数,这些自动对象是在进入 try 块时构造的。很遗憾的是 C++ 中长期存在的构造函数中的异常处理在 Borland C++4.0 中也没有得到很好的解决,这主要的是因为构造函数不返回值,而且也没有直接的方法向调用者报告失败。如果异常在构造某个对象时被抛出了,只有已完全构造好的对象才能调用析构函数。例如,当一个对象数组正在构造时异常抛出了,只有那些已完全构造好了的数组元素才能调用析构函数。这样如果在构造函数中申请了资源而又抛出了异常,所申请的这部分资源就悬挂起来了。解决这个问题目前还没有最好的方案,值得提倡的是使用函数闭包来处理。所谓闭包是借用数学中的一个概念,这里其含义是指一个代码域,它的边界是由一个前文和一个后文组成的,前文和后文必须同时存在,否则是一个运行时错误,例如:

```
hDC=GetDC(hWnd);
TextOut(hDC,x,y,string,size);
ReleaseDC(hWnd,hDC);
```

我们可以把它看成一个整体,成为一个闭包。

要较好解决前面提到的资源悬挂问题,我们建立一个单独的类,该类围绕资源建立一个函数闭包,类的构造函数相当于前文,析构函数相当于后文。当然无论异常情况是否被抛出,都得编一个小类,虽然增加了代码量,但是却不浪费资源是值得的。我们用一个小例子来说明这一方法。

```
#include<windows.h>
class DC{
    HDC hDC;
  public:
    DC(){hDC=GetDC(NULL);}
    operator HDC(){return hDC;}
    ~DC(){ReleaseDC(NULL,hDC);}
};
```

使用这个类就不必担心异常抛出时设备上下文挂起,需要设备上下文的函数可以如下使用 DC 类:

```
void MyFunc();
    {
      DC MyDc;
      //call the GDI function on the DC;
      //...
    }
```

当在设备上下文上处理 GDI 操作时,如有异常抛出,那么 MyFunc()将被退出,但是在退出前程序将调用 DC 的析构函数释放设备上下文。

函数闭包可以说是处理异常情况中资源问题的一种很妙的方法,我们提倡把所有的资源都用如此的方法封闭起来,以免出现内存漏洞,死包围,和死锁定的代码。在处理动态内存分配时这显得更为重要。

五、结束语

由于程序利用异常处理能以有秩序,有组织及一致的方法截取及处理异常情况,所以异常处理将成为 C++4.0 语言的下一个热点,而 Borland C++4.0 处理异常的机制总的来说是近乎完美的,它必将成为广大程序员的好帮手。

参考文献:

[1]C++异常处理<<微型计算机>>1994,3,

[2]Borland C++4.0 程序设计 成都科技大学出版社

[3]C++高级编程技术 电子工业出版社

将 FoxBASE+的数据转换为 WATCOM SQL 的数据

庞建民　宁　钰　(解放军信息工程学院)

摘要:本文给出了一种将 *FoxBASE+*中的数据转化为 *WATCOM SQL* 中数据的实现途径。本文中的程序采用 *Powerscript* 语言编写。

一、引 言

Powerbuilder 是目前最流行的数据库前台开发工具,它支持 Sybase、Oracle、Informix、Ingress、DB2 等多种数据库管理系统。另外,它提供的内置数据库管理系统 WATCOM SQL,在没有后台数据库管理系统的情况下,也可用来开发大型应用。由于我国用 dBASE、FoxBASE+等数据库管理系统开发的软件较多,因此无论从系统的互连还是系统的更新换代来讲,都可能需要将 FoxBASE+的数据转换成 WATCOM SQL 中的数据,为此我们用 Powerscrip 语言编写了一个程序,它能完成将 FoxBASE+中的数据转换成 WATCOM SQL 中的数据。下面我们以分析 FoxBASE+中数据存放格式开始,介绍我们的实现途径。

