Lab 8.1: Building a Database Viewer with DAO

Objectives
After completing this lab, you will be able to:

- Use AppWizard to create a DAO database application.
- Connect controls in a CDaoRecordView to their member variables.
- Add filters for data retrieval.

Prerequisites
You should have completed Chapter 8 and mastered the use of the dialog editor before attempting this lab.

Lab Setup
This demonstration shows what you will accomplish during the lab.

Estimated time to complete this lab: **40 minutes**.

Exercises
The following exercise provides practice working with the concepts and techniques covered in this chapter.

Exercise 1: Building a DAO Database Application

In this exercise, you will implement a complete two-table database viewer application using DAO.

There is no setup for this lab. The completed code for these exercises is in \Labs\C08\Lab01\Xxx, where Xxx is the exercise number.

Exercise 1: Building a DAO Database Application
In this exercise, you will implement a complete two-table database viewer application using DAO.

The database for this lab, Personnel.Mdb, contains two tables: Employee Pay Table and Employee Personal Info Table.

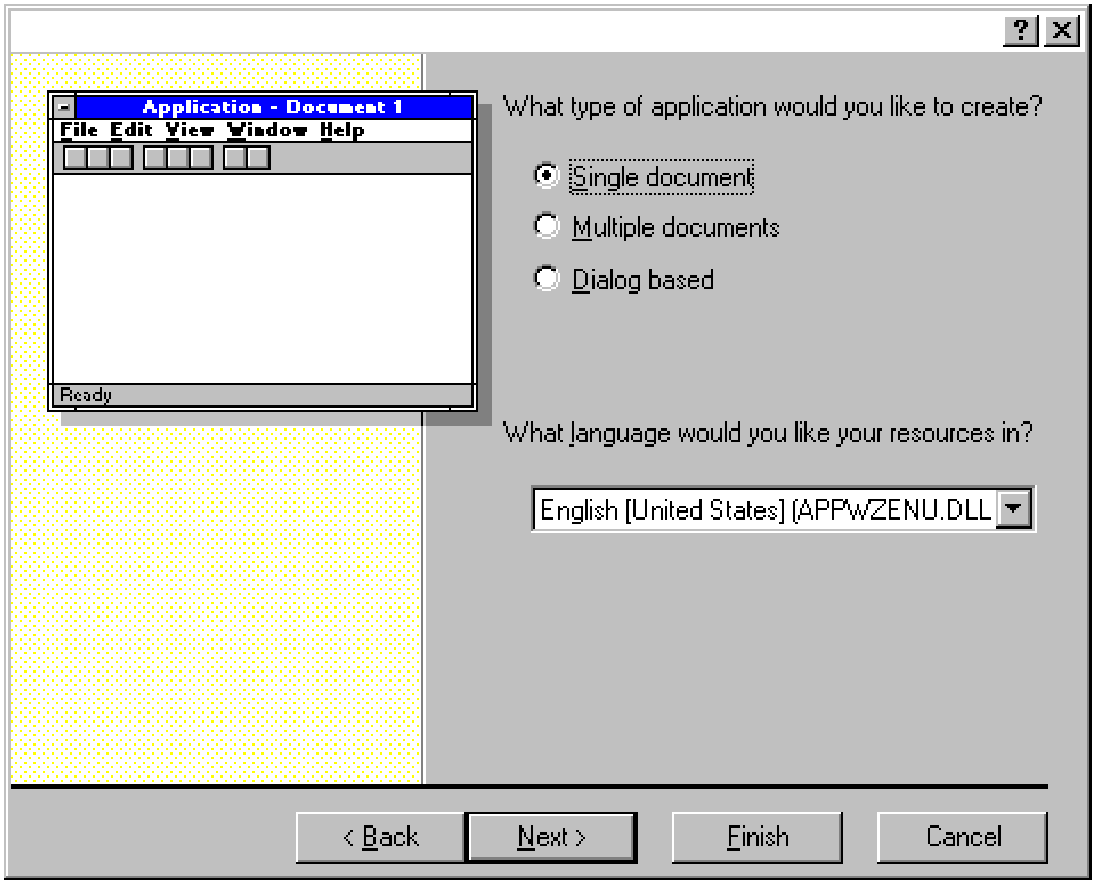
To see the schema for the Employee Pay Table, click this icon.
To see the schema for the Employee Personal Info Table, click this icon.
Employee Number is a common key between the two tables. The SQL query behind this viewer is:

```
Select *
    from [Employee Personal Info Table], [Employee Pay Table]
    where [Employee Personal Info Table].[Employee Number] =
          [Employee Pay Table].[Employee Number]
```

To see what your complete application will look like, click this icon.

