银行核心系统入门简介

今天整理老电脑的硬盘，发现以前学习的一些材料放在今天仍然是有很大的参考价值的。比如这篇《银行核心系统入门简介》，虽然是10年前的文章，但里面科目分类，会计原理，红冲，总分核对，到今天基本都没有太大变化。里面提到的招行一卡通，在当时是非常先进的理论，放到今天，大家在支付宝、微信支付里面看到各种余额账户，余额宝账户，基金账户等，也是类似的思想。

那变化的是什么呢？互联网在线支付的极高并发、高可用、7x24小时服务等，这些是通过技术手段来实现的。还有跨境场景下的资金方案，比如标价USD，欧洲用户支付了EUR，商家要在国内收CNY，如何既能满足合规要求，又能挣到更多外汇，还能更快地让商家拿到钱？这都是在跨境贸易高歌猛进的同时需要解决的问题。当我们唱响“全球收，全球付，全球汇”的口号时，就不能只看单币种怎么玩的啦。

下面是原文，未做删减，因作者不详，属名“佚名”。有些内容已经过时，请自行甄别。

---

1 前言

本文的目标读者是准备从事银行核心系统开发、维护的从业人员。请注意，是“准备”，换句话说，可以理解为一份对科技人员，尤其是对新入门的科技人员业务知识方面的培训手册，旨在让诸位从业务方面迅速上手（从技术角度上手的手册我已经贴过一份了，所以如果是用400的同行，可以结合本手册双剑合璧，效力倍增）。这里的着重点将会主要在于简单的银行会计原理，以及银行整体的业务流程，还有相应的模块实现手法和注意事项，对金融的会计知识方面应该可能会比较粗浅，这一点与金融系统常见的业务培训手册有所不同，注意体会。

基于此，本文将会假设读者具备一定的计算机技术，具备少量银行方面的业务知识，所以如果有从事非IT部门的读者（比如财务信贷的同事们），就请不要太计较里面的表述。当然如果有错误，还是非常欢迎指出的。

对于已具备了若干开发、维护知识，或者是即将采用国外系统来建设的同行们而言，本文的内容可能就过于浅显了，看得不爽不要怪我没有事先提醒。

考虑到某方面的问题，这里的系统简介将尽可能的脱离某个具体的系统，仅就银行业务核心系统的共性，进行介绍以及探讨。

最后再说一下，没有什么手册、心得是万能的，个人的LEVEL UP始终是要靠自己的领悟，这里只是希望能让诸位新人不用象很多人当年一样，独自摸索与徘徊。

2 科目常识

基本法则之一：资产 = 负债 + 所有者权益。

100万＝60万+40万

比如说，我们手头上有40万，买了一个100万的房子，找银行贷款了60万

简单的把所有者权益就理解成为是真正属于自己的钱。再引申一下，早些年乃至现在，香港人所谓的“负资产”的说法是非常错误的，因为“负资产”实际上是指房子的市值比向银行贷的钱还要小，也就是负债大于资产，所以严格的来说，应该称之为“负所有者权益”才对。资产，从理论上来说，是不可能为负的，最多也就是零。一个号称是金融中心的地方，实在是不应该出现这种失误，不过算了，不要和他们计较。

就银行业务而言，会使用会计科目号来对账务进行标识，会计科目号最长为5位，国家标准，通常分为下面六种，这里只做简单介绍，详细科目可结合著名的的**“业务状况表”**来进行理解。

2.1 资产

资产类的科目，用“1”作为首位科目号，如“1011”，表示现金。

所谓资产，也就是说“理论上属于银行的钱”，比如说现金，贷款等。比如说某家分行，有100万现金，然后把这100万都贷出去了，那么资产仍是100万，只不过归属（科目）由现金变成了贷款。至于这笔贷款能不能收回，这个不归我们管，就算不能回收，只要没被核销（核销，术语之一，可以理解为银行不要这笔贷款了），那么就仍然属于资产，所以我们称之为“理论上属于银行的钱”。

资产类科目都是借方科目，也就是借记时余额增加，贷记时余额减少。

2.2 负债

负债类的科目，用“2”作为首位科目号，如“2011”，表示对公存款。

本来不属于银行的钱，就称之为“负债”。比如说我们存在银行的钱，虽然银行可以使用这笔钱，比如说把它贷款贷出去啊，比如说打新股啊，买QDII啊，但是这笔钱只要我们去取，原则上银行就应该给我们，也即是大家常常在营业大厅里看到的“存款自愿，取款自由”之类的意思。这类钱，可以简单的理解为“本来不属于银行的钱”，也就银行欠我们的钱。

负债，很有趣的东西喔，银行是负债经营的，比如说一家银行贷款有100亿，其实它本身是没有那么多钱的，这些钱都是来自于我们存在它那的钱。如果大家一起都去银行的钱取出来，那它就经营不下去了，这种恶劣的行为，称之为“挤提”，是很不友善的，是要负责任的，我们不要去做。

负债类科目都是贷方科目，也就是借记时余额减少，贷记时余额增加。

2.3 所有者权益

所有者权益类的科目，用“3”作为首位科目号，如“3121”，表示利润分配。

上面说过了，所有者权益，也就是真正属于银行的钱，即是所谓的“核心资本”。原则上，它包括了一家银行注册时的资金，历年来的盈利（假设有盈利的话，当然还要扣除各类成本开销），如果是股份制银行的话，还包括股本金之类的吧。

