扣项类型维护

```sql
CREATE TABLE `fi_kx` (
  `series` bigint(20) NOT NULL DEFAULT '0' COMMENT '行号',
  `tenant_num_id` bigint(20) DEFAULT '0' COMMENT '租户ID',
  `data_sign` tinyint(4) DEFAULT '0' COMMENT '测试标识',
  `create_dtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `last_updtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '最后更新时间',
  `create_user_id` bigint(20) DEFAULT '0' COMMENT '用户',
  `last_update_user_id` bigint(20) DEFAULT '0' COMMENT '更新用户',
  `cancelsign` char(1) DEFAULT 'N' COMMENT '删除',
  `insertdata` char(1) DEFAULT 'Y' COMMENT '新增',
  `updatedata` char(1) DEFAULT 'N' COMMENT '更新',
  `iscfp` tinyint(4) DEFAULT '1' COMMENT '是否冲发票',
  `kx_num_id` bigint(20) DEFAULT '0' COMMENT '扣项代码',
  `kx_name` varchar(225) DEFAULT ' ' COMMENT '扣项名称',
  `kx_type` tinyint(4) DEFAULT '1' COMMENT '扣项类型',
  `accno` varchar(255) DEFAULT ' ' COMMENT '对应科目',
  `direction` tinyint(4) DEFAULT '0' COMMENT '科目方向',
  `kx_kk_type` tinyint(4) DEFAULT '0' COMMENT '扣项交款标志',
  `calc_flag` tinyint(4) DEFAULT '0' COMMENT '计算标志',
  `fee_type` tinyint(4) DEFAULT '0' COMMENT '费用类型',
  `income_type` tinyint(4) DEFAULT '0' COMMENT '收入类型',
  `fraction` tinyint(4) DEFAULT '0' COMMENT '扣项金额',
  `tax_rate` decimal(10,2) DEFAULT '0.00' COMMENT '税率',
  `sales_return_flag` tinyint(4) DEFAULT '2' COMMENT '退货处理标记：0-忽略，1-增加，2-扣减',
  `apply_flag` tinyint(4) DEFAULT '0' COMMENT '应用标识：0-总部+门店，1-仅总部，2-仅门店',
  `auto_delay_flag` tinyint(4) DEFAULT '0' COMMENT '是否自动延期：0-否，1-是',
  PRIMARY KEY (`series`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

固定扣项计算：

`com.ykcloud.soa.erp.api.fi.service.FiKxService.calcKx`

`com.ykcloud.soa.erp.fi.service.impl.calcKx`

1. 打印日志消息

```java
if (log.isDebugEnabled()) {log.debug("begin calcKx request:{}",
JsonUtil.toJson(request));}
```

2. 参数校验

```java
request.validate(Constant.SUB_SYSTEM, ErpExceptionType.VCE13006);
```

3. 获取请求参数信息

4. 生成lockKey

```
StringBuilder sb = new StringBuilder("ykcloud.fi.kx.gen_");
sb.append(tenantNumId).append("_").append(dataSign).append("_").append(subUnitNumId
).append("_")
.append(DateUtils.format(sellDate));
String distributedLockKey = sb.toString();
```

5. 加锁

```
lock = new RedisLock(stringRedisTemplate, distributedLockKey, 60 * 20);
if (!lock.lock()) {
throw new ValidateBusinessException(Constant.SUB_SYSTEM,ErpExceptionType.VBE23006,
"结算门店: " + subUnitNumId + ", 日结日期: " + DateUtils.format(sellDate) + "的供应商
扣项正在生成中, 请稍候...");
}
```

6. 判断新方案是否已配置启用日期

```
String kxSwitchDateVal = fiInnerService.getConfigValue(tenantNumId, dataSign,
"kx_switch_date");
if (StringUtils.isBlank(kxSwitchDateVal)) {
throw new ValidateBusinessException(Constant.SUB_SYSTEM,
ErpExceptionType.VBE23006,"请先配置新的扣项计算方案的启用日期[参数:
kx_switch_date]! ");
}
```

7. 生成新的扣项计算方案启用日期