二、FoxBASE+的数据库文件 (.DBF 文件)的结构

FoxBASE+中的数据以数据库文件为单位存放,有时

也涉及数据库备注文件。所以要想将 FoxBASE+中的数据转换为 WATCOM SQL 中的数据,必须先把握数据库文件(.DBF 文件),数据库备注文件(.DBF 文件)的结构,只有充分利用了这两个文件中的信息,才能正确地完成转换任务。

FoxBASE+的数据库文件由文件描述信息、字段描述信息、数据记录三部分信息构成。其中文件描述信息在数据库文件的开始处,占 32 个字节,它记录了文件所含记录数、记录的长度、有无备注字段等信息;字段描述信息由创建数据库文件时字段的定义信息构成,每个字段 32 位,在字段描述信息之后紧跟着的一个字节 ODH 是文件头结束标记。字段描述信息记录了字段名、类型、宽度等信息;数据记录在文件头结束标记之后,若记录未删除,则数据记录的第一个字节为空格(20H),若记录被作了删除标记,则数据记录的第一个字节为星号(2AH),数据字段按其定义时的顺序依次排列,没有字段间隔符,也没有记录终止符,在所有记录之后是数据库文件结束标记(1AH)即 EOF。

三、数据库备注文件(.DBF 文件)的结构

FoxBASE+的备注(Memo)字段在。DBF 文件的记录中占十个字节的长度,在这十个字节中存放了一个指针,这个指针是备注文件(.DBF 文件)中与此记录相关的备注字段的具体内容的开始地址。

数据库备注文件是与数据类型中有 Memo 字段时,由 FoxBASE+自动生成的,在数据库备注文件的第一个扇区中存放的信息为该备注文件占用的扇区数,每一个备注字段内容在备注文件中都以扇区(512 字节)为最小存放单位,结束标记为 0A1A,对于删除的 Moem 字段,仍然在数据库备注文件中占用空间,但第一个字节信息变为 1A。

四、WATCOM SQL 中的数据存放格式

在 WATCOM SQL 中,数据是以数据库为单位以.DB 文件形式存放的,在一个数据库中可有多个表,每个表对应 FoxBASE+中的一个数据库文件,可见 WATCOM SQL 中数据的存放与 FoxBASE+中数据的存放大不相同。

五、具体实现

本程序用 Powerbuilder 中的 Powerscript 语言来实现,大致流程为:先从编辑窗口中读入用户输入的.DBF 文件名,然后打开此文件,读取文件描述信息,在此先判断是否有 Memo 字段,有则打开相应的.DBT 文件,然后读出文件所含记录个数,文件头的长度,每个记录的长度,再调用用户自定义函数 readascstrnew,把各个字段信息读入到一个二维数组 pr 中,再读入用户输入的表名,形成创立表所需的串,再动态地创立表,把从 FoxBASE+的.DBF 文件中读出的数据存入刚创立的表中,从而实现将 FoxBASE+中的数据转换为 WATCOM SQL 中的数据。具体程序见下页。

六、结束语

本程序在 Intel 302 和 Compaq 486 机器上测试通过,只要输入要转换的 FoxBASE+中的.DBF 文件名和 WATCOM SQL 中的表名,数据就会快速地转换。由于 dBASE、FoxPRO 等数据库管理系统中数据存放格式与 BASE+相同,因此该程序同样适用于 dBASE、FoxPRO 中的数据到 WATCOM SQL 中数据的转换。

