# Redis事务简介

1. Redis事务通过multi、exec、discard、watch等命令实现。

·multi命令用于开启事务；

·exec命令用于提交事务；

·discard命令用于取消事务；

·watch命令用于监视键值对；

1. Redis事务由一条或多条Redis命令组成，当提交事务后，这些命令按照添加顺序依次执行，并且Redis服务器不会在执行事务的途中去执行其他客户端输入的命令，而是等待整个事务执行完毕后，才会执行其他客户端发送的命令请求。
2. 当客户端使用multi命令开启事务后，输入的命令会被放入一个命令队列中；这些命令并不会立即被执行，而是当服务器收到该客户端执行的exec命令才会顺序地执行这个队列中的所有命令。

# Redis事务的ACID特性

在关系型数据库中，事务具有ACID（原子性、一致性、隔离性、持久性）特性。同样的，Redis数据库也具有ACID特性。

## 原子性

1. Redis事务的原子性即一个事务中的所有命令要么全部执行成功，要么全部执行失败。
2. 使用multi命令开启事务后，该客户端向服务器发送的命令被服务器接收到后，先将它们放入一个队列中，而不是立即执行它们，当使用exec命令提交事务后，再按照顺序执行它们。
3. 当客户端处于事务状态，该客户端每向服务端发送一个命令，都会返回一个内容为QUEUED的回复，表示这条命令已成功进入Redis服务器事务的队列中；当使用exec命令提交事务后，将会以数组的形式返回执行的结果，数组中的每一个元素表示着其对应位置上命令的执行结果。
4. 在关系型数据库中，事务是支持回滚的，但在Redis中并不支持事务回滚。使用multi命令开启事务后，如果输入的命令语法错误（如输入了某个不存在的命令或命令所需的参数没有给足），则会导致整个事务执行失败；如果命令语法正确，但在执行时因为类型或者键不存在等原因报错，那么这个事务也会继续执行下去，而不是终止执行，该事务将会执行成功，因为Redis不具有事务回滚机制，即使遇到错误，也会执行下去。

## 一致性

1. 事务的一致性指的是数据库在执行事务之前是一致的，在事务之后，不管是否执行成功或失败，数据库中的数据也应该具有一致性，这种一致性指的是当前数据库从一种状态变为一种新的状态，数据在变化前后符合数据本身的定义和要求，同时不包含非法或无效的脏数据。
2. 在使用exec命令提交事务之前，如果该客户端输入的命令在服务端入队错误，如命令不存在或者输入的命令语法格式错误（如参数错误、参数数量错误、参数顺序错误），都会导致入队错误，从而导致事务执行失败。

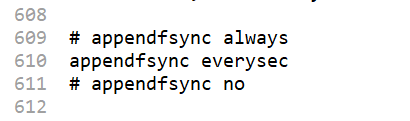
## 隔离性

事务的隔离性指的是当多个用户并发访问服务器时，都开启了事务。Redis事务不同于SQL中的事务，SQL中的事务具有隔离级别，而Redis事务没有隔离级别，每一个提交的事务在执行的过程中，其他的客户端提交的事务或输入的命令都必须等待此事务执行完成，才可以继续执行。简单来讲就是，提交事务后，其他客户端提交的事务都必须等待此事务执行完成后才能执行。

## 持久性

持久性指的是当一个事务正确执行完成后，它对数据库的改变是永久性的。

1. 事实上，Redis的事务并不具有持久性，Redis事务的持久性取决于Redis服务器有没有采用持久化。如果Redis服务器没有采用持久化（不管是RDB持久化方式，还是AOF持久化方式），那么事务不具有持久性。假如服务器发生故障（如停电、崩溃），将会丢失服务器上的所有数据。
2. 如果Redis采用的是AOF持久化方式：
3. 当Redis配置文件redis.conf中的appendfsync属性的值为always时，可以保证Redis事务具有持久性；每当服务器执行相关命令后，包括事务命令在内，程序都会调用sync同步函数，将命令数据及时保存到系统硬盘中，这就保证了数据的持久性。
4. 当Redis配置文件redis.conf中的appendfync属性的值为everysec时，服务器会每隔1秒执行一次数据同步操作，并将数据保存到硬盘中。如果服务器发生故障停机，可能刚好发生等待数据同步的那一秒之内，就会导致数据丢失，因此无法保证事务的持久性。
5. 当Redis配置文件redis.conf中的appendfync属性的值为no时，服务器命令数据同步保存到硬盘中的操作将由操作系统来控制；因此，事务数据再同步的过程中，可能会因为一些原因而丢失，这种情况也不能保证事务的持久性。



appendfsync属性在redis.conf配置文件中的配置。

1. 如果Redis采用的是RDB持久化方式，那么只有在特定的保存条件被满足时，服务器才会执行bgsave命令，实现数据的保存。而如果是异步执行gbsave命令，那么服务器并不能保证在第一时间将事务数据保存到硬盘中，因此也就不能保证事务的持久性。换句话说，RDB持久化方式不能保证事务具有持久性。

（关于数据持久化后面有详细章节进行讲解）

# 使用事务

使用事务有4个命令，它们是multi、exec、discard、watch命令。

## multi

multi命令用于开启事务。命令如下：

**multi**

执行multi命令后，该客户端进入事务状态，接着输入的命令会被服务器保存到一个命令队列中，但是还没有执行它们。

返回值：总是返回OK

## discard

discard命令用于取消本次事务。命令如下：

**discard**

当使用multi命令开启事务后，如果需要取消本次事务，则可以执行discard命令。

返回值：取消成功返回OK。

## exec

exec命令用于提交事务。命令如下：

**exec**

当使用multi命令开启事务后，向命令队列中添加了多条命令，如果需要执行这些命令，则使用exec提交该事务，提交该事务后，服务器会按照命令的添加顺序依次执行每个命令，在执行这些命令的过程中，其他客户端执行的命令都必须等待这些命令执行完成后才可以执行。

返回值：返回一个列表，该列表中的每一项表示事务中的每条命令执行的结果，与命令所在的位置一一对应。

## watch

watch用于监视键，来 保证事务执行前后数据的一致性。命令如下：

**watch key [key……]**

key表示要监视的键，可以有多个。

1. 当A客户端使用watch命令对某个键进行监视后，A开启一个事务，对这个键进行修改，但还未使用exex命令提交事务；如果这时，其他客户端对A客户端监视的这个键进行了修改，那么A客户端此时使用exec命令提交事务，该事务会执行失败，并返回nil。因为其监视的键被其他数据库修改了。
2. 也就是说watch命令用于保证一次事务中要操作的键在没有被其他客户端改变的前提下，该事务才会执行成功。
3. 当执行exec、discard命令后或客户端与服务器断开了连接，watch监视的键将会被取消监视，你也可以使用unwatch手动取消被监视的键；

返回值：总是返回OK