➢ **Create a new DAO database application**

1. From the File menu, choose New.
2. Select the Projects tab; choose MFC AppWizard (exe), and type **Join** in the Project Name box. Click OK.
3. In Step 1, choose Single Document Interface.



What type of application would you like to create?

[✓ Single document]

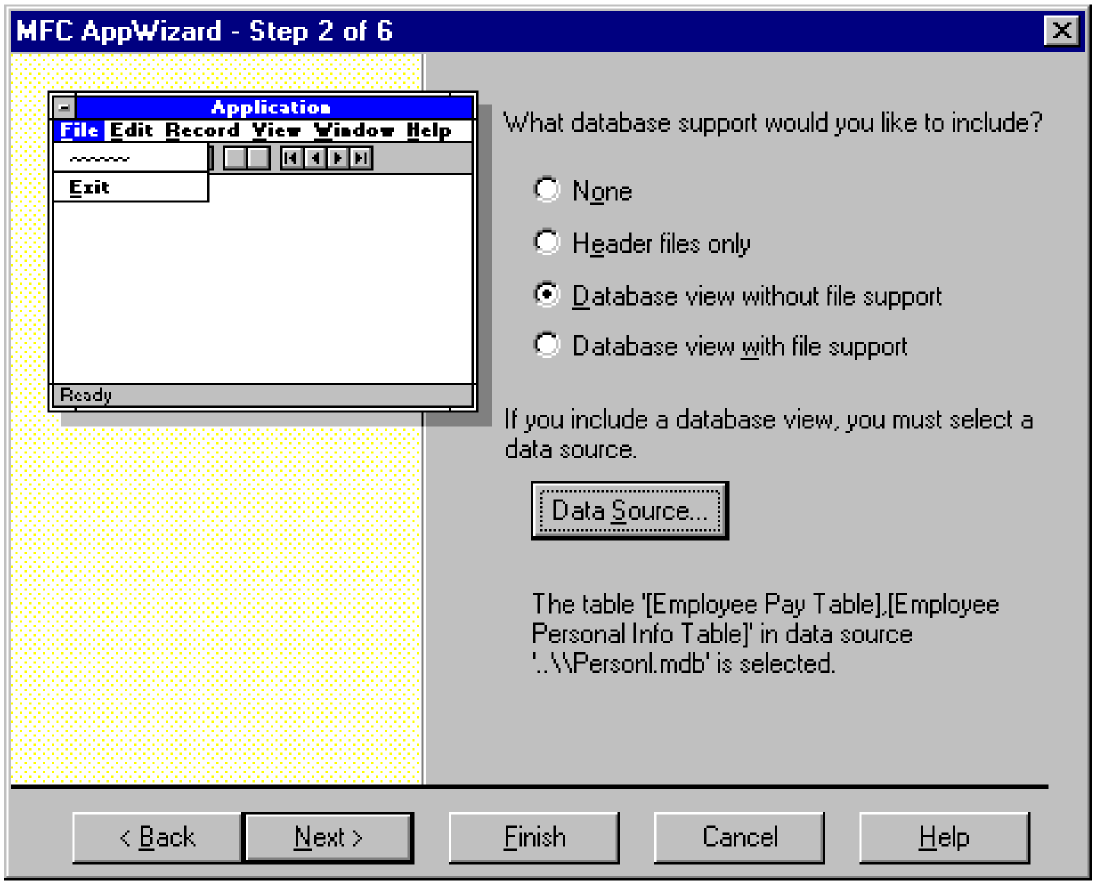
Multiple documents

Dialog based

What language would you like your resources in?

English [United States] (APPWZENU.DLL

4. In Step 2 (the Database Options Page):
   a. Select the “Database view without file support” option.


MFC AppWizard - Step 2 of 6

What database support would you like to include?

