**一、引言**  
  
    上一篇文章写了Redis的特征，使用场景，同时也介绍了Redis的基本数据类型，redis的数据类型是操作redis的基础，这个必须好好的掌握。今天我们开始介绍一些Redis的高级特性，虽然有些命令很简单，但是这些命令是使用很广泛的，并且是通用的一些命令，好好的把握这些命令，通过组合这些命令可以完成比较复杂的任务，这都是基础，我们就开始我们今天的学习之旅吧。  
  
**二、Redis的高级特性**

   下面我们就是Redis高级特性的每个知识点，只是个人的理解，随着学习的深入，也会更新相关的章节。  
  
**1、通用命令：**  
  
**1.1】、语法：keys pattern，pattern可以是类似正则的匹配规则，可以是\*，也可以是?，返回匹配规则的键值key的列表。**

[IMG_256](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>keys \*

1）"name"

2）"set2"

3）"set1"

192.168.127.128:6379>keys n\*

1）"name"

192.168.127.128:6379>keys s\*

1）"set2"

2）"set1"

192.168.127.128:6379>keys set?

1）"set2"

2）"set1"

192.168.127.128:6379>keys n?me

1）"name"

[IMG_257](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.2】、语法：exists key [key ...]，判断一个或者多个key是否存在，并返回存在key的个数，不存在返回0。**

[IMG_258](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>keys \*

1）"name"

2）"age"

192.168.127.128:6379>exists name age address(该值存在)

（integer）2

192.168.127.128:6379>exists name1 age1 address(该值存在)

（integer）0

[IMG_259](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.3】、语法：expire key seconds，给指定的key设置过期时间，单位是秒。用ttl命令可以查看剩余的时间，如果想更精确的、性能更好的倒计时功能，可以使用该功能。设置过期时间成功返回值为1，此命令可以多次执行，后面的过期值会覆盖前面的过期值。过期的数据会从Redis数据库里删除。**

[IMG_260](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>expire name 60 设置name过期时间为60秒

（integer）1

192.168.127.128:6379>expire name 50 设置name过期时间为50秒

（integer）1

192.168.127.128:6379>ttl name

（integer）48

192.168.127.128:6379>ttl name

（integer）45

ttl返回值：

-1、过期时间不存在

-2、过期操作成功，数据已经移除

[IMG_261](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.4】、语法：persist key，取消已经设置了过期时间的key的过期策略。取消过期策略成功返回1，key没有过期策略返回0。**

[IMG_262](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>keys \*

1）"name"

2）"age"

192.168.127.128:6379>expire name 60 设置name过期时间为60秒

（integer）1

192.168.127.128:6379>persist name

（integer）1 //成功取消设置的过期策略

192.168.127.128:6379>persist name

（integer）0 //上一步已经取消了，在此执行取消操作返回0。

[IMG_263](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.5】、语法：select index，在Redis中，逻辑划分为16个数据库，索引从0-15，我们在使用的过程中，可以针对不同的业务逻辑把数据缓存在不同的数据库，这是一个很好的架构设计，切记不要把所有的数据放在0号数据库里，不便于管理和分类。这16个数据库数据是分离的，不是共享的，或者说数据都是独立的。**

[IMG_264](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>select 1

OK

192.168.127.128:6379[1]>

192.168.127.128:6379[1]>select 5

OK

192.168.127.128:6379[5]>

[IMG_265](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.6】、语法：move key db,至指定名称为key的键值对转移到指定数据库db里面，db是数据下标索引。**

[IMG_266](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//索引为0的数据库

192.168.127.128:6379>keys \*

1）"name"

2）"age"

//索引为5的数据库

192.168.127.128:6379[5]>keys \*

(empty list or set)

//在索引为5的数据库里面增加sex键值对，然后把其移动到索引为0的数据库

192.168.127.128:6379[5]>set sex nan

OK

192.168.127.128:6379[5]>keys \*

1)"sex"

192.168.127.128:6379[5]>move sex 0

(integer)1

//索引为0的数据库

192.168.127.128:6379>keys \*

1）"name"

2）"sex" //从索引为5的数据已经移动到了索引为0的数据库

2）"age"

[IMG_267](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.7】、语法：randomkey，随机返回当前数据库中的key，其实这个功能我们可以使用在抽奖的场景，如果在并发比较高，而且有要保证性能的情况下，我建议大家使用Redis的这个功能来完成抽奖的功能。有时候会返回重复的值，如果做抽奖需要做处理一下。**

[IMG_268](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>keys \*

1）"name"

2）"sex"

2）"age"

192.168.127.128:6379>randomkey

"name"

192.168.127.128:6379>randomkey

"age"

192.168.127.128:6379>randomkey

"age"

[IMG_269](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.8】、语法：rename key newkey，重新给指定的key命名。**

[IMG_270](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>keys \*

1）"name"

2）"sex"

2）"age"

192.168.127.128:6379>rename name name1

OK

192.168.127.128:6379>keys \*

1）"name1"

