6.2　图书管理系统的整体介绍

在这一小节里，首先从用户角度将整个图书管理系统介绍一下，给读者一个感性的认识，以方便理解后面介绍的具体实现方法。运行光盘附带的library程序后，将看到一个窗口界面，如图6-2所示。

图 6-2　Library 窗口

在“系统”菜单中有“登录”子菜单项，“登录”子菜单主要完成员工登录。在这儿，只实现最简单的权限管理，只要是员工登录后，就可以进行所有的操作。在“基本资料维护”菜单下，有“用户资料维护”、“读者资料维护”、“图书资料维护”菜单项。这些资料维护分别完成相应基本资料的增、删、改。日常工作的子菜单项有借书、还书和缴纳罚单。借书处理主要是通过输入读者ID、图书条形码，完成 borrow 数据库记录的插入。还书处理通过输入图书条形码，判断是否超期。如果超期，则删除 borrow 数据库记录，同时向 fine 和 history 数据库插入一条记录，否则删除 borrow 数据库记录同时向 history 数据库插入记录。缴纳罚款处理主要是输入读者 ID，将其在 fine 数据库的记录删除同时插入一条记录到 fine_history 数据库中。“查询”菜单中有读者信息查询，主要实现根据读者 ID 寻找读者其他资料和借书罚款资料。

6.3　建立工程

从这一节开始具体阐述其实现步骤。

首先建立新工程，我们使用 MFC App Wizard 生成 library.dsw 工作空间。其步骤如下：

（1）从 Visual C++的 File 菜单中选择 New 命令，然后在 New 对话框中打开 Project 选项卡，出现如图 6-3 的窗口。选择 MFC AppWizard(exe)选项，同时工程名写为 library。点击 OK 按钮。

