

# Simple VB6 Demonstration Application

## Quickstart Guide

This Guide is designed to help you build your own VB6 database applications using Westfaro Corporation's VB6 Scripts as the starting point.

## A preview

Figure 1 shows a form that can be used to maintain the Products table in the NorthWind database. Note that the dropdowns show foreign key values instead of descriptions. If you tell Kickstart which column(s) to use that provide a user friendly interpretation, then it will generate code using that column.

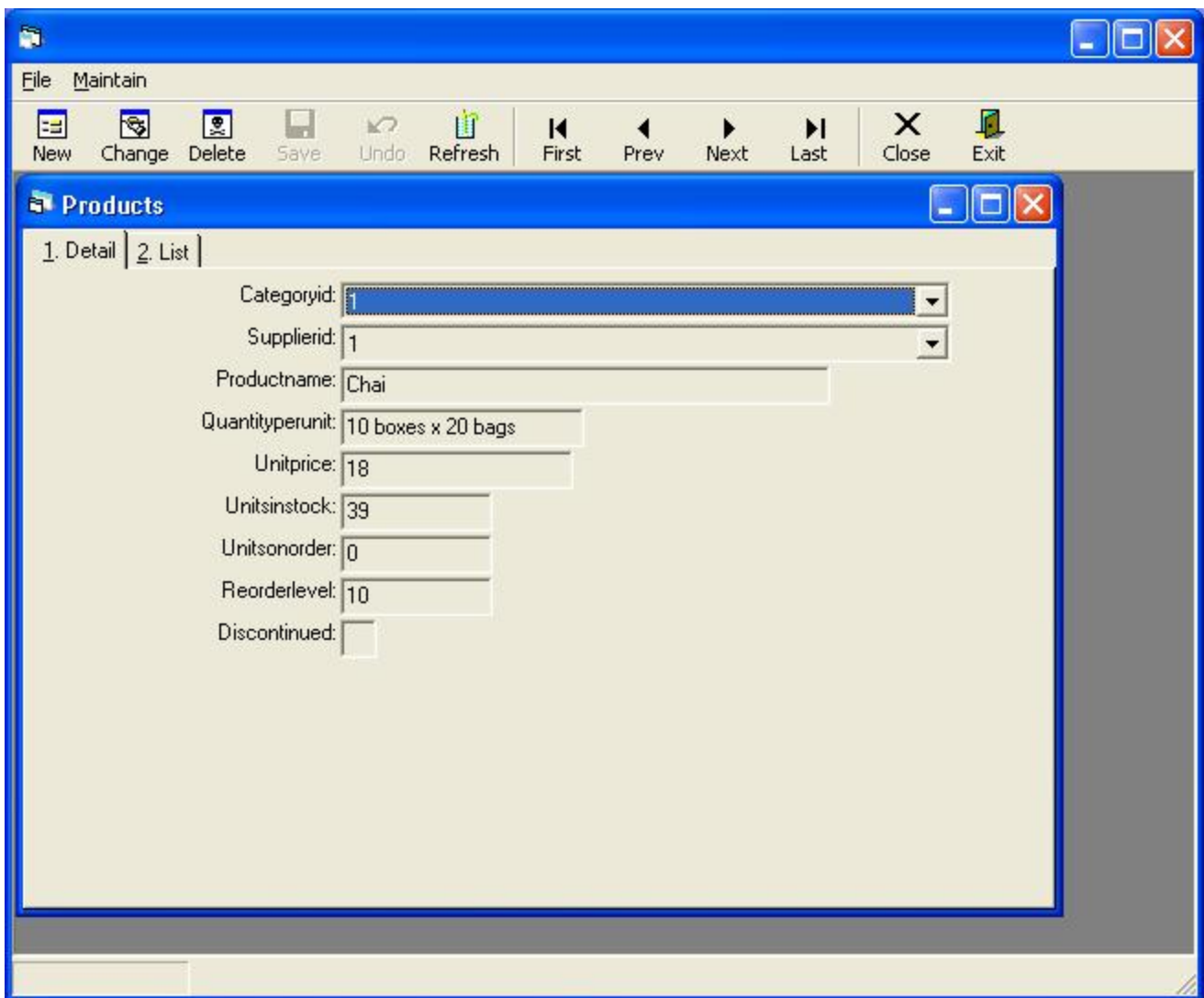


Figure 1 – Products Maintenance

Clicking on the List tab will show a grid view of the table. By default, it shows all rows and columns. You would probably want to customize the generated code to show the columns that are most often used

to look up products. You might also set a filter that limits the rows displayed to a particular subset based on user specified criteria. The grid view is shown in Figure 2.