- [ ] None
- [ ] Header files only
- [✓] Database view without file support
- [ ] Database view with file support

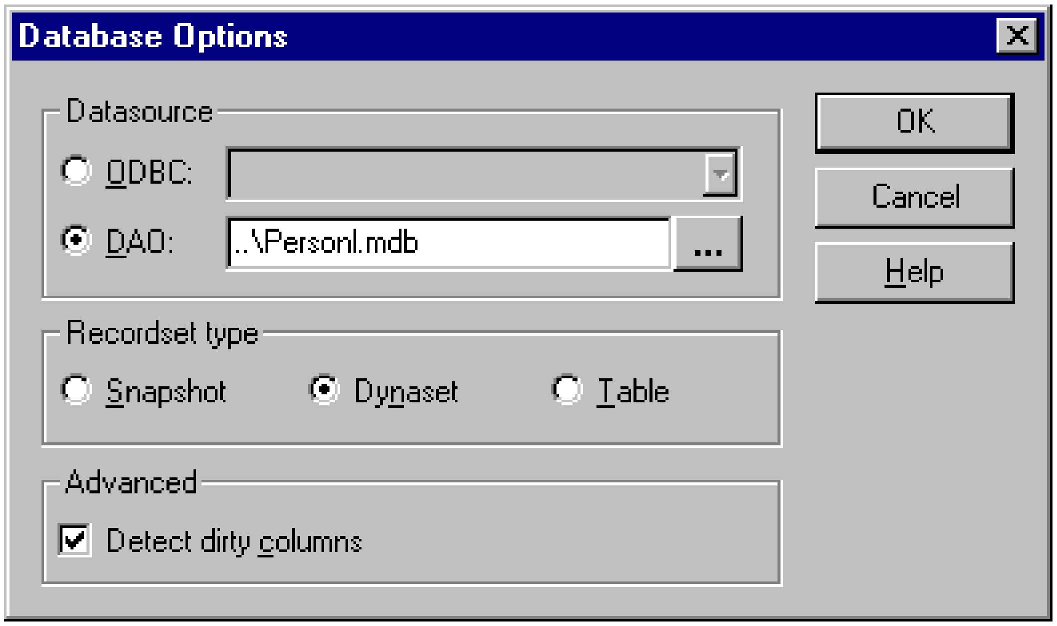
If you include a database view, you must select a data source.

Data Source...

The table '[Employee Pay Table],[Employee Personal Info Table]' in data source '..\\Personl.mdb' is selected.

< Back | Next > | Finish | Cancel | Help

b. Click Data Source. The Database Options page will be displayed. Choose the DAO option. Click Browse to find the Personnel.Mdb database. Accept Dynaset as the recordset type. Leave the Detect dirty columns option selected.


Database Options

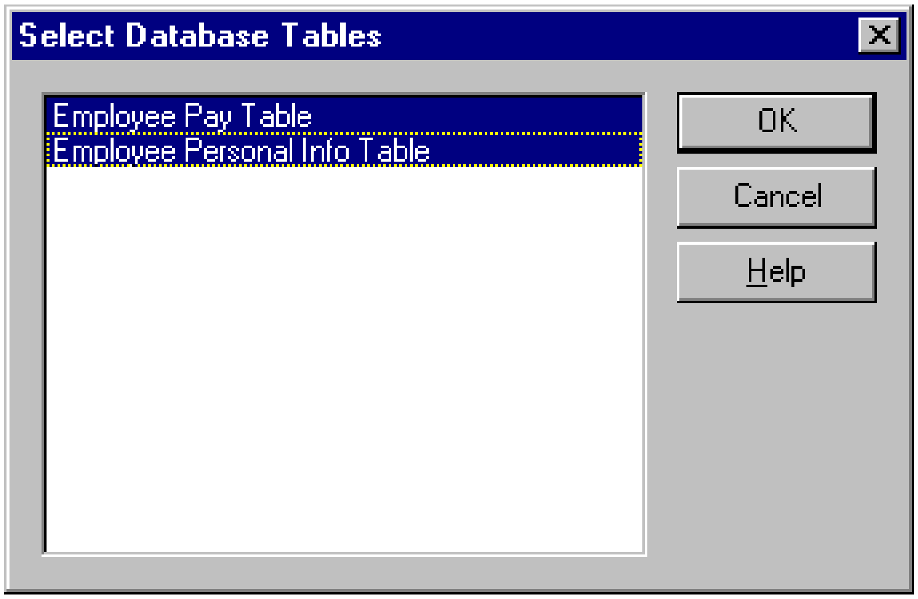
Datasource
- [ ] ODBC:
- [✓] DAO: ..\Personl.mdb