```
Date kxSwitchDate;
            try {
                kxSwitchDate = DateUtils.parse(kxSwitchDateVal);
            } catch (IllegalArgumentException ex) {
                throw new ValidateBusinessException(Constant.SUB_SYSTEM,
ErpExceptionType.VBE23006,
                        "请按照yyyy-MM-dd格式配置新的扣项计算方案的启用日期! ");
            }
```

8. 查询扣项政策，不分门店、大类，生成的结果还是要区分到门店、大类、部类、供应商（重复维护的扣项政策取审核日期最近的）

```
List<VenderKx> kxList;
List<Long> applyFlagList = new ArrayList<>();
applyFlagList.add(KxApplyFlagEnum.HEAD_AND_SHOP.getValue());
// 获取总部编号
Long headSubUnitNumId = fiInnerService.getHeadSubUnitNumId(tenantNumId, dataSign,
subUnitNumId).getHeadSubUnitNumId();
```

```java
if (headSubUnitNumId.equals(subUnitNumId)) {
    // 总部
    applyFlagList.add(KxApplyFlagEnum.ONLY_HEAD.getValue());
} else {
    // 门店
    applyFlagList.add(KxApplyFlagEnum.ONLY_SHOP.getValue());
}
// 忽略政策失效日期
kxList = fiVenderKxDtlDao.findKxListWithoutExpired(tenantNumId, dataSign, sellDate, applyFlagList);
```

9. 遍历计算每一个扣项

- 查询扣项完整信息

```java
kx = fiVenderKxDtlDao.getKxFullInfo(tenantNumId, dataSign,kx.getSupplyUnitNumId(), kx.getReservedId(), kx.getKxNumId(), kx.getKxType(),kx.getKxDirection(), kx.getKxKkType(), kx.getKxStyle());
```

- 获取**扣项计算方法**（按月，年或季度）和**扣项类型**（按**销售收入，销售成本，验收成本，净收货成本**或**验收件数**）

```java
KxStyleEnum kxStyle = KxStyleEnum.fromValue(kx.getKxStyle());
    if (kxStyle == null) {
        throw new ValidateBusinessException(Constant.SUB_SYSTEM, ErpExceptionType.VBE23006,
            "扣项政策中维护的计算方式：" + kx.getKxStyle() + "不是有效的计算方式，无法确定业务发生额的统计周期！");
    }
    KxTypeEnum kxType = KxTypeEnum.fromValue(kx.getKxType());
    if (kxType == null) {
        throw new ValidateBusinessException(Constant.SUB_SYSTEM, ErpExceptionType.VBE23006,
"扣项政策中维护的计算类型：" + kx.getKxStyle() + "不是有效的计算类型，无法确定业务发生额的统计来源！");
    }
```

- 得到计算周期清单，从生效日期开始逐期计算，差异部分需要生成差异扣项

```java
Date kxEndDate = DateUtils.daysBetween(sellDate, kx.getEndDate()) >= 0 ? sellDate : (KxAutoDelayFlagEnum.fromValue(kx.getAutoDelayFlag()) == KxAutoDelayFlagEnum.YES ? sellDate : kx.getEndDate());
List<DatePeriod> periodList = this.getPeriodList(kx.getBeginDate(), kxEndDate, KxStyleEnum.fromValue(kx.getKxStyle()));
```

