> 关于
>
> 目前网络上有许多Redis相关面试题的总结，但基本上都是差不多的题目，本文章内容中的题目参考整理自互联网，每道题目我都在参考答案的基础上做了重新整理和总结！并在一些相对比较重要的、面试几率大的题目前面使用不同数量的（★）进行了标注，三颗星及以上需要牢记哦！

预备知识

什么是NoSQL？有什么特点和使用场景？（★★）

我们可能都听说过关系型数据库，比如MySQL、SQL Server、Oracle等。

与之对应的，就有非关系型数据库，即NoSQL数据库。NoSQL有时也称作Not Only SQL的缩写，即“不仅仅是SQL”。是对不同于传统关系型数据库的数据库管理系统的统称。NoSQL不保证关系型数据库的**ACID特性**。当然，它们的功能都是存储数据。NoSQL被我们用得最多的当数key-value存储，当然还有其他的文档型的、列存储、图型数据库、xml数据库等。

NoSQL数据库的出现，弥补了关系型数据库（比如MySQL）在某些方面的不足，在某些方面能极大的节省开发成本和维护成本。

NoSQL数据库的特点：

- 易扩展：NoSQL 去掉了关系型数据库的关系型特性，数据与数据之间没有关系，这样就非常容易扩展，无形之间也在架构的层面上带来了可扩展的能力。
- 高可用：NoSQL在不太影响性能的情况，就可以方便的实现高可用的架构。
- 高性能：大多NoSQL数据库数据都是存储在内存中，读写快，尤其是在处理庞大数据时表现优秀。
- 灵活的数据模型：NoSQL无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。

NoSQL数据库适用场景：

- 对数据库性能要求较高。
- 对数据一致性要求不是很高。
- 对灵活性要求很强的系统。
- 一些数据模型比较简单。
- 给定 key 容易映射复杂值的环境。
- 大型系统的日志信息存储。

为什么要用缓存？（★★）

之所以要使用缓存，就是**为了提高系统性能进而提升用户体验以及应对更多的用户请求。**直接操作缓存能够承受的数据请求数量是远远大于直接访问数据库的，所以我们可以考虑把数据库中的部分数据添加到缓存中去，这样用户的一部分请求会直接到缓存这里而不用经过数据库。进而，我们也就提高的系统整体的并发。**缓存主要是可以降低对数据库的访问压力，缓存数据大部分位于内存中，进而降低IO成本，加快数据的访问速度。**

比如 CPU Cache 缓存的是内存数据用于解决 CPU 处理速度和内存不匹配的问题，内存缓存的是硬盘数据用于解决硬盘访问速度过慢的问题。再比如操作系统在 页表方案 基础之上引入了快表来加速虚拟地址到物理地址的转换。我们可以把块表理解为一种特殊的高速缓冲存储器（Cache）。

> 了解
>
> - `本地缓存`：指的是在应用中的缓存组件，其最大的优点是应用和缓存是在同一个进程内部，请求缓存非常快速，没有过多的网络开销等，在单应用不需要集群支持或者集群情况下各节点无需互相通知的场景下使用本地缓存较合适；同时，它的缺点也是应为缓存跟应用程序耦合，多个应用程序无法直接的共享缓存，各应用或集群的各节点都需要维护自己的单独缓存，对内存是一种浪费。
> - `分布式缓存`：指的是与应用分离的缓存组件或服务，其最大的优点是自身就是一个独立的应用，与本地应用隔离，多个应用可直接的共享缓存。

Redis基础

什么是Redis，Redis有哪些特点？（★★★★）

Redis是一种常用的基于键值对（key-value）的NoSQL数据库。Redis全称为：Remote Dictionary Server 即远程字典服务，是一个开源的使用ANSI C语言编写的C/S架构的程序。

Redis中的value支持string（字符串）、hash（哈希）、 list（列表）、set（集合）、zset（有序集合）、Bitmaps（位图）、 HyperLogLog、GEO（地理信息定位）等**多种数据结构，因此Redis可以满足很多的应用场景。**