这类科目相对数量较小，金额较大。

科目属性忘了。

2.4 资产负债共同类（往来类）

资产负债共同类，通常表示往来账户，用“4”作为首位科目号，如“46411”，表示通存通兑。

这类科目，通常是指一些往来类账户，所谓往来类账户，嗯，就是金融往来的账户喽。

这个科目有点麻烦，可能要结合具体业务来解释一下：

比如说我们在招行有个账户，然后跑到工行的ATM上去取钱，那么取款成功之后，我们的招行上的账户的钱就少了，工行ATM里面的现金也少了。这笔钱是工行替招行先支付的，要找招行要的。所以工行一定会有一个科目，用来标记它有多少钱要找招行要；而招行也要有一个科目，也是要用来标记它有多少钱要给工行。（怎么要，那在后面清算一节里面会提到。至于跨行ATM的取款原理，就不用再细说了吧。）这个用来标记应付，应收的科目，就是往来类科目，对于工行方而言，当时使用到的就是一个类似于资产类的科目（有点类似于应收账款的意思，或者也可以理解成一种短期的贷款，总之就是工行先付出的资金）；招行当时使用的就是类似于负债类的科目。

上面提到的，因为是银行与银行之间的业务往来，所以用来标识资产与负债的科目会有分别，如果是行内之间的往来，那么不会搞得那么复杂（或者也可以说搞得更复杂），就会用一个科目来搞定，这个科目根据具体需要，临时用的，有时表示资产，有时表示负债（其实也就是科目上的余额有时是借方，有时是贷方。因为这个科目既不是资产，也不是负债，只是临时用来表示营业往来的，通常每天会清零，也就是所谓的清算。

一般而言，城市级别的商业银行因为是一级法人，所以清算之后，行内往来账户上余额为不为零都没什么关系，反正都是自己家的钱；而信用社比较麻烦一点，因为通常一个联社都是由多个信用社组成，每个信用社都是一个法人，所以联社内部的往来类账户原则上每天应该都清零，否则账

务上就不好看了。（注意，这里指的只是行内的往来账，如果是银行与银行间的，那每天一定是要清零的，否则就是属于错误的情况了）

这类科目在我们做过的项目里，基本上都简化了，只有一个轧差类型的。也就是把当天的借方发生额和贷方发生额一减，哪个大就谁记在哪边。

2.5 损益类

损益类的科目，用“5”作为首位科目号，如“5011”，表示利息收入。

损益类科目，理解起来应该不难，就是指银行在一年的业务里面的收支科目。比如的存款利息，对于银行来说是一笔支出；贷款利息，对于银行来说，是一笔收入。这两个科目就都属于损益类科目。

一般来说：

收入类科目属贷方科目，借记时减少，贷记时增加；

支付类科目属借方科目，贷记时减少，借记时增加。

在理解上，可能与资产、负债类的科目有些相反：

资产是指属于银行自己的钱，是借方科目；对应于这里，收到的钱是银行自己的，却又是贷方科目。

这里，按会计原理来理解可能会更简单一点，下面一章会讲到。

2.6 或有资产负债类

或有资产负债类的科目，用“6”作为首位科目号，如“6011”，表示承兑汇票。

闻歌知雅意，顾名思义，“或有”，那自然就是“或者有”，也就是可能没有了，所以如果没见过也不奇怪。

这类科目见得少，一般可以忽视它的存在。

2.7 其它

这里再罗嗦一下，在科目下面呢，一般为了便于分类统计，所有的银行都会再设子目（一个子目一般又会对应多个小子目，或者说是说是多个账户），这个子目，有的地方叫“业务代号”，有

的地方叫“结算码”，总之都是一个意思。

要注意一下，科目号是国标，子目通常是自己内定的，对应于信用联社，就有可能是省里统一定的。也就是说科目这个东西走遍全国大致上都是一样，子目这个东西可能出省，出了城市，或者说一个市里不同的银行，可能都不一样。

3 简单会计原理

3.1 内部账户

所谓内部账户，是与客户账户相对应的。也就是说这些账户不是用来登记、反应客户的账户信息，而是反应行内的账务情况，比如说损益类科目的账户，就都是内部账户。

客户的账户，一般是客户来银行开户的时候，才建立的用来登记账务的账户；

内部账户，一般是分行成立之初，统一生成的。（一般都一个专门的程序，由操作人员来调用的吧）

其实对于内部账，在会计原则上，登记个科目发生可以。至于增加子目，乃至内部账户的概念，主要是为了后续的分类统计以及相应的分析。

说到这个账户，就顺便想起了表内表外的问题。表内账，都是正正式式，真金白银的钱；比如我们的存款什么之类的。而表外账，通常是一些统计之类的东西，比如说现在分行里有多少本存折啦，还有已经核销的贷款之类的。

表内账的单位，都是“元”；

表外账的单位，就百花齐放了，有的是“元”（比如说已核销贷款），有的是“本”或者是“张”，比如说存折或者说什么有价单证。而最后，表外账在汇总统计的时候，不管是什么单位，就是统统一加了事，银行会计上就是这样处理。

所以说，一般报表里面，大家会对表内账比较关注，对表外账的要求不是太严格。

3.2 复式记账法

也称为借贷记账法。

“　　有借必有贷，借贷必相等”，这两句经典的话，是针对表内账的。对于表外账，用的其实是单式记账法，有的叫“收”、“付”，也的也还是用“借”，“贷”，要结合具体的业务来理解，这里就不展开了。如果没有特别说明，下面的描述都是针对表内账的。

