Redis第二天

目录

[一、Redis数据类型 1](#_Toc531621806)

[1、string字符串 1](#_Toc531621807)

[2、hash哈希 1](#_Toc531621808)

[3、list链表 1](#_Toc531621809)

[4、集合(set) 1](#_Toc531621810)

[（1）sadd 2](#_Toc531621811)

[（2）smembers 2](#_Toc531621812)

[（3）sdiff 2](#_Toc531621813)

[（2）sinter 3](#_Toc531621814)

[（3）sunion 3](#_Toc531621815)

[（4）scard 4](#_Toc531621816)

[5、（有序集合）zset 4](#_Toc531621817)

[（1）zadd 5](#_Toc531621818)

[（2）zrange 5](#_Toc531621819)

[（3）zrevrange 6](#_Toc531621820)

[（4）zremrangebyrank 6](#_Toc531621821)

[（5）zcard 7](#_Toc531621822)

[（6）zscore 7](#_Toc531621823)

[二、主从复制 9](#_Toc531621824)

[1、什么是主从复制； 9](#_Toc531621825)

[2、主从通信过程 10](#_Toc531621826)

[3、具体配置步骤 11](#_Toc531621827)

[4、redis主从复制的缺陷 12](#_Toc531621828)

[三、Redis 中的事务 12](#_Toc531621829)

[1、基本操作 13](#_Toc531621830)

[2、discard取消事务 13](#_Toc531621831)

[3、其他说明 14](#_Toc531621832)

[4、watch命令 15](#_Toc531621833)

[四、电商秒杀案例 16](#_Toc531621834)

[五、综合案例 17](#_Toc531621835)

[1、用户注册 17](#_Toc531621836)

[2、用户列表 18](#_Toc531621837)

[3、用户修改 18](#_Toc531621838)

[4、用户删除 18](#_Toc531621839)

[5、用户登录 18](#_Toc531621840)

[6、添加关注 18](#_Toc531621841)

# 一、Redis数据类型

## 1、string字符串

## 2、hash哈希

## 3、list链表

## 4、集合(set)

redis的set是string类型的无序集合。

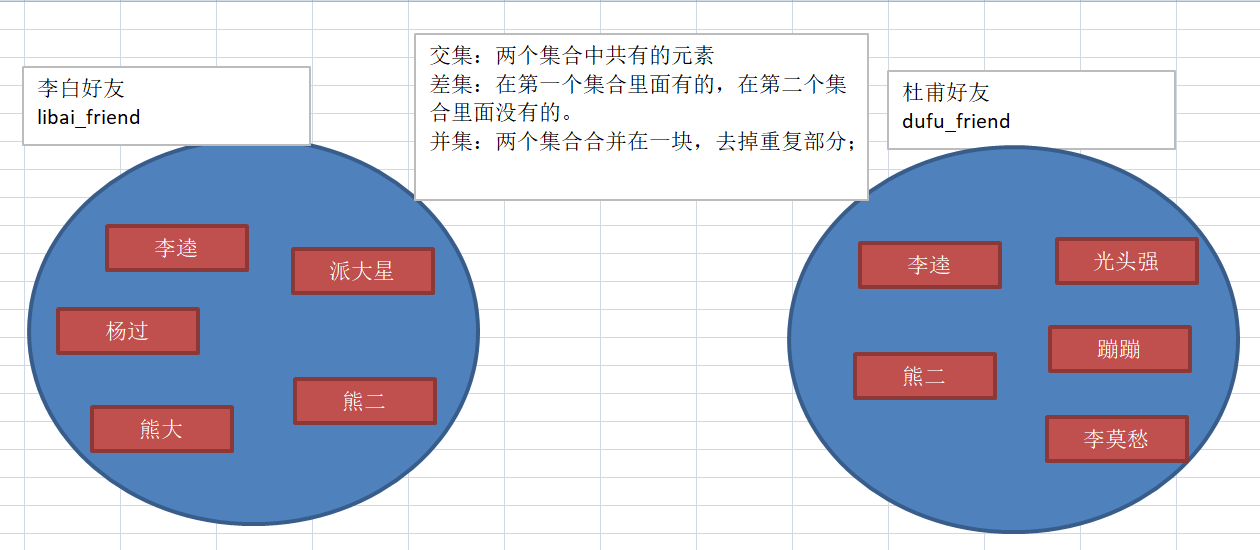
set元素最大可以包含(2的32次方-1)(整型最大值)个元素。

关于set集合类型除了基本的**添加、删除**操作，其他有用的操作还包含集合的取并集(union)，交集(intersection)，差集(difference)。通过这些操作可以很容易的实现sns中的好友推荐功能。

sina公司的好友关注关系就大量使用了set集合类型。

**注意：每个集合中的各个元素不能重复。**

该类型应用场合：qq好友推荐。



tom朋友圈(与某某是好友)：mary jack xiaoming wang5 wang6

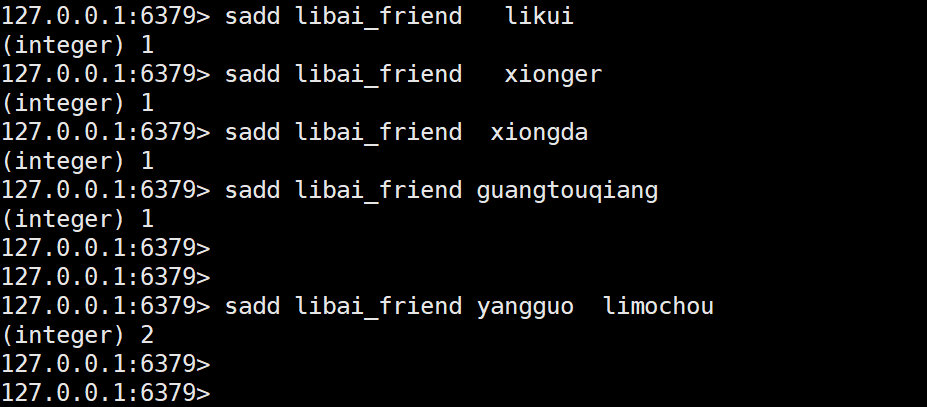
linken朋友圈(与某某是好友)：yuehan daxiong luce wang5 wang6

