Tema 9: ADO.NET (III)

- El control DataGridView.
- Actualización de registros.
- Modificación de un registro.
- Añadir un nuevo registro.
- Eliminar registros.
- Actualizar el origen de datos.
- Órdenes de actualización.

El control DataGridView

- Proporciona una interfaz para la visualización y edición de datos basada en el estilo de hoja de cálculo.

- Permite el enlace con distintos orígenes de datos:
  - Cualquier clase que implemente la interfaz IListSource, como las clases DataTable y Dataset.
  - Cualquier clase que implemente la interfaz IBindingList como la clase DataView.

- El enlace del control DataGridView con un origen de datos se realiza mediante las propiedades DataSource y DataMember.
  - La propiedad DataSource indica el origen de datos y lo asigna a algún objeto de las clase anteriores.
  - Si el origen de datos contiene varias tablas o listas, será necesario indicar a cual de ellas se va a enlazar mediante la propiedad DataMember.

El control DataGridView II

- Enlace de datos de una tabla de un DataSet con un DatagridView

El control DataGridView III

```
Private Sub frmGestiónClientes_Load(ByVal sender As System.Object, _
                                    ByVal e As System.EventArgs) Handles MyBase.Load
    ConfigurarAccesoADatos()

    'Enlazar DataGridView a la tabla Clientes
    DataGridView1.DataSource = ds
    DataGridView1.DataMember = "Clientes"

    ‘Si queremos que algunas columnas no se vean se puede utilizar:
    'DataGridView1.Columns("NombreColumna").Visible = False

    'También se puede cambiar la cabecera de una columna con:
    'DataGridView1.Columns("NombreColumna").HeaderText = "NuevoNombreColumna“

     DataGridView1.Columns("IdCliente").HeaderText = "ID.Cliente“
     DataGridView1.Columns("CP").HeaderText = "Código Postal“

    'Para ajustar el ancho de las columnas
    DataGridView1.AutoResizeColumns()
End Sub
```

El control DataGridView IV

```
Private Sub ConfigurarAccesoADatos()
   'Abre la conexión, establece el adaptador de datos y rellena el Dataset
   'Las variables cn, daClientes, daPedidos, daProductos, ds y nombreBBDD ya están declaradas

   ‘Abrir la conexión
   cn.ConnectionString = "PROVIDER=Microsoft.jet.oledb.4.0; " & _
                         "Data Source=" & nombreBBDD
   cn.Open()

   'Configurar los adaptadores de datos
   daClientes = New OleDbDataAdapter("SELECT * FROM Clientes", cn)
   daPedidos = New OleDbDataAdapter("SELECT * FROM Pedidos", cn)
   daProductos = New OleDbDataAdapter("SELECT * FROM Productos", cn)

   'Cargar el dataset
   daClientes.Fill(ds, "Clientes")
   daPedidos.Fill(ds, "Pedidos")
   daProductos.Fill(ds, "Productos")

   'Cerrar la conexión
   cn.Close()
End Sub
```

El control DataGridView V

- Enlace de datos de un DataView con un DatagridView

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)_
                                Handles Button1.Click

    Dim conexion As MySqlConnection = New MySqlConnection("server=localhost; user id=root;
                                                          password=''; database=master")

    'establecemos el select a ejecutar
    Dim dausuarios As MySqlDataAdapter = New MySqlDataAdapter("select * from usuarios", conexion)

    conexion.Open()

    Dim ds As New DataSet
    dausuarios.Fill(ds, "usuarios")

    Dim dv As DataView = ds.Tables("usuarios").DefaultView
    dv.RowFilter = "tipo='director'"

    conexion.Close()

