TTS 11.0 COOKBOOK

(NSD NOSQL DAY03)

版本编号 11.0

达内IT培训集团

2019-06

达内 IT 培训集团

NSD NOSQL DAY03

1. 案例 1：redis 主从复制

- **问题**
  - 具体要求如下：
  - 将主机 192.168.4.51 配置为主服务器
  - 将主机 192.168.4.52 配置为 192.168.4.51 的从服务器
  - 测试配置

- **方案**

部署 redis 一主一从复制结构，主机角色，如图-1 所示：

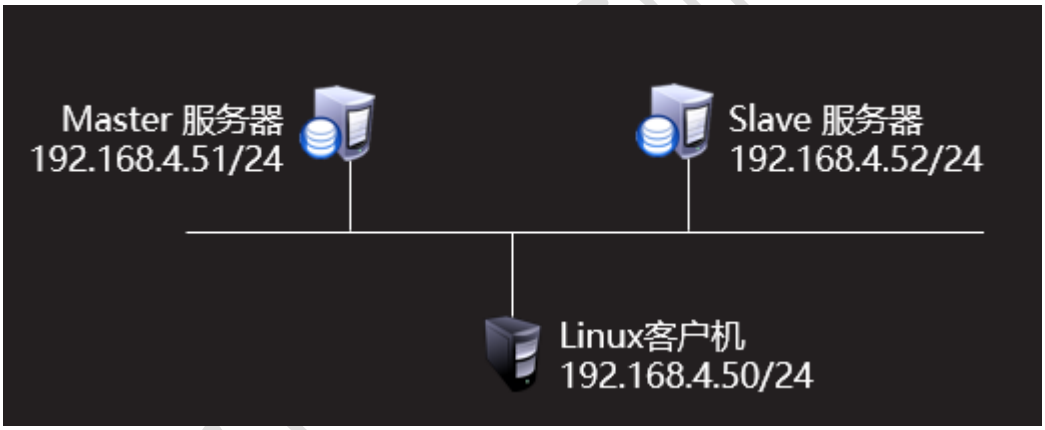

图 - 1

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：将主机 192.168.4.51 配置为主服务器

1） 每台 redis 服务器，默认都是主服务器;所以主服务器不需要配置。

```
[root@redisA ~]# redis-cli -h 192.168.4.51 -p 6351
192.168.4.51:6351> info replication        //查看复制信息
# Replication
role:master          //是 master 服务器
```

```
connected_slaves:0    //从服务器个数零台
master_replid:eaa14478158a71c41f947eaea036658c2087e8f2
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:0
second_repl_offset:-1
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
192.168.4.51:6351>
```

步骤二：将主机 192.168.4.52 配置为 192.168.4.51 的从服务器

1）命令行配置（马上生效）

```
[root@redisB ~]# redis-cli -h 192.168.4.52 -p 6352
192.168.4.52:6352> slaveof  192.168.4.51  6351  //指定主服务器 ip 地址与端口
OK
192.168.4.52:6352> info replication   //查看复制信息
# Replication
role:slave  //从服务器
master_host:192.168.4.51       //主服务器 ip 地址
master_port:6351 //主服务器端口
master_link_status:up //连接状态开启
master_last_io_seconds_ago:3
master_sync_in_progress:0
```

2）永久配置（重新 redis 服务后，依然有效）

```
[root@redisB ~]# vim /etc/redis/6379.conf
slaveof  192.168.4.51  6351    //在文件末尾添加或在原有配置项上修改都可以
:wq
```

3）在主服务器查看复制信息

```
[root@redisA ~]# redis-cli -h 192.168.4.51 -p 6351
192.168.4.51:6351> info replication  //查看复制信息
# Replication
role:master
connected_slaves:1 //从服务器个数 1 台
slave0:ip=192.168.4.52,port=6352,state=online,offset=14,lag=1    //从服务器信息
master_replid:db7932eb0ea4302bddbebd395efa174fb079319f
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:14
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:14
192.168.4.51:6351>
```

步骤三：测试配置

1）客户端连接主服务器存储数据

```
[root@client50 ~]# redis-cli -h 192.168.4.51 -p 6351
192.168.4.51:6351> set x  9
OK
192.168.4.51:6351> set y  8
OK
192.168.4.51:6351> set z  7
OK
192.168.4.51:6351>
```

