

## 第8章 数据库编程

Visual C++ 6.0为用户提供了ODBC、ADO及OLE DB三种数据库方式。这三种方式中最简单也最常用的是ODBC，因此本章先来重点介绍MFC的ODBC编程方法和技巧，然后介绍基于OLE DB的ADO(ActiveX Data Objects, ActiveX数据对象)技术，最后介绍一些用于数据库的ActiveX控件。

## 8.1 MFC ODBC数据库概述

ODBC是一种使用SQL的程序设计接口，使用ODBC能使用户编写数据库应用程序变得容易简单，避免了与数据源相连接的复杂性。在Visual C++中，MFC的ODBC数据库类CDatabase(数据库类)、CRecordSet(记录集类)和CRecordView(记录视图类)可为用户管理数据库提供了切实可行的解决方案。

# 8.1.1 数据库基本概念

## 1. 数据库和DBMS

数据库是指以一定的组织形式存放在计算机存储介质上的相互关联的数据的集合。例如，把一个学校的教师、学生和课程等数据有序地组织起来，存储在计算机磁盘上，就构成了一个数据库。

为了有效地管理数据库，常常需要一些数据库管理系统(DBMS)为用户提供对数据库操作的各种命令、工具及方法，包括数据库的建立和记录的输入、修改、检索、显示、删除和统计等。流行的DBMS都提供了一个SQL接口。

## 2. SQL

作为用来在DBMS中访问和操作的语言，SQL(结构化查询语言)语句分为两类：一是DDL(Data Definition Language, 数据定义语言)语句，它是用来创建表、索引等，另一是DML(Data Manipulation Language, 数据操作语言)语句，这些语句是用来读取数据、更新数据和执行其他类似操作的语句。

## 8.1.1 数据库基本概念

### 3. ODBC、ADO和OLE DB

ODBC提供了应用程序接口(API),使得任何一个数据库都可以通过ODBC驱动器与指定的DBMS相联。用户的程序就可以通过调用ODBC驱动管理器中相应的驱动程序达到管理数据库的目的。作为Microsoft Windows Open Standards Architecture (WOSA, Windows开放式服务体系结构)的主要组成部分, ODBC一直沿用至今。

ADO类似于用Microsoft Access或Microsoft Visual Basic编写的数据库应用程序,它使用Jet数据库引擎形成一系列的数据访问对象:数据库对象、表和查询对象、记录集对象等。它可以打开一个Access数据库文件(MDB文件),也可直接打开一个ODBC数据源以及使用Jet引擎打开一个ISAM(被索引的顺序访问方法)类型的数据源(dBASE、FoxPro、Paradox、Excel或文本文件)。

OLE DB试图提供一种统一的数据访问接口,并能处理除了标准关系型数据库中的数据之外,还能处理包括邮件数据、Web上的文本或图形、目录服务(Directory Services)以及主机系统中的IMS和VSAM数据。OLE DB提供一个数据库编程COM(组件对象模型)接口,使得数据的使用者(应用程序)可以使用同样的方法访问各种数据,而不用考虑数据的具体存储地点、格式或类型。

## 8.1.1 数据库基本概念

### 4. ADO

ADO 是目前在Windows环境中比较流行的客户端数据库编程技术。它是Microsoft为最新和最强大的数据访问范例OLE DB而设计的，是一个便于使用的应用程序层接口。ADO使用户应用程序能够通过“OLE DB提供者”访问和操作数

据库服务器中的数据。由于它兼具有强大的数据处理功能(处理各种不同类型的数据源、分布式的数据处理等等)和极其简单、易用的编程接口，因而得到了广泛的应用。

ADO技术基于COM(Component Object Model，组件对象模型)，具有COM组件的许多优点，可以用来构造可复用应用框架，被多种语言支持，能够访问包括关系数据库、非关系数据库及所有的文件系统。另外，ADO还支持各种B/S与基于Web的应用程序，具有远程数据服务RDS(Remote Data Service)的特性，是远程数据存取的发展方向。

## 8.1.2 MFC ODBC向导过程

用MFC AppWizard使用ODBC数据库的一般过程是：

- ①用Access或其他数据库工具构造一个数据库；
- ②在Windows中为刚才构造的数据库定义一个ODBC数据源；
- ③在创建数据库处理的文档应用程序向导中选择数据源；
- ④设计界面，并使控件与数据表字段关联。

## 8.1.2 MFC ODBC向导过程

### 1. 构造数据库

数据库表与表之间的关系构成了一个数据库。作为示例，这里用Microsoft Access 创建一个数据库Student.mdb，其中暂包含一个数据表score，用来描述学生课程成绩，如表8.1所示。在表中包括上、下两部分，上部分是数据表的记录内容，下部分是数据表的结构内容。

学号(studentno)↵		课程号(course)↵		成绩(score)↵		学分(credit)↵	
21010101↵		2112105↵		80↵		3↵	
21010102↵		2112348↵		85↵		2.5↵	
21010501↵		2121344↵		70↵		3↵	
21010502↵		2121331↵		78↵		3↵	
序 号↵	字段名称↵	数据类型↵	字段大小↵	小数位↵	字段含义↵		
1↵	studentno↵	文本↵	8↵	↵	学号↵		
2↵	course↵	文本↵	7↵	↵	课程号↵		
3↵	score↵	数字↵	单精度↵	1↵	成绩↵		
4↵	credit↵	数字↵	单精度↵	1↵	学分↵		

## 8.1.2 MFC ODBC向导过程

### 2. 创建ODBC数据源

Windows中的ODBC组件是出现在系统的“控制面板”管理工具中，如图8.1所示。

双击ODBC图标(在图8.1中已圈定)，进入ODBC数据源管理器。在这里，用户可以设置ODBC数据源的一些信息。其中，“用户DSN”页面是用来定义用户自己





## 2. 创建ODBC数据源

创建用户DSN的过程如下。

(1) 单击[添加]按钮，弹出有一驱动程序列表的“创建新数据源”对话框，在该对话框

框中选择要添加用户数据源的驱动程序，这里选择“Microsoft Access Driver”，如

图8.3所示。

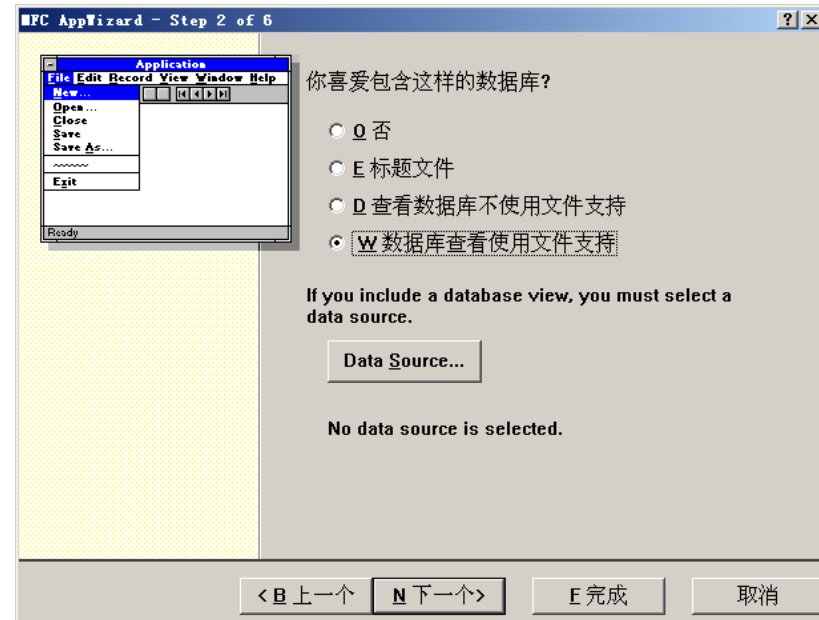
(2) 单击[完成]按钮，进入指定驱动程序的安装对话框，单击[选择]按钮将前面创建的数据源输入“Database Example For VC++”结果

图



## 2. 创建ODBC数据源

(3) 单击[确定]按钮，刚才创建的用户数据源被添加在“ODBC数据源管理器”的“用户数据源”列表中，如图0.5所示。



## 8.1.2 MFC ODBC向导过程

### 3. 在MFC AppWizard中选择数据源

用MFC AppWizard可以容易地创建一个支持数据库的文档应用程序，如下面的过程。

用MFC AppWizard创建一个单文档应用程序Ex\_ODBC。

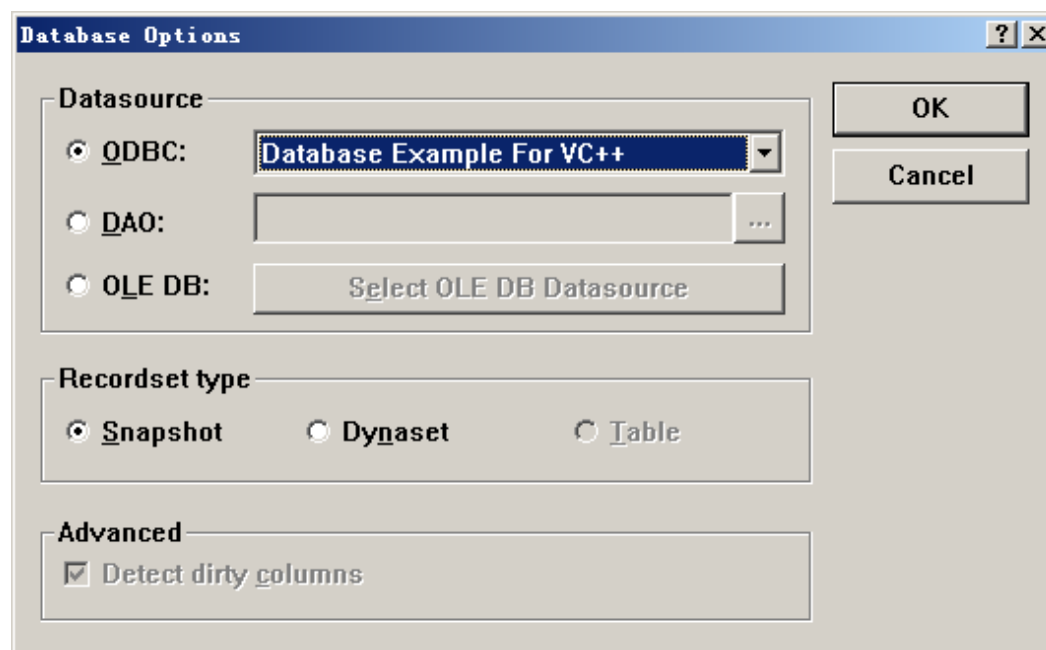
在向导的**第2步**对话框中加入数据库的支持，如图8.6所示。在该对话框中用户可以选择对数据库支持程序，其中各选项的含义如表8.2所示。

选项↵	创建的视图类↵	创建的文档类↵
否(None)↵	从 CView 派生↵	支持文档的常用操作，并在“文件”菜单中有“新建”、“打开”、“保存”、“另存为”等命令。↵
标题文件(Header files only)↵	从 CView 派生↵	除了在 StdAfx.h 文件中添加了“#include <afxdb.h>”语句外，其余与“None”选项相同↵
查看数据库不使用文件支持(Database view without file support)↵	从 CRecordView 派生↵	不支持文档的常用操作，也就是说，创建的文档类不能进行序列化，且在“文件”菜单中没有“新建”等文档操作命令。但用户可在用户视图在中使用 CRecordset 类处理数据库↵
数据库查看使用文件支持(Database view with file support)↵	从 CRecordView 派生↵	↵ 全面支持文档操作和数据库操作↵

### 3. 在MFC AppWizard中选择数据源

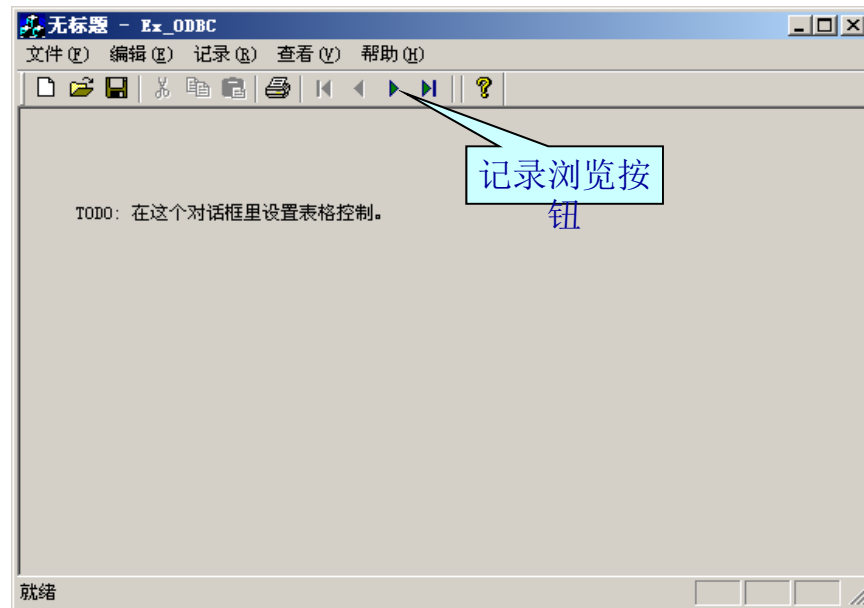
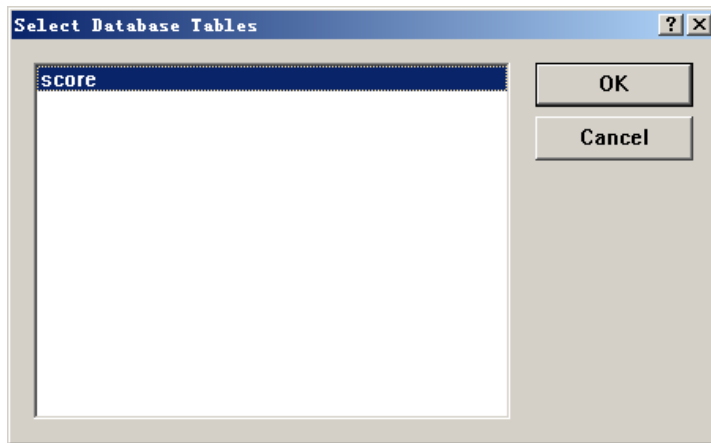
选中“数据库查看使用文件支持”项，单击[Data Source]按钮，弹出“Database