```java
/**
    * 生成计算周期清单
    *
    * @param beginDate   扣项政策生效日期
    * @param sellDate    日结日期
    * @param kxStyleEnum 计算方式：按年、按月、按季度
    * @return
    */
private List<DatePeriod> getPeriodList(Date beginDate, Date sellDate, KxStyleEnum kxStyleEnum) {
        List<DatePeriod> periodList = new ArrayList<>();
        switch (kxStyleEnum) {
        case MONTHLY:
            Date monthEndDate = DateUtils.parse(DateUtils.format(DateUtils.getEndDateTimeOfMonth(beginDate)));
            while (DateUtils.daysBetween(monthEndDate, sellDate) >= 0) {
                DatePeriod period = new DatePeriod();
                period.setBeginDate(DateUtils.getStartDateTimeOfMonth(monthEndDate));
                period.setEndDate(monthEndDate);
                periodList.add(period);
                monthEndDate = DateUtils
                    .parse(DateUtils.format(DateUtils.getEndDateTimeOfMonth(DateUtils.addMonths(monthEndDate, 1))));
            }
            break;
        case QUARTERLY:
            Date quarterlyEndDate = DateUtils.parse(
                DateUtils.format(com.ykcloud.soa.erp.common.utils.DateUtils.getEndDateTimeOfQuarter(beginDate)));
            while (DateUtils.daysBetween(quarterlyEndDate, sellDate) >= 0) {
                DatePeriod period = new DatePeriod();
                period.setBeginDate(
                    com.ykcloud.soa.erp.common.utils.DateUtils.getStartDateTimeOfQuarter(quarterlyEndDate));
                period.setEndDate(quarterlyEndDate);
                periodList.add(period);
                quarterlyEndDate = DateUtils.parse(DateUtils.format(com.ykcloud.soa.erp.common.utils.DateUtils
                    .getEndDateTimeOfQuarter(DateUtils.addMonths(quarterlyEndDate, 3))));
            }
            break;
        case PER_YEAR:
            Date yearEndDate = DateUtils.parse(
                DateUtils.format(com.ykcloud.soa.erp.common.utils.DateUtils.getEndDateTimeOfYear(beginDate)));
            while (DateUtils.daysBetween(yearEndDate, sellDate) >= 0) {
                DatePeriod period = new DatePeriod();
                period.setBeginDate(com.ykcloud.soa.erp.common.utils.DateUtils.getStartDateTimeOfYear(yearEndDate));
                period.setEndDate(yearEndDate);
                periodList.add(period);
                yearEndDate = DateUtils.parse(DateUtils.format(com.ykcloud.soa.erp.common.utils.DateUtils
.getEndDateTimeOfYear(DateUtils.addMonths(yearEndDate, 12))));
            }
            break;
        }
        return periodList;
    }
```

- 遍历计算周期清单，如果周期的结束日期小于等于日结日期，代表已经可以计算
- 如果周期的结束日期大于等于启用日期，按新的计算方案处理，否则按原来的计算方案处理

```
if (CollectionUtils.isNotEmpty(periodList)) {
    for (DatePeriod period : periodList) {
    // 周期的结束日期如果小于等于日结日期，则代表已经可以计算
        if (DateUtils.daysBetween(period.getEndDate(), sellDate) >= 0) {
            if (DateUtils.daysBetween(kxSwitchDate, period.getEndDate()) >= 0) {
                // period.getEndDate() >= 启用日期
                // 按新的计算方案处理
                calcKxNew(tenantNumId, dataSign, subUnitNumId, sellDate,userNumId, kx, kxType, period);
            } else {
                // 按原来的计算方案处理
                calcKxOld(tenantNumId, dataSign, subUnitNumId, sellDate,userNumId, kx, kxType, period);
            }
        }
    }
}
```

- 如果扣项政策有误，无法统计业务发生额，一个周期都没有计算（发生异常）
- 保存扣项计算日志 `fi_vender_kx_log`

```
catch (Exception ex) {
                        // 扣项政策维护有误，无法统计业务发生额，一个周期都没有计算
                        log.error(ex.getMessage(), ex);
                        VenderKxResult kxResult = new VenderKxResult();
                        kxResult.setSubUnitNumId(subUnitNumId);
                        kxResult.setPtyNum1(0L);
                        kxResult.setCalculateDate(sellDate);
                        kxResult.setBeginDate(null);
                        kxResult.setEndDate(null);
                        kxResult.setBusinessAmount(0D);
                        kxResult.setKxAmount(0D);
                        kxResult.setSuccessSign(0L);
                        kxResult.setCutSaveSign(0L);
                        kxResult.setRemark(ex.getMessage());
                        kxResult.setUserNumId(userNumId);
                        this.saveKxCalcLog(kx, kxResult);
                    }
```

