```
virtual CString GetDefaultSQL( );
```

Open 函数在必要时会调用该函数，返回缺省的 SQL 语句或表名以查询数据源中的记录。一般需要在 CRecordset 派生类中覆盖该函数，并在新版的函数中提供 SQL 语句或表名。下面是一些返回字符串的例子。

```
"Clerk"              //选择 Clerk 表中的所有记录到记录集中
"Clerk, Borrow"      //合并 Clerk 表和 Borrow 表的各列到记录集中
```

上面的例子说明，通过合理地安排 SQL 语句和表名，Open 函数可以十分灵活地查询数据源中的记录。用户可以合并多个表的字段，也可以只选择记录中的某些字段，还可以对记录进行过滤和排序。

在建立记录集时，CRecordset 会构造一个 SELECT 语句来查询数据源。如果在调用 Open 时只提供了表名，那么 SELECT 语句还缺少选择列参数 rfx-field-list。框架规定，如果只提供了表名，则选择列的信息从 DoFieldExchange 中的 RFX 语句里提取。例如，如果在调用 Open 时只提供了“Clerk”表名，那么将会构造如下一个 SELECT 语句：

```
SELECT Clerk_ID, Name, Password, ID_CARD, Post  FROM Clerk
```

建立记录集后，用户可以随时调用 Requery 成员函数来重新查询和建立记录集。Requery 有两个重要用途：

- 使记录集能反映用户对数据源的改变。
- 按照新的过滤或排序方法查询记录并重新建立记录集。

在调用 Requery 之前，可调用 CanRestart 来判断记录集是否支持 Requery 操作。要记住 Requery 只能在成功调用 Open 后调用，所以程序应调用 IsOpen 来判断记录集是否已建立，函数的声明为：

```
virtual BOOL Requery( );throw( CDBException, CMemoryException );
```

返回 TRUE 表明记录集建立成功，否则返回 FALSE，若函数内部出错则产生异常。

```
BOOL CanRestart( ) const;        //若支持 Requery 则返回 TRUE
BOOL IsOpen( ) const;            //若记录集已建立则返回 TRUE
```

CRecordset 类有两个公共数据成员 m_strFilter 和 m_strSort，用来设置对记录的过滤和排序。在调用 Open 或 Requery 前，如果在这两个数据成员中指定了过滤或排序，那么 Open 和 Requery 将按这两个数据成员指定的过滤和排序来查询数据源。

成员 m_strFilter 用于指定过滤器，m_strFilter 实际上包含了 SQL 的 WHERE 子句的内容，但它不含 WHERE 关键字。使用 m_strFilter 的一个例子为：

```
m_pSet->m_strFilter="Clerk_ID='11111111'";  // 只选择 Clerk_ID
为'11111111'的记录
if(m_pSet->Open(CRecordset::snapshot, "Clerk"))
. . . . . .
```

成员 m_strSort 用于指定排序，m_strSort 实际上包含了 ORDER BY 子句的内容，但它不含 ORDER BY 关键字。m_strSort 的一个例子为：

```
m_pSet->m_strSort="Clerk_ID DESC"; //按 CourseID 的降序排列记录
m_pSet->Open();
```

事实上，Open 函数在构造 SELECT 语句时，会把 m_strFilter 和 m_strSort 的内容放入 SELECT 语句的 WHERE 和 ORDER BY 子句中。如果在 Open 的 lpszSQL 参数中已包括了 WHERE 和 ORDER BY 子句，那么 m_strFilter 和 m_strSort 必须为空。

调用无参数成员函数 Close 可以关闭记录集。在调用了 Close 函数后，程序可以再次调用 Open 建立新的记录集。CRecordset 的析构函数会调用 Close 函数，所以当删除 CRecordset 对象时记录集也随之关闭。

6.5 登录模块的具体实现

通过上面对 CRecordset 和 CDatabase 的介绍，读者可能会感到特别抽象。下面我们介绍在登录模块使用记录集的具体操作方法。

图 6-9 “ODBC Microsoft Access 安装”对话框

首先，介绍 ODBC 数据源的建立。在控制面板上双击 ODBC 数据源图标，选择系统 DSN 选项，单击“添加”按钮。在数据源驱动程序列表框中选择 Microsoft Access Driver 选项，然后单击完成按钮。在出现的如图 6-9 所示的对话框中输入相应的信息。选择 library.mdb 数据库，单击“确定”按钮。即完成了数据源 library 的添加。

接着，右击登录模块对话框选择 ClassWizard 选项，在弹出的 ClassWizard 对话框中，单击 New Class 按钮。接着弹出如图 6-10 所示的对话框。按图所示输入记录集的名字和选择基类。

