

资金之旅——走进账务清算域（二）：账务篇上

 寇恂 2016-07-05 12:12:48

blog

微账务

账务

资金

修改标签

上一篇：[资金之旅——走进账务清算域（一）：账务清算概览](#)

蚂蚁账务体系

当前从应用系统的维度看，我们共有5个账务系统：

账务核心（acctrans）——也叫做主账务，负责支付宝用户的余额资产和待支付资金的管理和记录，是涉及到客户资金的最为重要的一个系统。过去、现在以及将来都只支持人民币这一唯一币种。关于这个系统，比较有传奇色彩，可以多说两句：它诞生于2007年账务三期项目，在它上线时，直接导致了支付宝历史上最为惊心动魄的“17小时停机”事件。有兴趣的同学可以去网络上搜一下，该事件最近的一篇公开报道文章应该是《扒一扒支付宝背后低调到神秘的“带头大哥”》。acctrans是账务清算域的第一个系统，也是全站最后一个去掉小机的系统。

微账务核心（minitrans）——用于管理用户除人民币余额之外的其他借记类资产的账务系统。采用平台化方式搭建，现在上面支撑用余额宝、集分宝、预付卡、用信记录、境外商户保证金等超过30种的业务。

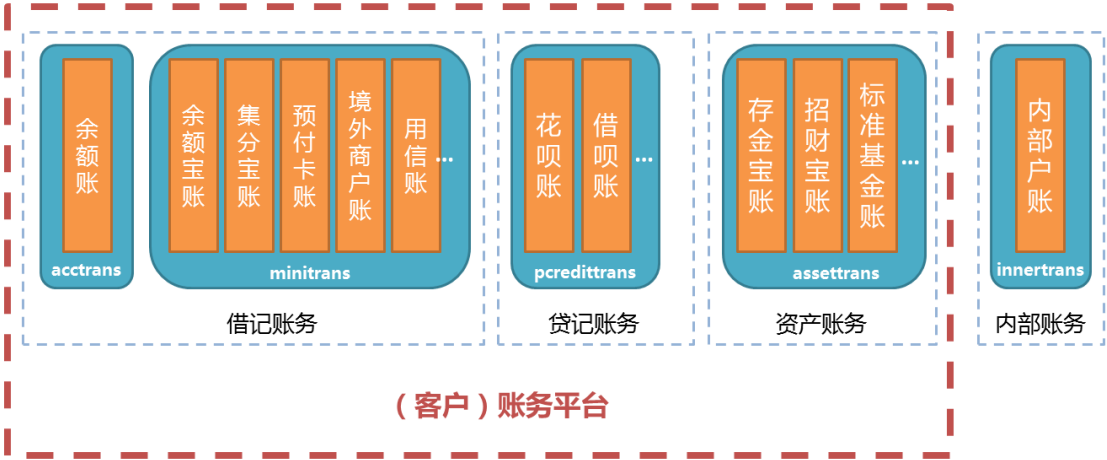
需要特别说明的是，其中的用信记录和其他账务不一样：其他账务的余额表示还剩多少可以使用，而用信记录的“余额”表示已经使用了多少。这是因为用户的授信额度的起始值并不是0，它是随着客户在蚂蚁体系的活动情况，以及一些临时活动在经常变化。这样的数值不适合放在账务中。故用信账务只记录了随着每次花呗、借呗正常交易后的占用额度，用户的可用授信额度由产品系统通过授信总额减去占用额度实时计算而得。

贷记账务（precitrans）——与minitrans类似，只是上面跑着的是贷记类资产，现在最大的两块业务是花呗和借呗。

资产账务（assettrans）——与基于余额数值进行记账管理的账务系统不同，它诞生的初衷是支撑单据类的资产管理。最典型的业务就是招财宝，用户在上面的每一笔资产，其实就是一条单据，随着自身的生命周期（申购、变现、赎回）而变迁。除了单据类资产，assettrans现在也支持类似余额数值的净值类资产的管理，代表业务就是数米基金的基金账。

内部户账务（innertrans）——账户大体可以分为两类：客户账和内部账。客户账的记账功能简单固定，内部账功能丰富多变，两者存在天然的矛盾。最开始这两类账户我们都放在acctrans中，后来为了尽可能减少对账务系统的变更影响，保障客户资金账户的稳定性，我们将内部户迁移到了新建的内部户账务系统中。innertrans的职责是承担内部账户（各种待清算、头寸、过渡户、待查户……）的管理。但不是所有的内部账户都在innertrans，由于历史原因，某些业务过渡户比如担保交易中间户、缴费还款过渡户本身已经深嵌在各个业务处理流程中，且他们自身的记账模式很固定，就先暂时仍然保留在acctrans中。不做伤筋动骨的迁移。