Options”对话框，从中选择ODBC的数据源“Database Example For VC++”，如图8.7所示。



### 3. 在MFC AppWizard中选择数据源

- (4) 保留其他默认选项，单击[OK]按钮，弹出如图8.8所示的“Select Database Tables”对话框，从中选择要使用的表score。
- (5) 单击[OK]按钮，又回到了向导的第2步对话框。
- (6) 单击[完成]按钮。开发环境自动打开表单视图CEx\_ODBCView的对话框资源模板IDD\_EX\_ODBC\_FORM以及相应的对话框编辑器。
- (7) 编译并运行，结果如图8.9所示。



## 8.1.2 MFC ODBC向导过程

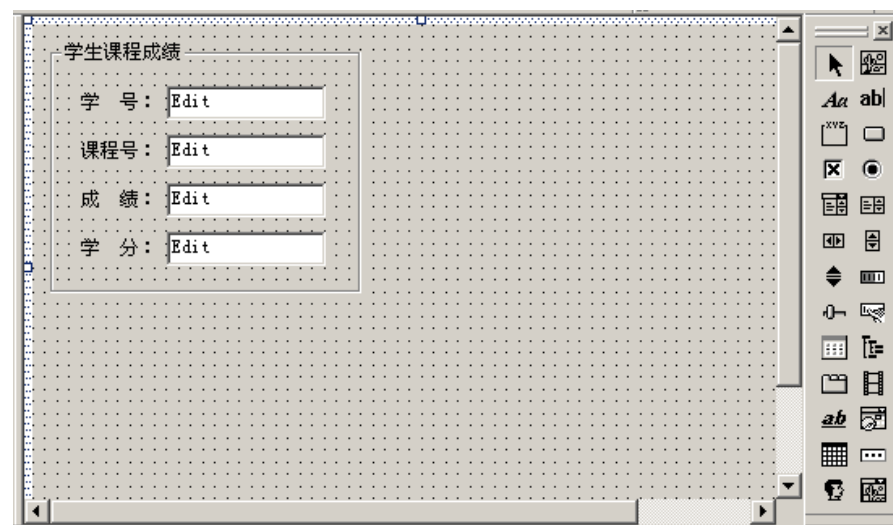
### 4. 设计浏览记录界面

在上面的Ex\_ODBC中，MFC为用户自动创建了用于浏览数据表记录的**工具按钮**和相应的“**记录**”**菜单项**。若用户选择这些浏览记录命令，系统会自动调用相应的

函数来移动数据表的当前位置。

若在表单视图CEx\_ODBCView中添加控件并与表的字段相关联，就可以根据表的当前记录位置显示相应的数据。其步骤如下。

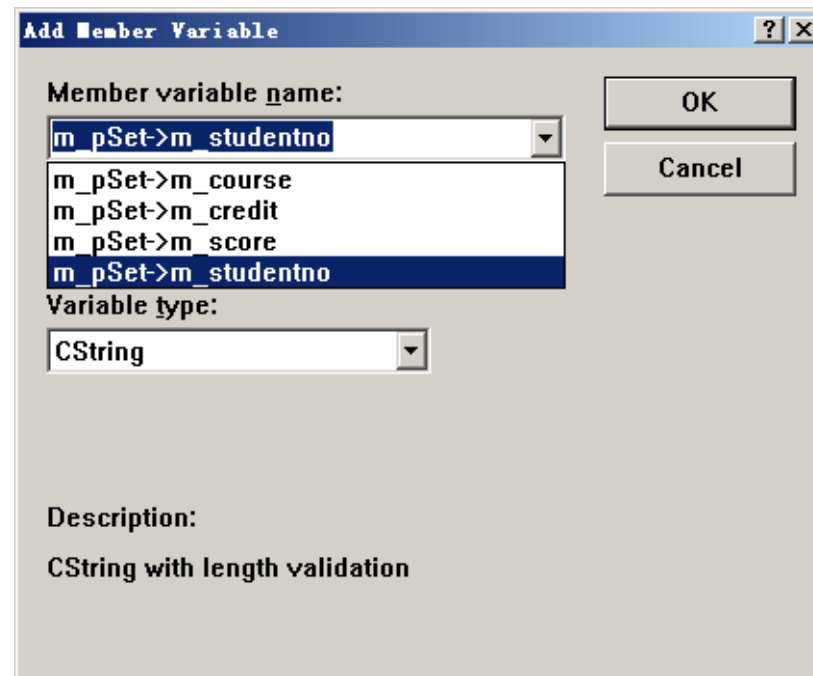
(1) 按照图8.10所示的布局，为表单对话框资源模板添加表8.3所示的控件。



添加的控件↵	ID 号↵	标 题↵	其他属性↵
编辑框(学号)↵	IDC_STUNO↵	——↵	默认 ↵
编辑框(课程号)↵	IDC_COURSENO↵	——↵	默认 ↵
编辑框(成绩)↵	IDC_SCORE↵	——↵	默认 ↵
编辑框(学分)↵	IDC_CREDIT↵	——↵	默认 ↵

## 4. 设计浏览记录界面

(2) 按快捷键**Ctrl+W**，弹出MFC ClassWizard对话框，切换到**Member Variables**页面，在Class name框中选择**CEx\_ODBCView**，为上述控件添加相关联的数据成员。与以往添加控件变量不同的是，这里添加的控件变量都是由系统自动定义的，并与数据库表字段相关联的。例如，双击**IDC\_STUNO**，在弹出的“Add Member Variable”对话框中的成员变量下拉列表中选择要添加的成员变量名**m\_pSet->m\_studentno**，选择后，控件变量的类型将自动设置，如图8.11所示。

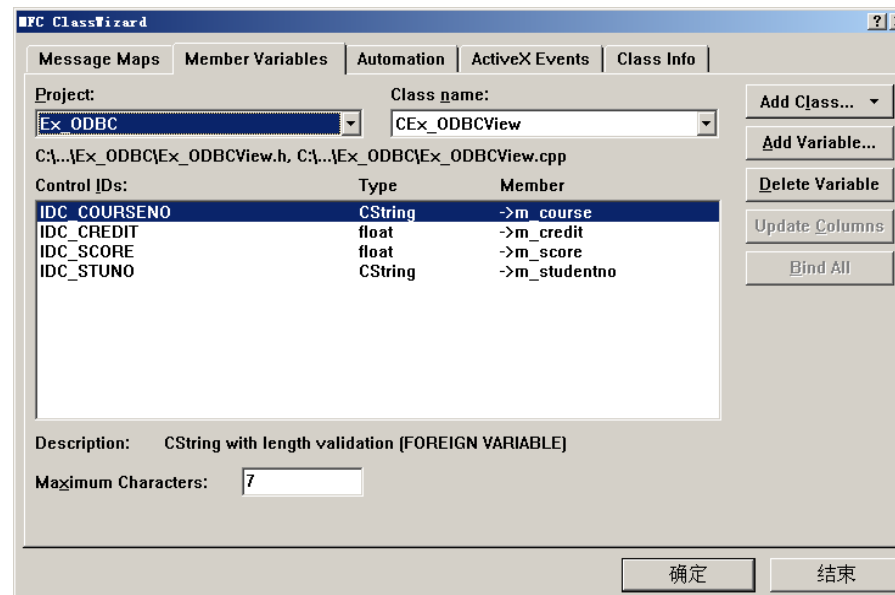




## 4. 设计浏览记录界面

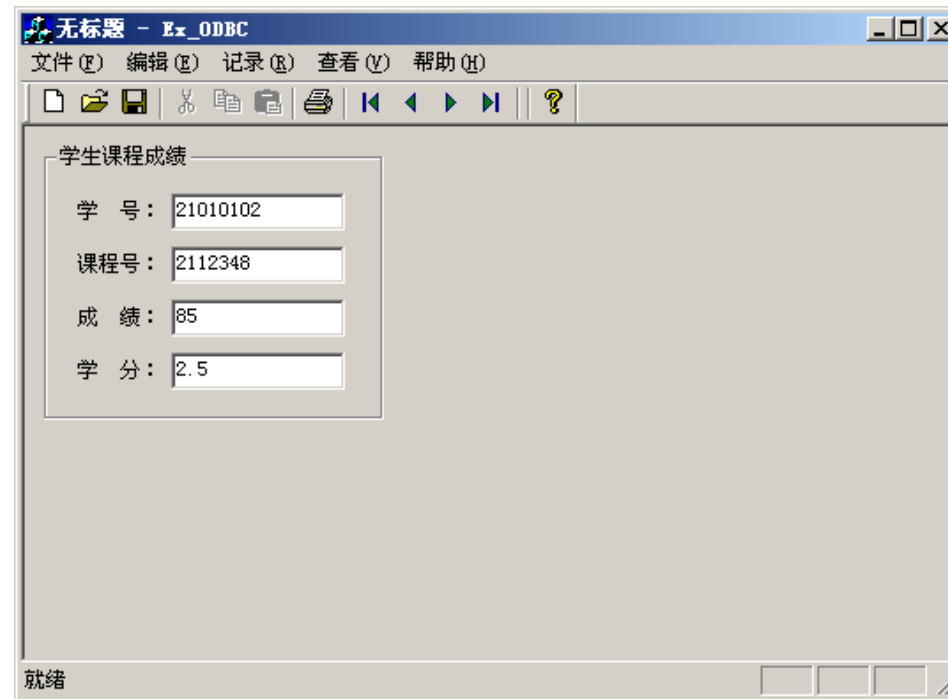
(3) 按照上一步骤的方法，为表8.4所示的其他控件依次添加相关联的成员变量。需要说明的是，控件变量的范围和大小应与数据表中的字段一一对应。结果如图8.12所示。

控件 ID 号↕	变量名↕	范围和大小↕
IDC_COURSENO↕	m_pSet->m_course↕	7↕
IDC_SCORE↕	m_pSet->m_score↕	0~100↕
IDC_SREDIT↕	m_pSet->m_credit↕	1~20↕



## 4. 设计浏览记录界面

(4) 编译运行并测试，结果如图8.13所示。

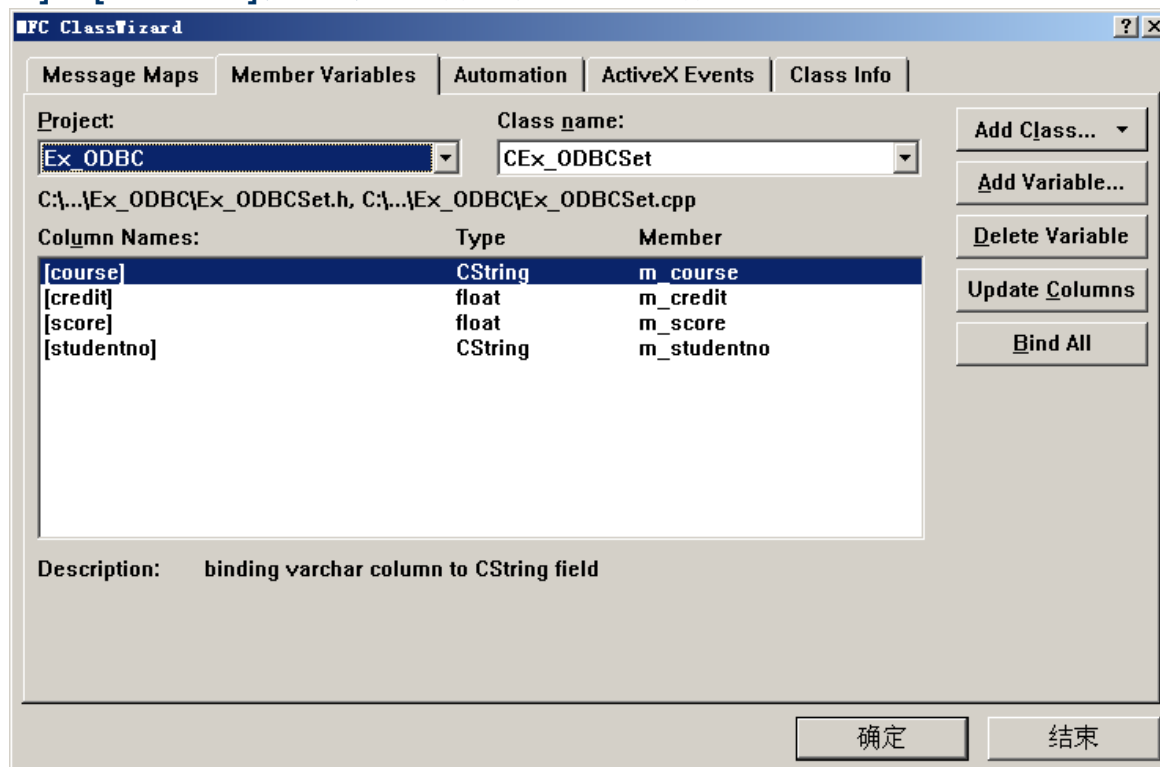


## 8.1.3 ODBC数据表绑定更新

其步骤如下：

- (1) 按快捷键`Ctrl+W`，打开MFC ClassWizard对话框，切换到“**Member Variables**”页面。
- (2) 在“Class name”的下拉列表中选择“**CEx\_ODBCSet**”，此时MFC ClassWizard对话框的