对于银行业务来说，最简单的是一借一贷，此外，还有一借多贷，一贷多借。多借多贷在银行业务里中不允许的，因为这样无法精确的体现账务的起始与流向。不过在企业会计中，多借多贷又是允许的，所以说凡事无绝对。

有些时候，基于某些特殊的的原因（常见的主要是频繁的锁表问题），可能会临时采用单边记账，但是最后一定会汇总补齐，否则就会出现“借贷不平”这样的严重问题。

3.3 冲账

做错了账，要改正它，就可以理解为冲账。

冲账有两种，一种是蓝字冲账，一种是红字冲账。

所谓的蓝字冲账，是指与原账务方向相反，金额为正的一种记账方式。

而红字冲账，就是指与原账务方向相同，金额为负的一种记账方式。

蓝字冲账，本质上是做一笔新的业务，仅仅只是实现了最终的余额正确，发生额会虚增，所以一般的明显有错的账务，会要求使用红字冲正。

红字冲账因为是负数发生，所以在统计的时候，发生额将会与原来的交易抵销，这样的话发生额就很严谨了。

实际上，对于一个系统而言，通常一笔业务的发生，并不仅仅只包括账务的登记，还会更改许多表中的数据。比如说一笔简单的取款交易，除了登记账务之外，客户的账户上的余额还会减少。那么在冲账的时候，还需要将客户上的钱给它加回去。

4 业务流程描述

对于一个没有在柜面实习过的人，描述一下银行的业务流程，可能是有助于理解系统架构的。

银行的业务，大致上可以分为财务类的业务，以及非财务类的业务。

财务类的业务，又可分为自动业务，以及非自动业务。

非自动业务，就是那些必须在柜台办理的业务，比如说一些转账业务，或者金额较大的存取款业务之类的。这类业务，因为是由柜员发起的，所以会有一些单据打印留底，以做传票使用。

而自动类业务，就是由系统自动处理的，比如说我们在A分行有个账户，然后非要跑到B分行去取钱，那么B分行那部分的账务，对于B分行而言就是非自动业务；而A分行那部分的账务，对于A分行而言就是自动业务。

自动业务因为是自动发生，所以需要业务人员打印报表的时候，才能知道发生了什么业务。

柜员日间做各种各样的业务，然后到了下午关门以后，打印一份“科目日结单”，然后用柜员手头留存的传票，按科目逐一汇总累计，与打印出的科目日结单上的金额进行比对。有错一定要一查到底。所以原则上，这时打印的科目日结单，应该不包括自动业务，否则就会对应不上。

业务系统在批处理的时候，还会进行一些自动的账务处理，然后最后系统还应该会再打印一份完整的科目日结单，以及日计表（可以理解为业务状况表的简洁版）。至于那些自动业务，系统在批处理的时候，或者是柜员主动查也行，总之就是会有一份“他代本”的传票（对应于上面提到的业务，A分行的自动业务就应该属于A分行的“他代本”传票。而B分行的传票因为是非自动业务，所以在交易当时就会有相应传票产生并打印了）

到了第二天，分行开门后开始营业前，业务人员需要下载打印各类报表，不过主要的就是前面说的那两份，然后再看看，如果借贷发生、余额都相等，所有的非自动业务都有传票，而且和整个科目日结单都可以对应上，那么就表示昨天的账务完整无误，然后大家就可以欢天喜地的开始新一天的业务了。

5 常见规范及检测

5.1 传票以及日志

从最基本的说起，通常来说，所有的账务程序都需要打印传票,传票格式通常都是统一的，找份以前看看就可以了。

对应于转账业务，需要打印转账借、贷方双方的传票。

而对于现金业务，则只打印一张传票就可以了，借贷方向采用非现金科目的方向。（我个人认为，可能是因为标识了现金传票，所以对方科目就自然是现金，于是就不需要再打印了，猜的）

所以我们在开发程序的时候，打印传票这一步，一般不会特别强调，都是默认要做的。如果不太清楚的时候，一定要主动向需求设计人员询向，千万不要嫌麻烦，抱有侥幸心理。这种东西如果测试的时候漏掉了，是一定会有人要求补上的。（我在N多项目里都见过漏写传票，然后在程序上线前夕被人要求赶紧加班补制的，所以千万不要嫌麻烦）

在日终批处理的时候，可能有些数量庞大的业务，比如说代收付，结息什么之类的，动不动就是几十万笔，一张张生成、打印太不经济，通常会考虑采用打印一张汇总传票，然后加上一份明细清单的方式。还有的时候，如果上百万的话，可能明细清单都省掉，想办法导成电子数据都是有可能的。

上面说的是账务相关的业务。而非账务类的业务，如果涉及到修改类的业务的话，比如说修改密码，修改客户名之类的，通常需要登记日志（LOG），用来记录，以便查询。

有的时候，为了统计业务量，或者是为了分析排障，还有可能要求对每一笔发送到主机的业务数据都登记下来，这时候最好采用一种统一的方式来进行登记，以及数据的定期清除，因为这类数据量应该比较大。

5.2 常见检测内容

发生一笔业务的时候，是一定需要进行若干检查的。比如最起码，我们去取钱的时候，就一定会检查密码。这里对一些经常见到的，较为普遍的检查简单介绍如下，套用一句合同上流行的话，叫做 -- 包括但不仅限于以下条款：