```
private void saveKxCalcLog(VenderKx kx, VenderKxResult kxResult) {
        FI_VENDER_KX_LOG kxLog = new FI_VENDER_KX_LOG();
        kxLog.setTENANT_NUM_ID(kx.getTenantNumId());
        kxLog.setDATA_SIGN(kx.getDataSign());
        kxLog.setSUB_UNIT_NUM_ID(kxResult.getSubUnitNumId());
        kxLog.setSUPPLY_UNIT_NUM_ID(kx.getSupplyUnitNumId());
        kxLog.setKX_NUM_ID(kx.getKxNumId());
        kxLog.setRESERVED_ID(kx.getReservedId());// 政策单号
        kxLog.setKX_TYPE(kx.getKxType());
        kxLog.setKX_DIRECTION(kx.getKxDirection());
        kxLog.setKX_KK_TYPE(kx.getKxKkType());
        kxLog.setKX_STYLE(kx.getKxStyle());
        kxLog.setRANGE1(KxRangeEnum.PTY_NUM_1.getValue());
        kxLog.setRANGE_ID(kxResult.getPtyNum1());
        kxLog.setFLAGTYPE(0L);// 废弃
        kxLog.setCALCULATE_DATE(kxResult.getCalculateDate());
        kxLog.setBEGIN_DATE(kxResult.getBeginDate());
        kxLog.setEND_DATE(kxResult.getEndDate());
        kxLog.setBUSINESS_AMOUNT(kxResult.getBusinessAmount());
        kxLog.setKX_AMOUNT(kxResult.getKxAmount());
        kxLog.setSUCCESS_SIGN(kxResult.getSuccessSign());
        kxLog.setCUT_SAVE_SIGN(kxResult.getCutSaveSign());
        kxLog.setREMARK(kxResult.getRemark());
        kxLog.setCREATE_USER_ID(kxResult.getUserNumId());
        kxLog.setLAST_UPDATE_USER_ID(kxResult.getUserNumId());
        kxLog.setCUT_RESERVED_NO(kxResult.getCutReservedNo());
        fiVenderKxLogDao.insertEntity(kxLog);
    }
```

10. 新的计算方案 `calcKxNew`

- 获取门店业务发生额(按大类分组) `getBusinessAmountByPtyNum1List`

```
List<BusinessAmountBySubUnitAndPtyNum1> businessAmountBySubUnitAndPtyNum1List = this.getBusinessAmountByPtyNum1List(tenantNumId, dataSign, subUnitNumId, kx, kxType, period);
```

统计来源是销售出库日报的情况

```
if (kxType == KxTypeEnum.SALES_INCOME || kxType == KxTypeEnum.SALES_COST) {// 统计来源是销售出库日报的情况
            List<Long> subUnitNumIds = fiInnerService.getSubUnitNumIdBySuperUnitNumId(tenantNumId, dataSign, subUnitNumId);
            if (CollectionUtils.isNotEmpty(subUnitNumIds)) {
                // 所有下游门店的数据
                for (Long subSubUnitNumId : subUnitNumIds) {
                    Date latestOverDate = fiInnerService.getSalesDateBySubUnitNumId(tenantNumId, dataSign,
                        subSubUnitNumId);// 获取门店日结到了哪一天
                    if (DateUtils.daysBetween(latestOverDate, period.getEndDate()) > 0) {
                        throw new ValidateBusinessException(Constant.SUB_SYSTEM, ErpExceptionType.VBE23006,
                            "检测到编号为：" + subSubUnitNumId + "的门店尚未日结完成！");
                    }
                    List<BusinessAmountBySubUnitAndPtyNum1> businessAmountByPtyNum1List = this.countSalesDirectWayAndDistributionGroupByPtyNum1(kx, subSubUnitNumId,
                        period.getBeginDate(), period.getEndDate());
                    businessAmountBySubUnitAndPtyNum1List.addAll(businessAmountByPtyNum1List);
                }
                // 加上自己的数据
                List<BusinessAmountBySubUnitAndPtyNum1> businessAmountByPtyNum1List = this.countSalesByPtyNum1(kx, subUnitNumId, period.getBeginDate(),
                    period.getEndDate());
                businessAmountBySubUnitAndPtyNum1List.addAll(businessAmountByPtyNum1List);
            } else {
                // 不是总部，只查自己的直送的数据
                List<BusinessAmountBySubUnitAndPtyNum1> businessAmountByPtyNum1List = this.countSalesDirectSendByPtyNum1(kx, subUnitNumId, period.getBeginDate(),
                    period.getEndDate());
                businessAmountBySubUnitAndPtyNum1List.addAll(businessAmountByPtyNum1List);
            }
        }
```

查自己的验收单

