# Redis简介：基本数据类型以及应用场景



 Redis 专栏收录该内容
5 篇文章 1 订阅

订阅专栏
本文使用的工具是：redis-desktop-manager

具体地址是：https://download.csdn.net/download/fenghuoliuxing990124/10848838

> 1.1：Redis数据库基本概念

1，redis也有数据库的概念，一个数据库中可以保存一组数据；

2，各个数据库之间是相互隔离的，当然也可以在不同数据库之间复制数据；

3，每一个数据库都有一个id号，默认的数据库id为0；

4，可以使用select命令选择当前使用的数据库：

5，redis初始化的时候会默认创建16个数据库（这个配置可以在redis配置文件中databases 16）；

6，特别注意，类似redis的key-value数据库系统，是绝对没有表的概念，可以简单理解为，所有的数据都是乱七八糟的堆在一起的；

> :1.2：Redis的命令组

Redis命令十分丰富，包括的命令组有Cluster、Connection、Geo、Hashes、HyperLogLog、Keys、Lists、Pub/Sub、Scripting、Server、Sets、Sorted Sets、Strings、Transactions一共14个redis命令组两百多个redis命令。每个命令都代表着一种操作，有点类似mysql的命令。

这些命令组可以按功能分为以下几种类型：
1，对数据的操作：
Strings、Lists、Sets、SortedSets、Hashes、Geo

其中Geo可以做附近的人的功能

2，发布/订阅相关操作：
Pub/Sub

3，事务控制：
Transactions

4，脚本命令；
Scripting

5，网络连接命令；
Connection

6，数据库服务相关命令；
Keys
Server
Cluster
HyperLogLog

> :1.3：Redis数据类型：key & value

1，key用来标记一个数据；一般在key中需要体现出数据模型的结构。

比如最简单的模型：

```
key:1

value:{
    "id":1,
    "name":"swk"
}

    //但是上面的key容易产生key覆盖的问题，因为假如多个对象都是用ID做key
    //所以一般都是这样做
    key:
    user:1
    employee:1

    value:
    employee:{
            "id":1,
            "deptId":1,
            "name":"mgr"
    }
    user:{
            "id":1,
            "name":"swk"
    }
```

2，value表示一个key对应的值；在redis中，value可以是任何内容，redis把所有的value都作为byte处理；所以可以用来**保存任何内容**；

3，redis最突出的特点是提供了5种常用的数据存储类型（value的类型），深刻理解这5中数据结构和各自的使用场景，对redis的使用有很大帮助；

> :1.4：Redis中的查询

1，在redis中，不支持对value进行任何形式的查询；

```
catalog:1
catalog:2
//想要查找catalog的话
keys catalog:*
```