1、账号/卡号是否存在，是否可以正常使用

2、账号与客户所提供的凭证（通常这是指存折客户，对于卡用户而言，账号就是卡号，或者是可以根据卡号查询出相应的账号）是否匹配。

3、密码、证件号码（如果需要检查的话）是否与主机数据一致（印鉴什么的需要业务人员肉眼核对。现在又出了一种加密机，如果采用了这种先进技术，那当然还需要检查这种加密后的信息是否一致了）

4、在转账的时候，一定要检查转出转入方的户名与账号/卡号中的户名是否一致。（对私客户还好办一点，如果是对公客户的话，名字又长，括号什么的再一加，经常会出现问题，总之是一定要检查）

5、如果是取款类业务（比如转账业务的转出方也算），一定要检查账户的可用余额是否足够。

6 系统架构及部分模块常见设计方案

6.1 常见总体架构

这里如果用图可能效果会更好，不过我不会用VISIO，所以就算了。

一般硬件架构，都是一个主机，一个前置机（大前置），前置机就对外了，比如业务人员用来作业务的终端啦，ATM，网银，电话银行什么之类的可能就都对应这个大前置了。大前置，或者

是中间业务平台，也是一个很值得探讨的问题，可以做得很大，比如建行的大前置，又比如X天的中间业务平台其实也不错，这里不做深究。

就软件架构而言，核心系统一般可以分为业务模块，账务模块，和总账模块。

总账模块通常记录了一些账务的汇总信息，比如说科目总账的日、月、年的发生、余额。银行中大部分的报表都需要通过取总账模块中的数据来生成。总账模块的数据一般是取自账务模块中，当天的账务数据。（当然，也有很多报表，需要整合业务模块与总账模块两部分的数据一起来出）

账务模块，就是用来登记账务的，这部分一般会做得比较通用化，方便各个业务模块来调用。

业务模块，当然就是实现各个业务的子模块了，通常模块之间相对独立又互有关联，如果是账务类业务，当然就要调用账务模块中的程序。如果是非账务类的业务，那可能业务模块内部处理一下就可以了吧。

一般业务模块的数据会对实时性要求较高，而总账模块没有什么实时性的要求，不过总账模块重在统计分析，所以数据量一般会比较大。

6.2 计息

有的系统可能没有把计息单独列为一个模块，而是直接嵌套在各个业务模块之间了，不过设计成一个模块，个人认为可能会显得比较专业一点，至于到底好不好用那就见仁见智了。

刚接触银行业务的时候，曾经很执着，很傻很天真的想过活期账户到底是怎样计息的，因为定期账户的计息方式相对简单，余额乘天数就对了，但是活期账户的余额是常常在发生变动的。

银行会计上，通常都会通过“积数”这个东西来计息。何谓积数？就是余额＊天数，所以积数的单位应该是“元天”

比如说 利息 = （账户余额＊天数＊利率）/ 360，在这个公式里，账户余额＊天数就等于积数，于是这条公式也可以写为利息 = （积数＊利息） / 360。

定期账户因为账户余额通常不发生变化，所以一般不会涉及到积数。

活期账户采用动户累计积数的方式来计息。也就是说账户余额没有发生变动，就什么事都不干；当账户余额需要发生了变动时（比如说取款），那么业务模块里就将上次账户变动日，到当前日期的天数计算一算，然后用变动之前的账户余额乘以这个天数，然后把这个积数累加到之前的积数上。最后计息的时候，就使用这个积数乘以利率再除360。

在设计的时候，就需要把每次账户变动的日期都登记下来，还需要有地方记录账户的当前积数。

对公计息，或者是一些需要计息内部账，有可能是每天计积数，也就是每天把账户余额累加到积数中。之所以这样设计，是因为对公以及内部账户的数量远小于对私账户，每天把每个账户都过一遍，花不了太多时间；而要是每天把储蓄账户都过一遍，就有点类似于结息了。（对私账户多的银行，有可能达到上千万户，尤其是些代理了社保，医保的银行，不可小看）不过现在有些很好很强大的国外系统，对于利息的处理，是每日计提，当然，这样设计也应该会有它的独到之处。

刚才这里提到的了需要计息的内部账，那么一般而言，什么样的内部账需要计息呢，我想，应该是不同法人之间上存下放的款项需要计息。对应于一般的商业银行以及统一了法人的信用联社，因为全市是一级法人，可能就没有需要计息的内部账了。而对于没有统一法人的联社，因为每个信用社都是一个独立的法人，那么信用社存放在联社的用来做往来清算用的资金，就是需要计算利息的。还有的银行，对于贷款的处理，也会有资金池的概念，这时总行下拨分行的用于贷款钱，也是要计息的。

这里可以看到，对于计息模块而言，积数是一个很好用的东西。积数除了计息，还有很多其它的用途。比如说招行的金卡，说的是“日均存款5万元以上不收取账户管理费”，那么，这个日均存款5万是如何判断呢，我很久以前曾经问过一个大堂里的MM（跟我同姓喔，惜乎已经有BF了），她说是根据积数来判断的，也就是每个月需要增加150万的积数，这样听起来就很合理了吧。

对于某些业务来说，可能需要登记利息的明细。比如说贷款的复利的计算，就是根据利息来的。无论是正常贷款，还是逾期贷款，都会生成利息。生成的利息如果未及时归还，则会再根据这笔利息生成相应的复利。复利的复利，喔，太可怕了，也还是视为复利吧。总之，我的意思就是说，储蓄、对公账户这样的结息，在计息模块中可以不用登记利息的明细，因为最后结息的时候根据积数一次搞定；而对于贷款（或者是其它有需要的模块），可能需要在每一笔利息产生之后，都把它登记下来，已保留行使进一步措施的权利。