```
else if (kxType == KxTypeEnum.RECEIPT_COST) {// 查自己的验收单
List<BusinessAmountBySubUnitAndPtyNum1> businessAmountByPtyNum1List = this.countReceiptCostByPtyNum1(kx, subUnitNumId, period.getBeginDate(), period.getEndDate());
businessAmountBySubUnitAndPtyNum1List.addAll(businessAmountByPtyNum1List);
}
```

- 查询扣项的计算规则

```
VenderKxRule kxRule = fiVenderKxDtlDao.findKxRule(tenantNumId, dataSign,
                    kx.getSupplyUnitNumId(), kx.getReservedId(),
                    kx.getKxNumId(), kx.getKxType(), kx.getKxDirection(), kx.getKxKkType(),
                    kx.getKxStyle(), businessAmountBySubUnitAndPtyNum1.getAmount());
```

- 调用扣项计算方法得到扣项金额

```
Double kxMoney = this.calcKxByRule(kx, kxRule, businessAmountBySubUnitAndPtyNum1.getAmount());
```

```
private Double calcKxByRule(VenderKx kx, VenderKxRule rule, Double inMoney) {
        if (rule == null) {
            throw new ValidateBusinessException(Constant.SUB_SYSTEM, ErpExceptionType.VBE23006,
                "业务发生额：" + inMoney + "，未匹配到任何分段规则！");
        }
        //最小扣项金额，因为有负数存在，所以这边不适用
//      Double outMoney = Math.max(
//          MathUtil.add(MathUtil.mutiply(MathUtil.substract(inMoney, rule.getKxSegment()),
//              MathUtil.divide(rule.getKxPercent(), 100, 4)), rule.getKxBasemoney()),
//          rule.getKxMinMoney());
        Double outMoney = MathUtil.add(MathUtil.mutiply(MathUtil.substract(inMoney, rule.getKxSegment()),
                        MathUtil.divide(rule.getKxPercent(), 100, 4)), rule.getKxBasemoney());
        if (rule.getPbasemoney() > 0) {
            outMoney = MathUtil.substract(rule.getPbasemoney(), outMoney);
        }
        return outMoney;
    }
```

- 生成扣项结果

```
VenderKxResult kxResult = new VenderKxResult();
                kxResult.setSoFromType(102L);
                kxResult.setSubUnitNumId(subUnitNumId);
                kxResult.setBalanceSubUnitNumId(businessAmountBySubUnitAndPtyNum1.getSubUnitNumId());
                kxResult.setPtyNum1(businessAmountBySubUnitAndPtyNum1.getPtyNum1());
                kxResult.setCalculateDate(sellDate);
                kxResult.setBeginDate(period.getBeginDate());
                kxResult.setEndDate(period.getEndDate());
                kxResult.setBusinessAmount(businessAmountBySubUnitAndPtyNum1.getAmount());
                kxResult.setKxAmount(kxMoney);
                kxResult.setSuccessSign(1L);// 计算成功
                kxResult.setUserNumId(userNumId);
                kxResult.setCutSaveSign(1L);// 初始值默认保存扣项记录
                kxResult.setRemark("");
                kxResult.setKxRuleReservedId(kx.getReservedId());
                kxResult.setKxRuleDtlSeries(kxRule.getRuleDtlSeries());
```

- 汇总之前的结果进行比较，判断是否需要进行补差

