12. Acceso a Bases de Datos con VB.NET

Una base de datos es un sistema formado por un conjunto de datos relacionados y almacenados en discos que permiten el acceso directo a ellos y una serie de programas que manipulan ese conjunto de datos (SGBD – sistema de gestión de bases de datos). Cada base de datos se compone de una o más tablas que guardan los datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que se quiere guardar en la tabla, cada fila de la tabla conforma un registro. Un registro contiene campos que pueden ser del mismo tipo dato o de diferentes tipos de datos. Entre las principales características de las bases de datos se pueden mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoria.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

12.1. Tipos de Bases de Datos.

Los tipos de bases datos más comunes son las relacionales y en el mercado existen varios sistemas de administración de bases de datos de ese tipo, algunos son: SQL Server, Access, Oracle, MySQL, PostgresSQL, etc.

12.1.1. Relacionales.

Son las que más se utilizan. Las bases de datos relacionales son un conjunto de tablas relacionadas entre sí, cada tabla está definida por una serie de campos. Los campos forman las columnas de las tablas; estos definen el tipo y la variedad de los datos. Las filas de datos se denominan registros (tuplas), cada tipo definido en un registro se le denomina atributo. Las tablas pertenecientes a una base de datos pueden relacionarse entre sí utilizando campos clave comunes entre las tablas.

12.1.2. Orientada a objetos.

El esquema de una base de datos por objetos está representado por un conjunto de clases que definen las características y el comportamiento de los objetos que poblarán la base de datos. Con una base de datos orientada a objetos, los objetos memorizados en la base de datos contienen tanto los datos como las operaciones posibles con tales datos. En cierto sentido, se podrá pensar en los objetos como en datos a los que se les ha puesto una inyección de inteligencia que les permite saber cómo comportarse, sin tener que apoyarse en aplicaciones externas.

12.2. Lenguaje de Consulta Estructurado (S.Q.L.).

Es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos. El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

12.2.1. Comandos S.Q.L.

Existen dos tipos de comandos SQL:

- **DLL**: que permiten crear y definir nuevas bases de datos, tablas, campos e índices.
- **DML**: que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Tabla 12.1. Comandos DLL y DML de SQL

Comandos DLL	
Comando	**Descripción**
CREATE	Utilizado para crear nuevas bases de datos, tablas, campos e índices.
DROP	Empleado para eliminar bases de datos, tablas e índices.
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.
Comandos DML	
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados.
DELETE	Utilizado para eliminar registros de una tabla de una base de datos.

12.2.2. Cláusulas S.Q.L.

Las cláusulas son condiciones de modificación utilizadas para definir los datos que se desean seleccionar o manipular.

Tabla 12.2. Cláusulas SQL

Cláusula	Descripción
FROM	Para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Para especificar las condiciones que deben reunir los registros a seleccionar.

GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Para ordenar los registros seleccionados.

12.2.3. Operadores lógicos S.Q.L

Los operadores lógicos comprueban la veracidad de alguna condición. Éstos devuelven el tipo de datos **Boolean** con el valor **TRUE** o **FALSE.**

Tabla 12.3. Operadores lógicos SQL.

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un patrón.
IN	Utilizado para especificar registros de una base de datos.
ALL	Devuelve True si el conjunto de comparaciones en verdad.

12.2.4. Operadores de comparación S.Q.L.

Los operadores de comparación comprueban si dos expresiones son iguales, devolviendo un valor booleano **True** o **False**. Se pueden utilizar en todas las expresiones excepto en las de los tipos de datos **text**, **ntext** o **image**.

Tabla 12.4. Operadores de comparación SQL

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que

12.2.5. Funciones de agregado S.Q.L.

Las funciones de agregado realizan un cálculo sobre un conjunto de valores y devuelven un solo valor. Si exceptuamos la función **COUNT**, todas las funciones de agregado ignoran los valores NULL. Las funciones de agregado se suelen utilizar con la cláusula **GROUP BY** de la instrucción **SELECT**.

Tabla 12.5. Funciones de agregado SQL

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de la selección.
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado.
MAX	Utilizada para devolver el valor más alto de un campo especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado.

12.3. Sentencias básicas S.Q.L

Se describirá muy brevemente algunas de las sentencias SQL para la manipulación de los datos de una tabla. Para trabajar las sentencias a continuación, se supondrá que se tiene creada una tabla llamada **usuarios** con la siguiente estructura:

Tabla 12.6. Estructura de los campos de la tabla USUARIOS.

Campo	Tipo de Dato	Longitud
identificacion	varchar	15
Nombres	varchar	20
Apellidos	varchar	20
Dirección	varchar	25
Teléfono	varchar	20
ciudad_nac	varchar	20
fecha_nac	date	

Y que contiene la siguiente información:

Tabla 12.7. Información de la tabla usuarios.

Identificación	Nombres	Apellidos	Dirección	Teléfono	Ciudad_nac	Fecha_nac
100	Carlos	Romero	Cra 7 # 20-10	4152584	Bogota	01/02/1980
101	María	Castro	Calle 25 # 25-10	3692581	Cali	15/03/1984
112	José	Peláez	Av. 35 # 32-45	1234567	Medellín	20/05/1960
114	Cristian	Vanegas	Cra 7 # 29-58	9874561	Manizales	31/08/1974

116	Rosa	Cetina	Calle 17 # 21-14	3571596	Buga	15/12/1985
118	Andrés	Vanegas	Tranvs 48 # 22-10	8527419	Bogotá	10/04/1978
130	Angélica	Morales	Cra 68 # 21-11	6549518	Medellín	20/06/1981
150	Johana	Duarte	Cra 2 # 45-38	9637534	Bogotá	12/06/1988
170	Mario	Vargas	Calle 1 # 99-18	6598743	Medellín	28/08/1980

12.3.1. CREATE DATABASE.

El comando **CREATE DATABASE** permite crear una base de datos. Su formato es:

CREATE DATABASE <nombre_base_de_datos)

12.3.2. DROP DATABASE

El comando **DROP DATABASE** permite eliminar una base de datos que previamente se había creado. Su formato es:

DROP DATABASE <nombre_base_de_datos_a_eliminar>

12.3.3. CREATE TABLE

El comando **CREATE TABLE** permite crear una tabla. Con este comando se especifica el nombre de la tabla, las columnas y sus tipos de datos, las claves primarias y si es necesario la clave externa. Su formato es:

CREATE TABLE nombre_tabla (campo1 tipodato DEFAULT,., campo2 tipodato, DEFAULT)

La cláusula **DEFAULT** indica la característica de cada columna: NOT NULL (no permite valores nulos), UNIQUE (dos filas no pueden tener un mismo valor en la misma columna), PRIMARY KEY (define una columna como clave principal).

12.3.4. DROP TABLE

El comando **DROP TABLE** permite eliminar una tabla que previamente se había creado. Su formato es:

DROP TABLE nombre_tabla

12.3.5. INSERT

La sentencia SQL de inserción de datos **INSERT** permite insertar información en una tabla. Su formato es:

INSERT INTO nombre_tabla (campo1, campo2,….) VALUES (valor1, valor2,….)

Para insertar un nuevo registro a la tabla **usuarios** se debería realizar la siguiente sentencia:

INSERT INTO usuarios (identificación, nombres, apellidos, dirección, teléfono, ciudad_nac, fecha_nac) VALUES (‘160’, ’Carmen’, ’Bolívar’, ’Calle 100 # 115-55’,’2014201’, ’Barranquilla’, ’18/11/1692’)

12.3.6. ALTER

La sentencia SQL **ALTER** permite insertar un nuevo campo en una tabla. Su formato es:

```
ALTER TABLE  nombre_tabla ADD nombre_campo tipo_de_dato ()
```

Para insertar un nuevo campo a la tabla **usuarios** llamado **credito** de tipo numérico se debería realizar la siguiente sentencia:

```
ALTER TABLE usuarios ADD credito numeric (18,0)
```

12.3.7. SELECT

La sentencia SQL que más se utiliza es la instrucción de selección **SELECT**. Como su nombre lo indica es una instrucción que permite seleccionar información de una tabla. Su formato es:

```
SELECT campos_tabla FROM nombre_tabla
```

A continuación se realizan algunos ejemplos:

a. Para visualizar toda la información que contiene la tabla **usuarios** se puede incluir con la instrucción SELECT el carácter '*' o cada uno de los campos de la tabla.

```
SELECT * FROM usuarios
        O
SELECT identificación, nombres,…….. FROM usuarios
```

b. Para visualizar solamente la identificación del usuario

```
SELECT identificacion FROM usuarios
```

c. Si se desea obtener los registros cuya identificación sea menor o igual a 116, se debe utilizar la cláusula WHERE que especifica las condiciones que deben reunir los registros que se van a seleccionar.

```
SELECT * FROM usuarios WHERE identificación<='116'
```

d. Si se desea obtener los registros cuyos nombres sean Andrés o Cristian, se debe utilizar el operador IN que especifica los registros que se quieren visualizar de una tabla.

```
SELECT nombres FROM usuarios WHERE nombres IN ('Andres','Cristian')
```

O se puede utilizar el operador OR

```
SELECT * FROM usuarios WHERE nombres='Andrés' OR nombres='Cristian'
```

e. Si se desea obtener los registros cuya identificación sea menor de '130' y la ciudad sea 'Bogota', se debe utilizar el operador AND.

```
SELECT * FROM usuarios WHERE identificación<='130' AND ciudad='Bogota'
```

f. Si se desea obtener los registros cuyos nombres empiecen por la letra ‘C’, se debe utilizar el operador LIKE que utiliza los patrones ‘%’ (todos) y ‘_’ (carácter).

```
SELECT * FROM usuarios WHERE nombres LIKE ‘C%’
```

g. Si se desea obtener los registros cuyos nombres contenga la letra ‘i’.

```
SELECT * FROM usuarios WHERE nombres LIKE ‘%i%’
```

h. Si se desea obtener los registros donde la segunda letra del nombre sea una ‘o’.

```
SELECT * FROM usuarios WHERE nombres LIKE _o%’
```

i. Si se desea obtener los registros cuya identificación este entre el intervalo 116 y 140, se debe utilizar la cláusula BETWEEN, que sirve para especificar un intervalo de valores.

```
SELECT * FROM usuarios WHERE identificación BETWEEN ’116’ AND ‘140’
```

12.3.8. DELETE

La sentencia SQL de eliminación de datos **DELETE** permite borrar todos o un grupo especifico de registros de una tabla. Su formato es:

```
DELETE FROM nombre_tabla
```

A continuación se realizarán algunos ejemplos:

a. Para eliminar todos los registros de la tabla **usuarios.**

```
DELETE FROM usuarios
```

b. Para eliminar solamente los registros cuya identificación sea mayor de ‘150’.

```
DELETE FROM usuarios WHERE identificación >’150’
```

c. Para eliminar los registros diferentes del nombre “Cristian”

```
DELETE FROM usuarios WHERE nombres NOT IN (‘Cristian’)
```

12.3.9. UPDATE

La sentencia SQL de actualización **UPDATE** permite actualizar un campo de una tabla. Su formato es:

```
UPDATE nombre_tabla SET nombre_campo=criterio
```

A continuación se realizan algunos ejemplos:

a. Para actualizar el campo credito con un valor de 100000 en la tabla **usuarios.**

```
UPDATE usuarios SET credito=100000
```

b. Para actualizar el campo credito en 200000 para los registros cuyo nombre empiecen por 'A'.

```
UPDATE usuarios SET  credito=credito +200000 WHERE nombres LIKE 'A%'
```

c. Para actualizar el campo credito en 50000 para los registros cuya ciudad sea igual a 'Bogota'.

```
UPDATE usuarios SET  credito=credito+50000 WHERE ciudad='Bogota'
```

12.3.10. INNER JOIN

Permite recuperar datos de 2 ó más tablas. Cuando se realiza la concatenación de las tablas, no necesariamente se deben mostrar todos los datos de las tablas, o sea, se pueden mostrar los campos de en realidad se desean ver. Su formato es:

```
SELECT tabla1.campo, tabla2.campo, tabla1.campo2,…. FROM tablaprincipal
INNER JOIN tablasecundaria ON campocomuntabla1=campocomuntabla2
```

Para realizar algunos ejemplos explicativos se utilizara la tabla **usuarios** y además se supondrá que se tiene otra tabla llamada **pedidos**, que contendrá la siguiente estructura:

Tabla 12.8. Estructura de los campos de la tabla USUARIOS

Campo	Tipo de Dato	Longitud
nropedido	varchar	15
identificacion	varchar	15
fechacompra	date	20
fechavence	date	25
observacion	varchar	30

Y que contiene la siguiente información:

Tabla 12.9. Información de la tabla usuarios

nropedido	Identificación	fechacompra	fechavence	observacion
10	100	01/02/2006	01/02/2006	Pago de contado
20	101	15/03/2006	15/03/2006	Descuento del 5%
30	100	20/05/2006	20/06/2006	Descuento del 2%
40	112	31/08/2006	31/10/2006	Pago a sesenta días
50	101	15/12/2006	30/12/2006	Pago de contado
60	118	10/04/2006	10/06/2006	Sin descuento
70	101	20/06/2006	20/07/2006	Descuento del 5%
80	100	12/06/2006	12/09/2006	Pago a noventa días
90	101	28/08/2006	28/09/2006	Pago de contado

a. Para visualizar los campos identificación, nombres, apellidos de la tabla **usuarios** y nropedido, fecha de compra, fecha de vencimiento y observación de la tabla **pedidos**, se debe realizar la siguiente instrucción:

```
SELECT usuarios.identificacion, usuarios.nombres, usuarios.apellidos,
pedidos.nropedido, pedidos.fechacompra, pedidos.fechavence, pedidos.observacion
FROM usuarios INNER JOIN pedidos
ON usuarios.identificacion = pedidos.identificacion
```

b. Para visualizar todos campos de las tablas **usuarios** y **pedidos** donde identificación sea igual a 100, se debe realizar la siguiente instrucción:

```
SELECT usuarios.*, pedidos.*
FROM usuarios INNER JOIN pedidos
ON usuarios.identificacion = pedidos.identificacion
WHERE usuarios.identificacion=100
```

12.4. Conexión a bases de datos con VB.NET

Visual Basic .NET utiliza la tecnología **ADO.NET (Activex Data Object)** que permite el acceso a bases de datos mediante proveedores para sistemas administradores de bases de datos que funcionan en el entorno .NET. La plataforma .NET incorpora cuatro proveedores: SQL SERVER, ORACLE, ODBC (Access), OLEDB.

ADO.NET proporciona acceso a orígenes de datos como Microsoft SQL Server y XML, así como a orígenes de datos OLE DB y ODBC. Las aplicaciones para usuarios que comparten datos pueden utilizar ADO.NET para conectar a estos orígenes de datos y recuperar, manipular y actualizar los datos contenidos.