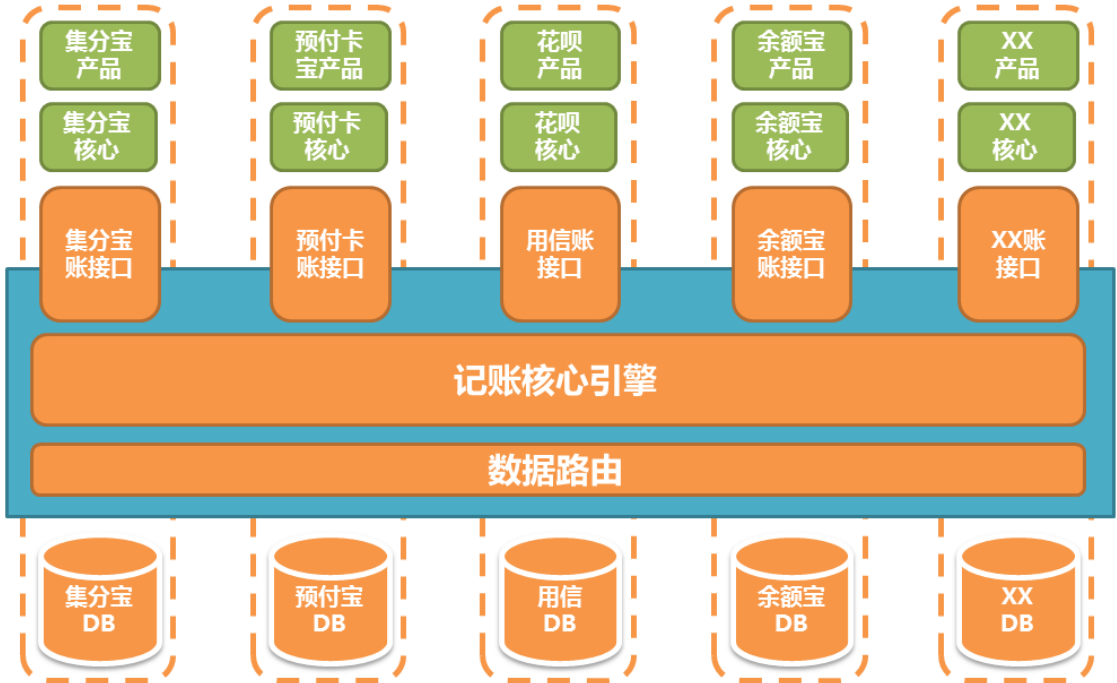
虽然以上五个账务系统名字都叫xxxtrans，但通常意义上所谓的账务平台，只包括的前四个账务系统。他们都是对客户资产进行管理的。而内部账户，主要服务于内部的资金核算管理，所以innertrans更适合放在核算平台。也就是说，**账务平台中的账务，特指“客户账务”**。



模板化的账务引擎

账务系统本身的业务逻辑并不复杂，在搭建acctrans时，我们基于分布式事务的两阶段处理开发了一套流程相对固定但又可灵活扩展的账务处理引擎，多年来经过大大小小的升级，这套引擎的骨架仍然运行至今，稳定可靠，令人信赖。在后面相继搭建minitrans、pcredittrans、assettrans、innertrans时，都使用了类似的结构，一方面本身这套引擎经过了时间的考验证明了其价值；另一方面也保持一个域内各个账务系统处理模式的统一，便于上手和维护。

随着时间的流逝，越来越多的业务都有记账的诉求。为了微支付，我们搭建了minitrans，紧接着又来了机票预存款，集分宝，是不是又要搭建一套“预存款trans”、“集分宝trans”？既然各个账务的核心处理逻辑都一致，何不将它剥离出来，加以一定的抽象，形成一个supertrans，一套代码上跑N种业务？这就是后来深入人心的平台化思想。但在实际设计的过程中，考虑到那时支付宝还是二代SOA架构，各个系统之间的调用链路成网状，为了确保上游各个业务产品系统的调用正确性，minitrans从façade层面进行了隔离，每种子账务都建立了专属的façade bundle，同时在数据层面也通过不同的数据源进行了隔离，形成了如下的架构：



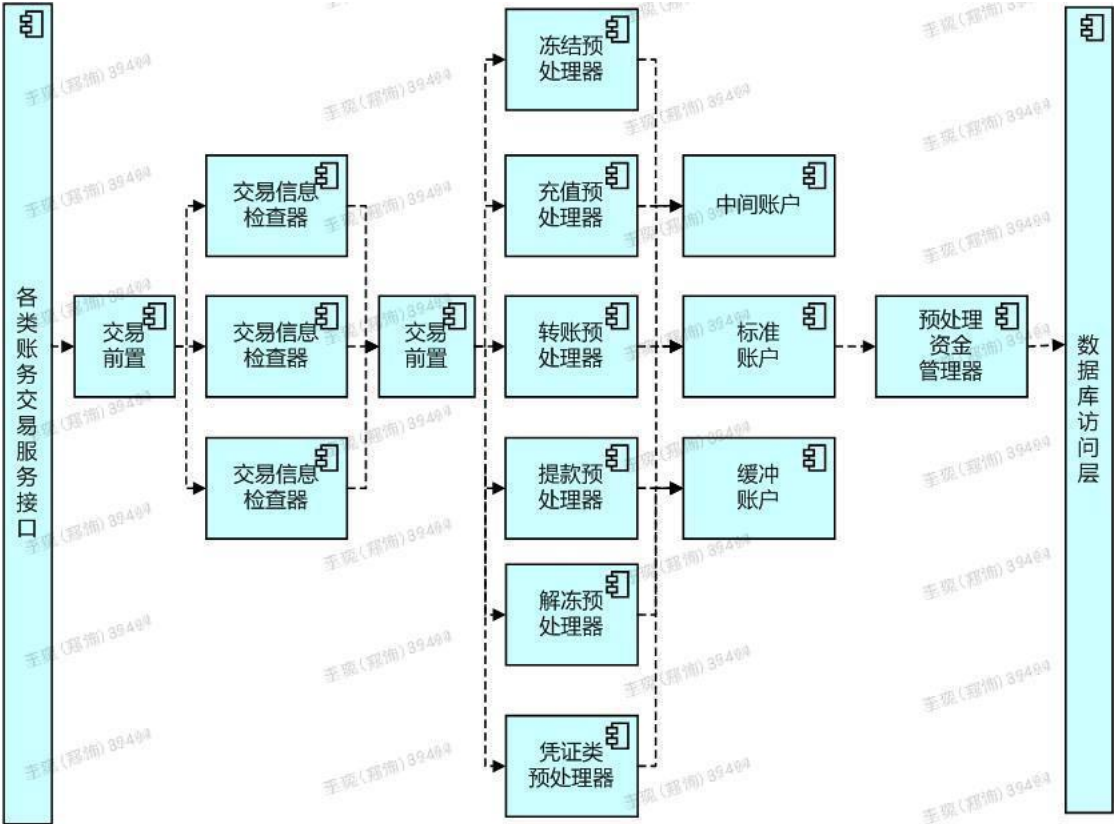
这种平台账务的设计思想，后面又依次应用在了贷记账和资产账上面，在开发效率和业务安全方面找到了一种平衡。下面我们来看看这套统一的账务处理模板是如何工作的。