因为**Redis会将所有数据都存放在内存中，所以它的读写性能非常出色。**Redis还可以将内存的数据利用快照和日志的形式保存到硬盘上，这样在发生类似断电或者机器故障的时候，内存中的数据不会“丢失”。同时他还支持网络、并提供了多种语言的API。此外，**Redis可用于缓存、数据库、事件发布或订阅，高速队列等场景。**

总结一下，Redis主要特点如下：

- Redis是一个高性能key/value内存型数据库，为了保证效率，Redis数据都是缓存在内存中。

  > 扩展
  >
  > 数据库有两种：一种是硬盘数据库，一种是内存数据库。
  >
  > 硬盘数据库是把值存储在硬盘上，在内存中就存储一下索引，当硬盘数据库想访问硬盘的值时，它先在内存里找到索引，然后再找值。问题在于，在读取和写入硬盘的时候，如果读写比较多的时候，它会把硬盘的IO功能堵死。
  >
  > 内存存储是讲所有的数据都存储在内存里面，数据读取和写入速度非常快。

- Redis支持丰富的数据类型，包括string(字符串)、list(链表)、set(集合)、zset(sorted set 有序集合)和hash（哈希类型）。通过选用不同的数据结构，我们可以使用Redis解决不同的问题。
- Redis支持各种不同方式的排序。
- Redis提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。

- Redis支持持久化，会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件。将数据存储在内存里面的数据保存到硬盘中，保证数据安全，方便进行数据备份和恢复。
- Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。
- Redis 6.0版本之前的是单线程。

你为什么要用 Redis，它有何优势？（★★★）

之所以使用Redis，主要有以下几个原因：

- **性能极高、速度快**：Redis 基于内存实现数据存储，它的读取速度是 110000次/s，写速度是 81000次/s；
- **支持丰富数据类型**：支持string，list，set，sorted set，hash等；
- **拥有丰富的特性**： 可以按key设置过期时间，过期后将会自动删除。Redis 还有很多的用途，比如可以用作缓存、消息队列、搭建 Redis 集群等；
- **拥有命令提示功能**：Redis 客户端拥有强大的命令提示功能，使用起来非常的方便，降低了学习门槛；
- **可支持事务** ：Redis对事务是部分支持的，如果是在入队时报错，那么都不会执行；在非入队时报错，那么成功的就会成功执行；
- **可移植性**：Redis 使用用标准 C语言编写的，能够在大多数操作系统上运行，比如 Linux，Mac，Solaris 等。

Redis数据库的使用场景？（★★★★）

1、缓存

这是Redis应用最广泛地方，基本所有的Web应用都会使用Redis作为缓存，来降低数据源压力，提高响应速度。缓存现在几乎是所有中大型网站都在用的必杀技，合理的利用缓存不仅能够提升网站访问速度，还能大大降低数据库的压力。Redis提供了键过期功能，也提供了灵活的键淘汰策略，所以，现在Redis用在缓存的场合非常多。

2、排行榜

很多网站都有排行榜应用的，如淘宝的月度销量榜单、商品按时间的上新排行榜等。Redis提供了列表和有序集合数据结构，合理地使用这些数据结构可以很方便地构建各种排行榜系统。

3、计数器

什么是计数器，如电商网站商品的浏览量、视频网站视频的播放数等。为了保证数据实时效，每次浏览都得给+1，并发量高时如果每次都请求数据库操作无疑是种挑战和压力。Redis提供的incr命令来实现计数器功能，基于内存操作而且计数性能非常好，可以用来记录浏览量、点赞量等等。

4、分布式会话

集群模式下，在应用不多的情况下一般使用容器自带的session复制功能就能满足，当应用增多相对复杂的系统中，一般都会搭建以Redis等内存数据库为中心的session服务，session不再由容器管理，而是由session服务及内存数据库管理。

5、分布式锁

在很多互联网公司中都使用了分布式技术，分布式技术带来的技术挑战是对同一个资源的并发访问，如全局ID、减库存、秒杀等场景，并发量不大的场景可以使用数据库的悲观锁、乐观锁来实现，但在并发量高的场合中，利用数据库锁来控制资源的并发访问是不太理想的，大大影响了数据库的性能。可以利用Redis的setnx功能来编写分布式的锁，如果设置返回1说明获取锁成功，否则获取锁失败，实际应用中要考虑的细节要更多。