### （1）sadd

向集合中添加元素

语法：

sadd 集合名（键名） 元素名称

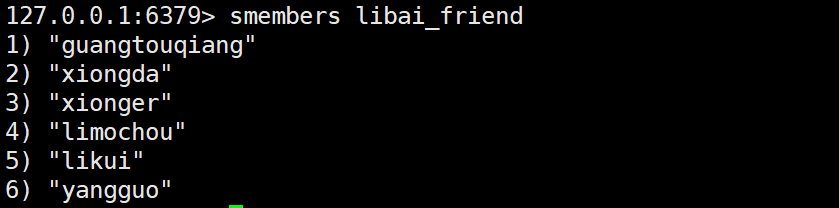


### （2）smembers

获取集合中的元素

语法：

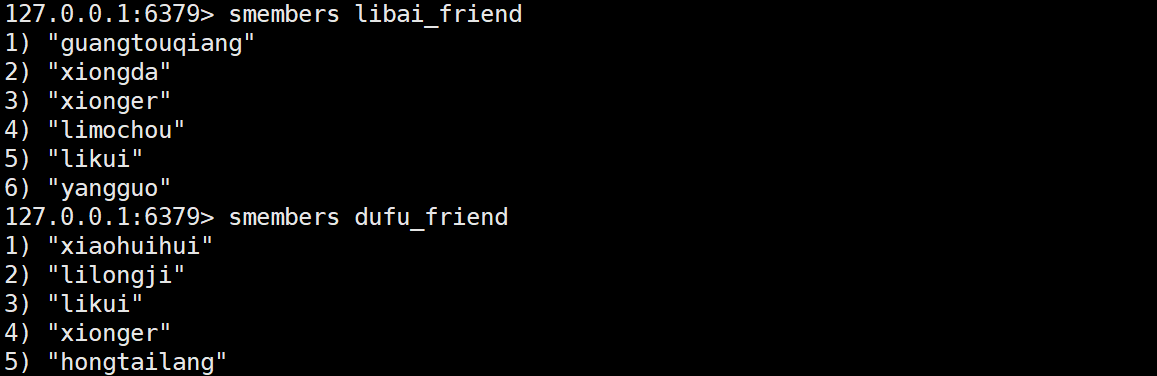
smembers 集合名

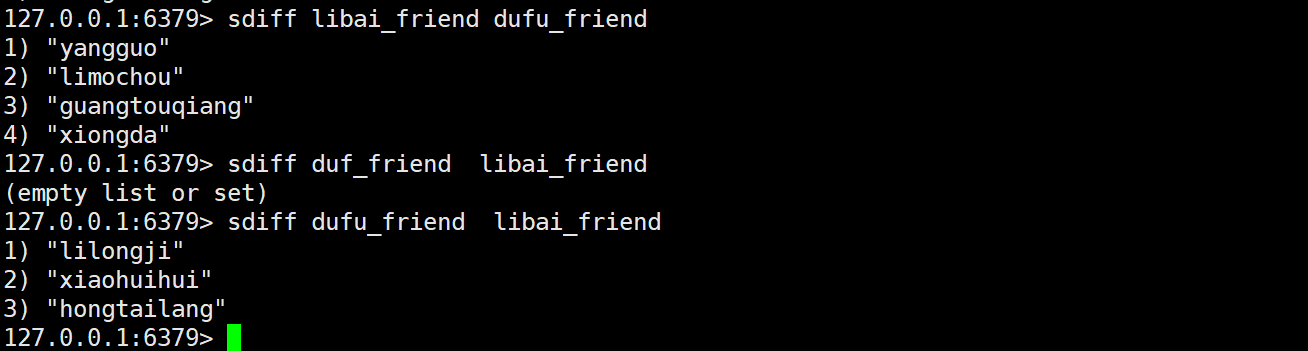


### （3）sdiff

获取集合中的差集（在集合1中存在，不在集合2中存在的元素）

语法：sdiff 集合1 集合2

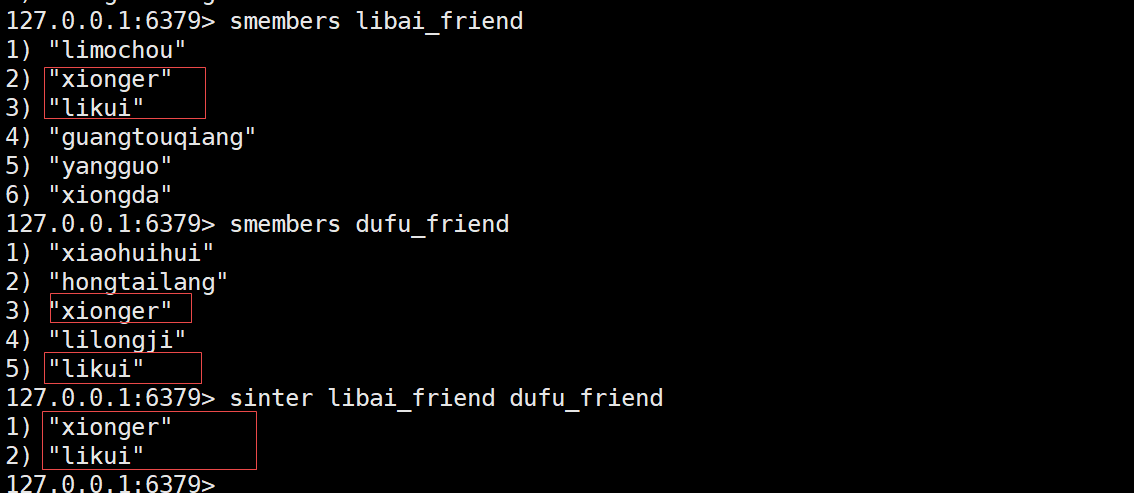




### （2）sinter

获取交集（在两个集合中都存在的元素）

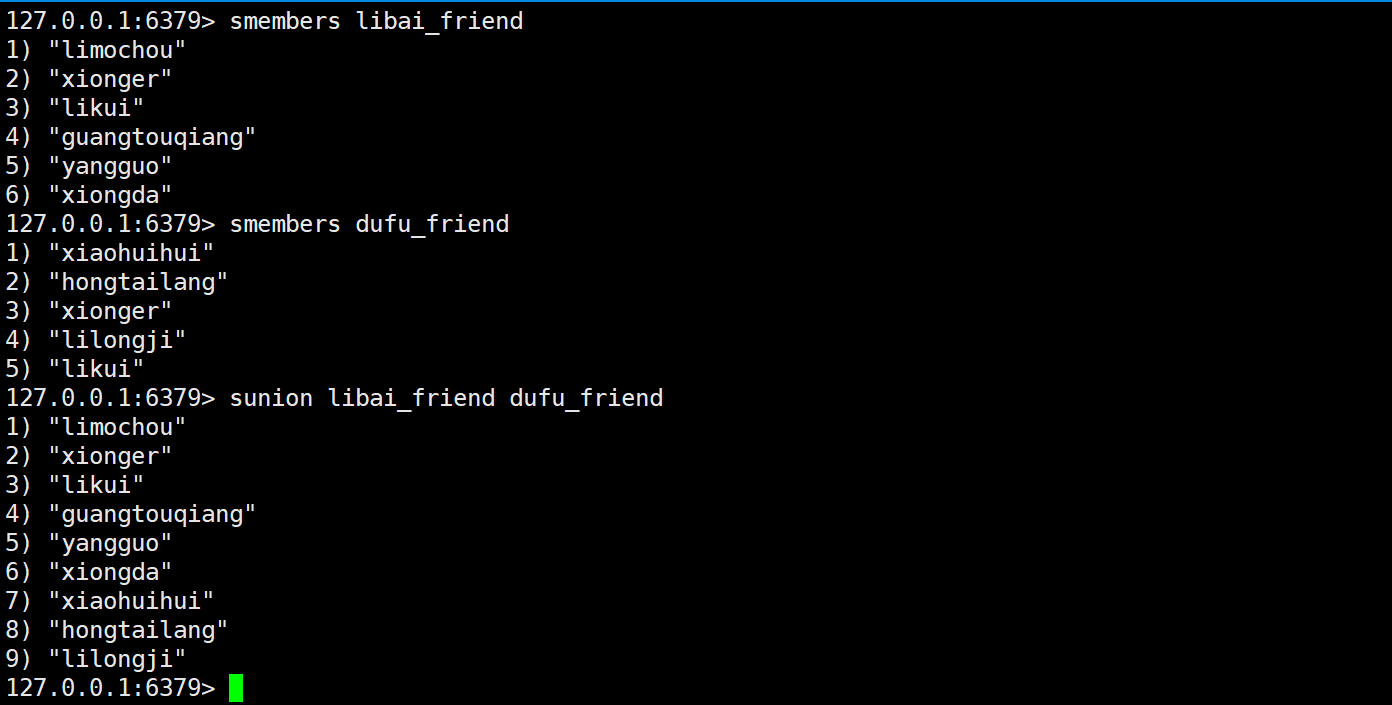
语法：sinter 集合1 集合2