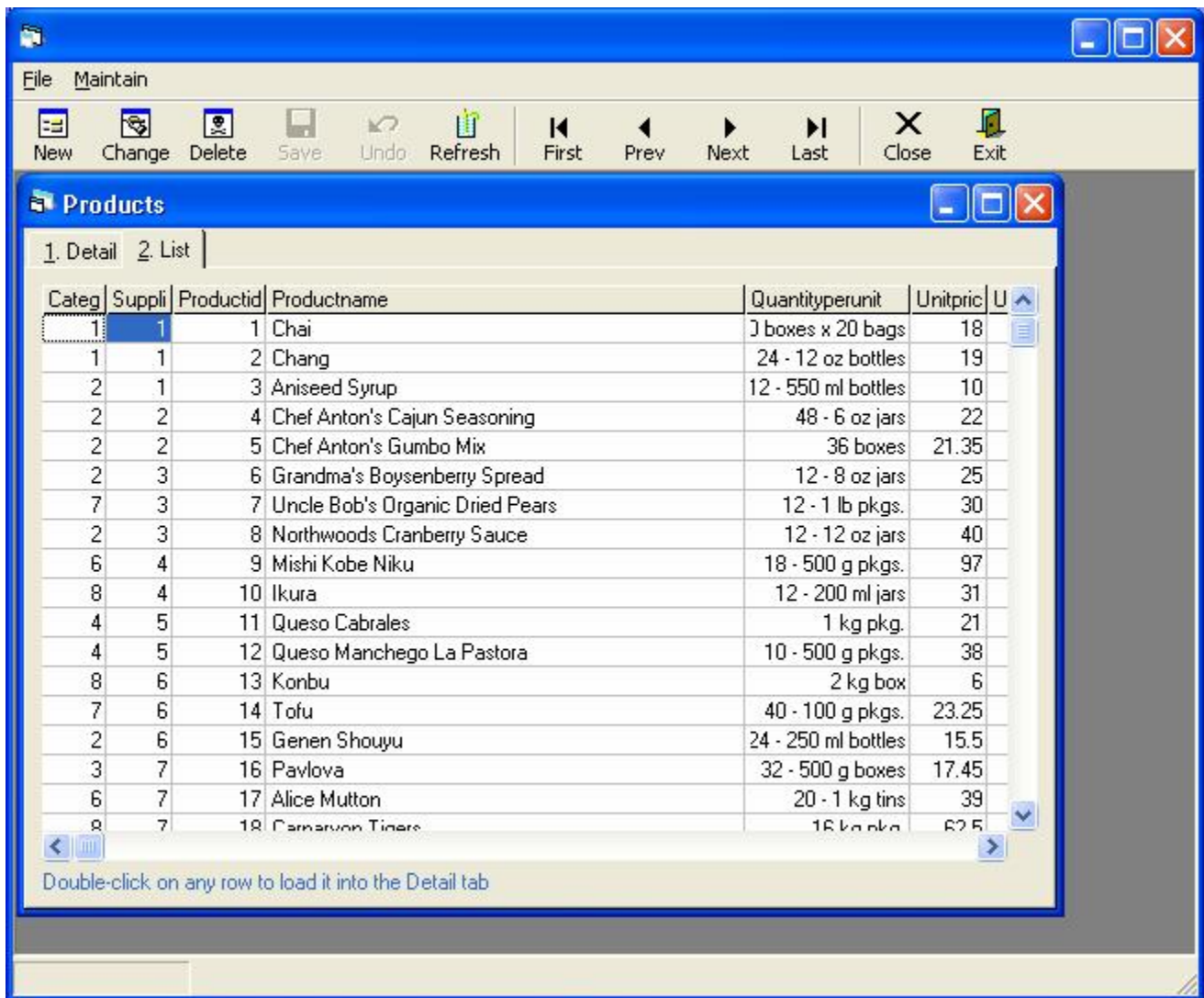


Figure 2 – Grid View of Products table

If you double-click a row, the first tab will be populated with the columns from that row.

### Creating the application

1. Download the SimpleVB01.zip file and extract the scripts (.ksc extension).
2. Connect to the Northwind database using Kickstart.
3. Select the VB Script IncludeAll.ksc
4. Select a destination folder.

5. Click the Generate button
6. Extract the Images folder from SimpleVB01.zip and create an Images subfolder in the destination folder to hold the images.
7. Start VB6 and open the project Simple.vbp.
8. Compile the program and save the compiled program in the destination folder
9. Run the compiled program
10. Experiment with editing, adding and deleting rows
11. Repeat for your own databases

### ***Next steps***

The raw generated code could obviously do with some cleaning up. The biggest problem is the way it displays foreign keys (lookups). Since the Database doesn't know which columns in the **Categories** and **Suppliers** tables should be displayed to the user, Kickstart uses the referenced column by default. However, you can use Kickstart to choose which columns should be displayed in a foreign key relationship. For **Categories**, you might choose **Categoryname** or **Description** or both.

A second problem is that the application shows every column on both tabs. In the case of the List tab, you will probably want to reduce the number of columns to those most useful for searching.

A third problem is that labels are taken directly from Database column names. You will probably want to change a few of them. If you use the convention of using meaningful column names with the component words separated by underscores, then Kickstart will display them better.

### ***Customizing the generated code***

Some application generators attempt to generate working applications but they are often difficult to customize. Others generate source code only a computer could like.

The thinking behind Kickstart is different. Lots of database and application code is repetitive. Programmers develop a template that works for one table and then cut and paste that code for the next table. Kickstart automates that grunt work. Write your best code for one table, convert that code into a Kickstart script and you've written the code for every table. When you have a large database with dozens or hundreds of tables, being able to replicate your best database code at the touch of a button is awesome.

Appendix 1 shows the VB6 code that Kickstart created for one database table using the demonstration VB script:

## Appendix 1 – VB6 Code created using demonstration VB script

```
Option Explicit
'
' Declare an mdiParent WithEvents so we can pick up the
' custom ButtonClick events that it raises.
Private WithEvents m_mdiParent As mdiParent
'
' Declare the record set that is used for communication with
' the back-end database that stores the Products table.
Private m_resProducts As ADODB.Recordset
'
' Declare PK(s) so we can navigate back to a record on refresh
Private m_lngProductID_PK As Long
'
' Declare the concurrency columns
'
' Declare the UDT arrays used to hold the values corresponding to the
' foreign key descriptions displayed in Combo boxes.
Private Type udtCategoriesProducts
    lngCategoryID As Long
End Type
Private m_uCategoriesProducts() As udtCategoriesProducts
'
' Declare index to save item selected from cboCategoriesProducts
Private m_lngCategoriesProductsIndex As Long
'
Private Type udtSuppliersProducts
    lngSupplierID As Long
End Type
Private m_uSuppliersProducts() As udtSuppliersProducts
'
' Declare index to save item selected from cboSuppliersProducts
Private m_lngSuppliersProductsIndex As Long
'
' Declare height and width of form
Private m_lngHeight As Long
Private m_lngWidth As Long
'
' Switch to indicate New as opposed to Change
Private m_blnNew As Boolean
'
' Switch to control list Grid loading
Private m_blnGridLoaded As Boolean
'
' Status variable to track state of Form
Private m_eFormStatus As enumFormState
Private Sub Form_Activate()
'
' Since this form is now active, we want to receive the
' custom ButtonClick events broadcast by the mdiParent.
Set m_mdiParent = mdiParent
mdiParent.SetToolBarState m_eFormStatus
End Sub
Private Sub Form_Deactivate()
'
' Since this form is no longer active, we don't want to receive
' the custom ButtonClick events broadcast by the mdiParent.
Set m_mdiParent = Nothing
End Sub
Private Sub Form_Initialize()

    m_eFormStatus = FORM_READY_VIEW

End Sub
Private Sub Form_Load()
    Dim strSQL As String

    SystemIsWorking
'
```

```

' Establish the initial size of the form and override VB's odd
' default choice for MDI children.
modPublic.FindTrueFormSize Me, m_lngWidth, m_lngHeight
'
' Load the Combo boxes that represent foreign keys
LoadcboCategoriesProducts
LoadcboSuppliersProducts
'
' Load the Products result set
PopulateRecordSet
With m_resProducts
  If Not (.EOF And .BOF) Then
    PerformPopulateFields
  Else
    m_eFormStatus = FORM_ADD_ONLY_VIEW
  End If
End With
'
' Stop the user from entering data until they choose New or Change
PerformLockFields
'
' Move the form into position and show it
Me.Move 0, 0, m_lngWidth, m_lngHeight
Me.Show
SystemIsWaiting

End Sub
Private Sub Form_Resize()
  Dim lngTwipsPerPixelX As Long
  Dim lngTwipsPerPixelY As Long
  '
  ' Don't try to resize on minimize
  If WindowState = vbMinimized Then
    Exit Sub
  End If

  lngTwipsPerPixelX = Screen.TwipsPerPixelX
  lngTwipsPerPixelY = Screen.TwipsPerPixelY
  '
  ' Ensure the Tab fills the available space on the form
  sstMain.Move 0, 0, ScaleWidth, ScaleHeight
  '
  ' Resize the panels on the current tab. We can't touch invisible
  ' tabs because the Left property is set to a large -ve value.
  Select Case sstMain.Tab
    Case 0 ' Detail tab
      fraDetail.Move lngTwipsPerPixelX * 2&, _
                    sstMain.TabHeight + lngTwipsPerPixelY * 2&, _
                    lngMax(sstMain.Width - lngTwipsPerPixelX * 4&, 1&), _
                    lngMax(sstMain.Height - sstMain.TabHeight - lngTwipsPerPixelY * 4, 1&)
      '
      ' Customization Point : If you want to add code to resize individual
      ' controls, do it here. Best done relative to
      ' fraDetail properties.
      ' =====
      ' =====
    Case 1 ' List tab
      fraGrid.Move lngTwipsPerPixelX * 2&, _
                  sstMain.TabHeight + lngTwipsPerPixelY * 2&, _
                  lngMax(sstMain.Width - lngTwipsPerPixelX * 4&, 1&), _
                  lngMax(sstMain.Height - sstMain.TabHeight - lngTwipsPerPixelY * 4, 1&)
      grdMain.Move lngTwipsPerPixelX * 8&, _
                  lngTwipsPerPixelY * 8&, _
                  lngMax(fraGrid.Width - lngTwipsPerPixelX * 16&, 1&), _
                  lngMax(fraGrid.Height - lngTwipsPerPixelY * 32, 1&)
      lblDoubleClickMessage.Move grdMain.Left, grdMain.Top + grdMain.Height + lngTwipsPerPixelY * 4
      AutoSizeGrid grdMain
      '
      ' Customization Point : If you add more tabs you can add resizing code
      ' here.
  End Select

```

```

' =====
' =====

End Select

End Sub
Private Sub Form_Unload(Cancel As Integer)
'
' Since this form is no longer active, we don't want to receive
' the custom ButtonClick events broadcast by the mdiParent.
Set m_mdiParent = Nothing
' Set the button states to nothing, if there's another form open
' it will set enable the appropriate buttons
mdiParent.SetToolBarState FORM_UNLOADED
End Sub
Private Sub m_mdiParent_ButtonClick(strKey As String)
'
' With a little bit of intermediate VB level code, we can
' (almost) treat the toolbar on the mdiParent as if it was
' on the form itself. It even shows up in the Code Window
' pull-downs.
'
' Provided we set the m_mdiParent to the mdiParent in this form's
' Activate event (and set it to nothing in the Deactivate event)
' then this form will receive the ButtonClick events raised by
' the mdiParent.
'
' Declare static variable to track status before Undo
Static eLastFormStatus As enmFormState

SystemIsWorking
Select Case strKey
Case "New"
    eLastFormStatus = m_eFormStatus
    m_eFormStatus = FORM_CHANGE
    mdiParent.SetToolBarState m_eFormStatus
    PerformNewAction
Case "Change"
    eLastFormStatus = m_eFormStatus
    m_eFormStatus = FORM_CHANGE
    mdiParent.SetToolBarState m_eFormStatus
    PerformChangeAction
Case "Delete"
    PerformDeleteAction
Case "Save"
    If AllFieldsPassValidation Then
        m_eFormStatus = FORM_READY_VIEW
        mdiParent.SetToolBarState m_eFormStatus
        PerformSaveAction
    End If
Case "Undo"
    m_eFormStatus = eLastFormStatus
    mdiParent.SetToolBarState m_eFormStatus
    PerformUndoAction
Case "Refresh"
    PerformRefreshAction
Case "First"
    PerformFirstAction
Case "Previous"
    PerformPreviousAction
Case "Next"
    PerformNextAction
Case "Last"
    PerformLastAction
Case "Close"
    Unload Me
End Select
SystemIsWaiting

End Sub