ADO.NET es un conjunto de clases que se encuentran en el archivo **System.Data.dll** y está integrada con las clases del archivo **System.Xml.dll**. Cuando se compila un código que utiliza el espacio de nombres **System.Data** se hace referencia a dichos archivos. Estas clases sirven para separar el acceso a la base de datos y la manipulación de los mismos.

Sus principales clases son:

- **DataSet:**Es el conjunto de datos donde se pueden incluir una o más tablas con la información acerca de las relaciones entre estas, y las restricciones que puedan tener los datos.

- **DataTable**: Permite la manipulación de los datos en la memoria y realiza operaciones como la exploración, ordenación, edición, aplicación de filtros, creación de vistas, etc.

- **DataView**: Permite representar los datos de la clase **DataTable**, creando múltiples vistas de los mismos.

Los proveedores de datos proporcionan el puente entre las bases de datos y las aplicaciones. Los principales objetos de un proveedor de datos .NET son:

- **Connection**: Sirve para establecer una conexión con una base de datos. Se utiliza **SqlConnection** para una conexión a SQL Server, **OleDbConnection** para una conexión a Access y **OracleConnection** para Oracle.

- **Command**: Sirve para ejecutar sentencias SQL y devolver resultados de una base de datos. Se utiliza **SqlCommand** para una conexión a SQL Server, **OleDbCommand** para una conexión a Access y **OracleCommand** para Oracle.

- **DataAdapter**: Es el adaptador de datos, el cual es un conjunto de objetos para intercambiar datos entre una base de datos y un conjunto de datos. Se utiliza **SqlDataAdapter** para una conexión a SQL Server, **OleDataAdapter** para una conexión a Access y **OracleDataAdapter** para Oracle.

12.5. Ejemplos prácticos de bases de datos.

12.5.1. Conexión a una base de datos SQL Server por código.

Dentro de un proyecto llamado **ConexionASQLServer,** realizar un programa que permita realizar una conexión a una base de datos de SQL Server llamada bdlibrovbnet.mdf y mostrar los registros de la tabla clientes en un control DataGridView, utilizando código de Visual Basic .NET.

NOTA: para este ejemplo el usuario tiene que tener instalado **Microsoft SQL SERVER 2005** o posterior y crear una base de datos llamada **bdlibrovbnet** y dentro de ella una tabla llamada **clientes** (**Ver anexo A, SQL Server, página 423**).

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **Button**, 2 **Label** y 1 **DataGridView**.

- **Establecer las propiedades de los objetos de la interfaz de usuario.**

Establezca las siguientes modificaciones a los controles:

Tabla 12.10. Propiedades de controles proyecto ConexionASQLServer.

Control	Propiedad	Valor
DataGridView1	Name	datos
Button1	Name	botoncargar

	Text	Cargar registros tabla.
Label1	Name	lblbd
	Text	Conexión a la base de datos bdlibrovbnet.mdf de SQL SERVER.
	Font	True
Label2	Name	lbltabla
	Text	Tabla : Clientes
	Font	True
Form1	Name	formulario
	Text	Conexión a una base de datos SQL SERVER.

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 12.1. Interfaz de usuario ConexionASQLServer.

- **Escribir código**

a) Antes de la apertura de la clase **formulario** se debe importar el espacio de nombres System.Data.SqlClient:

```
Imports System.Data.SqlClient
Public Class Formulario
      ....
      ......
End Class
```

El espacio de nombres **System.Data.SqlClient** es necesario para utilizar las diferentes clases que permitan las operaciones con bases de datos, en este caso, SQL Server.

b) Seleccione el objeto **botoncargar,** de doble clic para abrir el editor de código y escriba

el siguiente código:

```
Dim conexion As String
conexion = "Data Source=(local)\SQLEXPRESS;Database=bdlibrovbnet;
                              Integrated Security=True"
Dim seleccion As String = "SELECT * FROM clientes"
Dim adaptadordedatos As SqlDataAdapter
Dim tabladedatos As New DataTable
Try
  adaptadordedatos = New SqlDataAdapter(seleccion, conexion)
  adaptadordedatos.Fill(tabladedatos)
  tabla.DataSource = tabladedatos
Catch ex As Exception
  MsgBox("Error: " & ex.Message)
End Try
```

Se crea una variable llamada **conexion** de tipo **String** que contendrá la cadena de conexión a la base de datos **bdlibrovbnet.mdf** de SQL Server. La cadena de conexión debe contener como mínimo los siguientes parámetros:

- **Data Source**: Se le asigna la ruta donde se encuentra el servidor SQL Server, en este caso, SQL Server se instalo en el computador de trabajo por lo cual el nombre del servidor es **local**.
- **Database**: Se le asigna el nombre de la base de datos a la que se quiere conectar.
- **Integrated Security**: Se le asigna **True** o **False** para determinar si la seguridad es integrada o no.

Luego se crean los siguientes objetos: **seleccion** de tipo **String** a la cual se le asigna la información que se quiere mostrar, en este caso, todos los registros de la tabla **clientes** (Select * from Clientes); **adaptadordedatos** de tipo **SqlDataAdapter** el cual será el adaptador de datos para la base de datos a manipular; **tabladedatos** se le asigna espacio de memoria de tipo **DataTable** para guardar los datos en memoria y poder realizar operaciones con dichos datos. En un bloque **Try** se le asigna espacio de memoria de tipo **SqlDataAdapter** al objeto **adaptadordedatos**, al cual se le envía como parámetros los objetos **seleccion** (datos a mostrar) y **conexion** (cadena de conexión), luego se rellena (**fill**) el adaptador de datos con la tabla de datos (**tabladedatos**) y por ultimo al objeto **datos** en su propiedad **DataSource** se le establece el conjunto de datos que se van a mostrar al asignársele la tabla de datos. Por el bloque **Catch** se establece un mensaje en caso de que ocurra un error.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto y pulsar el botón **Cargar registros tabla**, se visualizara el formulario con los registros de la tabla clientes:

Figura 12.2. Formulario con los registros de la tabla clientes.

12.5.2. Conexión a SQL Server utilizando el Explorador de servidores.

Elaborar un proyecto llamado **ConexionBDExploradorServidores** y realizar un programa que permita a un usuario realizar una conexión a una base de datos de SQL Server y mostrar los registros de una tabla en un objeto **DataGridView** utilizando el Explorador de servidores.

NOTA: para este ejemplo se utilizara la base de datos SQL Server llamada **bdlibrovbnet.mdf** (**Ver anexo A, SQL Server, página 423**). De esta base de datos de utilizara la tabla clientes.

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **Label** y 1 **DataGridView**.

- **Establecer las propiedades de los objetos de la interfaz de usuario.**

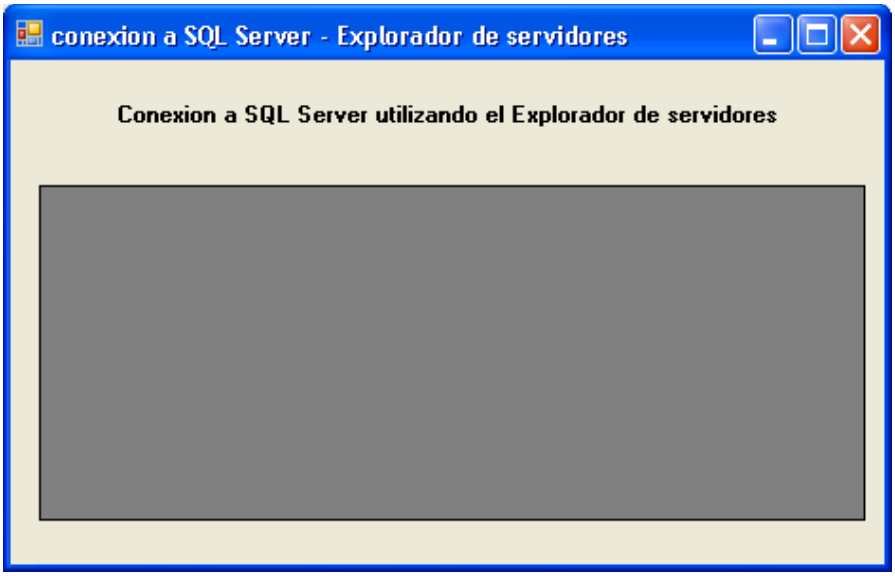
Establezca las siguientes modificaciones a los controles:

Tabla 12.11. Propiedades de los controles de ConexionBDExploradorServidores.

Control	Propiedad	Valor
DataGridView1	Name	datos
Label1	Name	lblbd
	Text	Conexión a SQL Server utilizando el Explorador de servidores.
	Font	True
Form1	Name	formulario
	Text	Conexión a SQL Server - Explorador de servidores.

La interfaz de usuario queda como se muestra en la siguiente figura:

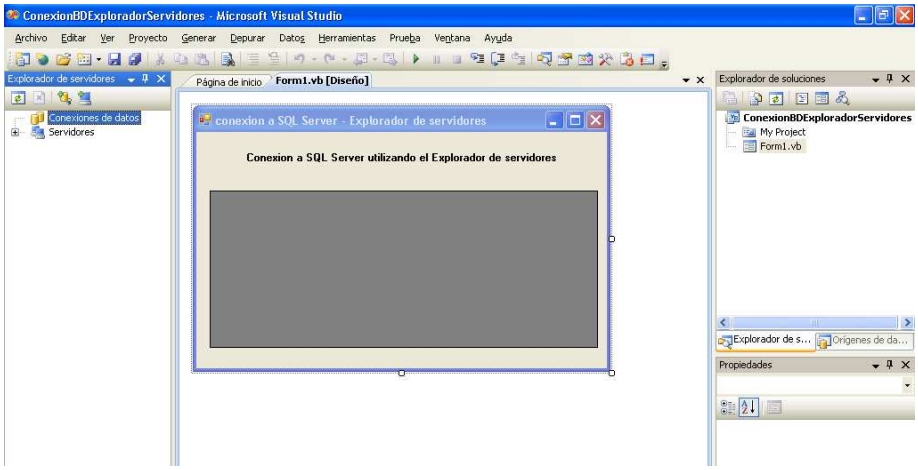
Figura 12.3. Interfaz de usuario (ConexionBDExploradorServidores).



- **Explorador de servidores.**

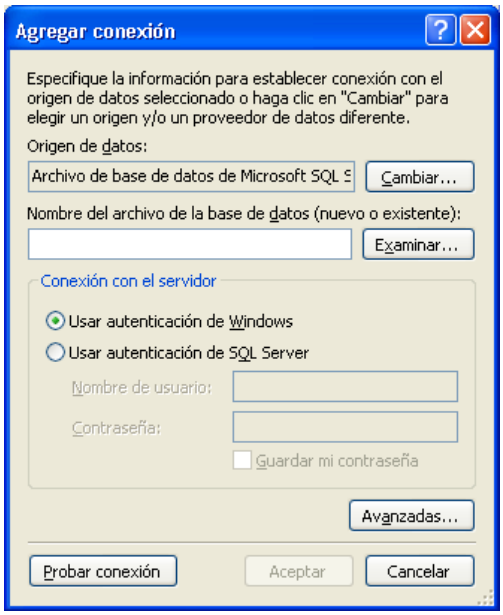
Del menú **Ver** seleccione la opción **Explorador de servidores** o pulse simultáneamente las teclas Ctrl+Alt+S, para visualizar la siguiente figura:

Figura 12.4. Ventana del Explorador de servidores.



Pulse el icono **conectar con bases de datos**  , para visualizar la ventana de **Agregar conexión** como lo muestra la figura:

Figura 12.4. Ventana Agregar conexión.

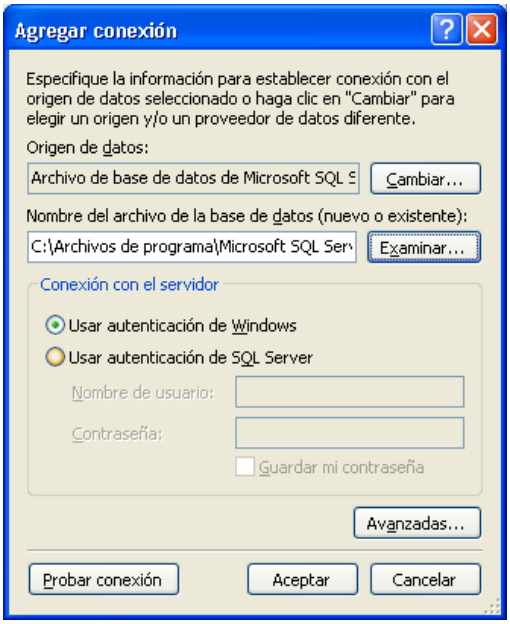


En esta ventana se pueden realizar lo siguiente:

- **Elegir proveedor base de datos:** Se puede elegir el proveedor de la base de datos. Por omisión se cargar el sistema administrador de base de datos SQL Server.
- **Elegir base de datos:** Permite elegir la base de datos a utilizar.
- **Probar la conexión**: Permite saber si una conexión ha sido exitosa o no.
- **Modificar la cadena de conexión**: Con la opción **Avanzadas** se puede modificar la conexión a la base de datos.

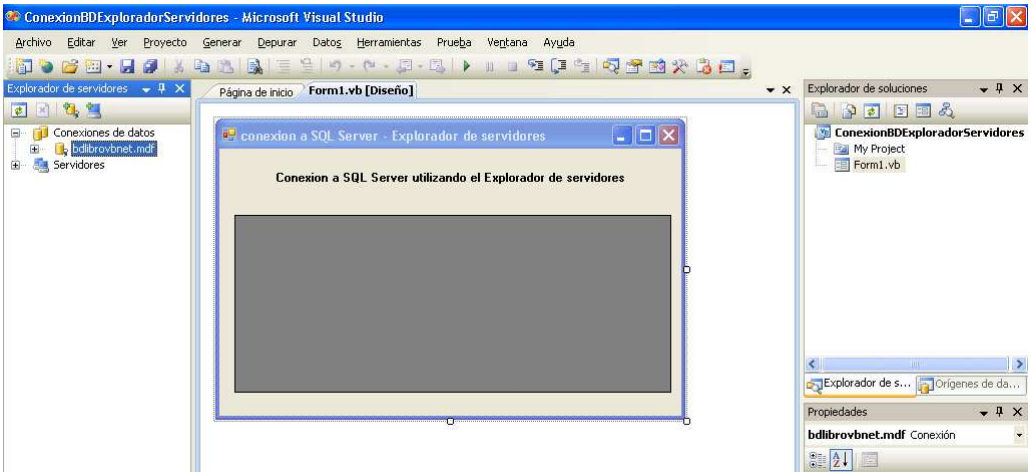
Para el ejemplo seleccione la base de datos **bdlibrovbnet.mdf**, la cual se encuentra en la carpeta **Data** de **SQL Server** (si la instalación la realizo en c:\ la ruta sería: C:\Archivos de programa\Microsoft SQL Server\MSSQL.1\MSSQL\Data.