- 1
- 2
- 3
- 4

2，redis**不是一个适用于任何场景**的存储方案，考虑使用redis需要对业务进行考评，用redis的思想去重新设计数据结构；

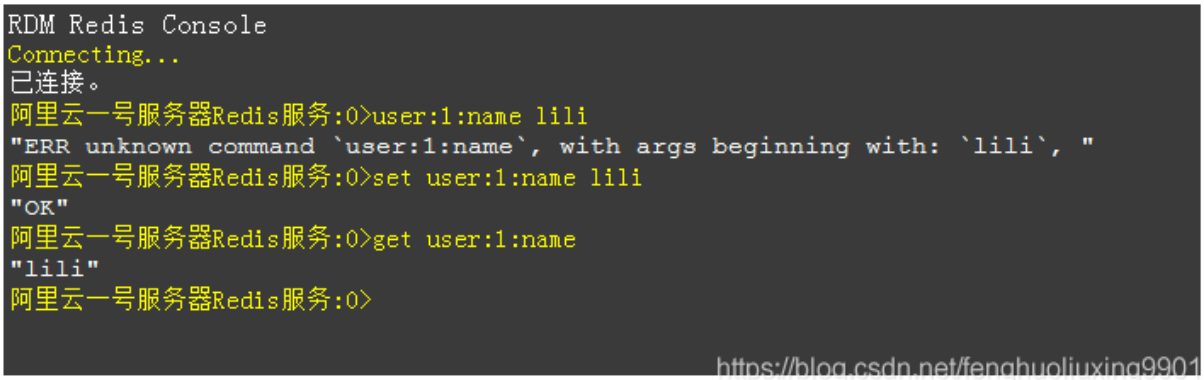
> 2,1 Redis的基本数据类型：String

1，redis中最常见的数据类型，内容可以是任何值；

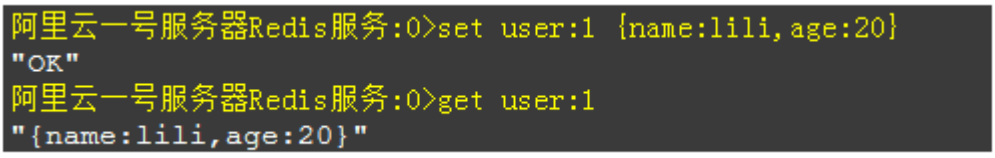
2，常用的字符串操作：

1）set key value：设置一个值；

2）get key：返回key对应的value；




并且这里还可以对user:1 设置额外的属性，比如以一个JSON对象去描述它




3）strlen key：返回key对应的value字符串长度；

4）append key value：给key对应的value追加值，如果key不存在，相当于set一个新的值；

```
阿里云一号服务器Redis服务:0>strlen user:1:name
"4"
阿里云一号服务器Redis服务:0>append user:1:name " hello"
"10"
阿里云一号服务器Redis服务:0>
```

注意：如果我们想要输入空格字符，那么只是敲一个空格是无法生效的，必须用双引号包一个空格字符的方式来引入空格

5）getrange key start stop：返回key对应value的一个子字符串，位置从start到stop；

```
阿里云一号服务器Redis服务:0>getrange user:1:name 0 4
"lili "
阿里云一号服务器Redis服务:0>getrange user:1:name 0 3
"lili"
阿里云一号服务器Redis服务:0>
```

与Java不同的是Redis中的区间通常是：前闭后闭

而Java中的区间通常是：前闭后开的

3，如果字符串的内容是数值（integer，在redis中，数值也是string），那么有如下常用的操作：

1）incr key：在给定key的value上增加1，并返回增加后的值，redis中的incr是一个原子操作，支持并发；

2）incrby key value：给定key的value上增加value值，并返回增加后的值，相当于key=key.value+value；这也是一个原子操作；

```
阿里云一号服务器Redis服务:0>incr user:1:age
"21"
阿里云一号服务器Redis服务:0>incrby user:1:age 10
"31"
```

3）decr：在给定key的value上减少1；

4）decrby key value：给定key的value上减少value值；

```
阿里云一号服务器Redis服务:0>decr user:1:age
"30"
阿里云一号服务器Redis服务:0>decrby user:1:age 10
"20"
```

### 2.1.1 String的应用场景

1、缓存功能：字符串最经典的使用场景，redis最为缓存层，Mysql作为储存层，绝大部分请求数据都是在redis中操作，由于redis具有支撑高并发特性，所以缓存通常能起到加速读写和降低 后端压力的作用。

2、计数器：许多运用都会使用redis作为计数的基础工具，他可以实现快速计数、查询缓存的功能。

如：视频播放数系统就是使用redis作为视频播放数计数的基础组件。

比如：优酷视频的播放：incr video:videoId:playTimes

或者：文章浏览量：incr article:aricleId:clickTimes

或者粉丝数量：取关 decr author:authorId:fansNumber

这样的话就能极快的提高视频或者是文章的**访问速度**

简单总结就是：能够利用redis在缓存中做统计的工作，省去了在sql联表查询的功夫

3、id生成器：我们在使用mysql时，把数据存在mysql后，该数据就会自动生成一个自增长的id，这个id肯定是不重复的，那类似这种生成一个不重复的id也可以用redis的string数据结构来做。

为user的模型设计一个id生成器

```
incr user:id:generate
     1
```

那么就可以利用该ID生成器做ID的自动生成

```
阿里云一号服务器Redis服务:0>incr user:id:generate
"1"
阿里云一号服务器Redis服务:0>incr user:id:generate
"2"
```

拿到的ID值，就可以赋予下面的对象

```
user:1{
        id:1,
        name:lili
}
```

- 1
- 2
- 3
- 4

### 2,2 Redis的基本数据类型：List

1、Redis中的List类似Java中的queue,也可以当做List来用.

List类型是一个链表结构的集合,其主要功能有push,pop,获取元素等.更详细的说,List类型是一个**双端链表**的结构,我们可以通过相关操作进行集合的头部或者尾部添加删除元素,list的设计非常简单精巧,即可以作为栈,又可以作为队列.满足绝大多数需求.