账务两阶段处理流程

账务相关的数据，尤其是余额，相当于真金白银，对其准确性和一致性要求非常高。基于支付宝SOA化架构，各个业务系统的数据库与各个账务的数据库相互独立，需要用分布式事务来保证。所以从诞生之日起，账务系统天然就是XTS的参与者。

根据分布式事务的两阶段提交协议（Two Phase Commit）的要求，在第一阶段，也就是账务预处理阶段，需要做的事情是将本次业务活动所需要使用的资源预留下来；而第二阶段在提交事务时，就使用一阶段预留的资源，完成业务处理活动及数据持久化。

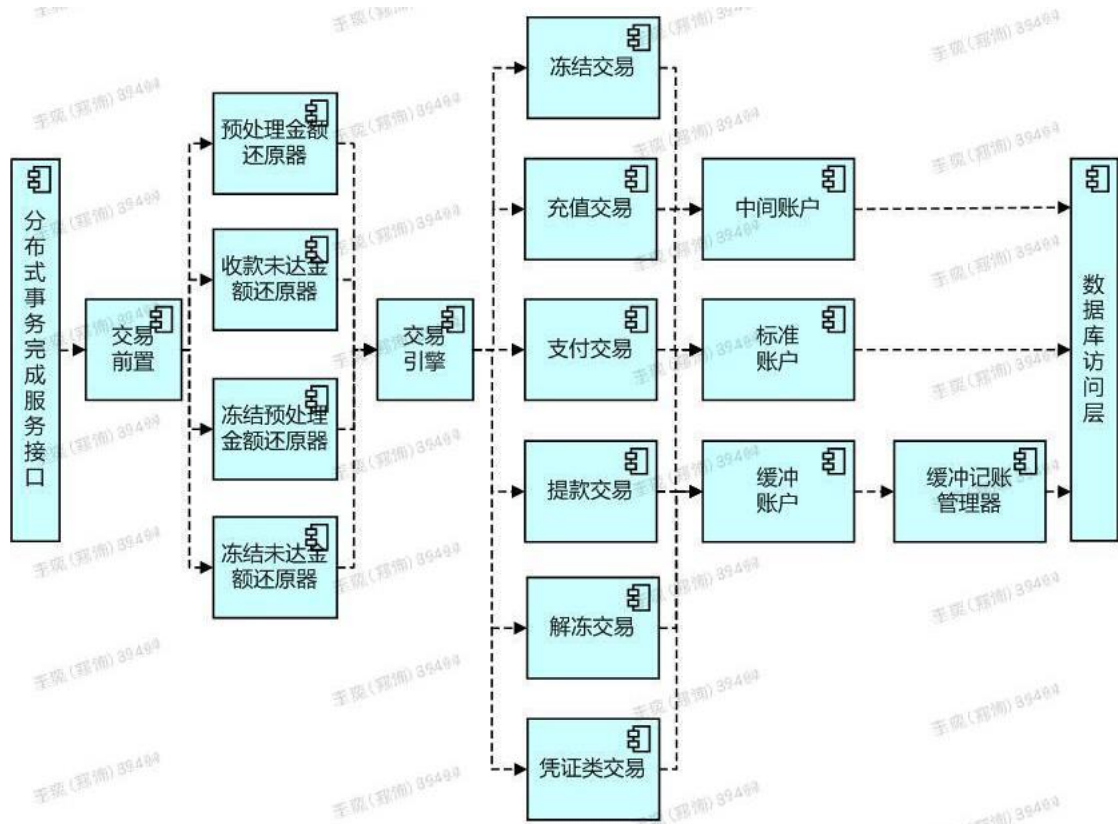
一阶段的处理流程如下图所示：



交易（记账请求）从façade端流入，首先经过交易前置处理，主要作用是进行必要的参数信息检查、幂等性校验，都通过后创建记录主事务和分支事务（一次主事务可以认为是一次完整的有账务参与的支付行为，一次分支事务代表一次账务调用，故一次支付过程可能有多次账务调用）。然后针对具体的记账类型，如充值、提现、转账、冻结、解冻，进行各自的资金预处理，即资源预留，将本次所要使用的资金给预先“冻”注，避免被其他请求给占用。

为了解决热点问题（后文会详细描述），账务系统设计了标准账户、汇总账户、担保交易中间账户（实质也是汇总账户）、缓冲账户等几种账务类型。目前每种记账类型，不同类型的账户处理模式有所差异，比如汇总账户和缓冲账户在这一过程中就不会去检查余额是否足够。

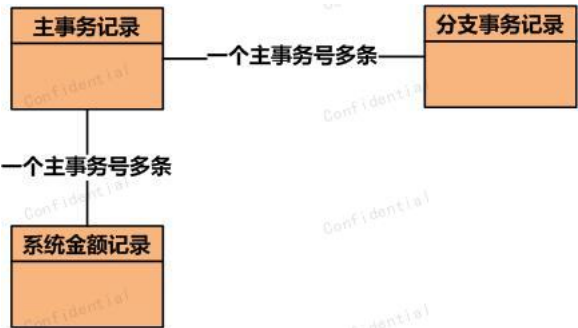
在账务二阶段，XTS可能提交也可能回滚。回滚的处理即将一阶段冻住的资金给解冻，账务状态还原。对于提交处理，流程如下图所示：