除了贷款之外，还有一些定期账户，也最好采用明细的方式进行处理，越细越好，比如什么零存整取，教育储蓄之类的，要是没有详细的每期存款登记，漏存登记等等，是很容易就被它玩死的。

通知存款无非就是通知取款，通知期限内的积数登记，然后取款又或者取消通知。可能最主要的，就在于通知期限内的积数计算。总之提取一个计息模块，为这类业务特别定制一些明细文件是很好的一个选择。

提到计息，也就顺便说一下利息税。国家在这十年来，调整了两次利息税税率，一次是涨成两分，一次是降成五厘，就那么一点钱，调来调去累不累，要收就收，不收拉倒，还搞什么分段计税，烦死个人。在这里，不知道有没有人是负责搞利息税这部分程序的，也不知道去年改这部分程序的时候，有没有很不爽过。其实要是早考虑到这种情况，倒是可以一开始就通过设置利息税参数表，然后修改计息程序，读取利息税参数表，最后根据不同阶段的参数，分段计息算税。这个方法倒是可行的，也实现过，对于整存整取的定期来说，算得上是一劳永逸，不过对于活期而言，每次调整利息税税率的时候可能就要搞一次类似于结息的东西了，好象没有一劳永逸的方法。

在国外的先进系统中，还有一种精采的倒起息可以让人一筹莫展。这种玩法的意思，就是说当客户来柜台前做个什么交易的时候，允许账户的起息日期在业务发生日之前。比如说有人7月14号来到柜台前还一笔贷款的款，然后说我这笔钱明明7月7号就到账上了啊，为什么银行不给我扣，非得让我贷款逾期之类的话。然后核查，如果属实，那就倒起息一把，现在虽然是7月14号，但还是当它是7月7号还的。（好象是这样，也可能是我说错了，大家对这段解释千万不要太放在心上）总之，如果有倒起息的需求，那必须在最开始设计的时候就与其它计息，以及业务流程整合在一起来考虑，如果中途加入这个需求，那改起设计来会比较费劲，改起代码来更是难上加难。

最后，我们再来说说计提，这个也和利息有关。计提常用于利息支出，比如说利息支出是5211，5字头，即是一个用于营业收支的损益类的科目。计提的会计分录中，对应的科目是应付利息2611， 2字头，是一个负债类的科目。所以说，计提的含义就在于，虽然当前客户利息并未产生（是结息的时候才产生），但是这笔利息（尤其是整存整取的定期利息）迟早是会产生的，所以这里预先计算，或者说估算出营业支出，计到负债的科目上（负债嘛，本来不属于银行的钱，迟早是要被取走的钱），然后到这类账户结息的时候，就直接从应付利息中支出，计到客户账户上，而不走利息支出这个科目了。看懂了吧，这里其实也就包含了管理会计中的概念，实际上是产生一个提前测算成本的动作。诸位搞IT的朋友们，你们看过《会计学原理》吗?

6.3 储蓄/对公

这部分模块一般没太多可讲的，通常的设计，都是搞个主文件，保存针对每个账户的信息（比如说账号，账户余额，当前积数什么之类的，总之就是与账户有关的信息），然后再搞个账户明细，用来记录每个账户发生过的业务。听闻有的系统设计，不知道是不是考虑到锁表的问题，计划取消主文件，直接上明细，愕然之余只能感叹自己见识浅薄，因为我总觉得明细要考虑冲账的问题，在读取上不如主文件一下搞定那么畅快。而且主文件可以有锁表保护，可以更好的保障数据的正确性。

所以私底下，我还是很推崇这种“主+明细”的设计方式。以前曾经很无奈地见过有人在新增业务模块时，把主文件和明细混在一起来搞，于是整个业务流向怅然若失，需求有变动时改动几乎无从下手，若非我多年功力，是断断不可能在加两天班后就理顺通过测试的。

说起储蓄呢，又忍不住再提一下招行，不可否认，它的一卡通做得真的挺好，本外币，定活期，一张卡全部搞定。我以前就经常把活期转成三个月定期。根据我本人看法，三个月定期从利率差与时间存放差上来说，性价比是最高的，也就是说一年期利率虽然高，但很难保障这点钱在一年内不用。所以推荐大家把5K以上的存款转成三个月定期，一般忍忍也就可以拿到利息了，当然了货币基金也是一个不错的选择。还有一次自做聪明搞了个一年期的零存整取，性价比不高，而且还得到柜台去办取款手续，把自己麻烦死了，不推荐使用。

扯远了，其实本来是想说，活期、定期、外币账户，这些都是一个又一个的账户，而在招行的设计之中，这些账户，都会与我们的那一张小小的卡片关联起来。换句话说，人家的卡号，应该只含具体的卡的信息，比如说卡的有效期，密码，磁道信息什么之类的，不直接对应某个具体账户的；而各个具体账户则应该会有一个与卡号的对应关系。然后到寄对账单的时候啊，打电话介绍买保险等等附加服务的时候，就还是根据卡号来提供服务。不过还是要根据账户的资金流动来分析消费习惯，以及贡献度的高低等等。

至于怎么实现，就根据各位自己的核心系统慢慢体会，不过这么多年了，也可能大部分银行都实现了这种功能或者是类似的一卡通，那就当我这段没有讲过吧，总之我觉得这种理念很好很强大，让我用得觉得很方便。

