基于 PB 多用户机制的实现

杨　炯　（中国石化工程建设公司信息技术中心 100101）

摘要：本文从程序设计的角度出发，说明了一个管理信息系统的多用户机制的实现方法。

关键词：多用户　数据库　依赖性　处理逻辑　PowerBuilder

1　前言

现在的管理信息系统由于几乎都是在网络环境下实现的，从而使多用户机制成为它的一个主要特点。如何实现这一多用户机制，可以有两种解决思路：一种是利用后台数据库系统的多用户机制来实现；另一种是通过开发自己的管理模块来实现。前一种方法实现起来很简单，但局限性很大，也缺乏灵活性，而且必须依赖于所使用的数据库系统。后一种方法是本文要详细说明的，这种方法的好处是多用户机制的实现独立于所使用的数据库系统，使用起来非常灵活，而且全部功能都可以被程序所控制。

2　设计思路

本文方法的整体思路是将所有的用户管理功能都由开发工具所开发的管理功能来实现，将特定数据库操作功能分离开，通过一个物理的数据库系统用户完成与系统的链接后，不再使用数据库系统的功能。通过这种方法将系统的依赖性控制在最小的范围内，使得本文介绍的方法可以适用于更多的数据库系统。图 1 是一个简单的概念图例：

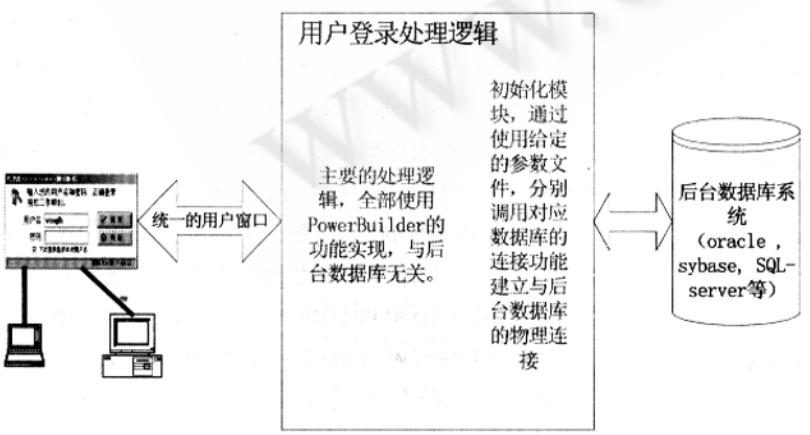

图 1　用户登录处理概图

与传统方法的比较：

(1) 传统的用户管理一般都交由后台数据库提供的用户管理功能来处理。这种方法使用效果的好坏取决于所选的后台数据库系统，如果数据库系统所提供的功能有限就有可能达不到我们的要求，如果所提供的手段比较复杂就又不容易操作。

(2) 用户管理是一项安全性比较高的操作，需要具有系统管理功能用户的权限，如果在用户业务逻辑中频繁出现此类的操作必定增加了安全隐患，对于提高系统的安全不利。

(3) 不同的数据库厂家对于自己的产品中有关用户管理的方法是彼此不同的，要使用系统提供的方法就要针对每种数据库系统写相应的处理逻辑，对于系统的移植和重复利用都非常不好。

本文介绍的方法只利用数据库的基本数据处理功能在其上实现自己的用户管理，即使在没有用户管理功能的数据库系统上也能实现相同的用户管理功能。对于安全性问题，本文所使用的方法只需要一个一般权限的用户即可，大大提高了系统的安全性。对于通用性，本文方法的处理逻辑完全通过开发工具提供的功能实现，没有利用数据库的专有特性，因此适用于各种数据库系统。

3　实现方法

实现方法分为两个部分：数据库基本表设计和处理逻辑设计。这种方法已经在作者开发的很多系统中得到了使用和验证，下面的介绍就以一个已实现的系统作为例子。

3.1　数据库基本表的设计与解释

本文介绍的方法将用户管理与用户操作模块的权限管理统一进行了考虑。实际情况也是这样要求的。通过实际的数据库用户连接后，我们自己开发的功能模块就接管了后面

的所有操作，通过用户验证处理逻辑来检验用户的登录合法性。这些逻辑帐号实际就是存储在数据库的基表中的数据。下面将详细说明实现这种功能的数据库设计，图 2 是基本表的 E－R 图：

表的结构很简单，概要解释如下：Operate_user 表中存储的就是允许进入系统的每个逻辑用户的信息，主要有用户名、登录密码等，在该表中密码是加密存储，这样即使是有人通过非正常途径查看到该表也是无法知道密码的。Application_menu 表中存储的是已开发的所有模块，Object 列中是实际可以被调用的 Powerbuilder 对象（后面有详细的解释）。Operate_item 表中存储的是上述两个表的对应关系信息，也即什么用户可以操作什么对象。上述三个表中的内容都在系统中有相应的模块去维护，也就是说通过这种设计，在系统中【功能模块维护】、【用户管理】以及【用户权限】都是作为一个可以操作的功能交给用户去使用了。对于触发器的使用只是为了保证相关完整性的需要，没有什么特别的意义。

