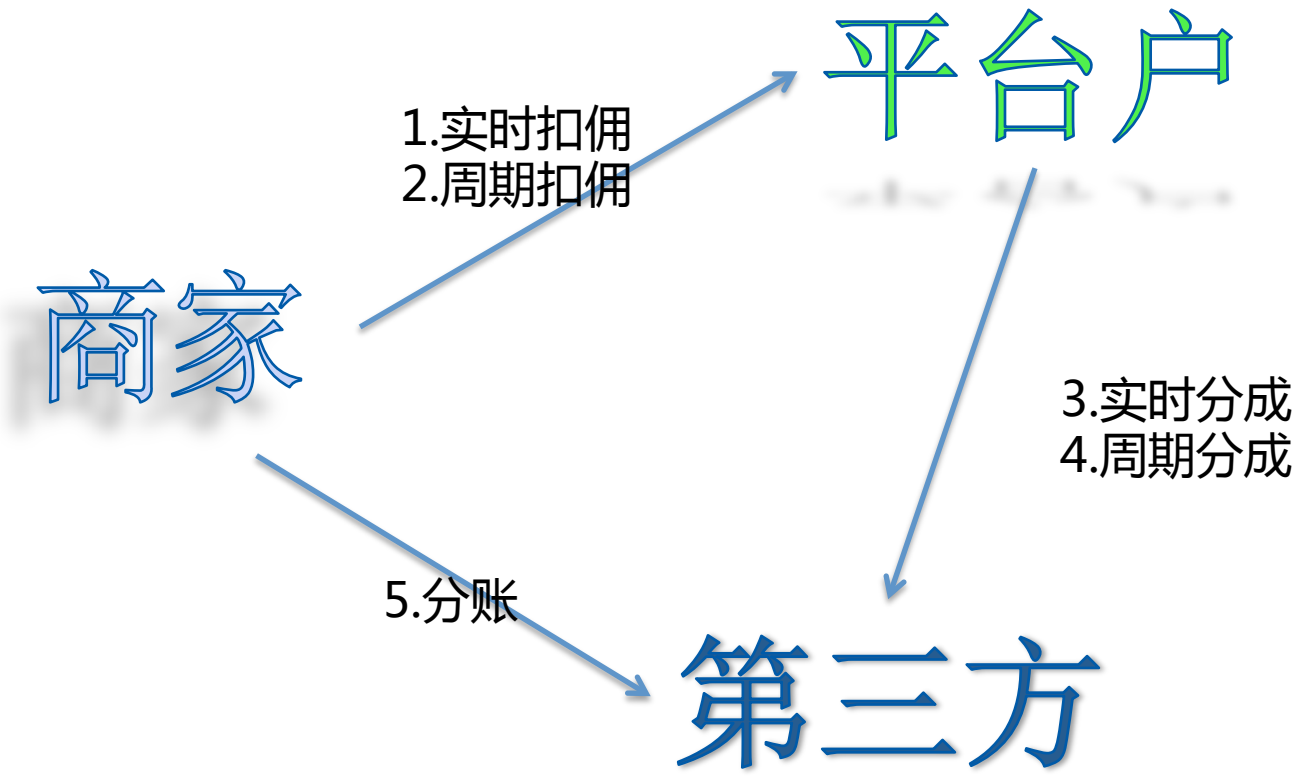