至于对公，好象就更加没什么可说的了。

6.4 客户信息

客户信息，卡号，账户号，这三者是层层细化的关系。所以说，整合好三者的关系也是一个不容易的事情。

在我见过的几套系统之中，最常见的问题，就是同一个客户对应多个客户信息。这通常又是个历史遗留问题，比如在手工或单机年代，开户时对于身份证明证件要求不是很严格，一个人可能开了很多账户，还可能是用化名开的账户。在移植上线的时候，常常由于重要信息不齐，又要考虑客户层面的因素，很少能强制性补齐客户资料，通常只能在移植时自动生成一些客户信息，这样就造成了很多冗余，而且也不好再做深层的数据挖掘和客户分析。相比较而言，新开立的分行可能这种情况会好一点，而且面对的客户高端一点的，又会更好一点。

在新系统上线，做数据移植的阶段，一般客户信息的问题是最先体现出来的，通常新系统会要求得比较理想化，而实际情况千奇百怪。这里说说常见的，比如说新系统一般会要求证件号码唯

一，但是因为很多客户的证件信息缺失，所以这个号码唯一可能会有困难；再比如说有时可能会出现证件号码重复，而且还真的不是同一个人。

总之这些问题，它不是新系统的错，也不能完全说是旧系统的错，最关键的是在移植的时候如何处理利用好这部分客户信息。

再一个问题，就是客户信息的更新。个人认为最好能有一个有效的途径来更新客户信息，尤其是工作单位，电话号码，对于很多流动人员来说，经常会变换。如果每次都要来柜台更新，我想那基本上就可以认为它是形同虚设了。

可以说，随着现在以客户为中心的概念的提出以及越来越多的实现，客户信息这个模块也应该会越来越受到重视，以前设计的表结构应该会有些不够用了。目前如果没有新系统要上的同行们，恐怕是要等着改结构加字段了，保重。

6.5 贷款

很多地方都会把一般的商业贷款与按揭贷款和消费贷款（比如车贷、分期付款之类的，总之有点类似于按揭贷款的）区分开来，这样自然有它的道理。我在这里只谈我个人的设计方案。

现在的商业贷款常常采用一笔发放，一笔回收的概念（当然有时会有提前还款，但不象按揭贷款这样有个具体还款计划），然后用合同号，或是借据号做为贷款的一个类似于唯一关键字这样的东西。但是有时公司的商业行为中，一个大项目里会包含多个子项目，然后对应不同的子合同，这些合同对应的贷款之前其实都是有关联的，尤其是在算逾期什么之类的时候，有的是一逾全逾，有的又不是。所以我个人觉得，贷款最好做成多笔发放，多笔回收的形式，发放与回收不必一一关联。但最好在贷款录入时（这时不一定已放款），就录入相应的还款计划。

贷款的账号，最好与具体的业务信息剥离，类似于储蓄里面“一卡通”的概念一样，每个贷款，有它自己独立的贷款号，然后正常、逾期、两呆，以及相应的利息账号都与这个贷款号关联起来，便于以后的跟踪追查。

而对于按揭贷款来说，因为期限长（常常是二三十年），而且比较具有规律性，所以一般就不用列出还款计划的明细了。不过要注意，一般按揭贷款的首月还款是按天算息的，稍微注意一下就可以了。

最后，特别强调提出一点，见过两家行，都推出过“等本等息”这种经典的业务产品，也就是客户每月按等额法算出的金额还款，但本金的计算则按等本的方式来算。

这里要大声疾呼，这种东西从原理上来说就已经是错误的！因为同样金额，同样期限的贷款，等额法的利息是要大于等本法的利息的。等本法计算方便，理解简单；而等额法是数学家们经过精确的计算，推导出公式，最后计算出的一种还款方法。也就是每个月的还本、还息都要严格按照计算出的公式，这样才能达到等额的效果。试想想，这个月还了一定的本金之后，下个月计算出的利

息就不一样了吧，这时要求下个月还的本金与还的利息加起来还是和这个月的一样多，而且还要求每个月还的本金加上利息都是一样多。所以，除非是数学学得特别好的同学，咱们一般的程序员不要妄想自己能推导出公式来，照着公式算就行了。如果强行按等额法计算出的钱来制订还款计划，又按等本法的方式还计算每期还款本金，虽然是方便了，但是在每年利率变更，重算利息时，必然会导致利息总和由等额法的利息渐渐趋近于等本法的利息，也就是总利息额将会越来越少，于是要么在本金与利息的问题上无法自圆其说，要么可能会出现利率上调还款金额反降，甚至负利息的问题，不可不查。

6.6 清算与结算

清算与结算本来是两种业务，不过因为结算中通常又会包括清算，要分成两小节，每小节又说不了太多话，所以干脆放在一起算了，而且这一节只谈流程，不讲设计，这种业务流程理顺了自然就可以设计了。

先约定一下，商业银行的级别，一般是 分行—支行两级，有的可能还会有储蓄所这种第三级。简化起见，暂时就分两级来说吧。如果对应到信用社，那就是联社营业部—信用社营业部。分社一级省略。