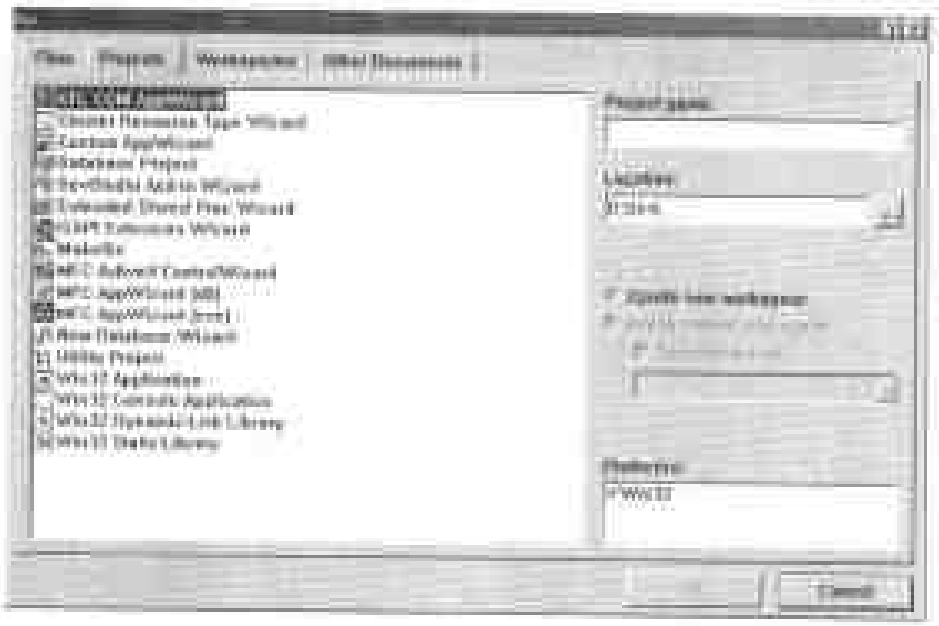

图 6-3 New 对话框

（2）接着出现下面的对话框如图 6-4 所示，选择 Single document 单选按钮，单击 Next 按钮。

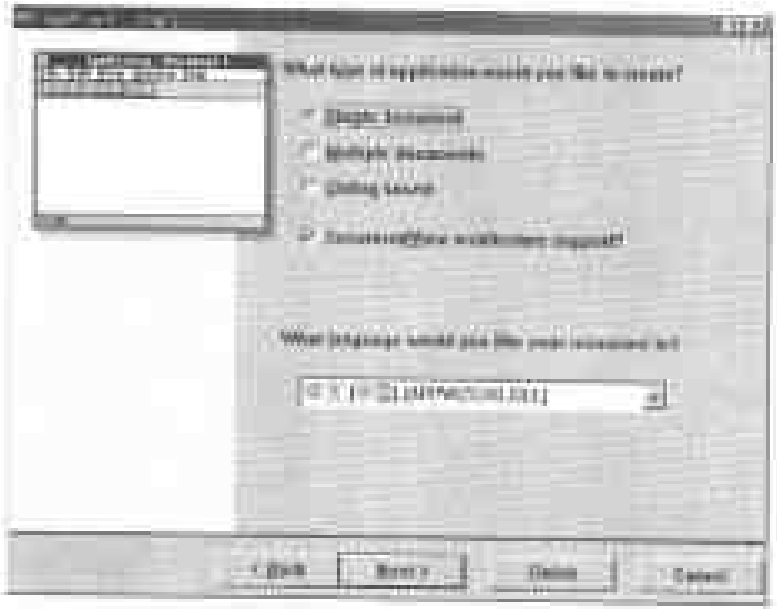

图 6-4 MFC AppWizard 对话框

（3）在后面出现的对话框中，全部按默认设置进行，最后单击 Finish 按钮，出现如所图 6-5 所示的对话框，单击 OK 按钮。

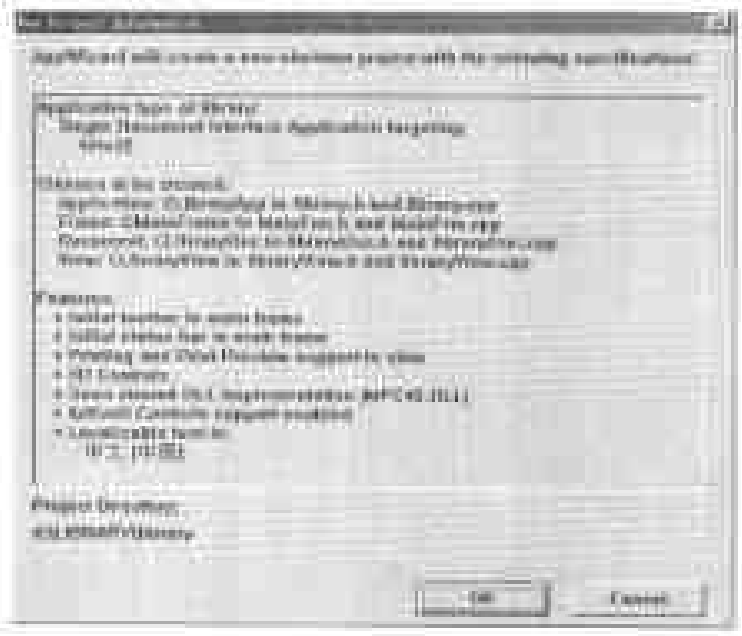

图 6-5　New Project Information 对话框

接着，开始编辑菜单。在 Resource View 的 Menu 文件夹中双击菜单资源菜单资源 IDR_MAINFRAM，打开 Visual C++ Menu Editor(菜单编辑器)，如图 6-6 所示。

