



Bases de Datos en VB.Net

©2007 Justo Sáez Arenas
<http://vb.jsaez.com/>

Contenidos:

Introducción.....	2
Herramientas visuales.....	2
Conexión.....	2
Adaptador de Datos.....	3
Conjunto de Datos.....	3
Enlazado de Controles y Datos.....	4
Utilización conjunta de controles enlazados a datos.....	4
Moveirse por los registros.....	4
Otras propiedades del Control DataGrid.....	5
Propiedades y métodos del DataSet.....	6
Consultar un campo de un registro.....	6
Recorrer el DataSet.....	6
Añadir Registros.....	7
Editar Registros.....	7
Eliminar Registros.....	8
Actualizar los datos origen.....	9
Buscar datos.....	9



1. Bases de datos.

Introducción.

El nuevo estándar de Microsoft para utilizar bases de datos se denomina 'ADO.NET', de Access Data Object. Este estándar es común a toda la plataforma .net e interiormente trabaja con formato de datos XML, lo que hace más sencilla la utilización de fuentes de datos XML existentes y la utilización de ADO.NET en programas diseñados para Internet.

La característica fundamental para trabajar con bases de datos es la utilización de un Dataset o conjunto de datos que se extrae de la base de datos que deseamos utilizar.

Este Dataset será proporcionado por el sistema gestor de base de datos.

La forma de trabajo sigue la siguiente secuencia:

- Conexión. Se establece la conexión a la correspondiente base de datos.
- Adaptador de Datos. Que nos sirve para enviar y recibir los datos.
- Conjunto de Datos. Con los que se desea trabajar en el programa. En realidad no se trabaja con los datos reales, sino una copia de los mismos.

Herramientas visuales

Explorador de servidores.

Contiene todas las herramientas básicas para asociar bases de datos y manipular sus objetos. Permite establecer conexiones con fuentes de datos basadas en Internet, cliente-servidor o locales.

Generador de consultas.

Nos sirve para generar consultas SQL.


Diseñador de bases de datos.

Nos permite trabajar con una base de datos completa, pudiendo crear y modificar sus tablas.

1. Trabajar con Bases de Datos.

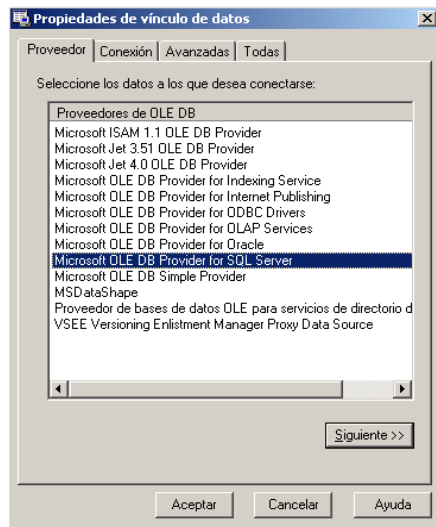
A continuación indicamos los pasos para una conexión a una base de datos Access.

Conexión.

Mediante el Explorador de Servidores, (*Menú Ver- Explorador de Servidores*), pulsaremos sobre el botón  'Conectar con Base de datos' ¹el cual nos mostrará la siguiente pantalla donde indicaremos los datos necesarios para establecer la conexión.

Como Proveedores, nos aparecerá el listado de gestores de bases de datos instalados en el sistema. Normalmente *Visual Studio .net* instala los más comunes. Y otras veces es el propio sistema gestor que lo instala.

¹ También se accede a esta pantalla , mediante *Menú Herramientas – Conectar con base de datos*.



Para bases de datos access elegiremos ‘*Microsoft Jet 4.0 OLE DB*’

El siguiente paso es seleccionar la base de datos (.mdb) a utilizar. Se selecciona y se puede ‘probar la conexión’ para estar seguros que se ha establecido correctamente.

Si todo va bien, aparecerá en el ‘Explorador de Servidores’ un nuevo nodo de conexión de datos, desde el podemos explorar la estructura de tablas y demás componentes de nuestra base de datos elegida.

Con ello tendremos establecida la conexión.