Figura 12.5. Ventana con la base de datos seleccionada.



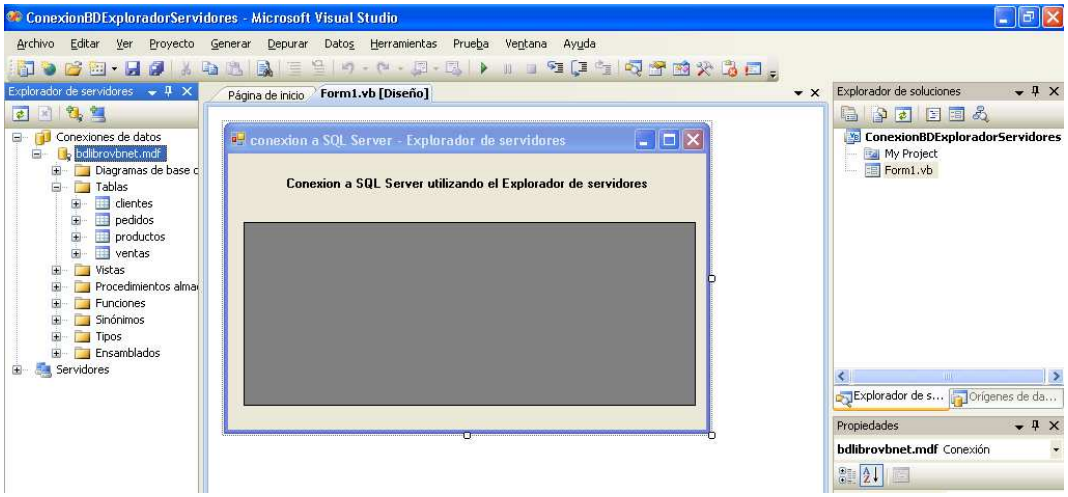
Pulse el botón **Aceptar** para visualizar la siguiente figura:

Figura 12.6. Explorador de servidores con la conexión a una base de datos.



Al pulsar en el signo (+) al lado del nombre de la base de datos se visualizara la estructura definida en la base de datos. Si desea ver las tablas que contiene la base de datos pulse el signo (+) al lado de Tablas.

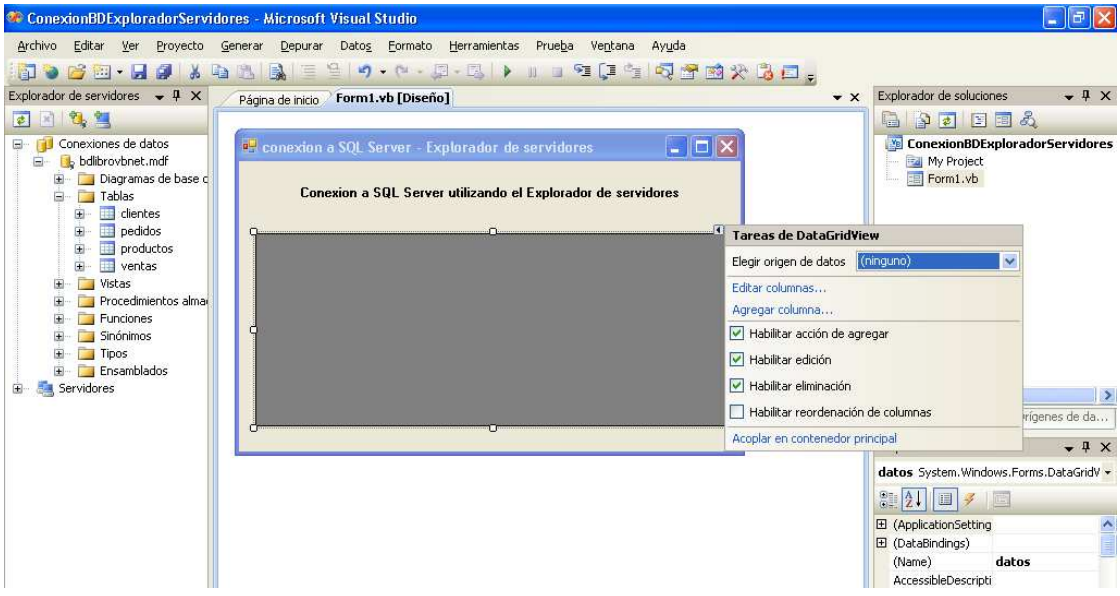
Figura 12.7. Estructura de la base de datos bdlibrovbnet.mdf.



- **Obtener el origen de datos para el DataGridView**

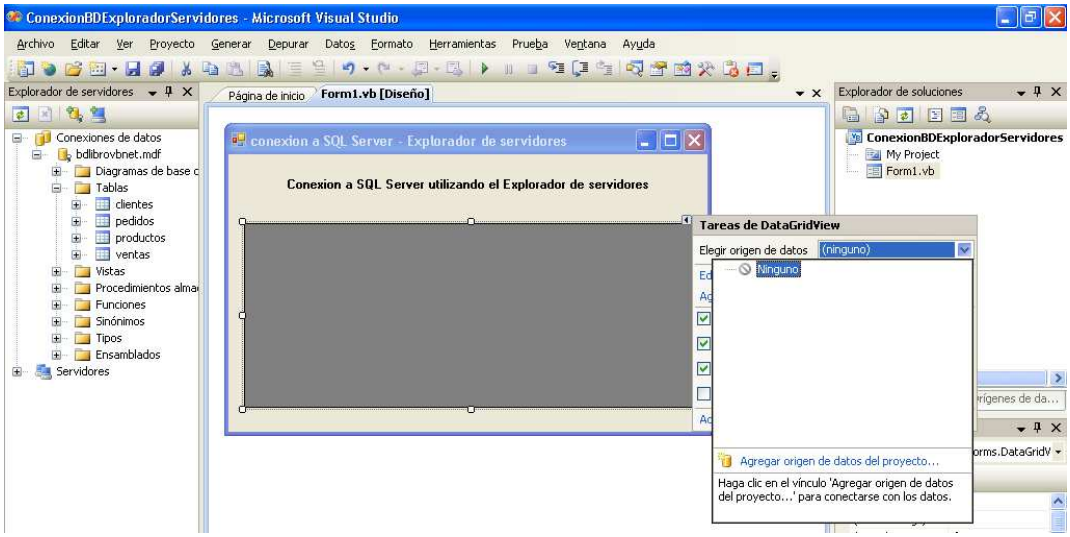
Seleccione el objeto **DataGridView** llamado **datos** y en la flecha que aparece en el lado superior derecho, de clic para visualizar las **Tareas de DataGridView**:

Figura 12.8. Ventana Tareas de DataGridView.



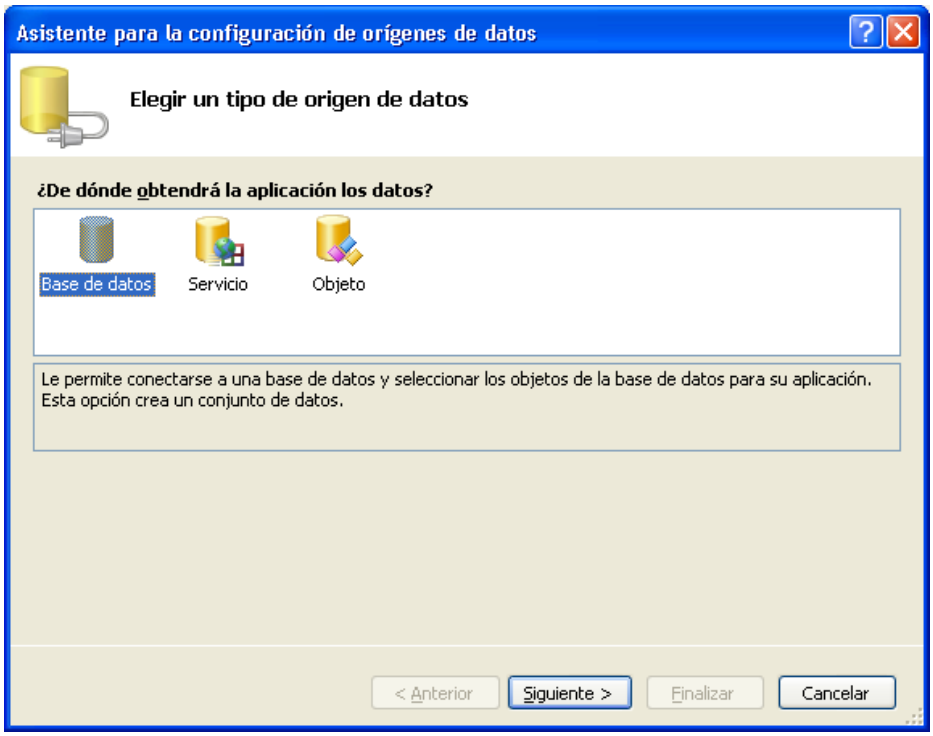
Escoja la opción **Elegir orígen de datos,** para visualizar la siguiente figura:

Figura 12.9. Ventana Elegir orígen de datos.



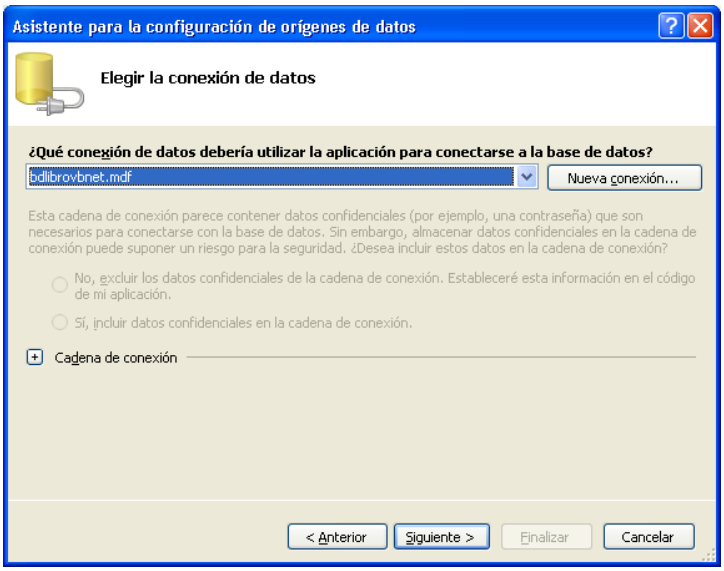
De clic sobre el link **Agregar orígen de datos del proyecto**, se visualizara la siguiente ventana:

Figura 12.10. Asistente para la configuración de orígenes de datos.



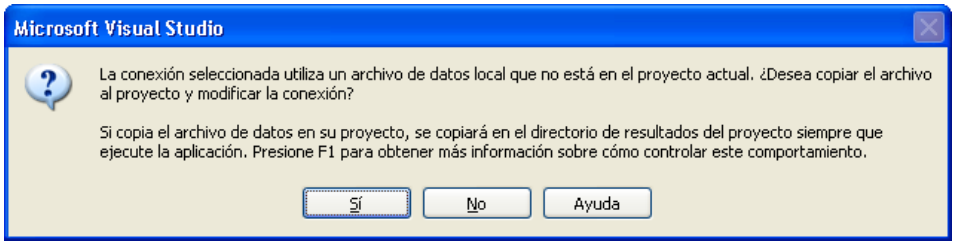
Seleccione el objeto **Base de datos** y pulse el botón **Siguiente>**, para visualizar la ventana de elección de la conexión de datos. Seleccione la base de datos **bdlibrovbnet.mdf**:

Figura 12.11. Ventana Elegir la conexión de datos



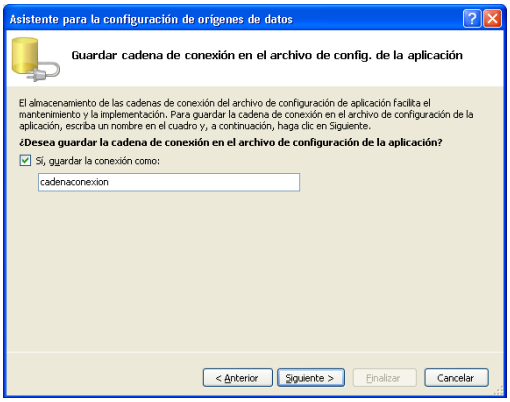
Al Seleccionar la base de datos que se va a trabajar y pulsando el botón **Siguiente>** se visualizar el siguiente mensaje:

Figura 12.12. Ventana para copiar el archivo de datos al proyecto.



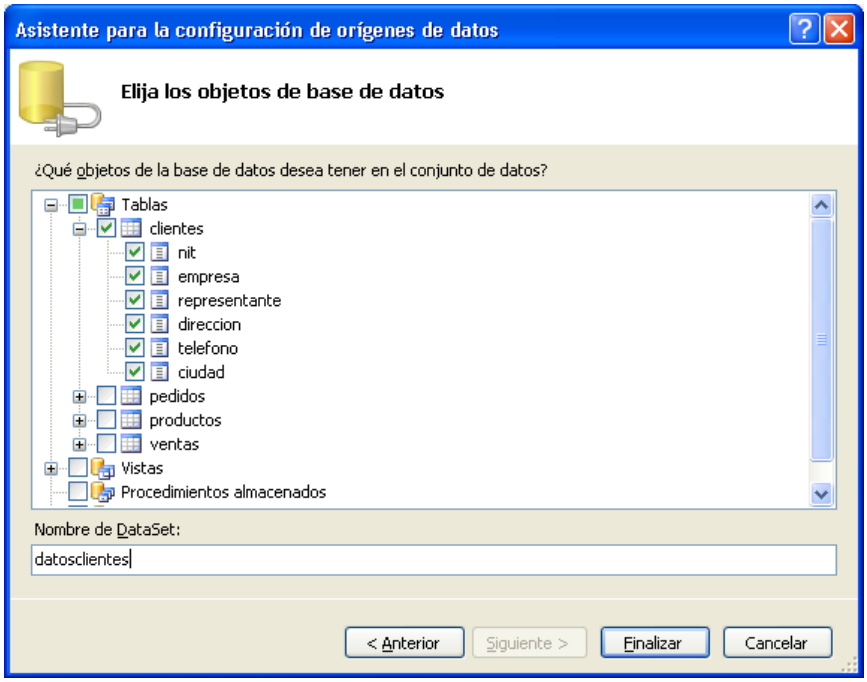
Pulse el botón **Sí** para copiar el archivo de datos en la carpeta donde guardo el proyecto y se visualizara la ventana de guardar cadena de conexión:

Figura 12.13. Ventana Guardar cadena de conexión.



