通用对账系统介绍与设计(上)

王兴建 OmniStack 2017-09-13

本文首先介绍了对账的概念、基本内容，其次讲解了对账系统中常见问题及解决方法，最后详细讲解了整个对账系统的流程设计、整体框架。本文所说的对账是一个通用概念，不针对具体行业，各应用领域可根据实际情况进行调整。

对账系统简介

对账是金融领域中的名词，对应的学科为大家熟知的会计学。在金融领域中，不仅银行、基金、第三方支付机构需要对账，其他任何涉及金融交易的公司/机构都需要对账，比如商户业务、信贷业务等。

对账是指对前一个清算周期的交易信息进行核对，以确认交易信息的一致性和正确性的过程。这是普遍认可的一个概念，可以用一句话总结：确保单个周期内，信息流和现金流的一致。

通过对账可以保证账证一致、账账一致、账实一致，三者一致的正确、真实、完整为后续的佣金、分润等计算提供基础。对账的准确性也为系统、人工平账提供了差错信息，确保平账后达到财务一致。

总之，对一个公司来说，通过对账，可以正确地反映企业的财务状态，及时发现差错，确保业务健康发展。

对账大部分涉及两方对账，极少情况下会有三方对账的情况。三方对账本身的系统逻辑比较复杂，特别是三方平账时的差错处理更加复杂，这里不作讨论。

对于没有虚拟账户，只有交易通道类的对账，有两种对账类型：总分对账和明细对账。

总分对账即根据不同的交易通道，把每个交易通道的进出或者单独的交易类型进行汇总，按照不同交易通道、不同交易类型进行总对总对账，确保和每个交易通道的进出是一致的，确保每个通道不会有长款或短款的情况。

对大部分系统来说，总分对账没有差错就不用进行明细对账了。

如果发现总分对账有差错，就需要进行明细对账，将具体差错信息找出来。明细对账，顾名思义就是将发生的每一笔交易的详细信息和交易通道的对账文件进行逐笔核对，找出是否有不一致的交易。

$$\sum T-1\text{日日终各账户余额} + \sum T\text{日流入金额交易} - \sum T\text{日流出金额交易} = \sum T\text{日终各账户余额}$$

（图1 账户总余额连续性公式）

对于有虚拟账户的对账，还需要每个清算日对账户金额进行核验检查。核验主要包括两部分：一是账户总余额金额连续性检查，如图1公式所示；二是记账准确性检查。

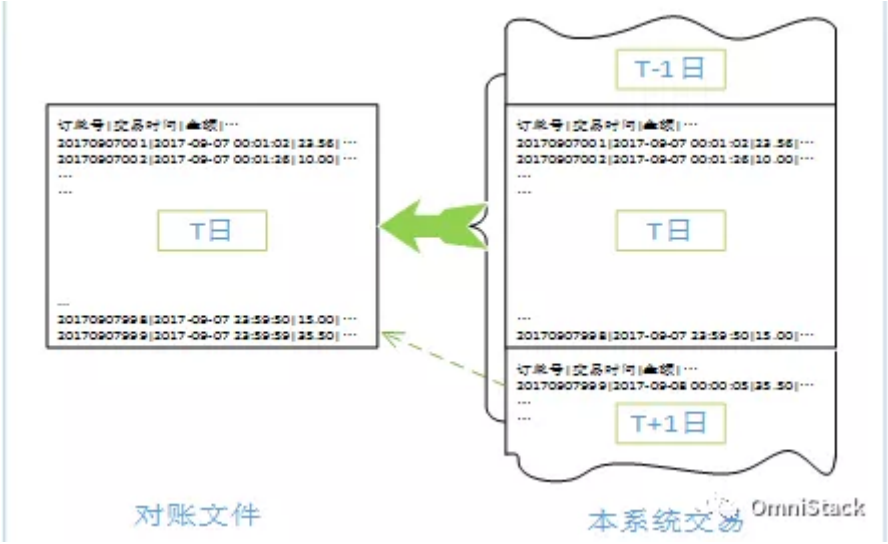
对账常见问题及处理措施

（图2 对账链路示意图）

对账一般都是以一方为对账基准进行轧账，出现差错时，通过各种差错交易。以基准方为准进行处理，最终达到平账。

对账的系统顺序：首先是金融交易最底层的银行内部进行对账以及平账处理，处理完毕后出具对账文件；和银行直接对接的第三方支付机构依此对账文件进行对账轧账，发现差错时，进行平账处理；依次往上，直到真正业务场景的用户。

从中可以发现，离用户越近的系统拿到对账文件的时间越晚，等最终整个业务场景链对完账，时间就比较晚了，比如 T+2、T+3，甚至更晚。典型的如消费金融公司，和支付渠道、合作方对完账，加上平账处理，基本T+2之后了，如图2所示。。



（图3 清算周期时间切分点示意图）

由于对账都是基于一个清算周期，清算周期就涉及到一个时间切分点，对账系统几乎都会碰到时间差问题。大部分系统都以一个自然日的起始，也就是0点到24点为清算日，但也有比较特殊的清算日规则，比如银联和银行之间的清算日是前一日的23点至当天的23点为一个清算日。