[**Update Columns**]和[**Bind All**]按钮被激活，如图8.14所示。



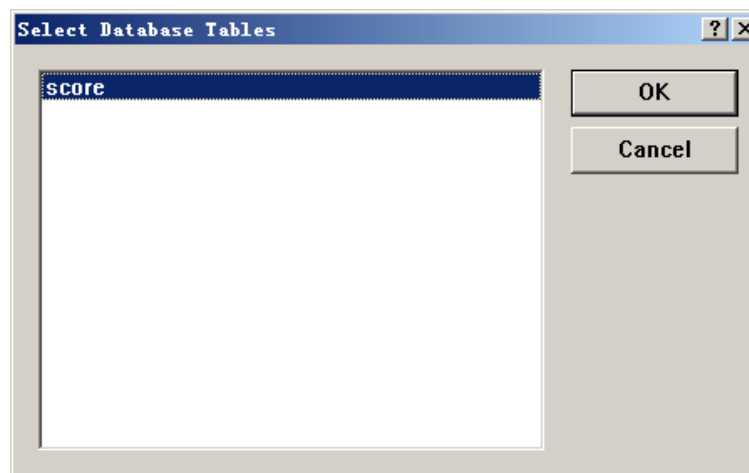
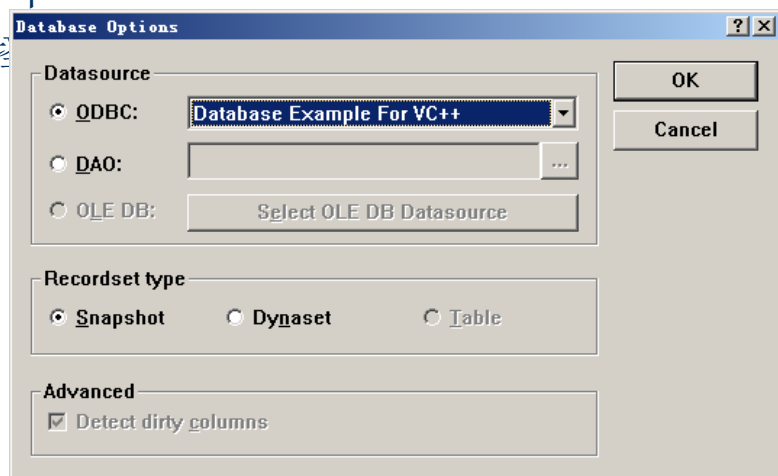
## 8.1.3 ODBC数据表绑定更新

(3) 单击[Update Columns]按钮，又弹出前面的“Database Options”对话框，选择

ODBC数据源“Database Example For VC++”，如图8.15所示。

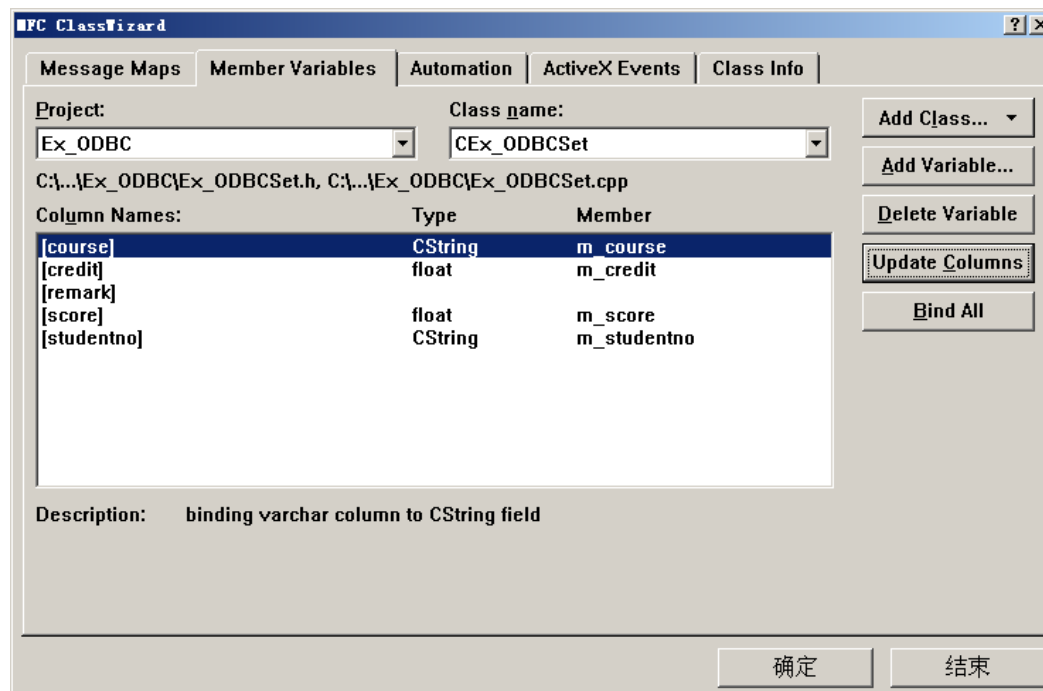
(4) 单击[OK]按钮，弹出如图8.16所示的“Select Database Tables”对话框，从

中



## 8.1.3 ODBC数据表绑定更新

- (5) 单击[OK]按钮，又回到MFC ClassWizard界面，如图8.17所示。
- (6) 单击[Bind All]按钮，MFC Wizard将自动为字段添加相关联的变量。



## 8.2 MFC ODBC应用编程

使用MFC所供的ODBC类：**CDatabase**(数据库类)、**CRecordSet**(记录集类)和**CRecordView**(可视记录集类)。

**CDatabase**类用来提供对数据源的连接，通过它可以对数据源进行操作；

**CRecordView**类用来控制并显示数据库记录，该视图是直接连到一个**CRecordSet**对象的表单视图。但在实际应用过程中，**CRecordSet**类是用户最关心的，因为它为用户提供了对表记录进行操作的许多功能，如查询记录、添加记录、删除记录、修改记录等，并能直接为数据源中的表映射一个**CRecordSet**类对象，方便用户的操作。

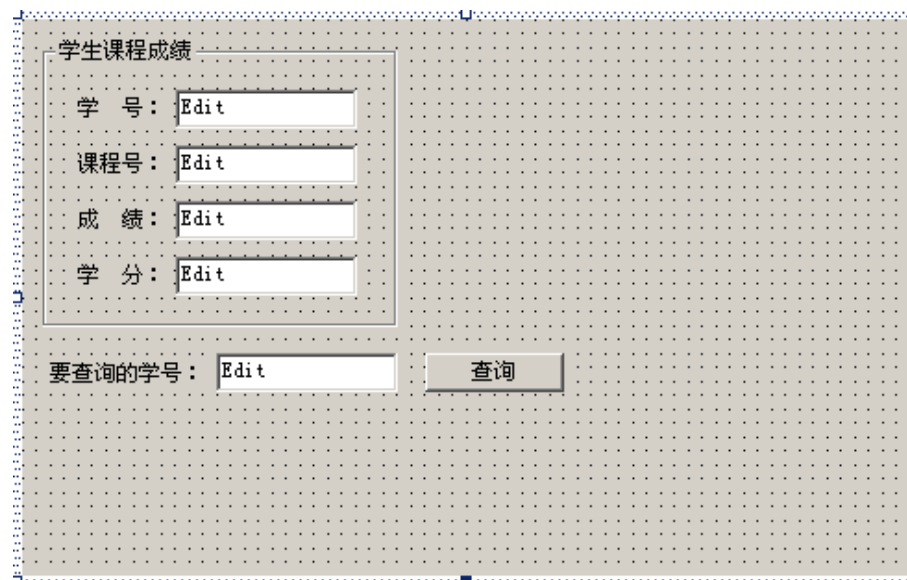
**CRecordSet**类对象提供了从数据源中提取出表的记录集，并提供了两种操作形式：

动态行集(**Dynasets**)和快照集(**Snapshots**)。动态行集能与其他用户所做的更改保持同步，而快照集则是数据的一个静态视图。

## 8.2.1 查询记录

使用CRecordSet类的成员变量m\_strFilter、m\_strSort和成员函数Open可以对表进行记录的查询和排序。先来看一个示例，该示例在前面的Ex\_ODBC的表单中添加一个编辑框和一个[查询]按钮，单击[查询]按钮，将按编辑框中的学号内容对数据表进行查询，并将查找到的记录显示在前面添加的控件中。具体过程如下：

(1) 打开Ex\_ODBC应用程序的表单资源，按图8.18所示的布局添加控件，其中添加的编辑框ID号设为IDC\_EDIT\_QUERY，“查询”按钮的ID号设为IDC\_BUTTON\_QUERY。



## 8.2.1 查询记录

(2) 用MFC ClassWizard为控件IDC\_EDIT\_QUERY添加关联变量m\_strQuery。

(3) 在CEx\_ODBCView类中添加按钮控件IDC\_BUTTON\_QUERY的BN\_CLICKED消息映射，并在映射函数中添加下列代码：

```
void CEx_ODBCView::OnButtonQuery()
{
    UpdateData();
    m_strQuery.TrimLeft();
    if (m_strQuery.IsEmpty()) {
        MessageBox("要查询的学号不能为空！");
        return;
    }
    if (m_pSet->IsOpen())
        m_pSet->Close();           // 如果记录集打开，则先关闭
    m_pSet->m_strFilter.Format("studentno='%s'",m_strQuery);
    // studentno是score表的字段名，用来指定查询条件
    m_pSet->m_strSort = "course";
}
```



```

// course是score表的字段名，用来按course字段从小到大排序
m_pSet->Open();                // 打开记录集
if (!m_pSet->IsEOF())           // 如果打开记录集有记录
    UpdateData(FALSE);         // 自动更新表单中控件显示的内容
else
    MessageBox("没有查到你要找的学号记录!");
}

```

代码中， m\_strFilter和m\_strSort是CRecordSet的成员变量，用来执行条件查询和结果排序。其中， m\_strFilter称为“过滤字符串”，相当于SQL语句中

WHERE后

的条件串；而m\_strSort称为“排序字符串”，相当于SQL语句中ORDER BY后的

字符串。若字段的数据类型是文本，则需要在m\_strFilter字符串中将单引号将查询的内容括起来，对于数字，则不需要用单引号。

## 8.2.1 查询记录

(4) 编译运行并测试，结果如图8.19所示。

无标题 - Ex\_ODBC

文件(F) 编辑(E) 记录(R) 查看(V) 帮助(H)

学生课程成绩

学号: 21010102

课程号: 2112348

成绩: 85

学分: 2.5

要查询的学号: 21010102 查询

就绪

## 8.2.2 编辑记录

**CRecordSet**类为用户提供了许多对表记录进行操作的成员函数用来添加记录、删除记录和修改记录等。

### 1. 增加记录

增加记录是使用**AddNew**函数，但要求数据库必须是以“可增加”的方式打开的。下面的代码是在表的末尾增加新记录：

```
m_pSet->AddNew();           // 在表的末尾增加新记录
m_pSet->SetFieldNull(&(m_pSet->m_studentno), FALSE);
// 设定m_studentno值不为空(NULL)
m_pSet->m_studentno = "21010503";
.....                      // 输入新的字段值
m_pSet->Update();            // 将新记录存入数据库
m_pSet->Requery();           // 刷新记录集，这在快照集方式是必须的
```

## 8.2.2 编辑记录

### 1. 删除记录

可以直接使用CRecordSet::Delete函数来删除记录。需要说明的是，要使删除操作有效，还需要移动记录函数。例如下面的代码：

```
CRecordsetStatus status;  
m_pSet->GetStatus(status);           // 获取当前记录集状态  
m_pSet->Delete();                     // 删除当前记录  
if (status.m_lCurrentRecord==0)      // 若当前记录索引号为0(0表示第  
                                     一条记录)则  
    m_pSet->MoveNext();               // 下移一个记录  
else  
    m_pSet->MoveFirst();               // 移动到第一个记录处  
UpdateData(FALSE);
```

## 8.2.2 编辑记录

### 3. 修改记录

函数CRecordSet::Edit可以用来修改记录，例如：

```
m_pSet->Edit();                // 修改当前记录
m_pSet->m_name="刘向东";        // 修改当前记录字段值
.....
m_pSet->Update();               // 将修改结果存入数据库
m_pSet->Requery();
```

### 4. 撤消操作

如果用户在进行增加或者修改记录后，希望放弃当前操作，则在调用CRecordSet::Update()函数之前调用CRecordSet::Move(AFX\_MOVE\_REFRESH)来撤消操作，便可恢复在增加或修改操作之前的当前记录。

## 8.2.3 字段操作

CRecordSet类中的成员变量m\_nFields (用于保存数据表的字段个数)和成员函数GetODBCFieldInfo及GetFieldValue可以简化多字段的访问操作。

GetODBCFieldInfo函数用来数据表中的字段信息，其函数原型如下：

**void GetODBCFieldInfo( short nIndex, CODBFieldInfo& fieldinfo );**

其中，nIndex用于指定字段索引号，0表示第一个字段，1表示第二个字段，以此类推。fieldinfo是CODBFieldInfo结构参数，用来表示字段信息。

CODBFieldInfo结构体原型如下：

```
struct CODBFieldInfo
{
    CString m_strName;           // 字段名
    SWORD m_nSQLType;            // 字段的SQL数据类型
    UDWORD m_nPrecision;        // 字段的文本大小或数据大小
    SWORD m_nScale;              // 字段的小数点位数
    SWORD m_nNullability;        // 字段接受空值(NULL)能力
};
```

## 8.2.3 字段操作

结构体裁中，SWORD和UDWORD分别表示short int和unsigned long int数据类型。

GetFieldValue函数用来获取数据表当前记录中指定字段的值，其常用的函数原型如下：

```
void GetFieldValue( short nIndex, CString& strValue );
```

其中，nIndex用于指定字段索引号，strValue用来返回字段的内容。

除了上述字段操作外，CRecordSet类的成员函数GetRecordCount和GetStatus，还可分别用来获得表中的记录总数和当前记录的索引，其原型如下：

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

```
void GetStatus( CRecordsetStatus& rStatus ) const;
```

其中，参数rStatus是指向下列的CRecordsetStatus结构的对象：

```
struct CRecordsetStatus
```

```
{
```

```
    long m_lCurrentRecord;           // 当前记录的索引，0表示第一个记录，
```

```
    // 1表示第二个记录，依次类推。但-1表示在第一个记录之前，-2表示不确定。
```

```
    BOOL m_bRecordCountFinal;       // 记录总数是否是最终结果
```

```
};
```

## 8.2.3 字段操作

[例Ex\_Field] 字段的编程操作

1) 为数据库Student.mdb添加一个数据表course

用Microsoft Access 为数据库Student.mdb添加一个数据表course，如表8.5所示。表中上部分是数据表的记录内容，下部分是数据表的结构内容。