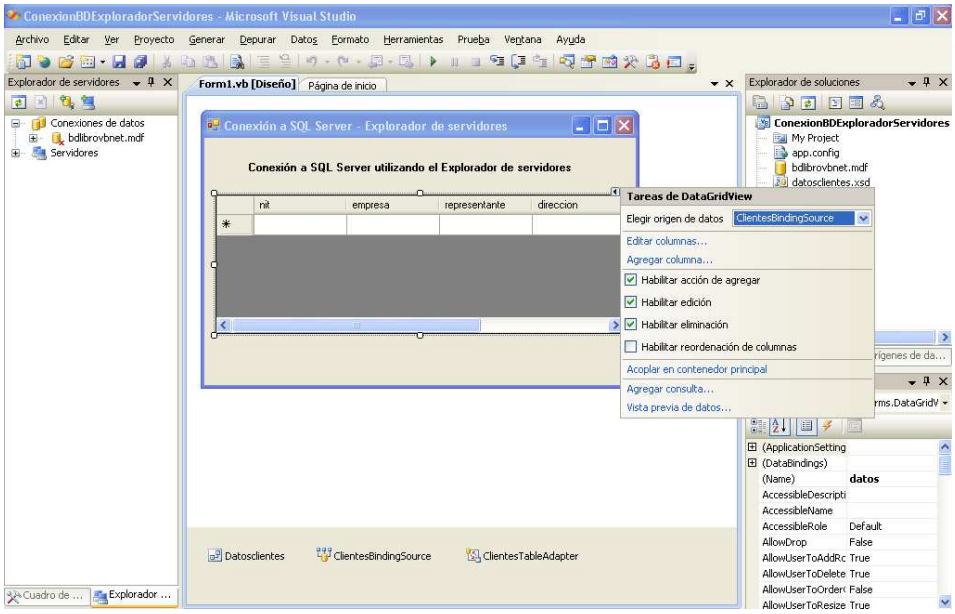
Cambie el nombre de la conexión que allí aparece por **cadenaconexion** y pulse el botón **Siguiente>,** se visualizara la siguiente figura:

Figura 12.14. Ventana de elección de objetos de la base de datos.



Pulse el signo (+) al lado de **Tablas** para desplegar las tablas de la base de datos y seleccione la tabla **clientes**. Por otro lado cambie el nombre del **DataSet** que allí aparece por **datosclientes** y pulse el botón **Finalizar,** para visualizara la figura:

Figura 12.15. DataGridView con los campos de la tabla clientes.

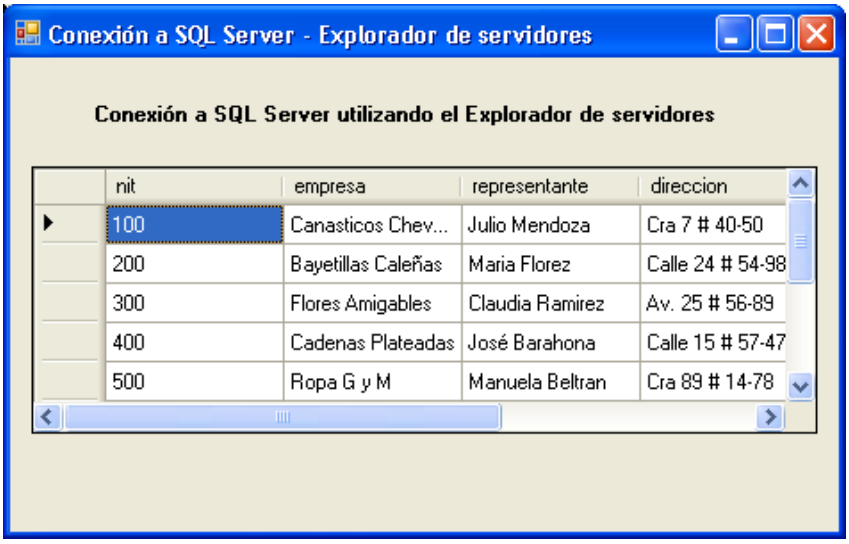


Como se puede apreciar se crearon tres nuevos objetos: el objeto **Datosclientes** que contendrá los datos seleccionados, el objeto **ClientesBindingSource** que en su propiedad **DataSource** se le asignara el objeto **Datosclientes** y en la propiedad **DataMember** el nombre de tabla seleccionada. Por otro lado se creó el objeto **ClientesTableAdapter**, el cual será el adaptador de datos para la conexión realizada.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto, se visualizara en el formulario los datos de la tabla **clientes** en el objeto **DataGridView** de la base de datos **bdlibrovbnet.mdf** de SQL Server:

Figura 12.16. DataGridView con los registros de la tabla clientes.



12.5.3. Conexión a una base de datos con DataGridView y BindingNavigator.

Diseñar un proyecto llamado **ConexionAutomatica y** realizar un programa que permita a un usuario realizar una conexión a una base de datos de SQL Server y mostrar los registros de una tabla llamada **clientes** en un objeto **DataGridView,** además**,** se debe poder desplazar por los registros de dicha tabla utilizando el control **BindingNavigator**.

NOTA: para este ejemplo se utilizara la base de datos SQL Server llamada **bdlibrovbnet** y su tabla **clientes** (**Ver anexo A, SQL Server, página 423**).

- **Crear la interfaz de usuario.**

Modifique las siguientes propiedades del formulario:

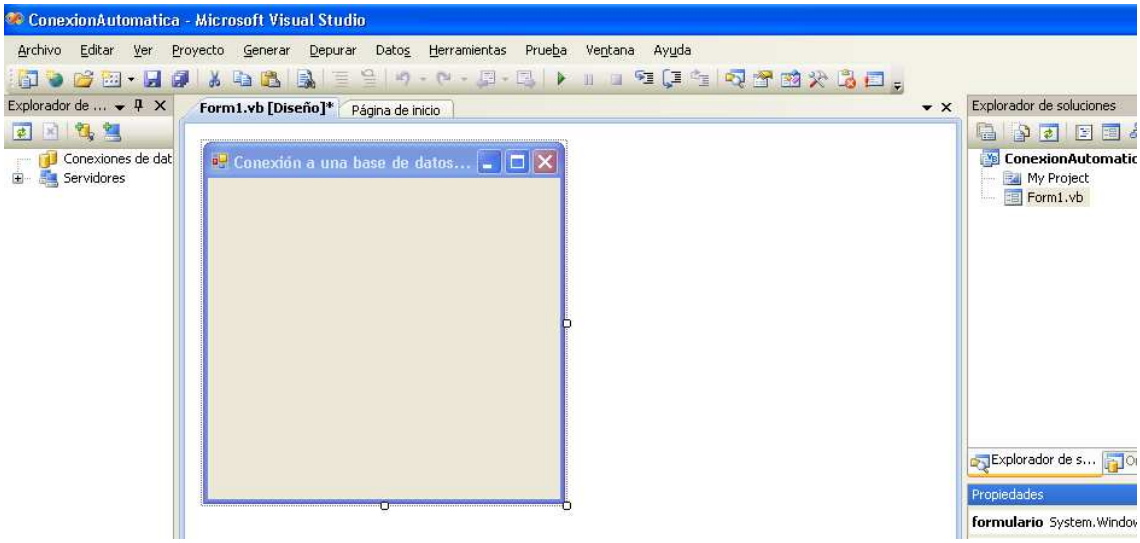
Tabla 12.12. Propiedades modificadas del formulario.

Control	Propiedad	Valor
Form1	Name	formulario
	Text	Conexión a una base de datos SQL Server.

- **Conectarse a la base de datos.**

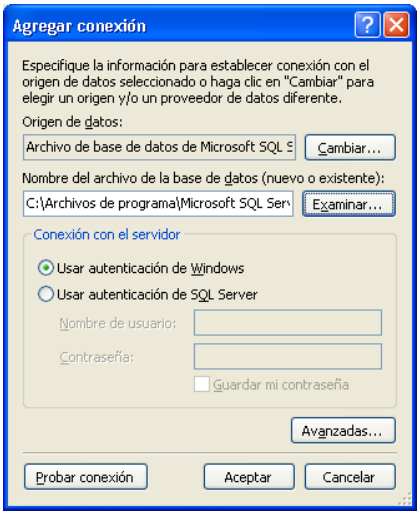
Del menú **Ver** seleccione la opción **Explorador de servidores**, para visualizar la siguiente figura:

Figura 12.17. Ventana Explorador de servidores



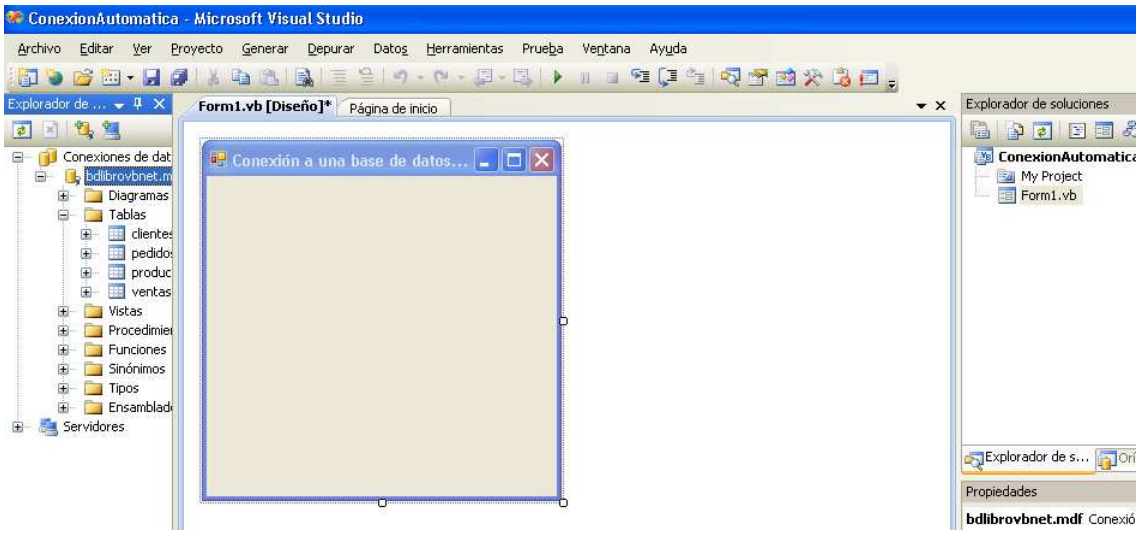
Al pulsar el icono **conectar con bases de datos**  , se visualizara la ventana de **Agregar conexión** como lo muestra la figura:

Figura 12.18. Ventana Agregar conexión.



Seleccione la base de datos **bdlibrovbnet.mdf**, la cual se encuentra en la carpeta **data** de **SQL Server** (si la instalación la realizo en c:\ la ruta sería: C:\Archivos de programa\Microsoft SQL Server\MSSQL.1\MSSQL\Data. Pulse el botón **Aceptar** para visualizar la siguiente figura:

Figura 12.19. Explorador de servidor - conexión a la base de datos SQL Server.

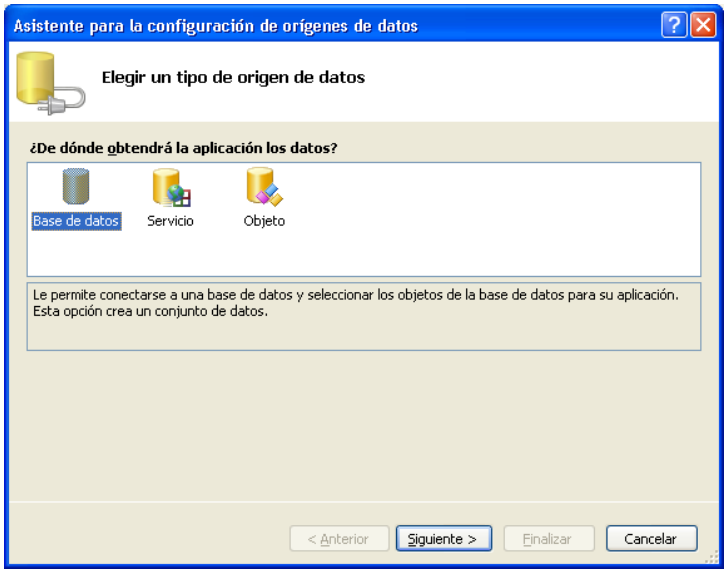


Al pulsar en el signo (+) al lado del nombre **bdlibrovbnet.mdf** se visualizara la estructura definida en la base de datos. Si desea ver las tablas que contiene la base de datos pulse el signo (+) al lado de Tablas.

- **Configurar orígen de datos.**

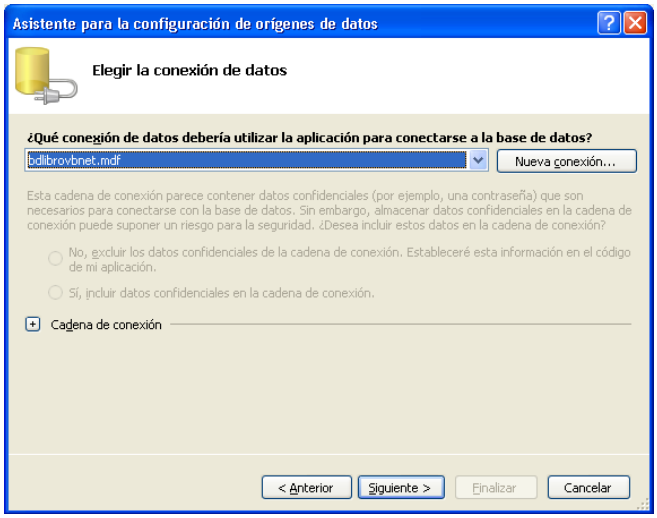
Del menú **Datos** seleccione la opción **Agregar nuevo orígen de datos…**, para visualizar la siguiente figura:

Figura 12.20. Asistente para la configuración de orígenes de datos.