6、 社交网络

点赞、踩、关注/被关注、共同好友等是社交网站的基本功能，社交网站的访问量通常来说比较大，而且传统的关系数据库类型不适合存储这种类型的数据，Redis提供的哈希、集合等数据结构能很方便的的实现这些功能。

7、最新列表

Redis列表结构，LPUSH可以在列表头部插入一个内容ID作为关键字，LTRIM可用来限制列表的数量，这样列表永远为N个ID，无需查询最新的列表，直接根据ID去到对应的内容页即可。

8、消息系统

消息队列是大型网站必用中间件，如ActiveMQ、RabbitMQ、Kafka等流行的消息队列中间件，主要用于业务解耦、流量削峰及异步处理实时性低的业务。 Redis提供了发布订阅功能和阻塞队列的功能，能实现一个简单的消息队列系统。另外，这个不能和专业的消息中间件相比。

> Redis的应用一般会结合项目去问，以一个电商项目的用户服务为例：
>
> - 用户注册发送验证码：使用Redis保存验证码并设置期限。
> - Token存储：用户登录成功之后，使用Redis存储Token。
> - 登录失败次数计数：使用Redis计数，登录失败超过一定次数，锁定账号。
> - 地址缓存：对省市区数据的缓存。
> - 分布式锁：分布式环境下登录、注册等操作加分布式锁。
> - 业务相关：缓存一些不易改变经常被查询的数据。

说一下 Redis 和 Memcached 的区别和共同点？（★★★）

共同点：

1. 都是基于内存的数据库，一般都用来当做缓存使用。
2. 都有过期策略。
3. 两者的性能都非常高。

区别：

1. **支持的数据类型不同**：Redis 支持更丰富的数据类型（支持更复杂的应用场景）。Redis 不仅仅支持简单的 k/v 类型的数据，同时还提供 list，set，zset，hash 等数据结构的存储。Memcached 只支持最简单的 k/v 数据类型。
2. **是否支持数据持久化**：Redis 支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用,而 Memcached 把数据全部存在内存之中。
3. **是否支持灾难恢复**：Redis 因为有持久化就可以实现灾难恢复，因为可以把缓存中的数据持久化到磁盘上。
4. **内存不足时表现不同**：Redis 在遇到内存使用完成之后利用淘汰机制，可以将不用的数据放到磁盘上，Memcached 在服务器内存使用完之后，就会直接报异常。

5. **是否支持集群模式**：Memcached 没有原生的集群模式，需要依靠客户端来实现往集群中分片写入数据；但是 Redis 目前是原生支持 cluster 模式的。
6. **是否支持多线程**：Memcached 是多线程，非阻塞 IO 复用的网络模型；Redis 使用单线程的多路 IO 复用模型。 （Redis 6.0 引入了多线程 IO ）
7. **过期数据的处理不同**：Memcached 过期数据的删除策略只用了惰性删除，而 Redis 同时使用了惰性删除与定期删除。
8. **其他功能的支持**：Redis 支持发布订阅模型、Lua 脚本、事务等功能，而 Memcached 不支持。并且，Redis 支持更多的编程语言。

Redis 常用的数据结构有哪些？（★★★★）

在Redis中有五种基本数据类型，分别为`String`（字符串）、`List`（列表）、`Set`（集合）、`Hash`（散列）、`Zset`（有序集合）。此外还有三种特殊的数据类型，`HyperLogLogs`（基数统计）、`Bitmap`（位存储）、`Geospatial`(地理位置)。

