01 | 创建和更新订单时，如何保证数据准确...

订单系统是整个电商系统中最重要的一个子系统，订单数据也就是电商企业最重要的数据资产。今天这节课，我来和你说一下，在设计和实现一个订单系统的存储过程中，有哪些问题是要特别考虑的。

一个合格的订单系统，最基本的要求是什么？**数据不能错。**

一个购物流程，从下单开始、支付、发货，直到收货，这么长的一个流程中，每一个环节，都少不了更新订单数据，每一次更新操作又需要同时更新好几张表。这些操作可能被随机分布到很多台服务器上执行，服务器有可能故障，网络有可能出问题。

在这么复杂的情况下，保证订单数据一笔都不能错，是不是很难？实际上，只要掌握了方法，其实并不难。

首先，你的代码必须是正确没 Bug 的，如果说是因为代码 Bug 导致的数据错误，那谁也救不了你。

然后，你要会正确地使用数据库的事务。比如，你在创建订单的时候，同时要在订单表和订单商品表中插入数据，那这些插入数据的 INSERT 必须在一个数据库事务中执行，数据库的事务可以确保：执行这些 INSERT 语句，要么一起都成功，要么一起都失败。

1.订单系统的核心功能和数据：

我先和你简单梳理一下一个订单系统必备的功能，它包含但远远不限于：

创建订单；

随着购物流程更新订单状态；

查询订单，包括用订单数据生成各种报表。

为了支撑这些必备功能，在数据库中，我们至少需要有这样几张表：

订单主表：也叫订单表，保存订单的基本信息。

订单商品表：保存订单中的商品信息。

订单支付表：保存订单的支付和退款信息。

订单优惠表：保存订单使用的所有优惠信息。

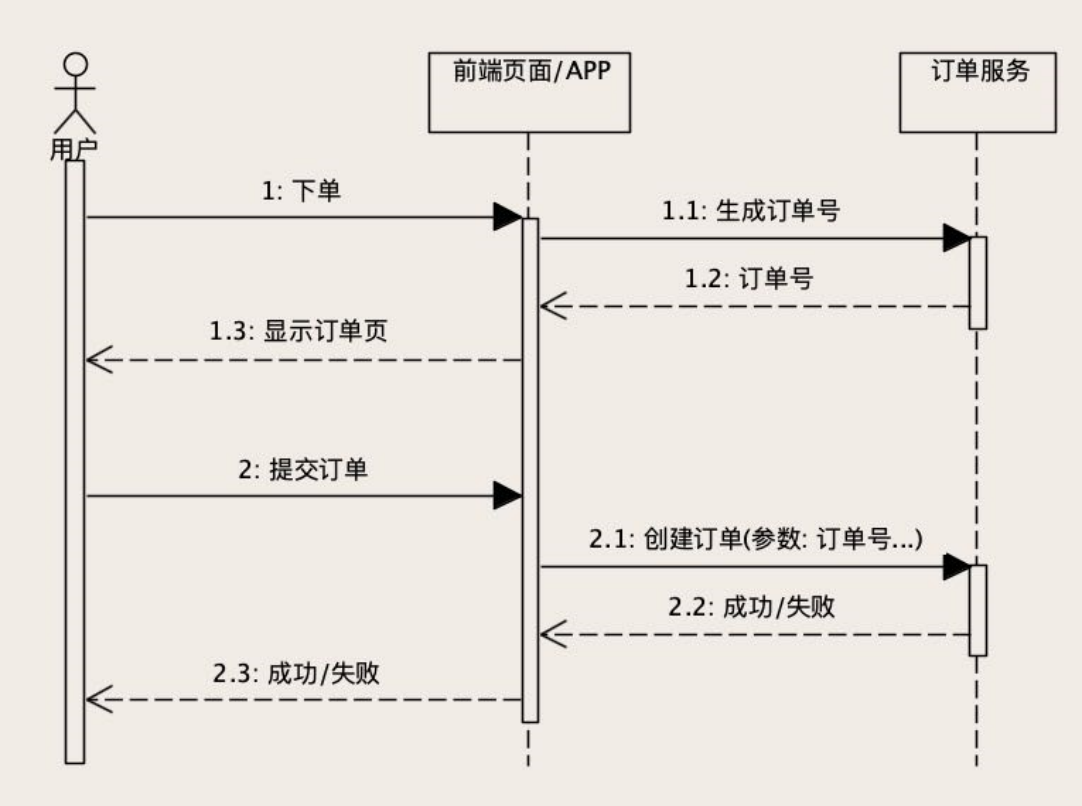
这几个表之间的关系是这样的：订单主表和后面的几个子表都是一对多的关系，关联的外键就是订单主表的主键，也就是订单号。