以0至24点一个清算日为例，0点为切分点，本系统发起的交易，到支付通道侧，可能已经是下一个清算日，从支付渠道自身来看，本笔交易会在第二天的对账文件出现，而不是前一个清算日。

如图3所示。对于这种切分点时间附近无法确认的交易，需要做一个时间窗口，时间窗口内的时间比清算日开始早一些、比清算日结束晚一些。

联机交易一般有严格的时间要求，比如必须在几秒内应答完毕，大于几秒就会造成客户体验差，流失大量客户。

而对账是典型的批量处理任务，几秒或者几分钟完成没有太大影响，但也不代表时间太长，几个小时就明显太长，会严重影响账务问题，无法开展日常业务。

对账是典型的批量处理任务，需要批量调度平台进行调度。对于比较复杂的调度逻辑（比如依赖关系调度、灵活触发、运营界面等），需要高可用、高并发、高伸缩的分布式调度系统，可以考虑用Zeus、Quarts、Elastic-Job、Azkaban等进行定制化开发。

明细对账

下面重点介绍一下明细对账以及技术实现方案。

（图4 明细对账总体流程）

明细对账的总体流程如图4所示，包括对账文件下载、文件预处理、轧账、平帐，以及运营平台对平帐过程的监控、预警，下面进行详细介绍。

对账文件下载

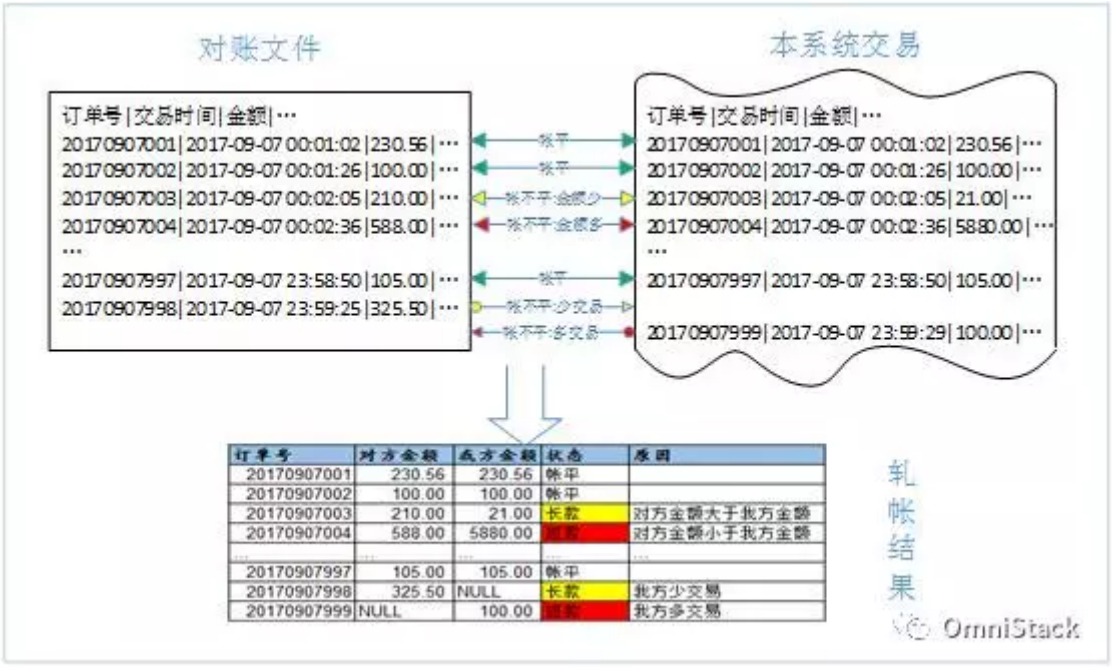
对账文件一般由对账对手方根据数据库的交易记录产生，并放置和对账方约定的地方，比如文件服务器、FTP服务器等。对账方根据事先约定的方式，获取对账文件，比如通过HTTP/HTTPS下载或者FTP/SFTP拉取。

对账预处理

为了安全性、客户保密性、防篡改，会对对账文件进行加密处理，常用的有3DES、AES等对称加密或者RSA非对称加密，以及MD5、SHA1、SHA512等摘要信息，还有各种签名机制。

获取到对账文件后，并不能立即进行对账，需要首先按特定安全机制进行解密，另外也需要把对账文件格式转换成内部系统方便处理的格式。这些都是轧账前的预处理。

轧帐



(图5 代收交易轧帐示意图)

轧帐是会计科目流程之一，定期或者企业需要时，核对总账与明细账，每日记账是否一致。轧帐是对账最核心的流程，用来确认双方或多方的账目是否一致，不一致的情况下，有哪些差错。