课程号 (course no) ↕	所属专业 (special) ↕	课程名 (course name) ↕	类型 (course type) ↕	开课学期 (open term) ↕	课时数 (hours) ↕	学分 (credit) ↕
2112105↕	机械工程及其自动化↕	C 语言程序设计↕	专修↕	3↕	48↕	3↕
2112348↕	机械工程及其自动化↕	AutoCAD↕	选修↕	6↕	51↕	2.5↕
2121331↕	电气工程及其自动化↕	计算机图形学↕	方向↕	5↕	72↕	3↕
2121344↕	电气工程及其自动化↕	Visual C++程序设计↕	通修↕	4↕	60↕	3↕
序 号↕	字段名称↕	数据类型↕	字段大小↕	小数位↕	字段含义↕	
1↕	course no↕	文本↕	7↕	——↕	课程号↕	
2↕	special↕	文本↕	50↕	——↕	所属专业↕	
3↕	course name↕	文本↕	50↕	——↕	课程名↕	
4↕	course type↕	文本↕	10↕	——↕	课程类型↕	
5↕	open term↕	数字↕	字节↕	——↕	开课学期↕	
6↕	hours↕	数字↕	字节↕	——↕	课时数↕	
7↕	credit↕	数字↕	单精度↕	1↕	学分↕	



## [例Ex\_Field]

### 2) 为文档应用程序添加ODBC的支持

(1) 用MFC AppWizard创建一个默认的单文档应用程序Ex\_Field，但在向导的第6步将CEx\_FieldView的基类由默认的CView选择为CListView类。

(6) 将项目工作区窗口切换到FileView页面，展开Header Files所有项，双击stdafx.h，打开该文件。

(3) 在stdafx.h中添加ODBC数据库支持的头文件包含#include <afxdb.h>，如下面的代码：

```
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>           // MFC support for Windows Common
                             Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
#include <afxdb.h>
```

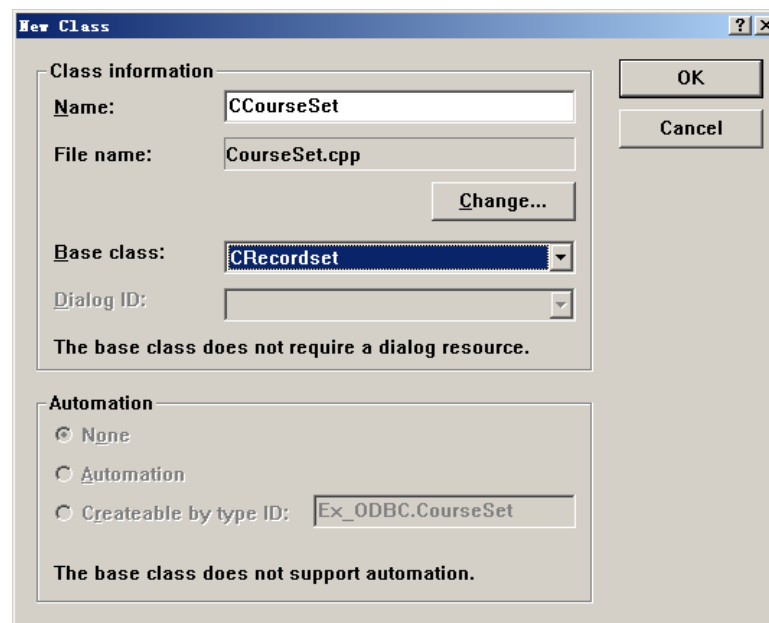
## [例Ex\_Field]

### 3) 创建数据表score的CRecordSet派生类

(1) 按快捷键Ctrl+W，打开MFC ClassWizard对话框。单击[Add Class]按钮，从弹出的下拉菜单中选择“New”。

(2) 在弹出的“Add Class”对话框中指定CRecordSet的派生类CCourseSet，结果

如图8.23所示。



### 3) 创建数据表score的CRecordSet派生类

(3) 单击[OK]按钮，弹出“Database Options”对话框。从中选择ODBC的数据源“Database Example For VC++”，单击[OK]按钮，弹出“Select Database Tables”

对话框，从中选择要使用的表course。

(4) 单击[OK]按钮回到MFC ClassWizard界面，单击[确定]按钮后，系统自动为用户生成CCourseSet类所需要的代码。

(5) 在CEx\_FieldView::PreCreateWindow函数中添加修改列表视图风格的代码：

```
BOOL CEx_FieldView::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.style &= ~LVS_TYPEMASK;
    cs.style |= LVS_REPORT;           // 报表方式
    return CListView::PreCreateWindow(cs);
}
```

### 3) 创建数据表score的CRecordSet派生类

(6) 在CEx\_FieldView::OnInitialUpdate函数中添加下列代码:

```
void CEx_FieldView::OnInitialUpdate()
{
    CListView::OnInitialUpdate();
    CListCtrl& m_ListCtrl = GetListCtrl();    // 获取内嵌在列表视图中的列表控件
    CCourseSet cSet;
    cSet.Open();                                // 打开记录集
    CODBCFieldInfo field;
    // 创建列表头
    for (UINT i=0; i<cSet.m_nFields; i++)
    {
        cSet.GetODBCFieldInfo( i, field );
        m_ListCtrl.InsertColumn(i,field.m_strName,LVCFMT_LEFT,100);
    }
}
```

```
// 添加列表项
int nltem = 0;
CString str;
while (!cSet.IsEOF())
{
    for (UINT i=0; i<cSet.m_nFields; i++)
    {
        cSet.GetFieldValue(i, str);
        if ( i == 0) m_ListCtrl.InsertItem( nltem, str );
        else        m_ListCtrl.SetItemText( nltem, i, str );
    }
    nltem++;
    cSet.MoveNext();
}
cSet.Close();                                // 关闭记录集
}
```

### 3) 创建数据表score的CRecordSet派生类

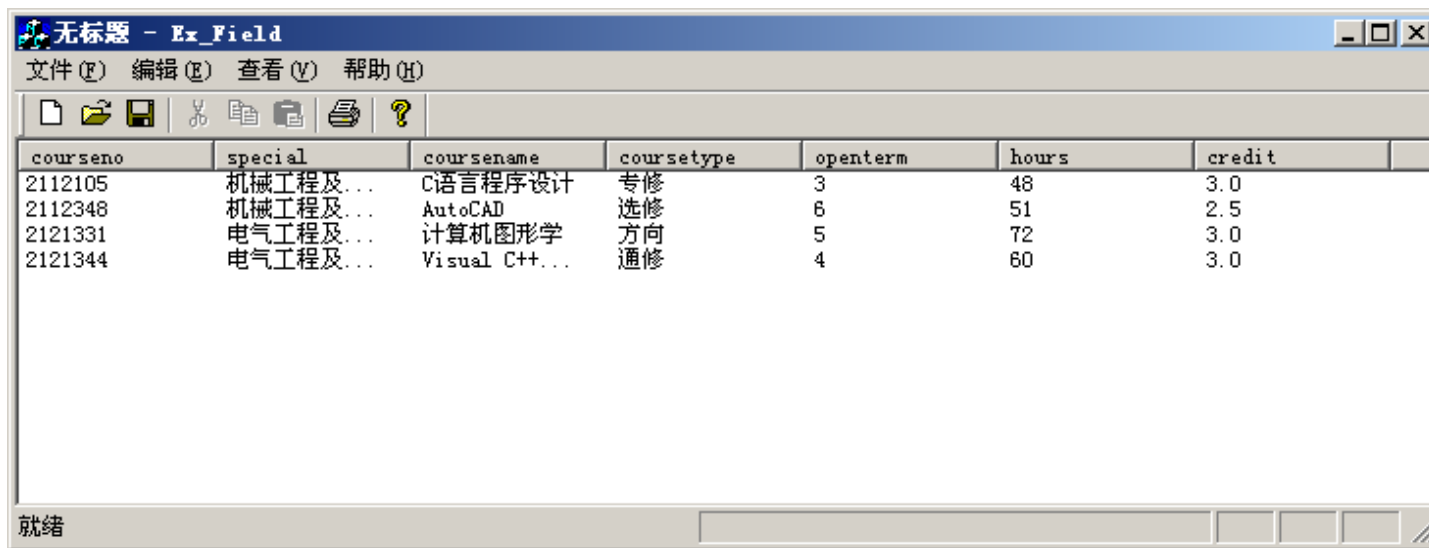
(7) 在Ex\_FieldView.cpp文件的前面添加CCourseSet类的头文件包含:

```
#include "Ex_FieldDoc.h"
```

```
#include "Ex_FieldView.h"
```

```
#include "CourseSet.h"
```

(8) 编译运行，结果如图8.24所示。



courseno	special	coursename	coursetype	openterm	hours	credit
2112105	机械工程及...	C语言程序设计	专修	3	48	3.0
2112348	机械工程及...	AutoCAD	选修	6	51	2.5
2121331	电气工程及...	计算机图形学	方向	5	72	3.0
2121344	电气工程及...	Visual C++...	通修	4	60	3.0

## [例Ex\_Field]

4) 在状态栏中显示当前记录号和记录总数

(1) 在MainFrm.cpp文件中，向原来的indicators数组添加一个元素，用来在状态

(2) 栏上增加一个窗格，修改的结果如下：

```
static UINT indicators[] =
```

```
{
```

```
    ID_SEPARATOR,
```

```
// 第一个信息行窗格
```

```
    ID_SEPARATOR,
```

```
// 第二个信息行窗格
```

```
    ID_INDICATOR_CAPS,
```

```
    ID_INDICATOR_NUM,
```

```
    ID_INDICATOR_SCRL,
```

```
};
```

## 4) 在状态栏中显示当前记录号和记录总数

(3)在Ex\_FieldView.cpp文件的前面添加一个全局函数DispRecNum成员函数，其代码如下：

```
void DispRecNum(CCourseSet *pSet)
{
    CString str;
    CMainFrame* pFrame = (CMainFrame*)AfxGetApp()->m_pMainWnd;

    // 获得主框架窗口的指针
    CStatusBar* pStatus = &pFrame->m_wndStatusBar;

    // 获得主框架窗口中的状态栏指针
    if (pStatus)
    {
        CRecordsetStatus rStatus;
        pSet->GetStatus(rStatus);           // 获得当前记录信息
        str.Format("当前记录:%d 总记录:%d",1+rStatus.m_lCurrentRecord,
                  pSet->GetRecordCount());
        pStatus->SetPaneText(1,str);        // 更新第二个窗格的文本
    }
}
```

该函数先获得状态栏对象的指针，然后调用SetPaneText函数更新第二个窗格的文本。



## 4) 在状态栏中显示当前记录号和记录总数

(4) 在CEx\_FieldView的OnInitialUpdate函数处添加下列代码:

```
void CEx_FieldView::OnInitialUpdate()
{
    ...
    CString str;
    while (!cSet.IsEOF())
    {
        ...
    }
    ::DispRecNum( &cSet );
    cSet.Close();           // 关闭记录集
}
```

## 4) 在状态栏中显示当前记录号和记录总数

(5) 在Ex\_FieldView.cpp文件的开始处增加下列语句:

```
#include "Ex_FieldDoc.h"
```

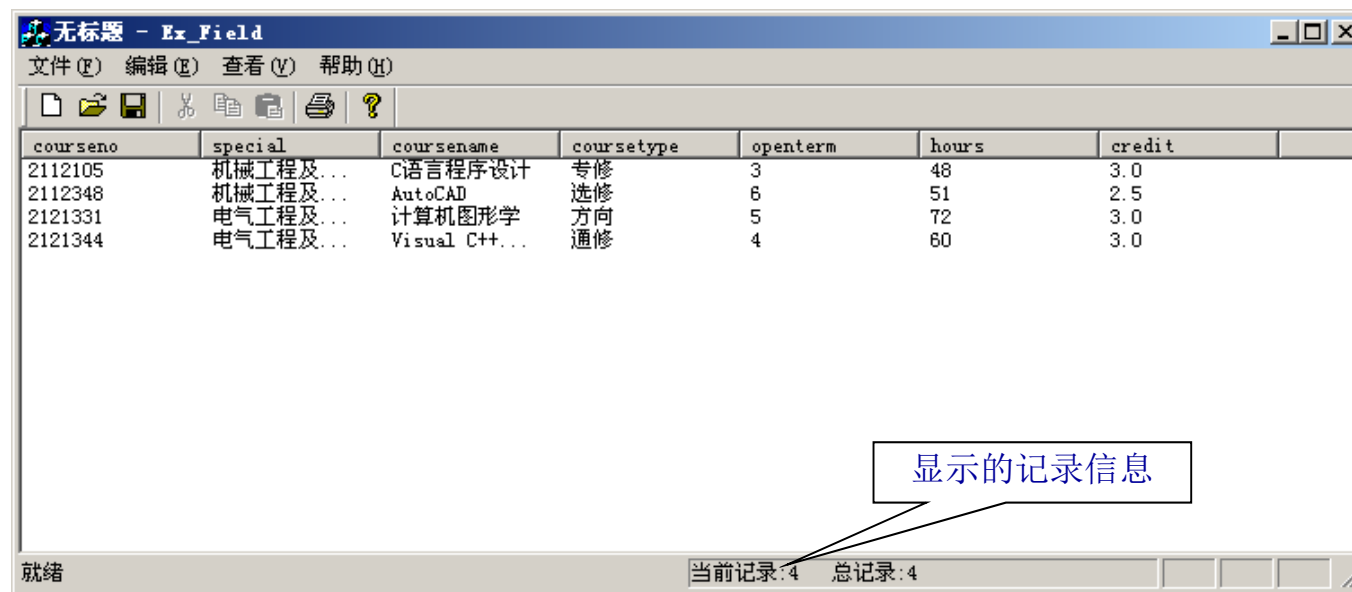
```
#include "Ex_FieldView.h"
```

```
#include "CourseSet.h"
```

```
#include "MainFrm.h"
```

