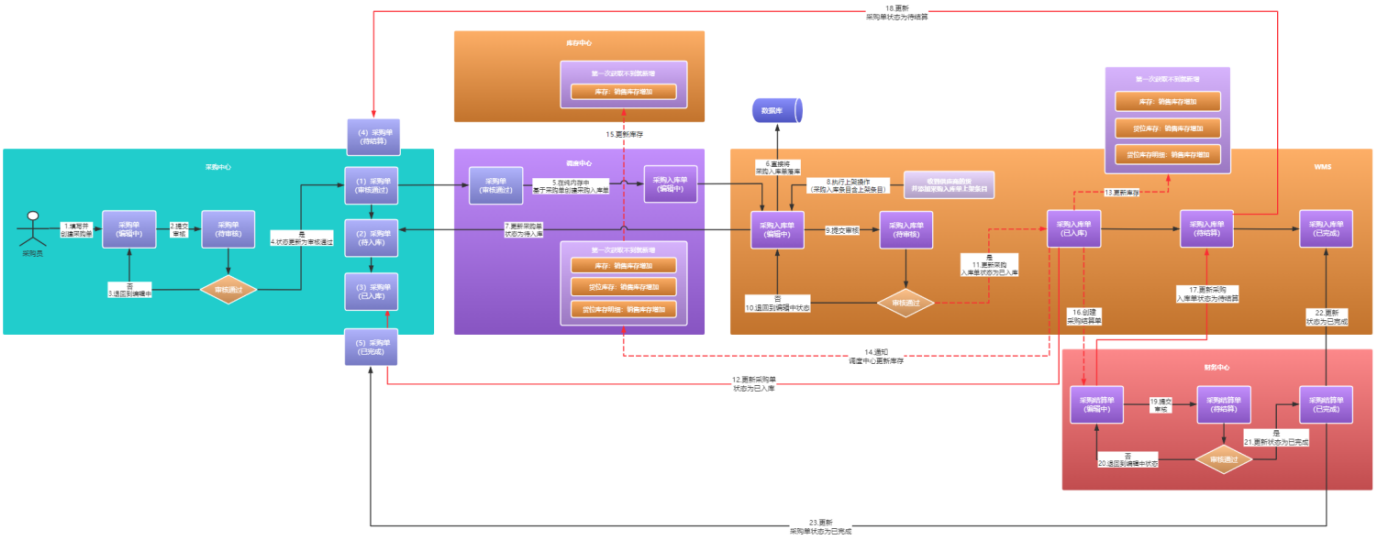
大型电商系统-采购核心链路接口一致性保障

1.分享概要

本次分享儒猿学院架构课中**《大型电商系统之核心交易接口的一致性保障实践-采购流程的接口幂等性方案以及开发》**，本次分享会带着大家一步一图，先梳理出架构课中大型电商系统v1.0版本中，商品采购的核心业务流程（注：以下的业务梳理会高度结合电商系统v1.0的业务代码逻辑，阅读前建议先复习下课程中的业务，并结合着代码来看），根据梳理出的业务流程，再分析和考量每个接口调用，看下哪些操作需要添加接口一致性的保障；针对那些需要接口一致性保障的操作，会给出具体的解决方案和思考建议，在分享之前大家可以先思考下如下面试题：