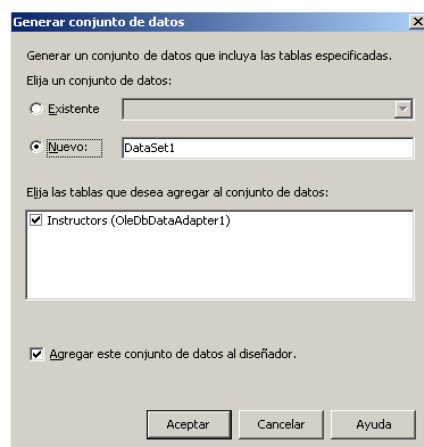
Adaptador de Datos.

El siguiente paso consiste en la creación del adaptador de datos. Para ello tenemos dos métodos:

- Arrastrar el componente deseado a nuestro formulario. Por ejemplo si deseamos utilizar una tabla al completo.
- Utilizar el control *OleDbDataAdapter* de las herramientas de Datos, que es un asistente que nos permitirá detallar más aspectos de los datos que deseamos utilizar. Nos guiará en los pasos para realizar la selección de datos, pudiendo utilizar el Generador de consultas SQL.

Conjunto de Datos.

Una vez que tenemos los datos con los que deseamos trabajar, deberemos crear un objeto que represente dichos datos, que es lo que conocemos como *Dataset*.²



Mediante *Menú Datos-Generar conjunto de datos*,

Se puede dar un nombre al conjunto de datos nuevo o utilizar uno nuevo existente.

En la carpeta donde se ubica la aplicación se creará un fichero con extensión ‘.xsd’, que es un esquema XML de datos.

Con estos pasos ya tenemos disponibles los datos para poder ser utilizados con los controles que necesitemos en nuestra aplicación.

² Existe otro control llamado *DataReader*, que permite leer datos de forma similar pero para cuando necesitamos consultas más puntuales y que no necesitan de una conexión tan constante a la fuente de datos.



Enlazado de Controles y Datos

Una vez que tenemos el DataSet, podemos enlazar dichos datos a los controles que lo permitan, como por ejemplo TextBox, mediante la categoría DataBindings y en su propiedad Text, especificar el campo al que se desea enlazar.

Para utilizar los datos se deberá escribir el código pertinente para su carga, consiste en dos líneas:

```
Nombre_DataSet.Clear()           `Para que no se acumulen datos.  
Nombre_Adaptador.Fill(Nombre_DataSet)  `Carga los datos al Adaptador.
```

Existe un objeto *CurrencyManager*, para cada *DataSet*, que nos permitirá saber el registro actual y el número total de registros.

Cada Formulario tiene un objeto *BindgContext*, que guarda información de todos los objetos *CurrencyManager*.

Enlazado de un Texbox.

Modificar la propiedad Databindings, Text y elegir el campo en concreto.

Enlazado de un Listbox.

Propiedades a modificar:

- DataSource, elegimos la fuente de datos.
- DisplayMember, elegimos el campo por el que queremos cargar el ListBox.

Utilización conjunta de controles enlazados a datos.

Podemos hacer un sencillo programa de consulta de datos, utilizando un listbox que nos muestre los datos por un determinado campo. Y al seleccionar uno de ellos, que se visualicen en sus correspondientes Textbox, los datos de todos los campos correspondientes al seleccionado.

Esto se realiza de forma automática teniendo enlazado el Listbox y los demás Textbox al conjunto de datos.

Moverse por los registros.

En el Formulario donde tenemos los controles enlazados al conjunto de datos, existe un objeto llamado *BindingContext* al que especificando el conjunto de datos y la tabla correspondiente, nos permitirá saber la posición y cantidad de registros de los que disponemos en el conjunto de datos.

BindingContext pertenece a la clase *BundingManagerBase* que se encarga de mantener sincronizados todos los controles que están enlazados al mismo origen de datos.

Para ello tenemos las dos propiedades siguientes:

- **.position**, que nos informa de la posición del registro activo en un momento dado. Hay que tener en cuenta que la primera posición es la 0.
- **.count**, que nos informa del número total de registros que tenemos en la tabla. Nos servirá para saber la posición del último registro.




Con estas propiedades nos bastará para movernos atrás, adelante, al primero y al último.