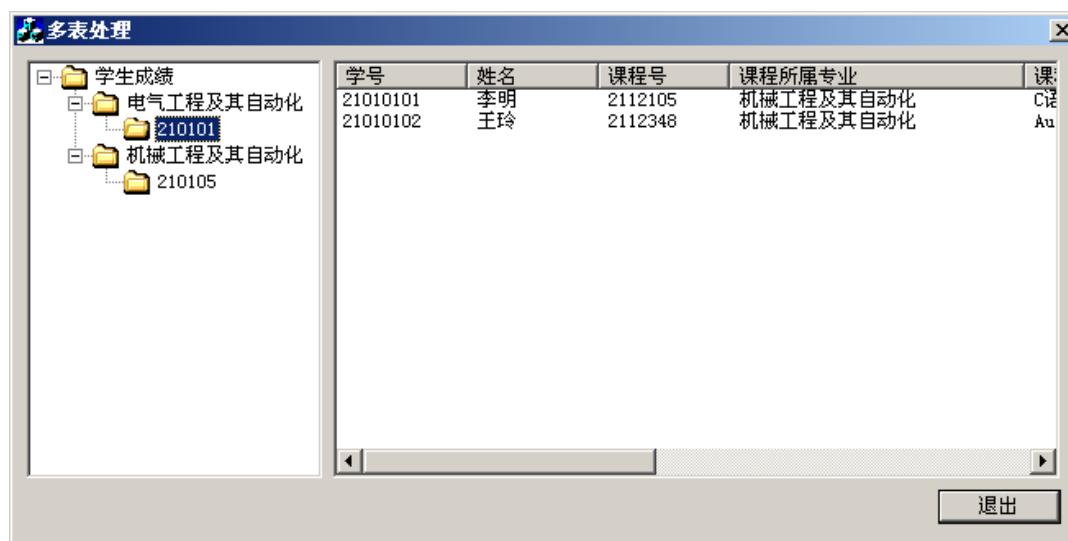
(6) 将MainFrm.h文件中的保护型变量m\_wndStatusBar变成公共变量。

(7) 编译运行并测试, 结果如图8.25所示。



## 8.2.4 多表处理

数据库中表与表之间往往存在着一定的关系，例如要显示一个学生的课程成绩信息，信息包括学号、姓名、课程号、课程所属专业、课程名称、课程类别、开课学期、课时数、学分、成绩，则要涉及到前面的学生课程成绩表(score)、课程表以及学生基本信息表。下面的示例在一个对话框中用两个控件来进行学生课程成绩信息的相关操作，如图8.26所示，左边是树视图，用来显示学生成绩、专业和班级号三个层次信息，单击班级号，所有该班级的学生课程成绩信息将在右边的列表视图中显示出来。



## 8.2.4 多表处理

[例Ex\_Student] 多表处理

1) 为数据库Student.mdb添加一个数据表student

用Microsoft Access 为数据库Student.mdb添加一个数据表student，如表8.6所示。表中上部分是数据表的记录内容，下部分是数据表的结构内容。

姓名↵ (studentname)↵	学号↵ (sudentno)↵	性别↵ (xb)↵	出生年月↵ (birthday)↵	专业↵ (special)↵	
李明↵	21010101↵	true↵	1985-1-1↵	电气工程及其自动化↵	
王玲↵	21010102↵	false↵	1985-1-1↵	电气工程及其自动化↵	
张芳↵	21010501↵	false↵	1985-1-1↵	机械工程及其自动化↵	
陈涛↵	21010502↵	true↵	1985-1-1↵	机械工程及其自动化↵	
序 号↵	字段名称↵	数据类型↵	字段大小↵	小数位↵	字段含义↵
1↵	studentname↵	文本↵	20↵	↵	姓名↵
2↵	studentno↵	文本↵	10↵	↵	学号↵
3↵	xb↵	是/否↵	↵	↵	性别↵
4↵	birthday↵	日期/时间↵	↵	↵	出生年月↵
5↵	special↵	文本↵	50↵	↵	专业↵

## [例Ex\_Student]

### 2) 创建并设计对话框应用程序

- (1) 用MFC AppWizard创建一个默认的基于对话框应用程序Ex\_Student。
- (2) 在打开的对话框资源模板中，删除[取消]按钮和默认的静态文本控件。
- (3) 调整对话框大小，将对话框的标题文本改为“处理多表”，将[确定]按钮的标题文本改为“退出”。
- (4) 参看图8.26的控件布局，向对话框中添加一个树控件，在其属性对话框中，选中“有按钮”、“有行(Lines,线)”、“Lines at root”和“总是显示选择”属性。
- (5) 向对话框中添加一个列表控件，在其属性对话框中，将“查看”属性选为“Report”。
- (6) 用MFC ClassWizard在CEx\_StudentDlg类中，添加树控件的控件变量为m\_treeCtrl，添加列表控件的控件变量为m\_listCtrl。

### 3) 添加对MFC ODBC的支持及记录集

在stdafx.h文件中添加ODBC数据库支持的头文件包含#include <afxdb.h>。  
用MFC ClassWizard为数据表student、course和score分别创建CRecordSet派生类CStudentSet、CCourseSet和CScoreSet。

## [例]Ex\_Student]

### 4) 完善左边树控件的代码

(1) 为CEx\_StudentDlg类添加一个成员函数FindTreeltem，用来查找指定节点下是否有指定节点文本的子节点，该函数的代码如下：

```
HTREEITEM CEx_StudentDlg::FindTreeltem(HTREEITEM hParent, CString str)
{
    HTREEITEM hNext;
    CString strltem;
    hNext = m_treeCtrl.GetChildItem( hParent);
    while (hNext != NULL)
    {
        strltem = m_treeCtrl.GetItemText( hNext );
        if ( strltem == str )
            return hNext;
        else
            hNext = m_treeCtrl.GetNextItem( hNext, TVGN_NEXT );
    }
    return NULL;
}
```

## 4) 完善左边树控件的代码

(2) 为CEx\_StudentDlg类添加一个CImageList成员变量m\_ImageList。

(3) 在CEx\_StudentDlg::OnInitDialog中添加下列代码:

```
BOOL CEx_StudentDlg::OnInitDialog()
```

```
{
```

```
...
```

```
SetIcon(m_hIcon, FALSE); // Set small icon
```

```
m_ImageList.Create(16, 16, ILC_COLOR8 | ILC_MASK, 2, 1);
```

```
m_treeCtrl.SetImageList( &m_ImageList, TVSIL_NORMAL );
```

```
SHFILEINFO fi; // 定义一个文件信息结构变量
```

```
SHGetFileInfo("C:\\Windows", 0, &fi, sizeof(SHFILEINFO),
```

```
SHGFI_ICON | SHGFI_SMALLICON); // 获取文件夹图标
```

```
m_ImageList.Add( fi.hIcon );
```

```
SHGetFileInfo("C:\\Windows", 0, &fi, sizeof(SHFILEINFO),
```

```
SHGFI_ICON | SHGFI_SMALLICON | SHGFI_OPENICON); // 获取打开文件夹图标
```

```
m_ImageList.Add( fi.hIcon );
```

```
HTREEITEM hRoot, hSpec, hClass;
```

```
hRoot = m_treeCtrl.InsertItem("学生成绩",0,1);
```

```
CStudentSet sSet;
```

```
sSet.m_strSort = "special"; // 按专业排序
```

```
sSet.Open();
while (!sSet.IsEOF()){
    hSpec = FindTreeItem( hRoot, sSet.m_special);
    // 查找是否有重复的专业节点
    if (hSpec == NULL)           // 若没有重复的专业节点
        hSpec = m_treeCtrl.InsertItem( sSet.m_special, 0, 1, hRoot);
    hClass = FindTreeItem( hSpec, sSet.m_studentno.Left(6));
    // 查找是否有重复的班级节点
    if (hClass == NULL)         // 若没有重复的班级节点
        hClass = m_treeCtrl.InsertItem(sSet.m_studentno.Left(6), 0, 1, hSpec);
    sSet.MoveNext();
}
sSet.Close();
return TRUE;    // return TRUE unless you set the focus to a control
}
```



## 4) 完善左边树控件的代码

(4) 在Ex\_StudentDlg.cpp文件的前面添加记录集类的包含文件，如下面的代码：

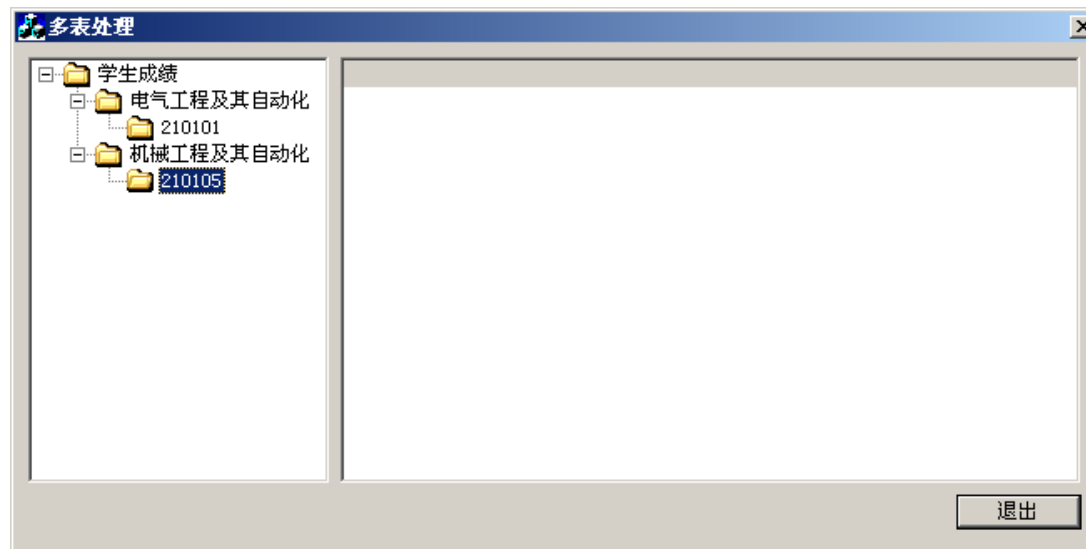
```
#include "Ex_StudentDlg.h"
```

```
#include "StudentSet.h"
```

```
#include "ScoreSet.h"
```

```
#include "CourseSet.h"
```

(5) 编译运行，结果如图8.27所示。



## [例Ex\_Student]

### 5) 完善右边列表控件的代码

(1) 在CEx\_StudentDlg::OnInitDialog函数中添加下列代码，用来创建列表标题头：

```
BOOL CEx_StudentDlg::OnInitDialog()
{
    ...
    sSet.Close();
    // 设置列表头
    CString strHeader[]={"学号","姓名","课程号","课程所属专业",
        "课程名称","课程类别","开课学期","课时数","学分","成绩"};
    int nLong[] = {80, 80, 80, 180, 180, 80, 80, 80, 80, 80};
    for (int nCol=0; nCol<sizeof(strHeader)/sizeof(CString); nCol++)
        m_listCtrl.InsertColumn(nCol,strHeader[nCol],LVCFMT_LEFT,nLong[nCol]);
    return TRUE;           // return TRUE unless you set the focus to a control
}
```

## 5) 完善右边列表控件的代码

(2) 为CEx\_StudentDlg类添加一个成员函数DispScoreAndCourseInfo，用来根据指定的条件在列表控件中用报表形式显示学生成绩的所有信息，该函数的代码如下：

```
void CEx_StudentDlg::DispScoreAndCourseInfo(CString strFilter)
{
    m_listCtrl.DeleteAllItems();           // 删除所有的列表项
    CScoreSet sSet;
    sSet.m_strFilter = strFilter;           // 设置过滤条件
    sSet.Open();                             // 打开score表
    int nItem = 0;
    CString str;
    while (!sSet.IsEOF())
    {
        m_listCtrl.InsertItem( nItem, sSet.m_studentno); // 插入学号
        // 根据score表中的studentno(学号)获取student表中的"姓名"
        CStudentSet uSet;
        uSet.m_strFilter.Format("studentno='%s'", sSet.m_studentno);
        uSet.Open();
        if (!uSet.IsEOF())
            m_listCtrl.SetItemText( nItem, 1, uSet.m_studentname);
        uSet.Close();
        m_listCtrl.SetItemText( nItem, 2, sSet.m_course);
        // 根据score表中的course(课程号)获取course表中的课程信息
    }
}
```

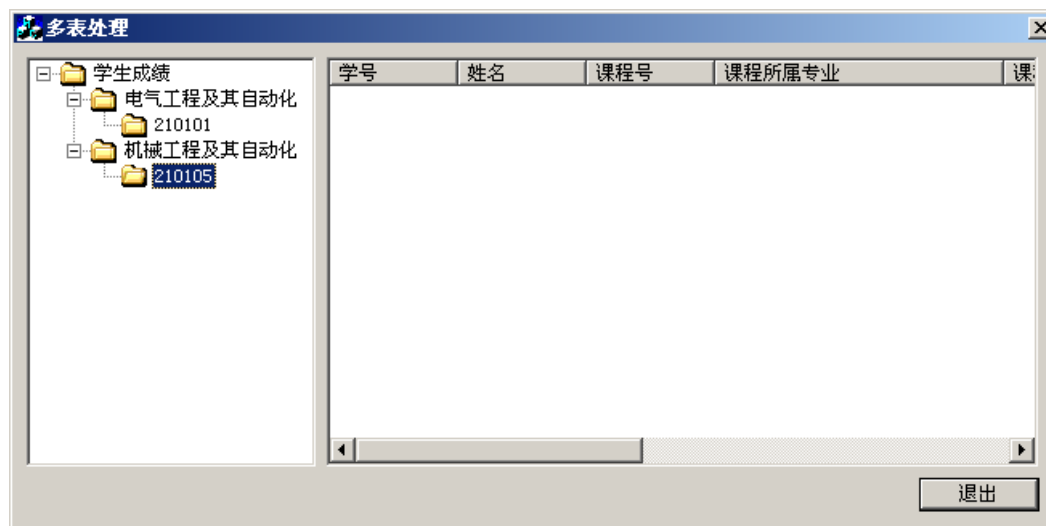
```

CCourseSet cSet;
cSet.m_strFilter.Format("courseno='%s'", sSet.m_course);
cSet.Open();
UINT i = 7;
if (!cSet.IsEOF())
{
    for (i=1; i<cSet.m_nFields; i++)
    {
        cSet.GetFieldValue(i, str);    // 获取指定字段值
        m_listCtrl.SetItemText( nltem, i+2, str);
    }
}
cSet.Close();
str.Format("%0.1f", sSet.m_score);
m_listCtrl.SetItemText( nltem, i+2, str);
sSet.MoveNext();
nltem++;
}
if (sSet.IsOpen()) sSet.Close();
}