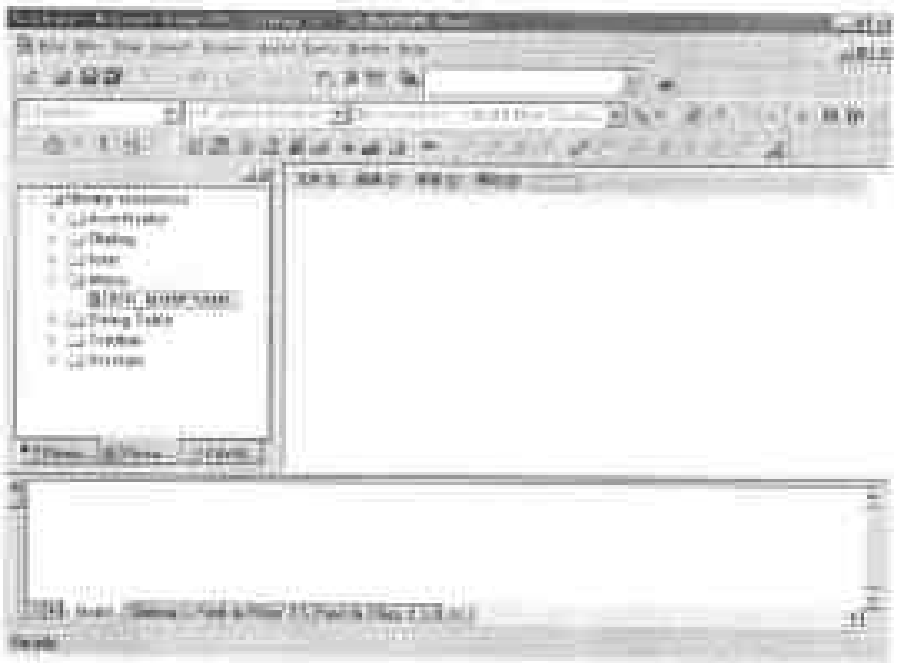

图 6-6　菜单编辑器

建立一个新菜单，将它命名为“系统”，子菜单有两项，分别为“登录”

和“退出”。按上述步骤建立其他菜单项，最后得到如图 6-6 所示的菜单。

在图6-6中，可以看到基本资料的子菜单选项都是灰化的。为什么会这样呢？因为当职员未登录该系统时，所有的操作是不允许的，当然除了登录操作。

说到这里，整个系统的框架结构已经出来了，在下一节，将以登录模块、读者资料维护模块及还书过程模块为例具体说明怎样编写一个简单的模块。

6.4 登录模块的实现

从这一节开始介绍登录模块的实现。在 Resource View 的 Dialog 文件夹中，右击 Dialog 文件夹，选择 Insert Dialog 选项。然后右击产生的新的对话框，选择 Properties，打开如图 6-7 所示的对话框。按图所示修改 ID 和 CAPTION 编辑框，接着关闭对话框。