Por ejemplo, si tenemos un conjunto de datos llamado “Ds_Instructores” y una tabla el mismo denominada “Instructores”, nos moveremos:

- Al siguiente:
`Me.BindingContext(Ds_Instructores, "Instructores").Position =
Me.BindingContext(Ds_Instructores, "Instructores").Position + 1`
O mejor, utilizando el operador abreviado:
`Me.BindingContext(Ds_Instructores, "Instructores").Position += 1`
- Al anterior:
`Me.BindingContext(Ds_Instructores, "Instructores").Position -= 1`
- Al primero:
`Me.BindingContext(Ds_Instructores, "Instructores").Position = 0`
- Al último:
`Me.BindingContext(Ds_Instructores, "Instructores").Position = _
Me.BindingContext(Ds_Instructores, "Instructores").Count - 1`

2. Uso del control DataGrid.

Tenemos el control DataGrid , que nos permite visualizar los datos en una rejilla de filas y columnas.

Básicamente consiste en colocar el control en el formulario donde tenemos establecida previamente una conexión y un conjunto de datos y cambiar las siguientes propiedades del DataGrid:

- **.datasource**, donde especificaremos el origen de los datos (el Dataset).
- **.datamember**, donde indicamos la tabla, consulta o elemento de la conexión del conjunto de datos.

Una vez que tenemos enlazado el DataGrid correctamente, podemos hacer que el usuario pueda realizar cambios y que estos se reflejen en la base de datos original. Para ello, utilizaremos las siguientes propiedades y métodos:

- **.readonly**, propiedad del DataGrid que con valor a False permite que el usuario pueda modificar los datos de la rejilla, aunque esto no implica que esos cambios se reflejen en la base de datos original.
- **.update**, método del adaptador de datos que hace que se actualicen los cambios del DataSet en la base de datos original. Tiene la siguiente sintaxis.

```
Nombre_Adaptador.Update(Nombre_Dataset)
```

Hay que tener en cuenta que hay ciertas operaciones como moverse por los registros, que implican un .update automático.

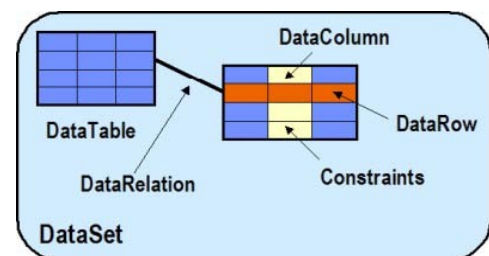
Otras propiedades del Control DataGrid.

- *AllowSorting*. Propiedad booleana que especifica si el control permite ordenar los registros por cualquiera de los campos, con la simple pulsación en el encabezado correspondiente.
- *PreferredColumnWidth* y *PreferredRowHeight*. Especifican alto y ancho de filas y columnas.
- *ColumnHeaderVisible*, boolean para mostrar u ocultar los encabezados.

- *HeaderBackColor*, *HeaderForeColor* y *HeaderFont*. Permiten cambiar las características de los encabezados.
- *GridLineColor* y *GridLineStyle*, permiten configurar las líneas de división la rejilla.
- *Autoformat*. Nos permite seleccionar entre una serie de formatos de presentación predeterminados. Para ello pulsamos con el botón derecho sobre el DataControl y elegimos Formato Automático.
- *TableStyles*. Nos permite editar una colección de estilos personalizados. Permitiendo cambiar propiedades como:
 - *MappingName*. Donde se debe especificar el campo adecuado de la tabla.
 - *GridColumnStyles*. Es otra colección para personalizar el estilos de cada columna de la tabla.

3. Utilización de los de datos mediante código.

Teniendo en cuenta que un DataSet está constituido de acuerdo al espacio de nombres que se muestra en la figura:



Podemos obtener la información que deseemos, utilizando su correspondiente clase.

Propiedades y métodos del DataSet.

- **.HasChanges**,
- **.Tables**,
 - **.Rows**,
 - **.Getchanges**
 - **.AcceptChanges**.
- **.Clear**
- **.DataRowCollection**,

Respecto a las operaciones de edición, debemos utilizar los miembros del objeto tabla del DataSet. Una vez terminado el proceso de edición, actualizaremos el almacén de datos original con el contenido del DataSet, empleando el DataAdapter.

Consultar un campo de un registro.

Para referirnos a un dato concreto de un registro, podemos utilizar la siguiente sintaxis:

```
Dataset.Tables("NombreTabla").Rows(NumReg).item(numCampo)
```

Recorrer el DataSet.