### （3）sunion

求并集（两个集合合并后，去掉重复的元素）

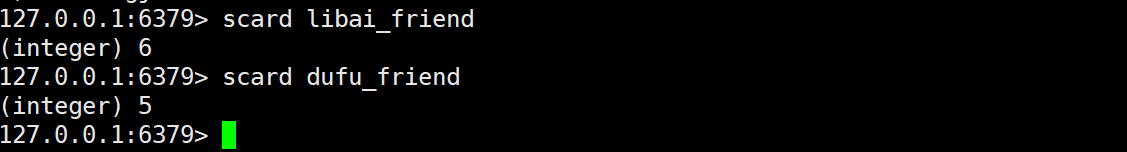
语法：sunion 集合1 集合2



### （4）scard

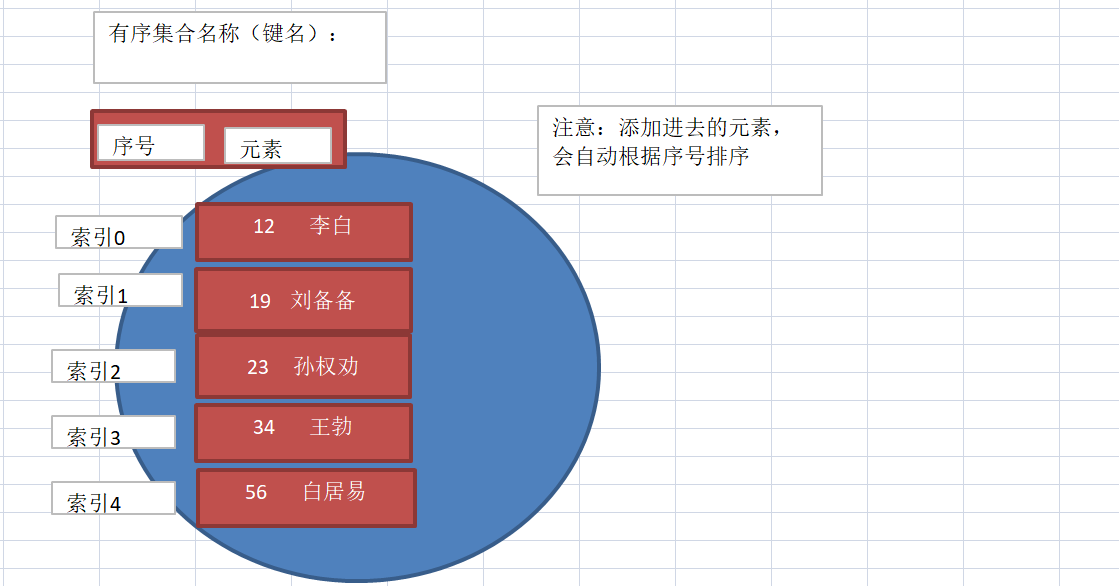
获取集合中元素的个数

语法：scard 集合名称



## 5、（有序集合）zset

sorted set是set的一个升级版本，他在set的基础上增加了一个**顺序属性(权值)**，这一属性在添加修改元素的时候可以指定，**每次指定后，zset会自动重新按新的值调整顺序。**操作中的key理解为zset的名字。

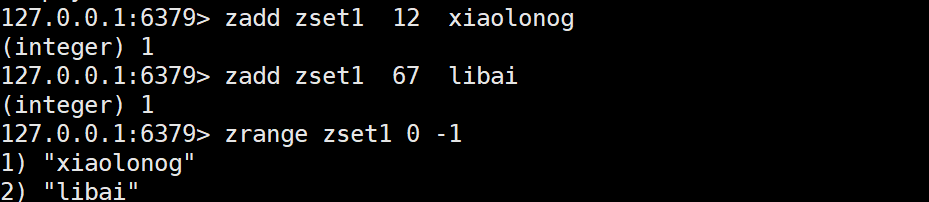


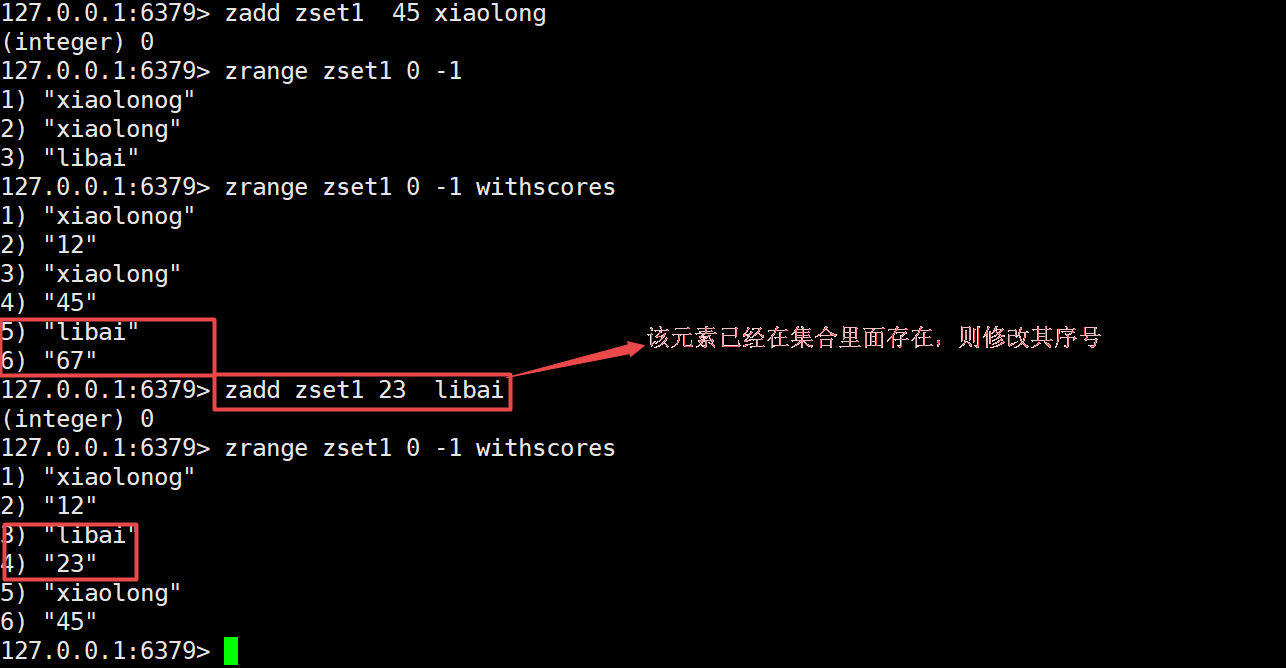
**注意：序号是可以重复的，在添加元素时，如果该元素已经存在，则更新序号的值。**