绝大部分订单系统它的核心功能和数据结构都是这样的。

2.如何避免重复下单：

解决办法是，让你的订单服务具备幂等性。

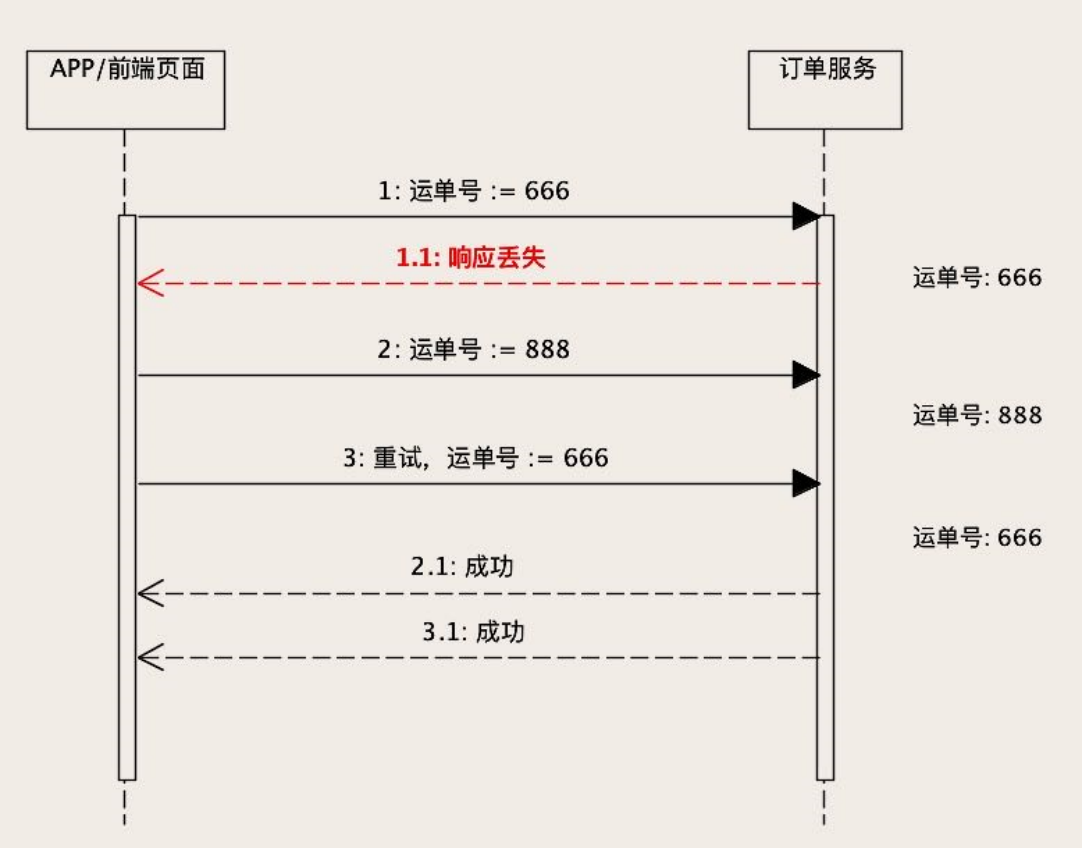
决创建订单服务的幂等性问题。
具体的做法是这样的，我们给订单系统增加一个“生成订单号”的服务，这个服务没有参数，返回值就是一个新的、全局唯一的订单号。在用户进入创建订单的页面时，前端页面先调用这个生成订单号服务得到一个订单号，在用户提交订单的时候，在创建订单的请求中带着这个订单号。
这个订单号也是我们订单表的主键，这样，无论是用户手抖，还是各种情况导致的重试，这些重复请求中带的都是同一个订单号。订单服务在订单表中插入数据的时候，执行的这些重复 INSERT 语句中的主键，也都是同一个订单号。数据库的唯一约束就可以保证，只有一次 INSERT 语句是执行成功的，这样就实现了创建订单服务幂等性。



2.如何解决ABA问题:

同样，订单系统各种更新订单的服务一样也要具备幂等性。
这些更新订单服务，比如说支付、发货等等这些步骤中的更新订单操作，最终落到订单库上，都是对订单主表的 UPDATE 操作。数据库的更新操作，本身就具备天然的幂等性，比如说，你把订单状态，从未支付更新成已支付，执行一次和重复执行多次，订单状态都是已支付，不用我们做任何额外的逻辑，这就是天然幂等。
那在实现这些更新订单服务时，还有什么问题需要特别注意的吗？还真有，在并发环境下，你需要注意 ABA 问题。
什么是 ABA 问题呢？我举个例子你就明白了。比如说，订单支付之后，小二要发货，发货完成后要填个快递单号。假设说，小二填了一个单号 666，刚填完，发现填错了，赶紧再修改成 888。对订单服务来说，这就是 2 个更新订单的请求。