先从结算说起，这里的结算业务，指的就是跨行转账，至少我是打算这么说。每家商业银行，都会在当地的人民银行有一个资金账户，可以理解为结算业务用的备付金账户。然后在自己行内，也会开立一个与之对应的“上存人行款项”的账户。理论上，人行的这个账户和我们自己行内的这个账户，表达的都是“该银行存放在人民银行的钱”的这个意思，所以金额也应该相等。那么，这两个账户在不同的银行（也即不同的系统中），如何保障它的一致性？这一般就是通过日终，营业终了时的对账来保障。所以对账是很重要的。

至于结算业务的流程，先从遥远的手工账/单机账年代说起吧。在那个时候，结算的途径、概念、术语可以说是五花八门，什么先直后横，先横后直，提出借方，提出贷方，提入借方，提入贷方，信汇，电汇等等等等，不把人转晕誓不罢休。现在好象大小额支付横空杀出，倒是简化了不少。当然也还有行间转账，同城支付，省金融平台，不过概念上渐渐趋向统一化，先不多说，先谈谈当时我理解中的流程：

首先如果要转账，我们要在柜台前填一份一式五联的单（一定要用力填哟，不然最后一张纸上看不到什么字迹的），然后这笔钱就从我们的账户上扣下来，划到银行内部的某个往来账户上了。

然后这些单据，再手工传递到上一级，上一级再手工传递到人行（当然，也可能上一级就是人行，这里不要太较真），每传一次，这笔资金都会在当前做业务的这一个银行的往来账户中流动，最后通过人行，流到你想转入的银行中，那个你手工填的单，也流到那家银行中。最最后，转入行

的业务人员核对单据，账号，户名都没问题，这笔钱就从往来账户划到我们所填的转入账户上去了。

在这些过程中，结算的同时就已进行了清算，资金的流向是

A银行的某支行　A银行的当地分行　A地人行　B地人行　B银行当地分行　B银行的某支行

也就是每一笔转账，在行间的这一步，都是通过它们在人行的资金往来账户，实现了资金的流动。

如果是上述的资金流向，就叫先直后横。如果是A地人行　B银行A地分行　B银行B地分行　B银行某支行这种方式，就叫先横后直。

这些单据的传递，都是手工的，或者说是落地的。如果是用信件的方式传递，那就是信汇；如果是用打电报的方式传递，那就是电汇。手工的传递都是有场次的，比如一天两场，或是一天一场之类的。所以这个转账的效率有多快，我就不说了。

现在科技进步了，手段丰富了，社会于是也就和谐的。先从我个人较为欣赏的大额支付说起。我一向认为大额这个业务设计得是相当的合理，因为资金是点对点，清算行对清算行，大大缩短了流程，更重要的是，信息的传递是自动的。还是上述的CASE，假设转出行与转入行都开通了大额业务，那么资金的流向是：

A银行的某支行　人行　B银行的某支行

原则上是这样的实现，当然行内的设计怎么处理我们就不多考虑了。行内当然也可以设计成为先从A银行的支行转到上级分行然后再发出，总之人行收到一笔大额的转账信息之后，是会自动、直接发向指定的转入行（假设转入行也开通了大额业务的话）

大额系统的对账，不考虑具体的客户账户，只考虑清算行。通俗的说，人行只管A银行今天给B银行转过去多少钱，转过去了，人行就不管了。至于B银行什么时候把这笔钱入到客户账户中，那是B银行的事，人行不管。听起来责任还是很清晰的吧，而且这样也有助于减少账户锁表而造成的行间转账失败。

因为大额的这种设计，所以实际转账中，几乎是实时的。我从某地信用社转到异地招行，在柜台还没最后签字，收款短信已经来了。

因为大额业务发生的时候，是支行对支行的，所以每发生一笔业务之后，实际上这笔资金是暂时体现在该支行的某个行间往来账户上。所以每天大额业务结束后，还需要按清算的流程，将这笔资金按往、来分别清算到上一级分行（或是总行吧，总之就是当地的最高节点），然后分行与人行

发下来的电子对账文件进行对账，检查汇总往、来数、金额是否相等。如果相等，那就可以把往来一轧差，转出多的时候就从存放在人行的账户里扣钱，转入多的时候就往那个账户里加钱。

至于这个清算的步骤，通常还是由手工发起，不过这里的手工，就不是指传递单据，而是指运行程序。当然，清算程序也可以自动运行，这个根据系统的不同，要求的不同，自行调整设计。

6.7 额度控制

和计息类似，可能有的系统没有把额度单列为一个模块来处理，而是仅仅作为业务模块之中的一个判断项。早期的业务中，的确可以这样处理。不过随着现在金融产品的不断推出，我个人认为还是把额度拿出来单独搞一下会更好处理一点。

比如说，一个账户，可能会有几次冻结，也能会有多项额度控制，每次的解冻，又或者是解除控制，都可能会对账户的额度造成不同的影响，如果夹杂在业务模块中，字段的设计，状态的控制可能都会有些问题，单独整成一个模块，或者说是一个大公函，在账务交易（或是账务模块中）的时候，用额度模块来进行一下判断，可以更方便的检测账户的可用额度是否足够。

另外，一些账户相关的透支什么的，也可以比较好的按客户来处理，而不是针对每个账户设置是否允许透支。以至于循环授信额度，这些概念都可以拿出来使用，简单的来说，有点类似于储蓄卡向贷记卡的管理方式倾斜，不过我没做过贷记卡，所以这里也提不出太多东西，只好拿个概念出来大家一起参详一下。

