问题背景

屈臣氏现在总共有4w多事务需要一次性点击全选创建按钮创建开票结算单，执行失败后，与屈臣氏沟通由供应商改为2w一次执行，也没跑过去，目前的报错在脚本内存超出，当前的逻辑是：

1. 生产者执行逻辑：按照条件查询数据，对这部分数据打上状态标记，并记录批次，根据批次发出mq消息
2. 消费者执行逻辑：
   1. 消费消息,根据批次查询数据
   2. 屈臣氏二开校验，传入了2.1中查询到的所有事务（通过埋点实现）
   3. 超过1.5w进行校验【原意是为了校验手工逻辑】
   4. 进入开票结算单创建逻辑按照规则并单/拆单生成结算单【业务上固定执行拆分】
   屈臣氏的问题现在出现在了2.2，超过内存大小了

其中生产者执行耗时：

```
生产者查询到20094行事务后打上标记，总共耗时45.2s
1、按照查询条件查询可开票事务：2024-02-01 10:23:31.201->2024-02-01 10:23:43.026 总共耗时：11.8s
2、校验是否可执行：2024-02-01 10:23:44.120->2024-02-01 10:24:06.980 总共耗时：22.9s [可优化为不执行db查询]
3、更新时间：2024-02-01 10:24:07.679 -> 2024-02-01 10:24:15.904 总共耗时：8.2s
```

目标

1. 支持用户全选创建最少2w事务一次创建结算单，尽量支持4w事务一次创建结算单
2. 原手工勾选创建校验最多1.5w行，且校验最多行的结算单行数为1.5w，该逻辑不变
3. 不能出现错误分批情况

解决方案

方案一

1. 生产者执行逻辑：
   1. 按照条件查询数据，对这部分数据更新状态为系统处理中
   2. 参考本方案2.4获取对应的并单/拆单规则，尽量查询字段以及查询db次数
   3. Java层对数据进行拆分批次，如果拆过1.5w，则继续拆分，记录批次(如果存在客户挑战该逻辑，天然手工勾选也存在该校验)
   4. 根据批次发出mq消息
2. 消费者执行逻辑：
   1. 消费消息,根据批次查询数据
   2. 屈臣氏二开校验，传入了2.1中查询到的所有事务（通过埋点实现）
   3. 超过1.5w进行校验【原意是为了校验手工逻辑】
   4. 进入开票结算单创建逻辑按照规则并单/拆单生成结算单【业务上固定执行拆分】
   优点：
3. 本次改动相对较小，本次只需要改动生产者发消息逻辑，2w数据预计耗时可以在35-40秒完成；
4. 可以在创建前就进行拆分，可以尽量避免后续创建链路超过埋点大小以及oom
   缺点：
5. 按照【公司、供应商、币种】进行预拆分后，超过1.5w可能还需要拆分，用户不一定能接受；
6. 需要检查二开调用
7. 1.5w个事务记录执行进入脚本，不一定可以保证不会超过内存限制 【目前脚本内存阈值为64MB，屈臣氏2w行事务，序列化后占用内存大小约为75MB，dev测试甄云标准1.5w约占用内存43MB，主要与扩展字段也有关系】

方案二

1. 现有逻辑中的屈臣氏二开校验改为通过批次作为参数查询数据,可以通过新增一个埋点执行
2. 原消费者执行逻辑中针对全选创建去掉超过1.5行校验【通过后续操作中的提交动作进行校验】
   优点：改动相对较小
   缺点：
3. 多个埋点都有超出内存问题风险，需要整改链路上的埋点都进行修改，目前最少是改动两个埋点；

4. 未来新增埋点也需要这么处理，比较耗费成本，容易出问题
5. 需要检查二开调用

方案三

通过并发管理器（调度任务）中转
优点：

1. 可以提供界面查看进度
   缺点：
2. 代码改动较大
3. 与目前通过mq执行存在类似的问题，也需要进行优化

最终方案

1. 临时通过产研拉取数据，客户按照条件查询存在大量数据的情况，进行全选创建
2. 开票结算单优化方案：按照方案一执行
3. 对账单、付款结算单参考方案一执行