```
Double originalAmount = fiVenderKxLogDao.countKxAmount(tenantNumId,
                    dataSign, subUnitNumId, kx.getSupplyUnitNumId(), kx.getKxNumId(),
                    kx.getKxType(), kx.getKxDirection(), kx.getKxKkType(),
                    kx.getKxStyle(), businessAmountBySubUnitAndPtyNum1.getPtyNum1(),
                    period.getBeginDate(), period.getEndDate());
if (originalAmount != null && originalAmount != 0) {
    Double diffAmount = MathUtil.substract(kxMoney, originalAmount);
    kxResult.setKxAmount(diffAmount);
    if (diffAmount != 0) {
        kxResult.setRemark("根据最新扣项政策计算得出的扣项金额为：【" + kxMoney
        + "】，与之前计算出的汇总金额：【" + originalAmount + "】比较后得到的差异金额为：【"+ diffAmount + "】，该记录是扣项补差！");
        this.saveKxResult(kx, kxResult);
    }
} else {
    this.saveKxResult(kx, kxResult);
}
```

- 发生日志，保存扣项计算日志

```
 catch (Exception ex) {
                log.error(ex.getMessage(), ex);
                VenderKxResult kxResult = new VenderKxResult();
                kxResult.setSoFromType(102L);
                kxResult.setSubUnitNumId(businessAmountBySubUnitAndPtyNum1.getSubUnitNumId());
                kxResult.setPtyNum1(businessAmountBySubUnitAndPtyNum1.getPtyNum1());
                kxResult.setCalculateDate(sellDate);
                kxResult.setBeginDate(period.getBeginDate());
                kxResult.setEndDate(period.getEndDate());
                kxResult.setBusinessAmount(businessAmountBySubUnitAndPtyNum1.getAmount());
                kxResult.setKxAmount(0D);
                kxResult.setSuccessSign(0L);
                kxResult.setCutSaveSign(0L);
                kxResult.setRemark(ex.getMessage());
                kxResult.setUserNumId(userNumId);
                this.saveKxCalcLog(kx, kxResult);
            }
```

```
private void saveKxCalcLog(VenderKx kx, VenderKxResult kxResult) {
        FI_VENDER_KX_LOG kxLog = new FI_VENDER_KX_LOG();
        kxLog.setTENANT_NUM_ID(kx.getTenantNumId());
        kxLog.setDATA_SIGN(kx.getDataSign());
        kxLog.setSUB_UNIT_NUM_ID(kxResult.getSubUnitNumId());
        kxLog.setSUPPLY_UNIT_NUM_ID(kx.getSupplyUnitNumId());
        kxLog.setKX_NUM_ID(kx.getKxNumId());
        kxLog.setRESERVED_ID(kx.getReservedId());// 政策单号
        kxLog.setKX_TYPE(kx.getKxType());
        kxLog.setKX_DIRECTION(kx.getKxDirection());
        kxLog.setKX_KK_TYPE(kx.getKxKkType());
        kxLog.setKX_STYLE(kx.getKxStyle());
        kxLog.setRANGE1(KxRangeEnum.PTY_NUM_1.getValue());
        kxLog.setRANGE_ID(kxResult.getPtyNum1());
        kxLog.setFLAGTYPE(0L);// 废弃
        kxLog.setCALCULATE_DATE(kxResult.getCalculateDate());
        kxLog.setBEGIN_DATE(kxResult.getBeginDate());
        kxLog.setEND_DATE(kxResult.getEndDate());
        kxLog.setBUSINESS_AMOUNT(kxResult.getBusinessAmount());
        kxLog.setKX_AMOUNT(kxResult.getKxAmount());
        kxLog.setSUCCESS_SIGN(kxResult.getSuccessSign());
        kxLog.setCUT_SAVE_SIGN(kxResult.getCutSaveSign());
        kxLog.setREMARK(kxResult.getRemark());
        kxLog.setCREATE_USER_ID(kxResult.getUserNumId());
        kxLog.setLAST_UPDATE_USER_ID(kxResult.getUserNumId());
        kxLog.setCUT_RESERVED_NO(kxResult.getCutReservedNo());
        fiVenderKxLogDao.insertEntity(kxLog);
    }
```

供应商固定扣项维护单

`com.ykcloud.soa.erp.fi.service.model.VenderKx`

`fi_vender_kx_hdr`