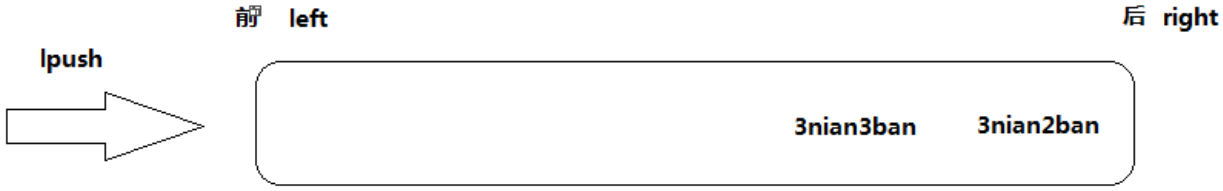
2、常用的list操作：

1）lpush key value...：在一个list最前面添加一个或多个元素。

```
阿里云一号服务器Redis服务:0>lpush class 3nian2ban 3nian3ban
"2"
阿里云一号服务器Redis服务:0>
```

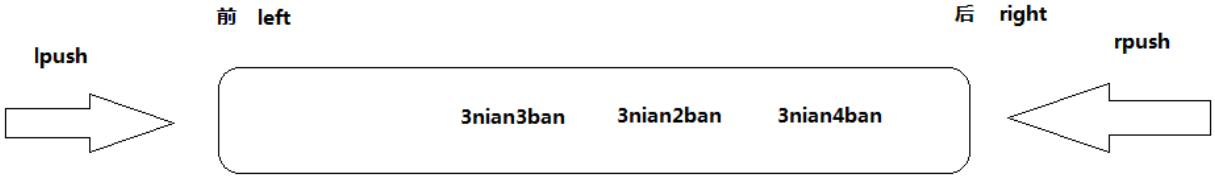
其具体的数据模型是：

由于redis中的List是个双端链表，所以插入数据后就变成了这样




2）rpush key value...：在一个list最后面添加一个或多个元素。

```
阿里云一号服务器Redis服务:0>rpush class 3nian4ban
"3"
阿里云一号服务器Redis服务:0>
```




3）llen key：返回指定list的元素个数。

4）lrange key start stop：获取一个list中的指定元素(如果是0，-1则表示从头取到末尾所有元素)

```
阿里云一号服务器Redis服务:0>llen class
"3"
阿里云一号服务器Redis服务:0>lrange class 0 -1
 1)  "3nian3ban"
 2)  "3nian2ban"
 3)  "3nian4ban"
阿里云一号服务器Redis服务:0>
```

这里有个特殊的语法:lrange class 0 -1 表示显示list中的所有内容

如果要开查看指定位置的元素

```
阿里云一号服务器Redis服务:0>lrange class 0 -1
 1)  "3nian3ban"
 2)  "3nian2ban"
 3)  "3nian4ban"
阿里云一号服务器Redis服务:0>lrange class 0 1
 1)  "3nian3ban"
 2)  "3nian2ban"
阿里云一号服务器Redis服务:0>lrange class 1 2
 1)  "3nian2ban"
 2)  "3nian4ban"
阿里云一号服务器Redis服务:0>https://blog.csdn.net/fenghuoliuxing990124
```

5）ltrim key start stop：裁剪指定list，剩下的内容从start到stop。

```
阿里云一号服务器Redis服务:0>ltrim class 0 1
"OK"
阿里云一号服务器Redis服务:0>lrange class 0 -1
 1)  "3nian3ban"
 2)  "3nian2ban"
阿里云一号服务器Redis服务:0>
```

这里相当于对list进行剪裁，剪裁范围以外的元素丢弃

6）lpop key ：从list最前面弹出一个元素。

```
阿里云一号服务器Redis服务:0>rpush class 3nian4ban

"3"
阿里云一号服务器Redis服务:0>lrange class 0 -1

 1)  "3nian3ban"

 2)  "3nian2ban"
 3)  "3nian4ban"
阿里云一号服务器Redis服务:0>lpop class

"3nian3ban"
阿里云一号服务器Redis服务:0>lrange class 0 -1

 1)  "3nian2ban"

 2)  "3nian4ban"
阿里云一号服务器Redis服务:0>
```



7）rpop key：从list最后面弹出一个元素。

```
阿里云一号服务器Redis服务:0>rpop class

"3nian4ban"
阿里云一号服务器Redis服务:0>lrange class 0 -1

 1)  "3nian2ban"
阿里云一号服务器Redis服务:0>
```

8）lindex key index：从list中获取索引为index的元素

比如：下面查看索引位置为0的元素

```
阿里云一号服务器Redis服务:0>lindex class 0

"3nian2ban"
阿里云一号服务器Redis服务:0>
```