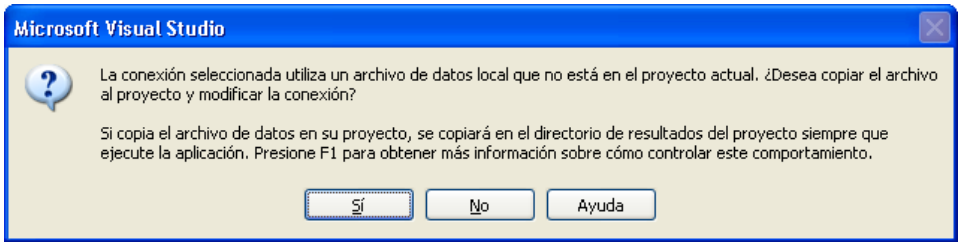
Seleccione el objeto **Base de datos** y pulse el botón **Siguiente>**, para visualizar la ventana **Elegir conexión de datos**. En la ventana se debe seleccionar la base de datos **bdlibrovbnet.mdf**:

Figura 12.21. Ventana Elegir la conexión de datos



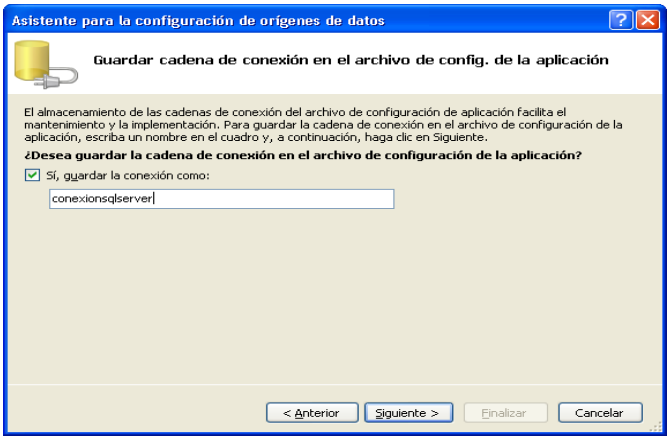
Pulse el botón **Siguiente>** para visualizar el siguiente mensaje:

Figura 12.22. Ventana para copiar la conexión al proyecto.



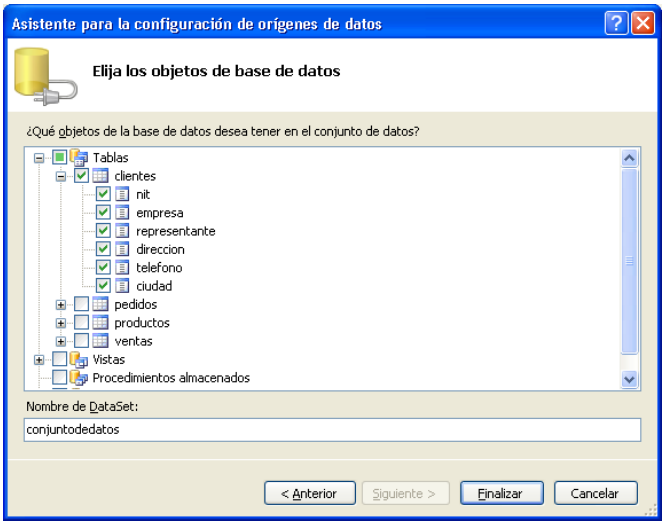
Pulse el botón **Sí** para copiar el archivo de datos en la carpeta donde guardo el proyecto. Se mostrara la ventana Guardar cadena de conexión en el archivo **config** de la aplicación:

Figura 12.23. Ventana Guardar cadena de conexión.



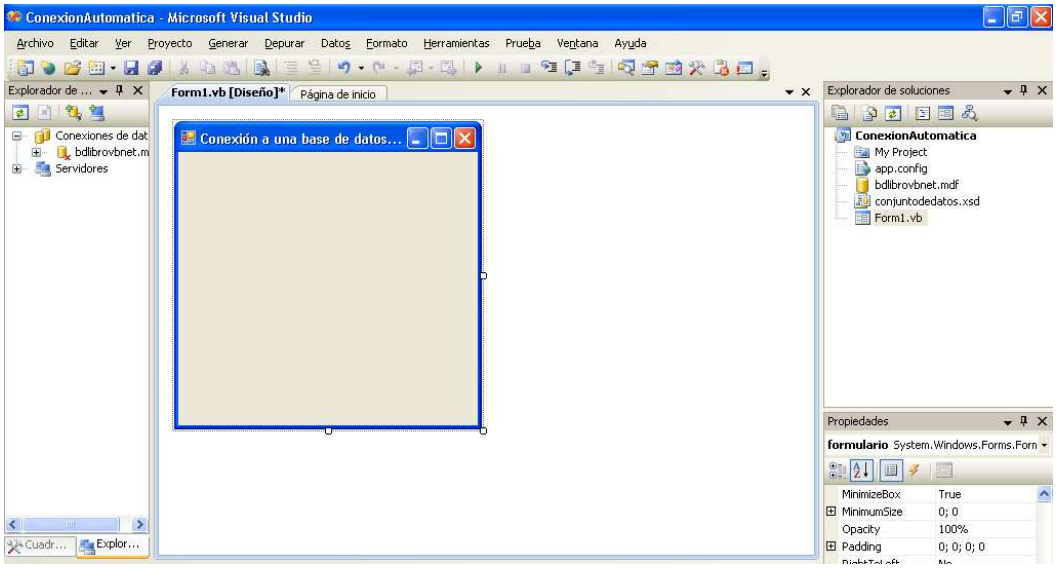
Cambie el nombre de la conexión que allí aparece por **conexionsqlserver** y pulse el botón **Siguiente>,** se visualizara la siguiente figura:

Figura 12.24. Ventana de elección de objetos de la base de datos.



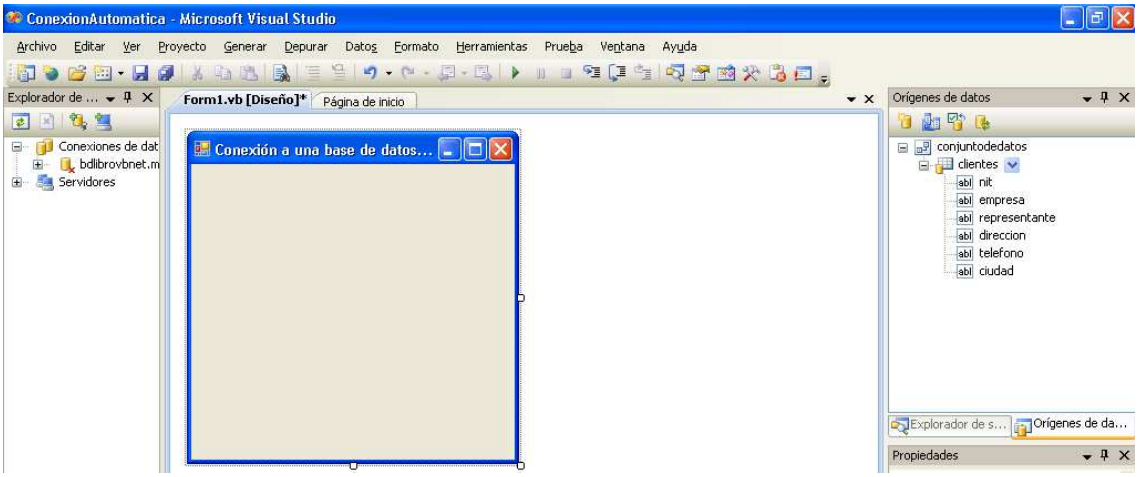
Pulse el signo (+) al lado de **Tablas** para desplegar las tablas de la base de datos y seleccione la tabla **clientes**. Por otro lado cambie el nombre del **DataSet** que allí aparece por **conjuntodedatos** y pulse el botón **Finalizar,** para visualizara la figura:

Figura 12.25. Proyecto con la base de datos y el orígen de datos agregados.



Nuevamente del menú **Datos** seleccione la opción **Mostrar orígenes de datos**, para visualizar la siguiente figura:

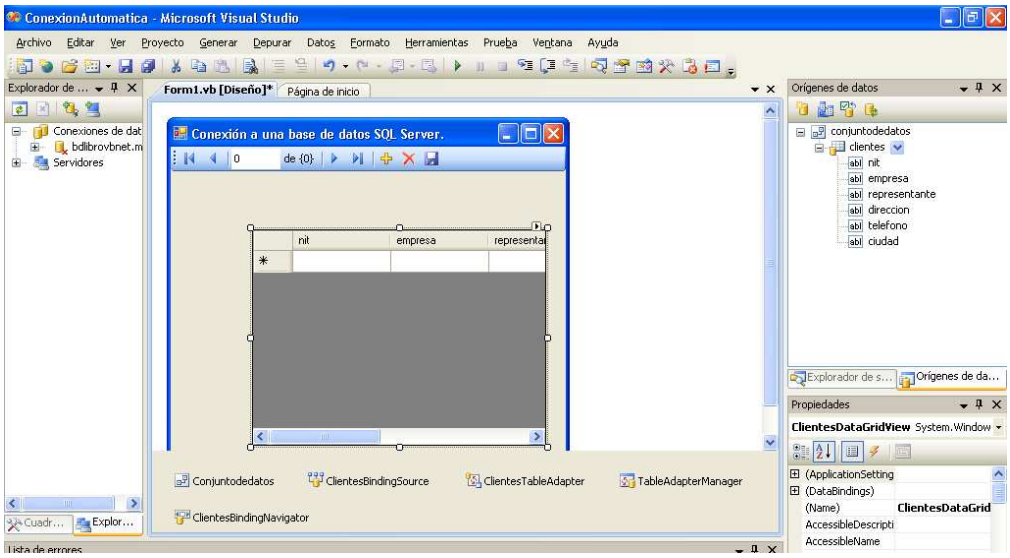
Figura 12.26. Proyecto con la base de datos y el orígen de datos agregados.



En la parte izquierda aparece la ventana **Orígenes de datos**. Allí estará la tabla que se había seleccionado. Pulse el signo (+) al lado de la tabla **clientes** para desplegar los campos.

De clic sobre la tabla **clientes**, pulse el botón izquierdo del mouse y arrastre el mouse hacia el formulario y automáticamente se crearan los siguientes objetos: un control **DataGridView**, un **BindingNavigator** (**ClientesBindingNavigator** – representa una manera normalizada para navegar y manipular los datos en un formulario), un **DataSet** (**conjuntodedatos** – representa un conjunto de datos recuperados de un orígen de datos), un **BindingSource** (**ClientesBindingsource** – encapsula el orígen de datos y proporciona las funciones de navegación, filtrado, ordenación y actualización de los datos), un **TableAdapter** (**ClientesTableAdapter** – crea una nueva tabla con los datos devueltos), un **AdapterManager** (**ClientesAdapterManeger** – administrador del adaptador de datos).

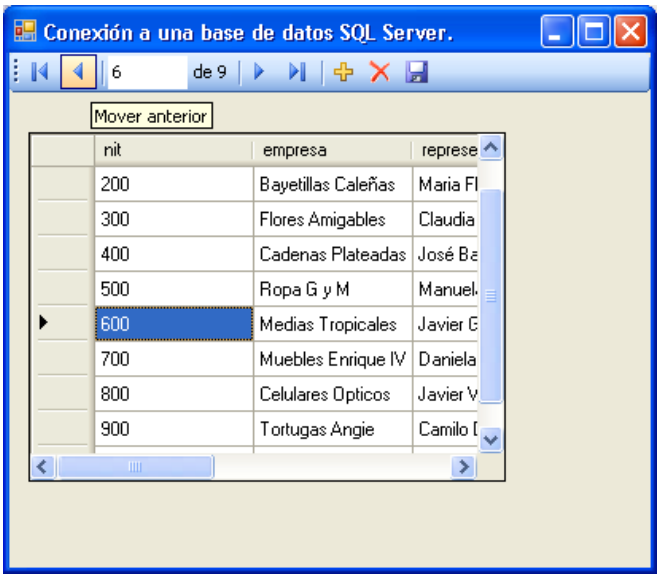
Figura 12.27. Aplicación con los objetos creados automáticamente.



- **Ejecutar el proyecto**

Al ejecutarse el proyecto, se visualizara en el formulario los datos de la tabla **clientes** en el objeto **DataGridView** de la base de datos **bdlibrovbnet.mdf** de SQL Server:

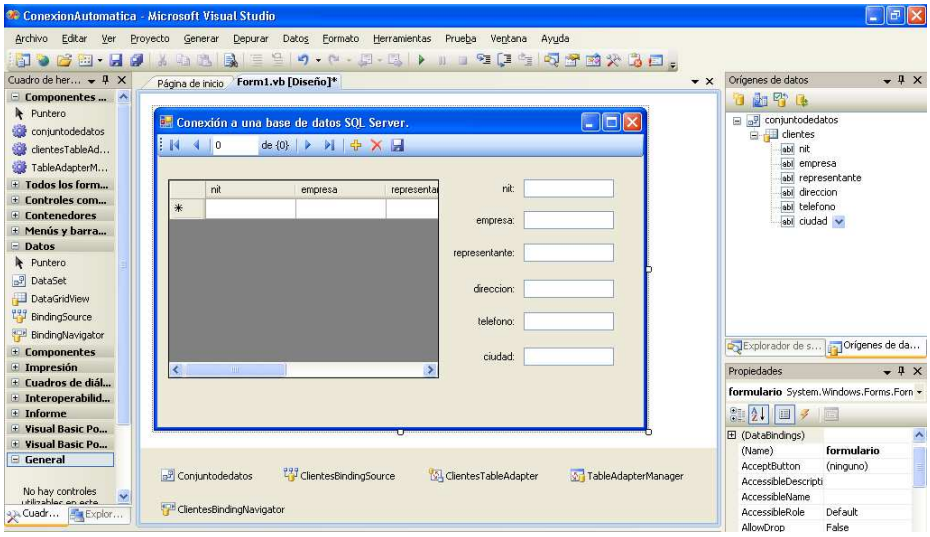
Figura 12.28. Ejecución de la aplicación ConexionAutomatica.



Con los iconos del objeto **BindingSource** se podrá desplazar por cada uno de los registros de la tabla.

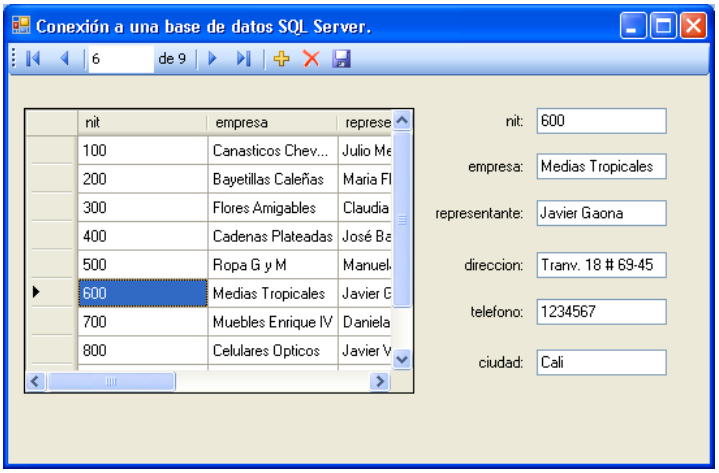
Estando en modo edición se pudo apreciar en la ventana **Orígenes de datos** que al pulsar el signo (+) al lado de la tabla **clientes** se despliegan los campos de esta. También es posible arrastrar cada uno de los campos de la tabla hacia el formulario. Realice esta operación hasta obtener la siguiente figura:

Figura 12.29. Formulario con los campos de la tabla clientes.