此时，需要首先将一阶段冻结的预处理资金给解冻，还原在开始的状态，然后干干净净地进行实际的充值、转账、提现等操作，包括更新余额，记录变更明细，发出通知等。同样，每种类型的账户的对应操作方式也不同，这些方法封装在各个的模型中。账务核心也是一个典型的领域驱动设计的系统。

基于余额的账务模型

除了资产账中的单据模型，其余账务系统的模型都比较类似。我们首先来看账务交易模型：

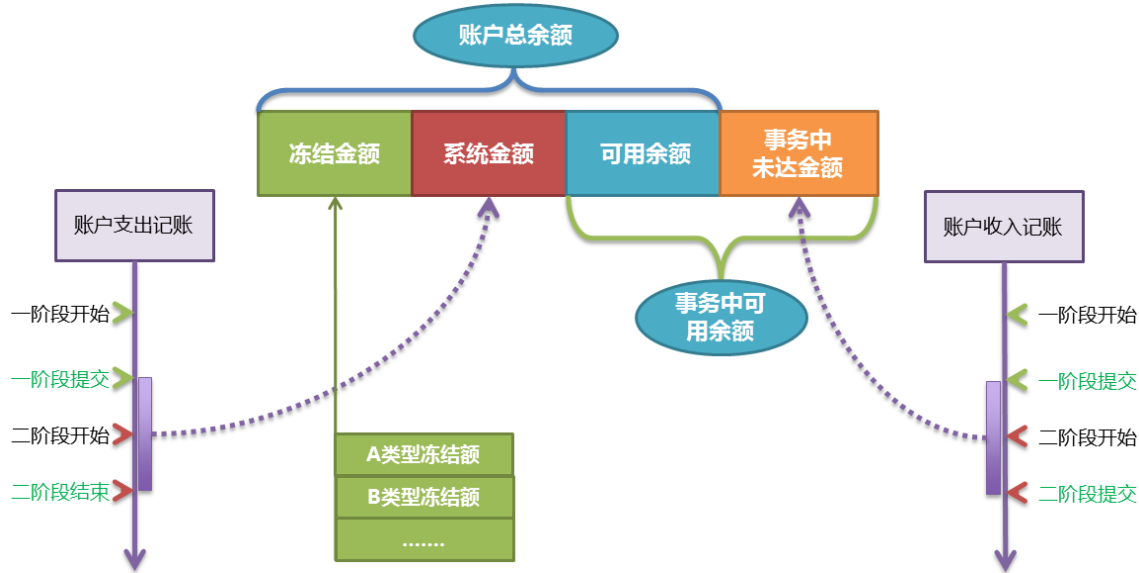


前文已经有说道，对于每一个涉及到账务的分布式事务号TXID，账务都会保留一条记录作为唯一性控制，叫做“主事务记录”。每一次具体的账务调用，账务这边会记录一条“分支事务信息”，数据中包含了必要的请求参数。在每条分支事务进行一阶段预处理时，为了预留资源而冻结的资金，会按照转出冻结、转入未达、解冻冻结、冻结未达等维度，生成对应的系统金额记录，这部分金额在同一条分布式事务中才可见。下面用一个例子更好的说明一次账务处理过程中，各个金额的变化。

首先，给大家展示一个公式：

可用余额 = 账户余额 - 冻结金额 - 系统金额 + 本事务未达余额

其中的“账户余额”表示账户中所有的资金金额，“冻结金额”表示业务冻结金额，比如提现冻结，保证金冻结，而“系统金额”就代表了被分布式事务预处理阶段因预留资源而冻结的资金。“本事务未达金额”表示该分布式事务中的应收款项，在同一事务中可以优先使用，对分布式事务之外不可见，即为0。用一张更加直观的图可以展示账务余额的组成：

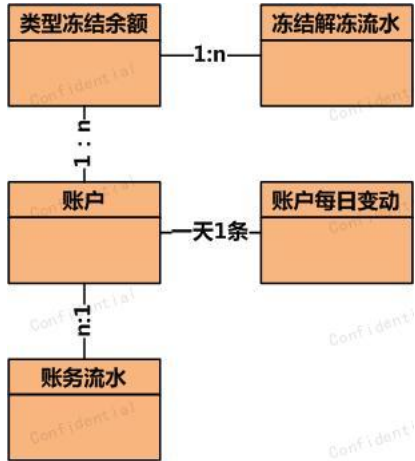


现在，假设有如下场景：

业务类型	交易金额	账户余额	系统金额	未达金额	事务中可用余额	事务外可用余额
初始	—	20	0	0	20	20
付款	5	20	5	0	15	15
收款	10	20	5	10	25	15
付款	15	20	10	0	10	10
收款	5	20	10	5	15	10
提交事务	—	15	0	0	15	15

从第二行开始简单讲解一下。起始余额20，付款5，需要将5先冻住，故系统金额是5，事务中和事务外可用余额都是20-5=15。第三行收款10，记在未达金额中，故事务中的可用余额是20-5+10=25，事务外对未达金额不可见，故可用余额是20-5+0=15。第四行，付款15，优先从未达金额中扣减，未达金额是10，直接扣成0，还差5，继续从账务余额中冻住，故此时系统金额是5+（15-10）=10，未达金额为0。事务中和事务外可用余额均为20-10=10。剩余步骤不再赘述。