轧帐的正确与否，影响到后续的差错处理即平帐。以向用户代收为例，轧帐的流程如图5所示。

为了确认交易记录的标准，需要双方约定，哪个或者哪些要素可以唯一确定一笔交易，比如订单号，有时也会加上时间。

在具体对账的内容上，一般只需比对金额即可，某些情况下为了完备性，还会校对清算日、客户信息等其他明细数据。

某些非金额因素会影响到佣金计算、多方分润等，所以也需要进行轧账确认。

技术是为商业目标服务，业务发展到什么规模，就有什么样的技术与之匹配。所以，对账系统没有标准完美的技术解决方案，不同的业务场景会有不同的特性需求。

尽管行业、业务规模不同导致技术方案有一定的差异性，但是差异是相对的，其中还有很多共性。下面所讨论的轧账技术方案是较通用的，当然还有很多技术细节，不展开讨论。

依次读取预处理后的对账文件，根据交易标识在数据库中找出对应的记录，进行比对，检查是否一致。

此种轧帐的方式非常简单，容易理解，程序实现也比较简单。缺点也显而易见，支持数据量少，如果交易量大，会导致内存不足，并且大量数据库查询，会耗尽数据库资源，对联机交

易有影响。

另外，因为是串行处理，在交易量大时，会大大延长轧帐时间，影响正常业务的进行。

改进措施

将对账文件按一定规则划分为小文件，确保每次只处理记录数有限的文件；

按对账文件的记录数进行划分，由不同的节点进行处理，比如1-n由节点1处理，(n+1)-2n由节点2处理，(i-1)n+1-i*n由节点i处理，...充分利用分布式系统进行处理；

为了减少对联机交易的影响，可以在日切之后将交易记录导入到专门的对账库或者历史库进行对账；

在数据库中，新增和对账文件结构一致的对账表，将预处理后的对账文件按记录逐条导入到对账表中。利用数据库提供的join, left join,right join, case when等SQL语法进行轧帐。

比如SQL：select case when a.amt<>b.amt then 1 else 0 end from a join b on a.order_no=b.order_no，可以把金额不一致的找出来。

select case when a.amt is null then 1 else 0 end from a join b on a.order_no=b.order_no，可以把本地系统中不存在的交易的找出来。其他以此类推，不一一介绍。

这种轧帐逻辑放在数据库，简单方便，后续方便扩展；对账对手方的数据都存储在数据库，可以非常容易可视化和排查问题。

缺点是：交易量大时，导入量大，数据预热慢，数据库性能很容易成为瓶颈；所有的计算节点集中于数据库，无法利用分布式。

改进措施

批量导入数据库；

根据轧帐标识进行分表分库处理，并进行汇总。

Redis是高性能的key-value数据库，支持5种基本类型(String, List, Set, ZSet, Hash)以及5种基本类型的扩展。对于Set数据类型，天然的支持交集、差集、并集等集合运算。

将预处理过的对账文件按轧帐标识和比对元素加载到Redis的一个Set A，本系统的交易记录也一并加载到Redis的另外一个Set B。Set A和Set B的交集即为对账对平的交易。

Set A-Set B差集并不是多的记录，还需要将每个元素和Set B-Set A的差集进行查找，如果有轧帐标识一样但比对元素不一样的，即为差错交易；如果没有轧帐标识一样的元素，才能判断是多的记录。同理，可以找出少的记录。

这种借助于Redis轧帐，是一种纯内存计算，预热快，速度快，充分利用Redis的成熟计算类型，没有复杂的程序逻辑处理，避免出错。但缺点是计算机内存有限，不能支持海量对账数据。

改进措施

按一定规则，进行数据分片，不同的数据按照轧账标识分片到不同的Redis，最后再进行汇总，但同时也增加了轧帐复杂性。

如果把轧帐进行数学模型抽象的话，可以抽象为一个典型的算法问题：构建2个无重复元素的集合，按照一定比对规则找出2个集合的差集、交集，对差集和交集的元素属性进行计算比对，找出差异性。

通过抽象后，我们可以看出，有很多实现方法，比如Java JDK提供的集合计算Collections、擅长搜索的Elastic Search等。

这种实现方式需要视具体情况进行分析，有些实现需要花大量的时间和逻辑在数据预热上，有些实现会导致汇总数据的逻辑复杂化。

总之，对账系统涉及公司的账务会计核心，尽早建立完备的对账系统，可以推动更加精细化掌握业务情况，特别是互联网金融公司。

在对账系统落地时，可以综合考虑本文提出的方案，找出适合公司业务发展的技术方案，尽量使用成熟技术解决业务问题，减少技术风险性。

本文的下半部分将讲述如何处理海量数据的对账，以及平帐/差错处理和虚拟账户对账，敬请期待。

作者介绍

作者介绍

王兴建

现任上海秦苍（买单侠）信息科技有限公司软件架构师，加入秦苍之前，曾在中国银联、证通任职。

专注于Java Core、NoSQL、分布式服务框架等技术领域，对移动支付、客户认证、红包、虚拟账户、信贷等业务领域有丰富的经验，擅长将成熟技术应用于业务。

目前主要负责秦苍运营商合作业务的架构设计，以及技术在业务中的落地工作。