Recordset type: Snapshot [✓ Dynaset] Table

Advanced
- [✓] Detect dirty columns

OK | Cancel | Help

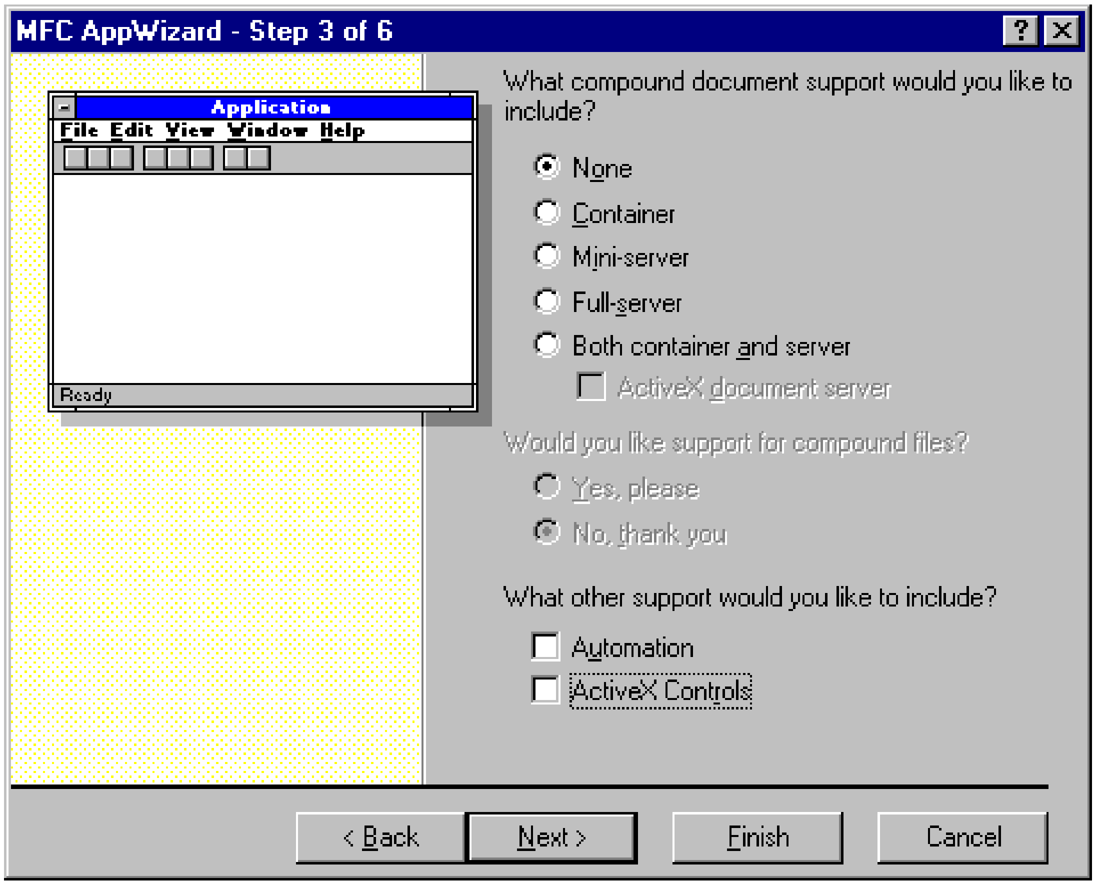
c. When you click OK, the Select Database Tables dialog will be displayed. Select both tables and click OK.


Select Database Tables

Employee Pay Table
Employee Personal Info Table

You will be returned to the Database Options page.

5. In Step 3, clear the ActiveX Controls support check box and click Next.


MFC AppWizard - Step 3 of 6

What compound document support would you like to include?