- **String** 是 Redis 中最简单同时也是最常用的一个数据结构。String 是一种二进制安全的数据结构，可以用来存储任何类型的数据比如字符串、整数、浮点数、图片（图片的 base64 编码或者解码或者图片的路径）、序列化后的对象。需要注意的是**值最大不能超过 512MB**。
- **List** 列表是简单的字符串列表，按照插入顺序排序。 列表是一种比较灵活的数据结构，它可以充当栈和队列的角色。可以添加一个元素到列表的头部（左边）或者尾部（右边）。
- **Set** 是集合类型也是用来保存多个的字符串元素，和列表类型不一样的是，集合中不允许有重复元素，并且集合中的元素是无序的。 集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是O(1)。
- **zset** 和 set 一样也是字符串类型元素的集合，且不允许重复的成员。 不同的是有序集合中的元素可以排序。每个元素都会关联一个double类型的分数。Redis正是通过分数来为集合中的成员进行从小到大的排序。zset的成员是唯一的,但分数(score)却可以重复。
- **hash** 是一个键值(key=>value)对集合。 Redis hash是一个字符串类型的field和value的映射表，hash特别适合用于存储对象。

能说一下Redis每种数据结构的使用场景吗？（★★★★）

String的使用场景

字符串类型的使用场景：信息缓存、计数器、共享Session、限速、分布式锁等等。

- **实战场景1：记录每一个用户的访问次数，或者记录每一个商品的浏览次数**

  每一个用户访问次数或者商品浏览次数的修改是很频繁的，如果使用mysql这种文件系统频繁修改会造成mysql压力，效率也低。而使用Redis的好处有二：使用内存，很快；单线程，所以无竞争，数据不会被改乱。

- **实战场景2：缓存频繁读取，但是不常修改的信息，如用户信息，视频信息**

  方案：先从Redis读取，有值就从Redis读取，没有则从MySQL读取，并写一份到Redis中作为缓存，注意要设置过期时间。

- **实战场景3：限定某个ip特定时间内的访问次数**

方案：用key记录IP，value记录访问次数，同时key的过期时间设置为60秒，如果key过期了则重新设置，否则进行判断，当一分钟内访问超过100次，则禁止访问。

- **实战场景4：分布式session**

  我们知道session是以文件的形式保存在服务器中的；如果你的应用做了负载均衡，将网站的项目放在多个服务器上，当用户在服务器A上进行登陆，session文件会写在A服务器；当用户跳转页面时，请求被分配到B服务器上的时候，就找不到这个session文件，用户就要重新登陆。

  如果想要多个服务器共享一个session，可以将session存放在Redis中，Redis可以独立于所有负载均衡服务器，也可以放在其中一台负载均衡服务器上；但是所有应用所在的服务器连接的都是同一个Redis服务器。

Hash的使用场景

哈希主要可以用于**缓存用户信息、缓存对象**的应用场景。

以购物车为例子，用户id设置为key，那么购物车里所有的商品就是用户key对应的值了，每个商品有id和购买数量，对应hash的结构就是商品id为field，商品数量为value。

如果将商品id和商品数量序列化成json字符串，那么也可以用上面讲的string类型存储。下面对比一下这两种数据结构：

对比项	string（json）	hash
效率	很高	高
容量	低	低
灵活性	**低**	高
序列化	简单	**复杂**

总结一下：

> 当对象的某个属性需要频繁修改时，不适合用string+json，因为它不够灵活，每次修改都需要重新将整个对象序列化并赋值；如果使用hash类型，则可以针对某个属性单独修改，没有序列化，也不需要修改整个对象。比如，商品的价格、销量、关注数、评价数等可能经常发生变化的属性，就适合存储在hash类型里。

List的使用场景

列表本质是一个有序的，元素可重复的队列。主要可以用于**消息队列、文章列表**的应用场景。

实战场景：定时排行榜

list类型的lrange命令可以分页查看队列中的数据。可将每隔一段时间计算一次的排行榜存储在list类型中，如QQ音乐内地排行榜，每周计算一次存储再list类型中，访问接口时通过page和size分页转化成lrange命令获取排行榜数据。

但是，并不是所有的排行榜都能用list类型实现，只有定时计算的排行榜才适合使用list类型存储，与定时计算的排行榜相对应的是实时计算的排行榜，list类型不能支持实时计算的排行榜，下面介绍有序集合sorted set的应用场景时会详细介绍实时计算的排行榜的实现。