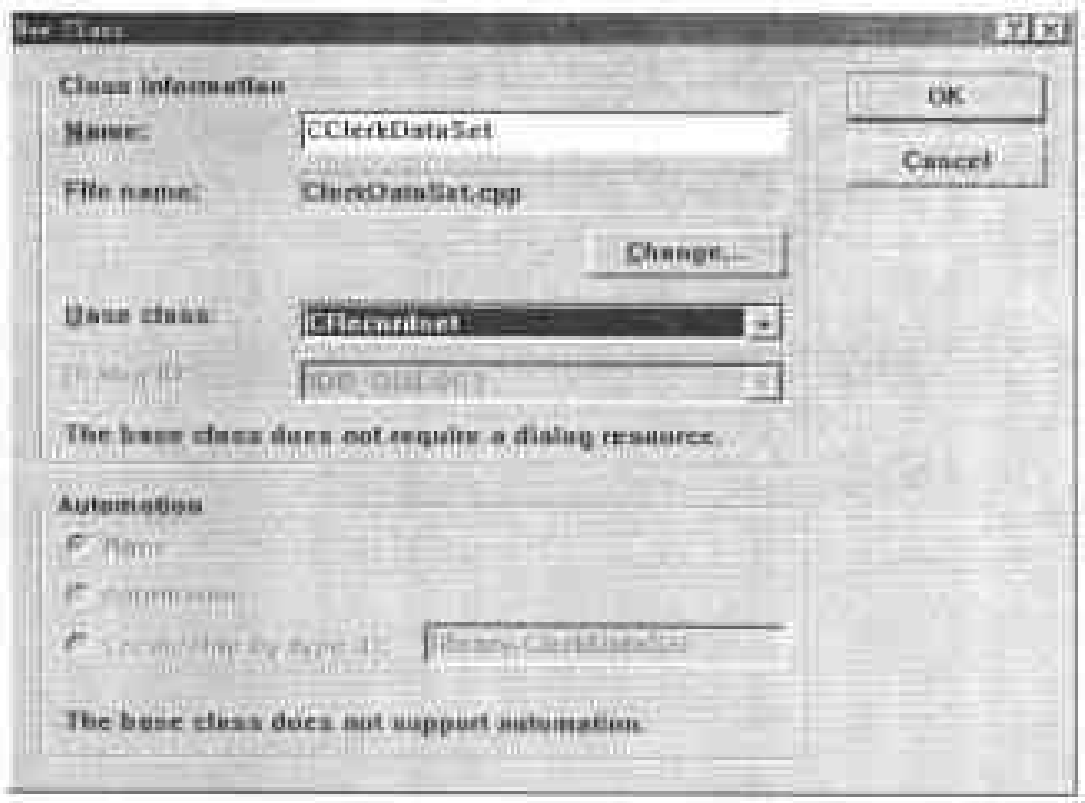

图 6-10　New Class 对话框

单击 OK 按钮则出现如图 6-11 所示的对话框。选择数据源 library ，单击 OK 按钮。出现图 6-12 所示的对话框。选择对应的表 CLERK ， 单击 OK 按钮，于是 CclerkDataSet 类的建立就算完成了。查看它的声明及 Dofieldexchange 函数，就能看到程序 6-5-1，6-5-2 的代码了，在此不再赘述。