[✓ None] Container Mini-server Full-server Both container and server

ActiveX document server

Would you like support for compound files?

Yes, please [✓ No, thank you]

What other support would you like to include?

Automation

ActiveX Controls

6. Accept the default settings in Steps 4 and 5.
7. In Step 6, note the base classes for the various classes in the application:

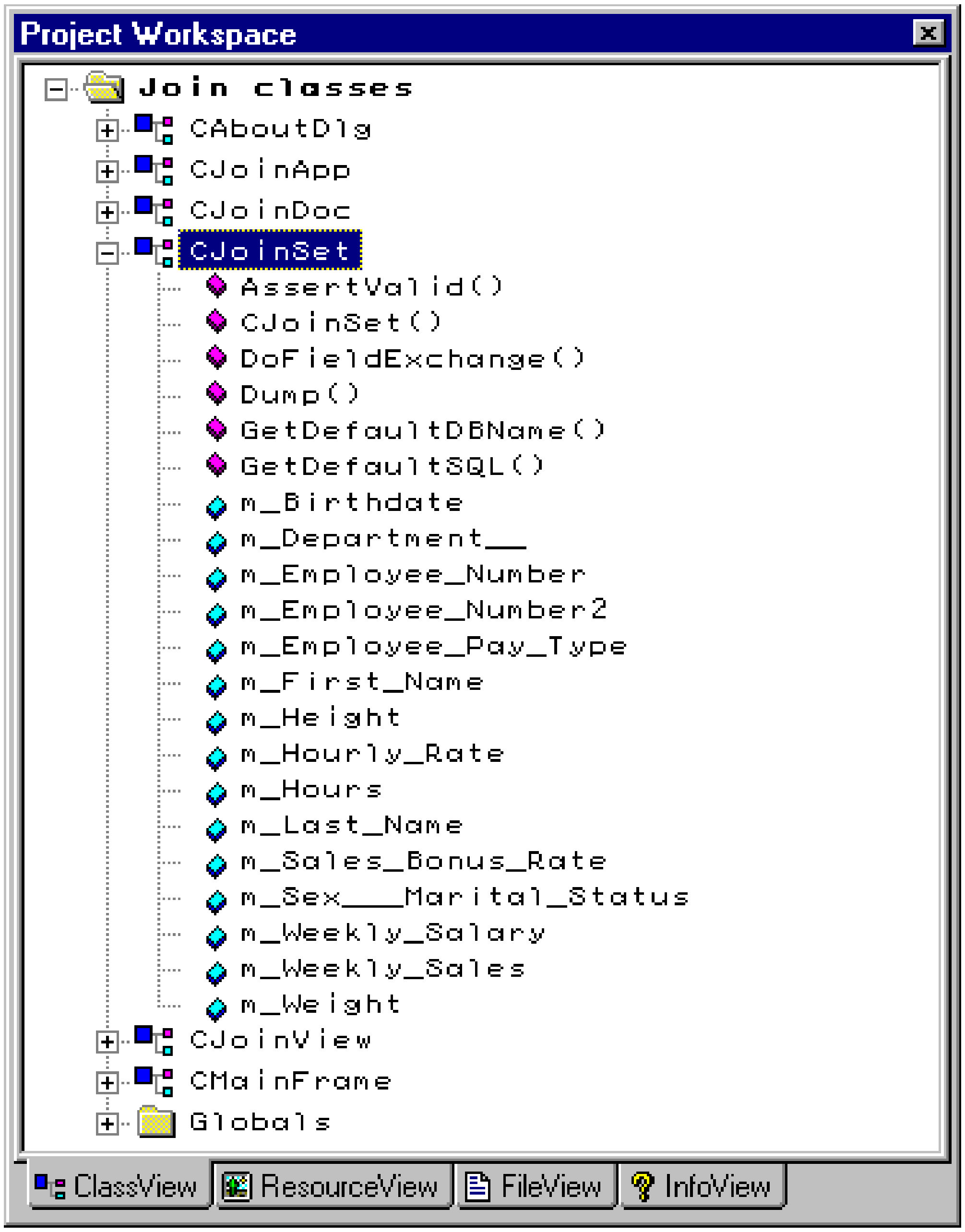
Class	Base Class

CJoinApp	CWinApp
CMainFrame	CFrameWnd
CJoinDoc	CDocument
CJoinView	CDaoRecordView
CJoinSet	CDaoRecordSet