Al ejecutarse nuevamente la aplicación y situarse en cualquier registro de la tabla se visualizara cada registro independiente en la parte derecha del formulario, como se aprecia en la siguiente figura:

Figura 12.30. Formulario con un registro seleccionado.



12.5.4. Insertar un nuevo registro a una tabla.

Escribir un proyecto llamado **InsertarNuevoRegistro** y realizar un programa que permita insertar un nuevo registro a la tabla **clientes** de la base de datos **SQL Server bdlibrovbnet.mdf**. El nuevo registro se debe adicionar en un control **DataGridView.**

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **Label**, 3 **Button**, 1 **DataGridView**.

- **Establecer las propiedades de los objetos de la interfaz de usuario.**

Establezca las siguientes modificaciones a los controles:

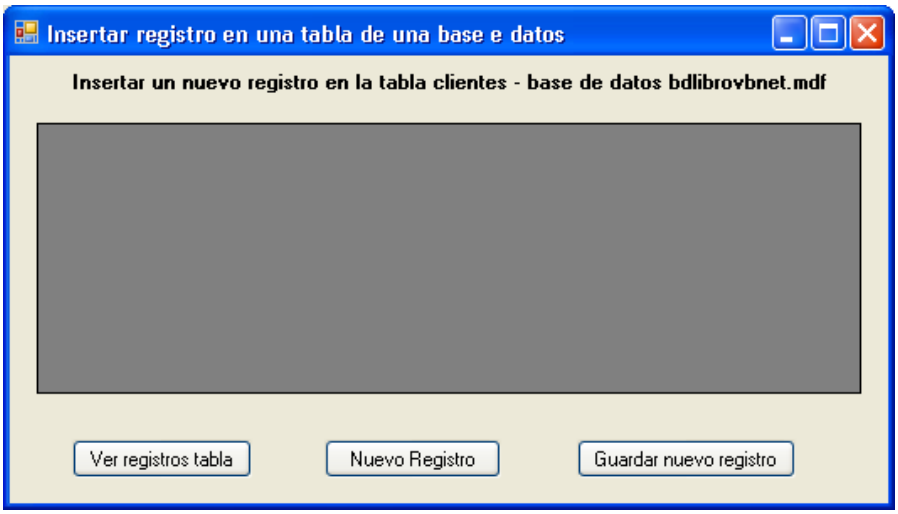
Tabla 12.13. Propiedades de controles proyecto InsertarNuevoRegistro

Control	Propiedad	Valor
Button1	Name	botonver
	Text	Ver registros tabla
Button2	Name	botonnuevo
	Text	Nuevo registro
Button3	Name	botonguardar
	Text	Guardar nuevo registro
DataGridView1	Name	registros
Label1	Name	lbltexto
	Text	Insertar un nuevo registro en la tabla clientes – base

		de datos bdlibrovbnet.mdf
	Font- Bold	True
Form1	Name	formulario
	Text	Insertar registro en una tabla de una base de datos.

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 12.31. Interfaz de usuario (InsertarNuevoRegistro)



- **Escribir código**

a) Antes de la apertura de la clase **formulario** se debe importar el siguiente espacio de nombres:

```
Imports System.Data.SqlClient
Public Class Form1
    ....
    :::::::
End Class
```

El espacio de nombres **System.Data.SqlClient** es necesario para utilizar las diferentes clases que permitan también las operaciones con la base de datos **SQL Server**.

b) Después de la apertura de la clase **formulario** y antes de los procedimientos **Sub**, inicialice las siguientes variables u objetos globales:

```
imports System.Data.SqlClient
Public class formulario
   Private transaccion As New BindingSource
   Private adaptador As SqlDataAdapter
  Dim conexion As String = "Data Source=(local)\SQLEXPRESS;Database=
                                    bdlibrovbnet; Integrated Security=True"
    .......
    .......
End Class
```

Se inicializan los siguientes objetos: **transaccion** de tipo **BindingSource** para encapsular y manipular el orígen de los datos; **adaptador** de tipo **SqlDataAdapter** para intercambiar datos con una base de datos **SQL Server**; **conexion** de tipo **String** al que se le asignara la cadena de conexión para **SQL Server**.

c) De doble clic sobre el formulario para abrir el editor de código y cree un nuevo procedimiento llamado **ver_datos**:

```
Public Sub ver_datos(ByVal sentenciasql As String, ByVal tabla As DataGridView)
  Try
    adaptador = New SqlDataAdapter(sentenciasql, conexion)
    Dim comando As New SqlCommandBuilder(adaptador)
    Dim tabladedatos As New DataTable()
    adaptador.Fill(tabladedatos)
    transaccion.DataSource = tabladedatos
    With tabla
        .Refresh()
        .FirstDisplayedScrollingRowIndex = transaccion.Position
    End With
  Catch ex As SqlException
    MsgBox(ex.Message.ToString)
  Catch ex As Exception
        MsgBox(ex.Message.ToString)
  End Try
End Sub
```

El procedimiento **ver_datos** recibe como parámetros una variable de tipo **String** llamada **sentenciassql** y un objeto **tabla** de tipo **DataGridView**. En un bloque **Try** se le asigna espacio de memoria al objeto **adaptador** de tipo **SqlDataAdapter**, el cual recibe la sentencia SQL y la cadena de conexión. Por otro lado se crea un objeto **tabladedatos** al cual se le asigna espacio de memoria **DataTable** (). Se rellena el adaptador de datos (**adaptador**) con la tabla de datos (**tabladedatos**) y a la propiedad **DataSource** del objeto **transaccion** se le asigna la tabla de datos. Por otro lado se refresca (**Refresh** ()) la cuadricula para cargar los datos y se obtiene la posición del primer registro de la cuadricula (**FirstDisplayedScrollingRowIndex = transaccion.Position**). Utilizando dos bloques **catch** se atrapan las excepciones **SQLException** (por si existe algún error en ejecución de las sentencias SQL) y **Exception** (por si existen algún problema con el sistema).

d) Seleccione el objeto **botonver,** de doble clic para abrir el editor de código y escriba el siguiente código:

```
registros.DataSource = transaccion
cargar_datos("select * from clientes", registros)
```

A la propiedad **DataSource** del objeto **registros** se le establece el orígen de datos (**transaccion)** y se llama al procedimiento **ver_datos** enviándole la sentencia SQL "**Select * from clientes**" y el objeto **registros**.

e) Seleccione el objeto **botonnuevo,** de doble clic para abrir el editor de código y escriba el siguiente código:

```
transaccion.AddNew()
```

Se utiliza el método **AddNew** del objeto **transaccion** para crear un nueva fila en el objeto **registros**.

f) Seleccione el objeto **botonguardar,** de doble clic para abrir el editor de código y escriba el siguiente código:

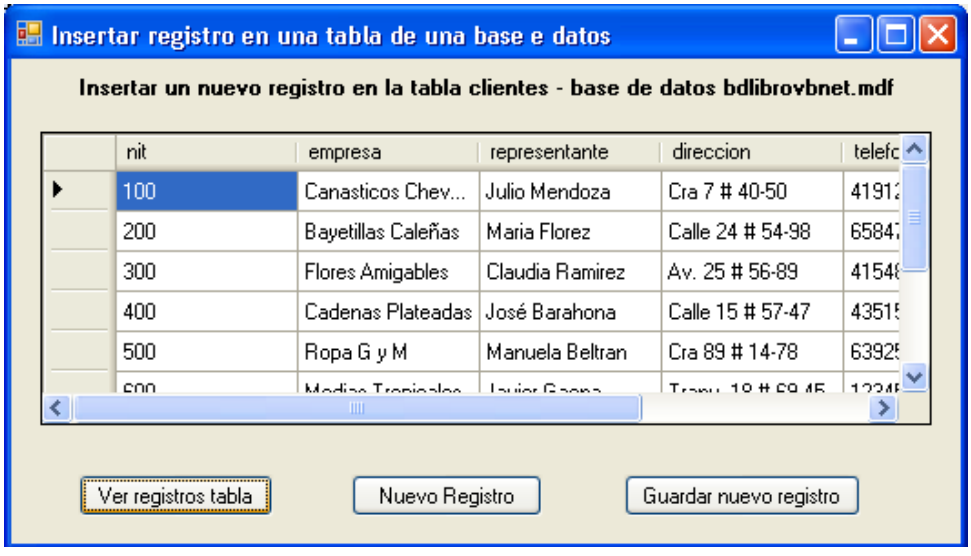
```
If Not transaccion.DataSource Is Nothing Then
    Adaptador.Update(CType(transaccion.DataSource, DataTable))
    cargar_datos("Select * From clientes", tabla)
Else
    MsgBox("No se pudo guardar el registro", MsgBoxStyle.Critical)
End If
```

Si la fila del objeto **registros** no es vacía por medio del método **Update** del objeto **adaptador** se guarda la nueva fila en la tabla **clientes,** enviándole el contenido del **DataSource** del objeto **transaccion** y un objeto **DataTable**. Por otro lado se llama al procedimiento **ver_datos** para mostrar nuevamente los registros de la tabla **clientes**. Si la fila es vacía se muestra el mensaje "No se pudo guardar el registro".

- **Ejecutar el proyecto**

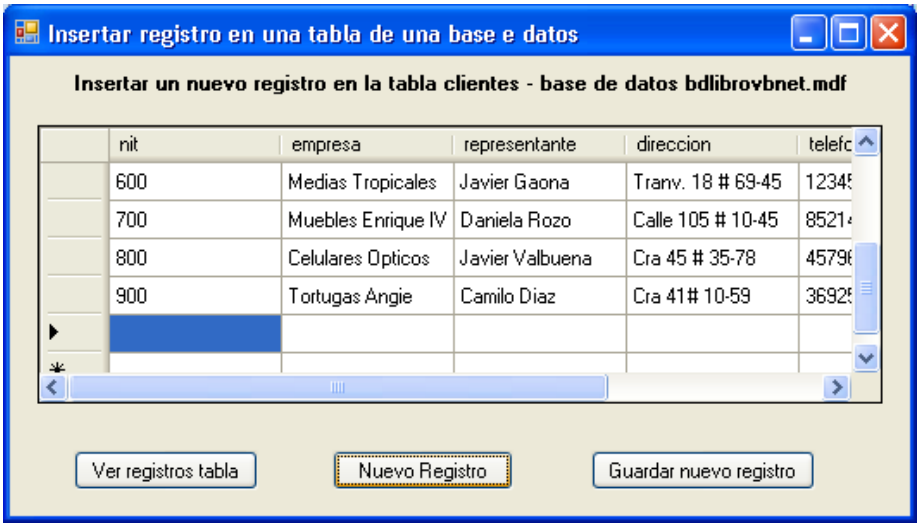
Al ejecutarse el proyecto y pulsar el botón **Ver registros tabla**, se visualizaran los registros de la tabla **clientes**, como se aprecia en la siguiente figura:

Figura 12.32. Formulario con los registros de la tabla clientes



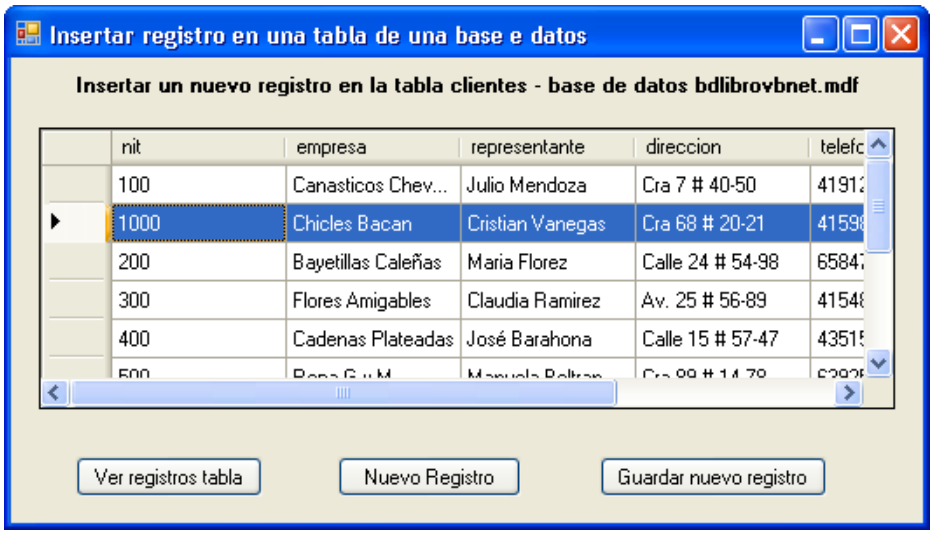
Al pulsar el botón **Nuevo Registro**, se seleccionara en una nueva fila el primer campo del objeto **registros**, el formulario queda de la siguiente forma:

Figura 12.33. Cuadricula con una nueva fila para insertar datos.



Si se captura la información en cada campo de la cuadricula (1000, Chicles Bacan, Cristian Vanegas, Cra 68 # 20-21, 4159854, Bogotá) y luego se pulsa el botón **Guardar nuevo registro**, se insertaran los datos en la tabla **clientes**. En la siguiente figura se aprecia el registro insertado:

Figura 12.34. Cuadricula con un nuevo registro insertado en la tabla clientes



12.5.5. Filtrar registros de una tabla.

Diseñar un proyecto llamado **FiltrarPorCampos,** y permitir realizar filtrar los registros de una tabla por cualquier campo de dicha tabla y visualizar todos los registros que cumplan la condición en un control **DataGridView**.

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 6 **Label,** 1 **TextBox,** 1 **Button**, 1 **DataGridView** y 2 **ComboBox**.