```

```

Private Sub PerformNewAction()
    '
    ' Clear the DataObject and the fields on the form
    cboCategoriesProducts.ListIndex = -1
    cboSuppliersProducts.ListIndex = -1
    txtProductName.Text = vbNullString
    txtQuantityPerUnit.Text = vbNullString
    txtUnitPrice.Text = vbNullString
    txtUnitsInStock.Text = vbNullString
    txtUnitsOnOrder.Text = vbNullString
    txtReorderLevel.Text = vbNullString
    txtDiscontinued.Text = FALSE_VALUE
    '
    ' Allow Data Entry
    PerformUnlockFields
    '
    ' Set flag to say this is New
    m_blnNew = True
End Sub
Private Sub PerformChangeAction()
    PerformUnlockFields
    '
    ' Turn off New flag
    m_blnNew = False
End Sub
Private Sub PerformDeleteAction()
    Dim strSQL As String
    Dim lngRowsAffected As Long
    Dim adoParm As ADODB.Parameter
    Dim adoCommand As ADODB.Command

    If m_resProducts Is Nothing Then
        MsgBox "Nothing to delete", vbInformation
        Exit Sub
    End If
    If m_resProducts.RecordCount = 0 Then
        MsgBox "Nothing to delete", vbInformation
        Exit Sub
    End If

    If MsgBox("Are you sure you wish to delete this record?", vbQuestion Or vbYesNo, "Delete Confirmation") = vbNo Then
        Exit Sub
    End If

    If m_resProducts.EOF Then
        m_resProducts.MoveLast
    ElseIf m_resProducts.BOF Then
        m_resProducts.MoveFirst
    End If
    '
    ' Create Delete Query
    strSQL = "DELETE FROM [Products] "
    '
    ' Create where clause using previous primary keys and concurrency values
    strSQL = strSQL & " WHERE "
    strSQL = strSQL & "ProductID = " & ConvertToSQL(m_resProducts!ProductID, "Long")
    '
    ' Execute SQL Delete command
    Set adoCommand = New ADODB.Command
    With adoCommand
        Set .ActiveConnection = g_adoConnection
        .CommandType = adCmdText
        .CommandText = strSQL
    End With
    ' Don't let JET handle errors that we can handle
    On Error GoTo HandleJetError
    .Execute lngRowsAffected

```

```

        On Error GoTo ErrorTrap
        Set .ActiveConnection = Nothing
    End With
    Select Case lngRowsAffected
        Case 0&
            MsgBox "Delete failed. No rows deleted. Probable conflict with another user or process or
earlier activity. Click Refresh before trying again.", vbExclamation
        Case 1&
            PopulateRecordSet
            With m_resProducts
                Select Case True
                    Case .EOF And .BOF
                        m_eFormStatus = FORM_ADD_ONLY_VIEW
                        mdiParent.SetToolBarState m_eFormStatus
                    Case .EOF
                        .MovePrevious
                    Case Else
                        .MoveNext
                        If .EOF Then
                            .MovePrevious
                            .MovePrevious
                        End If
                End Select
            End With
            If Not .BOF Then
                m_lngProductID_PK = .Fields("ProductID").Value
            Else
                m_lngProductID_PK = 0
            End If
            PerformPopulateFields
        End With
        Case Else
            MsgBox "Delete failed. Multiple rows deleted. Probable conflict between actual primary keys and
assumed primary keys", vbCritical
        End Select
        Set adoCommand = Nothing

        PopulateRecordSet
        PerformLockFields
        m_blnGridLoaded = False ' To force a Grid reload next time the tab is clicked

    Exit Sub

HandleJetError:
    MsgBox "(" & CStr(Err.Number) & ")" & Err.Description, vbExclamation
    Resume Next

ErrorTrap:
    PerformLockFields
    MsgBox "(" & CStr(Err.Number) & ")" & Err.Description, vbCritical
    Exit Sub

End Sub
Private Sub PerformSaveAction()
    Dim strNames() As String
    Dim strValues() As String
    Dim strSQL As String
    Dim lngRowsAffected As Long
    Dim adoParm As ADODB.Parameter
    Dim adoCommand As ADODB.Command

    Dim lngLastProductID As Long
    Dim lngProductID As Long
    Dim intName As Integer
    Dim intValue As Integer

    If m_blnNew Then
        ReDim strNames(0 To 8)
        ReDim strValues(0 To 8)
        '
        ' Initialize the Concurrency Column(s) for a new record
        '
    End If

```

```

' Get the last Identity assigned as a way to limit the size of the query for
' finding the Identity that was assigned by the Insert query.
lngLastProductID = LastProductID()
'
' Create Insert Query
If m_lngCategoriesProductsIndex > -1& Then
    strNames(0) = "CategoryID"
End If
If m_lngCategoriesProductsIndex > -1& Then
    strValues(0) = ConvertToSQL(m_uCategoriesProducts(m_lngCategoriesProductsIndex).lngCategoryID,
"Long")
End If
If m_lngSuppliersProductsIndex > -1& Then
    strNames(1) = "SupplierID"
End If
If m_lngSuppliersProductsIndex > -1& Then
    strValues(1) = ConvertToSQL(m_uSuppliersProducts(m_lngSuppliersProductsIndex).lngSupplierID,
"Long")
End If
strNames(2) = "ProductName"
strValues(2) = ConvertToSQL(txtProductName.Text, "String")
strNames(3) = "QuantityPerUnit"
strValues(3) = ConvertToSQL(txtQuantityPerUnit.Text, "String")
strNames(4) = "UnitPrice"
strValues(4) = ConvertToSQL(txtUnitPrice.Text, "Currency")
strNames(5) = "UnitsInStock"
strValues(5) = ConvertToSQL(txtUnitsInStock.Text, "Integer")
strNames(6) = "UnitsOnOrder"
strValues(6) = ConvertToSQL(txtUnitsOnOrder.Text, "Integer")
strNames(7) = "ReorderLevel"
strValues(7) = ConvertToSQL(txtReorderLevel.Text, "Integer")
strNames(8) = "Discontinued"
strValues(8) = ConvertToSQL(txtDiscontinued.Text, "Boolean")

ReDim Preserve strValues(0 To 8)
ReDim Preserve strNames(0 To 8)

Set adoCommand = New ADODB.Command
With adoCommand
    Set .ActiveConnection = g_adoConnection
    .CommandType = adCmdText
    .CommandText = FormulateInsertQuery("Products", strNames, strValues)
    On Error GoTo HandleJetError
    .Execute lngRowsAffected
    On Error GoTo ErrorTrap
    Set .ActiveConnection = Nothing
End With
Select Case lngRowsAffected
    Case 0&
        MsgBox "Insert failed. No rows inserted. Probable conflict with another user or process or
earlier activity. Click Refresh before trying again.", vbExclamation
    Case 1&
        m_lngProductID_PK = AssignedProductID(lngLastProductID, strNames(), strValues())
    Case Else
        MsgBox "Insert failed. Multiple rows Inserted. Probable conflict between actual primary keys
and assumed primary keys", vbCritical
End Select
Set adoCommand = Nothing
Else
'
' Create Update Query
ReDim strNames(0 To 10)
ReDim strValues(0 To 10)
intName = 0
intValue = 0
If m_lngCategoriesProductsIndex > -1& Then
    strNames(intName) = "CategoryID"
    intName = intName + 1
End If
If m_lngCategoriesProductsIndex > -1& Then