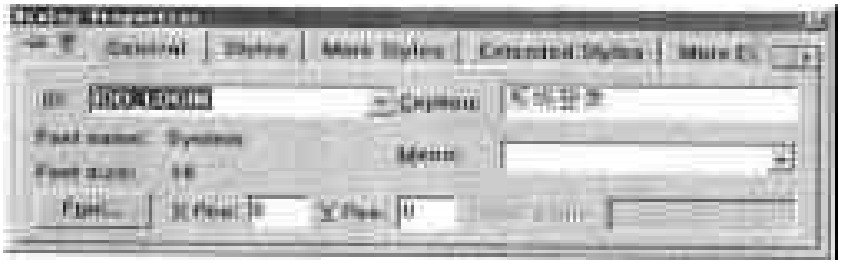

图 6-7 Dialog Properties 对话框

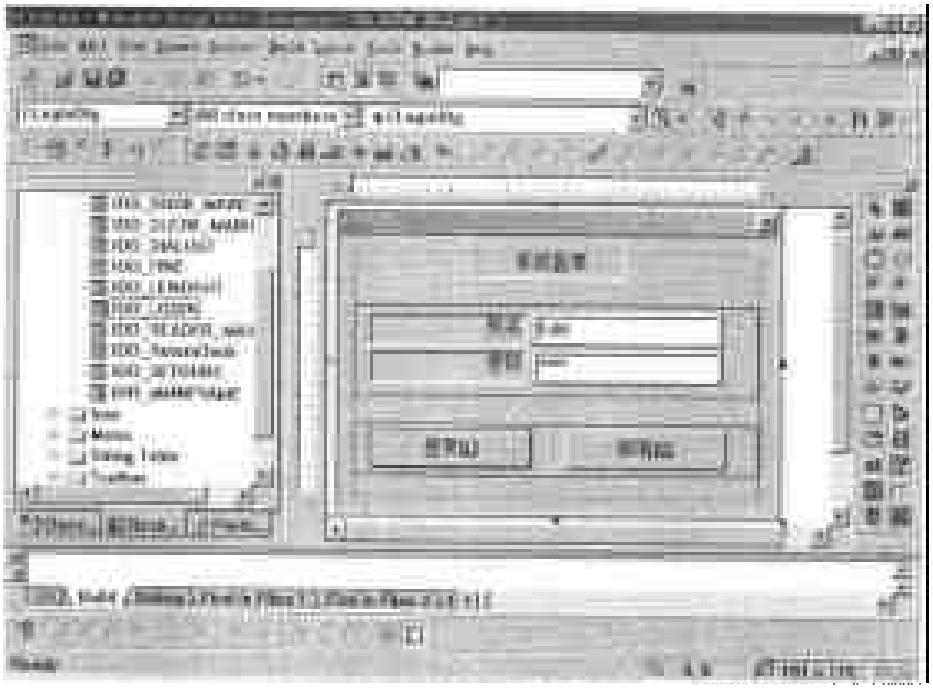

图 6-8 Library-IDD-LDGIN Dialog 对话框

登录模块是要根据用户输入的用户名和密码与数据库 Clerk 中的 Username 和 Password 的比较来判断用户名和密码是否合法。所以，从现在开始讨论在 Visual C++环境下数据库的操作。

Visual C++中包含了编写 Microsoft Windows 环境下 C++数据库应用程序所需的所有控件。实际上，这个产品包含了两个独立的面向用户的数据库访问系统：ODBC(Open Database Connectivity，开放数据库连接性)和 ADO（ActiveX Data Object）。在这个例子中，我们主要介绍通过 ODBC 访问数据库，这种方式比较简单，在后面的例子中，将介绍通过 ADO 来访问数据库。

Microsoft ODBC 标准不仅定义了 SQL 语法的规则，而且定义了 C 语言与 SQL 数据库之间的程序设计接口。现在经过编译的 C 或 C++程序就有可能对任何带有 ODBC 驱动程序的 DBMS 进行访问。ODBC 的 SDK(Software Development Kit，软件开发包），包括 Microsoft Access MDB 数据库、Microsoft Excel XLS 文件、Microsoft Foxpro 文件、ASCII 文本文件和 Microsoft SQL Server 数据库。

其他的数据库公司，包括Oracle，Informix，Progress，Ingres，都提供了基于自己DBMS的ODBC。ODBC实际上就把用户接口同具体的数据库管理过程分离开了。用户不必再专门到同一数据库供应商处购买相应的接口工具了。

MFC 的 ODBC 类对较复杂的 ODBC API 进行了封装，提供了简化的调用接口，从而大大方便了数据库应用程序的开发。程序员不必了解 ODBC API 和 SQL 的具体细节，利用 ODBC 类即可完成对数据库的大部分操作。