2）"sex"

2）"age"

192.168.127.128:6379>rename name1 name

OK

[IMG_271](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.9】、语法：echo message，在当前介质上打印message信息。**

192.168.127.128:6379>echo patrickLiu

"patrickLiu"

**1.10】、语法：dbsize，返回当前数据中键值对的个数，或者更准确的说是key的个数。**

[IMG_272](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>keys \*

1）"name"

2）"sex"

2）"age"

192.168.127.128:6379>dbsize

3

[IMG_273](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.11】、语法：info [section]，可以获取当前数据的详细信息。**

[IMG_274](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>info //显示当前数据的所有数据

192.168.127.128:6379>info replication //显示指定节的信息 # Replication

role：master

connected\_slaves:0

...

[IMG_275](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.12】、语法：config get \*，获取配置文件中的所有的配置数据。**

[IMG_276](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>config get \*

...

192.168.127.128:6379>config get bind

1）"bind"

2）"192.168.127.128"

192.168.127.128:6379>config get port

1）"port"

2）"6379"

[IMG_277](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1.13】、语法：flushdb，清空当前数据所有的键值对。**

192.168.127.128:6379>flushdb

OK

192.168.127.128:6379>keys \*

(empty list or set)

**1.14】、语法：flushall，清空所有数据库的所有键值对。**

[IMG_278](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>flushall

OK

192.168.127.128:6379[1]>keys \*

(empty list or set)

...

192.168.127.128:6379[5]>keys \*

(empty list or set)

...

[IMG_279](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**2、安全设置**  
  
     **2.1】、vi 或者 vim 编辑 redis.conf 文档，修改**  
       
          #requirepass 123456 （设置密码）  
            
           requirepass admin  
  
  **2.2】、重启服务器 pkill redis-server**

[IMG_280](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux~]# application/program/redis-tool/redis-cli -h 192.168.127.128 -p 6379

192.168.127.128:6379>keys \*

(error)NOAUTH Authentication required.

192.168.127.128:6379>auth admin //授权 OK

192.168.127.128:6379>keys \* //可以执行

(empty list or set)

[IMG_281](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

    **2.3】、[root@linux~]/root/application/program/redis-tool/redis-cli -h 192.168.127.128 -p 6379 -a admin**  
  
**3、事务模型**  
  
     redis也是支持事务的，但是这个事务是比较简单的，之所以说简单，是因为redis的事务不能保证操作的同时成功或者同时失败而进行提交或者回滚，这种情形很容易导致数据的不一致性，大家使用的时候要特别注意。  
  
**3.1】、首先通过multi命令开启事务，然后进行事务性的设置，这些设置完的数据会存储在虚拟队列中，最后可以通过exec命令执行事务。**

[IMG_282](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//当前数据库中无值

192.168.127.128:6379>keys \*

(empty list or set)

192.168.127.128:6379>multi

OK

192.168.127.128:6379>set p1 1

QUEUED

192.168.127.128:6379>set p2 2

QUEUED

192.168.127.128:6379>set p3 3

QUEUED

192.168.127.128:6379>exec

1)OK

2)OK

3)OK

192.168.127.128:6379>keys \*

1)"p3"

2)"p2"

3)"p1"

[IMG_283](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

   **3.2】、redis的事务可以通过discard命令取消事务。**

[IMG_284](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//当前数据库中的值

192.168.127.128:6379>keys \*

1)"p3"

2)"p2"

3)"p1"

192.168.127.128:6379>multi

OK

192.168.127.128:6379>set p4 4

QUEUED

192.168.127.128:6379>set p5 5

QUEUED

192.168.127.128:6379>discard //取消事务 OK

//值没变

192.168.127.128:6379>keys \*

1)"p3"

2)"p2"

3)"p1"

[IMG_285](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

  **3.3】、特别注意redis事务的不一致性。**

[IMG_286](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//当前数据库中的值

192.168.127.128:6379>keys \*

1)"age"

2)"name"

192.168.127.128:6379>get name

"huangfeihong"

192.168.127.128:6379>get age

(integer)18

192.168.127.128:6379>multi

OK

192.168.127.128:6379>incr name

QUEUED

192.168.127.128:6379>incr age

QUEUED

192.168.127.128:6379>exec

1)(error)ERR value is not an integer or out of range

2)(integer)19

192.168.127.128:6379>get name

"huangfeihong" //姓名没有改变

192.168.127.128:6379>get age

(integer)19 //当前年龄已经修改

[IMG_287](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

      再次强调，在真正的产品环境中，要特别注意Redis事务的这个特点，或者说是弊端，不能保证数据的一致性。  
  
**4、持久化**  
  
     Redis是一种高级key-value数据库。它跟memcached类似，不过数据可以持久化。Redis的所有数据都是保存在内存中，然后不定期的通过异步方式保存到磁盘上(这称为“半持久化模式”)；也可以把每一次数据变化都写入到一个append only file(aof)里面(这称为“全持久化模式”)。所以来说，总结起来，Redis的数据持久化就有两种方式。  
  