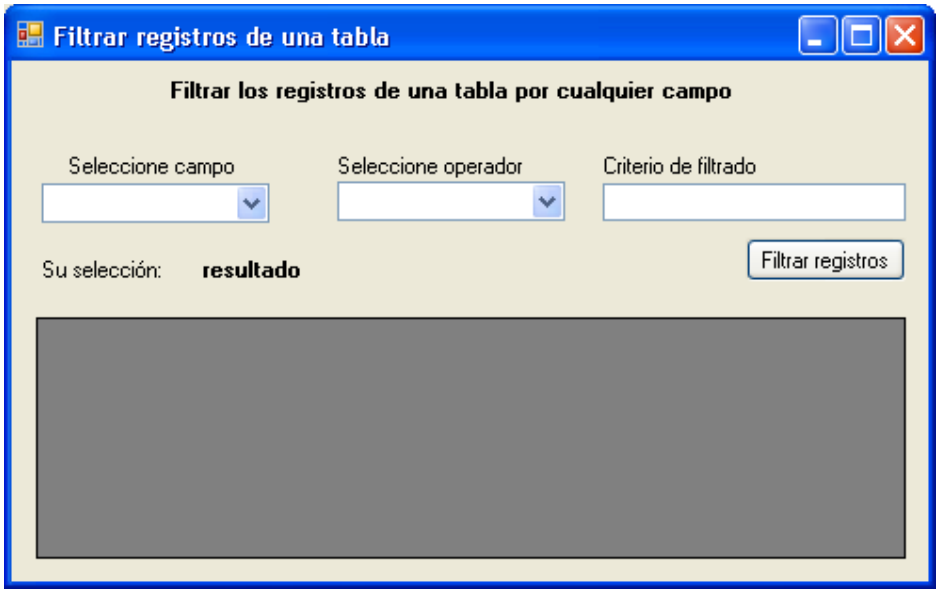
- **Establecer las propiedades de los objetos de la interfaz de usuario.**

Establezca las siguientes modificaciones a los controles:

Tabla 12.14. Propiedades de controles proyecto FiltrarPorCampos.

Control	Propiedad	Valor
Label1	Text	Filtrar los registros de una tabla por cualquier campo.
	Name	lblfiltrar
	Font /Bold	True
Label2	Text	Seleccione campo
	Name	lblcampo
Label3	Text	Seleccione operador
	Name	lbloperador
Label4	Text	Criterio de filtrado
	Name	lblcriterio
Label5	Text	Su selección:
	Name	lblseleccion
Label6	Text	resultado
	Name	lblresultado
	Font/ Bold	True
TextBox1	Name	valorcampo
Button1	Name	botonfiltrar
	Text	Filtrar registros
ComboBox1	Name	listacampos
ComboBox2	Name	listaoperadores
DataGridView1	Name	registrosfiltrados
Form1	Name	formulario
	Text	Filtrar registros de una tabla.

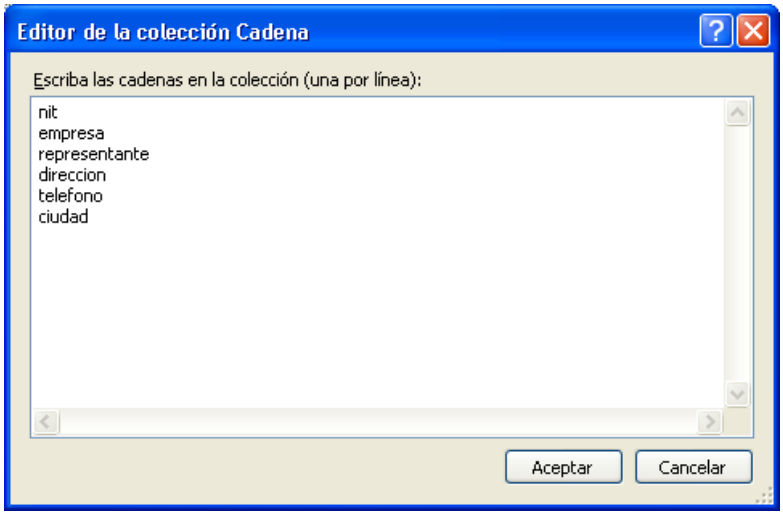
Figura 12.35. Interfaz de usuario (FiltrarPorCampos)



- **Escribir código**

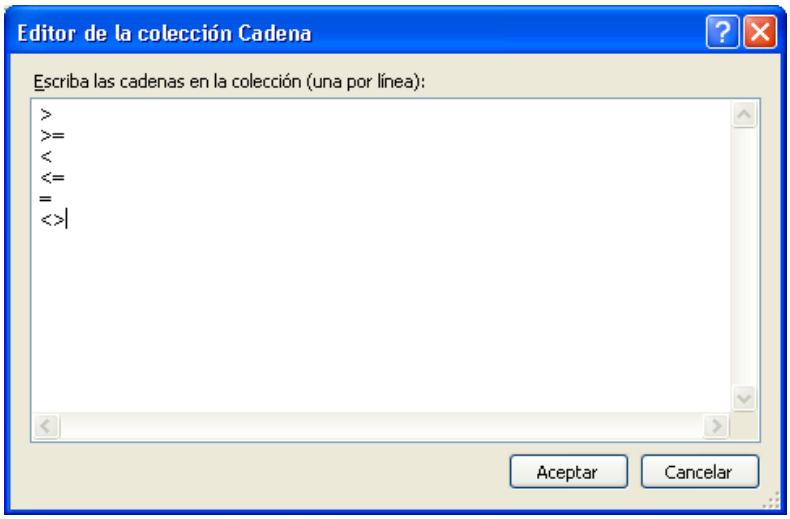
a) Seleccione el control **listacampos**, busque la propiedad **Items**, de clic en los tres puntos al lado de **Colección** y en ventana **Editor de la colección Cadena**, escriba por cada línea un campo de la tabla **clientes** y pulse **Aceptar**. Otra forma es dar clic sobre la flecha del control en la parte superior derecha y seleccionar la opción **Editar elementos**. La ventana de edición deberá quedar como se muestra en la figura:

Figura 12.36. Ventana Editor del objeto listacampos



b) Seleccione el control **listaoperadores**, busque la propiedad **Items**, de clic en los tres puntos al lado de **Colección** y en ventana **Editor de la colección Cadena**, escriba por cada línea los operadore**s: <, >=, <, >=, =, <>,** luego pulse el botón **Aceptar**. La ventana de edición deberá quedar como se muestra en la figura:

Figura 12.37. Ventana Editor del objeto listaoperadores



c) Antes de la apertura de la clase **formulario** se debe importar el siguiente espacio de nombres:

```
Imports System.Data.SqlClient
 Public Class formulario
      ……
End Class
```

d) Después de la apertura de la clase **formulario** y antes de los procedimientos **Sub**, inicialice la siguiente variable global:

```
Public class formulario
    Dim texto as String
      …….
End Class
```

La variable **texto** servira para guardar el contenido del objeto **lblresultado** en su propiedad **Text**.

e) Seleccione el objeto **listacampos,** de doble clic para abrir el editor de código y escriba el siguiente código:

```
lblresultado.Text = listacampos.SelectedItem.ToString
```

Al objeto **lblresultado** en su propiedad **Text** se le asigna el nombre del campo seleccionado por medio de la propiedad **SelectedItem** del objeto **listacampos**;

f) Seleccione el objeto **listaoperadores,** de doble clic para abrir el editor de código y escriba el siguiente código:

```
lblresultado.Text = lblresultado.Text  & listaoperadores.SelectedItem.ToString
texto=lblresultado.text
```

Al objeto **lblresultado** en su propiedad **Text** se le asigna su contenido actual concatenado con el operador seleccionado por medio de la propiedad **SelectedItem** del objeto **listaoperadores**. A la variable **texto** se le asigna el contenido del objeto **lblresultado**.

g) Seleccione el objeto **txtcriterio,** de doble clic para abrir el editor de código y escriba el siguiente código:

```
lblresultado.Text = texto  & txtcriterio.text
```

Al objeto **lblresultado** en su propiedad **Text** se le asigna el contenido de la variable **texto** concatenado con el valor escrito en el objeto **txtcriterio** en su propiedad **text.**.

g) Seleccione el objeto **botonfiltrar,** de doble clic para abrir el editor de código y escriba el siguiente código:

```
Dim conexion As String
conexion = "Data Source=(local)\SQLEXPRESS;Database=bdlibrovbnet;
                              Integrated Security=True"
Dim clave = valorcampo.Text
Dim nombre_campo As String = listacampos.SelectedItem.ToString
Dim valoroperador As String = listaoperadores.SelectedItem.ToString
Dim seleccion As String = "SELECT * FROM clientes where " + nombre_campo + " " +
                                    valoroperador + "'" + clave + "'"
Dim adaptador As SqlDataAdapter
Dim tabladedatos As New DataTable
Try
   adaptador = New SqlDataAdapter(seleccion, conexion)
   adaptador.Fill(tabladedatos)
   tabla.DataSource = tabladedatos
Catch ex As Exception
   MsgBox("Error: " & ex.Message)
End Try
```

Se crean las variables: **clave** que almacenara el valor que este escrito en el objeto **txtcriterio; nombre_campo** de tipo **String** que almacenara el nombre del campo seleccionado por medio de la propiedad **SelectedItem** del objeto **listacampos**; **valoroperador** de tipo **String** que almacenara el operador seleccionado por medio de la propiedad **SelectedItem** del objeto **listaoperadores**; también se crea una variable **seleccion** de tipo **String** que almacenara la instrucción SQL **Select.** En dicha instrucción se selecciona todos los campos de la tabla **clientes** (**Select * from clientes**), y se utiliza la cláusula **where** para mostrar solo aquellos registros cuya **nombre_campo** corresponda al criterio de **valoroperador** concatenado con **clave**. En un bloque **Try** se le asigna espacio de memoria de tipo **SqlDataAdapter** al objeto **adaptador**, al cual se le envía como parámetros los objetos **seleccion** (sentencia SQL) y **conexion** (cadena de conexión), luego

se rellena (**fill**) el adaptador de datos con la tabla de datos (**tabladedatos**) y se le asigna al objeto **tabla** por intermedio de la propiedad **DataSource** el objeto **tabladedatos**.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo, se visualizara la figura 12.35. Si se selecciona del objeto **listacampos** el campo "**nit**", del objeto **listaoperadores** el operador **">",** se escribe en el objeto **txtcriterio** "300" y se pulsa el botón **Filtrar registros**, se visualizara en la cuadricula todos aquellos registros cuyo **nit** sea mayor que **300**, como se muestra en la siguiente figura:

Figura 12.38. Formulario con los registros cuyo nit es mayor de 300

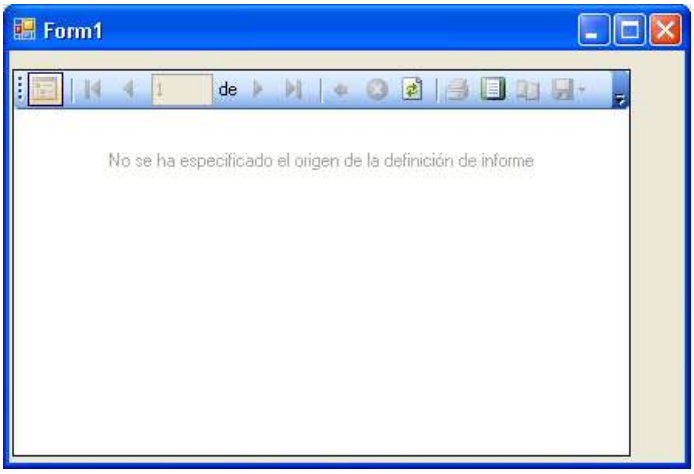
12.5.6. Informe de los registros de una tabla.

Crear un proyecto llamado **InformeDatosTabla,** que permita a un usuario visualizar en un formulario un reporte de todos los registros de la tabla **clientes** de la base de datos **bdlibrovbnet.mdf**.

- **Crear la interfaz de usuario.**

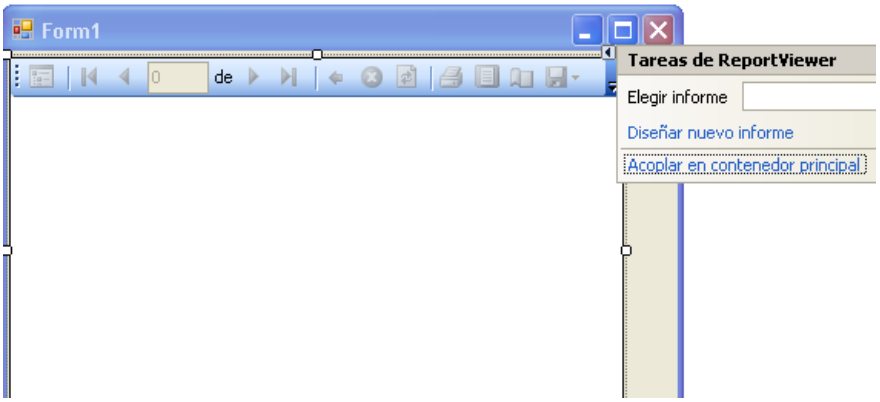
Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **MicrosoftReportViewer**.

Figura 12.43. Interfaz de usuario (InformeDatosTabla)



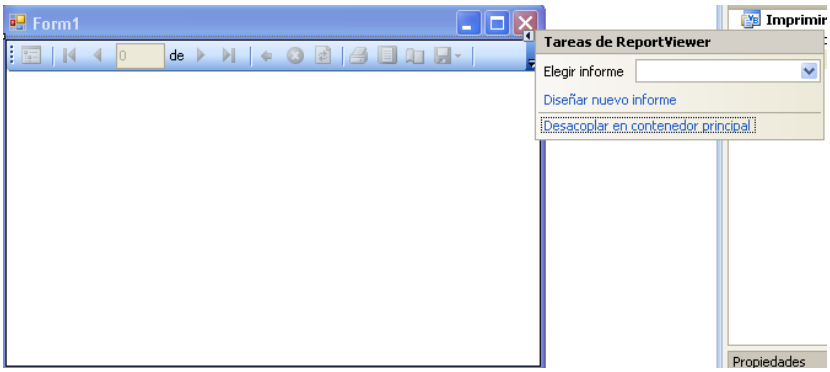
Para acoplar el control **MicrosoftReportViewer1** al formulario, pulse la flecha del lado superior derecha del objeto para ver la ventana **Tareas de ReportViewer.**

Figura 12.44. Interfaz de usuario (InformeDatosTabla)



Seleccione la opción "**Acoplar en contenedor principal**", se visualizara el formulario con el control acoplado como se aprecia en la siguiente figura:

Figura 12.45. Control MicrosoftReportViewer acoplado al formulario



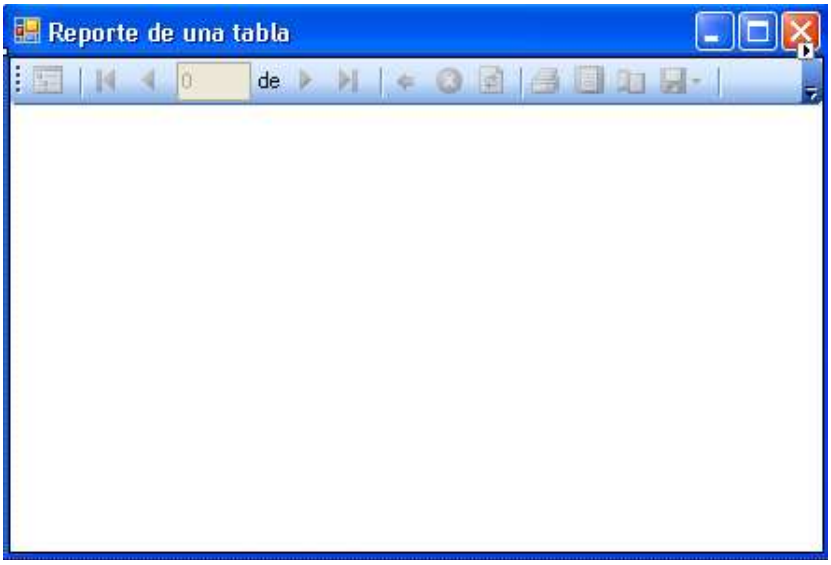
- **Establecer las propiedades de los objetos de la interfaz de usuario.**

Establezca las siguientes modificaciones a los controles:

Tabla 12.15. Propiedades de controles proyecto InformeDatosTabla.