```

```

        strValues(intValue) =
ConvertToSQL(m_uCategoriesProducts(m_lngCategoriesProductsIndex).lngCategoryID, "Long")
        intValue = intValue + 1
    End If
    If m_lngSuppliersProductsIndex > -1& Then
        strNames(intName) = "SupplierID"
        intName = intName + 1
    End If
    If m_lngSuppliersProductsIndex > -1& Then
        strValues(intValue) =
ConvertToSQL(m_uSuppliersProducts(m_lngSuppliersProductsIndex).lngSupplierID, "Long")
        intValue = intValue + 1
    End If
    If txtProductName.Text <> m_resProducts!ProductName Then
        strNames(intName) = "ProductName"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtProductName.Text, "String")
        intValue = intValue + 1
    End If
    If txtQuantityPerUnit.Text <> m_resProducts!QuantityPerUnit Then
        strNames(intName) = "QuantityPerUnit"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtQuantityPerUnit.Text, "String")
        intValue = intValue + 1
    End If
    If txtUnitPrice.Text <> m_resProducts!UnitPrice Then
        strNames(intName) = "UnitPrice"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtUnitPrice.Text, "Currency")
        intValue = intValue + 1
    End If
    If txtUnitsInStock.Text <> m_resProducts!UnitsInStock Then
        strNames(intName) = "UnitsInStock"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtUnitsInStock.Text, "Integer")
        intValue = intValue + 1
    End If
    If txtUnitsOnOrder.Text <> m_resProducts!UnitsOnOrder Then
        strNames(intName) = "UnitsOnOrder"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtUnitsOnOrder.Text, "Integer")
        intValue = intValue + 1
    End If
    If txtReorderLevel.Text <> m_resProducts!ReorderLevel Then
        strNames(intName) = "ReorderLevel"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtReorderLevel.Text, "Integer")
        intValue = intValue + 1
    End If
    If BooleanFieldToSQL(txtDiscontinued.Text) <> m_resProducts!Discontinued Then
        strNames(intName) = "Discontinued"
        intName = intName + 1
        strValues(intValue) = ConvertToSQL(txtDiscontinued.Text, "Boolean")
        intValue = intValue + 1
    End If

m_lngProductID_PK = m_resProducts!ProductID

    If intName = 0 Then
        GoTo NormalExit
    End If
    ReDim Preserve strNames(0 To intName - 1)
    ReDim Preserve strValues(0 To intValue - 1)
    '
    ' Create where clause using previous primary keys and concurrency values
    strSQL = " WHERE "
    strSQL = strSQL & "ProductID = " & ConvertToSQL(m_resProducts!ProductID, "Long")
    Set adoCommand = New ADODB.Command
    With adoCommand
        Set .ActiveConnection = g_adoConnection
        .CommandType = adCmdText
    End With

```

```

        .CommandText = FormulateUpdateQuery("Products", strNames, strValues) & strSQL
        On Error GoTo HandleJetError
        .Execute lngRowsAffected
        On Error GoTo ErrorTrap
        Set .ActiveConnection = Nothing
    End With
    Select Case lngRowsAffected
        Case 0&
            MsgBox "Update failed. No rows updated. Probable conflict with another user or process or
earlier activity. Click Refresh before trying again.", vbExclamation
        Case 1&
            m_lngProductID_PK = m_resProducts!ProductID
        Case Else
            MsgBox "Update failed. Multiple rows updated. Probable conflict between actual primary keys
and assumed primary keys", vbCritical
    End Select
    Set adoCommand = Nothing

End If

NormalExit:
    PopulateRecordSet
    PerformLockFields
    m_blnGridLoaded = False ' To force a Grid reload next time the tab is clicked
    '
    ' Turn off New flag
    m_blnNew = False
    '
    Exit Sub

HandleJetError:
    MsgBox "(" & CStr(Err.Number) & ") " & Err.Description, vbExclamation
    Resume Next

ErrorTrap:
    PerformLockFields
    MsgBox "(" & CStr(Err.Number) & ") " & Err.Description, vbCritical
    Exit Sub

End Sub
Private Function LastProductID() As Long
    '
    ' Function is used before an insert performed using SQL
    ' to retrieve the last Identity assigned by the database.
    ' This is used to optimize the retrieval of the actual
    ' value assigned by the database.
    Dim lngProductID As Long
    Dim intTry As Integer
    Dim adoRecordSet As ADODB.Recordset
    Dim strSQL As String

    LastProductID = -1&
    strSQL = "SELECT " & _
        "MAX(ProductID) AS MAX_ProductID " & _
        "FROM " & _
        "[Products]"
    Set adoRecordSet = New ADODB.Recordset
    With adoRecordSet
        .CursorLocation = adUseClient
        .Open strSQL, g_adoConnection, adOpenStatic, adLockReadOnly, adCmdText
        Set .ActiveConnection = Nothing
        If Not (.EOF And .BOF) Then
            If Not IsNull(adoRecordSet!MAX_ProductID) Then
                LastProductID = adoRecordSet!MAX_ProductID
            Else
                LastProductID = 0&
            End If
        End If
        .Close
    End With
    Set adoRecordSet = Nothing

```

End Function

Private Function AssignedProductID(ByVal lngLastProductID As Long, strNames() As String, strValues() As String) As Long

' Function is used after an insert performed using SQL instead of a  
' standard stored procedure to retrieve the ID assigned by the database.