8. Click Finish.
9. In the Project Information dialog box, click OK to create the project.

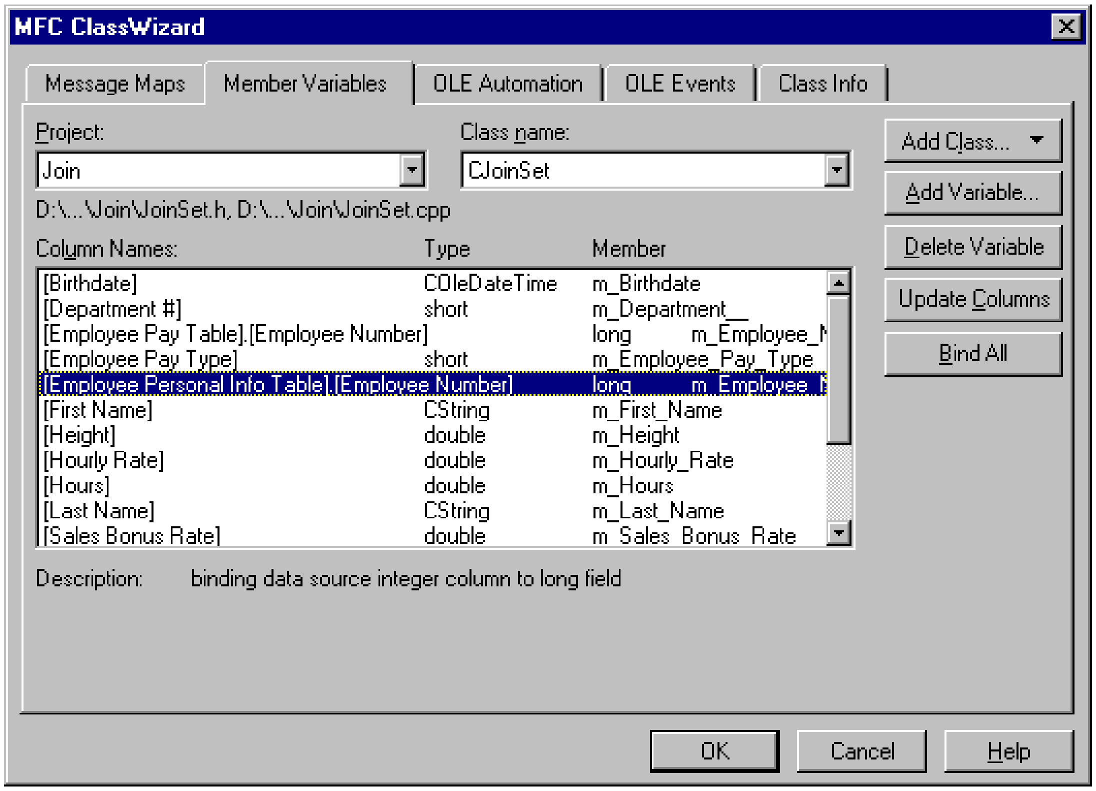
➢ **Examine the Join classes**

1. Expand the Join classes folder in the ClassView pane of the Project Workspace window.
2. Expand the **CJoinSet** class. Note that AppWizard has created instance variables for each of the columns in the two tables.



```
Join classes
    CAboutDlg
    CJoinApp
    CJoinDoc
    CJoinSet
        AssertValid()
        CJoinSet()
        DoFieldExchange()
        Dump()
        GetDefaultDBName()
        GetDefaultSQL()
        m_Birthdate
        m_Department__
        m_Employee_Number
        m_Employee_Number2
        m_Employee_Pay_Type
        m_First_Name
        m_Height
        m_Hourly_Rate
        m_Hours
        m_Last_Name
        m_Sales_Bonus_Rate
        m_Sex___Marital_Status
        m_Weekly_Salary
        m_Weekly_Sales
        m_Weight
    CJoinView
    CMainFrame
    Globals
```

Note There are two fields for the Employee Number column. To see the column bindings, display the Member Variables page of ClassWizard. Choose the **CJoinSet** to display its bindings.