主程序:

```
Script:clicked for cb 1
int numb,fhdbt
decimal deci
date dat
strins ch1,fieldnam
string pr1,pr2,str
char       ch
string chx
int a,z,i,j,loops
long flen,bytes read,new pos
blob tot_b,b,tot1_b
fhfox = File()pen(sle_1.text,streammode1,read!)
IF fhfox = -1 THEN
  MessageBox("ERROR!","filefox can not open!")
  goto err
END IF
SetPointer(HourGlass!)
flen = Filelength(sle_1.text)
/ / Determine how many FileRead calls are needed to read the
/ / entire file.
IF flen > 32765 THEN
   IF Mod(flen,32765) = 0 THEN
     loops = flen / 32765
   ELSE
     loops = flen / 32765
   END IF
   ELSE
   loops = 1
END IF
FOR i = 1 to loops
   bytes read = FileRead(fhfox,b)
```

```
    tot b = tot_b+b
    new pos = new pos+bytes read
    FileSeek (fhfox,new pos,FromBeginning!)
NEXT
x blob = tot_b
FileClose(fhfox)
filename = sle_1.text
/ * * * * * * * * * * * * * * * * * * * * * * *
BEGIN READ ROXBASE DBT FILE HEAD MESSAGE
  FIRST DECIDE IF THERE IS A FILE WHICH HAS MEMO FIELD
 * * * * * * * * * * * * * * * * * * * * * * *
xx = BlobMid(x blob,1,2)
a = integer(xx)
a = mod(a,256)
 IF a< >3 THEN
 a = Len(filename)
 dbtname = Replace(filename,a,i,"t")
 memo = 't'
 fhdbt = FileOpen(dbtname,streammode,read!)
 IF fhdbt = -1 THEN
    MessageBox("ERROR!","filedbt can not open!)
    goto err
END IF
/ / Determine how many FileRead calls are needed to read the
/ / entire file.
flen = Filelength(dbtname)
    IF Mod = (flen,32765) = 0 THEN
        loops = flen / 32765
    ELSE
        loops = (flen / 32765)+1
    END IF
    ELSE     loops = 1
END IF
FOR i = I to loops
    bytes read = FileRead(fhdbt,b)
    tot1 b = tot1_b+b
    new pos = new pos+bytes read
    FileSeek(fhdbt,new,pos,FromBeginning!)
NEXT
y blob = tot1_b
i = len(y_blob)
 fileclose(fhdbt)
 IF bytes read = -1 THEN
    MessageBox("ERROR!","filedbt can not be readed into y blob!")
     goto err
END IF
dbtlong = flen
 xx = BlobMid(y_blob,1,2)
 a = integer(xx)
 sectornum = Mod(a,256) / / sectornum is the number of dbtfile
END IF
/ * * * * * * * * * * * * * * * * * *
READ NUMBER OF RECORD,LENGTH OF FILEHEAD, AND LENGTH OF RECORD
 * * * * * * * * * * * * * * * * * * * *
xx = BlobMid(x blob,5,4)
recnum = integer(xx) / / recnum is the number of record
xx = BlobMid(x blob,9,2)
bytnum = integer(xx) / / bytnum is number of filehead
xx = BlobMid(x blob,11,2)
reclong = integer(xx) / / reclong is length of record
/ * * * * * * * * * * * * * * * * * *
BEGIN READ MESSAGE ABORT FIELD
 * * * * * * * * * * * * * * * * * * *
num = (bytnum-1) / 32-1 / / num is number of field
  startpoint = 32
i = 1
Do while i< = num
    pr[i,1] = readascstrnew (startpoint,11)
    pr[i,2] = readascstrnew(startpoint+11,1)
    pr[i,3] = string(asc(readascstrnew (startpoint+12,4)))
    pr[i,4] = string (asc(readascstrnew(startpoint+16,1)))
    pr[i,5] = string (asc(readascstrnew(startpoint+17,1)))
 startpoint+ = 32
 i++
loop
/ * * * * * * * * * * * * * * * * * *
CREATE A TABLE
 * * * * * * * * * * * * * * * * * * *
string t
tbname = sle table name.text
t = "create table "+tbname+"("
i = 1
do while i< = num
  t = t+pr[i,1]
  CHOOSE CASE pr[i,2]
   CASE "C"
     t = t+"char("+pr[i,4]+")"
   CASE "N"
      if pr[i,5] = "0" then
        t = t+"integer"
      else
        t = t+"decimal("+pr[i,4]+","+pr[i,5]+")"
      end if
   CASE "D"
     t = t+"date"
   CASE "L"
        t = t+"char(1)"
    CASE"M"
        t = t+"char(32765)"
   END CHOOSE
   if i< > num then
     t = t+","
   end if
  i++
```

```
loop
t=t+")"
if sqlca.sqlcode=-1 then
  messagebox("create table error!",sqlca.sqlerrtext,exclamation!)
  goto err
end if
```