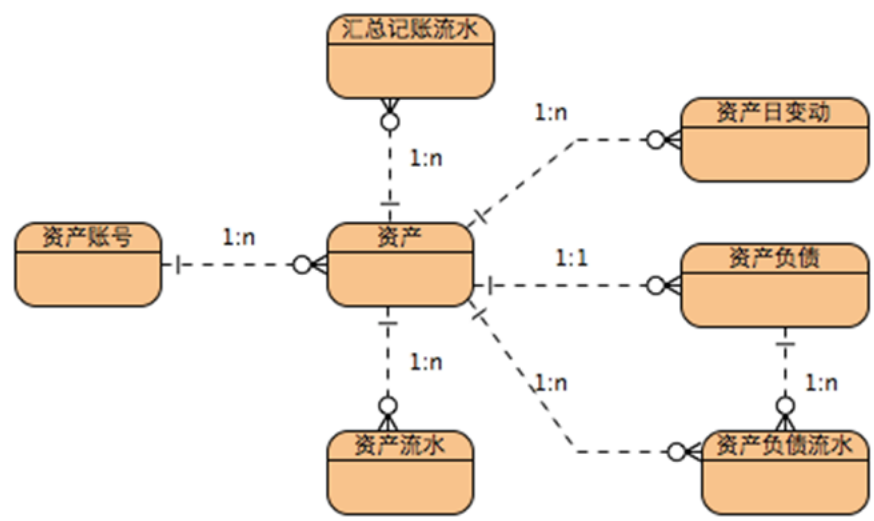
上面是账务交易部分的流程，关于账户本身的领域模型，可以参考下图：



在账户模型中，包含有账号、总余额、总业务冻结金额、总系统冻结金额等关键信息。账户余额的每一次变动，都会对应一笔账务明细，也叫做账务流水，在记账完成后，该明细也会通过消息的方式发送出去，供消费记录、会计等系统消费。这些变动也会按日（支付宝的会计日就等同于自然日）进行累计，记录每日的充值、提现、收入、支出生成额。此外，一些业务需要冻结或解冻资金，冻结与解冻不会影响账户的总余额，但会影响可用余额，本着专款专冻专解的原则，我们基于每种业务冻结类型，也记录了各自的冻结金额以及冻结明细。

基于单据的账务模型

资产账系统里有两种账务模型，和余额账务模型类似，用以支撑净值类资产，典型的基金；另外一种就是基于单据类型，典型的基金就是招财宝上可供变现可供赎回的资产。其模型如下：



基于单据类型，用户仍然会有一个资产账户，背后关联有若干资产。每项资产有其完整的生命周期，一般以购买为始，以到期赎回为终。对于变现（个人贷），实质上是资金借入人以这笔资产的到期本息为还款来源，向资金出借人进行借款。在这一过程中，不涉及到原始资产债权债务的转移，而是新创造了一笔对资金出借人的债务。所以，在上面的模型中，出现了资产负债的模型，描述的就是这种变现的情况。

可以比较上面两种不同的账务模型，也许有同学会发现两者有一定的相似性。在余额模型中，账户可以视为账号与资产是1:1的关系，而在单据模型中，两者是1:n。既然单据模型是后来者，那么是否可以复用余额模型，使用多个账户表示多笔资产？

我的理解是，其一，在资产账的场景中，资产的生命周期相对更敏感。虽然余额账户也有生命周期状态，但绝大部分账户不会走到生命的终态。而有期限的资产就完全不一样，其开始和结束是有固定标志，十分清晰明显，生命周期的每个阶段也有明确的业务处理逻辑。其二，用户理论上可以购买无限的资产，如果以账户余额去承载这类资产，那用户就需要开设无限的账户，从用户-账户的关系数量上，不合适也不允许出现这样的情况。其三，账号-单据资产这种模型，更贴近于实际体验和业务本质，更容易理解和维护。

关于资产账的记账示例，有兴趣的同学可以看看下面这张表格，重点关注下每一行相对于上一行的变化字段。就不逐步说明了。

业务类型		交易金额	本金	预期收益	保底收益	实际收益	余额	变现本金	已变现本金金额	已变现收益金额	已变现本金余额	已变现收益余额	系统金额	冻结金额	未达金额	资产可用余额
资产购买	资产购买	110	100	15	10	0	110	0	0	0	0	0	0	0	0	110
资产变现申请	变现：100，预处理冻结	100	100	15	10	0	110	0	0	0	0	0	100	0	0	10
	变现：100，二阶段	100	100	15	10	0	110	0	0	0	0	0	0	100	0	10
变现成交	解冻：本金90，收益10，预处理	100	100	15	10	0	110	0	0	0	0	0	100	0	100	10
	成交：本金90，收益10，二阶段	100	100	15	10	0	110	90	90	10	90	10	0	0	0	10
赎回	赎回：110，实际收益为15（预处理）	110	100	15	10	0	110	90	90	10	90	10	110	0	110	0
	赎回：110，实际收益为15	110	100	15	10	15	0	90	90	10	90	10	0	0	0	0
变现赎回	变现赎回：100	110	100	15	10	15	0	90	90	10	0	0	0	0	0	0

账务平台的那些事

在介绍完成蚂蚁账务体系后，接下来会针对账务平台中一些关键点做重点介绍，这些内容，无论对于账务清算域内的同学，还是对于跟账务清算域直接打交道的同学，都建议理解和掌握。

