17 documents)
16. 2018-09-28T18:26:16.893+0800 done

```

步骤二：数据的导入导出

1) 导出

用csv的格式导出

```

01. [root@mongodb51 ~]# mongoexport -- host 192.168.4.51 -- port 27077 -d ddsdb -c t:
02. //导出csv格式，必须要指定导出的字段名，导出name字段
03. 2018-09-28T18:29:24.653+0800 connected to: 192.168.4.51:27077
04. 2018-09-28T18:29:24.654+0800 exported 17 records
05. [root@mongodb51 ~]# cat lig1.csv
06. name
07. bob
08. jerry
09. zhangsan
10. lisi
11. alice
12. lilei
13. hehe
14. 呵呵
15. bobo
16.
17.
18.
19. jerry

```

[Top](#)

```

20.
21. y a y a
22.
23. [root@mongodb51 ~]# mongoexport --host 192.168.4.51 --port 27077 -d ddsdb -c t1
24. //从ddsdb的它1里导出名字为bob的name字段和age字段
25. 2018-09-28T18:31:25.627+0800 connected to: 192.168.4.51:27077
26. 2018-09-28T18:31:25.628+0800 exported 1 record

```

用json的格式导出

```

01. [root@mongodb51 ~]# mongoexport --host 192.168.4.51 --port 27077 -d ddsdb -c t1 --
02. //导出json格式
03. 2018-09-28T18:33:13.349+0800 connected to: 192.168.4.51:27077
04. 2018-09-28T18:33:13.350+0800 exported 17 records
05.
06. [root@mongodb51 ~]# mongoexport --host 192.168.4.51 --port 27077 -d ddsdb -c t1 --
07. //指定列名导出，导出name字段
08. 2018-09-28T18:33:35.914+0800 connected to: 192.168.4.51:27077
09. 2018-09-28T18:33:35.915+0800 exported 17 records

```

2) 导入

```

01. [root@mongodb51 ~]# mongo --host 192.168.4.51 --port 27077
02. > use ddsdb
03. switched to db ddsdb
04. > show tables;
05. col
06. t1
07. > db.t1.remove({})
08. WriteResult({ "nRemoved" : 17 })
09. > exit

```

用json的格式导入：表里要没有数据，不然导入不成功


```

01. [root@mongodb51 ~]# mongoimport --host 192.168.4.51 --port 27077 -d ddsdb -c t1 --
02. 2018-09-28T18:35:22.341+0800 connected to: 192.168.4.51:27077

```

[Top](#)

```
03. 2018-09-28T18:35:22.343+0800 imported 17 documents
04.
05. [root@mongodb51 ~] # mongo --host 192.168.4.51 --port 27077
06. > use ddsdb
07. switched to db ddsdb
08. > db.t1.count({})
09. 17
```



用csv的格式导入：表里可以有数据

```
01. [root@mongodb51 ~] # mongoimport --host 192.168.4.51 --port 27077 -d ddsdb -c t1 .
02. //必须指定文件的列名，不然不成功 -f和--headerline不能一起用 --headerline：把第一
03. 2018-09-28T18:37:36.778+0800 connected to: 192.168.4.51:27077
04. 2018-09-28T18:37:36.779+0800 imported 11 documents
```



[Top](#)