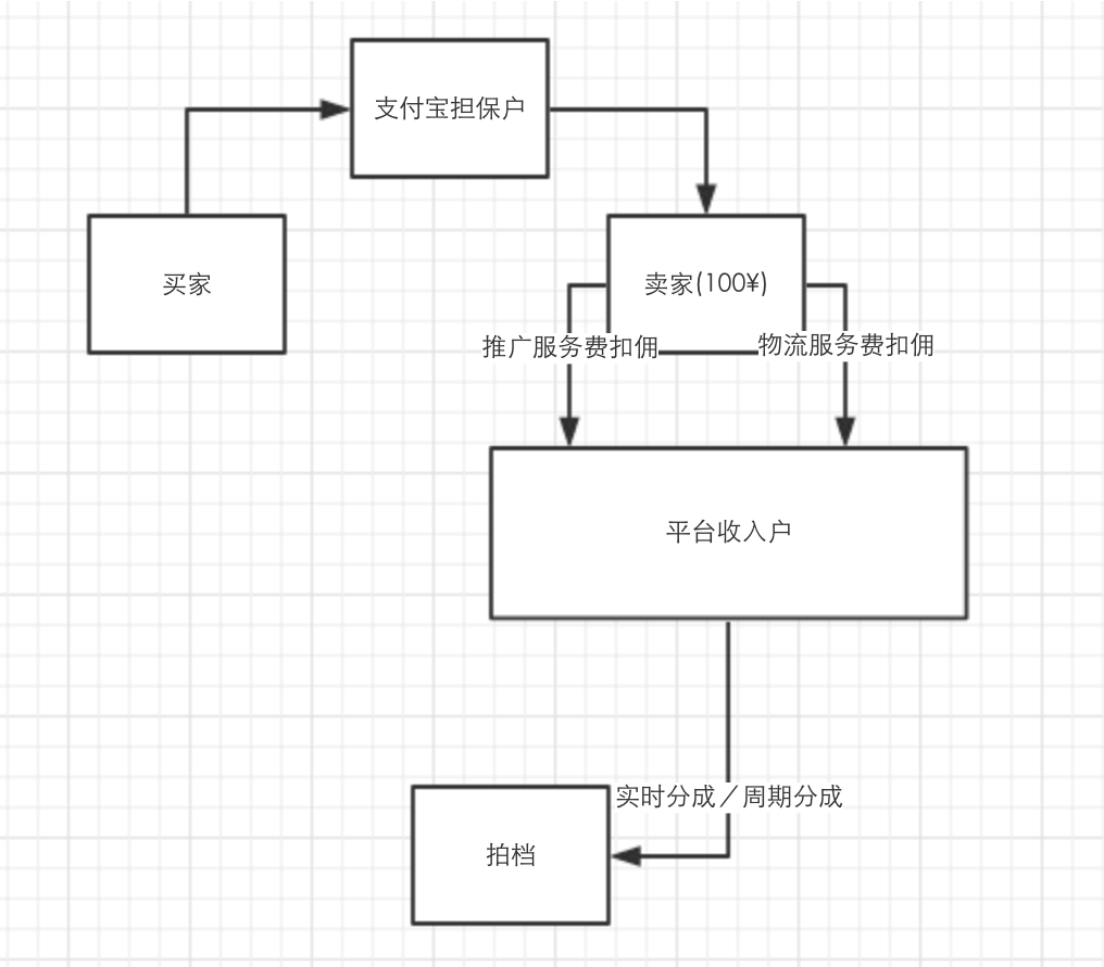
结算域设计