什么是接口幂等性？接口幂等性问题有哪些常见的解决方案？

完整商品采购业务流程如下图所示：

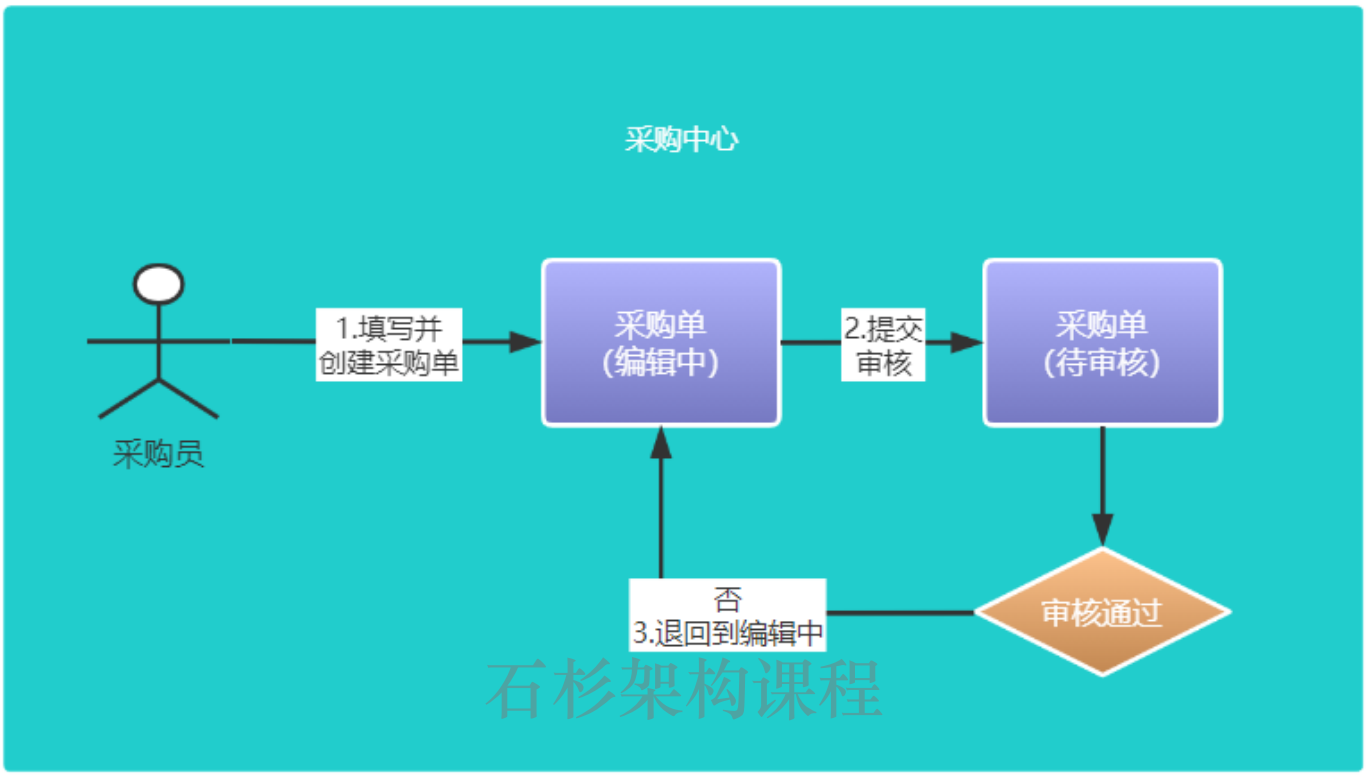


2.流程图解析

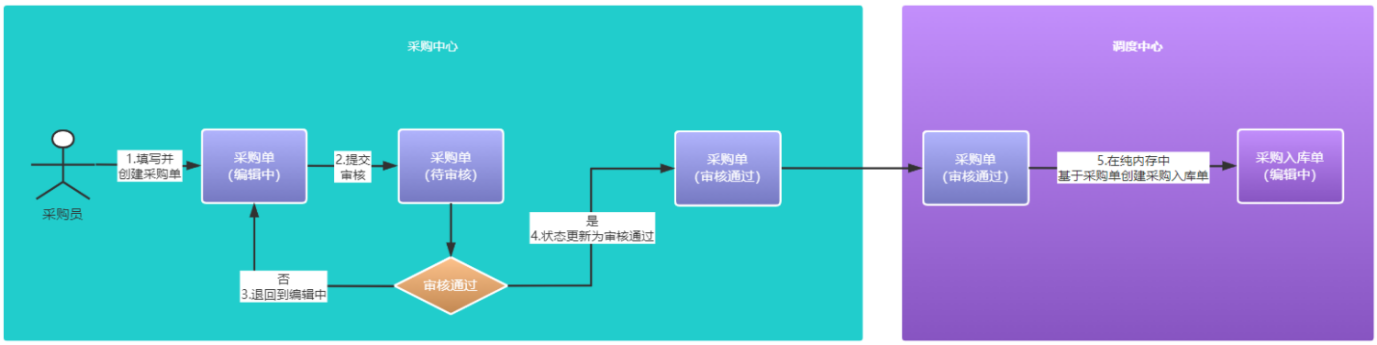
2.1 采购核心业务梳理

电商系统的核心业务简单来说就是销售商品，一开始仓库里当然什么货都没有，我们要做的第一件事就是采购货物。

首先采购员或者说有采购需求的部门的人，他们会填写并提交采购单发起采购请求，如果采购需求不合理或者采购超出预算等原因，采购单就会被驳回重新修改，如下图所示：

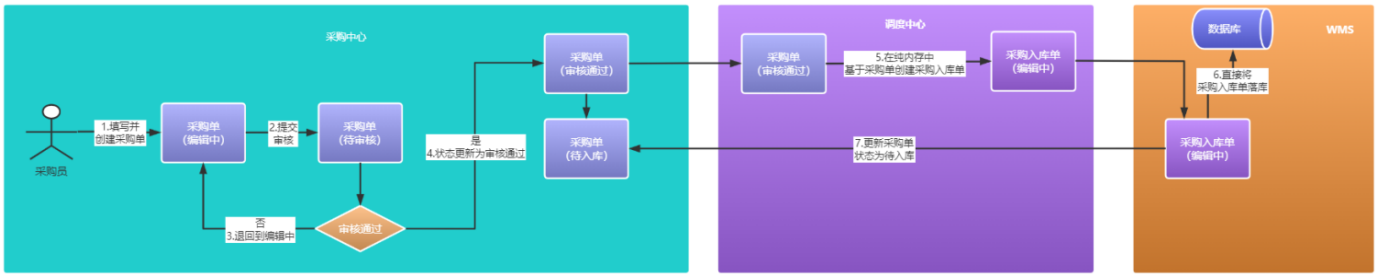


当采购需求、预算等条件都符合采购条件时，采购单就被审核通过了，这里审核采购单后就会触发调度中心的逻辑，调度中心会在内存中基于采购单的信息创建一个采购入库单，此时在调度中心中创建的采购入库单暂时还没有落库，仅仅是在内存层面创建采购入库单，如下图所示：



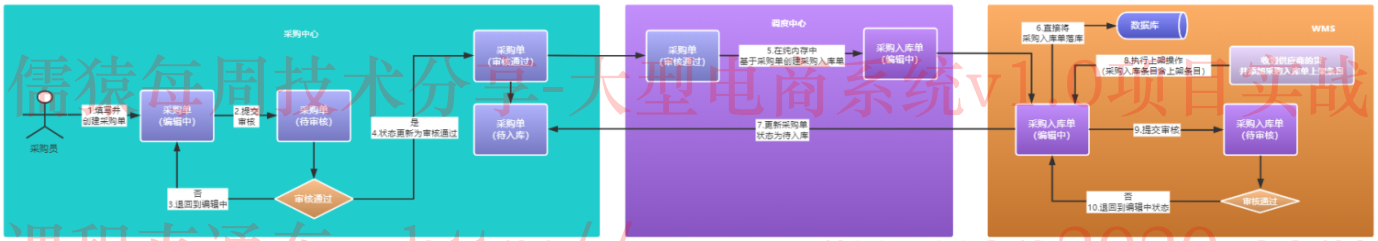
紧接着调度中心调用WMS的接口，把创建好的采购入库单传递给WMS，在WMS中才会把采购入库单给保存到数据库中，同时采购单的状态会被更新为待入库的状态。