Dim lngProductID As Long  
Dim adoRecordSet As ADODB.Recordset  
Dim strSQL As String

AssignedProductID = -1&

strSQL = "SELECT " & \_  
"ProductID " & \_  
"FROM " & \_  
"[Products] " & \_  
"WHERE " & \_  
"ProductID > " & lngLastProductID & " and " & \_  
FormulateWhereSubClauses(strNames(), strValues())

Set adoRecordSet = New ADODB.Recordset  
With adoRecordSet  
  .CursorLocation = adUseClient  
  .Open strSQL, g\_adoConnection, adOpenStatic, adLockReadOnly, adCmdText  
  Set .ActiveConnection = Nothing  
  If Not (.EOF And .BOF) Then  
    AssignedProductID = CLng(adoRecordSet!ProductID)  
  End If  
  .Close  
End With  
Set adoRecordSet = Nothing

End Function

Private Sub PerformUndoAction()

  PerformPopulateFields  
  PerformLockFields

End Sub

Private Sub PerformFirstAction()

  On Error GoTo ErrorTrap  
  
  If Not m\_resProducts Is Nothing Then  
    m\_resProducts.MoveFirst  
    PerformPopulateFields  
  End If  
  Exit Sub

ErrorTrap:

  ReportADOErrors Me.Name  
  Exit Sub

End Sub

Private Sub PerformPreviousAction()

  On Error GoTo ErrorTrap  
  
  If Not m\_resProducts Is Nothing Then  
    With m\_resProducts  
      If Not .BOF Then  
        .MovePrevious  
      If .BOF Then  
        .MoveNext  
      End If  
    End With  
    PerformPopulateFields

  Exit Sub

ErrorTrap:

  If m\_resProducts.BOF = False Then  
    ReportADOErrors Me.Name

```

End If
Exit Sub
End Sub
Private Sub PerformNextAction()
On Error GoTo ErrorTrap

If Not m_resProducts Is Nothing Then
With m_resProducts
If Not .EOF Then
.MoveNext
If .EOF Then
.MovePrevious
End If
End If
End With
PerformPopulateFields
End If
Exit Sub
ErrorTrap:
If m_resProducts.EOF = False Then
ReportADOErrors Me.Name
End If
Exit Sub
End Sub
Private Sub PerformLastAction()
On Error GoTo ErrorTrap

If Not m_resProducts Is Nothing Then
m_resProducts.MoveLast
PerformPopulateFields
End If
Exit Sub
ErrorTrap:
ReportADOErrors Me.Name
Exit Sub

End Sub
Private Sub PerformUnlockFields()
'
' Let the user touch the fields & show they can
cboCategoriesProducts.Locked = False
cboCategoriesProducts.BackColor = UNLOCKED_COLOR
cboSuppliersProducts.Locked = False
cboSuppliersProducts.BackColor = UNLOCKED_COLOR
txtProductName.Locked = False
txtProductName.BackColor = UNLOCKED_COLOR
txtQuantityPerUnit.Locked = False
txtQuantityPerUnit.BackColor = UNLOCKED_COLOR
txtUnitPrice.Locked = False
txtUnitPrice.BackColor = UNLOCKED_COLOR
txtUnitsInStock.Locked = False
txtUnitsInStock.BackColor = UNLOCKED_COLOR
txtUnitsOnOrder.Locked = False
txtUnitsOnOrder.BackColor = UNLOCKED_COLOR
txtReorderLevel.Locked = False
txtReorderLevel.BackColor = UNLOCKED_COLOR
txtDiscontinued.Locked = False
txtDiscontinued.BackColor = UNLOCKED_COLOR
'
' Set focus to first field on form, if possible
On Error Resume Next
cboCategoriesProducts.SetFocus
End Sub

Private Sub PerformPopulateFields()
Dim lngX As Long
Dim vntFields As Variant

If m_resProducts Is Nothing Then
Exit Sub
End If

```

```

With m_resProducts
  If .EOF And .BOF Then
    Exit Sub
  End If
  m_lngCategoriesProductsIndex = -1&

  ReDim vntFields(0 To 0)
  vntFields(0) = !CategoryID
  For lngX = 0 To cboCategoriesProducts.ListCount - 1
    If m_uCategoriesProducts(lngX).lngCategoryID = vntFields(0) _
      Then
      cboCategoriesProducts.ListIndex = lngX
      m_lngCategoriesProductsIndex = lngX
    End If
  Next lngX
  m_lngSuppliersProductsIndex = -1&

  ReDim vntFields(0 To 0)
  vntFields(0) = !SupplierID
  For lngX = 0 To cboSuppliersProducts.ListCount - 1
    If m_uSuppliersProducts(lngX).lngSupplierID = vntFields(0) _
      Then
      cboSuppliersProducts.ListIndex = lngX
      m_lngSuppliersProductsIndex = lngX
    End If
  Next lngX
  txtProductName.Text = ForceNullToZeroLengthString(!ProductName)
  txtQuantityPerUnit.Text = ForceNullToZeroLengthString(!QuantityPerUnit)
  txtUnitPrice.Text = ForceNullToZeroLengthString(!UnitPrice)
  txtUnitsInStock.Text = ForceNullToZeroLengthString(!UnitsInStock)
  txtUnitsOnOrder.Text = ForceNullToZeroLengthString(!UnitsOnOrder)
  txtReorderLevel.Text = ForceNullToZeroLengthString(!ReorderLevel)
  txtDiscontinued.Text = IIf(SQLFieldToBoolean(!Discontinued), TRUE_VALUE, FALSE_VALUE)
End With

End Sub
Private Sub PerformLockFields()
  '
  ' Don't let the user touch the fields & show they can't
  cboCategoriesProducts.Locked = True
  cboCategoriesProducts.BackColor = LOCKED_COLOR
  cboSuppliersProducts.Locked = True
  cboSuppliersProducts.BackColor = LOCKED_COLOR
  txtProductName.Locked = True
  txtProductName.BackColor = LOCKED_COLOR
  txtQuantityPerUnit.Locked = True
  txtQuantityPerUnit.BackColor = LOCKED_COLOR
  txtUnitPrice.Locked = True
  txtUnitPrice.BackColor = LOCKED_COLOR
  txtUnitsInStock.Locked = True
  txtUnitsInStock.BackColor = LOCKED_COLOR
  txtUnitsOnOrder.Locked = True
  txtUnitsOnOrder.BackColor = LOCKED_COLOR
  txtReorderLevel.Locked = True
  txtReorderLevel.BackColor = LOCKED_COLOR
  txtDiscontinued.Locked = True
  txtDiscontinued.BackColor = LOCKED_COLOR

End Sub
Private Function AllFieldsPassValidation() As Boolean
  Dim strMessage As String

  With cboCategoriesProducts ' Primary Key
    If .Text = vbNullString And m_lngCategoriesProductsIndex > -1& Then
      strMessage = lblCategoriesProducts.Caption & " required"
      GoTo ErrorTrap
    End If
  '
  ' Add further checks here, if required