MFC 的 ODBC 类主要包括：

- CDatabase 类：主要功能是建立与数据源的连接。
- CRecordset 类：该类代表从数据源选择的一组记录（记录集），程序可以选择数据源中的某个表作为一个记录集，也可以通过对表的查询得到记录集，还可以合并同一数据源中多个表的列到一个记录集中。通过该类可对记录集中的记录进行滚动、修改、增加和删除等操作。
- CRecordView类：提供了一个表单视图与某个记录集直接相连，利用对话框数据交换机制(DDX)在记录集与表单视图的控件之间传输数据。该类支持对记录的浏览和更新，在撤消时会自动关闭与之相联系的记录集。
- CFieldExchange类：支持记录字段数据交换（DFX），即记录集字段数据成员与相应的数据库的表的字段之间的数据交换。该类的功能与CDataExchange类的对话框数据交换功能类似。
- CDBException类：代表ODBC类产生的异常。

概括地讲，CDatabase针对某个数据库，它负责连接数据源；CRecordset针对数据源中的记录集，它负责对记录的操作；CRecordView 负责界面，而

CFieldExchange负责CRecordset与数据源的数据交换。

利用AppWizard和ClassWizard，用户可以方便地建立数据库应用程序，但这并不意味着可以对MFC的ODBC类一无所知。读者应注意阅读下面的内容，为学习后面的例子打好基础。

CRecordset类代表一个记录集，该类是MFC的ODBC类中最重要、功能最强大的类。

在多任务操作系统或网络环境中，多个用户可以共享同一个数据源。共享数据的一个主要问题是如何协调各个用户对数据源的修改。例如，当某一个应用改变了数据源中的记录时，其他的连接至该数据源的应用应该如何处理。对于这个问题，基于MFC的ODBC应用程序可以采取几种不同的处理办法，这将由程序采用哪种记录集决定。

记录集主要分为快照(Snapshot) 和动态集(Dynaset)两种，CRecordset类对这两者都支持。这两种记录集的不同表现在它们对其他的应用改变数据源记录采取了不同的处理方法。

快照型记录集提供了对数据的静态视。快照是个很形象的术语，就好像对数据源的某些记录照了一张照片一样。当其他用户改变了记录时（包括修改、添加和删除），快照中的记录不受影响，也就是说，快照不反映其他用户对数据源记录的改变，直到调用了CRecordset::Requery重新查询后，快照才会反映变化。对于像产生报告或执行计算这样的不希望中途变动的工作，快照是很有用的。需要指出的是，快照的这种静态特性是相对于别的用户而言的，它会正确反映由本身用户对记录的修改和删除，但对于新添加的记录直到调用Requery后才能反映到快照中。

动态集提供了数据的动态视。当其他用户修改或删除了记录集中的记录时，会在动态集中反映出来：当滚动到修改过的记录时对其所作的修改会立即反映到动态集中，当记录被删除时，MFC 代码会跳过记录集中的删除部分。对于其他用户添加的记录，直到调用 Requery 时，才会在动态集中反映出来。本身应用程序对记录的修改、添加和删除会反映在动态集中。当数据必须是动态的时侯，使用动态集是最适合的。例如，在一个火车票联网售票系统中，显然应该用动态集随时反映出共享数据的变化。

在记录集中滚动，需要有一个标志来指明滚动后的位置（当前位置）。ODBC驱动程序会维护一个光标，用来跟踪记录集的当前记录，可以把光标理解成跟踪记录集位置的一种机制。

光标库(Cursor Library)是处于ODBC驱动程序管理器和驱动程序之间的动态链接库(ODBCCR32.DLL)。光标库的主要功能是支持快照以及为底层驱动程序提

供双向滚动能力，高层次的驱动程序不需要光标库，因为它们是可滚动的。光标库管理快照记录的缓冲区，该缓冲区反映本程序对记录的修改和删除，但不反映其他用户对记录的改变，由此可见，快照实际上相当于当前的光标库缓冲区。

应注意的是，快照是一种静态光标(Static Cursor)。静态光标直到滚动到某个记录才能取得该记录的数据，因此，要保证所有的记录都被快照，可以先滚动到记录集的末尾，然后再滚动到感兴趣的第一个记录上。这样做的缺点是滚动到末尾需要额外的开销，会降低性能。

