基于 PowerBuilder 的数据库连接参数加密实现①

Realization of Encrypting of the Database Connection Parameters Based on PowerBuilder

赵学锋 张金隆 蔡淑琴 张 圆 (武汉 华中科技大学管理学院 430074)

摘要:分析了几种常用的对数据库连接参数赋值的几种方法,比较了它们的优、缺点,介绍了在 PowerBuilder 中如何利用生成的动态链接库(DLL)来实现对数据库连接参数加密、解密的基本原理,并结合实例给出了实现方法。

关键词:PowerBuilder 数据库连接参数 动态链接库 加密 解密

1 引言

大型数据库前端开发工具 PowerBuilder(以下简称 PB),是目前非常流行的一种开发工具,它以其高效率、可视化、支持面向对象技术、标准的 Windows 界面等卓越性能,为广大数据库应用软件开发者所青睐[1]。

不管是在 PB 环境下操作数据库,还是在应用程序中使用数据库,都必须建立与数据库的连接。这种连接是建立在驱动程序(数据库接口)之上的。PB 对常用的大型数据库管理系统(如 Oracle、Sybase、Informix、SQL Server、DB2 等)提供了专用的数据库接口,而对一些小型数据库管理系统(如 xBase、Access 等)和其他数据源(如 Excel)则提供了 ODBC 接口。在使用这些接口之前,必须对数据库的一些连接参数赋值,比如数据库名称、用户名、密码等[2][3]。这些参数在某些情况下对用户来说,是一些不能公开的信息,如何对这些数据库的连接参数进行加密解密,同时又不影响应用程序的应用,是用 PB 开发数据库应用程序要考虑的问题。本文通过分析几种常用的对数据库连接参数赋值的几种方法,比较了它们的优、缺点,介绍了在 PowerBuilder 中如何利用生成的动态链接库(DLL)来实现对数据库连接参数加密、解密的基本原理,并结合实例给出了实现方法。

2 常用的数据库连接参数赋值的比较

在 PB 中,实现和数据库的连接都是通过默认的全局事务对象变量 SQLCA(SQL Communications Area)来实现的,对数据库连接参数的赋值实际上就是对全局事务对象变量 SQLCA 的赋值。事务对象变量 SQLCA 有 15 个属性,其中有 10 个属性用于数据库的连接,5 个属性用于接收数据库返回的操作状态信息(成功或失败)。关于具体的事务对象的属性,可以参考相关的资料[1][2][3]。下面分析比较几种常用的对数据库连接参数赋值的几种方法。

2.1 从数据库配置文件中(Profile)复制

在 PB 的开发环境中,单击 PowerBar 上的 DB Profile 工具按钮,选择 preview 选项卡,可以看到连接数据库的参数设置语句,可以将这些语句方便地拷贝和粘贴到相应的程序中,实现和数据库的快速连接,其语句如下所示:

```
// Profile lgpj //数据库配置文件名
SQLCA.DBMS = "MSS Microsoft SQL Server 6.x" //DBMS 标识符
SQLCA.Database = "lgpj" //数据库名称
SQLCA.LogPass = "* * * * * *" //注册到数据库服务器的口令
SQLCA.ServerName = "mitzxf" //数据库驻留的服务器名
SQLCA.LogId = "sa" //注册到数据库服务器的用户名
SQLCA.AutoCommit = False //自动事务提交
SQLCA.DBParm = "" //ODBC 数据源连接数据库的参数
```

这种方法最简单,对于初学者和在单机上开发小型应用程序的用户是合适的。但是如果把应用放到其他的机器上或者数据库环境发生一点变化,比如数据库服务器名、用户名称等发生变化的话,要重新到 PB 的环境中去修改源程序,这将给开发者带来很大的不便。

2.2 将数据库连接参数写进一个外部初始化文件中

为了避免修改源程序,将数据库的连接参数写到一个外部的初始化文件中,然后通过 PB 提供的函数来读取文件中的参数。这样,即使数据库的环境发生了变化,也不需修改源程序,只需要修改初始化文件就够了,这也是目前绝大多数人采用的一种方法[3]。假设外部的初始化文件为 lg.ini,其文件的形式如下:

```
[Database] //DATABASE 标识,便于 profilestring 函数读取参数
DBMS=MSS Microsoft SQL Server 6.x //DBMS 标识符
```

① 国家自然科学基金资助项目(70271030)