by xiangmao.lxm

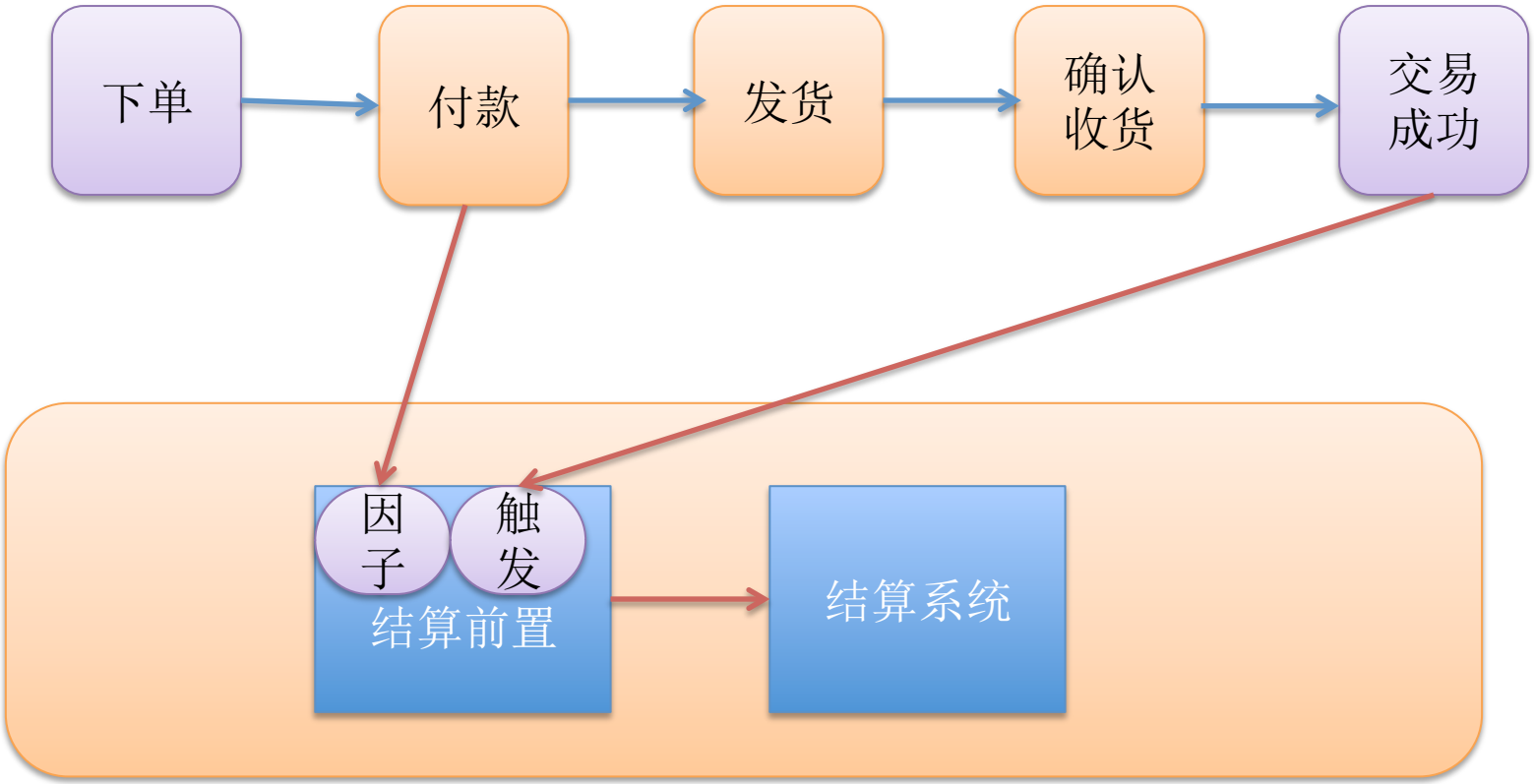
一. 经典结算模式



二. 经典资金流

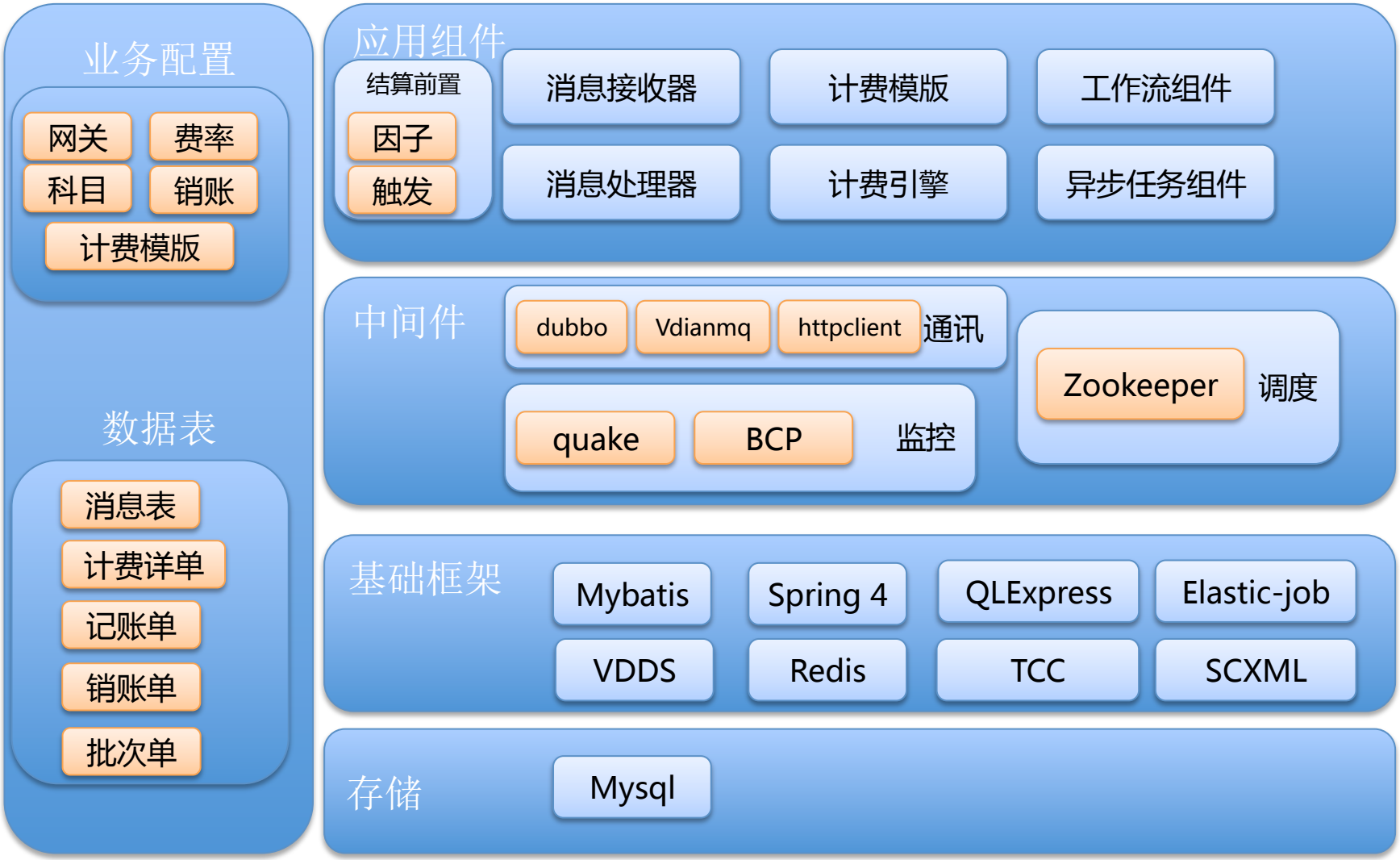


三. 状态流转

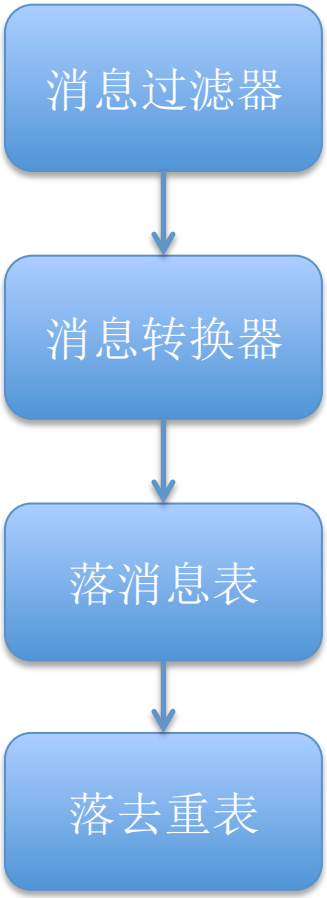


四. 微店结算系统业务架构

五. 微店结算系统技术架构



4.1. 网关



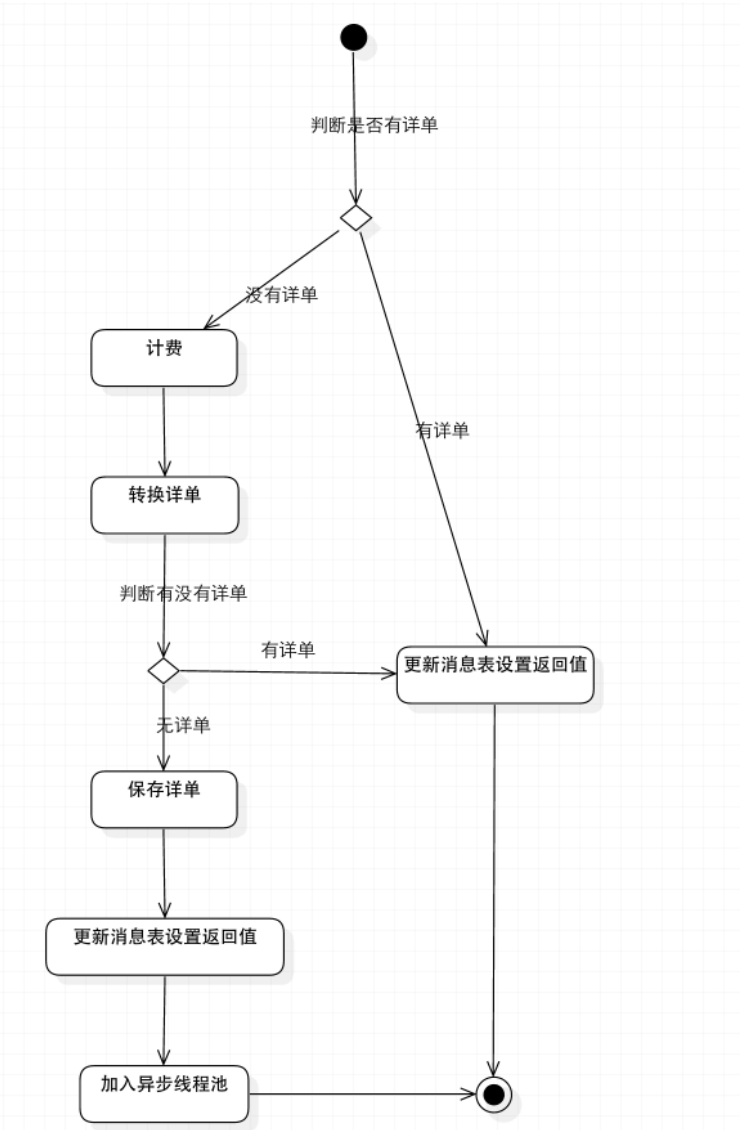
无论是同步调用还是异步调用，根据业务类型过滤掉不需要扣佣的业务

消息转换是将不同业务方透传过来的扣拥请求转换为统一的计费请求对象: ⚠**主订单还是子订单**

失败重试

幂等控制

4.2 结算


判断是否有详单
没有详单
有详单
计费
转换详单
判断有没有详单
有详单
更新消息表设置返回值
无详单
保存详单
更新消息表设置返回值
加入异步线程池

4.3 费率

费率规则

表达式	优先级
OFFER_ID==?	10
LEAT_CAT_ID==?	20
BASE_LINE==?	30

费率规则实例

表达式	费率
OFFER_ID==123	{rate:0.01}