List应用场景：
一个List很典型的应用场景：社区应用
社区应用是一个很广泛的概念，我们平时用的微博，朋友圈，博客，论坛都算是社区应用，社区应用无非就是一些帖子或状态，然后可以给这些帖子或状态回贴或评论，还有就是可以点赞。

> 2,2.1 Redis的基本数据类型：List的应用场景

示例：1，点赞；

1，创建一条微博内容：set user:1:post:91 “hello world”;
2，点赞：
lpush post:91:good “kobe.png”

lpush post:91:good “jordan.png”
lpush post:91:good “James.png”

3，查看有多少人点赞: llen post:91:good
4，查看哪些人点赞：lrange post:91:good 0 -1

```
阿里云一号服务器Redis服务:0>set user:1:post:91 "helloworld"

"OK"
阿里云一号服务器Redis服务:0>lpush post:91:good "swk.png"

"1"
阿里云一号服务器Redis服务:0>lpush post:91:good "zbj.png"

"2"
阿里云一号服务器Redis服务:0>lpush post:91:good "swj.png"

"3"
```

```
阿里云一号服务器Redis服务:0>llen post:91:good

"3"
阿里云一号服务器Redis服务:0>lrange post:91:good 0 -1

 1)  "swj.png"

 2)  "zbj.png"
 3)  "swk.png"
```



思考，如果用数据库实现这个功能，SQL会多复杂？？

示例2：回复

1，创建一条微博内容：set user:1:post:92 “oh my lady gaga”
2，回复：lpush post:92:reply “heihei”
lpush post:92:reply “enen”
5，查询微博的回复：lrange post:92:reply 0 -1

```
阿里云一号服务器Redis服务:0>set user:1:post:95 "dfadsfd"

"OK"
阿里云一号服务器Redis服务:0>lpush post:95:reply "请不要灌水 by 管理员"

"1"
阿里云一号服务器Redis服务:0>lpush post:95:reply "小手一抖，经验到手，斜眼"

"2"
阿里云一号服务器Redis服务:0>
```

```
阿里云一号服务器Redis服务:0>lrange post:95:reply 0 -1
 1)  "小手一抖，经验到手，斜眼"
 2)  "请不要灌水 by 管理员"
阿里云一号服务器Redis服务:0>
```

> 2,3 Redis的基本数据类型：Set

1，set结构和java中差不多，数据没有顺序，并且每一个值不能重复；

2，set结构的常见操作：

1）sadd key member...：给set添加一个或多个元素

2）scard key：返回set的元素个数

```
阿里云一号服务器Redis服务:0>sadd language java js h5
"3"
阿里云一号服务器Redis服务:0>scard language
"3"
阿里云一号服务器Redis服务:0>
```

3）smembers key：返回指定set内所有的元素，以一个list形式返回

4）srem key member：从set中移除一个给定元素

```
阿里云一号服务器Redis服务:0>smembers language
 1)  "js"
 2)  "java"
 3)  "h5"
阿里云一号服务器Redis服务:0>srem language "h5"
"1"
```