En el siguiente ejemplo³ se ofrece un sencillo ejemplo de creación de un objeto DataSet que llenaremos con un DataAdapter. Una vez listo el DataSet, recorreremos los datos que contiene y mostraremos valores de sus columnas en un ListBox.

```
' crear conexión
Dim oConexion As New SqlConnection()
oConexion.ConnectionString = "Server=(local);Database=Northwind;uid=sa;pwd="
```

³ Tomado de "Programación en Visual Basic.net", L.M. Blanco, Ed. Eidos.



```
' crear adaptador
Dim oDataAdapter As New SqlDataAdapter("SELECT * FROM Customers ORDER BY
ContactName", oConexion)

' crear conjunto de datos
Dim oDataSet As New DataSet()

oConexion.Open()
' utilizar el adaptador para llenar el dataset con una tabla
oDataAdapter.Fill(oDataSet, "Customers")
oConexion.Close()

' una vez desconectados, recorrer la tabla del dataset
Dim oTabla As DataTable
oTabla = oDataSet.Tables("Customers")

Dim oFila As DataRow

    For Each oFila In oTabla.Rows
        ' mostrar los datos mediante un objeto fila
        ' cargamos un listbox con los datos de la fila
        Me.ListBox1.Items.Add(oFila.Item("CompanyName") & _
            " - " & oFila.Item("ContactName") & " - " & _
            oFila.Item("Country"))
    Next
```

Añadir Registros

En el siguiente ejemplo tenemos un DataSet llamado Instructores1, construido sobre una tabla “Instructors” con un campo denominado “Instructor”. En el siguiente procedimiento se añade un nuevo registro y se introduce un dato que se encuentra en un TextBox1.

```
Private Sub Btn_Nuevo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Btn_Nuevo.Click
```

```
Dim oTabla As DataTable
```

```
oTabla = Instructores1.Tables("Instructors")
```

```
Dim NuevaFila As DataRow
```

```
' obtener un nuevo objeto fila de la tabla del dataset
NuevaFila = Instructores1.Tables("Instructors").NewRow
TextBox1.Text = ""
TextBox1.Enabled = True
TextBox1.Focus()
```

```
' asignar valor a los campos de la nueva fila
NuevaFila("Instructor") = TextBox1.Text
```

```
' añadir el objeto fila a la colección de filas de la tabla del dataset
oTabla.Rows.Add(NuevaFila)
End Sub
```

Editar Registros

Para editar los datos de un DataSet, no tenemos más que asignar los nuevos valores a su correspondiente campo del DataRow.

```
Private Sub btnModificar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnModificar.Click
dim reg_activo as integer
```



```
Dim oDataRow As DataRow
' obtener el objeto fila de la tabla del dataset
' en el que estamos posicionados
reg_activo = Me.BindingContext(DS_Instructores, "Instructores").Position
oDataRow = Me.oDataSet.Tables("Clientes").Rows(reg_activo)

' modificar las columnas de la fila
' excepto la correspondiente al identificador cliente
oDataRow("Nombre") = Me.txtNombre.Text
oDataRow("FIngreso") = Me.txtFIngreso.Text
oDataRow("Credito") = Me.txtCredito.Text

End Sub
```

Eliminar Registros

Para eliminar registros tenemos el método *.Delete*,

Formato:

Nombre_ObjetoDataRow.Delete()

Ejemplo:

En el siguiente ejemplo vemos el código del botón Eliminar, dentro del cual, obtenemos la fila a borrar mediante un objeto DataRow, procediendo a su borrado con el método Delete().

Para actualizar los borrados realizados, empleamos el método GetChanges() del objeto DataTable, obteniendo a su vez, un objeto tabla sólo con las filas borradas; información esta, que pasaremos al DataAdapter, para que actualice la información en el origen de datos.

```
Private Sub btnEliminar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEliminar.Click
    Dim oDataRow As DataRow
    ' obtener el objeto fila, de la tabla del dataset
    ' en el que estamos posicionados
    oDataRow = Me.oDataSet.Tables("Clientes").Rows _
    (Me.BindingContext(Ds_Instructores, "Instructores").Position)
    oDataRow.Delete()
    ' borrar la fila
    ' mediante el método GetChanges(), obtenemos una tabla
    ' sin las filas borradas
    Dim oTablaBorrados As DataTable

    oTablaBorrados = _
    Me.oDataSet.Tables("Clientes").GetChanges(DataRowState.Deleted)

    'Aquí podríamos preguntar al usuario si de verdad quiere eliminar el
    ' registro. En cuyo caso pasaríamos ala siguiente instrucción de
    ' actualización de la base de datos.

    ' actualizar en el almacén de datos las filas borradas
    Me.oDataAdapter.Update(oTablaBorrados)

    ' confirmar los cambios realizados
    Me.oDataSet.Tables("Clientes").AcceptChanges()

    ' Habrá que reposicionar a un registro concreto, por ejemplo el primero
    Me.BindingContext(oDataset, "Instructores").Position = 1

End Sub
```




Actualizar los datos origen.