LEAT_CAT_ID==456	{fixedAmount:100}
BASE_LINE==100	…

4.4 计费模版

```
        rate = ((FixRate) 当前分成方费率规则).getRate();
        当前分成方分成金额 = 请求金额.multiply(rate).setScale(2, RoundingMode.HALF_UP);
        当前分成方比例 = rate;
        /**固定金额类型**/
    } else if (当前分成方费率规则.getType().getType()==2) {
        当前分成方分成金额 = ((FixAmount) 当前分成方费率规则).getAmount().abs();
        /**避免divide by zero问题**/
        if (请求金额.compareTo(new BigDecimal(0)) == 0) {
            当前分成方比例 = 0;
        } else {
            当前分成方比例 = 当前分成方分成金额.divide(请求金额,2,RoundingMode.HALF_UP);
        }
    } else {
        exception("unsupported rate type~~, rate type : " + 当前分成方费率规则.getType().getType());
    }
    剩余金额 = 剩余金额.subtract(当前分成方分成金额);
    剩余分成比例 = 剩余分成比例.subtract(当前分成方比例);
    if (剩余金额.compareTo(new BigDecimal(0)) == -1) {
        exception("fc amount not enough~~, remain amount : " + 剩余金额);
    }
    if (剩余分成比例.compareTo(new BigDecimal(0)) == -1) {
        剩余分成比例 = new BigDecimal(0);
    }

    计费明细 = new StdFcRatingDetailATO();
    计费明细.quantity = 请求金额;
    计费明细.amount = 当前分成方分成金额;
    计费明细.proration = 当前分成方比例;
    计费明细.fcCode = 分成code;
    计费结果.addRatingDetail(计费明细);
```