2）在从服务器本机登录，查看数据（与主服务器数据一致）

```
[root@redisB ~]#  redis-cli -h 192.168.4.52 -p 6352
192.168.4.52:6352> keys  *
1) “x”
2) “y”
3) “z”
192.168.4.52:6352>
```

2. 案例 2：配置带验证的主从复制

- **问题**
  - 具体要求如下：
  - 基于案例 1 的配置
  - 设置主服务器 192.168.4.51 设置连接密码 123456
  - 配置从服务器 192.168.4.52

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：设置主服务器 192.168.4.51 设置连接密码 123456

1） 修改主服务器的配置文件，设置密码。

```
[root@redisA ~]# vim +501 /etc/redis/6379.conf
requirepass  123456   //设置密码
:wq

[root@redisA ~]# vim +43  /etc/init.d/redis_6379  //修改脚本
     $CLIEXEC -h 192.168.4.51 -p 6351 -a 123456  shutdown //添加密码
:wq
[root@redisA ~]# /etc/init.d/redis_6379 stop //停止服务

[root@redisA ~]# /etc/init.d/redis_6379  start //启动服务
Starting Redis server...
[root@redisA ~]#
[root@redisA ~]# netstat -utnlp  | grep  :6351  //查看端口
tcp         0       0 192.168.4.51:6351        0.0.0.0:*                   LISTEN
11523/redis-server
```

步骤二：配置从服务器 192.168.4.52

1） 修改配置文件，设置主服务器连接密码。

```
[root@redisB ~]# /etc/init.d/redis_6379 stop //停止服务

[root@redisB ~]# vim +289 /etc/redis/6379.conf
masterauth  123456   //设置密码
:wq

[root@redisA ~]# /etc/init.d/redis_6379  start //启动服务
Starting Redis server...
[root@redisA ~]#
[root@redisA ~]# netstat -utnlp  | grep  :6351  //查看端口
tcp         0       0 192.168.4.51:6351        0.0.0.0:*                   LISTEN
11523/redis-server
```

2） 在从服务器本机连接服务，查看复制信息

```
[root@redisB ~]# redis-cli -h 192.168.4.52 -p 6352
192.168.4.52:6352> info replication   //查看复制信息
# Replication
role:slave  //从服务器
master_host:192.168.4.51       //主服务器 ip 地址
master_port:6351 //主服务器端口
master_link_status:up //连接状态开启
master_last_io_seconds_ago:3
master_sync_in_progress:0
……
……
192.168.4.52:6352>
```

3. 案例 3：哨兵服务

- **问题**
  - 具体要求如下：
  - 基于案例 2 配置
  - 配置哨兵服务
  - 测试配置

- **方案**

角色规划如图-1 所示：

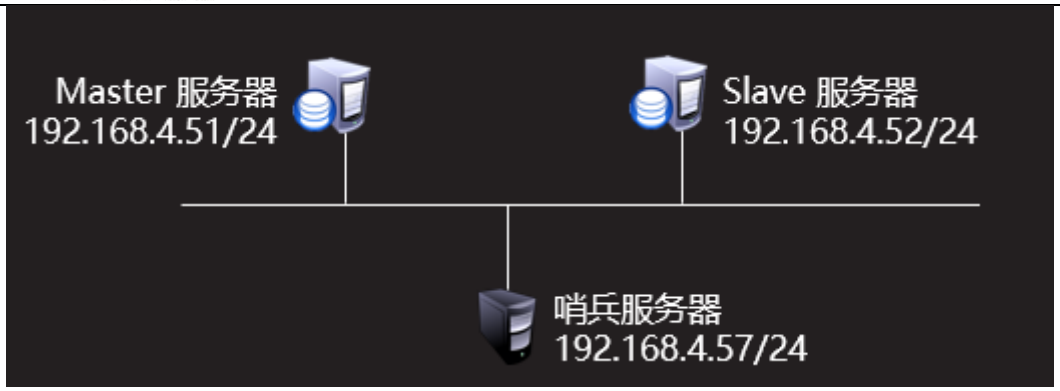

图 - 1

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：配置哨兵服务（192.168.4.57）

1） 安装源码软件 redis ，无需做初始化配置。

```
[root@redis57 redis]# yum -y install gcc
[root@redis57 redis]# tar -zxf redis-4.0.8.tar.gz
[root@redis57 redis]# cd redis-4.0.8/
[root@redis1 redis-4.0.8]# make
[root@redis1 redis-4.0.8]# make install
```