复式记账法

复式记账法是会计中术语，与之相对应的是单式记账法。可以看看两者的定义和区别：

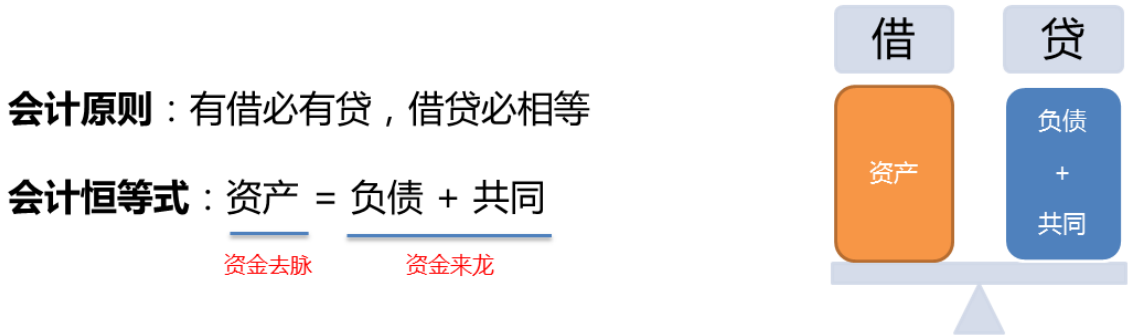
单式记账法——对发生的每一项经济业务，只在一个账户中加以登记的记账方法。故它只能反映经济业务的一个侧面，不能全面、系统地反映经济业务的来龙去脉。此外，单式记账法下，账户与账户之间没有必然的内在联系，也没有相互对应的平衡关系及勾稽关系，不便于检查账簿记录的正确性。

复式记账法——对每项经济业务按相等的金额在两个或两个以上有关账户中同时进行登记的方法。因此，复式记账法则更能反映经济活动中资金的来龙去脉，所有的发生额处于平衡的状态，各个账户或科目间存在严格的勾稽关系，能够起到资金核算和监督的作用。

复式记账法又可以分为借贷记账法、收付记账法、加减记账法。目前金融机构及绝大部分公司企业都使用借贷记账法。这是因为随着市场经济的发展，经济活动的多样化，需按**权责发生制**原则进行核算的业务大量增加。仅从资金收付或加减角度（收付实现制）来记录复杂的经济业务，会遇到很多困难。所以，在我国自1996年实行新的会计准则以后，就基本全部切换借贷记账法。如无特别说明，**复式记账法一般都指借贷记账法**。

在当年开始搭建账务系统时，老苗眯着眼睛深吸一口烟，天空飘来五个字：“复式记账法”，于是乎一群毫无金融背景的技术人员就连蒙带学地干开。因此，账务系统从诞生时就天然按照复式借贷记账的方式进行资金处理，与专业接轨。“复式记账”也是账务清算新同学加入组织的第一课。

不了解复式记账法同学，请自行搜索相关普及文章，厂内厂外一大堆，这儿就不再多说。大家只需要牢记这张图即可：



另外再补充说明一下，上面的会计恒等式是支付宝的一个变种。会计中一般都是“资产=负债+所有者权益”，或者“资产+费用=负债+所有者权益+收入”。这是由于支付宝的这一套账务系统及后面的会计核算系统管理的都是客户备付金，不存在所有者权益，收入费用什么的也不体现在备付金中。故用“共同”替换“所有者权益”，表示一些既可能体现在资产一方，也可能体现在负责一方的账户和科目。比如跟银行的待清算账户。

目前复式记账法只在主账务系统以及核算平台中的内部户账务系统、会计系统中使用。微账务、贷记账务、资产账务中仍然以单式记账法为主（记账基本都是单记增加、减少，后来资产账出现了内转场景，但仍然没采用借贷记账）。究其原因，是因为复式记账法要求严谨，账务处理相对复杂。而对于上述三种账务而言，基本上全是客户账户，每个账户的资产负债性质也已经确定，没有复杂的双边记账场景，记录每个账户的余额已足以满足前台业务及后台核算的需要，因此就简化了记账方式。

尽管如此，我们在每个子账务中，专门开设了一个总账户，其余余额就等于其他所有用户账户的余额之和。每日日终根据所有账户的收入支出情况汇总而得。这个账户，也可以用来和对应的人民币账户进行核对。比如，集分宝的总账户的余额，理论上等于主账务中集分宝沉淀资金账户的余额。

子交易代码的前世今生

子交易代码，可以说是账务的一个知名度相当高的代名词。它定义于各个账务系统中，但可以说已经贯穿于资金处理的方方面面。无论是直接面向用户的消费记录、商户账单，还是内部核对的各种脚本，抑或是离线数据的清洗关联，哪儿都有他的身影；无论是客户资金部的资金管理，还是产品经理的新业务接入，亦或是技术同学的开发测试，都不可避免要与之打交道。而且，由于子交易代码在代码中还是通过“阿里巴巴枚举”定义，一旦新增都要重新发布代码，为了确保各个系统能正常解析，每次在提交MVN变更时都需要提成不向下兼容。审批人痛苦，催审批的人也痛苦。因此，已经数不清有多少人问过：你们分配子交易代码的原则是什么？这次为什么要新增？

最开始的设计中，子交易代码被赋予了从核算维度进行账务记账的职责，但随着业务的爆炸式发展，这样或者那样的原因，子交易代码如今的职责已经泛化。归结下来有以下几个方面。