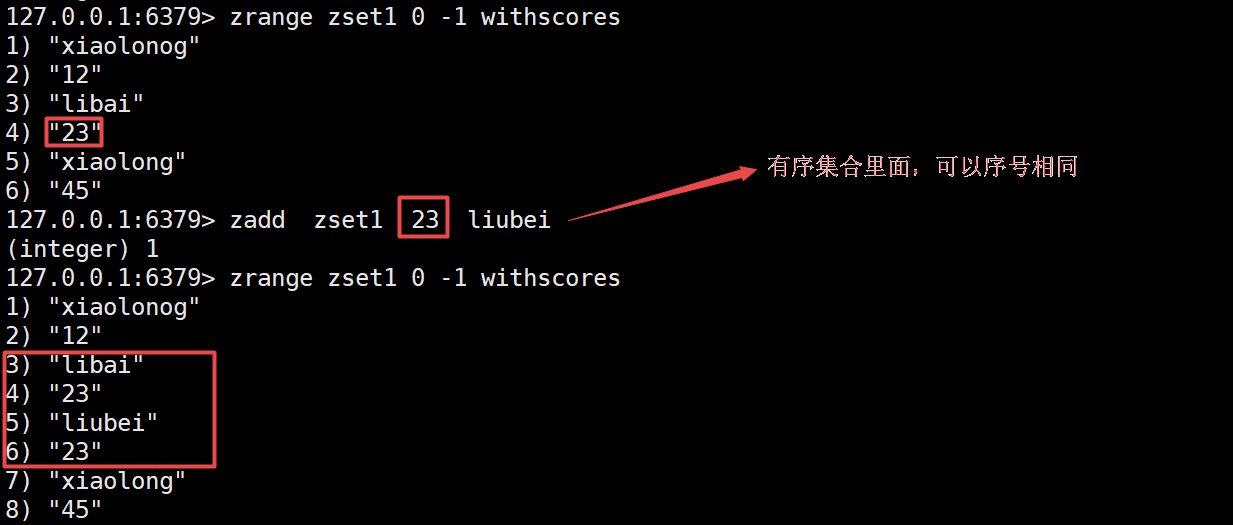
### （1）zadd

向有序集合中添加元素。如果该元素存在，则更新其序号。

语法：zadd 集合名 序号 元素







### （2）zrange

(把集合排序后,返回名次[start,stop]的元素

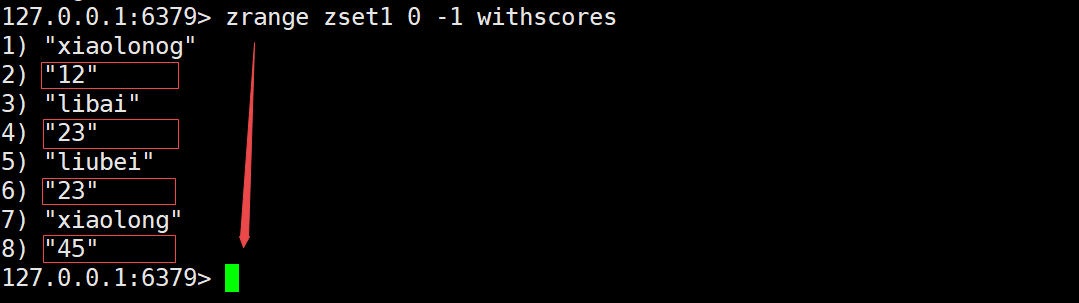
默认是升续排列

Withscores 是把score也打印出来

)