2）编辑主配置文件

```
[root@redis57 redis]# vim  /etc/sentinel.conf  //创建主配置文件
sentinel   monitor   server51   192.168.4.51   6351   1 //监视主服务器
bind  0.0.0.0    //哨兵服务地址（表示本机所有网络接口）
sentinel auth-pass  server51   123456   //主服务器密码
:wq
```

3）启动哨兵服务

```
[root@redis57 redis]# redis-sentinel /etc/sentinel.conf //启动哨兵服务
25371:X 28 Sep 11:16:54.993 # +sdown master redis51 192.168.4.51 6351
25371:X 28 Sep 11:16:54.993 # +odown master redis51 192.168.4.51 6351 #quorum 1/1
25371:X 28 Sep 11:16:54.993 # +new-epoch 3
25371:X 28 Sep 11:16:54.993 # +try-failover master redis51 192.168.4.51 6351
25371:X        28        Sep        11:16:54.994        #        +vote-for-leader
be035801d4d48eb63d8420a72796f52fc5cec047 3
...
25371:X 28 Sep 11:16:55.287 * +slave slave 192.168.4.51:6351 192.168.4.51 6351 @
```

```
redis51 192.168.4.52 6351
   25371:X 28 Sep 11:17:25.316 # +sdown slave 192.168.4.51:6379 192.168.4.51 6379 @
redis51 192.168.4.52 6352
```

步骤二：测试配置

1）停止主服务器 51 的 redis 服务

```
[root@redisA ~]#  /etc/init.d/redis_6379 stop
Stopping ...
Waiting for Redis to shutdown ...
Redis stopped
[root@redisA ~]#
```

2）在服务器 52 主机，查看复制信息

```
 [root@redisB ~]# redis-cli -h 192.168.4.52 -p 6352
192.168.4.52:6352> info replication
# Replication
role:master  //角色是 master
connected_slaves:0
……
……
```

4. 案例 2 ： 使用 RDB 文件恢复数据

- **问题**
  - 要求如下：
  - 启用 RDB
  - 设置存盘间隔为 120 秒且 10 个 key 改变数据自动存盘
  - 备份 RDB 文件
  - 删除数据
  - 使用 RDB 文件恢复数据

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：使用 RDB 文件恢复数据

RDB 介绍：

Redis 数据库文件，全称 Reids DataBase

数据持久化方式之一

在指定时间间隔内，将内存中的数据集快照写入硬盘

术语叫 Snapshot 快照

恢复时，将快照文件直接读到内存里

相关配置参数

文件名

dbfilename “dump.rdb” 文件名

数据从内存保存到硬盘的频率

save 900 1 900 秒内且有 1 个 key 改变

save 300 10 300 秒内且有 10 个 key 改变

save 60 10000 60 秒内且有 10000 个 key 改变

```
[root@redisA ~]# vim /etc/redis/6379.conf
dbfilename dump.rdb
save 900 1
#save 300 10 //注释原有设置
save 120 10 //时间修改为 120 秒
save 60 10000
: wq

[root@redisA ~]#  /etc/init.d/redis_6379 stop  //停止服务
Stopping ...
Waiting for Redis to shutdown ...
Redis stopped
[root@redisA ~]#

[root@redisA ~]# rm -rf  /var/lib/redis/6379/* //清空数据库目录
[root@redisA ~]# /etc/init.d/redis_6379 start //启动服务
Starting Redis server...
[root@redisA ~]#

[root@redisA ~]# ls /var/lib/redis/6379  //此时，查看数据库目录下没有 dump.rdb 文件
[root@redisA ~]#


[root@redisA ~]# redis-cli -h 192.168.4.51  -p 6351  -a 123456  //连接服务，在 200 秒内存储 10 个变量，就会自动在数据库目录下创建 dump.rdb 文件
192.168.4.51:6351> set v1 k1
OK
192.168.4.51:6351> set v2 k1
OK
192.168.4.51:6351> set v3 k1
OK
192.168.4.51:6351> set v4 k1
OK
192.168.4.51:6351> set v45 k1
OK
192.168.4.51:6351> set v46 k1
OK
192.168.4.51:6351> set v7 k1
OK
192.168.4.51:6351> set v8 k1
OK
192.168.4.51:6351> set v9 k1
OK
```

```
192.168.4.51:6351> set v10 k1
OK
192.168.4.51:6351> keys *
 1) "v2"
 2) "v9"
 3) "v10"
 4) "v45"
 5) "v4"
 6) "v1"
 7) "v46"
 8) "v8"
 9) "v7"
10) "v3"
192.168.4.51:6351>exit

[root@redisA ~]# ls /var/lib/redis/6379  //此时，查看数据库目录下有 dump.rdb 文件
dump.rdb
[root@redisA ~]#
```