目前采购中心有一张审核通过的采购单，WMS中有一张与之对应的采购入库单，负责采购的人员看到有一张审核通过的采购单，就会根据采购单信息去找供应商进货，而在仓库的工作人员同时也在等着供应商将货物运到仓库里，如下图所示：



仓库工作人员可能等了几天后，终于等到供应商把货物运送到了仓库中了，仓库中的工作人员这时打开系统中的采购入库单，一边把运来的货物上到货物架上，一边把具体什么货物、什么时间、上了几件、上到哪个货位架上，把这些信息同时给填写到采购入库单对应的上架条目信息栏中。

当货物全部上架成功后，采购入库单中的上架条目中就记录好了这些货物具体的上架信息了，此时货物上架的工作人员就提交采购入库单，下一步如果审核人发现实际到货的情况和采购单中的信息不符，就会把采购入库单的状态退回到编辑态，这时就需要再核对下收到的货物中哪里出了问题，是数量不对还是质量不合格等等，如下图所示：

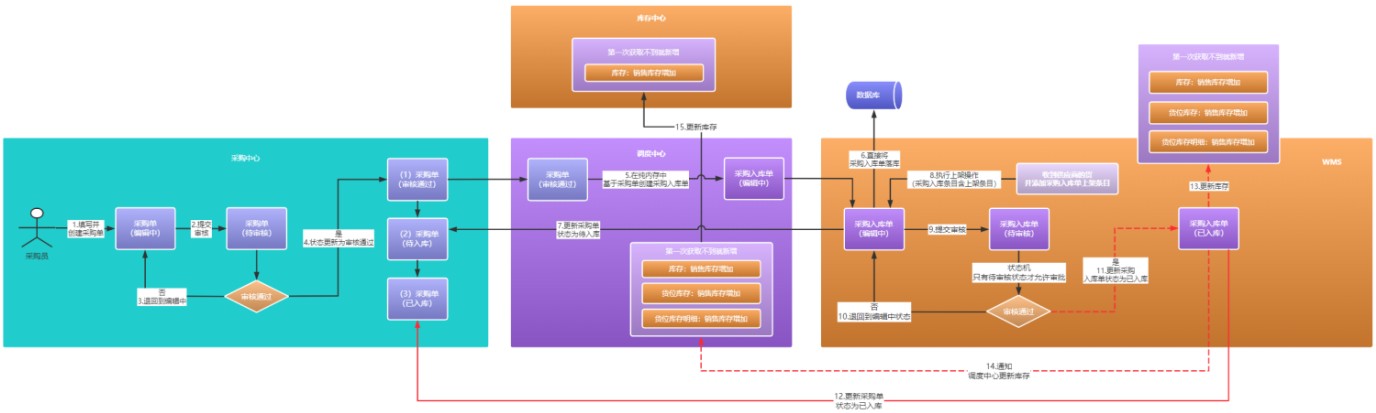


当解决完货物的问题后，仓库管理人员就会审批通过采购入库单，这时采购入库单的审批操作在电商系统后台系统中触发的操作比较多了：

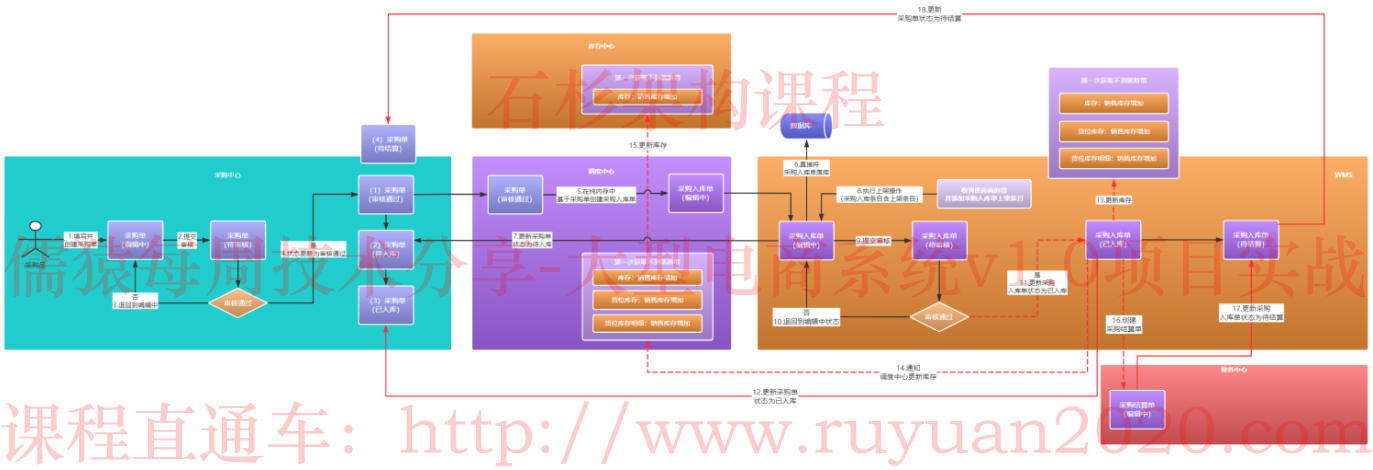
一方面要更新下采购单和采购入库单的状态，因为当前货物已经入库检验完毕，所以采购入库单状态从待入库状态流转为已入库状态，同时采购单的状态也从待入库状态流转为已入库状态，采购单和采购入库单的状态在关键的节点出要联动着更新。

