

ODBC2.WRI

This example shows a number of important techniques you will need to create applications using ODBC. ODBC2 is a data drill down example that constructs crosstab queries on the fly and populates spreadsheets with the results. The sheets have some summary calculations and formatting added.

About the demo:

The first thing to do is run the demo to see what it does and then read the rest of this file to see how it does it. When you run the application, the first sheet is populated from the Location table in the database in the project directory. Double clicking on a cell or column header will generate a new query which is placed on a new sheet. The sheet is formatted and a row and column of data analysis is added. If you return to the Location sheet you can generate a new query. The Data Analysis menu item lets you choose the type of formula that is placed in the right column and the bottom row of the active sheet and all subsequent queries.

How it works:

The DLLs mentioned in the following discussion were obtained through MSVC++ 2.2 and VB 4. Help for these DLLs comes from the ODBC 2.0 book from the Microsoft Press and the associated help files.

The first problem to tackle is how to insulate the user from the complexity of using ODBC databases. The first step is to connect to a database using Formula One's ODBCConnect. For this you need a connect string. When using this string, Formula One will prompt for any information that is not provided by either the data source or the string.

The VB method DBEngine.RegisterDatabase will register the data source name but not the database name or system database name. There are four ways to do this: Let the user figure it out at run time, Write the INI and Registry yourself, configure the data source ahead of time with the ODBC admin 32 program on every machine that will run your program, or use DLL calls as shown in this example. One warning - the VB DBEngine methods are also very slow compared to this example. See Accessing External Databases in the VB4 help (Professional Version) for more info on creating the DataConnectString.

The following shows the VB method. It adds an entry to the ODBC.INI file and registry. It is done one time prior to use. This code can be done every time and only one entry will be made, but this slows load time and destroys the settings for an existing data source that are made with the ODBC admin 32 program (such as database name, system.mda).

```
DBEngine.IniPath = App.Path & "\ODBC2.INI"  
DBEngine.RegisterDatabase DataSourceName, _  
    "Microsoft Access Driver (*.mdb) (32 bit)", _  
    True, "Driver32=C:\WINNT35\System32\odbcjt32.dll"
```

The problem with the above method is that the data source is not completely configured and the user will have to deal with the ODBC dialog. The above code is not used in this example.

For this project we set all information for the data source with SQLConfigDataSource and just tell Formula One to use ODBC and the data source name. Since the data source is completely configured, the user never sees an ODBC dialog.

The following sets up a Data Source name and all the attributes that are necessary to connect to the data source without showing a dialog. The data source is reconfigured each time the program is run but this is pretty fast. As an alternative, you could have an initialization program that runs during setup or check for the existence of the data source on startup.

SQLConfigDataSource requires that the attributes are in null terminated keyword-value pairs with two nulls at the end of the string. Passing Form1.hWnd will generate a dialog for the user to modify the info provided but, if you want the user to do this, you can just use the Formula One generated dialogs to create a new data source and skip all this dll stuff. Note that we are providing our own database and system

database and they are located in the project directory.

```
Let dbDriver = "Microsoft Access Driver (*.mdb)"
Let dbAttributes = "DSN=" & DataSourceName & Chr$(0) _
    & "DBQ=" & App.Path & "\ODBC2.MDB" & Chr$(0) _
    & "SystemDB=" & App.Path & "\system.mda" & Chr$(0) _
    & "DefaultDir=" & App.Path & Chr$(0) _
    & "UID=admin" & Chr$(0) _
    & "PWD=" & Chr$(0) & Chr$(0)

'' If the next statement fails, we can't get to the database without
'' the user going through some hoops, so we exit.
result = SQLConfigDataSource(0, ODBC_ADD_DSN, dbDriver, dbAttributes)
If (False = result) Then
    MsgBox "Establishing DSN failed! Error: " & result
End
End If
```

Now that the data source is created, we are ready to connect Formula One and do a query. The following code implements the Location table fetch. The rest of the queries are created on the fly and require a lot of processing by the database engine. You could speed this up a bit by using predefined queries in your database.

```
Call Fetch(F1Book1, 1, 1, 1, DataConnectString, _
    "Select City, State, Region From Location_Table", True, True, True, True)
```

The Fetch routine (ODBC2.TXT) handles the connect, fetch, and disconnect. If you are doing a few queries, you will want to leave the connection open while you are fetching.

```
With SS
    .Sheet = whichSheet
    .ODBCConnect dsName, True, returnCode
    .ODBCQuery query, startRow, startCol, False, setColNames, _
        setColFormats, setColWidths, setMaxRC, returnCode
    .ODBCDisconnect
End With
```

The DataConnectString is used as dsName in the ODBCConnect. The last four booleans in the call to Fetch set ColNames, ColFormats, ColWidths, and the MaxRC.

Instead of on form load, the above code is executed in a timer so the form can be refreshed, Formula One initialized, and the load time doesn't look so long. Two things take a while to do - registering the database and getting an open connection to it. After that everything is quite speedy. You will probably want to hide this activity in a timer also. The user will probably stare at the screen for three or four seconds before starting anyway. Put up a help screen or introduction to keep them busy.

There are additional comments in the form's code that tell how the project and the queries were developed.