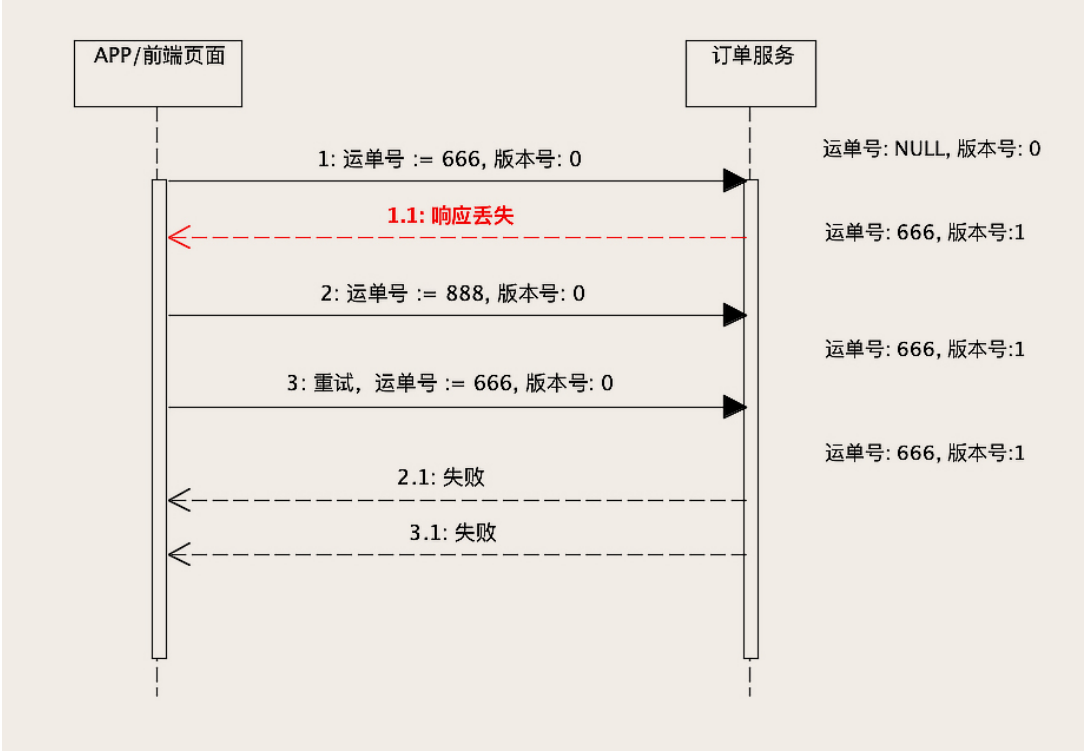
正常情况下，订单中的快递单号会先更新成 666，再更新成 888，这是没问题的。那不正常情况呢？666 请求到了，单号更新成 666，然后 888 请求到了，单号又更新成 888，但是 666 更新成功的响应丢了，调用方没收到成功响应，自动重试，再次发起 666 请求，单号又被更新成 666 了，这数据显然就错了。这就是非常有名的 ABA 问题。



ABA 问题怎么解决？这里给你提供一个比较通用的解决方法。给你的订单主表增加一列，列名可以叫 version，也即是“版本号”的意思。每次查询订单的时候，版本号需要随着订单数据返回给页面。页面在更新数据的请求中，需要把这个版本号作为更新请求的参数，再带回给订单更新服务。
订单服务在更新数据的时候，需要比较订单当前数据的版本号，是否和消息中的版本号一致，如果不一致就拒绝更新数据。如果版本号一致，还需要再更新数据的同时，把版本号 +1。“比较版本号、更新数据和版本号 +1”，这个过程必须在同一个事务里面执行。

具体的 SQL 可以这样来写：

```
UPDATE orders set tracking_number = 666, version = version + 1
WHERE version = 8;
```



订单号生成：

如果单纯是生成GUID（全局唯一ID）方法有很多，比如小规模系统完全可以用MySQL的Sequence或者Redis来生成。大规模系统也可以采用类似雪花算法之类的方式分布式生成GUID。

但是订单号这个东西又有点儿特殊要求，比如在订单号中最好包含一些品类、时间等信息，便于业务处理，再比如，订单号它不能是一个单纯自增的ID，否则别人很容易根据订单号计算出你大致的销量，所以订单号的生产算法在保证不重复的前提下，一般都会加入很多业务规则在里面，这个每家都不一样，算是商业秘密吧

1.创建订单如何解决重复下单？

通过提前预先生成订单号【这里生成订单号可以走单独的id生成器，如何设计一个好的id生成器需要学习】，利用数据库订单号唯一约束【订单号是主键，但主键不是由数据库自增，而是由外部传入】来避免重复入库，实现幂等

2.订单更新如何解决ABA问题，在表中增加一个代表版本号的version字段，每次更新时用version做条件，并对其＋1入库