另一方面就是库存信息的更新，这里的库存信息就涉及到了WMS的库存、调度中心的库存以及库存中心的库存。如果电商系统是第一次使用，WMS、调度中心以及库存中心对应的库存信息在最开始都是没有的，都是先要通过采购入库单的信息先创建出来的，比如库存中心需要创建商品库存信息，调度中心和WMS需要创建商品库存信息、商品货位库存信息和商品货位库存明细信息。

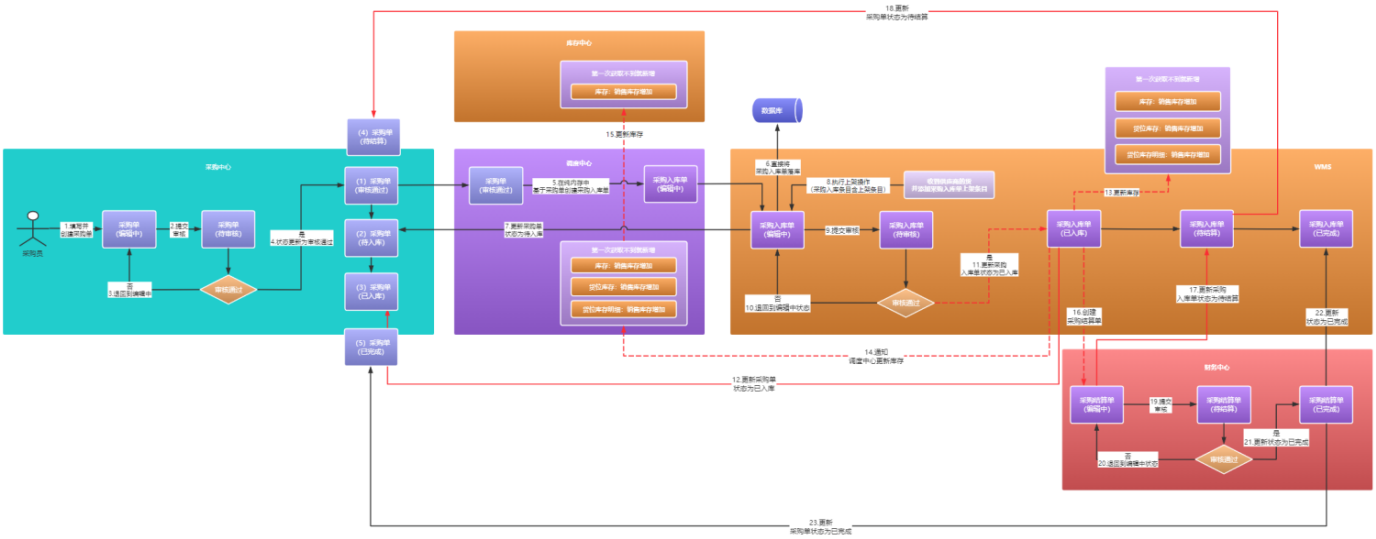
这里比较容易混淆的，就是在调度中心以及WMS中商品货位库存明细的信息是从哪来的，还记得当货物送到仓库时不是要把每件货物都一一上架吗，而之前已经将上架的详细信息填写到采购入库单上的上架条目中了，所以采购入库单上架条目就是货位库存明细信息的数据来源，如下图所示：



订单状态更新以及库存信息更新完成之后，最后还会发送请求到财务中心，根据审核通过的采购入库单信息创建一个采购结算单，此时整体的业务状态就是等待财务付款给供应商了，所以采购入库单、采购单的状态这里也联动着同步流转为待结算状态，如下图所示：

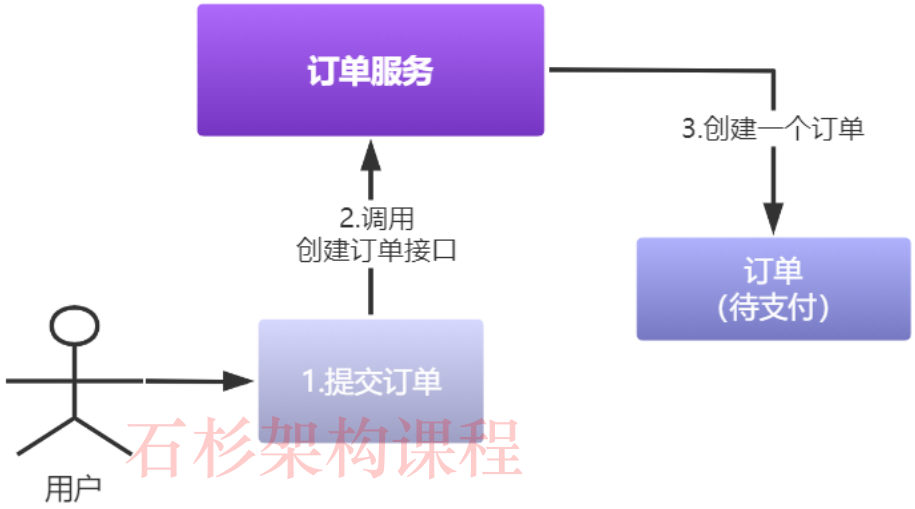


最后财务这边会审核采购结算单信息，看下采购了哪些商品需要付多少钱，确认无误后，会按月或者按季度审批后给供应商打款，采购结算单、采购入库单以及采购单的状态此时都会同时联动流转为已完成状态，至此整个采购链路的业务算是完成了，如下图所示：