```

```

'=====
'
End With
With cboSuppliersProducts ' Primary Key
If .Text = vbNullString And m_lngSuppliersProductsIndex > -1& Then
    strMessage = lblSuppliersProducts.Caption & " required"
    GoTo ErrorTrap
End If
'
' Add further checks here, if required
'=====
'
End With
With txtProductName
'
' Add further checks here, if required
'=====

End With
With txtQuantityPerUnit
'
' Add further checks here, if required
'=====

End With
With txtUnitPrice
'
' Add further checks here, if required
'=====

End With
With txtUnitsInStock
'
' Add further checks here, if required
'=====

End With
With txtUnitsOnOrder
'
' Add further checks here, if required
'=====

End With
With txtReorderLevel
'
' Add further checks here, if required
'=====

End With
With txtDiscontinued
'
' Add further checks here, if required
'=====

End With

AllFieldsPassValidation = True
Exit Function

ErrorTrap:
MsgBox strMessage, vbInformation, "Validation Error"
AllFieldsPassValidation = False
Exit Function

End Function
Private Sub cboCategoriesProducts_Click()
    m_lngCategoriesProductsIndex = cboCategoriesProducts.ListIndex
End Sub
Private Sub cboCategoriesProducts_KeyPress(KeyAscii As Integer)

```

```

    If KeyAscii = vbKeyReturn Then
        m_lngCategoriesProductsIndex = cboCategoriesProducts.ListIndex
    End If
End Sub
Private Sub LoadcboCategoriesProducts()
    Dim strSQL As String
    Dim lngX As Long
    Dim adoRecordSet As ADODB.Recordset
    Dim strCBOItem As String

    strSQL = "Select Distinct "
    strSQL = strSQL & "CategoryID,"

    If Right$(strSQL, 1) = "," Then
        Mid$(strSQL, Len(strSQL), 1&) = " "
    End If
    strSQL = strSQL & " From Categories"
    '
    ' Customization point
    'strSQL = strSQL & " Where ..."
    Set adoRecordSet = New ADODB.Recordset
    With adoRecordSet
        .CursorLocation = adUseClient
        .Open strSQL, g_adoConnection, adOpenStatic, adLockReadOnly, adCmdText
        Set .ActiveConnection = Nothing
        If Not (.EOF And .BOF) Then
            lngX = 0&
            ReDim m_uCategoriesProducts(0& To 999&)
            .MoveFirst
            Do Until .EOF
                strCBOItem = vbNullString
                strCBOItem = strCBOItem & ForceNullToZeroLengthString(!CategoryID) & ", "
                cboCategoriesProducts.AddItem Left$(strCBOItem, Len(strCBOItem) - 2&), lngX
                If lngX > UBound(m_uCategoriesProducts) Then
                    ReDim Preserve m_uCategoriesProducts(0& To lngX + 1000&)
                End If
                m_uCategoriesProducts(lngX).lngCategoryID = !CategoryID & " "
                lngX = lngX + 1&
                .MoveNext
            Loop
            ReDim Preserve m_uCategoriesProducts(0 To lngX - 1&)
        Else
            ReDim m_uCategoriesProducts(0& To 0&)
        End If
        .Close
    End With
    Set adoRecordSet = Nothing

End Sub
Private Sub cboSuppliersProducts_Click()
    m_lngSuppliersProductsIndex = cboSuppliersProducts.ListIndex
End Sub
Private Sub cboSuppliersProducts_KeyPress(KeyAscii As Integer)
    If KeyAscii = vbKeyReturn Then
        m_lngSuppliersProductsIndex = cboSuppliersProducts.ListIndex
    End If
End Sub
Private Sub LoadcboSuppliersProducts()
    Dim strSQL As String
    Dim lngX As Long
    Dim adoRecordSet As ADODB.Recordset
    Dim strCBOItem As String

    strSQL = "Select Distinct "
    strSQL = strSQL & "SupplierID,"

    If Right$(strSQL, 1) = "," Then
        Mid$(strSQL, Len(strSQL), 1&) = " "
    End If
    strSQL = strSQL & " From Suppliers"
    '

```

```

' Customization point
'strSQL = strSQL & " Where ..."
Set adoRecordSet = New ADODB.Recordset
With adoRecordSet
    .CursorLocation = adUseClient
    .Open strSQL, g_adoConnection, adOpenStatic, adLockReadOnly, adCmdText
    Set .ActiveConnection = Nothing
    If Not (.EOF And .BOF) Then
        lngX = 0&
        ReDim m_uSuppliersProducts(0& To 999&)
        .MoveFirst
        Do Until .EOF
            strCBOItem = vbNullString
            strCBOItem = strCBOItem & ForceNullToZeroLengthString(!SupplierID) & ", "
            cboSuppliersProducts.AddItem Left$(strCBOItem, Len(strCBOItem) - 2&), lngX
            If lngX > UBound(m_uSuppliersProducts) Then
                ReDim Preserve m_uSuppliersProducts(0& To lngX + 1000&)
            End If
            m_uSuppliersProducts(lngX).lngSupplierID = !SupplierID & ""
            lngX = lngX + 1&
            .MoveNext
        Loop
        ReDim Preserve m_uSuppliersProducts(0 To lngX - 1&)
    Else
        ReDim m_uSuppliersProducts(0& To 0&)
    End If
    .Close
End With
Set adoRecordSet = Nothing

End Sub
Private Sub sstMain_Click(PreviousTab As Integer)

    If sstMain.Tab = 1 Then
        If m_blnGridLoaded = False Then
            PopulateGrid
        End If
    End If
    '
    ' Do the resize so that the tab is sized correctly. Note that you
    ' can't easily resize the tab while it is hidden because the Left values are
    ' all set to large negative values.
    Form_Resize

End Sub
Private Sub txtProductName_GotFocus()
    SelectField txtProductName
End Sub
Private Sub txtQuantityPerUnit_GotFocus()
    SelectField txtQuantityPerUnit
End Sub
Private Sub txtUnitPrice_GotFocus()
    SelectField txtUnitPrice
End Sub
Private Sub txtUnitsInStock_GotFocus()
    SelectField txtUnitsInStock
End Sub
Private Sub txtUnitsOnOrder_GotFocus()
    SelectField txtUnitsOnOrder
End Sub
Private Sub txtReorderLevel_GotFocus()
    SelectField txtReorderLevel
End Sub
Private Sub txtDiscontinued_Click()

    With txtDiscontinued
        If .Locked Then
            Exit Sub
        End If
        If .Text = TRUE_VALUE Then
            .Text = FALSE_VALUE
        End If
    End With
End Sub

