基于.NET 中间件的自动催费解决方案

Solution of auto – urge charge Base on . NET Middleware

陈文斌 （新疆石河子开发区汇通信息技术开发有限责任公司 .832000）

摘要：本文通过对某电力客户服务中心原有自动催费业务的分析，提出了利用.NET 中间件解决异构数据库环境下自动催费的思路，对中间件涉及的关键技术如异构数据库数据同步技术、异构数据库的事务处理技术进行了分析和说明。该中间件可以广泛应用于其他客户服务中心的自动催费业务。

关键词：自动催费 .NET 中间件 异构数据库 事务处理

1 自动催费功能及存在的问题

“电力客服”原来自动催费的业务流程主要由五个环节组成（如图1），各个环节主要任务如下：环节1：用电营销部根据催费策略选择欠费客户并提交。环节2：用电营销部电话告知“电力客服”值班人员催费。环节3：“电力客服”值班人员接到电话后打开欠费客户记录。环节4：“电力客服”值班员根据欠费记录生成催费外拨记录。环节5：催费记录进入排队机等候排队外拨。为叙述问题方便，将整个流程划分为两个阶段：“催费前期”、“催费后期”，1至4环节为“催费前期”，主要解决自动外拨前的数据交换问题，第5环节是“催费后期”，主要解决自动外拨问题。自动外拨由CTI服务器指挥IVR根据外拨事件列表自动完成，其相关技术已经很成熟，功能也可以满足要求，在这里不做过多讨论。自动催费主要解决的是”催费前期”的问题，一是数据交换问题，二是处理流程的自动化问题。“电力客服”建立在用电营销系统之上，客户档案、电费信息均在营销系统中。营销系统采用的是Sybase数据库，“电力客服”采用Oracle数据库。催费信息需要从营销系统中提取，因此自动催费首先要解决异构数据库的数据交换问题，即如何把用户欠费信息从Sybase数据库中自动导入Oracle数据库中。原来的设计是：1、在Sybase库中建立中间表，中间表的数据从营销系统中获得。2、“电力客服”从中间表读数据到Oracle库中形成催费表。原设计虽然成功解决了异构数据库间数据交换问题，但却忽视了处理流程的自动化问题。自动催费的目的就在于尽可能地减少人为干预，批量、自动地外拨催费，而原有流程除排队外拨外，每个环节都需要人工启动，致使用户感到很不方便。从这个意义上说，原有系统只是实现了自动催费的数据交换问题，没有解决”催费前期”流程的自动化问题，它存在以下问题：

（1）催费只能人工发起。原来的自动催费系统必须由营销部人员根据欠费信息发起催费流程，而人工发起催费必然造成启动时间、催费客户量的不紧凑，造成外拨服务忙闲不均，催费效率低下。

（2）催费环节需要手工进行连接，受人为因素的影响，容易出现催费信息在各环节停留时间过长、部门之间扯皮等现象，造成有效催费时间缩短、催费不及时的不利后果。

（3）操作复杂。催费名单的确定需要抄表人员从欠费表中逐一筛选过滤完成，工作量大、时间长且容易遗漏和发生错误。

2 解决方案

鉴于原有催费流程中存在的问题，我们对原有流程进行了改造，主要思路是利用中间件技术对”催费前期”处理流程进行“自动化”改造。改造后的业务流程如图2。

与改造前相比，改造后的流程增加了中间件。所谓中间件实际上是一个伺职催费启动、控制的软件，它负责监听并启动催费流程的各个环节。例如：到了催费期，中间件会根据催费策略自动启动营销系统的欠费生成模块，自动完成欠费单的生成；之后自动启动客

户服务中心的催费生成模块，完成催费记录的自动生成，最后自动生成催费外拨记录。整个过程不需要人为干预，计算机将根据催费策略自动完成整个催费过程，真正实现了催费的自动化。从根本上解决了原有催费系统存在的问题。