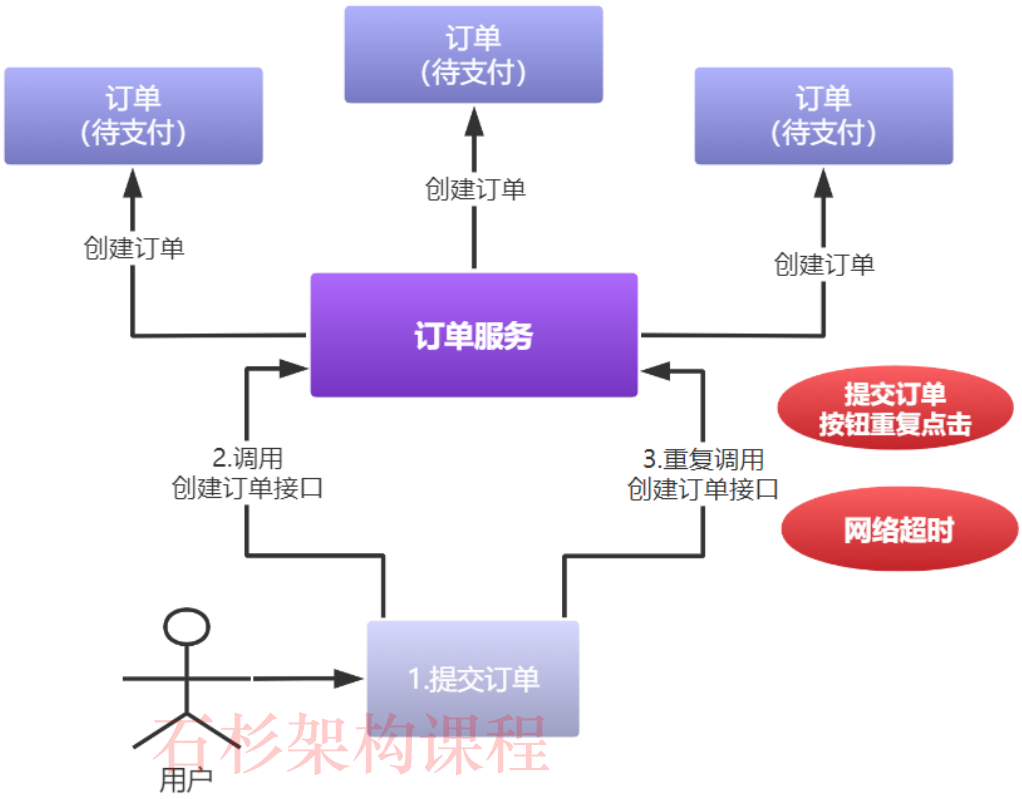


2.2 接口幂等性问题的介绍

在分析采购业务中的幂等性问题前，我们先了解下什么是接口幂等性问题呢？比如说我在某电商网站上选中了一个商品后提交订单，后台当然会暴露一个接口来被调用创建订单，那么正常情况下调用一次接口，后台就创建一个订单了，如下图所示：



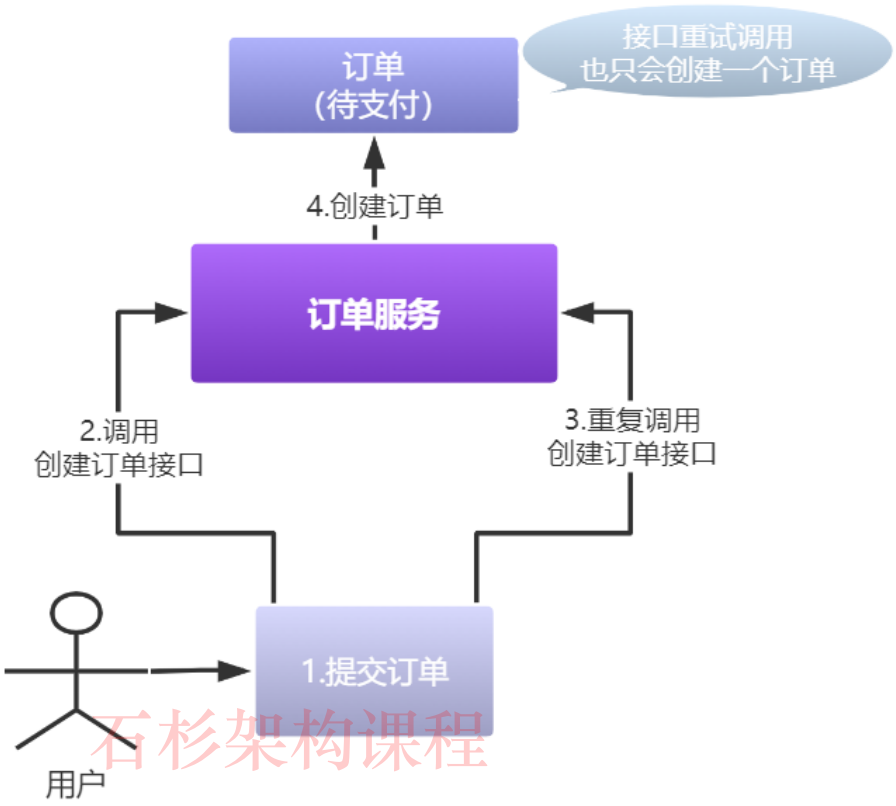
假如说你现在所处的网络环境不是很好，此时你提交订单了，创建订单的接口在网络不好时就容易调用超时，一般接口都有容错机制，比如超时重试机制，此时调用接口超时就可能被重复调用，导致同一件商品在同一时刻创建好几个订单；重复调用接口还可能有其他各种原因，比如前台页面响应不是很及时，导致用户在前台界面上疯狂点击提交按钮，也会重复调用创建订单的接口创建很多订单，如下图所示：