Project: Join

Class name: CJoinSet

D:\...\Join\JoinSet.h, D:\...\Join\JoinSet.cpp

Column Names:	Type	Member
[Birthdate]	COleDateTime	m_Birthdate
[Department #]	short	m_Department__
[Employee Pay Table].[Employee Number]	long	m_Employee_N
[Employee Pay Type]	short	m_Employee_Pay_Type
[Employee Personal Info Table].[Employee Number]	long	m_Employee_
[First Name]	CString	m_First_Name
[Height]	double	m_Height
[Hourly Rate]	double	m_Hourly_Rate
[Hours]	double	m_Hours
[Last Name]	CString	m_Last_Name
[Sales Bonus Rate]	double	m_Sales_Bonus_Rate

Description: binding data source integer column to long field

3. Close ClassWizard and return to the Project Workspace.
4. Double-click the **CJoinDoc** class icon to display JoinDoc.H.

In most applications, the document stores data and serializes it to a file on disk. Often, the application reads the whole file into memory at once, and writes it back to disk as a whole. In a database application, however, the data is stored in the database, and the user usually views the data as records. Such an application does not need a file.

A document in a database application, then, is not normally used for its serialization support. Why does Join have a document class?

The following code, at the beginning of JoinDoc.H, reveals that the role of the document class in Join is to own the recordset.

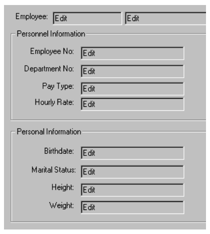
```
class CJoinDoc : public CDocument
{

// Attributes
public:
    CJoinSet m_joinSet;
```

The recordset object, m_joinSet, is embedded in the document object. Therefore, the recordset object is automatically constructed when the document object is constructed, and automatically deleted when the document object is deleted. The document class can own any number of recordset objects in this way

In a sense, the document class is a proxy for the database. If you design your database application to use the document class this way, you can better take advantage of the framework's document/view architecture. For example, if you have multiple views (forms) simultaneously showing some of the contents of the database, you can use **CDocument::UpdateAllViews** to conveniently notify all views about an update that might have been initiated in one of the views.

➢ **Customize the Dialog Template for the Join form**

1. Because **CJoinView** is derived from **CDaoRecordView**, and **CDaoRecordView** is derived from **CFormView**, you will use a dialog template to define the client area. Along with the classes it created, AppWizard created a dialog resource, IDD_JOIN_FORM, for you to lay out. AppWizard places one static text control in the resource, labeled “TO DO: Place form controls on this dialog.” Open IDD_JOIN_FORM by clicking the icon for this dialog in the ResourceView.
2. Delete the static control. You will develop a dialog box similar to the one below:



Type	ID	Caption
Dialog	IDD_JOIN_FORM	
Right aligned text	IDC_STATIC	Employee:
Right aligned text	IDC_STATIC	Employee No:
Right aligned text	IDC_STATIC	Department No:
Right aligned text	IDC_STATIC	Pay Type:
Right aligned text	IDC_STATIC	Hourly Rate:
Right aligned text	IDC_STATIC	Birthdate:
Right aligned text	IDC_STATIC	Marital Status:
Right aligned text	IDC_STATIC	Height:
Right aligned text	IDC_STATIC	Weight:
Group box	IDC_STATIC	Personnel Information
Group box	IDC_STATIC	Personal Information
Edit control	IDC_FNAME	
Edit control	IDC_LNAME	
Edit control	IDC_EMP_NO	
Edit control	IDC_DEPT	
Edit control	IDC_PAY_TYPE	
Edit control	IDC_HOUR_RATE	
Edit control	IDC_BIRTH	
Edit control	IDC_MARITAL_STATUS	
Edit control	IDC_HEIGHT	
Edit control	IDC_WEIGHT	

Note: All edit controls are read-only

3. Save Join.Rc.

➢ Bind controls to member variables

You can use an extension to the dialog editor, or the ClassWizard, to bind controls to member variables according to the following table.