Para que se haga un volcado del DataSet a la base de datos original, debemos indicarlo explícitamente mediante el método *.update* del adaptador. Cuando damos esta orden el adaptador de datos se encargará de revisar los cambios habidos y reflejarlos en los datos origen.

Formato:

Nombre_Adaptador.Update(Nombre_Dataset, "Elemento del dataset")

Ejemplo:

```
Private Sub btnActualizar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnActualizar.Click
    ' actualizar los cambios realizados en el dataset
    ' contra la base de datos real
    Me.oDataAdapter.Update(Me.oDataSet, "Clientes")
End Sub
```

Buscar datos.

Tenemos dos métodos para realizar la búsqueda de registros, uno utiliza un método y solo busca en el campo clave, el otro utiliza una sentencia sql.

- Utilizando el **método Find** del objeto DataRowCollection, el cual nos permite hacer una búsqueda por un campo clave.

Formato:

Nombre_DataRowCollection.Find(valor de campo clave)

Si hemos generado el Dataset mediante código y nos da problemas al utilizarlo en la búsqueda, podemos generar un nuevo dataset mediante el asistente para utilizarlo en la búsqueda con el *.Find*.

Previamente se deberá haber cargado el dataset generado con la tabla correspondiente.

Ejemplo:

El siguiente ejemplo busca el registro cuyo campo clave sea igual al valor introducido en el TextBox3. Tenemos un Dataset (DS_Instructores) y una tabla (Instructores) con un campo clave (InstructorID).

```
Private Sub btn_Busca_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_Busca.Click
    Dim miTabla As DataTable = DS_Instructores.Instructores
    Dim FilaEncontrada As DataRow
    FilaEncontrada = miTabla.Rows.Find(TextBox3.Text)
    'Realiza la busqueda
    If FilaEncontrada Is Nothing Then
        MsgBox("No se encontró nada")
    Else
        MsgBox(FilaEncontrada("Instructor"))
        'Muestra el campo Instructor
        Me.BindingContext(DS_Instructores, "Instructores").Position = FilaEncontrada("InstructorID") - 1
    End If
End Sub
```



```
        `Se posiciona en el registro encontrado.  
    End If  
End Sub
```

Cuando la tabla sobre la que queremos buscar tiene más de un campo clave, se puede dimensionar un array con tantos elementos como campos claves y asignar a cada uno el valor del campo que buscamos y poner como objeto de la búsqueda dicho array, por ejemplo:

```
Private Sub FindInMultiPKey(ByVal myTable As DataTable)  
    Dim FilaEncontrada As DataRow  
    ' Create an array for the key values to find.  
    Dim findTheseVals(2) As Object  
    ' Set the values of the keys to find.  
    findTheseVals(0) = "John"  
    findTheseVals(1) = "Smith"  
    findTheseVals(2) = "5 Main St."  
    FilaEncontrada = myTable.Rows.Find(findTheseVals)  
    ' Display column 1 of the found row.  
    If Not (FilaEncontrada Is Nothing) Then  
        msgbox(FilaEncontrada(1).ToString())  
    End If  
End Sub
```

- Utilizando el **método Select** de un objeto Tabla,

Formato:

Nombre_Objeto_Tabla.Select(criterio de clausula Where sql)

El criterio sql debe ser equivalente a las sentencias que pueden seguir a una cláusula Where de sql.

La principal fuente de errores que nos puede dar, proviene de la generación de la cadena a utilizar en el criterio de búsqueda. Cuando queremos utilizar variables o literales, deben ir entre comillas simples.

Así mismo hay que tener en cuenta que la operación devuelve un array con las filas que coinciden con el criterio sql especificado, por lo que nos debemos asegurar que hemos declarado un objeto capaz de contener un array de registros, así que será del tipo **DataRow()**.

Para ver si ha tenido éxito podemos consultar el valor máximo de dicho array (GetUpperBound), si fuese -1 es que no se ha encontrado nada.