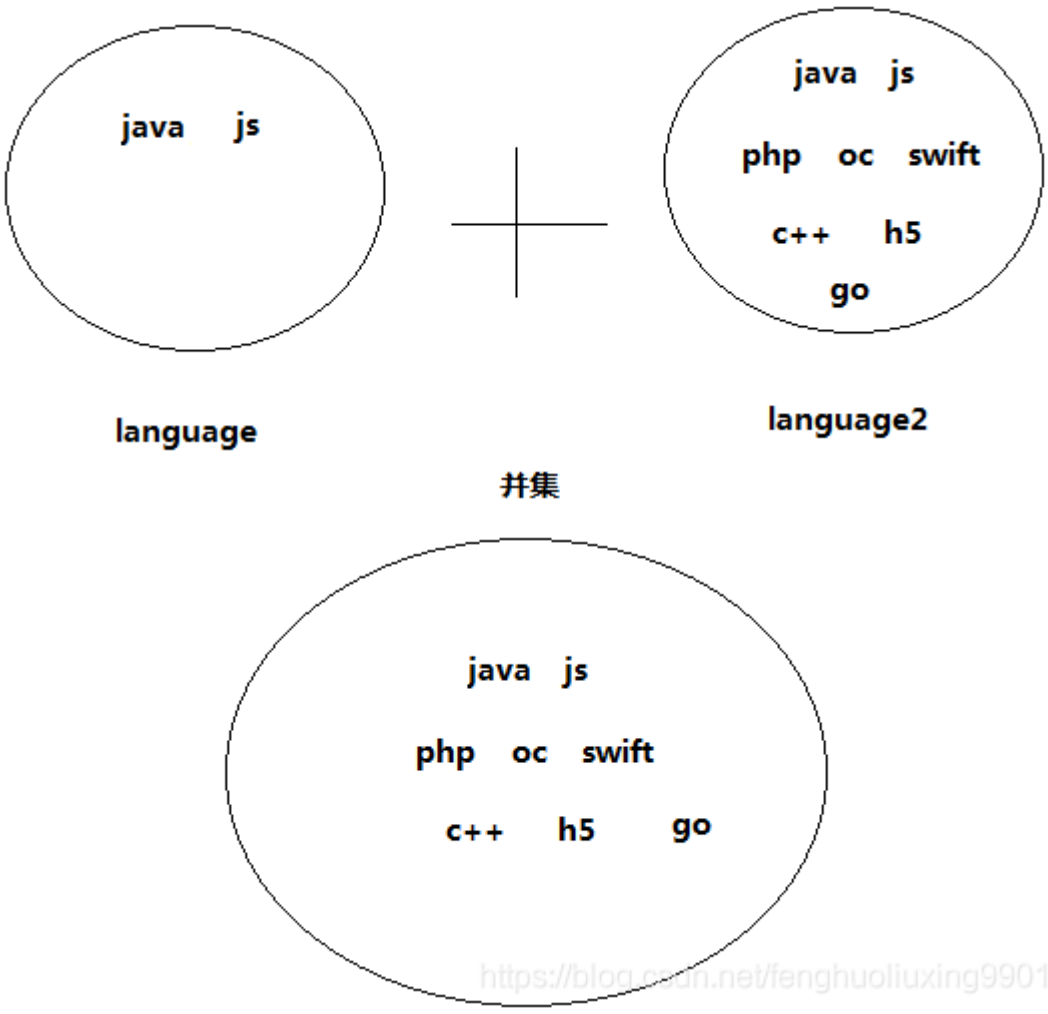
5）sismember key member：判断给定的一个元素是否在set中，如果存在，返回1，如果不存在，返回0

```
阿里云一号服务器Redis服务:0>sismember language "h5"
"0"
阿里云一号服务器Redis服务:0>sismember language "java"
"1"
```

6）srandmember key count:返回指定set中随机的count个元素

```
阿里云一号服务器Redis服务:0>srandmember language 1

 1)  "js"
阿里云一号服务器Redis服务:0>srandmember language 1

 1)  "js"
阿里云一号服务器Redis服务:0>srandmember language 1

 1)  "java"
```

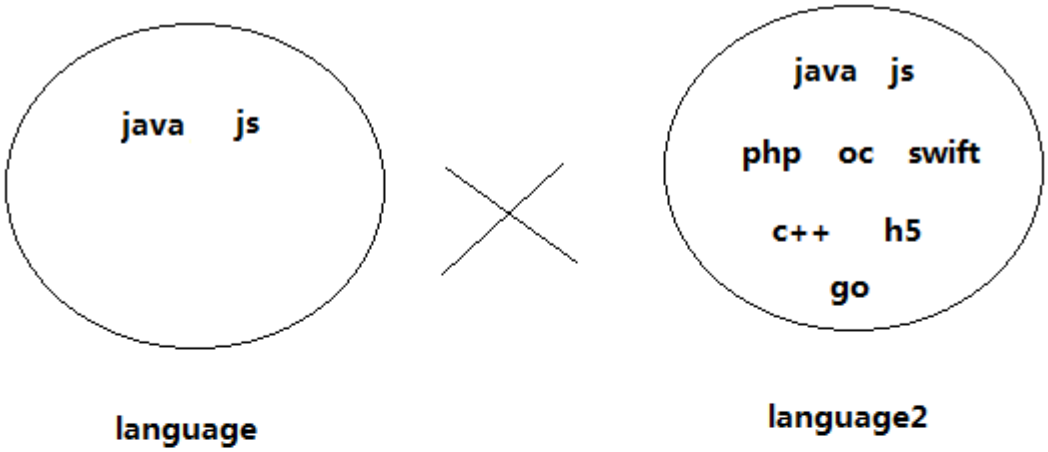
7）sunion key key...：用key和key...做并集，结果返回一个list;