Control	Datatype	Variable
IDC_BIRTH	COleDateTime	m_pSet->m_Birthdate
IDC_DEPT	short	m_pSet->m_Department__
IDC_EMP_NO	long	m_pSet->m_Employee_Number
IDC_FNAME	CString	m_pSet->m_First_Name
IDC_HEIGHT	double	m_pSet->m_Height
IDC_HOUR_RATE	double	m_pSet->m_Hourly_Rate
IDC_LNAME	CString	m_pSet->m_Last_Name
IDC_MARITAL_STATUS	BYTE	m_pSet->m_Sex___Marital_Status
IDC_PAY_TYPE	short	m_pSet->m_Employee_Pay_Type
IDC_WEIGHT	double	m_pSet->m_Weight

1. To bind using the dialog editor, CTRL+double-click the control to display the Add Member Variable dialog.

   To bind using the ClassWizard, display the Member Variables page and browse the Control IDs. Click Add Variable to display the Add Member Variable dialog box.
2. Use the Member variable name combo box to select the variable name. Leave the category as Value, and the Variable type should default to the datatypes listed above.
3. Close the ClassWizard or the Dialog editor and save all files.

➢ Set the query

CDaoRecordSet constructs its query from two of its member variables: The SQL **WHERE** clause is in **CDaoRecordSet.m_strFilter**, and the SQL **ORDER BY** clause is in **CDaoRecordSet.m_strSort**.

1. Open **CJoinSet** to its constructor. At the end of the constructor, set m_strFilter to create the join.

```
m_strFilter = "[Employee Pay Table].[Employee Number] = "
              "[Employee Personal Info Table].[Employee Number]";
```

2. Set m_strSort to create the order clause.

```
m_strSort   = "[Employee Pay Table].[Last Name], "
              "[Employee Pay Table].[First Name]";
```

3. Save JoinSet.Cpp. The complete constructor follows.

```
CJoinSet::CJoinSet(CDaoDatabase* pdb)
    : CDaoRecordset(pdb)
{
    //{{AFX_FIELD_INIT(CJoinSet)
    m_Employee_Number = 0;
    m_Last_Name = _T("");
    m_First_Name = _T("");
    m_Department__ = 0;
    m_Employee_Pay_Type = 0;
    m_Hours = 0.0;
    m_Hourly_Rate = 0.0;
    m_Weekly_Salary = 0.0;
    m_Sales_Bonus_Rate = 0.0;
    m_Weekly_Sales = 0.0;
    m_Employee_Number2 = 0;
    m_Sex___Marital_Status = 0;
    m_Height = 0.0;
    m_Weight = 0.0;
    m_nFields = 15;
    //}}AFX_FIELD_INIT
    m_nDefaultType = dbOpenDynaset;

    m_strFilter = "[Employee Pay Table].[Employee Number] = "\
                  "[Employee Personal Info Table].[Employee Number]";

    m_strSort   = "[Employee Pay Table].[Last Name], "\
                  "[Employee Pay Table].[First Name]";
}
```

➤ **To size the view to the dialog template**

1. Open JoinView.Cpp to **CJoinView::OnInitialUpdate**. To the end of this function, add a call to **CFrameWnd::RecalcLayout** to determine the positions of all the controls, including the dialog frame.

   ```
   GetParentFrame()->RecalcLayout();
   ```

2. Call **CScrollView::ResizeParentToFit** to size the window.

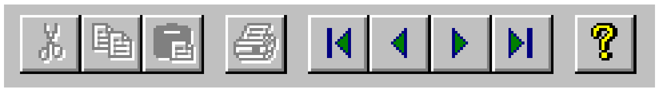
   ```
   ResizeParentToFit(FALSE);
   ```

3. Save JoinView.Cpp. The complete function follows.

   ```
   void CJoinView::OnInitialUpdate()
   {
       m_pSet = &GetDocument()->m_joinSet;
       CDaoRecordView::OnInitialUpdate();
   //add following
       GetParentFrame()->RecalcLayout();
       ResizeParentToFit(FALSE);}
   ```

➤ **Build and run Join.Exe**

1. Save all files and build Join.Exe.
2. Run Join.Exe.
3. Notice the Record menu and its menu items and the corresponding controls on the toolbar. These are all provided by AppWizard.



4. As an additional exercise, change the values of m_strFilter and m_strSort to build different queries into the database.

The completed code for this exercise is in \Labs\C08\Lab01\Ex01.