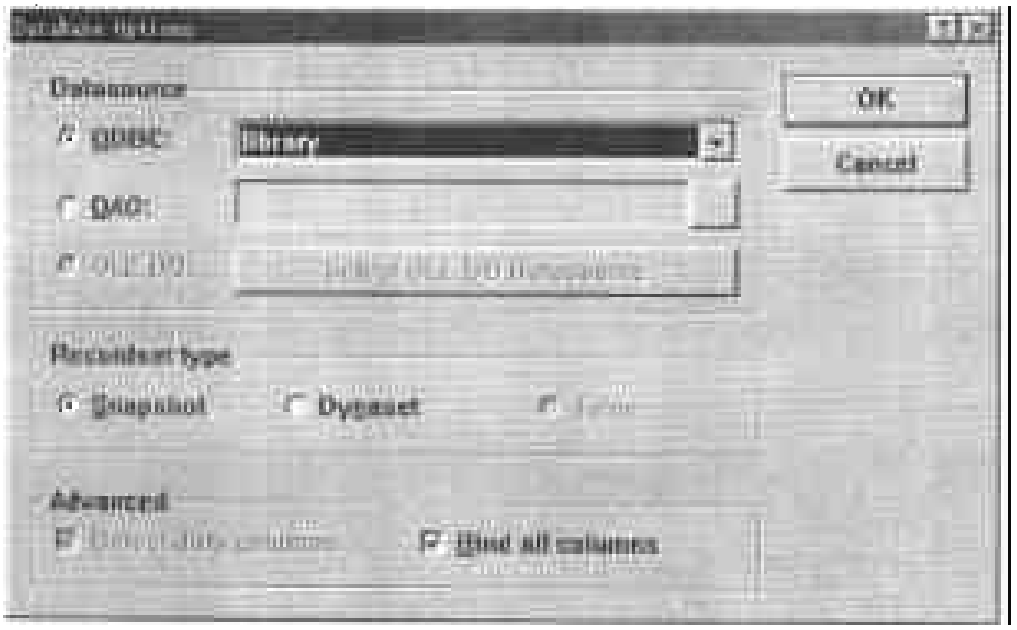

图 6-11 Database Options 对话框

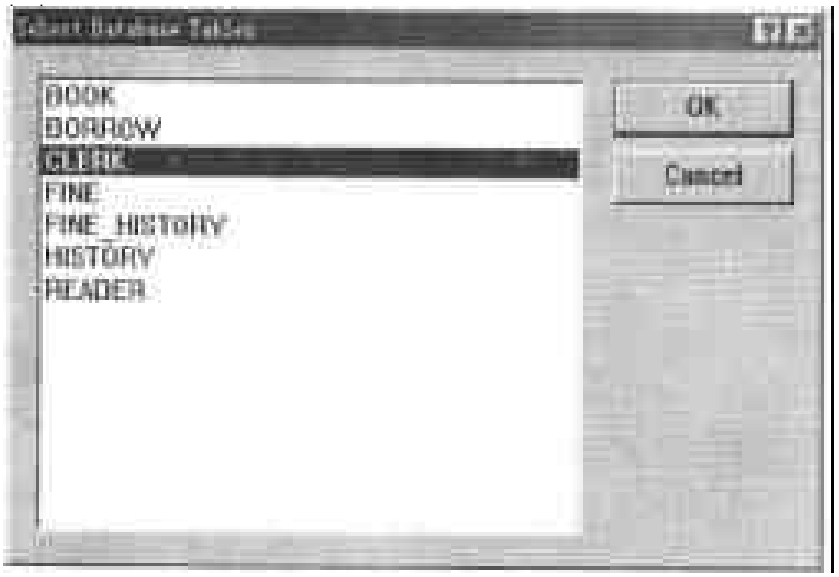

图 6-12 Select Database Tables 对话框

第四步编写登录对话框中登录按钮的脚本。右击对话框的“登录”按钮，选择 classwizard 选项。如图 6-13 所示。双击 Messsages 列表框的 BN_CLICKED，然后确定弹出的对话框。如果要编辑代码，则单击 EDIT CODE 按钮。参照程序 6-5-3。