FoxPro 数据库文件备份与恢复程序的设计

陈学中 (山东建材学院)

一、问题的提出

数据备份与恢复功能模块设置与否以及设计是否合理,将直接关系到一个管理信息系统(MIS)的安全可靠性和数据的可维护性。目前,国内运行的 MIS 很大部分是用 FoxPro 系统设计的,而 FoxPro 并没有提供数据库备份与恢复的直接功能。DOS 操作系统虽然提供了备份 BACKUP 和恢复 RESTORE 命令,但由于 DOS 版本不同,其备份与恢复命令的差异较大,不宜于纳入基于 FoxPro 开发的系统。为此,我们开发 MIS 时,设计了 FoxPro 数据库文件备份与恢复的程序。

二、程序设计思想

1.备份数据库文件的选择

在一个 MIS 中,并不是所有数据库文件都需要备份。为了能够有选择地备份文件,需设计一数据库文件 SJBFK.DAT,其结构如下:

字段名称	类型	长度	小数	字段描述
NAME	字符	8		备份数据库文件名
EXT	字符	3		备份文件扩展名
FILE_REC	数字	8		备份文件记录总数
SL	数字	10		备份文件长度(字节)
DESCRIBE	字符	16		备份文件内容简单描述
BZ	字符	1		是否备份标志

对于要备份的某个文件,需要将其对应的字段 BZ 设置为"*"。

2.选择盘别并检测软盘

将数据从硬盘备份到软盘或从软盘恢复到硬盘时,软盘既可以是 A 盘也可以是 B 盘。由于 FoxPro 系统没有提供对软盘驱动器中软盘状态的检测功能,这往往会破坏优美的屏幕画面,本文采用汇编语言模块实现对软盘状态的检测(请参阅《计算机系统应用》1995.5 第 48～50 页"FoxPro 下对软盘驱动器软盘状态的检测")。

3.备份盘的识别特征

用本文的备份程序所备份软盘的特征是,在软盘的根目录下有一特征文件和一个名为 BACKUP 的子目录。

(1)特征文件。将备份日期转换为字符型数据做为文件名,再加上字符型备份盘顺序号做为扩展名,生成备份盘的识别文件名。例如,1995 年 8 月 1 日备份的磁盘,其特征文件分别是:19950801.001(第一张备份盘)、19950801.002(第二张备份盘)、……特征文件的内容是本张软盘所备份的数据库文件名称、长度及其是否结束的标识符,其结构如下:

字段名称	类型	长度	小数	字段描述
NAME	字符	8		备份数据库文件名
EXT	字符	3		备份文件扩展名
FILE_LEN	数字	10		备份文件长度(字节)
FILE_END	数字	1		备份文件结束标志

字段 FILE_END 等于 0,表示该文件已全部备份到本张软盘上。

字段 FILE_END 等于 1,表示该文件尚未在本张软盘上备份完毕,尚需在下一张软盘上继续备份。

字段 FILE-[-]ND 等于 9,表示上张软盘没有备份完的文件已经在本张软盘上备份完毕。

当字段 NAME 为"En_flag"且字段 EXT 为"End"时,表示本次备份工作全部完成。

(2)子目录 BACKUP。考虑到软盘根目录所能包含文件个数的有限性,建立 BACKUP 子目录,并将备份文件放到该子目录下。子目录的建立,采用汇编语言模块 MRCD。BIN(将 MRCD.ASM 编译连接转换而成)。

4.数据库(DBF 类)文件分析

(1)FoxPro 数据库文件结构及其长度计算。DBF 文件结构,由文件头、字段表和数据区组成。

文件头长度 32(字节)

字段表长度=字段个数×32+2(字节)

数据区长度=记录个数×单个记录长度(字节)

DBF 文件长度=文件头长度+字段表长度+数据区长度(字节)

(2)FoxPro 数据库文件分类。相对软盘来说,FoxPro 数据库文件可以按其长度分为大于软盘容量的文件和小