按序号升序获取有序集合中的内容，

语法：zrange 集合名称 开始下标 结束下标

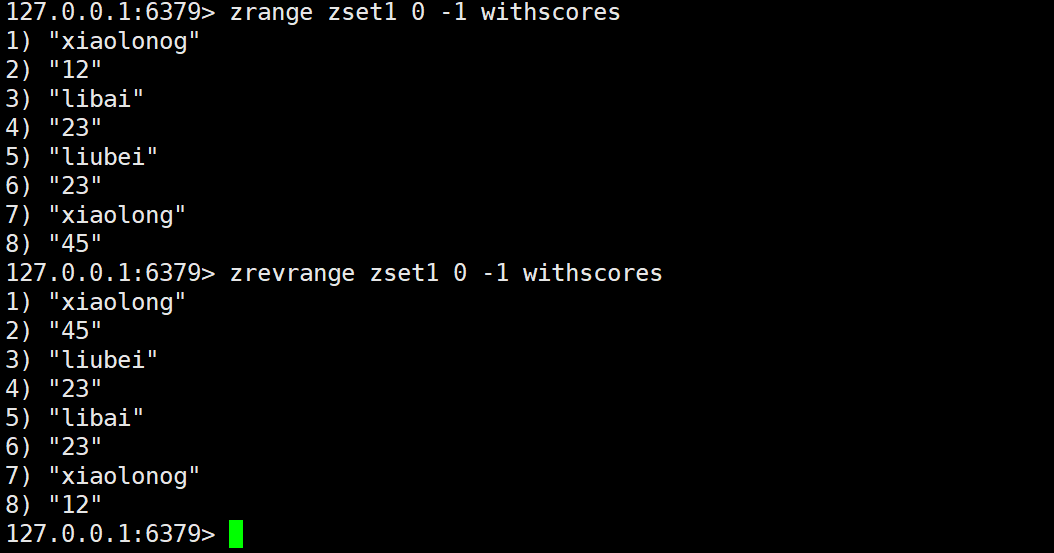


**注意：下标不是序号，是数据的索引**

### （3）zrevrange

按序号降序获取有序集合中的内容。

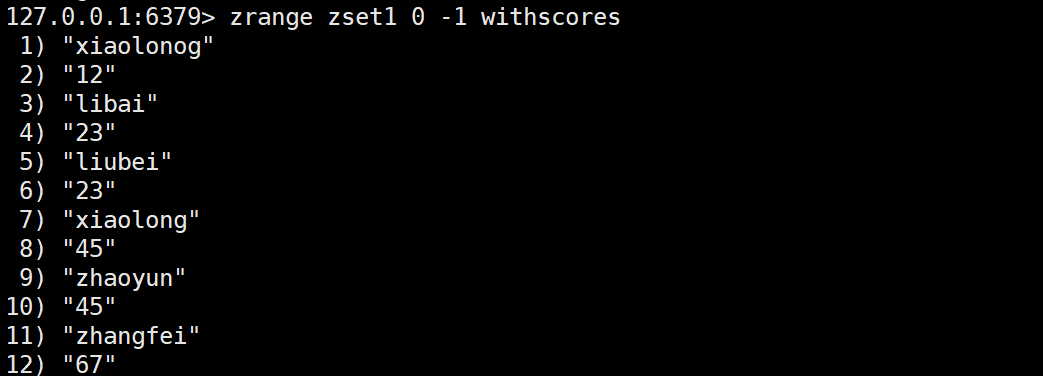
语法：zrevrange 集合名称 开始下标(索引) 结束下标（索引）

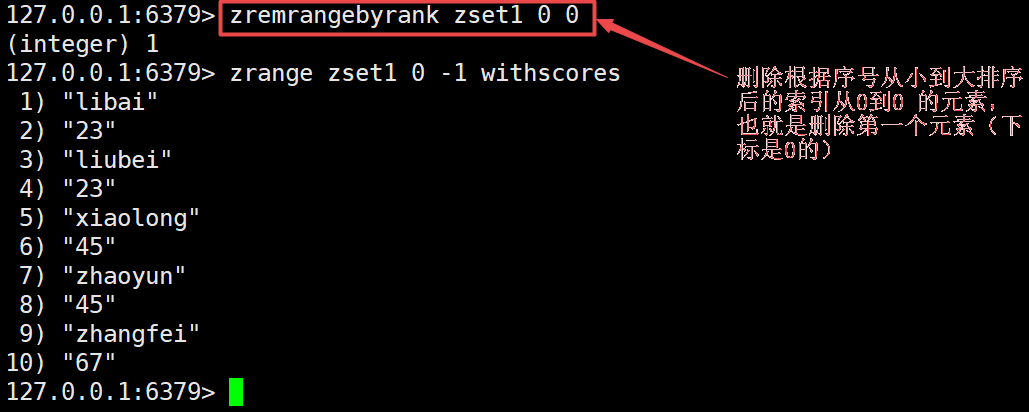


### （4）zremrangebyrank

zremrangebyrank 删除集合中排名在指定范围的元素（顺序从小到大排序）

语法：zremrangebyrank key min max

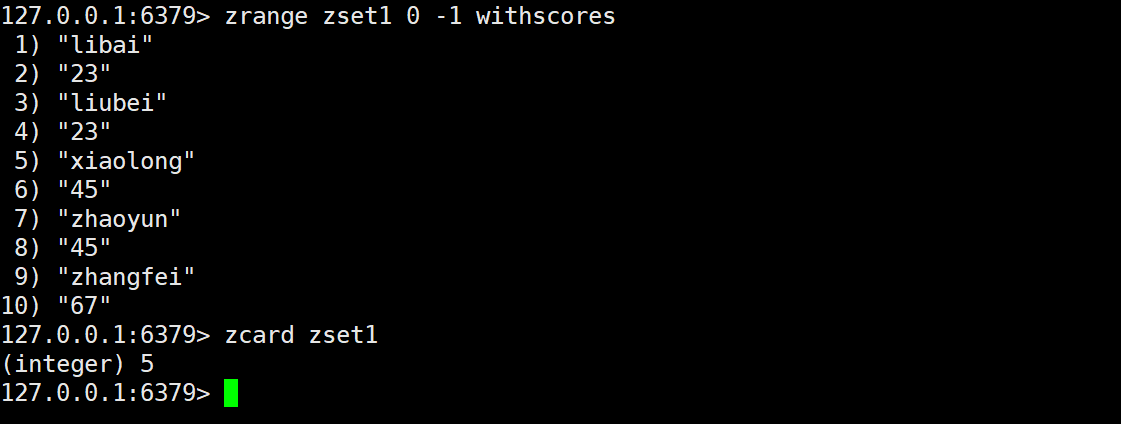




### （5）zcard

语法：zcard key

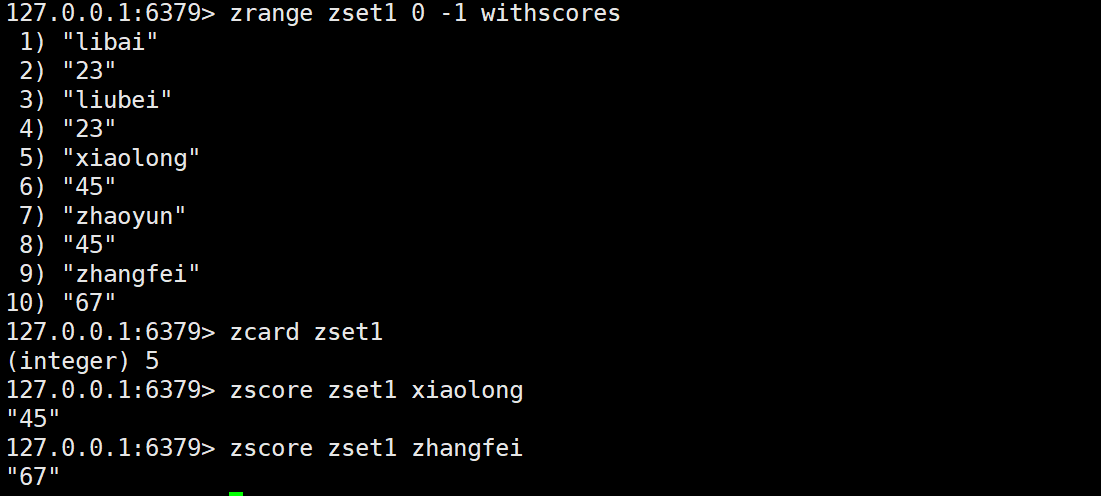
返回有序集合中元素的个数



### （6）zscore

语法：zscore key 元素

返回给定元素对应的score

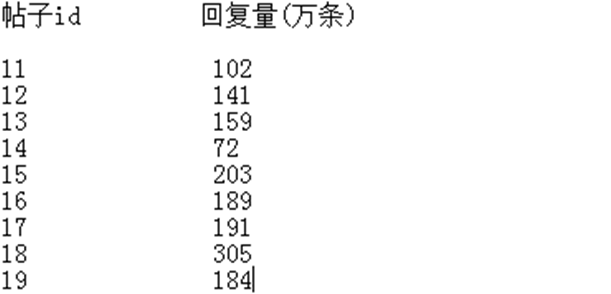


案例：利用sort set实现获取最热门的前5帖子信息

获得最热门(回复量)前5个帖子信息：

select \* from message order by backnum desc limit 5;

(以上需求可以通过简单sql语句实现，但是sql语句比较耗费mysql数据库资源)



每个帖子都有机会进入该“热门帖子集合”中，但是只保留回复量最高的5个帖子。

回复写入操作

用户

同步帖子/回复量信息

读取

排序集合中的每个元素都是值、权的组合

(之前的set集合类型每个元素就只是一个 值)

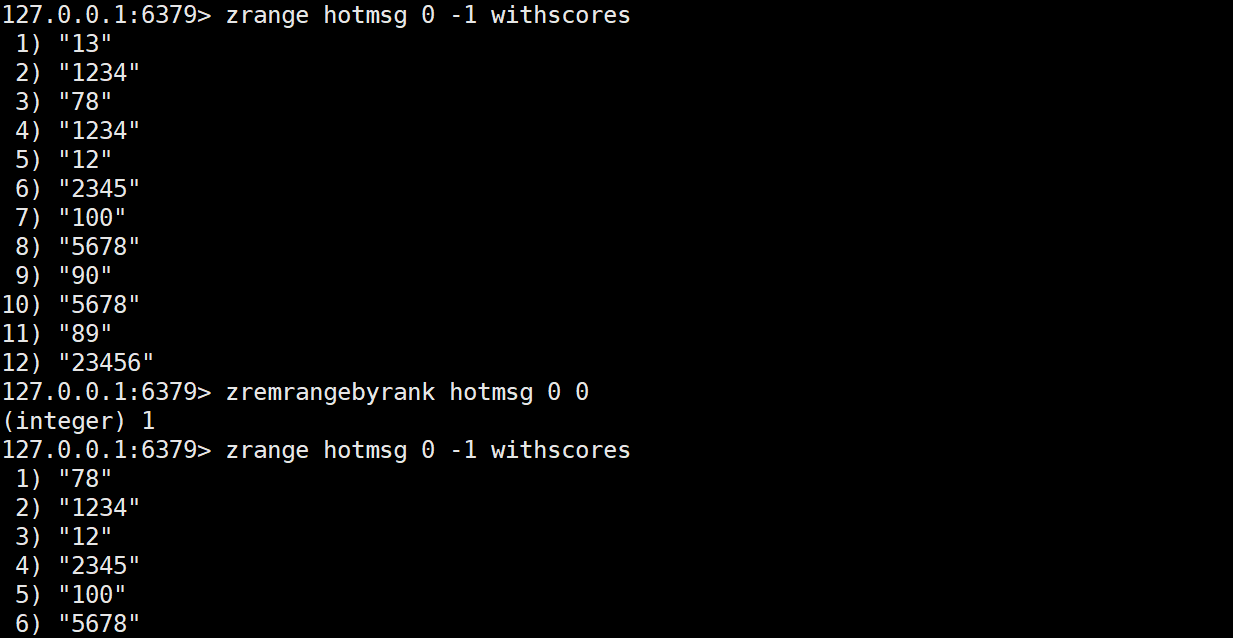
我们只做一个sort set排序集合，里边只保留5个元素信息，该5个元素是回复量最高的，

每个帖子被回复的时候，都有机会进入该集合里边，但是只有回复量最高的前5个帖子会存在于在集合，回复量低的就被删除。

第一步：创建一个hotmsg的排序集合



每增加一个新元素，就删除一个权值最小的旧元素，保留权值最高的5个元素。



# 二、主从复制

## 1、什么是主从复制；

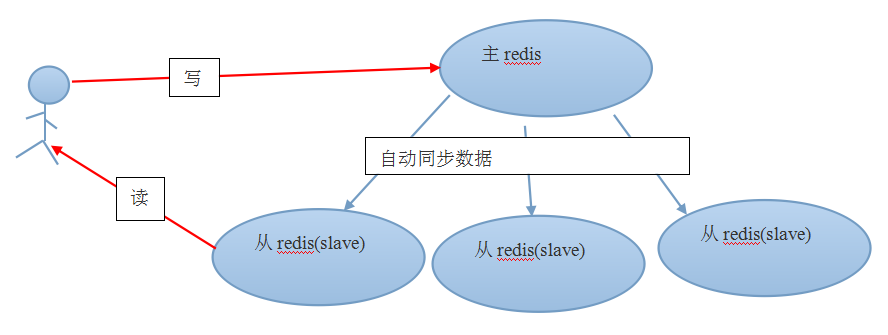
网站运行，mysql的写入、读取操作的sql语句比例：1:7

mysql为了降低每个服务器负载，可以设置读写分类(有写服务器、有读取服务器)