    DataGridView1.DataSource = dv
End Sub
```

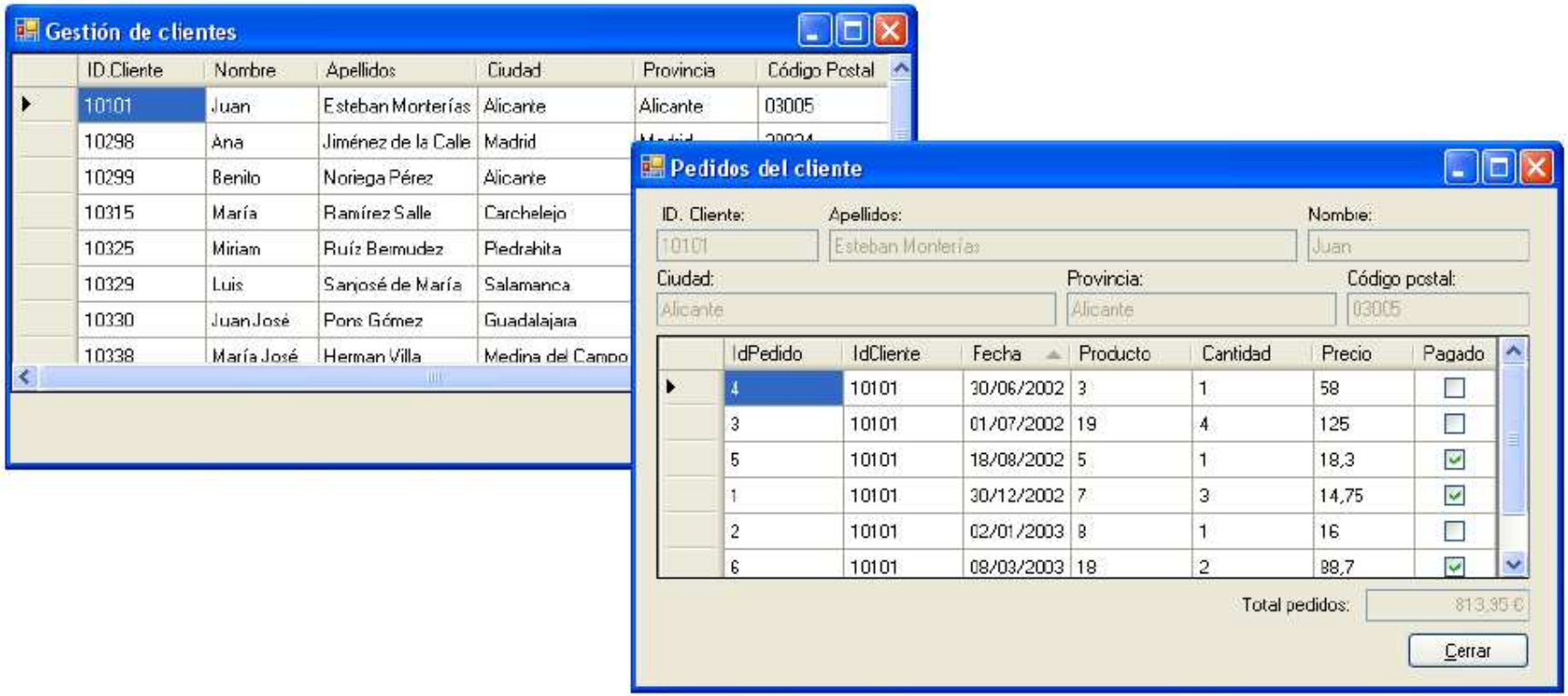
El control DataGridView VI

- Acceso al contenido de las celdas.
  - La propiedad CurrentCell hace referencia a la celda activa.
    - ✓ Devuelve un objeto de tipo DataViewGridCell.
    - ✓ Propiedad Value, devuelve el valor una celda.
  - DataGridView1.CurrentCell.Value, devuelve el valor de la celda activa.
    - ✓ Propiedades RowIndex y ColumnIndex de la celda activa devuelve la fila y la columna de una celda.
  - Para establecer la celda activa…
    - ✓ DataGridView1.CurrentCell = DataGridView1*(columna*, *fila*).
  - Ejemplo: recorrer todas las filas de un DataGridView…