```

```

Else
    .Text = TRUE_VALUE
End If
End With

End Sub
Private Sub txtDiscontinued_KeyPress(KeyAscii As Integer)

    If KeyAscii = Asc(" ") Then
        txtDiscontinued_Click
    End If
    KeyAscii = 0

End Sub
Private Sub PopulateGrid()
    Dim rsRecordSet As ADODB.Recordset
    Dim strSQL As String
    Dim lngCol As Long
    Dim lngColumnCount As Long
    Dim lngRow As Long
    Dim lngRowCount As Long
    Dim vntData As Variant
    Dim strReferenceColumnNameNames() As String

    SystemIsWorking
    strSQL = "select "
    ReDim strReferenceColumnNameNames(0 To 1)
    strSQL = strSQL & "Categories_1.CategoryID AS CategoryID_1,"
    strReferenceColumnNameNames(0) = "Categories_1.CategoryID"
    strSQL = strSQL & "Suppliers_2.SupplierID AS SupplierID_2,"
    strReferenceColumnNameNames(1) = "Suppliers_2.SupplierID"
    strSQL = strSQL & "[Products].[ProductID],"
    strSQL = strSQL & "[Products].[ProductName],"
    strSQL = strSQL & "[Products].[QuantityPerUnit],"
    strSQL = strSQL & "[Products].[UnitPrice],"
    strSQL = strSQL & "[Products].[UnitsInStock],"
    strSQL = strSQL & "[Products].[UnitsOnOrder],"
    strSQL = strSQL & "[Products].[ReorderLevel],"
    strSQL = strSQL & "[Products].[Discontinued],"
    If Right(strSQL, 1) = "," Then
        Mid$(strSQL, Len(strSQL), 1) = " "
    End If
    strSQL = strSQL & " from "
    strSQL = strSQL & "[Products]"
    strSQL = strSQL & ",Categories Categories_1"
    strSQL = strSQL & ",Suppliers Suppliers_2"
    strSQL = strSQL & " where "
    strSQL = strSQL & "[Products].[CategoryID] = " & strReferenceColumnNameNames(0)
    strSQL = strSQL & " and "
    strSQL = strSQL & "[Products].[SupplierID] = " & strReferenceColumnNameNames(1)
    modPublic.PopulateGrid strSQL, grdMain
    m_blnGridLoaded = True
    SystemIsWaiting

End Sub
Private Sub grdMain_DblClick()
    '
    ' This code relies on the grid and the m_resProducts recordset
    ' being in perfect synchronization. You can show different columns
    ' in the grid but you can't have a different sequence or a different
    ' rowcount.
    Dim lngRow As Long

    If m_resProducts.EOF And m_resProducts.BOF Then
        Exit Sub
    End If
    With grdMain
        lngRow = .Row
        If lngRow > 0& Then
            m_resProducts.AbsolutePosition = lngRow
            PerformPopulateFields
        End If
    End With

```

```

        sstMain.Tab = 0&
    End If
End With

End Sub

Private Sub PopulateRecordSet()
    Dim strSQL As String
    Dim blnFind As Boolean
    Dim strFind As String
    '
    ' Load the Products result set
    strSQL = "Select "
    strSQL = strSQL & "[Products].[ProductID],"
    strSQL = strSQL & "[Products].[ProductName],"
    strSQL = strSQL & "[Products].[SupplierID],"
    strSQL = strSQL & "[Products].[CategoryID],"
    strSQL = strSQL & "[Products].[QuantityPerUnit],"
    strSQL = strSQL & "[Products].[UnitPrice],"
    strSQL = strSQL & "[Products].[UnitsInStock],"
    strSQL = strSQL & "[Products].[UnitsOnOrder],"
    strSQL = strSQL & "[Products].[ReorderLevel],"
    strSQL = strSQL & "[Products].[Discontinued],"
    If Right(strSQL, 1) = "," Then
        Mid$(strSQL, Len(strSQL), 1) = " "
    End If
    strSQL = strSQL & " from "
    strSQL = strSQL & "[Products] "
    If m_resProducts Is Nothing Then
        Set m_resProducts = New ADODB.Recordset
    Else
        m_resProducts.Close
    End If
    With m_resProducts
        .CursorLocation = adUseClient
        .Open strSQL, g_adoConnection, adOpenStatic, adLockReadOnly, adCmdText
        Set .ActiveConnection = Nothing
        If Not (.EOF And .BOF) Then
            '
            ' Create Find clause using previous primary keys
            strFind = ""
            If m_lngProductID_PK <> 0& Then
                strFind = strFind & "ProductID = " & ConvertToSQL(m_lngProductID_PK, "Long")
                blnFind = True
            End If
            If blnFind Then
                .Filter = strFind
            Else
                .MoveFirst
            End If
        End If
    End With
End With

End Sub

Private Sub PerformRefreshAction()
    Dim lngAbsolutePosition As Long
    Dim vntBookMark As Variant

    If sstMain.Tab = 0 Then
        If m_resProducts Is Nothing Then
            Exit Sub
        End If

        With m_resProducts
            If .EOF And .BOF Then
                PopulateRecordSet
                If .EOF And .BOF Then
                    Exit Sub
                End If
                PerformPopulateFields
            Exit Sub
        Else
    
```

```
        On Error Resume Next
        lngAbsolutePosition = .AbsolutePosition
        vntBookMark = .Bookmark
        On Error GoTo 0
        PopulateRecordSet
        If .EOF And .BOF Then
            Exit Sub
        End If
        On Error Resume Next
        .AbsolutePosition = lngAbsolutePosition
        .Bookmark = vntBookMark
        On Error GoTo 0
        PerformPopulateFields
    End If
End With
Else
    PopulateGrid
End If
End Sub
```