为了降低每个redis服务器的负载，可以多设置几个，并做主从模式

一个服务器负载“写”(添加、修改、删除)数据，其他服务器负载“读”数据

主服务器数据会“自动”同步给从服务器



Redis支持简单易用的主从复制（master-slave replication）功能，该功能可以让从服务器（slave server）成为主服务器（master server）的精确复制品。

主要作用：

主从备份，防止主服务器宕机；

读写分离，分担主服务器的任务；

任务分离，从服务器分别担任备份工作和计算工作。

注意点：

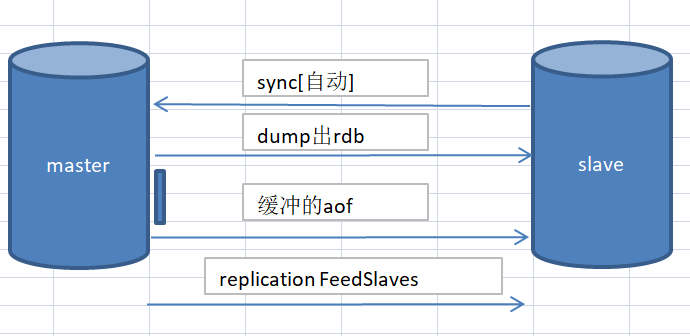
Redis使用异步复制，

一个主服务器可以有多个从服务器

不仅主服务器可以有从服务器，从服务器也可以有自己的从服务器，

复制功能不能回阻塞主、从服务器

## 2、主从通信过程



## 3、具体配置步骤

准备两台虚拟机；

名称php69 IP192.168.1.69 主服务器

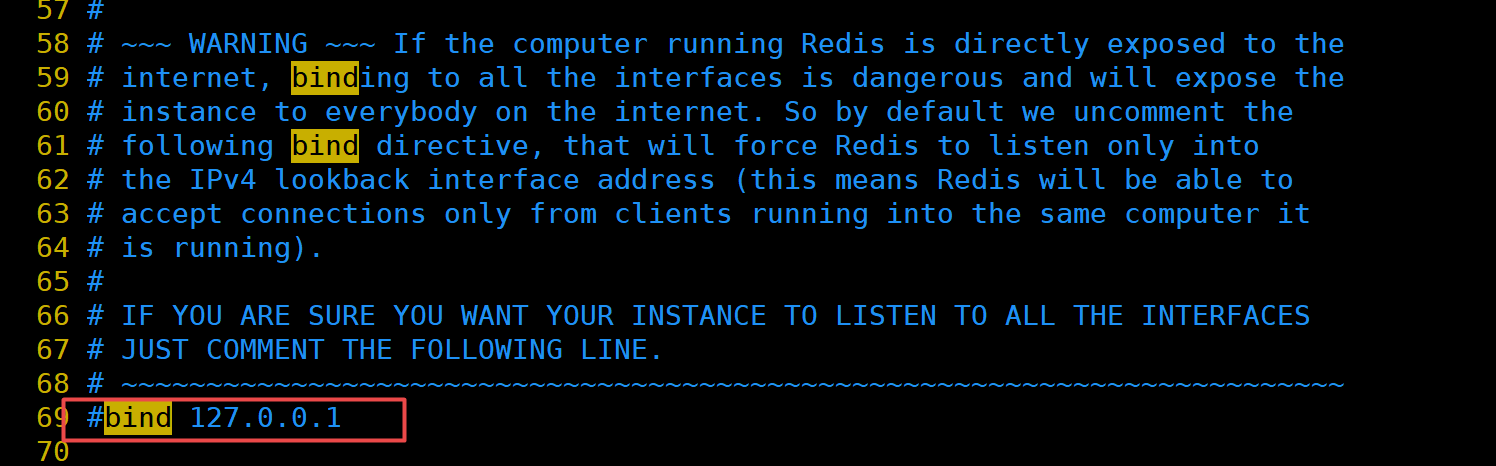
名称lnmp IP192.168.1.70 从服务器

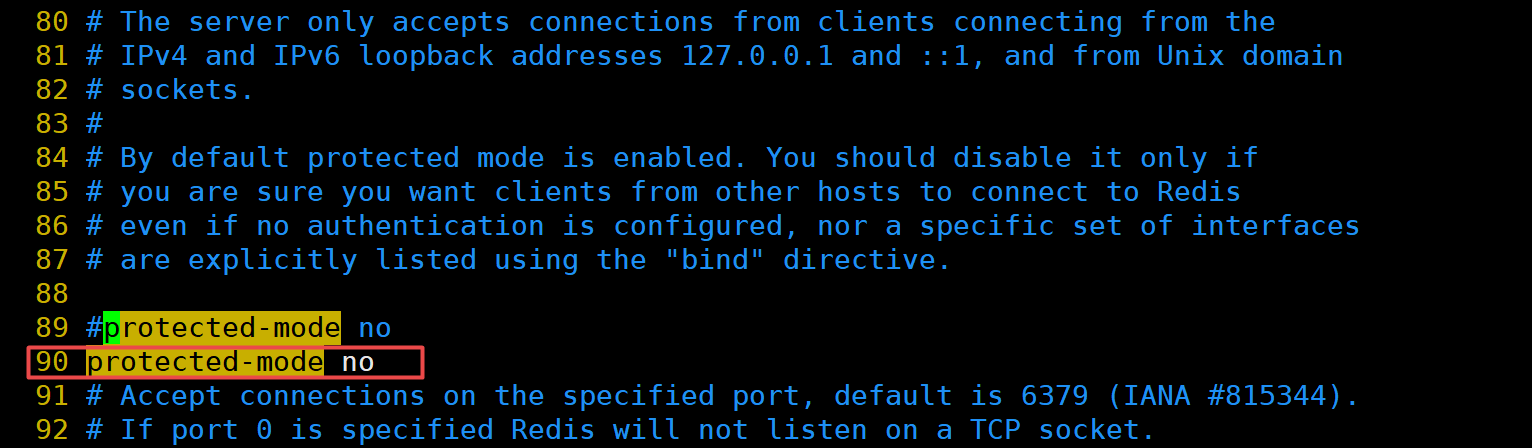
配置操作，

**主服务器配置**：

（1）bind 127.0.0.1改为 #bind 127.0.0.1

（2）protected-mode yes 改为 protected-mode no





**从服务器配置**

（1）通过slaveof指定自己的角色，主服务器的地址和IP