备份数据

```
[root@redisA ~]# cd /var/lib/redis/6379/
[root@redisA 6379]# ls
dump.rdb
[root@redisA 6379]# cp dump.rdb  /tmp/dump.rdb  //备份 dump.rdb 文件
[root@redisA 6379]# scp  /tmp/dump.rdb  root@192.168.4.56:/root/ //传递备份文件给目标主机
```

删除数据 （56 主机模拟误删除数据）

```
[root@redis56 ~]# redis-cli -h 192.168.4.56 -p 6356     //连接服务
192.168.4.56:6356> flushall
OK
192.168.4.51:6379> keys *      //已经没有数据
(empty list or set)
192.168.4.56:6356> exit
[root@redis56 ~]#
```

恢复数据(56 主机使用备份文件恢复数据)

```
[root@redis56 ~]# /etc/init.d/redis_6379 stop  //停止服务
Stopping ...
Waiting for Redis to shutdown ...
Redis stopped
[root@redis56 ~]#
[root@redis56 ~]# rm  -rf  /var/lib/redis/6379/*  //清空数据库目录
[root@redis56 ~]# cp  /tmp/dump.rdb   /var/lib/redis/6379/ //拷贝备份文件到数据库目录下
[root@redis56 ~]# /etc/init.d/redis_6379  start  // 启动服务
Starting Redis server...

[root@redis56 ~]# redis-cli -h 192.168.4.56 -p 6356 //访问服务
192.168.4.56:6356> keys *  //查看数据
 1) "v7"
 2) "v46"
 3) "v45"
```

```
 4) "v8"
 5) "v4"
 6) "v2"
 7) "v1"
 8) "v3"
 9) "v9"
10) "v10"
192.168.4.56:6356>
```

5. 案例 5 ：使用 AOF 文件恢复数据

- **问题**
  - 具体要求如下：
  - 启用 AOF
  - 备份 AOF 文件
  - 删除数据
  - 使用 AOF 文件恢复数据
- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：使用 AOF 文件恢复数据

1）修改配置文件

```
[root@redisA ~]# redis-cli -h 192.168.4.51 -p 6351 -a 123456 //连接服务
192.168.4.51:6351>config  set   appendonly yes   //启用 aof，默认 no
192.168.4.51:6351> config  rewrite //写进配置文件
192.168.4.51:6351> save
192.168.4.51:6351> exit

[root@redisA ~]# ls  /var/lib/redis/6379/   //会出现 appendonly.aof 文件
appendonly.aof  dump.rdb
[root@redisA ~ ]#
```

2）备份 AOF 文件

```
[root@redisA ~]# cd /var/lib/redis/6379/
[root@redisA 6379]# cp appendonly.aof /tmp/appendonly.aof
[root@redisA 6379]# scp /tmp/appendonly.aof  root@192.168.4.57:/root/  //传递备份文件给目标主机
```

3）删除数据（在 57 主机 默认数据误删除）

```
[root@redis57 ~]# redis-cli -h 192.168.4.57 -p 6357  //连接服务
192.168.4.57:6357> flushall  //清除数据
OK
192.168.4.57:6357> keys * //查看数据
(empty list or set)
```

```
192.168.4.57:6357> exit
[root@redis57  ~ ]#
```

4）使用 AOF 文件恢复数据

```
[root@redis57 ~]# vim +673 /etc/redis/6379.conf
appendonly  yes  //启用 AOF
:wq
[root@redis57 ~]#
[root@redis57 ~]# /etc/init.d/redis_6379 stop  //停止服务
Stopping ...
Waiting for Redis to shutdown ...
Redis stopped
[root@redis57 ~]#
[root@redis57 ~]#  /etc/init.d/redis_6379  start //启动服务
Starting Redis server...
[root@redis57 ~]#
[root@redis57 ~]# rm  -rf  /var/lib/redis/6379/* //删除没有数据的文件
[root@redis57 ~]# cp  /root/appendolny.aof /var/lib/redis/6379/  //拷贝文件
[root@redis57 ~]# /etc/init.d/redis_6379  start  //启动服务
Starting Redis server...
[root@redis57 ~]# redis-cli -h 192.168.4.57 -p 6357 //连接服务
192.168.4.57:6357> keys *  //查看数据
 1) "v9"
 2) "v5"
 3) "v8"
 4) "v2"
 5) "v1"
 6) "v4"
 7) "v10"
 8) "v6"
 9) "v7"
10) "v3"
192.168.4.57:6357>
```