与快照不同，动态集不用光标库维持的缓冲区来存放记录。实际上，动态集是不使用光标库的，因为光标库会屏蔽掉一些支持动态集的底层驱动程序功能。动态集是一种键集驱动光标(Keyset-Driven Cursor)，当打开一个动态集时，驱动程序保存记录集中每个记录的键，只要光标在动态集中滚动，驱动程序就会通过键来从数据源中检取当前记录，从而保证选取的记录与数据源同步。

从上面的分析中可以看出，快照和动态集有一个共同的特点，那就是在建立记录集后，记录集中的成员就已经确定了，这就是为什么两种记录集都不能反映别的用户添加记录的原因。

CRecordset 类代表一个记录集，用户一般需要用 ClassWizard 创建一个 CRecordset 的派生类。ClassWizard 可以为派生的记录集类创建一批数据成员，这些数据成员与记录的各字段相对应，被称为字段数据成员或域数据成员。例如，对于表 6-1 所示的将在后面例子中使用的数据库表，ClassWizard 会在派生类中加入 6 个域数据成员，如程序 6-4-1 所示。可以看出域数据成员与表中的字段名字类似，且类型匹配。

表6-1　字段类型

字段名称	数据类型
Clerk_ID	TEXT
NAME	TEXT
ID_CARD	TEXT(18)
PASSWORD	TEXT
POST	TEXT

程序6-4-1派生类中的域数据成员。

```
class CClerkDataSet : public CRecordset
{
 public:
```

```
CClerkDataSet(CDatabase* pDatabase = NULL);
DECLARE_DYNAMIC(CClerkDataSet)

// Field/Param Data
//{{AFX_FIELD(CClerkDataSet, Crecordset)
CString m_CLERK_ID;                    //职员ID
CString m_NAME;                        //职员姓名
CString m_ID_CARD;                     //身份证号
CString m_PASSWORD;                    //口令
CString m_OFFICIER;                    //官员
//}}AFX_FIELD

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CClerkDataSet)
public:
virtual CString GetDefaultConnect();    // Default connection string
virtual CString GetDefaultSQL();    // Default SQL for Recordset
virtual void DoFieldExchange(CFieldExchange* pFX);  // RFX support
//}}AFX_VIRTUAL

// Implementation
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif
};
```

域数据成员用来保存某条记录的各个字段，它们是程序与记录之间的缓冲区。域数据成员代表当前记录，当在记录集中滚动到某一记录时，框架自动地把记录的各个字段拷贝到记录集对象的域数据成员中。当用户要修改当前记录或增加新记录时，程序先将各字段的新值放入域数据成员中，然后调用相应的CRecordset成员函数把域数据成员拷贝到数据源中。

不难看出，在记录集与数据源之间有一个数据交换问题。CRecordset类使用“记录域交换”(Record Field Exchange，缩写为 RFX)机制自动地在域数据成员

和数据源之间交换数据。RFX 机制与对话数据交换(DDX)类似，CRecordset 的成员函数 DoFieldExchange 负责数据交换任务，在该函数中调用了一系列 RFX 函数。当用户用 ClassWizard 加入域数据成员时，ClassWizard 会自动在 DoFieldExchange 中建立 RFX。典型 DoFieldExchange 如程序 6-4-2 所示：

程序6-4-2典型的DoFieldExchange函数：

```
void CClerkDataSet::DoFieldExchange(CFieldExchange* pFX)
{
//{{AFX_FIELD_MAP(CClerkDataSet)
pFX->SetFieldType(CFieldExchange::outputColumn);
RFX_Text(pFX, _T("[CLERK_ID]"), m_CLERK_ID);
RFX_Text(pFX, _T("[NAME]"), m_NAME);
RFX_Text(pFX, _T("[ID_CARD]"), m_ID_CARD);
RFX_Text(pFX, _T("[PASSWORD]"), m_PASSWORD);
RFX_Text(pFX, _T("[OFFICIER]"), m_OFFICIER);
//}}AFX_FIELD_MAP
}
```