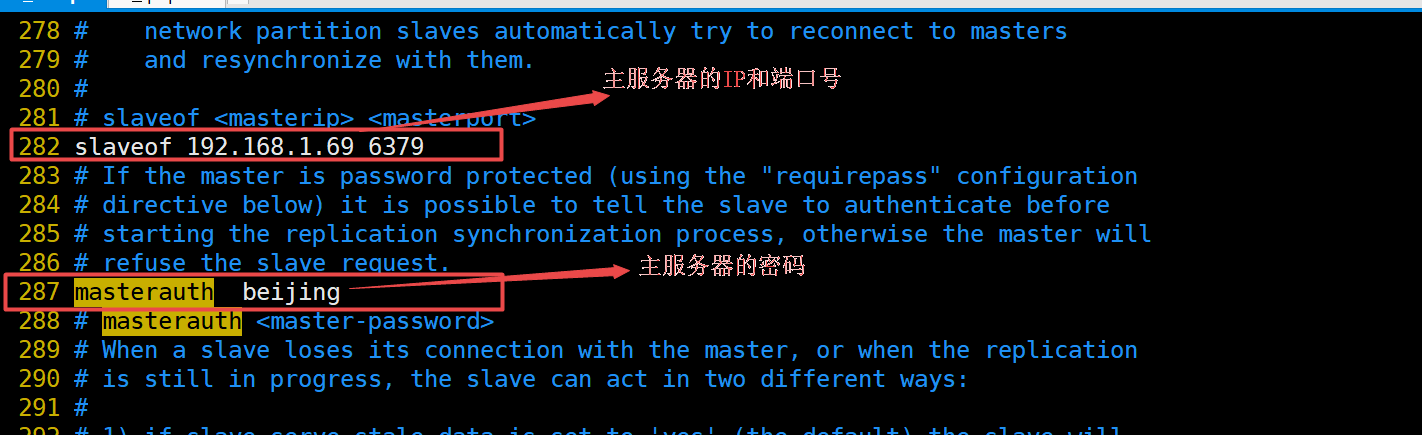
slaveof 主服务器ip 端口号

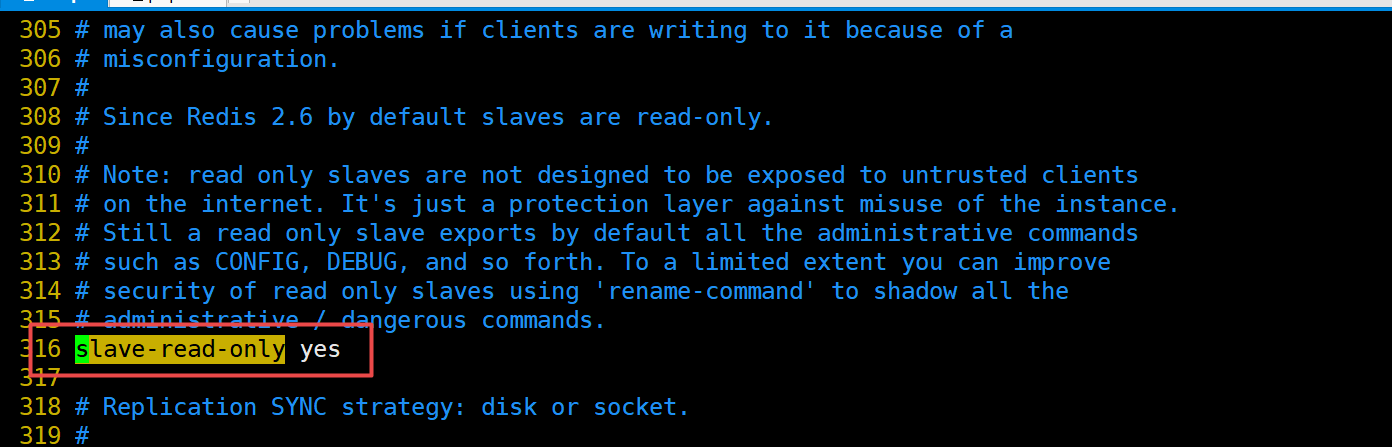
（2）从服务器只读

从redis2.6开始，从服务器支持只读模式，通过slave-read-only配置项配置，该模式为从服务器的默认模式

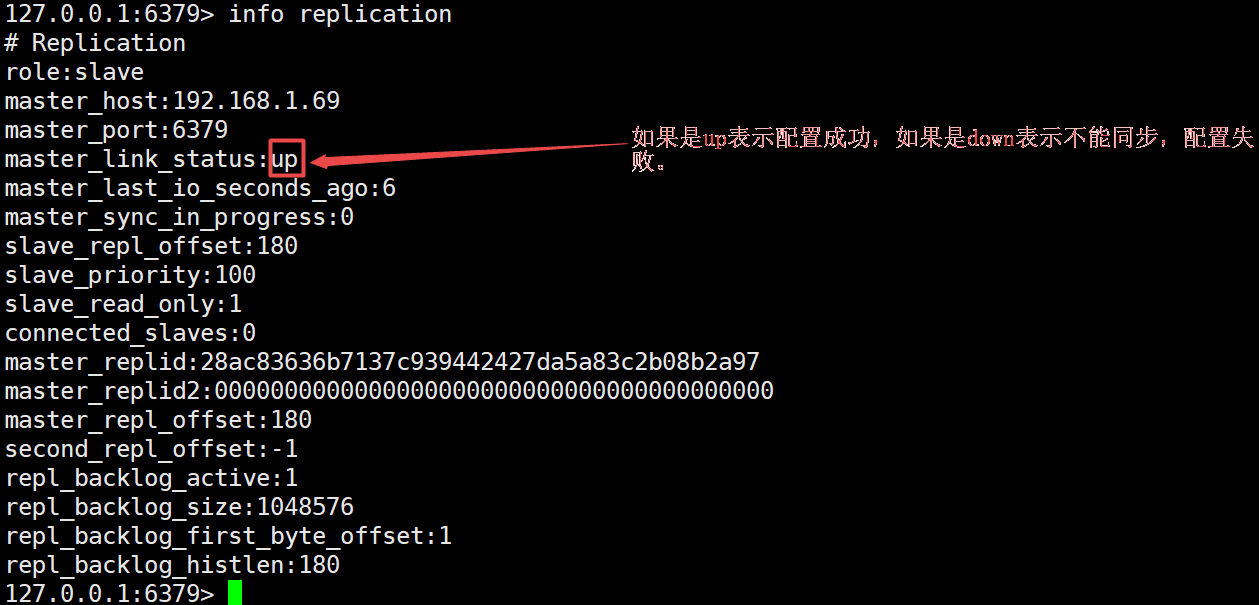
（3）指定从服务器连接主服务器的密码

如果主服务器通过requirepass选项设置了密码，为了让从服务器同步操作顺利进行，通过masterauth配置连接主服务器密码。





在从服务器执行info replication命令查看是否配置正确；



如果撤销，只需在从服务器里面屏蔽上面的配置即可。

## 4、redis主从复制的缺陷

每次slave服务器断开后，无论是主动断开，还是网络故障，再连接master，从服务器都要从master服务器全部dump出来rdb,再aof;即同步的过程都要重新执行一遍，所以要记住如果是多台从服务器时，不要一下子都启动起来。

# 三、Redis 中的事务

Redis支持简单的事务

Redis与 mysql事务的对比

|  |  |  |
| --- | --- | --- |
|  | Mysql | Redis |
| 开启 | start transaction | multi |
| 语句 | 普通sql | 普通命令 |
| 失败 | rollback 回滚 | discard 取消 |
| 成功 | commit | exec |