```
'Recorre todas las celdas de un DataGridView
'Recorre todas las filas
For i As Integer = 0 To DataGridView1.Rows.Count – 1
    'Recorre todas las columnas de la fila
    For j As Integer = 0 To DataGridView1.Columns.Count – 1
        'Accede a cada una de las celdas j,i
         Debug.Write(DataGridView1(j, i).Value & " ")
    Next
    Debug.WriteLine("")
Next
```

El control DataGridView VII

- Acceso a los datos de las filas relacionadas.
  - La propiedad RowIndex devuelve el índice de la fila activa, y por ella será posible acceder al objeto DataRow del origen de datos.
- Ejemplo: al hacer doble click en una celda, aparecen los pedidos del cliente.



El control DataGridView VIII

```
'Este evento se produce al hacer doble Click sobre una celda de DataGridView
Private Sub DataGridView1_CellContentDoubleClick(ByVal sender As Object, _
                                                 ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) _
                                                 Handles DataGridView1.CellContentDoubleClick

  'Rellenar los cuadros de texto del formulario frmDetallesCliente con la información del cliente de la celda seleccionada.
  La orden With, permite abreviar la referencia al objeto
  With My.Forms.frmDetallesCliente
    .txtIdCliente.Text = ds.Tables("Clientes").Rows(e.RowIndex).Item("IdCliente")
    .txtApellidos.Text = ds.Tables("Clientes").Rows(e.RowIndex).Item("Apellidos")
    .txtNombre.Text = ds.Tables("Clientes").Rows(e.RowIndex).Item("Nombre")
    .txtPoblación.Text = ds.Tables("Clientes").Rows(e.RowIndex).Item("Ciudad")
    .txtProvincia.Text = ds.Tables("Clientes").Rows(e.RowIndex).Item("Provincia")
    .txtCP.Text = ds.Tables("Clientes").Rows(e.RowIndex).Item("CP")
  End With

  'Establecer los pedidos del cliente a partir de una vista de la tabla Pedidos
  Dim dv As DataView = ds.Tables("Pedidos").DefaultView
  dv.Sort = "Fecha"
  dv.RowFilter = "IdCliente =" & ds.Tables("Clientes").Rows(e.RowIndex).Item("IdCliente")
  My.Forms.frmDetallesCliente.DataGridView1.AutoResizeColumns()
  My.Forms.frmDetallesCliente.DataGridView1.DataSource = dv

My.Forms.frmDetallesCliente.Show() 'Mostrar el formulario secundario
End Sub
```

El control DataGridView IX

- Cada fila del DataGridView es un objeto de la clase DataGridViewRow.
  - La propiedad Rows, permite acceder a la colección de objetos DataGridViewRow del control.

- Ejemplo: cargar el total del importe de los pedidos en el cuatro de texto txtTotalPedidos.