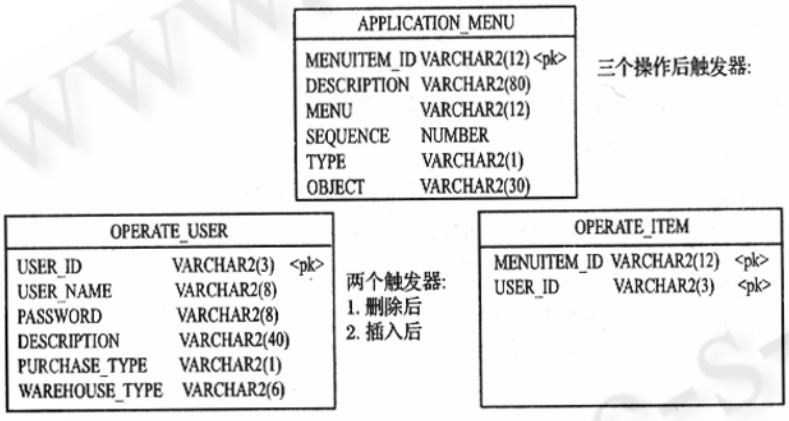

图 2　基本用表 E－R 图

以上三张表就是本文方法使用到的全部表，由此也可以看出数据库依赖性的简单。它们是建立在一个具有普通数据库操作权限的用户下，如果为了操作方便也可以将其他业务用表创建在同一个用户下（这也简化了数据的备份与恢复）。通过该用户帐号与数据库系统建立连接的过程都是在用户登录处理逻辑的初始化模块中完成的，对于最终用户来说是透明的。只要应用系统能够通过这样一个帐号存取这些表的内容，就可以实现我们介绍的用户管理功能。而所有功能的实现利用到的只是数据库最基本的数据操作功能，这些功能是标准的 SQL 语句，是所有数据库系统都支持的。为了提供一个起始可登录系统的用户，在建立数据库后，就在 operate_user 中追加一个系统用户。此项操作可以通过在数据库建立脚本中添加一个标准语句来实现。

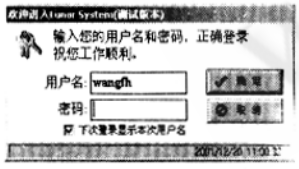

图 3　用户登录窗口

3.2　处理逻辑的设计与解释

本文介绍的方法是使用 sybase 公司的开发工具 PowerBuilder 来实现的，该软件是一个支持第四代语言的面向对象式的高效开发工具。在处理逻辑清楚的情况下实现界面是非常方便和高效的，图 3 是用户登录界面。

有了完善的数据库表结构，操作界面的实现就很简单了：分别提供两个界面完成对表 operate_user 和 Application_menu 的操作就实现了操作模块和用户维护模块的功能，这两个功能都是单表操作没有什么可以解释的。需要解释的是 Application_menu 表中的 object 列，该列中存储的是实际已开发完成的 powerbuilder 模块，而且必须是窗口对象名称，因此对这列的维护必须与开发人员联系。该列的内容如果为空或是不正确，那在系统运行时就会出现系统错误。为了避免这种不友好的界面出现，我们的做法一般是：先建立一个空窗口（w_under_construction）对象，并把该对象的名字作为该列内容的缺省值。这样做一方面避免了错误的发生，另一方面也可以为后续开发提供接口。

用户权限的分配是通过对表 operate_item 的操作来实现的。由于该表的内容都来自上面两表，因此主要提供托拽的操作方式直观的来实现。

处理逻辑的实现主要分为两个部分：用户验证和操作模块执行。

（1）用户验证：该部分主要发生在用户登录窗口的确认事件中。根据输入的用户名/密码，检查合法性并保存一个全局 uid，作为该用户在此次操作中的唯一标识；根据权限的分配情况为该用户显示一个包含可操作的所有模块的工作台面，完成初始化工作。

（2）操作模块执行：该部分主要发生在用户选定了工作台面上的一个具体模块。根据选定的模块名称（object 列的内容），调用 powerbuilder 打开窗口的功能函数：

opensheet(w_sheet, ls_window, parentwindow(), 99, LAYERED!)object 列中的内容

将该模块的操作窗口显示为工作台面的一个子窗口，这种 MDI（多文档界面）模式允许用户同时操作多个不同功能，甚至是同一功能的多个窗口。

（下转第 21 页）

(上接第 56 页)

4 总结

本文介绍了一种多用户机制系统的 powerbuilder 实现,从数据库的设计到整体框架的建立都为这一机制做了考虑。该方法支持不同的用户在网络上同时登入系统,也允许使用相同的用户帐号在不同的机器上登录。这种实现方法独立于业务流程也独立于所使用的数据库系统。因此,具有一定的通用性。