## 1、基本操作

set zhao 100

set wang 500

multi

decrby zhao 50

incrby wang 50

exec



## 2、discard取消事务

mget zhao wang

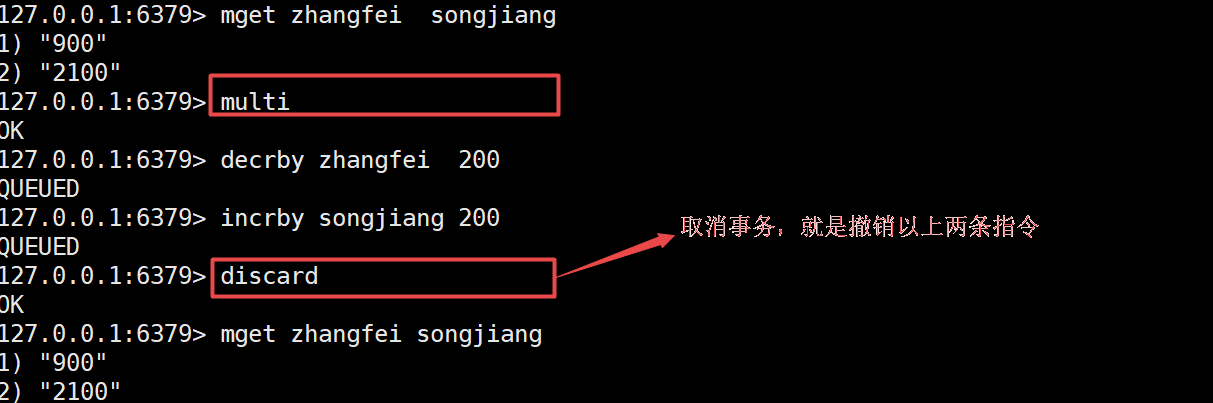
multi

incrby zhao 350

decrby wang 100

discard

mget zhao wang



## 3、其他说明

注:在multi后面的语句中, 语句出错可能有2种情况

（1）语法就有问题,

**这种,exec时,报错, 所有语句得不到执行**

mget zhao wang

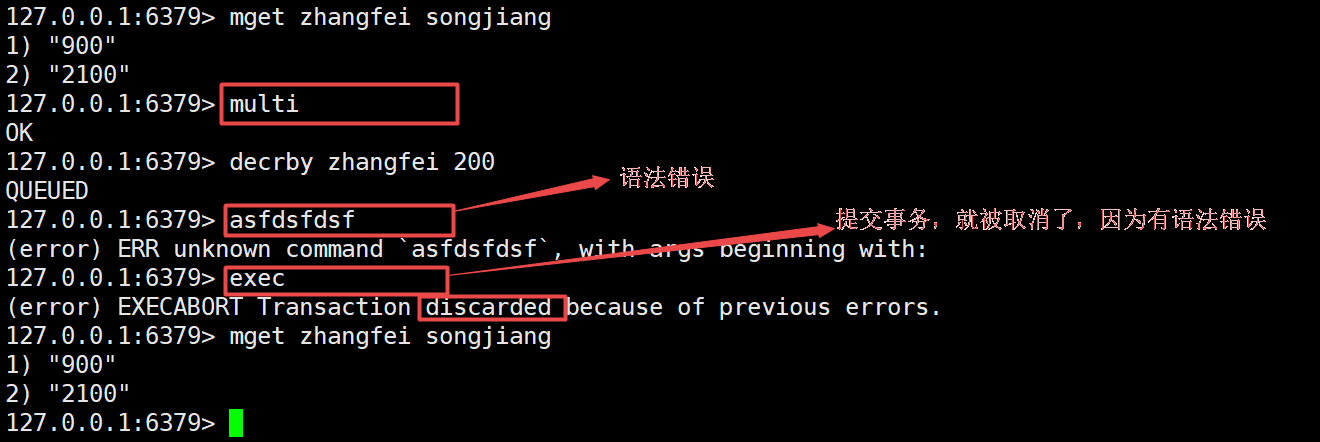
multi

incrby zhao 100

abc

exec

mget zhao wang



（2）语法本身没错,但适用对象有问题. 比如 zadd 操作list对象

Exec之后,会执行正确的语句,并跳过有不适当的语句.

mget zhao wang

multi

incrby zhao 100

sadd wang 200

exec



(如果zadd操作list这种事怎么避免? 这一点,由程序员负责)

## 4、watch命令

WATCH命令可以监控一个或多个键，一旦其中有一个键被修改（或删除），之后的事务就不会执行，监控一直持续到EXEC命令（事务中的命令是在EXEC之后才执行的，EXEC命令执行完之后被监控的键会自动被UNWATCH）

我正在买票

Ticket -1 , money -100

而票只有1张, 如果在我multi之后,和exec之前, 票被别人买了---即ticket变成0了.

我该如何观察这种情景,并不再提交

悲观的想法:

世界充满危险,肯定有人和我抢, 给 ticket上锁, 只有我能操作. [悲观锁]

乐观的想法:

没有那么人和我抢,因此,我只需要注意,

--有没有人更改ticket的值就可以了 [乐观锁]

Redis的事务中,启用的是乐观锁,只负责监测key没有被改动.

set ticket 1

set money 100

multi

decr ticket

decrby money 100

exec



具体的命令---- watch命令

例:

redis 127.0.0.1:6379> watch ticket

OK

redis 127.0.0.1:6379> multi

OK

redis 127.0.0.1:6379> decr ticket

QUEUED

redis 127.0.0.1:6379> decrby money 100

QUEUED

redis 127.0.0.1:6379> exec

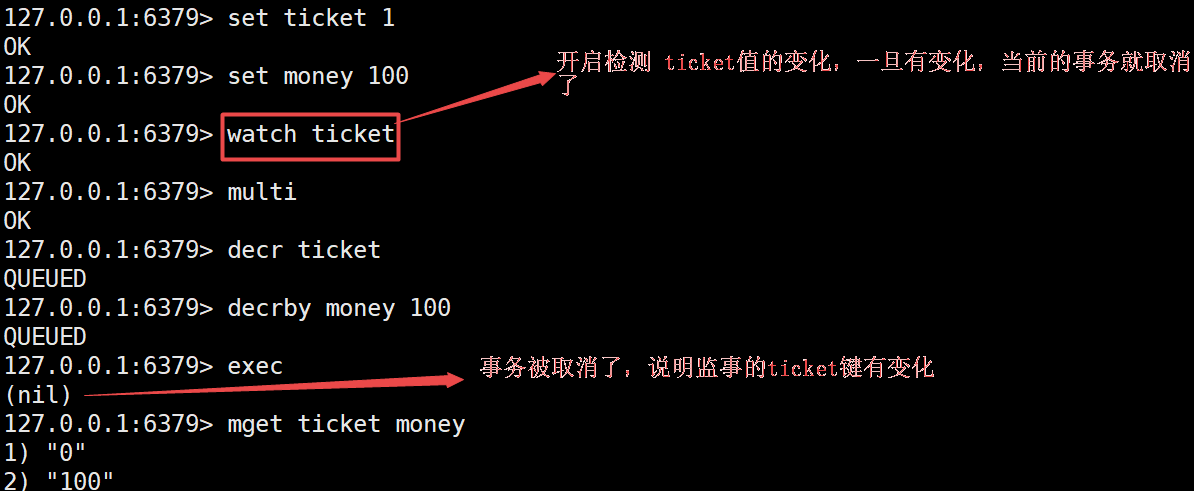
(nil) // 返回nil,说明监视的ticket已经改变了,事务就取消了.

redis 127.0.0.1:6379> get ticket

"0"

redis 127.0.0.1:6379> get money

"200"



watch key1 key2 ... keyN

作用:监听key1 key2..keyN有没有变化,如果有变, 则事务取消

unwatch

作用: 取消所有watch监听

# 四、电商秒杀案例

**实现原理：使用redis链表中队列，进行pop操作，因为pop操作是原子的，即使有很多用户同时到达，也是依次执行，推荐使用。**

**第一步：先将商品库存如队列**

<?php

$store=1000;

$redis=new Redis();

$result=$redis->connect('127.0.0.1',6379);

$goods\_number = 100;

for($i=0;$i<$goods\_number;$i++){

$redis->lpush('goods\_store',1);

}

echo $redis->llen('goods\_store');

?>

第二步：抢购开始，设置库存的缓存周期

$redis->setTimeout(‘goods\_store’,60);

第三步：客户端执行下单操作，下单前判断redis队列库存量

$redis=new Redis();

$redis->connect('127.0.0.1',6379);

$count=$redis->lpop('goods\_store');

if(!$count){

echo ‘抢购失败’;

return;

}

//跳转到下单页面，完成下单操作



# 五、综合案例

## 1、用户注册

添加数据里面，涉及到三种类型，

字符串类型，获取id, 存储id,登录时用

哈希类型，存储具体的数据的

链表类型，存储添加用户的id的；

register.html和register.php页面

## 2、用户列表

list.php页面；

## 3、用户修改

update.php和updateok.php页面

## 4、用户删除

删除要删除 哈希类型 链表里面的元素

## 5、用户登录

用户登录，获取用户名和密码，

用户信息是存储到哈希里面，如何获取用户信息？

## 6、添加关注