```
Private Sub frmDetallesCliente_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
                                    Handles MyBase.Load
Dim totalPedidos As Decimal = 0
For Each dr As DataGridViewRow In DataGridView1.Rows
   totalPedidos += dr.Cells("Precio").Value * dr.Cells("Cantidad").Value
Next
txtTotalPedidos.Text = String.Format("{0:C}", totalPedidos)
End Sub
```

El control DataGridView X

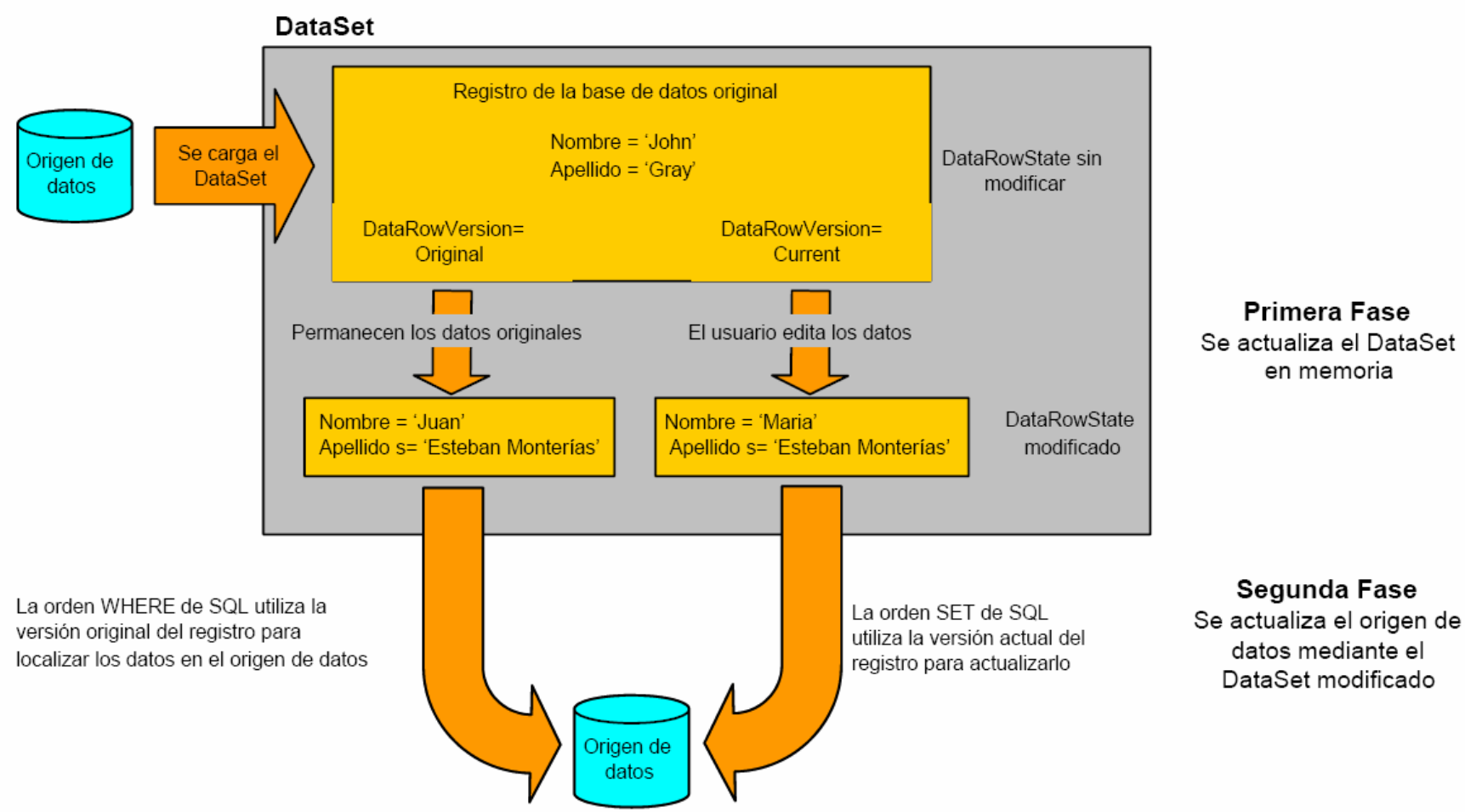
- Formato de las celdas del DataGridView
  - Las propiedades RowsDefaultCellStyle y AlternatingRowsDefaultCellStyle permiten definir el formato de todas las filas y las de las filas impares.
  - El evento CellFormatting permite modificar el formato de una celda en el momento que se va a mostrar. Para ello se comprueba primero el valor del campo en el que estamos y posteriormente su valor.
- Ejemplo: poner fondo rojo al campo tipo para el valor director.

```
Private Sub DataGridView1_CellFormatting (ByVal sender As Object, ByVal e As_
                                          System.Windows.Forms.DataGridViewCellFormattingEventArgs)_
                                          Handles DataGridView1.CellFormatting
    If DataGridView1.Columns(e.ColumnIndex).Name = "tipo" Then
       If e.Value = "director" Then
          e.CellStyle.BackColor = Color.Red
       End If
    End If
 End Sub
```

Actualización de registros

- La actualización de datos en un DataSet se realiza sólo en memoria.
  - Los datos originales se mantienen en el origen de datos.
- Una vez cargado el conjunto de datos, la actualización del origen de datos se hace en dos fases:
  - Primero se modifica el conjunto de datos (se modifica el valor de las columnas, se añaden nuevos registros, se eliminan filas).
    - ✓ Si el conjunto de datos se utiliza sólo como almacenamiento temporal para pasar los datos a otras aplicaciones, el proceso terminaría aquí.
  - Después, si se desea modificar el origen de datos, es necesario enviar las modificaciones de forma explícita con el método Update del adaptador de datos.
    - ✓ El método Update manda instrucciones SQL al origen de datos para cada fila modificada.

Actualización de registros II



Modificación de un registro

- Modificar un registro de una tabla de un DataSet, simplemente supone localizar un registro y modificar las columnas necesarias.

Ejemplo: Actualizar la ciudad del cliente con IdCliente=10101