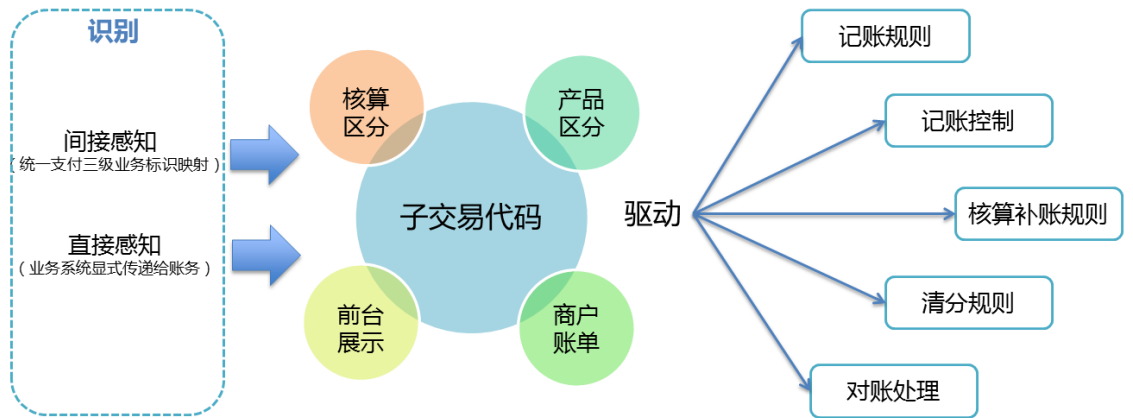
核算区分：主要面向客资，区分不同的核算方案。简单说来，一种新资金流的业务的接入，一般是需要新增子交易代码。比如，有一种新的转账业务，资金流不再是两个客户账户直接转账，而是先转到一个业务过渡户，待收款人确认后，再从业务过渡户转给收款人。那么这种情况，需要用一个新的子交易代码来区分。

产品区分：几乎没有人能够说得清支付宝有多少种产品，对于一些重要的产品，我们需要在资金管理时将它的资金变动数据与其他产品所区分，系统处理层面，也能相对细粒度地进行监控、限流等控制。比如在已经有交易退款子交易代码的背景下，考虑到基金业务比较特殊和重要，故新分配一个子交易代码专门区分基金购买退款的场景。这一块挑战是要区分好一个度，不能所有产品都独占一个子交易码，但也不能揉成一团，一个极端的反例就是CAE代扣，各种代扣业务都用301105，完全无法梳理。

前台展示：在PC端和钱包端的收支明细页面，部分场景有特殊展示需求。比如通常的入金业务，收支明细上都显示充值。但是余额宝赎回场景，需要调整文案为余额宝赎回，故针对该场景新分配子交易代码，收支明细页面也针对该自交易代码做特殊处理。

商户账单：这个也比较好理解，客户第一，商户尤其是大商户必须要服务好，如果在商户账单上有特殊要求需要区分资金，也会新增子交易代码。

以上是对外的部分。子交易代码作为账务清算域内部的“产品事件码”，域内很多处理逻辑都是由它来直接驱动。账务系统的记账流程编排，记账控制，核算域的补账规则、监管取数，清算域的清分规则，对账处理等等等等，都有子交易代码的影子。



蚂蚁的不同平台，都会有自己的处理规则和业务划分，一般都会定义自己平台内部的一种“码”。如同账务清算的子交易代码，金融网络有金融交换代码，统一支付有三级业务标识等。由于这些码分散在各个平台内部，各自为政，当需要将一条业务流水完整生命周期的数据全部串联起来时，大家发现十分困难。于是，从2016年初，全站开始了轰轰烈烈的“两码一号”治理运动。遵从“两码一号”的目标方案，子交易代码也将会逐渐全部退回到账务清算域内部，专注于内部处理逻辑的识别和编排。对外的衔接串联职责，将会由新的“两码一号”来承担。对此我们也跟目前管理子交易代码的客户资金部核算中心有过沟通，期望今后子交易代码将会缩小为：入金、普通出金、退款出金、内转、冻结、解冻以及相应的撤销等几种类型。

由于切换过程不是一朝一夕就可以完成的，可以预见在一个相当的时期内，子交易代码还是无法完全卸下自己的历史使命。

资金安全的兜底保障

资金记账管理，是账务平台一项显而易见的功能。但在我们看来，账务平台更大的价值，是它具备一系列手段，在实时在线支付链路中，起着资金安全的兜底保障作用，是全站最终的资金守护神。账务有以下兜底手段。

余额控制（Balance）。对于标准记账模式而言，账务在每一次记账的预处理和提交阶段完成后，都会去校验账户的总余额、可用余额、冻结金额、系统金额等，确保不会因为业务异常而导致透支。余额足够是账务记账的前置条件，各个产品一般都会在记账前进行相应检查，但账务仍然需要做此控制，对资金安全作最终确保。

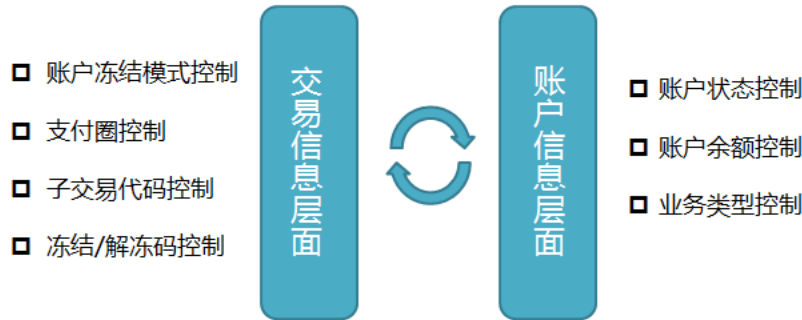
账户状态（AccountEnableStatus）。从粗的来看，账户状态大体可分为正常、冻结、销户三类。细化来看，冻结还可以分为“允许收不允许付”、“不允许收也不允许付”、“不允许收付但系统及后台操作的除外”等类型。对于最后一种，可以举一个例子，比如一个账户被冻结了，不允许付款。但是对于收费驱动的付款，系统可以放过。这个哪些场景属于系统驱动场景允许放过的区分，就是通过子交易代码来实现的。