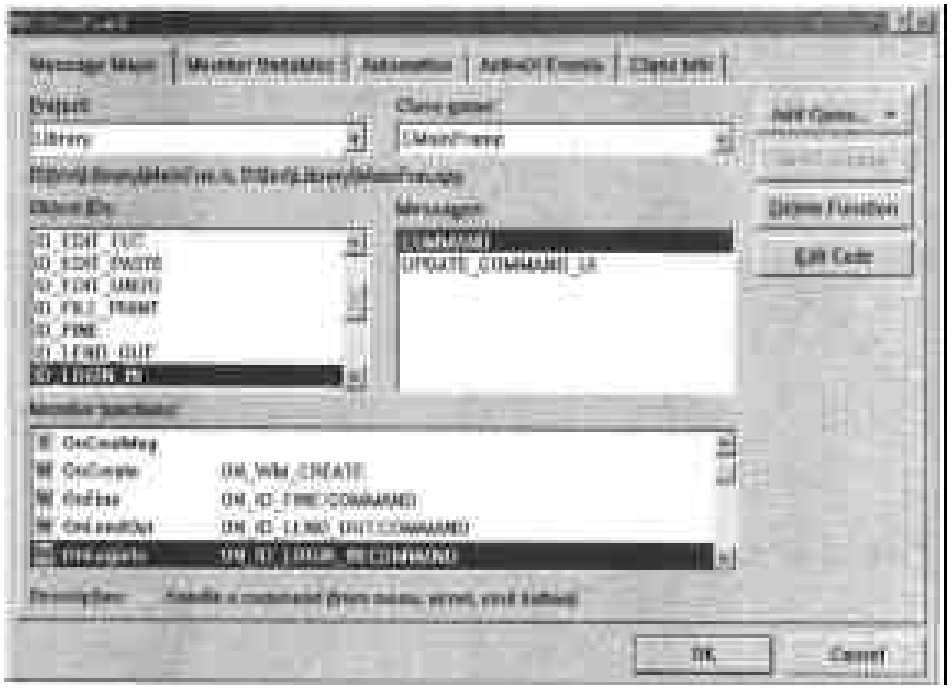

图 6-13　MFC ClassWizard 对话框

程序 6-5-3

```
void CLoginDlg::OnConfirm()
{
// TODO: Add your control notification handler code here
CClerkDataSet mrsDataSet;  /*声明记录集*/
CString mSqlStr;

UpdateData(TRUE);

if (m_strName.IsEmpty())    /*判断用户名信息是否为空*/
{
    AfxMessageBox("请输入用户名！");
    return;
}

mSqlStr = "SELECT * FROM CLERK WHERE NAME='";
```

```
mSqlStr = mSqlStr + m_strName;
mSqlStr = mSqlStr + "' AND PASSWORD='";
mSqlStr = mSqlStr + m_strPassword;
mSqlStr = mSqlStr + "'"; /*mSqlStr 存放 SQL 语句，该语句是查询 Clerk 表
中是否有用户名为 m_strName，并且密码为 m_strPassword 的记录 */
if (!mrsDataSet.Open(AFX_DB_USE_DEFAULT_TYPE, mSqlStr))
/*打开记录集，查询是否有符合条件的记录*/
{
    AfxMessageBox("CLERK 表打开失败！");
    return;
}
if (!mrsDataSet.IsEOF()) /* 如果记录集不为空，则说明有符合条件的记录，
                            说明登录合法*/
{
    // Open all function for user
    m_bSuccess = TRUE;
    m_strUSERID = mrsDataSet.m_CLERK_ID ;
    CDialog::OnOK();
}
else
{
    AfxMessageBox("登录失败！");
    m_strUSERID = _T("");
    return;
}
}
```

第五步，按第四步的方式，在菜单的“登录”选项的单击事件中编写程序 6-12 脚本。

程序 6-5-4：

```
void CMainFrame::OnLoginIn()
{
// TODO: Add your command handler code here
CLoginDlg mDlg;
if (mDlg.DoModal() == IDOK) /*判断用户是否通过了身份认证*/
{
    m_bLogin = TRUE;
    m_strUserName = mDlg.m_strName ;
    m_strUserID = mDlg.m_strUSERID;
}
else
{
    m_bLogin = FALSE;
    m_strUserName = _T("");
    m_strUserID = _T("");
}
CString Name;
Name = "当前用户：" + m_strUserName;
m_wndStatusBar.SetPaneText(0, Name);/*在状态栏中显示用户状态信息*/
}
```

至此登录模块介绍完毕。通过对登录模块实现的介绍，读者知道了 Visual C++ 通过 MFC ODBC 操作数据库的基本方式。

6.6　读者资料维护模块的实现

从这一节开始，介绍读者资料维护模块。在介绍之前，先补充一些 CRecordSet 的知识。

CRecordset 提供了几个成员函数用来在记录集中滚动，如下所示。当用这些

函数滚动到一个新记录时，框架会自动地把新记录的内容拷贝到域数据成员中。

```
void MoveNext( ); //前进一个记录
void MovePrev( ); //后退一个记录
void MoveFirst( ); //滚动到记录集中的第一个记录
void MoveLast( ); //滚动到记录集中的最后一个记录
```

void SetAbsolutePosition(long nRows); //该函数用于滚动到由参数 nRows 指定的绝对位置处。若 nRows 为负数，则从后往前滚动。例如，当 nRows 为－1 时，函数就滚动到记录集的末尾。注意，该函数不会跳过被删除的记录。

```
virtual void Move( long nRows, WORD wFetchType =SQL_FETCH_
RELATIVE );
```

该函数功能强大，通过将 wFetchType 参数指定为 SQL_FETCH_NEXT、SQL_FETCH_PRIOR 、 SQL_FETCH_FIRST 、 SQL_FETCH_LAST 和 SQL_FETCH_ABSOLUTE，可以完成上面五个函数的功能。若 wFetchType 为 SQL_FETCH_RELATIVE，那么将相对当前记录移动，若 nRows 为正数，则向前移动，若 nRows 为负数，则向后移动。