Set的使用场景

集合的特点是无序性和确定性（不重复）。

实战场景：收藏夹

例如QQ音乐中如果你喜欢一首歌，点个『喜欢』就会将歌曲放到个人收藏夹中，每一个用户做一个收藏的集合，每个收藏的集合存放用户收藏过的歌曲id。key为用户id，value为歌曲id的集合。

Sorted Set的使用场景

有序集合的特点是有序，无重复值。与set不同的是sorted set每个元素都会关联一个score属性，redis正是通过score来为集合中的成员进行从小到大的排序。

实战场景：实时排行榜

QQ音乐中有多种实时榜单，比如飙升榜、热歌榜、新歌榜，可以用redis key存储榜单类型，score为点击量，value为歌曲id，用户每点击一首歌曲会更新redis数据，sorted set会依据score即点击量将歌曲id排序。

Redis单线程为什么还那么快呢？（★★★★）

Redis的速度非常的快，单机的Redis就可以支撑每秒十几万的并发，相对于MySQL来说，性能是MySQL的几十倍。速度快的原因主要有几点：

1. 主要是因为**完全基于内存操作**，大部分请求是纯粹的内存操作，非常快速。数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)；
2. 数据结构简单，对数据操作也简单，Redis中的数据结构是专门进行设计的；
3. 采用单线程，恰恰避免了不必要的上下文切换和竞争条件，也不存在多进程或者多线程导致的切换而消耗 CPU，不用去考虑各种锁的问题，不存在加锁释放锁操作，没有因为可能出现死锁而导致的性能消耗；
4. **使用多路I/O复用模型，非阻塞IO**；
5. C语言实现，优化过的数据结构，基于几种基础的数据结构，Redis做了大量的优化，性能极高。

说一下I/O多路复用。（★★★★）

为什么 Redis 中要使用 I/O 多路复用这种技术呢?

首先，Redis 是跑在单线程中的，所有的操作都是按照顺序线性执行的，但是由于读写操作等待用户输入或输出都是阻塞的，所以 I/O 操作在一般情况下往往不能直接返回，这会导致某一文件的 I/O 阻塞导致整个进程无法对其它客户提供服务，而 **I/O 多路复用**就是为了解决这个问题而出现的。

什么是I/O多路复用呢?

简单理解就是：一个服务端进程可以同时处理多个套接字描述符。

“多路”是指多个网络连接，“复用”是指同一个线程，I/O多路复用模型是利用`select`、`poll`、`epoll`函数可以同时监测多个I/O流的能力，这些函数会轮询一遍所有的I/O流（epoll只轮询真正发出了事件的流），依次处理返回了数据，处于就绪状态的流，让单个线程高效地处理了多个连接请求，尽量减少网络IO的时间消耗，而且Redis是在内存中操作的，速度非常快。所以，Redis具有很高的吞吐量。

引用知乎上一个高赞的回答来解释什么是I/O多路复用。假设你是一个老师，让30个学生解答一道题目，然后检查学生做的是否正确，你有下面几个选择：

- 第一种选择：按顺序逐个检查，先检查A，然后是B，之后是C、D。。。这中间如果有一个学生卡住，全班都会被耽误。这种模式就好比，你用循环挨个处理socket，根本不具有并发能力。
- 第二种选择：你创建30个分身，每个分身检查一个学生的答案是否正确。这种类似于为每一个用户创建一个进程或者- 线程处理连接。
- 第三种选择，你站在讲台上等，谁解答完谁举手。这时C、D举手，表示他们解答问题完毕，你下去依次检查C、D的答案，然后继续回到讲台上等。此时E、A又举手，然后去处理E和A。

第一种就是阻塞IO模型，第三种就是`I/O复用模型`。

为什么Redis是单线程的？（★★★）

官方表示，因为Redis是基于内存的操作，CPU不是Redis的瓶颈，Redis的瓶颈最有可能是机器内存的大小或者网络带宽。既然单线程容易实现，而且CPU不会成为瓶颈，那就顺理成章地采用单线程的方案了，毕竟采用多线程会可能有很多麻烦。