```
CREATE TABLE `fi_vender_kx_hdr` (
  `series` bigint(20) NOT NULL DEFAULT '0' COMMENT '行号',
  `tenant_num_id` bigint(20) DEFAULT '0' COMMENT '租户ID',
  `data_sign` tinyint(4) DEFAULT '0' COMMENT '测试标识',
  `create_dtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `last_updtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '最后更新时间',
  `create_user_id` bigint(20) DEFAULT '0' COMMENT '用户',
  `last_update_user_id` bigint(20) DEFAULT '0' COMMENT '更新用户',
  `cancelsign` char(1) DEFAULT 'N' COMMENT '删除',
  `insertdata` char(1) DEFAULT 'Y' COMMENT '新增',
  `updatedata` char(1) DEFAULT 'N' COMMENT '更新',
  `audit_updtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '审核日期',
  `audit_user_id` bigint(20) DEFAULT '0' COMMENT '审核人',
  `reserved_id` bigint(20) DEFAULT '0' COMMENT '扣项单号',
  `supply_unit_num_id` bigint(20) DEFAULT '0' COMMENT '供应商编号',
  `note` varchar(255) DEFAULT ' ' COMMENT '备注',
  `isclhhzc` tinyint(4) DEFAULT '0' COMMENT '是否处理坏货折扣',
  `isshowhistory` tinyint(4) DEFAULT '0' COMMENT '是否显示历史页',
  `sub_unit_num_id` bigint(18) DEFAULT '0' COMMENT '结算门店',
  `status_num_id` int(8) DEFAULT NULL COMMENT '状态',
  `tax_rate` decimal(10,2) DEFAULT '0.00' COMMENT '税率',
  `kx_no` bigint(20) DEFAULT NULL COMMENT '原扣项单号',
  PRIMARY KEY (`series`),
  UNIQUE KEY `ux_fi_vender_kx_hdr` (`reserved_id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

`fi_vender_kx_dtl`

```
CREATE TABLE `fi_vender_kx_dtl` (
  `series` bigint(20) NOT NULL DEFAULT '0' COMMENT '行号',
```

```sql
  `tenant_num_id` bigint(20) DEFAULT '0' COMMENT '租户ID',
  `data_sign` tinyint(4) DEFAULT '0' COMMENT '测试标识',
  `sub_unit_num_id` bigint(18) DEFAULT '0' COMMENT '结算门店',
  `create_dtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `last_updtme` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '最后更新时间',
  `create_user_id` bigint(20) DEFAULT '0' COMMENT '用户',
  `last_update_user_id` bigint(20) DEFAULT '0' COMMENT '更新用户',
  `cancelsign` char(1) DEFAULT 'N' COMMENT '删除',
  `reserved_id` bigint(20) DEFAULT '0' COMMENT '扣项单号',
  `kx_num_id` bigint(20) DEFAULT '0' COMMENT '扣项代码',
  `kx_type` tinyint(4) DEFAULT '0' COMMENT '扣项类型',
  `kx_direction` tinyint(4) DEFAULT '0' COMMENT '扣项方向',
  `kx_segment` decimal(22,4) DEFAULT '0.0000' COMMENT '分段开始值',
  `kx_segment_end` decimal(22,4) DEFAULT '0.0000' COMMENT '分段截止值',
  `kx_min_money` decimal(22,4) DEFAULT '0.0000' COMMENT '最小扣项金额',
  `kx_percent` decimal(5,2) DEFAULT '0.00' COMMENT '扣项百分比',
  `kx_basemoney` decimal(22,4) DEFAULT '0.0000' COMMENT '比例调整金额',
  `kx_kk_type` tinyint(4) DEFAULT '0' COMMENT '扣项交款标志',
  `kx_style` tinyint(4) DEFAULT '0' COMMENT '计算方法',
  `range1` tinyint(4) DEFAULT '0' COMMENT '扣项范围',
  `range_id` bigint(22) DEFAULT '0' COMMENT '类别编码或商品编码',
  `begin_date` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '生效日期',
  `end_date` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '结束日期',
  `note` varchar(255) DEFAULT ' ' COMMENT '备注',
  `pbasemoney` decimal(22,4) DEFAULT '0.0000' COMMENT 'pbasemoney',
  `flagtype` tinyint(4) DEFAULT '0' COMMENT '0 只落一档 1 只落一档减开始金额 2 多档减开始金额累计',
  PRIMARY KEY (`series`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```