```
'Localizar el registro
'Se supone que la propiedad PrimaryKey contiene la columna IdCliente
'encontrado es un dato de la clase DataRow
encontrado = ds.Tables("Clientes").Rows.Find(10101)

'Modificar el valor de la columna Ciudad
encontrado.Item("Ciudad") = "Benidorm"
```

Añadir un nuevo registro.

- La inserción de un nuevo registro implica tres pasos.

  1. Crear un nuevo registro vacío con el método NewRow de la clase DataTable.
     - ✓ NewRow crea un nuevo registro vació con el mismo esquema que la tabla que lo ha llamado.

  2. Introducir nuevos valores para cada una de las columnas de la tabla en las que sea necesario.
     - ✓ En la definición de la tabla en el gestor de bases de datos se puede especificar si cada columna puede tener campos vacíos o no.
     - ✓ En la definición de la tabla se puede definir un campo como autoincremental. El valor de dicho campo se calculará de forma automática.

  3. Añadir el nuevo registro a la colección Rows del objeto DataTable mediante el método Add.
     - ✓ Si existe una clave principal y está repetida se generará una excepción de tipo System.Data.ConstraintException.
     - ✓ Si no se proporcionan datos en las columnas en las que sea obligatorio se genera una excepción de tipo System Data NoNullAllowedException

Añadir un nuevo registro II.

```
'Añadir un nuevo registro a la tabla clientes
'Se supone que la propiedad PrimaryKey de la tabla tiene la columna IdCliente
'Crear un nuevo registro vacío
Dim nuevo As DataRow = ds.Tables("Clientes").NewRow

'Llenar los datos
System.Console.Write("Id. cliente:")
nuevo.Item("IdCliente") = System.Console.ReadLine()
System.Console.Write("Apellidos:")
nuevo.Item("Apellidos") = System.Console.ReadLine()
System.Console.Write("Nombre:")
nuevo.Item("Nombre") = System.Console.ReadLine()
System.Console.Write("Ciudad:")
nuevo.Item("Ciudad") = System.Console.ReadLine()

'Añadir el nuevo registro a la colección Rows comprobando si se inserta una fila con la clave duplicada
Try
    ds.Tables("Clientes").Rows.Add(nuevo)
Catch ex As ConstraintException
     System.Console.WriteLine("Error: " & ex.Message)
End Try
```

Eliminar registros.

- Una vez localizado el registro, sólo es necesario llamar al método Delete de la clase DataRow.

```
'Eliminar un registro
'Localizar el registro.Se supone que la propiedad PrimaryKey establecida
'encontrado es un dato de la clase DataRow

