# **[Redis事务机制和分布式锁](https://www.cnblogs.com/laojiao/p/9580653.html)**

## **Redis事务机制**

严格意义来讲,Redis的事务和我们理解的传统数据库(如mysql)的事务是不一样的；Redis的事务实质上是命令的集合，在一个事务中要么所有命令都被执行，要么所有事物都不执行。   
一个事务从开始到执行会经历以下三个阶段：

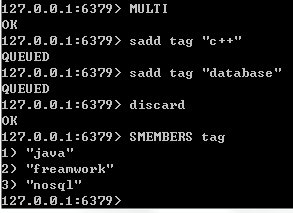
1. 开始事务。
2. 命令入队。
3. 执行事务。

在MySQL中我们使用START TRANSACTION 或 BEGIN开启一个事务，使用COMMIT提交一个事务；而在Redis中我们使用**MULTI** 开始一个事务，由 **EXEC** 命令触发事务， 一并执行事务中的所有命令。



可以看到，**MULTI** 开始到 **EXEC**结束前，中间所有的命令都被加入到一个命令队列中；当执行 **EXEC**命令后，将QUEUE中所有的命令执行。

此外我们可以使用DISCARD取消事务。



需要注意的是:   
1.Redis的事务没有关系数据库事务提供的回滚（rollback），所以开发者必须在事务执行失败后进行后续的处理；   
2.如果在一个事务中的**命令出现错误**，那么**所有的命令都不会执行**；   
3.如果在一个事务中出现**运行错误**，那么**正确的命令会被执行**。

**WATCH**

研究过java的J.U.C包的人应该都知道CAS，CAS是一种保证原子性的操作。那么redis也提供了这样的一个机制，就是利用watch命令来实现的。

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。

## **分布式锁**

**什么是分布式锁？**

分布式锁是控制**分布式系统之间同步访问共享资源的一种方式**。如果不同的系统或是同一个系统的不同主机之间共享了一个或一组资源，那么访问这些资源的时候，往往需要互斥来防止彼此干扰来保证一致性，在这种情况下，便需要使用到分布式锁。

实现分布式锁有很多实现方式和工具，如Zookeeper、Redis等。

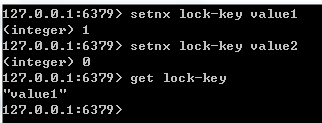
使用Redis实现分布式锁原理：

Redis为**单进程单线程**模式，采用队列模式将并发访问变成串行访问，且多客户端对Redis的**连接**并不存在竞争关系，基于此，Redis中可以使用SETNX命令实现分布式锁。

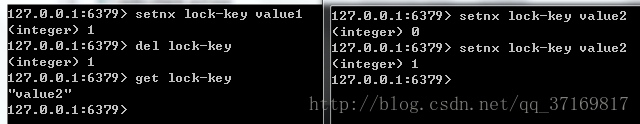
SETNX——SET if Not eXists（如果不存在，则设置）：

setnx key value

将 key 的值设为 value ，**当且仅当 key 不存在**。   
若给定的 key 已经存在，则 SETNX 不做任何动作。



如果需要解锁，使用 del key 命令就能释放锁：



左图首先使用setnx对键加锁成功返回1，右图再次使用setnx命令对键加锁失败返回0，说明有客户端持有锁。使用del释放锁以后，右图就可以使用setnx命令对键加锁。

## **解决死锁**

如果一个持有锁的客户端失败或崩溃了不能释放锁，该怎么解决？

答：给锁设置一个过期时间，可以通过两种方法实现：通过命令 “setnx 键名 过期时间 “；或者通过设置锁的expire时间，让Redis去删除锁。

**第一种实现方式：**   
使用 setnx key “当前系统时间+锁持有的时间”和getset key “当前系统时间+锁持有的时间”组合的命令就可以实现。   
具体做法如下：

客户端2发送**SETNX lock.test** 想要获得锁，由于之前的客户端1还持有锁，所以Redis返回一个0   
客户端2发送**GET lock.test** 以检查锁是否超时了，如果没超时，则等待或重试。  
反之，如果已超时，客户端2通过下面的操作来尝试获得锁：   
**GETSET lock.test 过期的时间**   
通过GETSET，客户端2拿到的时间戳**如果仍然是超时的，那就说明，客户端2如愿以偿拿到锁了。**   
如果在客户端2之前，有个客户端3比客户端2快一步执行了上面的操作，那么客户端2拿到的时间戳是个未超时的值，这时，说明客户端2没有如期获得锁，需要再次等待或重试。   
尽管客户端2没拿到锁，但它改写了客户端3设置的锁的超时值，不过这一点非常微小的误差带来的影响可以忽略不计。

**第二种就非常简单了：**   
通过Redis中expire()给锁设定最大持有时间，如果超过，则Redis来帮我们释放锁。

1.客户端1使用**setnx**获得了锁，并且使用**expire**设定一个过期时间，假定是10ms

2.过了4ms后，客户端1不幸运的宕机了，此时客户端2想要通过**setnx**尝试获得锁，但是锁还没有过期，任然被客户端1所持有。

3.到了11ms时，锁过期了，Redis帮我们删除了锁，此时客户端2想要通过**setnx**尝试获得锁，此时就能成功获得锁。

在实际过程中，我们可以设定一个时间T，用来表示客户端在初次尝试获得锁失败以后，在多次尝试获得锁所花的时间。如果次时间为0，表示除此尝试获得锁失败以后就不会再去尝试获得锁了。