详细来说，单线程的话**避免了各种锁的性能消耗**：Redis的数据结构并不全是简单的Key-Value，还有list，hash等复杂的结构，这些结构有可能会进行很细粒度的操作，比如在很长的列表后面添加一个元素，在hash当中添加或者删除一个对象。这些操作可能就需要加非常多的锁，导致的结果是同步开销大大增加。总之，在单线程的情况下，就不用去考虑各种锁的问题，不存在加锁释放锁操作，没有因为可能出现死锁而导致的性能消耗。再有采用单线程，**避免了不必要的上下文切换和竞争条件**，也不存在多进程或者多线程导致的切换而消耗 CPU。

Redis 单线程模型了解吗？（★★★）

Redis 内部使用`文件事件处理器`，这个文件事件处理器是单线程的，所以 Redis 才叫做单线程的模型。它采用 IO 多路复用机制同时监听多个 socket，根据 socket 上的事件来选择对应的事件处理器进行处理。

为啥 Redis 单线程模型也能效率这么高?

- 纯内存操作
- 核心是基于非阻塞的 IO 多路复用机制
- 单线程反而避免了多线程的频繁上下文切换问题

Redis6.0 之前为什么不使用多线程？（★★★）

虽然说 Redis 是单线程模型，但是，实际上，**Redis 在 4.0 之后的版本中就已经加入了对多线程的支持。**不过，Redis 4.0 增加的多线程主要是针对一些大键值对的删除操作的命令，使用这些命令就会使用主线程之外的其他线程来“异步处理”。

大体上来说，**Redis 6.0 之前主要还是单线程处理。那，Redis6.0 之前为什么不使用多线程?**

我觉得主要原因有下面 3 个：

1. 单线程编程容易并且更容易维护；
2. Redis 的性能瓶颈不在 CPU ，主要在内存和网络；
3. 多线程就会存在死锁、线程上下文切换等问题，甚至会影响性能。

Redis6.0 之后为何引入了多线程？(★★★)

Redis 6.0 之后的版本开始选择性使用多线程模型。Redis 选择使用单线程模型处理客户端的请求主要还是因为 CPU 不是 Redis 服务器的瓶颈，使用多线程模型带来的性能提升并不能抵消它带来的开发成本和维护成本，系统的性能瓶颈也主要在网络 I/O 操作上；而 Redis 引入多线程操作也是出于性能上的考虑，主要是为了提高网络 IO 读写性能，**对于一些大键值对的删除操作，通过多线程非阻塞地释放内存空间也能减少对 Redis 主线程阻塞的时间，提高执行的效率。**

虽然，Redis6.0 引入了多线程，但是 Redis 的多线程只是在网络数据的读写这类耗时操作上使用了，执行命令仍然是单线程顺序执行。因此，也不需要担心线程安全问题。

Redis6.0 的多线程默认是禁用的，只使用主线程。如需开启需要修改 redis 配置文件 `redis.conf` ：

```
io-threads-do-reads yes
```

开启多线程后，还需要设置线程数，否则是不生效的。同样需要修改 redis 配置文件 `redis.conf` ：

```
io-threads 4 #官网建议4核的机器建议设置为2或3个线程，8核的建议设置为6个线程
```

Redis事务

Redis事务相关的命令有哪几个？(★★)

(1) WATCH

可以为Redis事务提供 check-and-set （CAS）行为。被WATCH的键会被监视，并会发觉这些键是否被改动过了。 `如果有至少一个被监视的键在 EXEC 执行之前被修改了， 那么整个事务都会被取消` ， EXEC 返回nil-reply来表示事务已经失败。

(2) MULTI

`用于开启一个事务，它总是返回OK`。MULTI执行之后，客户端可以继续向服务器发送任意多条命令， 这些命令不会立即被执行，而是被放到一个队列中，当 EXEC命令被调用时， 所有队列中的命令才会被执行。

（3）UNWATCH

`取消 WATCH 命令对所有 key 的监视，一般用于DISCARD和EXEC命令之前`。如果在执行 WATCH 命令之后， EXEC 命令或 DISCARD 命令先被执行了的话，那么就不需要再执行 UNWATCH 了。因为 EXEC 命令会执行事务，因此 WATCH 命令的效果已经产生了；而 DISCARD 命令在取消事务的同时也会取消所有对 key 的监视，因此这两个命令执行之后，就没有必要执行 UNWATCH 了。