记录集的建立实际上主要是一个查询过程，SQL 的 SELECT 语句用来查询数据源。在建立记录集时，CRecordset 会根据一些参数构造一个 SELECT 语句来查询数据源，并用查询的结果创建记录集，明白这一点对理解 CRecordset 至关重要，SELECT 语句的句法如下：

```
SELECT rfx-field-list FROM table-name [WHERE m_strFilter]
[ORDER BY m_strSort]
```

其中 table-name 是表名，rfx-field-list 是选择的列(字段)，WHERE 和 ORDER BY 是两个子句，分别用来过滤和排序。下面是 SELECT 语句的一些例子：

```
SELECT Clerk_ID, name FROM Clerk;
SELECT Clerk_ID, name FROM Clerk where Clerk_ID='11111111';
```

其中第一个语句从 Clerk 表中选择 Clerk_ID 和 name 字段，第二个语句从 CLerk 表中选择 Clerk_ID 为 '11111111' 的记录。要注意在 SQL 语句中引用字符串、日期或时间等类型的数据时要用单引号括起来，而数值型数据则不用。

要建立记录集，首先要构造一个 CRecordset 派生类对象，然后调用 Open 成员函数查询数据源中的记录并建立记录集。在 Open 函数中，可能会调用 GetDefaultConnect 和 GetDefaultSQL 函数，函数的声明为：

```
CRecordset( CDatabase* pDatabase = NULL);
```

参数 pDatabase 指向一个 CDatabase 对象，用来获取数据源。如果 pDatabase

为 NULL，则会在 Open 函数中自动构建一个 CDatabase 对象。如果 CDatabase 对象还未与数据源连接，那么在 Open 函数中会建立连接，连接字符串由成员函数 GetDefaultConnect 提供。

```
virtual CString GetDefaultConnect( );
```

该函数返回缺省的连接字符串。Open 函数在必要的时候会调用该函数获取连接字符串，以建立与数据源的连接。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连接字符串。

```
virtual BOOL Open(UINT nOpenType = AFX_DB_USE_DEFAULT_TYPE, LPCTSTR
lpszSQL     = NULL,  DWORD dwOptions = none );
throw( CDBException,  CMemoryException );
```

该函数使用指定的 SQL 语句查询数据源中的记录并按指定的类型和选项建立记录集。参数 nOpenType 说明了记录集的类型，如表 6-2 所示，如果要求的类型驱动程序不支持，则函数将产生一个异常。参数 lpszSQL 是一个 SQL 的 SELECT 语句，或是一个表名。函数用 lpszSQL 来进行查询，如果该参数为 NULL，则函数会调用 GetDefaultSQL 获取缺省的 SQL 语句，参数 dwOptions 可以是一些选项的组合，常用的选项在表 6-3 中列出。若创建成功，则函数返回 TRUE，若函数调用了 CDatabase::Open 且返回 FALSE，则函数返回 FALSE。

表 6-2 记录集的类型

类　型	含　义
AFX_DB_USE_DEFAULT_TYPE	使用缺省值
Crecordset::dynaset	可双向滚动的动态集
Crecordset::snapshot	可双向滚动的快照
CRecordset::dynamic	提供比动态集更好的动态特性，大部分 ODBC 驱动程序不支持这种记录集
CRecordset::forwardOnly	只能前向滚动的只读记录集

表 6-3 创建记录集时的常用选项

选　项	含　义
CRecordset::none	无选项（缺省）
CRecordset::appendOnly	不允许修改和删除记录，但可以添加记录
CRecordset::readOnly	记录集是只读的
CRecordset::skipDeletedRecords	有些数据库（如 FoxPro）在删除记录时并不真删除，而是做个删除标记，在滚动时将跳过这些被删除的记录