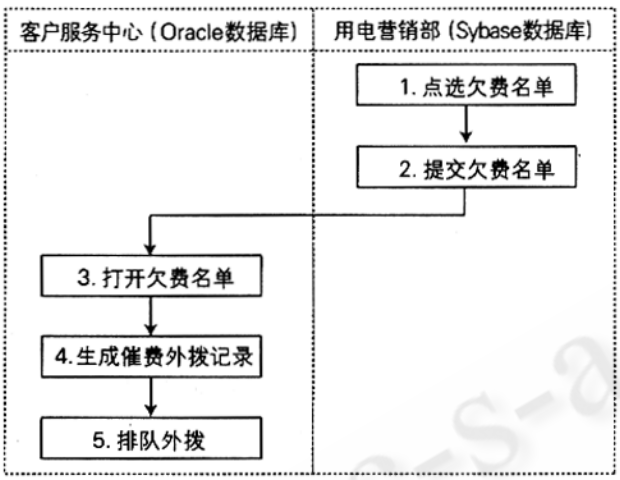

图 1　改造前的自动催费业务流程

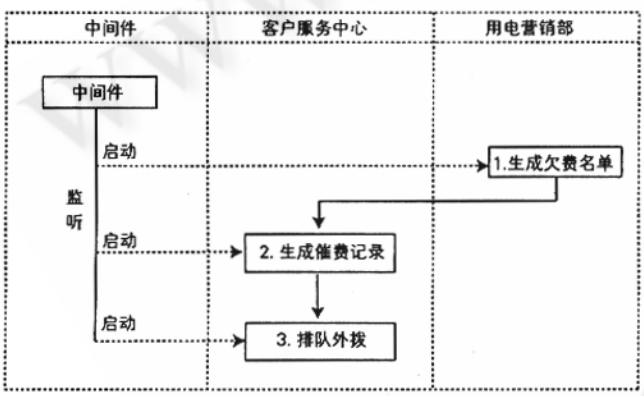

图 2　改造后的自动催费流程

3　关键技术

(1) 中间件及控制原理。本文的催费中间件是一个三层结构的软件(如图 3)，包括控制层、业务逻辑层和数据访问层。控制层主要负责监听催费策略表，根据监听信息调用业务逻辑功能及驱动催费流程。业务逻辑层负责为控制层提供催费策略、数据同步、事物处理等业务功能服务(详见关键技术说明之“2”、“3”、“4”)。数据访问层为控制层和业务逻辑层提供数据服务，包括多数据源的连接、数据集访问、数据集更新、存储过程调用等功能。中间件的控制层采用定时器触发机制，根据催费策略自动触发，其实现原理如图 4。值得注意的是，定时器“Time3”不是必需的，可以和“Time2 合并”，但是必须牺牲“当前时间与 ΔT(催费时间范围)”的比较时间(见表 1)，即该时间必须大于等于执行催费流程所需要的时间(不包括自动外拨)，否则会造成多次启动催费流程。

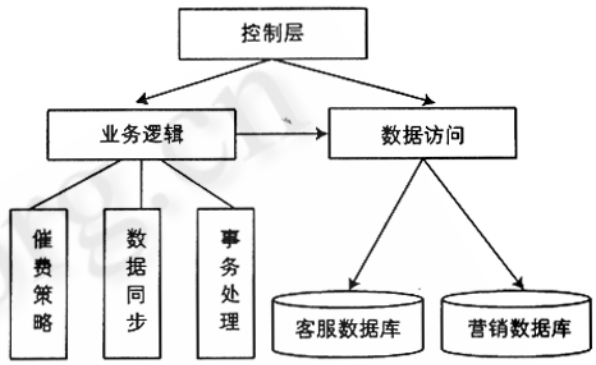

图 3　中间件结构

(2)催费策略。自动催费的启动、控制是由中间件根据催费策略自动完成的。催费策略包括催费条件和催费控制两部分内容。催费条件包括催费启动时间(立即启动、一般定时启动、特殊定时启动)、筛选条件(户号范围、欠费金额等)，信息保存在用电营销数据库之中。催费控制包括催费开始时间、终止时间、催费次数、时间间隔等内容，信息保存在“电力客服”数据库中。催费条件由用电营销部制定和维护。催费控制由“电力客服”制定和维护。催费策略的原理如图 4。

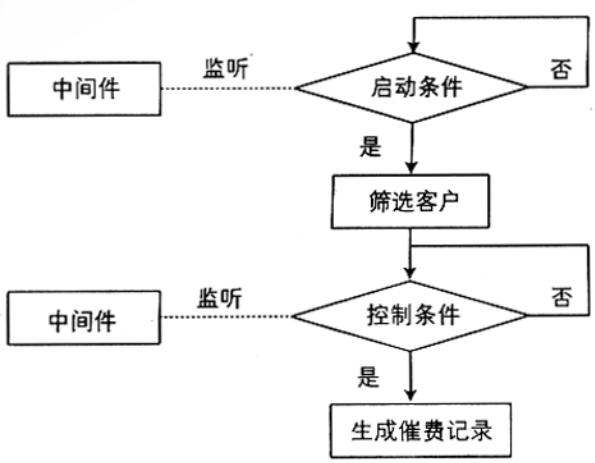

图 4　催费策略

(3) 异构数据库数据同步。本文“电力客服”自动催费涉及到两个不同的数据库,中间件需要访问这两个数据库,而且欠费记录生成的同时,要产生催费记录及催费外拨记录,欠费记录的更新需要同步更新催费记录和催费外拨记录。对这个问题可以有几种解决办法,传统的方法是采用循环读取 Sybase 库中的欠费表,然后插入 Oracle 库中的催费表的相应位置。我们采用的是. NET DataSet 的 Merge 方法。关键代码如下:

表 1　控制层实现原理

定时器	Time1	Time2	Time3	Time4
启动、停止条件	每天在催费时间以外(如凌晨时间),读取催费策略的催费时间范围 ΔT,若读取成功则启动 Time2,否则继续	每隔 n 秒对当前时间和 ΔT 进行比较,如果当前时间在 ΔT 之内则启动 Time3 停止 Time2,否则继续比较	启动催费流程,如果执行成功(不包括自动外拨)则启动 Time4 停止 Time1、Time3,否则停止 Time3 启动 Time2	如果时间超过催费截止时间则停止 Time4,启动 Time1
建议时间间隔	每天	1 秒	大于催费流程执行需要的时间,也可以设置成足够长,如 1 小时	10 秒

```
//---从营销系统中得到欠费名单
connStr1 = connectionString1;
DataSet ds_sql = new DataSet();
//从用电营销欠费表中选取“户号”、“总电费”,存入 ds_sql
sql1 = " Select user_no,total_money from arrearage";
OleDbDataAdapter da_sql = new OleDbDataAdapter(sql1,connStr1);
da_sql.Fill(ds_sql," dt_sql");
DataTable dts = ds_sql.Tables[" dt_sql"];

//---从客服系统中得到催费名单
ConnStr2 = connectionString2;
//从客户服务中心催费表中选取“户号”、“总电费”,存入 ds_sy
string sql2 = " Select user_no,total_money from urge_charge";
OdbcDataAdapter da_sy = new OdbcDataAdapter(sql2,connStr2);
da_sy.Fill(ds_sy," dt_sql");
//根据欠费名单生成催费名单
DataTable dtt;
DataTable dtm = ds_sy.Tables[" dt_sql"];
//将 dtm 中变化的数据放入 dtt 中
dtt = dtm.GetChanges();
//根据 dtt 更新 ds_sql
ds_sql.Merge(dtt);
ds_sql.AcceptChanges();
```

需要注意的是,第一,要更新的异构数据库表的字段名必须相同,如果不相同可以采用别名的办法处理,如 select user_id as user_no,charge as total_money from arrearage。注意不同的数据库的写法。第二,Fill 方法必须指定相同的 tablemapping。

(4) 异构数据库的事务处理技术。自动催费功能的关键是图 2 中第 1、2 步的实现,即生成欠费名单和生成催费记录,而且要确保全部成功或失败,因此,必须采用事务处理来实现。由于是异构数据库,不能采用数据库的事务处理机制,我们采用了. NET 的 Component Service 事务处理技术,实现过程大致如下:首先新建一个. NET 的事务处理组件,该组件需要添加支持事务处理的引用 System. EnterpriseServices。事务处理组件必须继承 ServicedComponent。由于 Component Service 基本结构是面向 COM 的,所以需要将事务处理组件作为 COM 对象用 Component Service 进行组册。为此需要做两件事:

第一,使用 Sn 工具把事务处理组件进行强名称处理(生成密钥)。Sn 是. NET 软件包的一个命令行工具,位于. NET Framework SDK 路径下的 bin 目录下。使用时加“-k”选项,并说明组件密钥的位置和名称,如:Sn - k “组件路径\组件名. snk”。生成密钥后要把 AssemblyInfo. cs 中的[assembly: AssemblyKeyFile(“”)]及[assembly: AssemblyKeyFile(“”)]替换成[assembly: AssemblyKeyFileAttribute(“组件路径\组件名. snk”)],最后重新进行编译。

第二,使用 Component Service 的注册工具 RegSvcs 进行注册,RegSvcs 是. NET Framework 的一部分,

(下转第 47 页)

(上接第 50 页)

可以在. NET 的命令行状态下直接使用,如 RegSvcs "组件路径\组件名. dll"。注册完成后,打开 Component Service 控制台,可以看到新建的事务组件,但此时它并没有事务处理的能力,还需要打开组件的属性框的事务页,把事务支持可选项置为"需要新建"。也可以在事务处理组件源代码上添加属性,这样可以省去手工设置组件的事务处理属性。具体做法如下:

```
[ TransactionAttribute ( TransactionOption.
RequiresNew) ]
    public class 事务处理组件:ServicedComponent
```

4 总结

本文应用. NET 中间件、异构数据库的数据交换、事物处理等技术较好地解决了"电力客服"自动催费业务,而且催费中间件只与应用数据库有关,独立于原有应用系统的任何处理过程,对原有的应用系统不需要做任何修改,对"电力客服"的正常运行不会产生任何不良影响。自动催费中间件可以广泛应用于其他客户服务中心的自动催费业务。

参考文献

1 李安渝,《Web Services 技术与实现》,国防工业出版社,2002。

2 Simon Robinson K. Scott Allen 著,杨浩等译,清华大学出版社,2002。