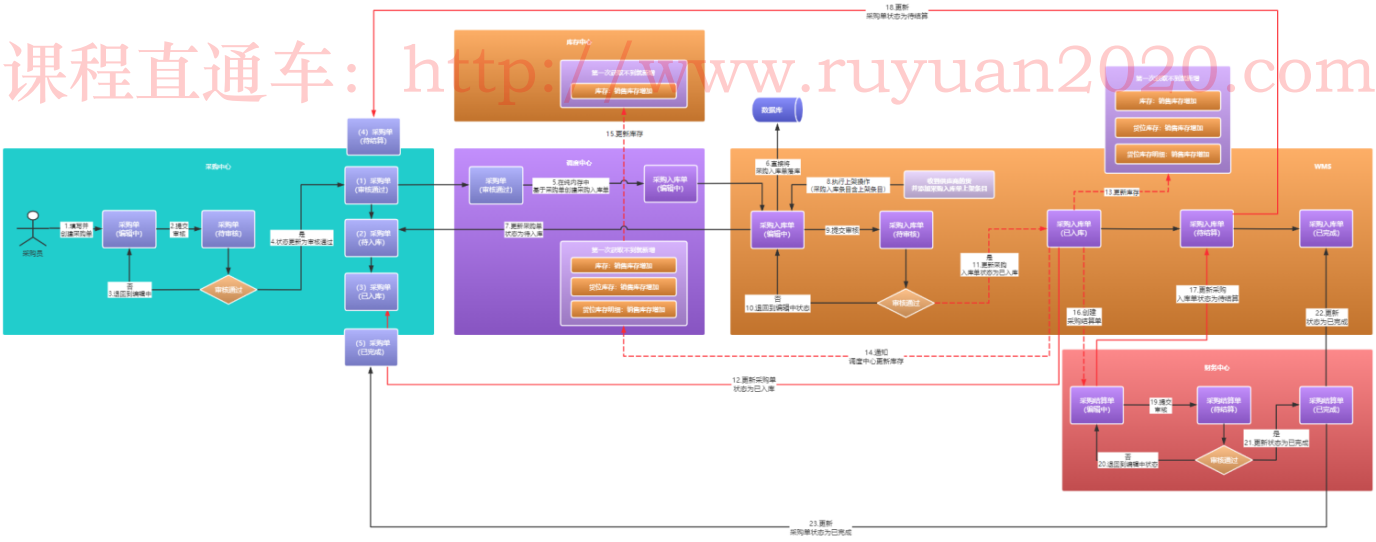
对于以上重复创建订单的情况肯定是不能接受的；如果说重复创建几个订单对于用户而言可能感触不是很深，那么接下来就是付款操作了，如果在用户付款操作时，调用扣款接口时也因为网络延迟重试，正常情况下只会扣一次的钱，结果接口被重试了好几次导致用户的钱多扣了几次，这下问题可就大了。

通过以上案例，大家差不多就发现了问题所在，就是一个接口如果不做一些校验机制直接让它裸奔执行的话，就会面临着被重复调用、接口逻辑重复被执行的糟糕结果。

接口幂等性保障，简单来说就是在调用接口时会先判断下接口之前有没有被调用过，如果没有被调用过当然可以执行，倘若发现接口已经被调用一次了，那么此时就要想方设法不能让同一逻辑被重复执行，这里处理的方式可以有很多种，比如方法没有返回值就直接return，或者直接报错都是可以的，简而言之就是接口要具备识别出已经执行过一次的逻辑就不能被重复执行了，如上面创建订单的案例，就算网络延迟重试了，也只允许创建一个订单，付款时也只允许扣款一次，如下图所示：



2.3 采购业务中幂等性问题分析



以下对采购业务中存在的接口幂等性问题分析，我们就结合上图标好的接口执行序号一步步分析：

1：创建采购单，这里一般不需要考虑幂等性问题，因为这里创建采购单是人为的在创建，并发量很低，并且就算重复创建采购单，后续审核的人也会及时发现。

2~4：这些操作都只是更新一下采购单的状态，只涉及到一个状态字段的更新，更新多次和更新一次的效果都是一样的，按照这样的思路，以上图中只要涉及到更新的操作，都不需要考虑接口幂等性的问题了。

5：这里是在内存中基于采购单创建采购入库单，可以忽略。

6：这里就会把在调度中心中，基于内存创建好的采购入库单给保存到WMS中的数据库中，这里就涉及到了数据库的新增操作，如果不加以控制，接口重复调用就会创建多个采购入库单，导致一个采购单对应多个采购入库单，这样是不合理的。

7~12：这些操作和2~4一样，只是简单的状态更新可以不用考虑，其中8主要是线下操作。

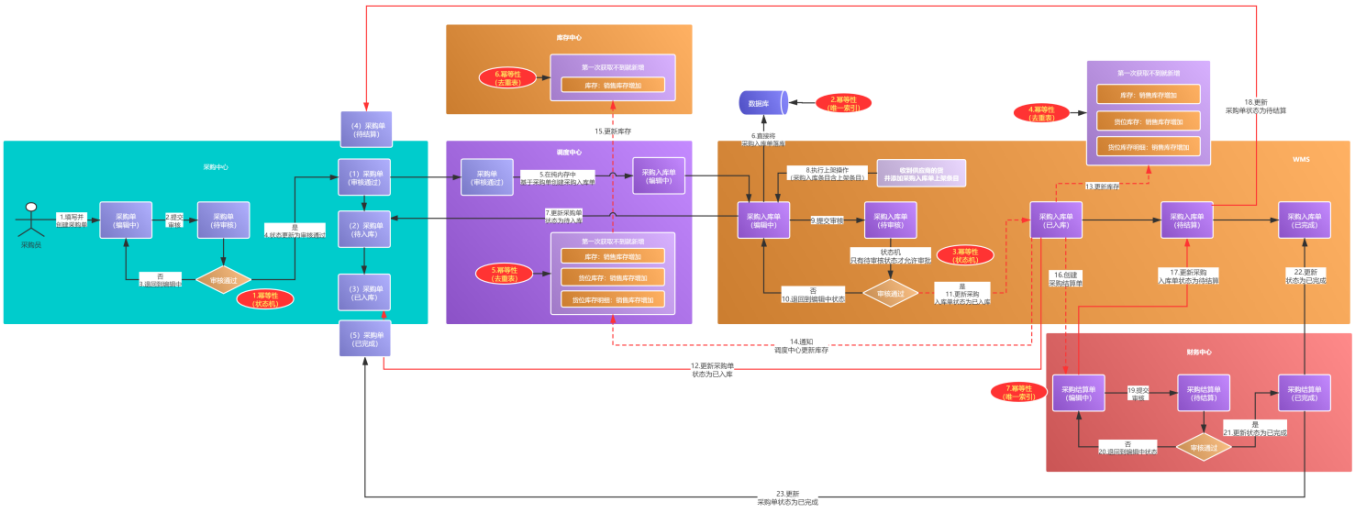
13、14、15：这三步操作都会更新库存的数量，这里的库存数量更新涉及到库存的计算，并不是简单像2~4和7~12一样只是简单更新一个字段。不管是库存中心、调度中心还是WMS，每次进来一批货物，对应的库存数量都会被累加的。假如说更新库存的接口重复调用了，比如本来只到货了10件商品，但是这块更新库存接口没有做幂等性保障，同时恰好遇到网络延时等原因导致接口被重复调用了10次，库存的数量本来累加10才正确，但是现在却要累加100，多出来的那90件商品在库存中压根就不存在，假如某一天用户下单，可能就会导致实际库存不够发不出货了。