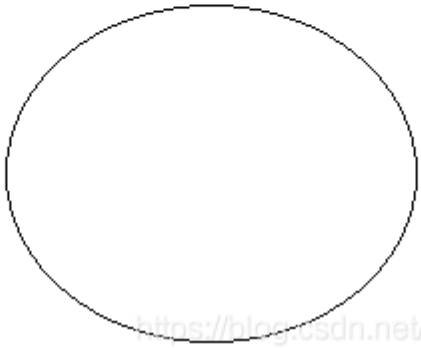



```
阿里云一号服务器Redis服务:0>sunion language language2

 1)  "go"

 2)  "swift"
 3)  "js"
 4)  "c++"
 5)  "oc"
 6)  "java"
 7)  "php"
 8)  "h5"
阿里云一号服务器Redis服务:0>
```

8）sdiff key key...：用key和key...做差集，结果返回一个list;




以language为主减language





此时结果为空集，假如在 sadd langugage c

那么最后的差集会是 c

9）sinter key key ... ：用key和key...做交集，结果返回一list；

```
阿里云一号服务器Redis服务:0>sinter language language2
 1)  "js"
 2)  "java"
阿里云一号服务器Redis服务:0>
```

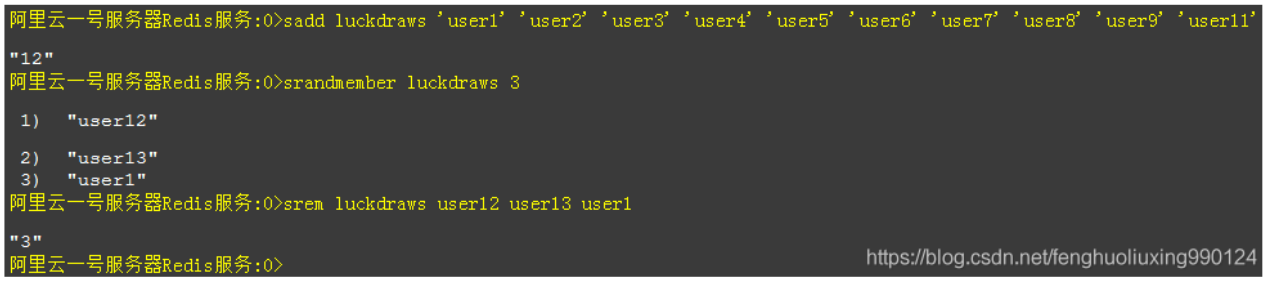
10）sunionstore destination key key ... ：获取多个key对应的set之间的并集，并保存为新的key值；目标值也是一个set；

set的使用场景：
1，去重；

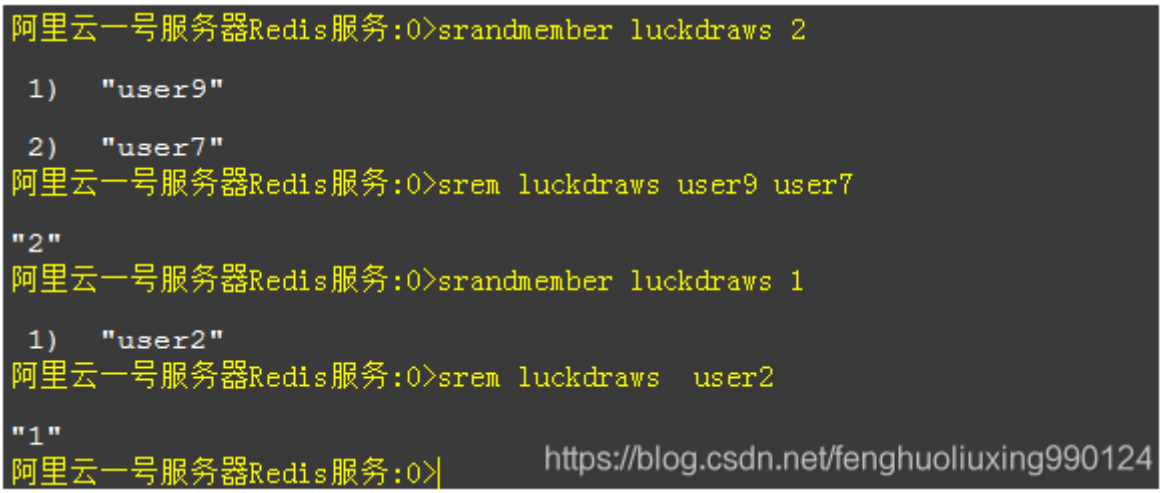
2，抽奖

1）初始化用户池：

sadd luckdraws ‘user1’ ‘user2’ ‘user3’ ‘user4’ ‘user5’ ‘user6’ ‘user7’ ‘user8’ ‘user9’ ‘user11’ ‘user12’ ‘user13’
2）抽3个三等奖：
srandmember luckdraws 3
srem luckdraws user...




