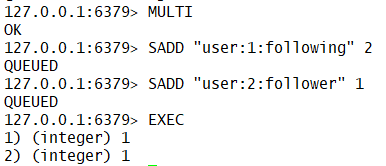
**1. redis事务**

Redis事务的原理：先将属于一个事务的命令发送给redis，然后再让redis依次执行这些命令。



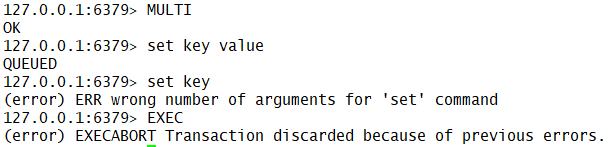
使用MULTI命令告诉redis：下面我发给你的命令属于同一个事务。

使用EXEC命令的返回值就是这些命令的返回值组成的列表，返回值顺序和命令的顺序相同。

如果在发送EXEC命令前客户端断线了，则redis会清空事务队列，事务中的所有命令都不会执行。一旦客户端发送了exec命令，所有命令都会被执行，即使此后的客户端断线也没关系，因为redis已经记录了所有要执行的命令。

**1.1 错误处理**

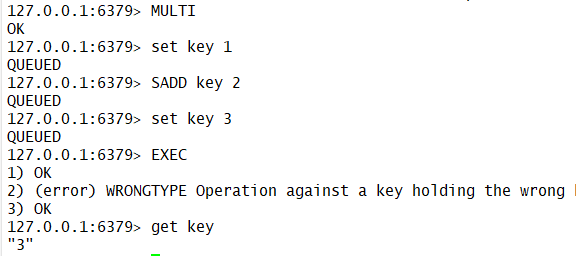
①语法错误。语法错误指命令不存在或者命令参数的个数不对。



如果开始事务后，只要有一个命令有语法错误，执行EXEC命令后，redis就会直接返回错误，连语法正确的命令也不会执行。

②运行错误

运行错误指在命令执行时出现的错误，比如使用散列类型的命令操作集合类型的键，这种错误在实际执行之前redis是无法发现的，所以在事务里这样的命令是会被redis接受并执行的。如果事务里的一条命令出现了运行错误，事务里**其他命令依然会继续执行**。

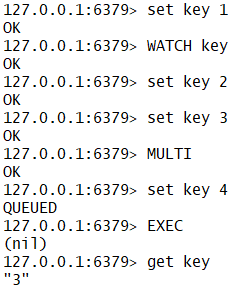


Redis事务没有回滚功能，需要自己处理错误。

**1.2 watch命令介绍**

一个事务中只有当所有命令都一次执行完后才能得到每个结果的返回值，可是有些情况下需要先**获得一条指令的返回值**，然后在根据这个值执行下一条命令，例如，介绍INCR命令时曾经说过使用get和set命令自己实现incr函数会出现竞态条件

**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，所以在multi命令后可以修改watch监控的键值）



事务执行前使用watch监控了key，并对key进行了修改 set key 3等，所以事务中的set key 4 就不会执行。

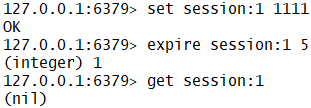
**提示：由于watch命令的作用只是当被监控的键值被修改后阻止之后一个事务的执行，而不能保证其他客户端不修改这一个键值，所以需要在exec执行失败后重新执行**

执行exec命令后会取消对所有键的监控，如果不想执行事务中的命令，也可以使unwatch来取消监控。

**2.生存时间**

Redis中可以使用expire命令设置一个键的生存时间，到时间后redis会自动删除它。

Expire key seconds，seconds表示键的生存时间，单位是秒，Pexpire命令单位是毫秒。



TTL命令，返回的是key的剩余时间（单位是秒）



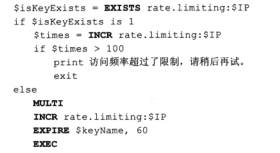
如果想取消键的生存时间设置（即将键恢复成永久的），可以使用persist命令，如果生存时间被清除成功，返回1；否则返回0；

除了persist之外，使用set或者getset为键复制也会同时消除键的生存时间，

**2.2实线访问频率限制之一**

为了减轻服务器的压力，需要限制每个用户（ip）一段时间的最大访问量，与时间有关的操作很容易想到expire命令。

如果要限制每分钟每个用户最多只能访问100个页面，思路就是对每个用户使用一个名为“rate.limiting: 用户IP”的字符串类型键，每次用户访问则使用INCR命令递增该键的键值，如果递增后的值是1（第一次访问页面），则同时还要设置该键的生存时间为1分钟。这样每次用户访问页面时，都读取该键的键值，如果超过100就表明该用户的访问频率超过了限制，需要提示用户稍后访问。该键每分钟就会被删除，所以下一分钟的用户访问次数又会重新计算。

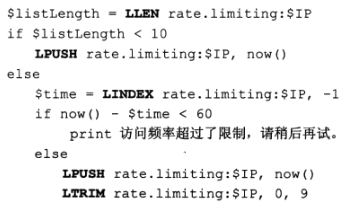


**2.3 实现访问频率限制之二**

如果一个用户在一分钟的第一秒访问了一次博客，在同一分钟的最后一秒访问了9次，又在下一分钟的第一秒访问了10词，这样的访问是可以通过现在的访问频率限制的，但实际上该用户在2秒钟内访问19次博客，这与每个用户每分钟只能访问10次的限制差距较大。

如何解决？

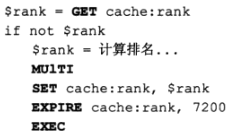
对每个用户，我们使用一个列表类型的键来记录他最近10次访问博客的时间。一旦键中的元素超过10个，就判断时间最早的元素距现在的时间是否小于1分钟，如果是表明用户最近1分钟的访问次数超过了10次；如果不是就将现在的时间加入到列表中，同时把最早的元素删除。



代码中的now是获取当前unix的系统时间。LTRIM修剪(trim)一个已存在的 list

**2.4实现缓存**

统计前10名学生的姓名，由于学生成绩总在不断的变化，需要每隔两小时就重新计算一次排名，这可以通过给键设置生存时间的方式实现。每次用户访问首页时，程序先查询缓存键是否存在，如果存在则直接使用缓存的值；否则重新计算排名并将计算结果赋值给该键，并同时设置该键的生存时间为两小时。



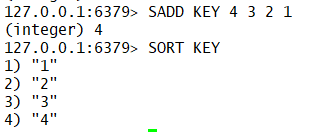
设置生存时间过长，会使得redis占满内存，如果生存时间过短，就会导致redis内存的大量空闲。因此需要按照一定的规则淘汰不需要的缓存键，这种方式只将redis用做缓存系统时非常实用。

**Redis支持的淘汰策略**

|  |  |
| --- | --- |
| 规则 | 说明 |
| Volatile-lru | lru算法删除一个键（只对设置生存时间的） |
| Allkeys-lru | 使用lru删除一个键 |
| Volatile-random | 设置生存时间的，随机删除 |
| Allkeys-random | 随即删除 |
| Volatile-ttl | 删除生存时间最少的一个键 |
| Noeviction | 不删除键，只返回错误 |

**3.排序**

**3.1 sort命令**



还能对list和sorted set进行排序