如果在建立记录集时选择了 CRecordset::skipDeletedRecords 选项，那么除了 SetAbsolutePosition 外，在滚动记录时将跳过被删除的记录，这一点对像 FoxPro 这样的数据库十分重要。如果记录集是空的，那么调用上述函数将产生异常，另外，必须保证滚动没有超出记录集的边界。调用 IsEOF 和 IsBOF 可以进行这方面的检测。

```
BOOL IsEOF( ) const;
```

如果记录集为空或滚动过了最后一个记录，那么函数返回 TRUE，否则返回 FALSE。

```
BOOL IsBOF( ) const;
```

如果记录集为空或滚动过了第一个记录，那么函数返回 TRUE，否则返回 FALSE。

下面是一个使用 IsEOF 的例子：

```
while(!m_pSet->IsEOF( ))
    m_pSet->MoveNext( );
```

调用 GetRecordCound 可获得记录集中的记录总数，该函数的声明为

```
long GetRecordCount( ) const;
```

要注意这个函数返回的实际上是用户在记录集中滚动的最远距离，要想真

正返回记录总数，只有调用 MoveNext 移动到记录集的末尾(MoveLast 不行)。

要修改当前记录，应该按下列步骤进行：调用 Edit 成员函数，调用该函数后就进入了编辑模式，程序可以修改域数据成员。注意不要在一个空的记录集中调用 Edit，否则会产生异常。Edit 函数会把当前域数据成员的内容保存在一个缓冲区中，这样做有两个目的，一是可以与域数据成员作比较以判断哪些字段被改变了，二是在必要的时候可以恢复域数据成员原来的值。若再次调用 Edit，则将从缓冲区中恢复域数据成员，调用后程序仍处于编辑模式。调用 Move(AFX_MOVE_REFRESH) 或 Move(0) 可退出编辑模式(AFX_MOVE_REFRESH 的值为 0)，同时该函数会从缓冲区中恢复域数据成员。

如果要设置域数据成员的新值，可调用 Update 完成编辑。Update 把变化后的记录写入数据源并结束编辑模式。

要向记录集中添加新的记录，应该按下列步骤进行：调用 AddNew 成员函数。调用该函数后就进入了添加模式，该函数把所有的域数据成员都设置成 NULL(注意，在数据库术语中，NULL 是指没有值，这与 C++的 NULL 是不同的)。与 Edit 一样，AddNew 会把当前域数据成员的内容保存在一个缓冲区中，在必要的时候，程序可以再次调用 AddNew 取消添加操作并恢复域数据成员原来的值，调用后程序仍处于添加模式。调用 Move(AFX_MOVE_REFRESH)可退出添加模式，同时该函数会从缓冲区中恢复域数据成员。

如果要设置域数据成员，可以调用 Update。Update 把域数据成员中的内容作为新记录写入数据源，从而结束了添加。

如果记录集是快照，那么在添加一个新的记录后，需要调用 Requery 重新查询，因为快照无法反映添加操作。

要删除记录集的当前记录，应按下面两步进行：

① 调用 Delete 成员函数，该函数会同时给记录集和数据源中当前记录加上删除标记，注意不要在一个空记录集中调用 Delete，否则会产生一个异常。

② 滚动到另一个记录上以跳过删除记录。

上面提到的函数声明为：

```
virtual void Edit( );throw( CDBException, CMemoryException );
virtual void AddNew( );throw( CDBException );
virtual void Delete( );throw( CDBException );
virtual BOOL Update( );throw( CDBException );
```

若更新失败则函数返回 FALSE，且会产生一个异常。

在对记录集进行更改以前，程序也许要调用下列函数来判断记录集是否是

可以更改的，因为如果在不能更改的记录集中进行修改、添加或删除将导致异常的产生。

```
BOOL CanUpdate( ) const; //返回 TRUE 表明记录是可以修改、添加和删除的。
BOOL CanAppend( ) const; //返回 TRUE 则表明可以添加记录
```

下面介绍读者资料维护模块的具体操作步骤：

第一步，生成对话框 IDD_READER_MAITAIN，按图 6-14 所示的方式放上相应的控件，同时生成 CreaderMDlg 类。

图 6-14 “用户资料维护”对话框

对应于编辑框的成员变量设为如下名称：

```
// Dialog Data
    //{{AFX_DATA(CReaderMDlg)
    enum { IDD = IDD_READER_MAINTAIN };
    CString m_strReaderIDQ;
    CString m_strReaderID;
    CString m_strReaderName;
    CString m_strReaderNameQ;
    CString m_strIDCard;
    //}}AFX_DATA
```

第二步，在“第一笔”按钮的事件脚本中插入如下代码（程序 6-6-1）：