3）抽2个二等奖：
srandmember luckdraws 2
srem luckdraws user...
4）抽1个一等奖：
srandmember luckdraws 1




3，做set运算（二度好友推荐）
1）初始化好友圈：
sadd user:1:friends ‘user:2’ ‘user:3’ ‘user:5’
sadd user:2:friends ‘user:1’ ‘user:3’ ‘user:6’
sadd user:3:friends ‘user:1’ ‘user:7’ ‘user:8’
2）给user1推荐你可能认识的好友：
a、用user1的好友的好友做并集
sunionstore user:1:groups user:2:friends user:3:friends user:5:friends
b、用user:1:group和user:1的好友做差集
sdiffstore user:1:groups user:1:groups user:1:friends
c、去除自己
srem user:1:groups user:1
3）在user:1:groups中随机推荐指定的好友数
srandmember user:1:groups 2

除此之外，set数据结构还可以用来做共同关注，共同爱好好友推荐等等...

> 2,4 Redis的基本数据类型：SortedSet

1，set是一种非常方便的结构，但是**数据无序**，redis提供了一个sorted set，每一个**添加的值都有一个对应的分数**，放进去的值按照该**分数升序**存在一个集合中，可以通过这个分数进行**相关排序的操作**。

**好处：可以避免在SQL中使用Order by语句（十分低效）**

2，SortedSet的常用操作：

1）zadd key score member：添加一个带分数的元素，也可以同时添加多个.

2） zcount key min max ：给定范围分数的元素个数：

可见guys04 40分最后一名

```
阿里云一号服务器Redis服务:0>zadd test 90 guys01 80 guys02 70 guys03  40 guys04
"4"
阿里云一号服务器Redis服务:0>zcount test 60 90
"3"
```

3） zrank key member ：查询指定元素的分数在整个列表中的排名（从0开始）

4）zrange key start stop：获取集合中指定范围的元素，按分数升序排序输出

```
阿里云一号服务器Redis服务:0>zrank test guys04
"0"
阿里云一号服务器Redis服务:0>zrange test 70 90

阿里云一号服务器Redis服务:0>zrange test 0 1
 1)  "guys04"
 2)  "guys03"
阿里云一号服务器Redis服务:0>
```

5）zrevrange key start stop：获取集合中指定范围的元素，按分数降序排序输出

```
阿里云一号服务器Redis服务:0>zrange test 0 1
 1)  "guys04"
 2)  "guys03"
阿里云一号服务器Redis服务:0>zrevrange test 0 1
 1)  "guys01"
 2)  "guys02"
```

查询排名：zrank key member

```
阿里云一号服务器Redis服务:0>zrevrank test guys01

"0"
阿里云一号服务器Redis服务:0>zrevrank test guys04

"3"
阿里云一号服务器Redis服务:0>
```

6）zincrby key score member：给指定的元素增加指定的分数

比如：guys04入侵了后台系统，修改了分数，那么此时它的排名为：

```
阿里云一号服务器Redis服务:0>zincrby test 60 guys04

"100"
阿里云一号服务器Redis服务:0>zrevrank test guys04

"0"
阿里云一号服务器Redis服务:0>
```

7）zrem key member：移除指定的元素

很可惜，guys04作弊被在他身后的管理员&老师发现，于是他的成绩被踢出了排行

```
阿里云一号服务器Redis服务:0>zrem test guys04

"1"
阿里云一号服务器Redis服务:0>zrevrange test 0 -1

 1)  "guys01"

 2)  "guys02"
 3)  "guys03"
阿里云一号服务器Redis服务:0>
```

> SortedSet 应用场景

排行榜：有序集合经典使用场景。例如社交网站需要对用户的帖子或者微博做排行榜，榜单维护可能是多方面：
按照点击量、按照获得的赞数，安照评论数等等。

按照点击量排行：
1、初始化数据：
zadd post:good:sort 1000 post:91 1020 post:92

2、给id为91的帖子增加8000个赞

zincrby post:good:sort 30 post:91

3、首页推荐10个最热门的帖子
zrevrange post:good:sort 0 10

4、从帖子点击量排行榜中移除id为91的帖子
zrem post:good:sort post:91

```
阿里云一号服务器Redis服务:0>zadd post:good:sort 1000 post:91 1020 post:92

"2"
阿里云一号服务器Redis服务:0>zincrby post:good:sort 8000 post:91

"9000"
阿里云一号服务器Redis服务:0>zrevrange post:good:sort 0 10

 1)  "post:91"

 2)  "post:92"
阿里云一号服务器Redis服务:0>zrem post:good:sort post:91

"1"
阿里云一号服务器Redis服务:0>
```



> 2,4 Redis的基本数据类型：Hash

为什么要用Hash?