```

## 5) 完善右边列表控件的代码

(3) 编译并运行，结果如图8.28所示。



## [例Ex\_Student]

### 6) 完善两控件的关联代码

从上图中可以看出，学生成绩还没有显示出来，下面将实现单击左边的班级号，在右边视图中显示该班级的所有学生成绩信息。

(1) 用MFC ClassWizard为CEx\_StudentDlg类添加TVN\_SELCHANGED消息处理，并添加下列代码：

```
void CEx_StudentDlg::OnSelchangedTree1(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
    HTREEITEM hSelItem = pNMTreeView->itemNew.hItem;        // 获取当前选择的节点
    // 如果当前的节点没有子节点，那说明该节点是班级号节点
    if (m_treeCtrl.GetChildItem(hSelItem) == NULL)
    {
        CString strSelItem, str;
        strSelItem = m_treeCtrl.GetItemText( hSelItem );
        str.Format("studentno LIKE '%s%%'", strSelItem.Left(6));
        DispScoreAndCourseInfo(str);
    }
    *pResult = 0;
}
```

## 6) 完善两控件的关联代码

代码中，调用DispScoreAndCourseInfo 函数是的参数是用来设置数据表(记录集)打开的过滤条件。str是类似的这样内容“studentno LIKE 210101%”，它使得所有学号前面是210101的记录被打开。%是SQL使用的通配符，由于%也是Visaul C++格式前导符，因为在代码中需要两个%。

(2) 编译运行并测试。

## 8.3 ADO数据库编程

ADO最主要的优点是易于使用、速度快、内存开销小，它使用最少的网络流量，并且在前端和数据源之间使用最少的层数，它是一个轻量、高性能的接口。ADO实际上就是由一组Automation对象构成的组件，因此可以像使用其它任何Automation对象一样使用ADO。ADO中最重要的对象有三个：**Connection**、**Command**和**Recordset**，它们分别表示“连接”对象、“命令”对象和“记录集”对象。



## 8.3.1 ADO编程的一般过程

在MFC应用程序中使用ADO数据库的一般过程是：

- ①添加对ADO的支持；
- ②创建一个数据源连接；
- ③对数据源中的数据库进行操作；
- ④关闭数据源。

### 1. 添加对ADO的支持

ADO编程有三种方式：使用预处理指令`#import`、使用MFC中的`CIDispatch`和直接使用COM提供的API。这三种方式中，第一种最为简便，故这里采用这种方法。

# 1. 添加对ADO的支持

## [例Ex\_ADO] 添加对ADO的支持

(1) 用MFC AppWizard创建一个默认的单文档应用程序Ex\_ADO，但在向导的第6步将CEx\_ADOView的基类由默认的CView选择为CListView类，以便更好地显示和操作数据表中的记录。

(2) 在CEx\_ADOView::PreCreateWindow函数添加下列代码，用来设置列表视图内嵌列表控件的风格：

```
BOOL CEx_ADOView::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.style |= LVS_REPORT;           // 报表风格
    return CListView::PreCreateWindow(cs);
}
```

## [例Ex\_ADO]

(3) 在stdafx.h文件中添加对ADO支持的代码:

```
#endif // _AFX_NO_AFXCMN_SUPPORT  
#include <afxcmn.h>          // MFC support for Windows Common Controls  
#import "C:\Program Files\Common Files\System\ADO\msado15.dll" \  
no_namespace rename("EOF", "adoEOF")  
#include <icsint.h>  
//{{AFX_INSERT_LOCATION}}
```

代码中, 预编译命令#import是编译器将此命令中所指定的动态链接库文件引入到程序中, 并从动态链接库文件中抽取出其中的对象和类的信息。

icsint.h文件包含了Visual C++扩展的一些预处理指令、宏等的定义, 用于与数据库数据绑定。

## [例Ex\_ADO]

(4) 在`CEx_ADOApp::InitInstance`函数中添加下列代码，用来对ADO的COM环境进行初始化：

```
BOOL CEx_ADOApp::InitInstance()
{
    ::CoInitialize(NULL);
    AfxEnableControlContainer();    ...
}
```

(5) 在`Ex_ADOView.h`文件中为`CEx_ADOView`定义三个ADO对象指针变量：  
public:

```
    _ConnectionPtr    m_pConnection;
    _RecordsetPtr      m_pRecordset;
    _CommandPtr        m_pCommand;
```

代码中，`_ConnectionPtr`、`_RecordsetPtr`和`_CommandPtr`分别是ADO对象`Connection`、`Recordset`和`Command`的智能指针类型。

## 8.3.1 ADO编程的一般过程

### 2. 连接数据源

只有建立了与数据库服务器的连接后，才能进行其他有关数据库的访问和操作。ADO使用Connection对象来建立与数据库服务器的连接，它相当于MFC中的CDatabase类。和CDatabase类一样，调用Connection对象的Open即可建立与服务器的连接。

**HRESULT Connection::Open(\_bstr\_t ConnectionString, \_bstr\_t UserID, \_bstr\_t Password, long Options)**

其中，ConnectionString为连接字串，UserID是用户名，Password是登录密码，Options是选项，通常用于设置同步和异步等方式。\_bstr\_t是一个COM类，用于字符串BSTR(用于Automation的宽字符)操作。

需要说明的是，正确设置ConnectionString是连接数据源的关键。不同的数据，其连接字串有所不同，见表8.7所示。

数据源↗	格式↗
ODBC↗	"[Provider=MSDASQL] {DSN = name   FileDSN = filename}; [DATABASE=database;] UID = user; PWD = password"↗
Access 数据库↗	"Provider = Microsoft.Jet.OLEDB.4.0; Data Source = databaseName; User ID = userName; Password = userPassWord"↗
Oracle 数据库↗	"Provider = MSDAORA; Data Source = serverName; User ID = userName; Password = userPassword;"↗
MS SQL 数据库↗	"Provider = SQLOLEDB; Data Source = serverName; Initial Catalog = databaseName; User ID = user; Password = userPassword;"↗

## 8.3.1 ADO编程的一般过程

### 3. 关闭连接

用MFC ClassWizard为CEx\_ADOView映射WM\_DESTROY消息，并添加下列代码：

```
void CEx_ADOView::OnDestroy()
{
    CListView::OnDestroy();
    if (m_pConnection)
        m_pConnection->Close();// 关闭连接
}
```

## 8.3.1 ADO编程的一般过程

### 4. 获取数据源信息

Connection对象除了建立与数据库服务器的连接外，还可以通过OpenSchema来获取数据源的自有信息，如：数据表信息、表字段信息以及所支持的数据类型等。下面的代码用来获取student.mdb的数据表名和字段名，并将信息内容显示在列表视图中：

```
void CEx_ADOView::OnInitialUpdate()
{
    CListView::OnInitialUpdate();
    m_pConnection.CreateInstance(__uuidof(Connection)); // 初始化Connection指针
    m_pRecordset.CreateInstance(__uuidof(Recordset));    // 初始化Recordset指针
    m_pCommand.CreateInstance(__uuidof(Command));      // 初始化Recordset指针
    // 连接数据源为"Database Example For VC++"
    m_pConnection->ConnectionString = "DSN=Database Example For VC++";
    m_pConnection->ConnectionTimeout = 30;              // 允许连接超时时间，单位为秒
    HRESULT hr = m_pConnection->Open("", "", "", 0);
    if (hr != S_OK) MessageBox("无法连接指定的数据库！");
    // 获取数据表名和字段名
    _RecordsetPtr pRstSchema = NULL;                  // 定义一个记录集指针
    pRstSchema = m_pConnection->OpenSchema(adSchemaColumns); // 获取表信息
```

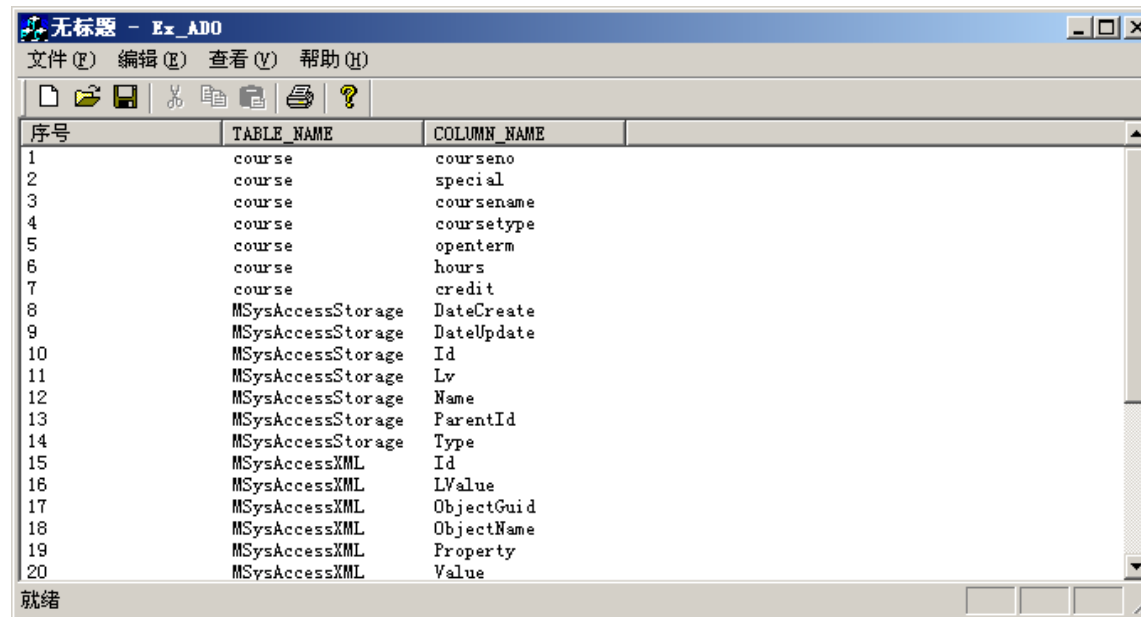
```
// 将表信息显示在列表视图控件中
CListCtrl& m_ListCtrl = GetListCtrl();
CString strHeader[3] = {"序号","TABLE_NAME","COLUMN_NAME"};
for (int i=0; i<3; i++)
    m_ListCtrl.InsertColumn( i, strHeader[i], LVCFMT_LEFT, 120);
int nItem = 0;
CString str;
_bstr_t value;
while(!(pRstSchema->adoEOF))
{
    str.Format("%d", nItem+1 );
    m_ListCtrl.InsertItem( nItem, str );
    for (int i=1; i<3; i++)
    {
        value = pRstSchema->Fields->GetItem((_bstr_t)(LPCSTR)strHeader[i])->Value;
        m_ListCtrl.SetItemText( nItem, i, value );
    }
    pRstSchema->MoveNext();
    nItem++;
}
pRstSchema->Close();
}
```



## 4. 获取数据源信息

代码中，\_\_uuidof用来获取对象的的全局唯一标识(GUID)。ConnectionTimeout是连接超时属性，单位为秒。OpenSchema方法中的adSchemaColumns是一个预定义的枚举常量，用来获取与“列”(字段)相关的信息记录集。该信息记录集的主要字段名有“TABLE\_NAME”、“COLUMN\_NAME”；类似的，若在OpenSchema方法中指定adSchemaTables枚举常量，则返回的记录集的字段名主要有“TABLE\_NAME”、“TABLE\_TYPE”。

上述代码运行结果如图8.29所示。



序号	TABLE_NAME	COLUMN_NAME
1	course	courseno
2	course	special
3	course	coursename
4	course	coursetype
5	course	openterm
6	course	hours
7	course	credit
8	MSysAccessStorage	DateCreate
9	MSysAccessStorage	DateUpdate
10	MSysAccessStorage	Id
11	MSysAccessStorage	Lv
12	MSysAccessStorage	Name
13	MSysAccessStorage	ParentId
14	MSysAccessStorage	Type
15	MSysAccessXML	Id
16	MSysAccessXML	LValue
17	MSysAccessXML	ObjectGuid
18	MSysAccessXML	ObjectName
19	MSysAccessXML	Property
20	MSysAccessXML	Value

图8.29 获取数据源表信息

## 8.3.2 Recordset对象使用

Recordset是用来从数据表或某一个SQL命令执行后获得记录集，通过Recordset对象的AddNew、Update和Delete方法可实现记录的添加、修改和删除等操作。