（4）DISCARD

当执行 DISCARD 命令时， `事务会被放弃， 事务队列会被清空`，并且客户端会从事务状态中退出。

（5）EXEC

`负责触发并执行事务中的所有命令`：如果客户端成功开启事务后执行EXEC，那么事务中的所有命令都会被执行。

> 如果客户端在使用MULTI开启了事务后，却因为断线而没有成功执行EXEC,那么事务中的所有命令都不会被执行。需要特别注意的是：即使事务中有某条/某些命令执行失败了，事务队列中的其他命令仍然会继续执行，Redis不会停止执行事务中的命令，而不会像我们通常使用的关系型数据库一样进行回滚。

如何使用 Redis 事务？(★★★)

Redis 可以通过 `MULTI`，`EXEC`，`DISCARD` 和 `WATCH` 等命令来实现事务(transaction)功能。

使用 `MULTI` 命令后可以输入多个命令。Redis 不会立即执行这些命令，而是将它们放到队列，当调用了 `EXEC` 命令将执行所有命令。

这个过程是这样的：

1. 开始事务（`MULTI`）。
2. 命令入队(批量操作 Redis 的命令，先进先出（FIFO）的顺序执行)。
3. 执行事务(`EXEC`)。

也可以通过 `DISCARD` 命令取消一个事务，它会清空事务队列中保存的所有命令。

`WATCH` 命令用于监听指定的键，当调用 `EXEC` 命令执行事务时，如果一个被 `WATCH` 命令监视的键被修改的话，整个事务都不会执行，直接返回失败。

Redis 事务支持原子性吗？(★★★)

先看关系型数据库ACID 中关于原子性的定义：一个事务(transaction)中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被恢复(Rollback)到事务开始前的状态，就像这个事务从来没有执行过一样。

官方文档对事务的定义：

- **事务是一个单独的隔离操作**：事务中的所有命令都会序列化、按顺序地执行。事务在执行的过程中，不会被其他客户端发送来的命令请求所打断。
- **事务是一个原子操作**：事务中的命令要么全部被执行，要么全部都不执行。EXEC 命令负责触发并执行事务中的所有命令：如果客户端在使用 MULTI 开启了一个事务之后，却因为断线而没有成功执行 EXEC ，那么事务中的所有命令都不会被执行。另一方面，如果客户端成功在开启事务之后执行 EXEC ，那么事务中的所有命令都会被执行。

官方认为Redis事务是一个原子操作，这是站在执行与否的角度考虑的。但是从ACID原子性定义来看，**严格意义上讲Redis事务是非原子型的**，因为在命令顺序执行过程中，一旦发生命令执行错误Redis是不会停止执行然后回滚数据。也就是Redis 事务在运行错误的情况下，除了执行过程中出现错误的命令外，其他命令都能正常执行。并且，**Redis 是不支持回滚（roll back）操作的。**因此，Redis 事务其实是不满足原子性的（而且不满足持久性）。

可以将 Redis 中的事务就理解为 ：**Redis 事务提供了一种将多个命令请求打包的功能。然后，再按顺序执行打包的所有命令，并且不会被中途打断。**

除了不满足原子性之外，事务中的每条命令都会与 Redis 服务器进行网络交互，这是比较浪费资源的行为。因此，Redis 事务是不建议在日常开发中使用的。

其他

Redis常用的客户端有哪些？（★★）

- Jedis：是老牌的Redis的Java实现客户端，提供了比较全面的Redis命令的支持。
- Redisson：实现了分布式和可扩展的Java数据结构。促使使用者对Redis的关注分离，提供很多分布式相关操作服务，例如，分布式锁，分布式集合，可通过Redis支持延迟队列。
- Lettuce：高级Redis客户端，用于线程安全同步，异步和响应使用，支持集群，Sentinel，管道和编码器。基于Netty框架的事件驱动的通信层，其方法调用是异步的。Lettuce的API是线程安全的，所以可以操作单个Lettuce连接来完成各种操作。

微信关注