假如要存储的对象有很多属性，那么为了表示该对象，必然有很多Key来描述该对象，但是接下来的问题就变成了，假如我修改某个值，我是不是还需要把这个对象取出来，再加，是不是很麻烦？所以就有了Hash

```
user:1:name "swk"
user:1:age  500
user:1:id   1

user:1 {id:1,name:"swk",age:500}
```

- 1
- 2
- 3
- 4
- 5

1，hashes可以理解为一个map，这个map由一对一对的字段和值组成，所以，可以用hashes来保存一个对象：

2，hashes的常见操作：

1)，hset key field value:给一个hashes添加一个field和value；

2)，hget key field:可以得到一个hashes中的某一个属性的值：

```
阿里云一号服务器Redis服务:0>hset user:100 name swk

"1"
阿里云一号服务器Redis服务:0>hset user:100 age 500

"1"
阿里云一号服务器Redis服务:0>hget user:100 name

"swk"
阿里云一号服务器Redis服务:0>hget user:100 age

"500"
```

3)，hgetall key：一次性取出一个hashes中所有的field和value，使用list输出，一个field，一个value有序输出；

```
阿里云一号服务器Redis服务:0>hgetall user:100
 1)  "name"
 2)  "swk"
 3)  "age"
 4)  "500"
阿里云一号服务器Redis服务:0>
```

4)，hmset key field1 value1 field2 value2 field3 value3...:一次性的设置多个值(hashes multiple set)

```
阿里云一号服务器Redis服务:0>hmset user:101 name zbj age 8000 home 高家庄
"OK"
```

5)，hmget key field1 field2...：一次性的得到多个字段值(hashes multiple get)，以列表形式返回；

```
阿里云一号服务器Redis服务:0>hgetall user:101
 1)  "name"
 2)  "zbj"
 3)  "age"
 4)  "8000"
 5)  "home"
 6)  "高家庄"
阿里云一号服务器Redis服务:0>
```

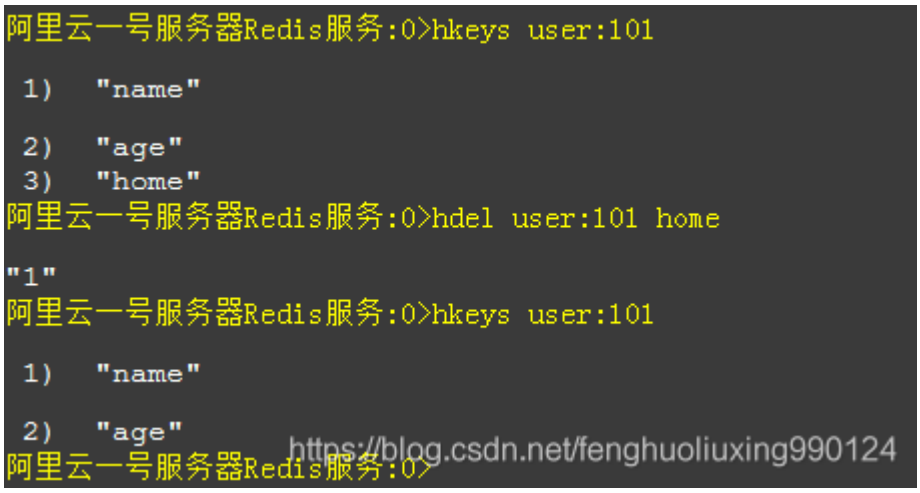
6)，hincrby key field number：给hashes的一个field的value增加一个值(integer)，这个增加操作是原子操作：

```
阿里云一号服务器Redis服务:0>hincrby user:101 age 1000
"9000"
阿里云一号服务器Redis服务:0>hgetall user:101
 1)  "name"
 2)  "zbj"
 3)  "age"
 4)  "9000"
 5)  "home"
 6)  "高家庄"
阿里云一号服务器Redis服务:0>
```



7)，hkeys key：得到一个key的所有fields字段，以list返回：

8)，hdel key fieldh:删除hashes一个指定的filed；




> Hash的应用场景

1，hashes使用场景：替代string，以更合理的方式保存对象；