### 1. 读取数据表全部记录内容

下面的过程是将student.mdb中的course表中的记录显示在列表视图中。

(1) 打开菜单资源IDR\_MAINFRAME，在顶层菜单“查看”下添加一个“显示Course表

记录”子菜单，将其ID号设为ID\_VIEW\_COURSE。

(2) 按快捷键Ctrl+W，弹出ClassWizard对话框，向CEx\_ADOView类添加ID\_VIEW\_COURSE的COMMAND消息映射，保留默认的映射函数OnViewCourse，并在该函数中添加下列代码：

```
void CEx_ADOView::OnViewCourse()
{
    CListCtrl& m_ListCtrl = GetListCtrl();
    // 删除列表中所有行和列表头
    m_ListCtrl.DeleteAllItems();
    int nColumnCount = m_ListCtrl.GetHeaderCtrl()->GetItemCount();
    for (int i=0; i<nColumnCount; i++)
```

```
        m_ListCtrl.DeleteColumn(0);
m_pRecordset->Open( "Course",                // 指定要打开的表
        m_pConnection.GetInterfacePtr(),      // 获取当前数据库连接的接口指针
        adOpenDynamic,                        // 动态游标类型，可以使用Move等操作
        adLockOptimistic,
        adCmdTable);
// 建立列表控件的列表头
FieldsPtr flds = m_pRecordset->GetFields();   // 获取当前表的字段指针
_variant_t Index;
Index.vt = VT_I2;
m_ListCtrl.InsertColumn(0, "序号", LVCFMT_LEFT, 60 );
for (i = 0; i < (int)flds->GetCount(); i++)
{
    Index.iVal=i;
    m_ListCtrl.InsertColumn(i+1, (LPSTR)flds->GetItem(Index)->GetName(),
        LVCFMT_LEFT, 140 );
}
// 显示记录
_bstr_t str, value;
int nItem = 0;
CString strItem;
while(!m_pRecordset->adoEOF)
```

```

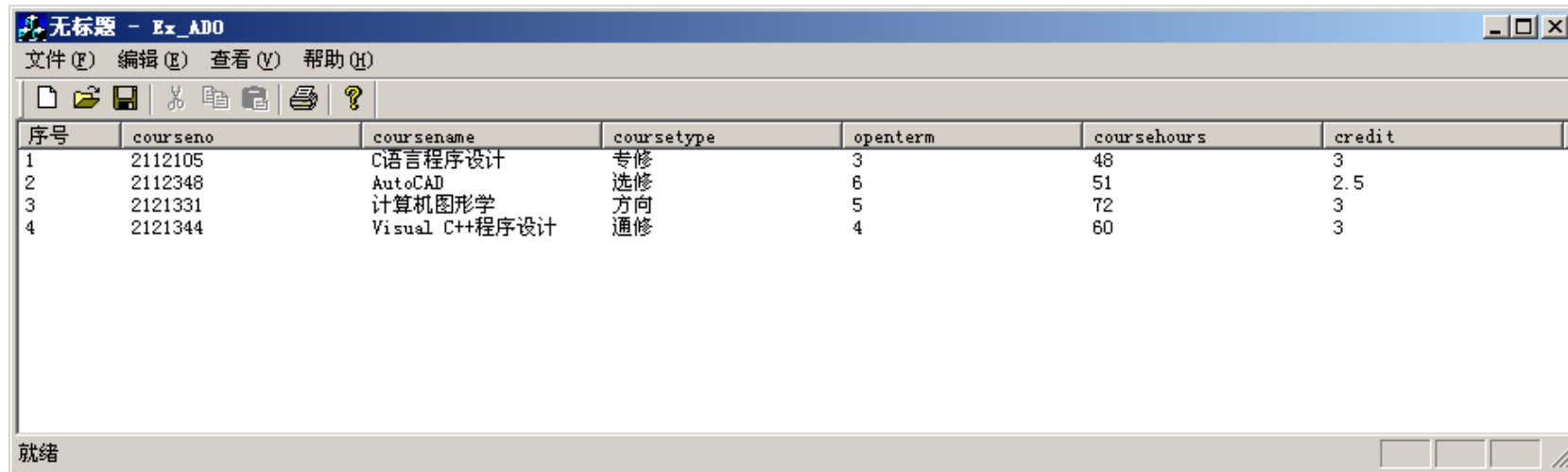
{
    strItem.Format("%d", nItem+1);
    m_ListCtrl.InsertItem(nItem, strItem );
    for (i = 0; i < (int)flds->GetCount(); i++)
    {
        Index.iVal=i;
        str = flds->GetItem(Index)->GetName();
        value = m_pRecordset->GetCollect(str);
        m_ListCtrl.SetItemText( nItem, i+1, (LPCSTR)value );
    }
    m_pRecordset->MoveNext();
    nItem++;
}
m_pRecordset->Close();
}

```

代码中，`_variant_t`是一个用于COM的VARIANT类，VARIANT类型是一个C结构，由于它既包含了数据本身，也包含了数据的类型，因而它可以实现各种不同的自动化数据的传输。

# 1. 读取数据表全部记录内容

(3) 编译运行并测试，结果如图8.30所示。



序号	courseno	coursename	coursetype	openterm	coursehours	credit
1	2112105	C语言程序设计	专修	3	48	3
2	2112348	AutoCAD	选修	6	51	2.5
3	2121331	计算机图形学	方向	5	72	3
4	2121344	Visual C++程序设计	通修	4	60	3

就绪

## 8.3.2 Recordset对象使用

### 2. 添加、修改和删除记录

记录的添加、修改和删除是通过Recordset对象的AddNew、Update和Delete方法来实现的。

例如，向course表中新添加一个记录可有下列代码：

// 打开记录集

```
m_pRecordset->AddNew();                // 添加新记录
    m_pRecordset->PutCollect("courseno",_variant_t("2112111"));
    m_pRecordset->PutCollect("coursehourse",_variant_t(60));
    ...
    m_pRecordset->Update();                // 使添加有效
// 关闭记录集
```

## 2. 添加、修改和删除记录

若从course表中删除一个记录可有下列代码:

```
// 打开记录集
```

```
...
```

```
m_pRecordset->Delete(adAffectCurrent);    // 删除当前行
```

```
m_pRecordset->MoveFirst();                // 调用Move方法, 使删除有效
```

```
// 关闭记录集
```

若从course表中修改一个记录可有下列代码:

```
// 打开记录集
```

```
m_pRecordset->PutCollect("courseno",_variant_t("2112111"));
```

```
m_pRecordset->PutCollect("coursehourse",_variant_t(60));
```

```
...
```

```
m_pRecordset->Update();                    // 使修改有效
```

```
// 关闭记录集
```

特别强调的是, 数据库的表名不能与**ADO**的某些关键字串同名, 例如: **user**等。另外, 通常用**Command**对象执行**SQL**命令来实现数据表记录的查询、添加、更新和删除等操作, 而用**Recordset**对象获取记录集, 用来显示记录内容。

## 8.3.3 Command对象使用

Command对象用来执行SQL命令。下面先来简单介绍SQL几个常用语句。

### 1. SELECT语句

一个典型的SQL查询可以从指定的数据库表中“选择”信息，这时就需要使用SELECT语句来执行。SELECT语句格式如下：

**SELECT** 字段名 **FROM** 表名 [**WHERE**子句] [**ORDER BY**子句]...

它的最简单形式是：

**SELECT \* FROM** *tableName*

其中，星号(\*)用来指定从数据库的tableName表中选择所有的字段(列)。若要从表中选择指定字段的记录，则将星号(\*)用字段列表来代替，多个字段之间用逗号分隔。



# 1. SELECT语句

## 1) WHERE子句

在数据表查询SELECT语句中，经常还需要使用WHERE子句来设定查询的条件。它的一般形式如下：

**SELECT** *column1, column2,...* **FROM** *tableName* **WHERE** *condition*

WHERE子句中的条件可以<(小于)、>(大于)、<=(小于等于)、>=(大于等于)、=(等于)、<>(不等于)和LIKE等运算符。其中，LIKE用于匹配条件的查询，它可以使用“%”和“\_(下划线)”等通配符，“%”表示可以出现0个或多个字符，“\_”表示该

位置处只能出现1个字符。例如：

**SELECT \* FROM Score WHERE studentno LIKE '21%'**

则将Score表中所有学号以21打头的记录查询出来。注意，LIKE后面的字符串是以单引号来标识。再如：

**SELECT \* FROM Score WHERE studentno LIKE '210105\_\_'**

则将Score表中所有学号以210105打头的，且学号为8位的记录查询出来。

WHERE子句中的条件还可用AND(与)、OR(或)以及NOT(非)运算符来构造复合条件查询，例如：若查询Score表中成绩(score)在70分到80分之间的记录，则可有如下语句：

**SELECT \* FROM Score WHERE score<=80 AND score>=70**

# 1. SELECT语句

## 2) ORDER BY子句

在数据表查询SELECT语句中，若将查询到的记录进行排序，则可使用ORDER BY子句。如下面的形式：

```
SELECT column1, column2,... FROM tableName [WHERE condition]  
ORDER BY col1, col2,... ASC | DESC
```

其中，ASC表示升序(从低到高)，DESC表示降序(从高到低)，col1、col2、...分别用来指定是按什么字段来排序。当指定多个字段时，则先按col1排序，当有相同col1的记录时，则相同的记录按col2排序，依此类推。

## 8.3.3 Command对象使用

### 2. INSERT语句

INSERT语句是用来向表中插入一个新的记录。该语句的常用形式是：

**INSERT INTO** *tableName*(*col1,col2,col3,...,colN*)  
**VALUES** (*val1,val2,val3,...valN*)

其中，**tableName**用来指定插入新记录的数据表，**tableName**后跟一对圆括号，包含一个以逗号分隔的列(字段)名的列表，**VALUES**后面的圆括号内是一个以逗号分隔的值列表，它与**tableName**后面的列名列表是一一对应的。需要说明的是，若某个记录的某个字段值是字符串，则需要用单引号来括起来。例如：

**INSERT INTO Student(studentno,studentname) VALUES ('21010503','张小峰')**

将在Student中插入一个新行，其中studentno(学号)为“21010503”，studentname(学生姓名)为“张小峰”，对于该记录的其它字段值，由于没有指定相应的值，其结果

由系统决定。

## 8.3.3 Command对象使用

### 3. UPDATE语句

UPDATE语句用于更新表中的数据。该语句的常用形式是：

```
UPDATE tableName SET column1=value1, column2=value2,...,columnN=valueN  
WHERE condition
```

该语句可以更新*tableName*表中一行记录或多行记录的数据，这取决于WHERE后面的条件。关键字SET后面是以逗号分隔的“列名/值”列表。例如：

```
UPDATE Student SET studentname = '王鹏' WHERE studentno = '21010503'
```

将学号为“21010503”的记录中的studentname字段内容更新为“王鹏”。

### 4. DELETE语句

DELETE语句用来从表中删除记录，其常用形式如下：

```
DELETE FROM tableName WHERE condition
```

该语句可以删除*tableNam*表中一行记录或多行记录，这取决于WHERE后面的条件。

## 8.4 数据库相关的ActiveX控件

在前面的数据库处理中，一次只能显示出一行记录，且修改或添加等操作不能“可视化”地进行。为了弥补MFC的这种不足，在Visual C++ 6.0中允许用户使用一些ActiveX控件用来更好地操作数据库，这些控件包括MSFlexGrid、Remote Data、DBGrid等。

### 8.4.1 使用MSFlexGrid控件

Microsoft FlexGrid (MSFlexGrid) 控件可以显示网格数据，也可以对其进行操作。它提供了高度灵活的网格排序、合并和格式设置功能，网格中可以包含字符串和图片。利用MSFlexGrid可以将某个表的所有记录显示。下面以示例的形式来说明其使用过程。

## 8.4.1 使用MSFlexGrid控件

[例Ex\_Grid] 使用RemoteData和DBGrid控件

1. 将控件的类添加到项目中

(1) 用MFC AppWizard创建一个单文档应用程序Ex\_Grid。

(2) 在向导的第2步对话框中，选中“数据库查看使用文件支持”项，单击[Data Source]按钮，弹出“Database Options”对话框，从中选择ODBC的数据源“Datab

ase Example For VC++”。保留其他默认选项，单击[OK]按钮，在弹出的“Select

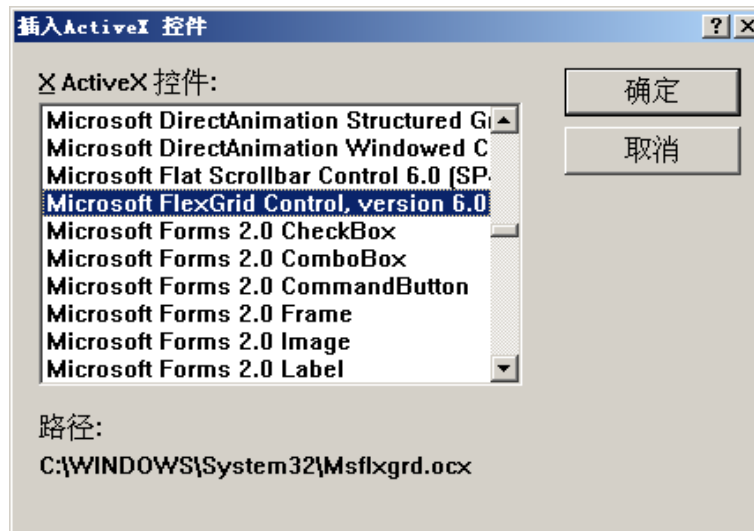
Database Tables”对话框中，选择要使用的表score。

(3) 单击[OK]按钮，又回到了向导的第2步对话框。单击[完成]按钮。开发环境自动打开表单视图CEx\_GridView的对话框资源模板IDD\_EX\_GRID\_FORM。

# 1. 将控件的类添加到项目中

(4) 在打开的表单资源模板中右击鼠标，从弹出的快捷菜单中选择“**Insert Active Control**”命令，出现如图8.31所示“插入 Active 控件”对话框。

(5) 在对话框的控件列表中选择**Micorsoft FlexGrid Control**控件，单击[确定]按钮，该控件就添加到表单资源中，参看图8.33，调整其大小和位置。



# [例Ex\_Grid]

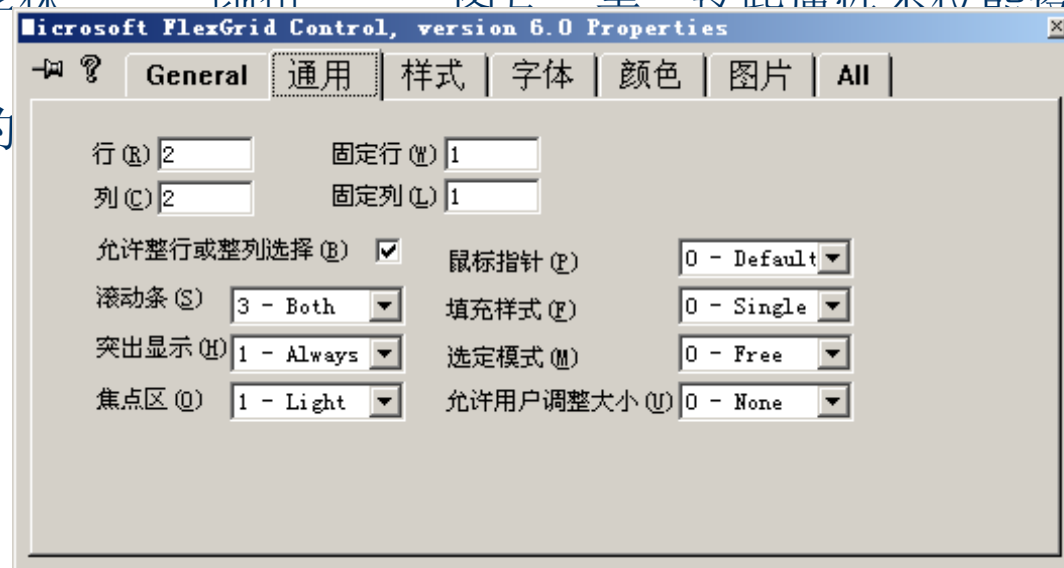
## 2. 修改MSFlexGrid控件属性

右击添加的Microsoft FlexGrid Control控件，从弹出的菜单中选择“属性”或“属性

MSFlexGrid Object”命令均可打开该控件的属性对话框，如图8.32所示。

MSFlexGrid控件的属性要比Visual C++的控件属性要多，如“General”、“通用”、

“样式”、“字体”、“颜色”、“图片”等。此属性不仅能设置控件的字体、颜色，而且能设置网格的





## [例Ex\_Grid]

### 3. 编程控制

- (1) 保留默认的属性及其控件标识符IDC\_MSFLEXGRID1。
- (2) 用MFC ClassWizard在CEx\_GridView类中为刚才添加的MSFlexGrid控件设置一个相关的控件变量m\_MSFGrid。需要说明的是，在此步骤中会出现一些对话框，用于询问是否要添加相关控件的类代码等，选择[是]。
- (3) 在CEx\_GridView类的OnInitialUpdate函数中添加下列代码：

