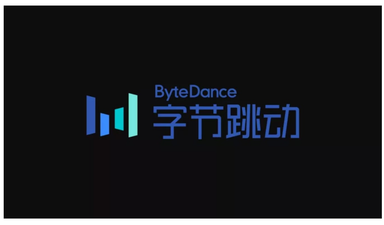
# 拼多多/字节跳动面试必问：高并发下的接口幂等性如何解决？

21CTO 昨天






来源：rrd.me/fffMq

**一、背景**

我们实际系统中有很多操作，是不管做多少次，都应该产生一样的效果或返回一样的结果。

例如：

1. 前端重复提交选中的数据，应该后台只产生对应这个数据的一个反应结果。
2. 我们发起一笔付款请求，应该只扣用户账户一次钱，当遇到网络重发或系统bug重发，也应该只扣一次钱；
3. 发送消息，也应该只发一次，同样的短信发给用户，用户会哭的；
4. 创建业务订单，一次业务请求只能创建一个，创建多个就会出大问题。

等等很多重要的情况，这些逻辑都需要幂等的特性来支持。

**二、幂等性概念**

幂等（idempotent、idempotence）是一个数学与计算机学概念，常见于抽象代数中。

在编程中,一个幂等操作的特点是其任意多次执行所产生的影响均与一次执行的影响相同。幂等函数，或幂等方法，是指可以使用相同参数重复执行，并能获得相同结果的函数。

这些函数不会影响系统状态，也不用担心重复执行会对系统造成改变。例如，“getUsername()和setTrue()”函数就是一个幂等函数.

更复杂的操作幂等保证是利用唯一交易号(流水号)实现.

我的理解：幂等就是一个操作，不论执行多少次，产生的效果和返回的结果都是一样的

**三、技术方案**

**1. 查询操作** 查询一次和查询多次，在数据不变的情况下，查询结果是一样的。select是天然的幂等操作

**2. 删除操作** 删除操作也是幂等的，删除一次和多次删除都是把数据删除。(注意可能返回结果不一样，删除的数据不存在，返回0，删除的数据多条，返回结果多个)

**3.唯一索引，防止新增脏数据** 比如：支付宝的资金账户，支付宝也有用户账户，每个用户只能有一个资金账户，怎么防止给用户创建资金账户多个，那么给资金账户表中的用户ID加唯一索引，所以一个用户新增成功一个资金账户记录

**要点：** 唯一索引或唯一组合索引来防止新增数据存在脏数据 （当表存在唯一索引，并发时新增报错时，再查询一次就可以了，数据应该已经存在了，返回结果即可）

**4. token机制，防止页面重复提交**

业务要求：

页面的数据只能被点击提交一次

发生原因：由于重复点击或者网络重发，或者nginx重发等情况会导致数据被重复提交

解决办法：集群环境：采用token加redis（redis单线程的，处理需要排队） 单JVM环境：采用token加redis或token加jvm内存

处理流程：

1. 数据提交前要向服务的申请token，token放到redis或jvm内存，token有效时间
2. 提交后后台校验token，同时删除token，生成新的token返回

token特点：

要申请，一次有效性，可以限流

注意：redis要用删除操作来判断token，删除成功代表token校验通过，如果用select+delete来校验token，存在并发问题，不建议使用

**5. 悲观锁** 获取数据的时候加锁获取

```
select * from table_xxx where id='xxx' for update;
```

注意：id字段一定是主键或者唯一索引，不然是锁表，会死人的

悲观锁使用时一般伴随事务一起使用，数据锁定时间可能会很长，根据实际情况选用

**6. 乐观锁** 乐观锁只是在更新数据那一刻锁表，其他时间不锁表，所以相对于悲观锁，效率更高。

乐观锁的实现方式多种多样可以通过version或者其他状态条件：

1、通过版本号实现

```
updatetable_xxxsetname=#name#,version=version+1whereversion=#version#
```

如下图(来自网上):



2、通过条件限制

```
updatetablexxxsetavaiamount=avaiamount-#subAmount#where avaiamount-#subAmount#>=0
```

要求：quality-#subQuality# >= ，这个情景适合不用版本号，只更新是做数据安全校验，适合库存模型，扣份额和回滚份额，性能更高

注意：乐观锁的更新操作，最好用主键或者唯一索引来更新,这样是行锁，否则更新时会锁表，上面两个sql改成下面的两个更好

```
updatetablexxxsetname=#name#,version=version+1whereid=#id#andversion=#version#
updatetablexxxsetavaiamount=avaiamount-#subAmount#whereid=#id#andavai_amount-#subAmou
```

**7. 分布式锁** 还是拿插入数据的例子，如果是分布是系统，构建全局唯一索引比较困难，例如唯一性的字段没法确定

这时候可以引入分布式锁，通过第三方的系统(redis或zookeeper)，在业务系统插入数据或者更新数据，获取分布式锁，然后做操作，之后释放锁

这样其实是把多线程并发的锁的思路，引入多多个系统，也就是分布式系统中得解决思路。

要点：某个长流程处理过程要求不能并发执行，可以在流程执行之前根据某个标志(用户ID+后缀等)获取分布式锁，其他流程执行时获取锁就会失败，也就是同一时间该流程只能有一个能执行成功，执行完成后，释放分布式锁(分布式锁要第三方系统提供)

**8. select + insert** 并发不高的后台系统，或者一些任务JOB，为了支持幂等，支持重复执行，简单的处理方法是，先查询下一些关键数据，判断是否已经执行过，在进行业务处理，就可以了

注意：核心高并发流程不要用这种方法

**9. 状态机幂等** 在设计单据相关的业务，或者是任务相关的业务，肯定会涉及到状态机(状态变更图)，就是业务单据上面有个状态，状态在不同的情况下会发生变更，一般情况下存在有限状态机

如果状态机已经处于下一个状态，这时候来了一个上一个状态的变更，理论上是不能够变更的，这样的话，保证了有限状态机的幂等。

注意：订单等单据类业务，存在很长的状态流转，一定要深刻理解状态机，对业务系统设计能力提高有很大帮助

**10. 对外提供接口的api如何保证幂等**

如银联提供的付款接口：需要接入商户提交付款请求时附带：source来源，seq序列号

source+seq在数据库里面做唯一索引，防止多次付款，(并发时，只能处理一个请求)

**重点**对外提供接口为了支持幂等调用，接口有两个字段必须传，一个是来源source，一个是来源方序列号seq，这个两个字段在提供方系统里面做联合唯一索引

这样当第三方调用时，先在本方系统里面查询一下，是否已经处理过，返回相应处理结果；没有处理过，进行相应处理，返回结果。

注意，为了幂等友好，一定要先查询一下，是否处理过该笔业务，不查询直接插入业务系统，会报错，但实际已经处理了。

**总结**

幂等性应该是合格程序员的一个基因。在设计系统时，是首要考虑的问题，尤其是在像支付宝，银行，互联网金融公司等涉及的都是钱的系统，既要高效，数据也要准确，所以不能出现多扣款，多打款等问题，这样会很难处理，用户体验也不好

如果觉得文章对您有帮助，**请转发朋友圈、点在看**，让更多人获益，感谢您的支持！

**相关阅读：**

字节跳动如何用7年, 成为腾讯最可怕的对手?
面试鹅厂，我被虐的体无完肤
面试官要是问你如何解决web高并发，你就这样回答