Database=lgpj //数据库名称
LogPass=lu cy //注册到数据库服务器的口令
ServerName=mitzxf //数据库驻留的服务器名
LogId=sa //注册到数据库服务器的用户名
DbParm= //ODBC 数据源连接数据库的参数

把初始化文件定义好以后,就可以用 PB 提供的系统函数 Profilestring()来读取参数了,其程序如下:

```
……
SQLCA.DBMS = ProfileString ("lg.ini", "Database ", "DBMS ","")
SQLCA.Database = ProfileString ("lg.ini", "Database ", "Database ","")
SQLCA.LogId = ProfileString ("lg.ini", "Database ", "LogId ","")
SQLCA.LogPass = ProfileString ("lg.ini", "Database ", "LogPass", "")
SQLCA.ServerName = ProfileString ("lg.ini", "Database ", "ServerName ", "")
SQLCA.DbParm = ProfileString ("lg.ini", "Database ", "DbParm ", "")
……
```

上述方法可以在数据库环境发生变化的情况下,避免修改源程序,相对于第一种方法已经有了很大的改进。但是初始化文件仍然可以被打开,一些数据库参数依然可以被非法获取,这种状况对于一些大型的应用系统和保密性要求较高的程序来说是要尽量避免的,此时要考虑对数据库的参数进行加密,防止非法盗用的情况发生。

3 利用动态链接库实现数据库连接参数的加密解密

3.1 数据加密的基本原理和动态链接库 PBDLL 的功能

为了能对数据库的连接参数进行加密,首先要确定加密体制和算法。经过比较分析,决定采用公开密钥密码体制。公开密钥密码体制是指加密密钥(即公开密钥)PK 是公开信息,加密算法 E 和解密算法 D 也都是公开的,而解密密钥(即秘密密钥)SK 是保密的。在公开密钥密码体制中,应用最多的是 RSA 体制,RSA 算法是由 Rivest,shamir 和 Adleman 于 1978 年提出的,曾被 ISO/TC97 的数据加密技术委员会 SC20 推荐为公开密钥数据加密标准。关于 RSA 体制的基本原理可参考资料[4][5]。

笔者根据 RSA 体制的基本原理,在 VC 环境中生成了名字为 PBDLL 的动态链接库(DLL),其主要功能是把要加密的字符串(明文)利用加密密钥转化为二进制流文件(密文),解密时利用解密密钥将密文还原成明文。限于篇幅,不详细介绍生成过程和具体的程序实现,这里主要介绍用于加密和解密的两个函数的功能,RSAEncrypt(char * P)和 RSADecrypt()。

(1) 加密函数 RSAEncrypt(char * P)

此函数是完成使用 RSA 算法对字符串加密的功能,并将密文存入一个指定的文本文件。

入口参数:s_en_str ,表示要加密的密文

返回值:成功则返回 1,否则返回 0

(2) 解密函数 RSADecrypt()

此函数完成从指定的加密文件取出密文的功能。

入口参数:无

返回:解密的字符串

3.2 数据库连接参数加密解密的过程

数据库连接参数加密解密的过程可分如下几个步骤:

(1) 首先获取数据库的连接参数,经过一定的变换后,形成明文(待加密的字符串);

(2) 运行加密程序,生成密文(本例中是生成一个加密文件);

(3) 运行解密程序,从加密文件中获得明文;

(4) 经过变换,将数据库连接参数还原;

其过程如图 1 所示,以下将详细介绍其实现过程。

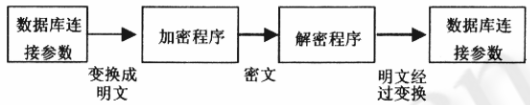

图 1　数据库连接参数加密解密示意图

3.3 数据库连接参数的变换和加密

建立如图 2 所示的窗口,在窗口上分别放置四个单行编辑框控件(如果需要输入更多的参数,可以放置更多),其名称分别为 sle_1、sle_2、sle_3、sle_4,分别输入数据库服务器名、数据库名、注册到数据库服务器的用户名和口令。放置两个命令按钮,其名称分别为 cb_encrypt 和 cb_cancel,分别用来执行加密和关闭窗口的功能。