16：审核采购入库单时，同时也会触发调用财务中心接口生成采购结算单，这里也存在着和第6步一样的数据库层面重复新增的问题，创建采购结算单接口重复被调用，可能就会导致一个采购入库单对应多个采购结算单，所以这里同样也有接口幂等性问题。

17~23：这些操作也只涉及一个状态字段的更新，接口被重复调用也没关系，无需考虑幂等性问题。

另外我们可以发现，往往是审核操作容易触发后续一系列的问题，比如图中采购单审核可能导致重复创建采购入库单问题、采购入库单审核可能导致库存信息重复累计的更新问题，所以我们可以尝试在审核操作做接口幂等性的保障，这样可以在源头上保证后续的操作不会重复执行。

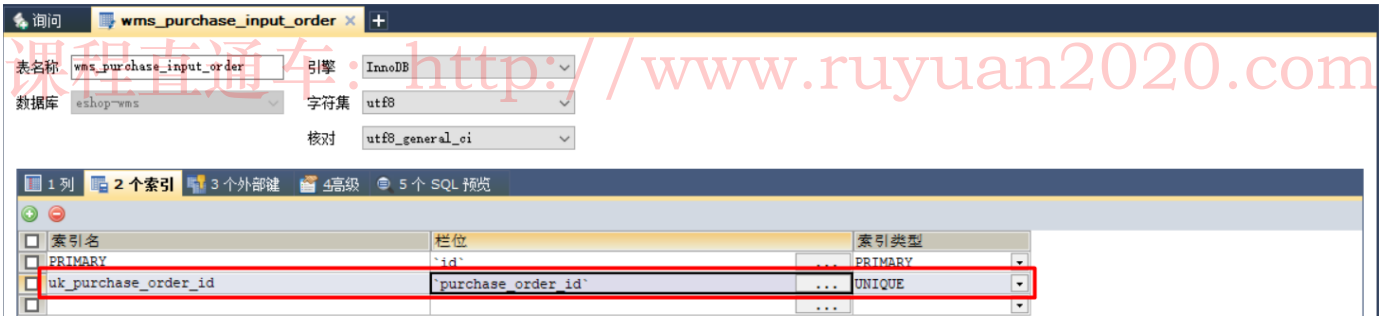
最后采购业务流程需要做接口幂等性保障的点如下图所示：



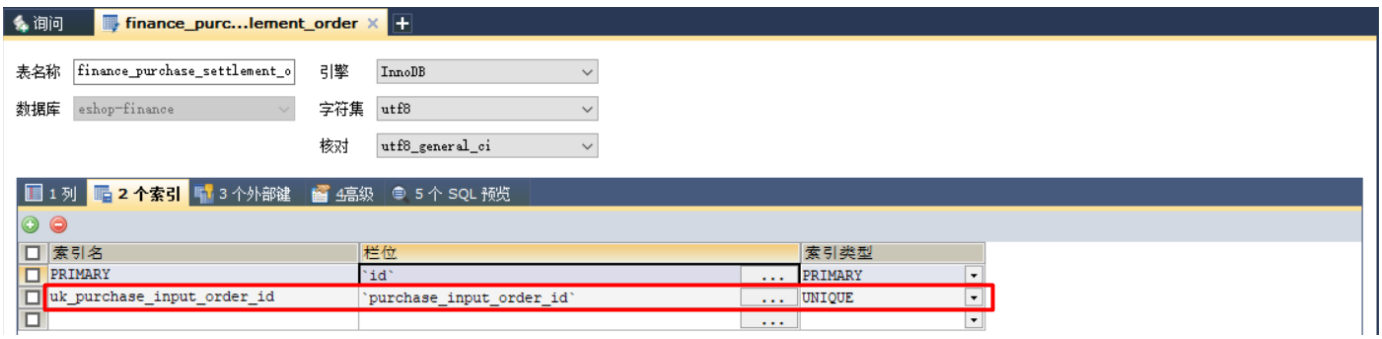
2.4 采购业务中接口幂等性问题解决方案

（1）唯一索引

根据以上采购流程的分析，我们可以看出第6步和第16步本质都是一类问题，就是在数据库中重复新增数据，对于该类问题最好的解决方案就是在数据库表中，创建唯一索引，比如第6步中，是新增采购入库单，那么在业务层面上，一个采购单只能唯一对应一个采购入库单，我们可以给采购入库单表中的采购单id加上唯一索引，如下图所示：