6. 案例 6：string 字符串

- **问题**
  - 练习命令的使用，具体命令如下：
  - set getrange strlen append setbit bitcount
  - decr decrby incr incrby incrbyfloat

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：string 字符串

1）set key value [ex seconds] [px milliseconds] [nx|xx]
设置 key 及值，过期时间可以使用秒或毫秒为单位
setrange key offset value

```
192.168.4.56:6356> set  x 9 ex 10 //单位秒
```

```
OK
192.168.4.56:6356>
192.168.4.56:6356> set  y 29 px 10 //单位毫秒
OK
192.168.4.56:6356>
192.168.4.56:6356> set  y 39 NX //不存在赋值
OK
192.168.4.56:6356> get y //变量值没变
"39"
192.168.4.56:6356>
192.168.4.56:6356> set  y 49 xx //变量存在赋值
OK
192.168.4.56:6356> get y //变量变了
"49"
192.168.4.56:6356>
```

2）从偏移量开始复写 key 的特定位的值

```
192.168.4.51:6351> set  first  "hello world"
OK
192.168.4.51:6351> setrange  first  6  "Redis"  //改写为 hello Redis
(integer) 11
192.168.4.51:6351> get first
"hello Redis"
```

3）strlen key，统计字串长度

```
192.168.4.51:6379> strlen first
(integer) 11
```

4）append key value 存在则追加，不存在则创建 key 及 value，返回 key 长度

```
192.168.4.51:6379> append myname jacob
(integer) 5
```

5）setbit key offset value　对 key 所存储字串，设置或清除特定偏移量上的位(bit)，value 值可以为 1 或 0，offset 为 0~2^32 之间，key 不存在，则创建新 key

```
192.168.4.51:6379> setbit  bit  0  1        //设置 bit 第 0 位为 1
(integer) 0
192.168.4.51:6379> setbit  bit  1  0        //设置 bit 第 1 位为 0
(integer) 0
```

6）bitcount key　统计字串中被设置为 1 的比特位数量

```
192.168.4.51:6379> setbit  bits 0 1         //0001
(integer) 0
192.168.4.51:6379> setbit  bits 3 1         //1001
(integer) 0
192.168.4.51:6379> bitcount  bits           //结果为 2
(integer) 2
```

记录网站用户上线频率，如用户 A 上线了多少天等类似的数据，如用户在某天上线，则使用 setbit，以用户名为 key，将网站上线日为 offset，并在该 offset 上设置 1，最后计算用户总上线次数时，使用 bitcount 用户名即可，这样即使网站运行 10 年，每个用户

仅占用 10*365 比特位即 456 字节

```
192.168.4.51:6379> setbit  peter  100  1        //网站上线 100 天用户登录了一次
(integer) 0
192.168.4.51:6379> setbit  peter  105  1        //网站上线 105 天用户登录了一次
(integer) 0
192.168.4.51:6379> bitcount  peter
(integer) 2
```

7）decr key 将 key 中的值减 1，key 不存在则先初始化为 0，再减 1

```
192.168.4.51:6379> set z 10
OK
192.168.4.51:6379> decr z
(integer) 9
192.168.4.51:6379> decr z
(integer) 8


192.168.4.51:6379> decr bb
(integer) -1
192.168.4.51:6379> decr bb
(integer) -2
```

8）decrby key decrement 将 key 中的值，减去 decrement

```
192.168.4.51:6379> set count 100
OK
192.168.4.51:6379> DECRBY cc 20    //定义每次减少 20（步长）
(integer) -20
192.168.4.51:6379> DECRBY cc 20
(integer) -40
```

9）getrange key start end 返回字串值中的子字串，截取范围为 start 和 end，负数偏移量表示从末尾开始计数，-1 表示最后一个字符，-2 表示倒数第二个字符

```
192.168.4.51:6379> set x 123456789
OK
192.168.4.51:6379> getrange x -5 -1
"56789"
192.168.4.51:6379> getrange x 0 4
"12345"
```