支付圈控制（Channel）。支付圈可以理解为一个更粗粒度的支付场景。目前有三类：默认圈，航旅圈，理财圈。绝大多数商业生活场景都属于默认圈。航旅圈和理财圈从名字上就可以看得出来其场景。这个功能，可以允许我们对指定账号在不同支付圈的使用进行控制。比如，为了防止机票代理商利用机票款去做购物或转账操作，我们设置了该账户在航旅圈可以收付充提，但在默认圈只允许充提的控制规则。支付圈需要上游产品系统在调用账务是传递过来，账务系统匹配对应的控制规则后判断是否限制。绝大多数账号均只在默认圈里处理。

业务类型控制（BusType）。用来防止两个不同类型的账号之间相互转账。最典型的一个场景是，之前基金业务需要形成一个资金闭环，不允许和普通余额资金打通。也就是说，不允许用户用他的余额账户直接向基金公司的账户支付来申购基金。所以，我们给用户专门做基金业务的基金专户和基金公司的销售账户都打上基金业务类型的标记——F，这两个账户之间可以进行账务往来以完成基金申购或赎回。而用户的余额账户打上的业务类型是普通——D，当账务发现两个账户的业务类型没有交集时，将会拒绝他们的直接转账。所以之前我们要申购基金，都是先将资金从余额账户提现，并充值到其基金专户后，再支付给基金公司。

子交易代码/冻结代码控制（TransControlRule）。这是一个非常细粒度的控制措施，支持白名单与黑名单。比较典型的场景是司法冻结和风控冻结，通过配置白名单，限制了某个账号只能进行指定几个子交易代码的交易。

其他控制（ReserveChecker）。如交易金额大于零，两账号不能相等、压测账号拦截、账号科目校验等等。



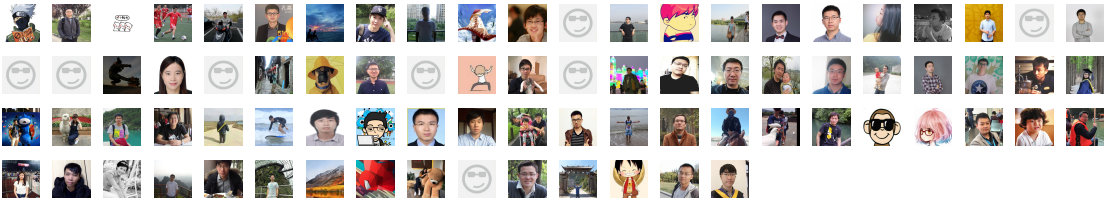
账务资金兜底手段

以上账务平台基于实时链路的资金兜底，再协同核算平台的日终核算兜底措施，共同筑起了账务清算域的资金安全保障网，发挥了巨大的作用。

下一篇：[资金之旅——走进账务清算域（三）：账务篇下](#)

他们赞过该文章





评论文章(6)



沐友 · 2016-07-07 17:42:10 置顶

1F

正好在户账分离项目中对接账务，现在我们遇到的一个问题是账务中的各种账的冻结状态无法统一，而且每种账户的账户类型也没有统一，我觉得后续账务是否可以考虑对这一块的账户属性进行统一管理，方便业务方使用？

0 | 1



寇恂 · 2016-07-07 18:11:43

你说的这个问题，一方面各种账从业务模型上来说是有差异的，如果大一统式地端平，也会引起一些困惑。另一方面，这些子账务在开发的时候，没有对命名等做统一的约束，也客观上造成了现状。目前的模式，在各个账务平台上保持统一的（主账、微账、贷记账、资产账），最大的问题在于账户的管理接口，并没有进行收归。像你提到的户账分离，开户时就需要识别开户类型，然后定位到该去调哪个服务，对客户端友好度较差。这一块我们将来会进行改进。



奕宏 · 2016-07-13 13:27:47

2F

期待来个 云的账务体系 和 蚂蚁账务体系对比

0 | 1



寇恂 · 2016-07-15 18:32:28

金融业务和主站账务大同小异，系统处理逻辑、功能都是同源的。只是背后的业务一个是银行存贷，一个是支付与互金。



沫枫 · 2016-07-31 17:20:08

3F

写得真好啦，收获良多，有没有定期培训之类的呀。

0 | 1



寇恂 · 2016-07-31 22:58:01

之前有过几次分享，但分享的形式太重了，有问题可以当面沟通



越恒 · 2016-11-06 13:06:56

4F

好文，学习了，感谢分享！

0 | 0



宫博 · 2017-07-06 15:07:02

5F

好文，赞！ 账户余额的处理方式没有理解，这样处理的好处在哪里？冻结金额 从逻辑意义上 = 系统金额(贷记金额)，能不能用一个替代掉？ 用字段识别类型，是为XTS而设计的吗？这种余额管理方式为账务的账户流水 提供了方便吗？


1 | 2



寇恂 · 2017-07-06 20:49:22

账户表中，冻结金额这个字段和系统金额是有区别的，冻结金额是指业务流程中冻结住的资金，是用户能够感知到的，比如消费者保证金冻结、提现冻结资金等。而系统金额指XTS两阶段过程中预留住的资源，这部分用户感知不

到。这两种资金，都是用户总余额（balance）的组成部分，并不是余额类型。也和流水没有关系。




宫博

· 2017-07-06 20:54:01

@寇恂

赞！说的很明白了，谢谢！



芊诺

· 2018-02-28 17:39:37

6F 好文章~~赞赞~~

0 | 0

文章点评

评论