同理一个采购入库单只能对应一个采购结算单，可以在采购结算单表中对采购入库单id建立唯一索引，如下图所示：



（2）去重表

对于以上13、14、15这些更新库存的操作，这类操作不是简单的直接往数据库中新增数据，而且一般也不会存在数据重复的问题，不能简单的使用唯一索引来解决；更新操作往往基于数据库中现有的数据，经过一系列的逻辑计算后再把得到的结果给更新回数据库中，这个时候我们就要在接口代码开始执行时来预防操作重复执行。

可以通过去重表的方式，就是在数据库中创建一张表，这张表中除了主键id外只有一个unique_value的字段，该字段会建立唯一索引，建表语句如下所示：

唯一记录表

```
CREATE TABLE `unique_record` (
 `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '主键',
 `unique_value` varchar(255) NOT NULL COMMENT '唯一值',
 PRIMARY KEY (`id`),
 UNIQUE KEY `uk_unique_value` (`unique_value`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```


然后在执行接口逻辑最开始处，先往这张去重表中插入一条数据，如果接口已经执行过一次了，表中肯定会有一条记录，这时如果由于重试导致接口再次被调用，由于unique_value唯一索引会导致新增失败，从而接口后续的逻辑都无法执行，防止接口被重复调用执行同样的逻辑，如下为库存中心更新库存的核心代码：


```
/**
  * 通知库存中心，“采购入库完成”事件发生了
  * @param  purchaseInputOrderDTO 采购入库单DTO
  * @return  处理结果
  * @throws  Exception
  */
@Override
public Boolean informPurchaseInputFinished(
            @RequestBody PurchaseInputOrderDTO purchaseInputOrderDTO) throws Exception {
    uniqueRecordMapper.insert("InventoryService_informPurchaseInputFinished_"
                            + purchaseInputOrderDTO.getId());

    StockUpdater goodsStockUpdater = purchaseInputStockUpdaterFactory
                                    .create(purchaseInputOrderDTO)
```

```
    ;
15      goodsStockUpdater.updateGoodsStock();
16      return true;
17  }
```

去重表方案核心思路和唯一索引都是一样的，即对一个接口操作都要有一个唯一标识，唯一索引是在目标数据表中为唯一标识字段添加索引来达到去重的目的，而这里的去重表直接往数据库插入唯一标识字段，如果执行失败就说明接口已经执行过了，就会报错防止逻辑重复被执行，其本质都是一样的。

（3）状态机

采购流程中最后一类就是审批操作，如采购单审批、采购入库单审批和采购结算单审批，这类操作有个关键的特征就是执行操作之后，对象的状态会改变，如采购单审批后就从待审核状态流转为审核通过状态、采购入库单审核后从待审核状态流转为已入库状态等。

基于这类特征，我们可以使用状态机的方案来实现接口幂等性的保障，即每次执行接口前，必须先判断下当前的对象状态是否允许我们执行该操作，如采购入库单审核时，只有当状态为待审核状态才允许审核操作执行，这样的话假如之前接口已经执行过一次了，如果执行成功了采购入库单就变为待入库状态，否则执行失败采购入库单就回退到编辑中状态，反正一定不会为待审核状态，这样的话就算接口被重复调用，也能够根据状态的判定，来防止接口的核心逻辑被重复的执行，如下为审核采购入库单时关于状态校验的核心代码：


```
1  /**
2    * 对采购入库单进行审核
3    * @param  id 采购入库单id
4    * @throws  Exception
5    */
6  @PutMapping("/approve/{id}")
7  public Boolean approve(@PathVariable("id") Long id, Integer appro
   veResult) throws Exception {
8      try {
9          PurchaseInputOrderDTO purchaseInputOrderDTO = purchaseInp
   utOrderService.getById(id);
10         if (purchaseInputOrderDTO == null
11                 || !PurchaseInputOrderStatus.WAIT_FOR_APPROVE
12                         .equals(purchaseInputOrderDTO.getStatus()
   )) {
13             return false;
14         }
15
```

```
16          purchaseInputOrderService.approve(id, approveResult);
17
18          if(PurchaseInputOrderApproveResult.PASSED.equals(approveR
    esult)) {
19              PurchaseInputOrderDTO purchaseInputOrder = purchaseIn
    putOrderService.getById(id);
20              PurchaseInputOrderHandler handlerChain = handlerFacto
    ry.getHandlerChain();
21              handlerChain.execute(purchaseInputOrder);
22          }
23
24          return true;
25      } catch (Exception e) {
26          logger.error("error", e);
27          return false;
28      }
29  }
```


3.面试题剖析

什么是接口幂等性？接口幂等性问题有哪些常见的解决方案？

（1）基于以上的采购核心链路的分析，对于什么是接口幂理解的应该非常深刻了，接口幂等性简单来说就是本来应该执行一次接口操作，不论是因为一些人为原因也好、网络不佳系统故障原因也好，接口就是被重复调用了，那么添加了接口幂等性保障的接口，重复调用时就会自动识别你这次来执行已经不是第一次了，就一定不会让你执行成功。

（2）常见的接口幂等性问题解决方案，在采购核心链路中已经应用了三种了，唯一索引：在数据表中对一些具有唯一性业务属性的字段建立唯一索引；去重表：每次执行接口逻辑前先往去重表中插入一条唯一记录；状态机：根据当前的对象的状态判断当前是否可以执行接口的逻辑。

其中去重表的方案，暴露的问题就比较明显，因为它的本质就是往数据库中先插入一条数据，数据库插入操作是非常不适用于高兵发的场景的，如mysql数据库的单机能承载的并发也才几千，当并发高达几十万甚至上百万时，mysql数据库直接就被打垮了，此时就需要换一些能承载高兵发的系统如redis，我们完全可以把插入一条数据的操作给写到redis中，这样既同样也能达到去重的目的，同时也还能承载高并发的业务场景。