    **4.1】、第一种方法filesnapshotting（rdb）**：默认redis是会以快照的形式将数据持久化到磁盘的（一个二进制文件，dump.rdb，这个文件名字可以指定），在配置文件中的格式是：save N（秒数） M（改变的个数），表示在N秒之内，redis至少发生M次修改，redis才生成快照并持久化到磁盘。当然我们也可以手动执行save或者bgsave（异步）做快照。该模式无论怎么设置，都会产生时间间隔，如果在这时间间隔断电，数据还是会丢失，会对数据持久化产生不利的影响，在实际的生产环境中，不建议使用该模式。  
  
          save 900 1     900秒内有超过1个的key值被修改了就发起快照保存  
  
          save 300 30    300秒内有超过30个的key值被修改了就发起快照保存  
  
          save 60  10000 60秒内有超过10000个的key值被修改了就发起快照保存  
  
  
   **4.2】、第二种方法是Append-only-file（aof）**：Append-only方法可以做到全部数据不丢失，但redis的性能就要差些。AOF就可以做到全程持久化，只需要在配置文件中开启，默认是不打开AOF功能的（appendonly no），appendonly yes开启AOF之后，redis每执行一个修改数据的命令，都会把它添加到aof文件中，当redis重启时，将会读取AOF文件进行“重放”以恢复到redis关闭前的最后时刻。  
  
          appendfsync有三个选项：always、everysec和no：  
  
**1、always：服务器会在每执行一个事件就把AOF缓冲区的内容强制性的写入硬盘上的AOF文件里，可以看成你每执行一个redis写入命令就往AOF文件里记录这条命令，这保证了数据持久化的完整性，但效率是最慢的，却也是最安全的；**  
  
**2、everysec：服务端每执行一次写操作也会把该条命令追加到一个单独的AOF缓冲区的末尾，并将AOF缓冲区写入AOF文件，然后每隔一秒才会进行一次文件同步把内存缓冲区里的AOF缓存数据真正写入AOF文件里，这个模式兼顾了效率的同时也保证了数据的完整性，即使在服务器宕机也只会丢失一秒内对redis数据库做的修改；**  
  
        **3、no：你完全可以接受Redis数据的丢失，它虽然也会把每条写命令追加到AOF缓冲区的末尾，然后写入文件，但什么时候进行文件同步真正把数据写入AOF文件里则由系统自身决定，即当内存缓冲区的空间被填满或者是超过了设定的时限后系统自动同步。这种模式下效率是最快的，但对数据来说也是最不安全的，如果redis里的数据都是从后台数据库如mysql中取出来的，属于随时可以找回或者不重要的数据，那么可以考虑设置成这种模式。**  
  
        [root@linux~]# more application/program/redis-tool/redis.conf  
        dir "/root/application/program/redis-tool/datas/"           #AOF或者RDB文件存放目录  
        appendonly yes                       #启用AOF持久化，默认关闭，RDB模式就关闭了，两者是二选一的情况  
        appendfilename "appendonly.aof"      #AOF持久化的文件名称（默认）  
        appendfsync no                       #AOF持久化的策略  
        auto-aof-rewrite-percentage 100      #触发AOF文件重写的条件（默认）  
        auto-aof-rewrite-min-size 64mb       #触发AOF文件重写的条件（默认）  
  
  
**5、发布订阅**  
  
     Redis的这个发布订阅功能有点类似MQ（消息队列）功能，但是这功能不能滥用，它的使用场景非常有限，并且数据量也不能太大。比如：成千上万条的数据场景是可以满足的，如果订阅和发布的数据量超过N万、十万、千万或者亿级别，这样的场景就很难满足了，到那时候我们就需要使用真正的MQ来满足该功能了。  
  
     5.1】、**语法：subscribe channel [channel ...]**，进行订阅监听，可以接受来之发布端的消息了。可以定义多个频道，这个频道的名称是自己可以定义的。

[IMG_288](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>subscribe c1 c2

Reading messages...(press Ctrl-C quit)

1)"subscribe"

2)"c1"

3)(integer)1

1)"subscribe"

2)"c2"

3)(integer)2

//接受的到的消息

1)"message"

2)"c2"

3)"redis"

1)"message"

2)"c1"

3)"hello"

//处于等待状态，可以接收消息了

[IMG_289](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

     5.2】、**语法：publish channel message**，针对指定的消息频道发布消息，订阅端就可以接受到消息了。

192.168.127.128:6379>publish c1 hello

(integer)1 //发布消息成功

192.168.127.128:6379>publish c2 redis

(integer)1 //发布消息成功

**三、总结**  
  
   随着学习的深入，理解的越来越全面和深刻了，以前很多联系不起来的知识点都联系起来了，以后还需继续努力，如果有了新的学习心得，还会持续更新相关的章节。随着学习的继续，也让自己养成学习的习惯。努力吧！！！！