encontrado = ds.Tables("Clientes").Rows.Find(System.Console.ReadLine())
encontrado.Delete()
```

- Si el registro ya ha sido eliminado previamente se provocará una excepción de tipo DeleteRowInaccesibleException.

- El método Delete realiza un borrado lógico.
  - La propiedad RowState del registro se marca con el valor Deleted.
  - Los registros borrados se incluirán en el número de filas de la colección y aparecerán en los listados.

Actualizar el origen de datos.

❑ El método Update de los adaptadores de datos transmiten las modificaciones que se han realizado en memoria al origen de datos.

- Recorre la colección Rows de la tabla analizando la propiedad RowState para identificar la acción a realizar:

  ✓ Si contiene el valor Modified, llama a la instrucción SQL contenida en la propiedad UpdateCommand para enviar la modificación.
  ✓ Si contiene el valor Added, llama a la instrucción SQL contenida en la propiedad InsertCommand para enviar la modificación.
  ✓ Si contiene el valor Deleted, llama a la instrucción SQL contenida en la propiedad DeleteCommand para enviar la modificación.
  ✓ Si se desea actualizar varias tablas, será necesario llamar al método Update de cada adaptador que se ha utilizado para llenar la tabla.

❑ Rechazar los cambios pendientes.

- El método RejectChanges de la clase DataTable permite rechazar los cambios pendientes, restaurando las filas con su valor original.

Órdenes de actualización.

❑ Antes de utilizar el método Update es necesario configurar las propiedades UpdateCommand, InsertCommand y DeleteCommand del adaptador de datos.

- Utilizando un procedimiento almacenado en la base de datos que realice la instrucción SQL.
- Creando manualmente para cada una un nuevo comando que contenga las instrucciones SQL UPDATE, INSERT y DELETE.
  - ✓ Normalmente será necesario incluir en la orden parámetros que permitan realizar la acción en cada caso concreto.
- Permitir que ADO.NET genere las órdenes de forma automática

Órdenes de actualización II.

- Generar órdenes de actualización automáticamente.
  - Requisitos:
    - ✓ Es necesario haber rellenado previamente la tabla mediante el método Fill del adaptador de datos.
    - ✓ La orden para rellenar la tabla deberá hacer referencia a toda una tabla de la base de datos (por ejemplo "SELECT * FROM Clientes").
    - ✓ Si se cumplen estos requisitos, el adaptador de datos habrá generado una propiedad SelectCommand.
    - ✓ La propiedad SelectCommand deberá devolver entre los datos que se han recuperado del adaptador de datos una columna que contenga una clave primaria (valor sin duplicados).
  - Restricciones a la generación automática.
    - ✓ **Control de concurrencia**. No se permitirá controlar la actualización simultánea de una fila por varios usuarios.
    - ✓ **Tablas relacionadas**. Las órdenes de actualización sólo funcionan si se han recuperado del origen de datos tablas independientes no relacionadas.
    - ✓ **Nombres de tabla y columna**. Deberán estar compuestos de caracteres alfanuméricos, no permitiéndose espacios en blanco o caracteres especiales.

Órdenes de actualización III.

- La clase CommandBuilder.
  - Permite generar las órdenes de actualización de un adaptador de datos de forma automática.

```
'La conexión cn se supone ya creada y abierta. Crear los adaptadores de datos
daClientes = New OleDbDataAdapter("SELECT * FROM Clientes", cn)

'La propiedad SelectCommand se carga con SELECT * FROM Clientes
'Rellenar el dataset
daClientes.Fill(ds, "Clientes")

'Creación del objeto CommandBuilder del adaptador de datos
Dim cb As OleDbCommandBuilder = New OleDbCommandBuilder(daClientes)
```

- Actualizar los registros.
  - La orden Update utiliza como argumentos el DataSet que se va a utilizar y la tabla que se actualizará.

Órdenes de actualización IV.

```
'La conexión cn se supone ya creada y abierta. Crear los adaptadores de datos
daClientes = New OleDbDataAdapter("SELECT * FROM Clientes", cn)

'La propiedad SelectCommand se carga con SELECT * FROM Clientes. Rellenar el dataset
daClientes.Fill(ds, "Clientes")

'Creación del objeto CommandBuilder del adaptador de datos
Dim cb As OleDbCommandBuilder = New OleDbCommandBuilder(daClientes)
cn.Close()

'Modificar la ciudad del Cliente 10101 (el primer cliente)
ds.Tables("Clientes").Rows(0).Item("Ciudad") = "Benidorm"

'Añadir el cliente 10103
Dim nuevaFila As DataRow = ds.Tables("Clientes").NewRow
nuevaFila.Item("IdCliente") = 10103
nuevaFila.Item("Apellidos") = "Martínez"
nuevaFila.Item("Nombre") = "Juana"
ds.Tables("Clientes").Rows.Add(nuevaFila)

'Borrar el cliente 10102
Dim filaBorrada As DataRow() = ds.Tables("Clientes").Select("IdCliente=10102")
filaBorrada(0).Delete()

'Actualizar el origen de datos
daClientes.Update(ds, "Clientes")
```