6.8 冲账

本着想到哪里就说到哪里的原则，刚才突然想起冲账还没有说，那么这里就说说冲账。

冲账的概念前面已经提过，这里我们指的，就是红字冲账。因为蓝字冲账就是再做一笔别的账务，从IT人员的角度出发，其实是另一个合法的正交易，不能算是冲账。

在设计程序的时候，只要是财务类的业务，就一定要考虑冲账的问题，不能偷懒，不能妄想测试人员会遗漏。就算别人忘记了测试，如果在真实业务中发生了问题，是很麻烦的，所以要养成良好的设计、测试的习惯。（这里不谈编码，因为设计好了自然就会写代码的）

关于冲账的实现，我知道的有两种方式：

第一是正反交易的概念。也就是普通的账务交易，称之正交易。每一个正交易，都需要有一个与之匹配的反交易，如果是按交易码来管理的话，可能会有一个标准来定义反交易的交易码，比如说正交易码加上5000就是相应的反交易之类的。（这里只是随便举个例子，比如说0001表示取款，那么5001就表示取款的反交易）因为冲账在账务处理上，具有一些共性，比如说都是按原来的财务的会计分录，只是金额为负发生账务即可，所以有可能会有一些公共函数来调用，不过总的

来说，都是小函数的概念。这种设计的缺点很显而易见，就是交易码，代码量都要翻倍。业务人员在冲账的时候也需要稍微算算交易码，有可能会输错。好处也是很明显的，就是程序之间互相不影响，修改维护都很容易。

第二种设计思路就是大函数的概念，也就是使用一个交易来实现冲账。因为前面说过，冲账业务具有一些普遍的共性，就基本原则来说，找到这笔正交易最初的账务，就可以了。所以使用一个大交易来实现。至于各个业务模块冲账后，在财务处理完之后的业务冲账，那可能就需要不断的在这个大交易中挂上各类外挂了。这种设计的缺点也很明显，就是维护起来很不方便，因为相当于把业务模块的冲账都集成到一个大交易中，在版本控制，大量测试的时候可能会有较多冲突。好处就是不占用交易码，也可以减少很多代码量，对于很标准的冲账，甚至不需要特别去考虑冲账的问题。（不过怕的就是不那么标准的冲账）

这两种方法各有优缺点，不知道大家的系统中，使用的哪种方式。这里我提出一个集合两者的第三种方法，一起来参详一下：

仍然考虑以大交易为主，不过大交易按某个参数表，来决定调用业务模块中的部分程序解决业务模块的冲账问题。如果是非常标准的冲账，就不需要刻意写冲账程序。如果是不标准的冲账，就在参数表中按设计中自已定义好的各类标识符，使大交易可以判断出何时调用业务模块中的冲账子程序（这些冲账子程序可以随时新增，名字也可以自定义，总之是在参数表中来定义）。至于大交易与冲账子程序中间的程序入口参数的传递，因为各个业务模块要求都有所不同，所以可考虑使用一个大字符型字段，或是数据队列传递字符流的方式来解决。

6.9 其它

暂时先想到这么多，其实还有其它的东西。比如说日终批处理，不过做到这一块的同行们想来不是技术骨干就是业务能手，也就没必要看这份入门级的东西了。还有拆借，贴现，不过这部分在核心系统里面占的份量很小，业务理解起来也不难，抓住一个熟悉业务的人多问问就问出来了。还有代理业务，不过这种业务的设计也多半是主+明细的方式（比如说代理单位的汇总信息，以及相应代理业务的明细记录），麻烦的可能反而主要在数据的交互上，也就是什么倒盘啊，信息录入啊什么的，又或者是具体的程序运行效率上，和这个整体设计的关系倒不大。

关于批处理，我做得比较多，还是再简单说两句。一般来说，会要求维护人员按各自的业务模块，维护批处理中的相应程序。不过最后，仍然需要一个总体上能把握的人来协调调度。批处理的程序大致上可以分为三种功能：

实现各类日终自动业务。比如说到期自动扣款（用过信用卡的朋友们应该深有体会吧），贷款的形态转移，储蓄结息（居然现在变成一年四结，有些先进的国外系统还要天天计提，我只能说系

统的设计出发点各有不同啊），可能还会有上面提到的日终清算。当然，还包括了各类的日初业务初始化。

实现账务模块数据向总账模块数据的转换，也就是更新总账模块的数据。严谨一点的系统，在更新了总账模块的数据之后，还会用程序来检查一下总账模块的数据与业务模块中的数据是否匹配，一致。（也就是传说中的总分核对）

生成各类报表。这部分可能主要是从总账模块中出，也可能需要综合一下业务模块中的数据。

批处理的发起，是由固定的操作人员来执行，没见过设计成按时间点自动运行的。

刚才说到批处理的三项基本功能，而其实在各类功能中，程序的运行顺序还是颇有讲究，不能随意乱放，否则可能会出现无法预知的问题。

批处理的排障，也是一个比较痛苦麻烦的事情，这里真诚的建议各位维护批处理的同行在有大模块上线前，做好心理准备，多多祈祷，实在不行还可以试试拜拜土地。

（正文完）

深耕境内/跨境支付架构设计十余年，欢迎关注并星标公众号“隐墨星辰”，和我一起深入解码支付系统的方方面面。

专栏系列文章PDF合集不定时更新，欢迎关注我的公众号“隐墨星辰”，留言“PDF”获取。

有个支付讨论群，添加个人微信（yinmon_sc）进入，添加时请备注：666。

隐墨星辰 个人微信

10年顶尖境内/跨境支付公司架构经验

著有《图解支付系统设计与实现》

备注666进支付讨论群