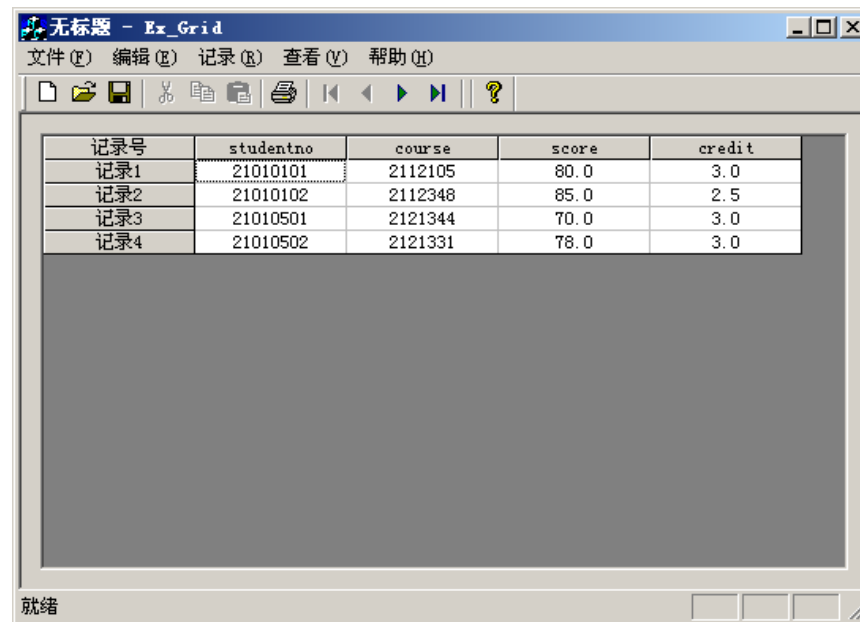
```
void CEx_GridView::OnInitialUpdate()
{
    ...
    while (!m_pSet->IsEOF()) m_pSet->MoveNext();
    m_pSet->MoveFirst();
    m_MSFGrid.SetCols(m_pSet->m_nFields+1 ); // 根据字段数设置单元格最大列数
    m_MSFGrid.SetRows(m_pSet->GetRecordCount()+1); // 根据记录数设置单元格最大行数
    m_MSFGrid.SetColWidth(-1,1440);
    // 将所有的单元格都设为相同的列宽。-1表示所有的列，列宽单位为一个点的
    // 1/20(一个点是1/72英寸)，也就是说，1440刚好为1英寸。
```

```
// 定义单元格的表头
m_MSFGGrid.SetRow(0);    m_MSFGGrid.SetCol(0);        // 定位到(0,0)单元格
m_MSFGGrid.SetText("记录号");        // 设置其显示内容
m_MSFGGrid.SetCellAlignment(4);    // 设置单元格对齐方式，4表示水平和垂直居中
CDBCFieldInfo field;
for (UINT i=0; i<m_pSet->m_nFields; i++)
{
    m_MSFGGrid.SetRow(0);        m_MSFGGrid.SetCol(i+1);
    m_pSet->GetODBCFieldInfo(i,field);        // 获取指定字段信息
    m_MSFGGrid.SetText(field.m_strName);
    m_MSFGGrid.SetCellAlignment(4);
}
int iRow=1;
while (!m_pSet->IsEOF())
{    // 将表的记录内容显示在单元格中
    CString str;
    str.Format("记录%d",iRow);
    m_MSFGGrid.SetRow(iRow);    m_MSFGGrid.SetCol(0);
    m_MSFGGrid.SetText(str);
    m_MSFGGrid.SetCellAlignment(4);
    for (UINT i=0; i<m_pSet->m_nFields; i++)
```

```
{
    m_MSFGGrid.SetRow(iRow);  m_MSFGGrid.SetCol(i+1);
    m_pSet->GetFieldValue(i, str);      // 获取指定字段值，并自动
                                         转换成字符串
    m_MSFGGrid.SetText(str);
    m_MSFGGrid.SetCellAlignment(4);
}
iRow++;
m_pSet->MoveNext();
}
m_MSFGGrid.SetRow(1);          m_MSFGGrid.SetCol(1);
m_pSet->MoveFirst();
}
```

### 3. 编程控制

4. 编译运行，结果如图8.33所示。



The screenshot shows a window titled '无标题 - Ex Grid'. The menu bar includes '文件 (F)', '编辑 (E)', '记录 (R)', '查看 (V)', and '帮助 (H)'. The toolbar contains icons for file operations (new, open, save, print), editing (cut, copy, paste), and navigation (back, forward, home, end, search). The main area displays a table with the following data:

记录号	studentno	course	score	credit
记录1	21010101	2112105	80.0	3.0
记录2	21010102	2112348	85.0	2.5
记录3	21010501	2121344	70.0	3.0
记录4	21010502	2121331	78.0	3.0

The status bar at the bottom left indicates '就绪' (Ready).

## 8.4.2 RemoteData和DBGrid控件

**MSFlexGrid**控件提供界面友好的网格，使表的记录内容能全部地显示出来，但却没有表处理的常用功能，如添加记录、修改记录和删除记录等。而**DBGrid**控件不仅能自动全部显示表的记录内容，而且常见的记录操作也都能很好地支持。需要注意的是，**DBGrid**控件还必须用**RemoteData**控件来提供数据源，但它最大的好处是不需要任何程序代码就能实现表的处理。

通过被绑定的控件提供对存储在远程 **ODBC** 数据源中数据的存取。**RemoteData**控件允许在某一记录集的行与行之间移动，且允许显示和操作来自于被绑定的控件各行里的数据。

**RemoteData**控件在远程数据对象(**RDO**)和数据识别的被绑定的控件之间提供了接口。通过**RemoteData**控件，能够：

- 建立起与基于其本身属性的数据源的连接。
- 创建**RDO**的结果集。
- 把当前行的数据传送给相应被绑定的控件。
- 允许对当前行指针进行定位。
- 将对被绑定的控件所做的任何更改反传给数据源。

## 8.4.2 RemoteData和DBGrid控件

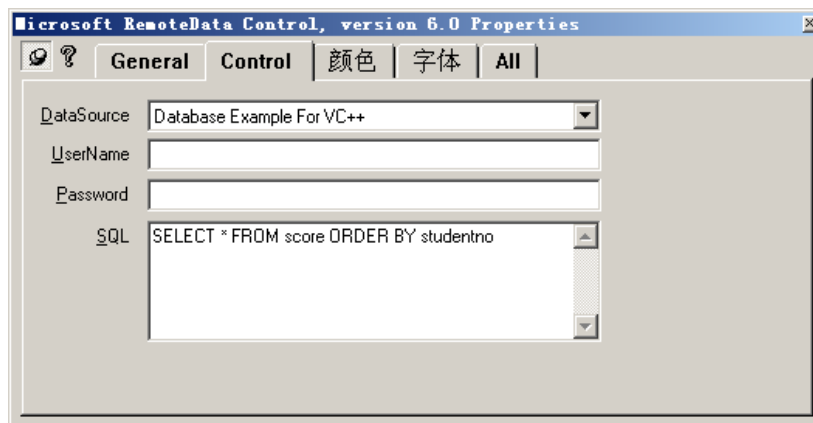
[例Ex\_DBCtrl] 使用RemoteData和DBGrid控件

- (1) 用MFC AppWizard创建一个默认的单文档应用程序Ex\_DBCtrl，但在向导的第6步将CEx\_DBCtrlView的基类由默认的CView选择为CFormView类。
- (2) 在打开的表单资源模板中右击鼠标，从弹出的快捷菜单中选择“Insert Active Control”命令，出现“插入 Active 控件”对话框。
- (3) 在对话框的控件列表中选择Microsoft RemoteData Control，单击[确定]按钮，RemoteData控件就添加到表单资源中，调整其大小和位置。
- (4) 右击该控件，从弹出的菜单中选择“属性”或“Properties RemoteDataCtl Object”命令，打开该控件的属性对话框(参看图8.34)。
- (5) 在“Control(控件)”页面中，从“DataSource”的下拉列表中选择所需要的数据源  
名 “Database Example For VC++”。

## [例Ex\_DBCtrl]

(6) 在“SQL”编辑框中键入SQL操作语句“**SELECT \* FROM score ORDER BY studentno**”是检索学生课程成绩表score的所有记录，并按学号排序。设置的结果如图8.34所示。

(7) 将RemoteData控件属性对话框切换到**All**页面，单击**CursorDriver**选项，在右侧的组合框中将其属性选择“**1-ODBC cursor**”。结果如图8.35所示。

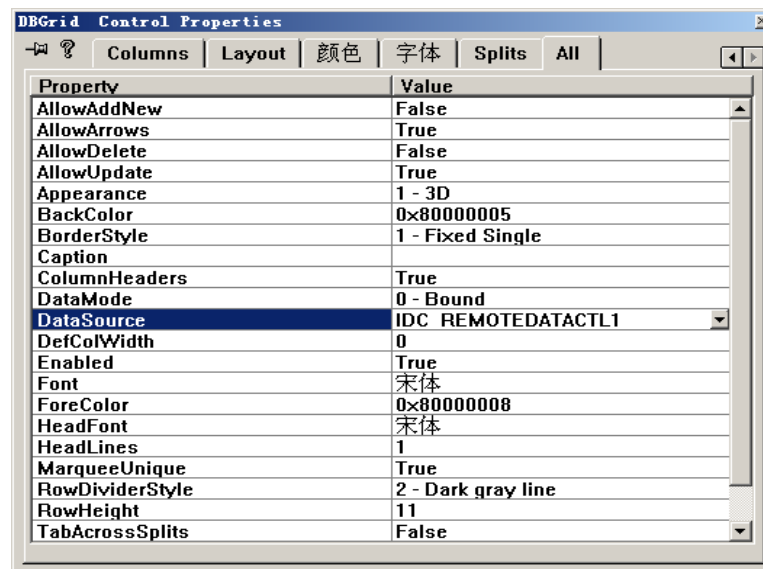


## [例Ex\_DBCtrl]

(8) 再次右击表单资源模板，从弹出的快捷菜单中选择“**Insert Active Control**”命令，在弹出的“插入 Active 控件”对话框中找到要添加的**DBGrid**控件，单击[确定]

按钮。

(9) 参看图8.38，调整添加的DBGrid控件的大小和位置，打开该控件的属性对话框，将数据源(**DataRource**)设置为RemoteData控件**IDC\_REMOTEDATACTL1**，如图8.36所示。

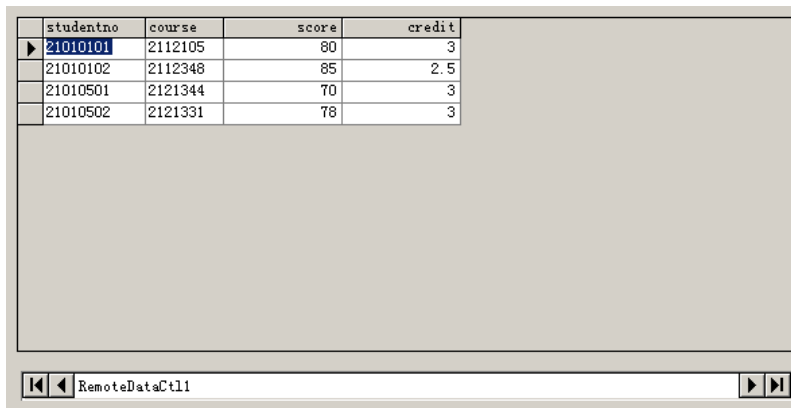




## [例Ex\_DBCtrl]

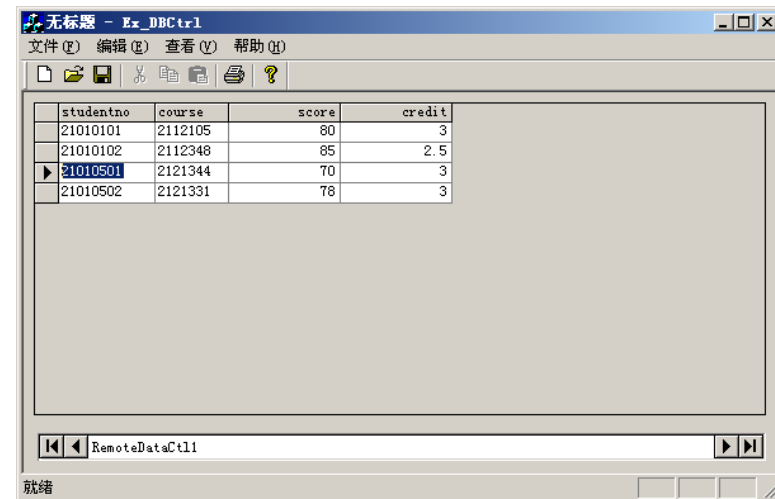
(10) 在对话框编辑器的控件布局栏上，单击测试工具按钮(), 结果如图8.37所示。按ESC键结束测试。

(11) 编译运行并测试，结果如图8.38。



A screenshot of a dialog box editor. It features a table with four columns: studentno, course, score, and credit. The table contains five rows of data. The first row is selected, and the studentno '21010101' is highlighted. Below the table, there is a status bar with the text 'RemoteDataCtrl1' and navigation buttons.

studentno	course	score	credit
21010101	2112105	80	3
21010102	2112348	85	2.5
21010501	2121344	70	3
21010502	2121331	78	3



A screenshot of a running application window titled '无标题 - Ex\_DBCtrl'. The window has a menu bar with '文件(F)', '编辑(E)', '查看(V)', and '帮助(H)'. Below the menu bar is a toolbar with icons for file operations. The main area contains the same table as in the previous screenshot. The studentno '21010501' is now selected and highlighted. At the bottom, there is a status bar with the text '就绪' (Ready) and 'RemoteDataCtrl1'.

studentno	course	score	credit
21010101	2112105	80	3
21010102	2112348	85	2.5
21010501	2121344	70	3
21010502	2121331	78	3