10）incr key 将 key 的值加 1，如果 key 不存在，则初始为 0 后再加 1，主要应用为计数器

```
192.168.4.51:6379> set page 20
OK
192.168.4.51:6379> incr page
(integer) 21
```

11）incrby key increment 将 key 的值增加 increment

```
192.168.4.51:6379> set x 10
OK
192.168.4.51:6379> incr x
(integer) 11
192.168.4.51:6379> incr x
(integer) 12
```

12) incrbyfloat key increment 为 key 中所储存的值加上浮点数增量 increment

```
192.168.4.51:6379> set num 16.1
OK
192.168.4.51:6379> incrbyfloat num 1.1
"17.2"
```

7. 案例 7：list 列表

- **问题**
  - 练习命令使用，具体如下：
  - lpush llen lrange lpop
  - lindex lset rpush rpop

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：list 列表

1) lpush key value [value…] 将一个或多个值 value 插入到列表 key 的表头，Key 不存在，则创建 key

```
192.168.4.51:6379> lpush list a b c        //list 值依次为 c b a
(integer) 3
```

2) lrange key start stop 从开始位置读取 key 的值到 stop 结束

```
192.168.4.51:6379> lrange list 0 2     //从 0 位开始，读到 2 位为止
1) "c"
2) "b"
3) "a"
192.168.4.51:6379> lrange list 0 -1    //从开始读到结束为止
1) "c"
2) "b"
3) "a"
192.168.4.51:6379> lrange list 0 -2        //从开始读到倒数第 2 位值
1) "c"
2) "b"
```

3) lpop key 移除并返回列表头元素数据，key 不存在则返回 nil

```
192.168.4.51:6379> lpop list       //删除表头元素，可以多次执行
"c"
192.168.4.51:6379>  LPOP list
"b"
```

4) llen key 返回列表 key 的长度

```
192.168.4.51:6379>  llen list
(integer) 1
```

5）lindex key index　　返回列表中第 index 个值

```
192.168.4.51:6379> lindex  list  1
"c"
```

6）lset key index value　　将 key 中 index 位置的值修改为 value

```
192.168.4.51:6379> lpush list a b c d
(integer) 5
192.168.4.51:6379> lset list 3 test        //将 list 中第 3 个值修改为 test
OK
```

7）rpush key value [value…]　将 value 插入到 key 的末尾

```
192.168.4.51:6379> rpush list3  a b c  //list3 值为 a b c
(integer) 3
192.168.4.51:6379> rpush list3 d   //末尾插入 d
(integer) 4
```

8）rpop key　　删除并返回 key 末尾的值

```
192.168.4.51:6379> RPOP list3
"d"
```

8. 案例 8：hash 表

- **问题**
  - 练习命令使用，具体如下：
  - hset　hmset　hgetall hkeys　hvals
  - hget　hmget　hdel

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：hash 表

1）hset key field value　　将 hash 表中 field 值设置为 value

```
192.168.4.51:6379> hset site google 'www.g.cn'
(integer) 1
192.168.4.51:6379> hset site baidu 'www.baidu.com'
(integer) 1
```

2）hget key filed　　获取 hash 表中 field 的值

```
192.168.4.51:6379> hget site google
"www.g.cn"
```

3）hmset key field value [field value…]　同时给 hash 表中的多个 field 赋值

```
192.168.4.51:6379> hmset site google www.g.cn  baidu www.baidu.com
OK
```

4）hmget key field [field…]　返回 hash 表中多个 field 的值

```
192.168.4.51:6379> hmget site google baidu
1) "www.g.cn"
2) "www.baidu.com"
```

5）hkeys key　返回 hash 表中所有 field 名称

```
192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com
OK
192.168.4.51:6379> hkeys  site
1) "google"
2) "baidu"
```

6）hgetall key　返回 hash 表中所有 key 名和对应的值列表

```
192.168.4.51:6379> hgetall site
1) "google"
2) "www.g.cn"
3) "baidu"
4) "www.baidu.com"
```

7）hvals key　返回 hash 表中所有 key 的值

```
192.168.4.51:6379> hvals site
1) "www.g.cn"
2) "www.baidu.com"
```

8）hdel key field [field…]　删除 hash 表中多个 field 的值，不存在则忽略

```
192.168.4.51:6379> hdel  site  google  baidu
(integer) 2
```