图 2　对数据库连接参数进行加密的窗口

(1) 定义一个全局外部函数(Global External Function)RSAEncrypt,其定义格式如下:

```
function int RSAEncrypt( char P[]) Library "pb-
```

dll.dll" //利用 pbdll.dll 定义全局外部函数 //利用私钥加密字符串

(2)对 cb_encrypt 按钮的 clicked 事件编写脚本

cb_encrypt 的功能主要是对数据库的连接参数进行组合和变换,变成要加密的明文,然后调用全局外部函数和动态链接库(DLL)将明文经过加密后生成一个密文。为了便于区分每个参数,在参数之间用'&'分隔,其脚本如下:

```
string ls_code
int li_tt
char ch_a[300] //定义一个字符数组,作为函数 RSAEncrypt 的传入参数
ls_code=trim(sle_1.text)+&+trim(sle_2.text)+&+& //将四个参数通过'&'连接成一个
    trim(sle_3.text)+&+trim(sle_4.text) //经过变换的字符串
ch_a = ls_code
setpointer(HourGlass!) //将鼠标的形状变为漏斗状
li_tt = RSAEncrypt(ch_a) //调用定义的全局外部函数 RSAEncrypt(char P[])
if li_tt=1 then //根据函数的返回值判断加密是否成功
  messagebox('','新的密码文件已生成!')
  setpointer(arrow!)
else
  messagebox('','数据加密出错!')
  return
end if
```

假设在窗口中输入的四个参数分别为 mitzxf、lgpj、sa、lu cy,经过变换后的字符串为'mitzxf&lgpj&sa&lu cy',经过加密后形成的加密文件如图 3 所示:

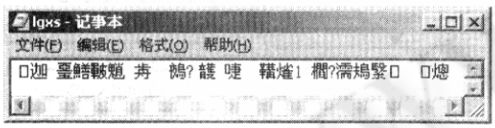

图 3　经过加密后的密文示意图

3.4 数据库连接参数的解密和还原

数据库连接参数解密的窗口比较简单,在窗口上放置两个按钮,一个'解密'按钮,另一个'退出'按钮。这里主要介绍'解密'按钮的 clicked 事件的脚本,如下所示:

```
string ls_desstr, ls_str[4] // 定义一个字符串数组,存放经过还原的数据库连接参数
int ii,i
ls_desstr = RSADecrypt() //调用定义的全局外部函数 RSADecrypt(),解密返回明文
//数据库连接参数的还原
for i=1 to 3
  ii=pos(ls_desstr,&,1) //查找分隔符'&'的位置
  if ii>0 then
    ls_str[i] = Left(ls_desstr, ii - 1) //取出用分隔符分隔的数据库连接参数
    ls_desstr = Mid(ls_desstr, ii + 1) //将取出的数据库连接参数从字符串中去除
  end if
next
ls_str[4]=ls_desstr //获取最后一个数据库连接参数
//利用取出的数据库连接参数对事务对象变量 SQLCA 赋值
SQLCA.ServerName =ls_str[1]
SQLCA.Database =ls_str[2]
SQLCA.LogId = ls_str[3]
SQLCA.LogPass = ls_str[4]
……
```

至此,已完成了数据库连接参数的变换、加密、解密、还原等过程。在用户使用过程中,先使用加密程序对数据库连接参数加密,然后再运行解密程序对加密的密文进行解密,最后再还原出原始的数据库连接参数,这样可以较好地维护数据库连接参数的保密性和安全性。

4 小结

本文首先分析了几种常用的对数据库连接参数赋值的几种方法,比较了它们的优、缺点,介绍了在 PowerBuilder 中如何利用生成的动态链接库(DLL)来实现对数据库连接参数加密、解密的基本原理,并通过全局外部函数调用动态链接库(DLL)来实现对参数的加密和解密,最后结合实例给出了实现方法。通过这种方法,在数据库环境发生了变换后,可以动态改变数据库连接参数,而不用改动源程序,同时较好地保证了数据库连接参数的安全性。

参考文献

1 崔巍,PowerBuilder 8.0 数据库应用系统开发教程[M],清华大学出版社,2002。
2 崔巍,PowerBuilder 8.0 高级应用技术[M],清华大学出版社,2002。
3 刘红岩,PowerBuilder 原理与应用指南[M],北京:电子工业出版社,1999。
4 黄叔武、杨一平等,计算机网络工程教程[M],清华大学出版社,1999。
5 谢希仁、陈鸣等,计算机网络[M],电子工业出版社,1994。