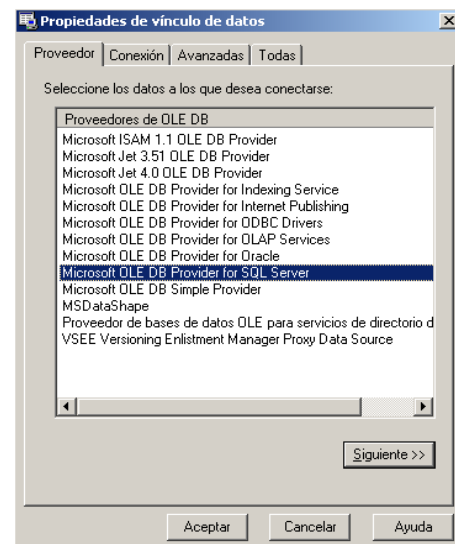
Ejemplo:

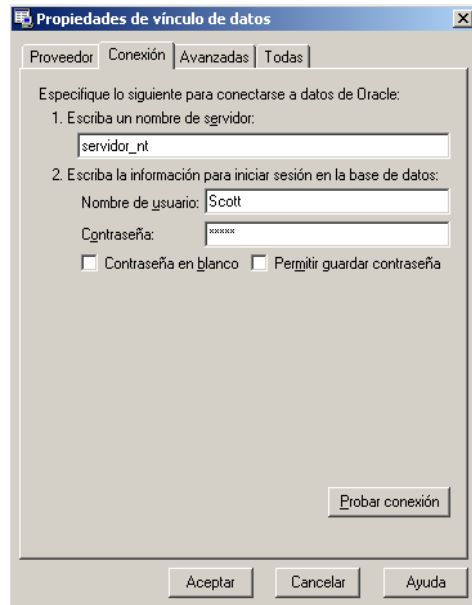
```
If TextBox1.Text = Nothing Then
    MsgBox("Necesitas introducir un texto de búsqueda")
Else
    Dim tabla as DataTable = DataSet2.Datos
    Dim nuevafila As DataRow()
    Dim sentencia As String
    Dim foundRows As DataRow()
    Dim campo as string = "Cod"
    sentencia = campo & " = " & TextBox1.Text & ""
    nuevafila = tabla.Select(sentencia)
    If nuevafila.GetUpperBound(0) Is -1 Then
        MsgBox("No se encontró ninguna coincidencia")
    Else
        MsgBox(nuevafila(0).Item("Cod") & " " & _
            nuevafila(0).Item("Nombre") & " " & nuevafila(0).Item("Telefono"))
    End If
End If
```

4. Conexión a Oracle.

Para conectar con bases de datos Oracle se procede de forma similar a lo explicado en anteriores apartados.

1. Elegimos el proveedor adecuado. Por defecto Visual Studio trae un controlador para Oracle (Microsoft OLE DB Provider for Oracle





Especificaremos:

- Nombre del servidor de la base de datos Oracle.
- Nombre de usuario y contraseña con el que solicitaremos la conexión.

Probamos la conexión con el fin de asegurarnos que están bien detallados los datos para la misma.

Aceptamos y quizás nos pida de nuevo la contraseña.

Una vez realizada la conexión, Crearemos el Adaptador de datos. (En Cuadro de Herramientas-Datos).

Seguiremos los pasos del Asistente

- Eligiendo la conexión antes creada.
- Especificando la sentencia Sql, o generándola.

Nos aparecerán todas las tablas del Servidor de Oracle, especificando entre paréntesis el nombre del usuario al que pertenecen. Las que no se especifica ese datos son las propias del usuario especificado en la conexión

A continuación generaremos el conjunto de datos, como de costumbre.

a. Bibliografía.

- “*VB.Net. Aprenda Ya*”, Michael Halvorson, de McGraw-Hill
Cap 19 pág, 517
Comenzar a utilizar ADO.NET.
- ”*La Biblia de VBNet*”, E. Petroustos. Ed.Anaya Multimedia.
Pág. 1252 y siguientes
Trae un ejemplo de cómo hacer un Datagrid con dos tablas relacionadas.
- “*Programación en Visual Basic.net*”, L.M. Blanco, Ed. Eidos.
- ”*El lenguaje de programación Visual Basic.Net*”, F.J. Ceballos. Ed. Ra-Ma.
Pág.347 y siguientes.