Control	Propiedad	Valor
MicrosoftReportViewer1	Name	reporte
Form1	Name	formulario
	Text	Reporte de una tabla.

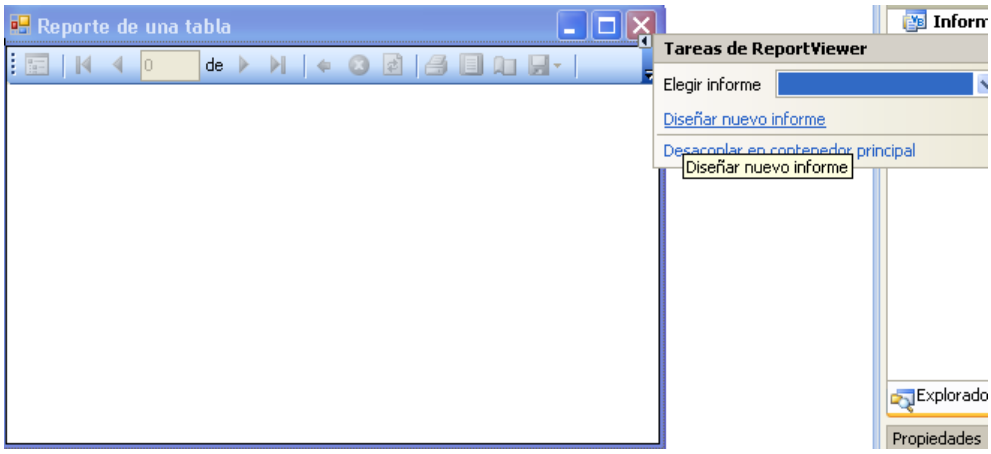
Figura 12.46. Interfaz de usuario modificada (InformeDatosTabla).



- **Diseñar el informe**

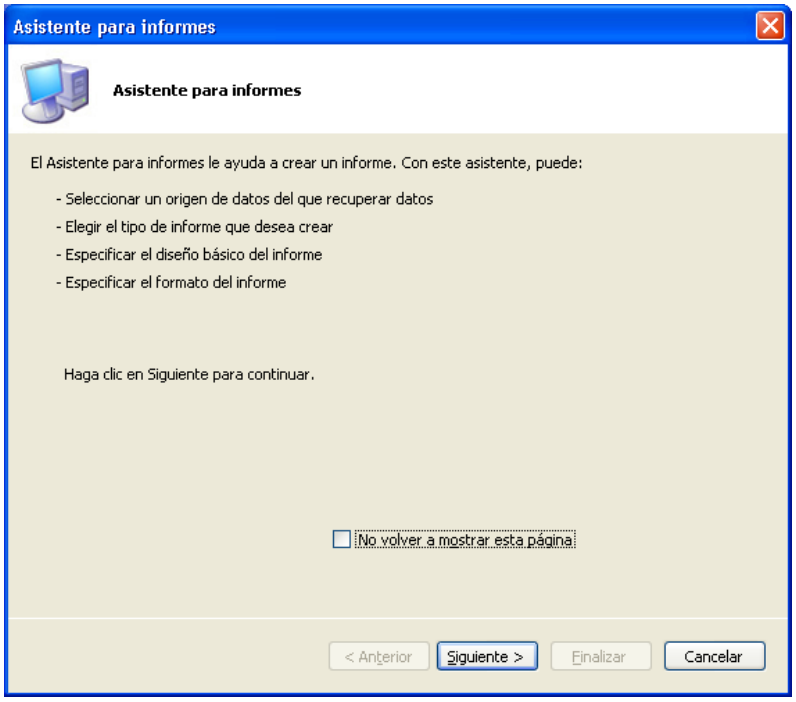
Para diseñar el informe, pulse la flecha del lado superior derecha del objeto **reporte** para ver la ventana **Tareas de ReportViewer** y escoja la opción **Diseñar nuevo informe**.

Figura 12.47. Seleccionar Diseñar nuevo informe



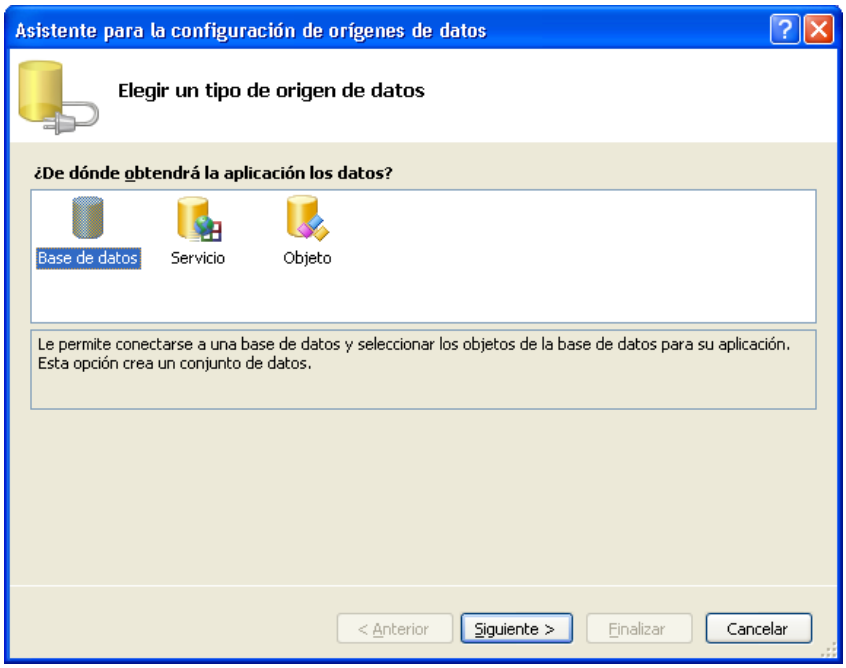
Al seleccionar **Diseñar nuevo informe**, se visualizara la siguiente figura:

Figura 12.48. Asistente para informes.



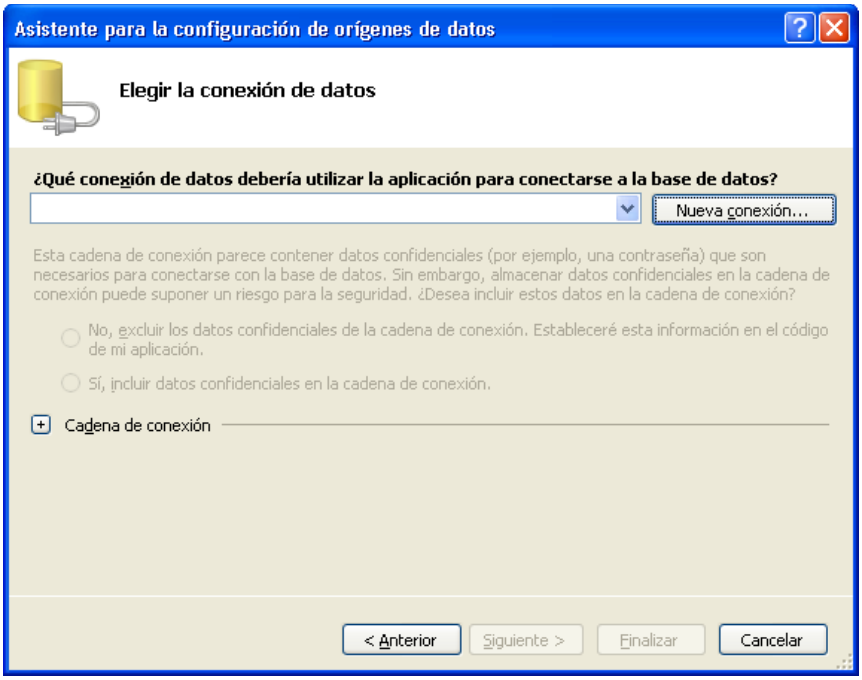
De clic en **Siguiente>** para ver la ventana del **Asistente para la configuración de orígenes de datos**.

Figura 12.49. Asistente para la configuración de orígenes de datos.



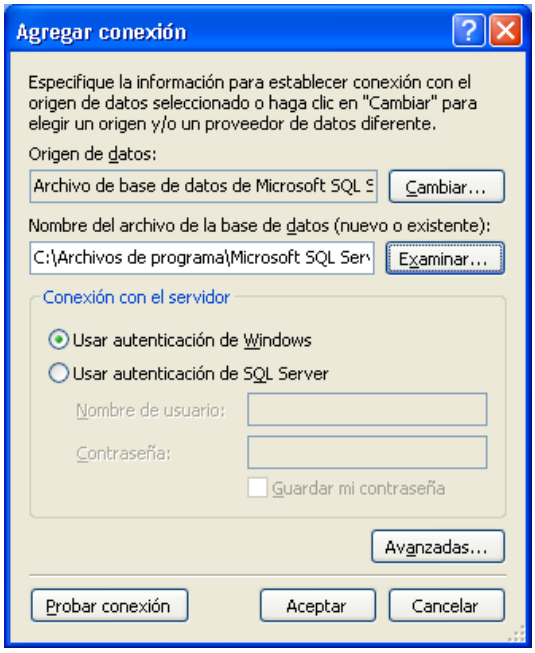
Seleccione el objeto **Base de datos** y pulse el botón **Siguiente>**, para visualizar la ventana de elección de la base de datos:

Figura 12.50. Ventana Elegir la conexión de datos



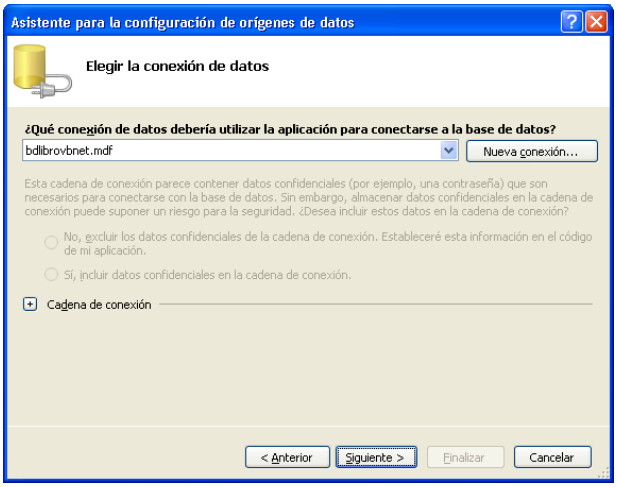
De clic en el botón **Nueva Conexión**. En la ventana que se visualiza elija como orígen de datos **Microsoft SQL Server** y la base de datos **bdlibrovbnet.**

Figura 12.51. Ventana Agregar conexión.



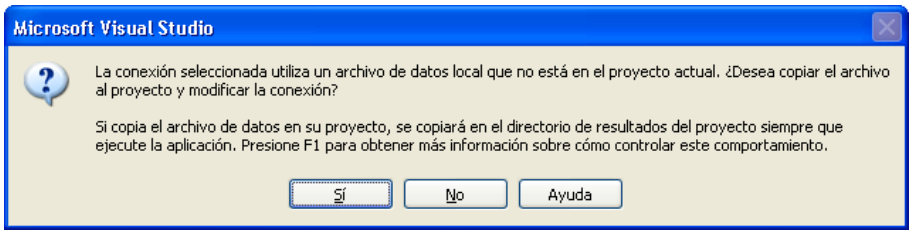
Al pulsar el botón **Aceptar** se visualizara nuevamente la ventana de elección de conexión de datos con la base de datos **bdlibrovbnett.mdf**, como se aprecia en la siguiente figura:

Figura 12.52. Ventana con la base de datos seleccionada



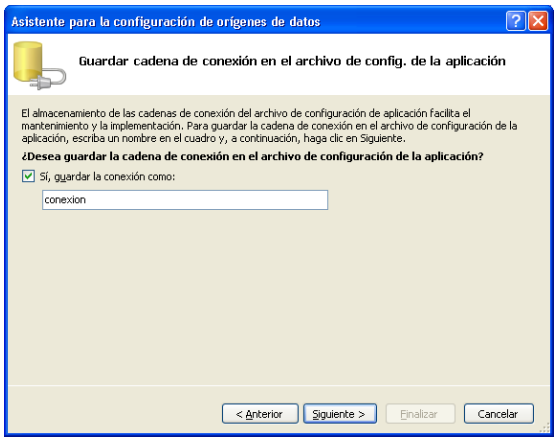
Pulse **Siguiente**> para visualizar el siguiente mensaje:

Figura 12.53. Ventana Microsoft Visual Studio



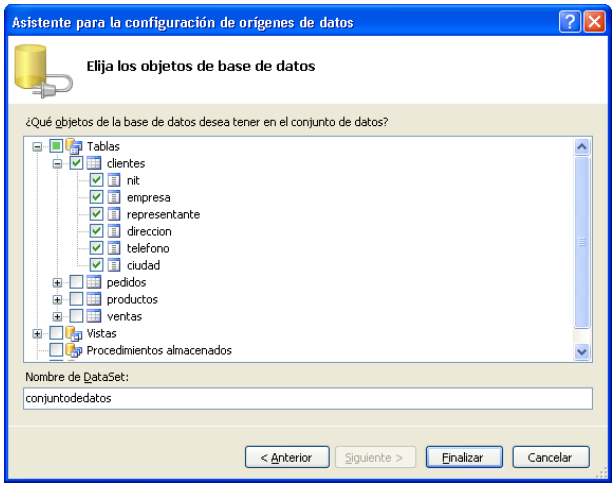
Pulse el botón **Sí** para copiar el archivo de datos en la carpeta donde guardo el proyecto y visualizara la ventana de guardar cadena de conexión:

Figura 12.54. Ventana Guardar cadena de conexión.



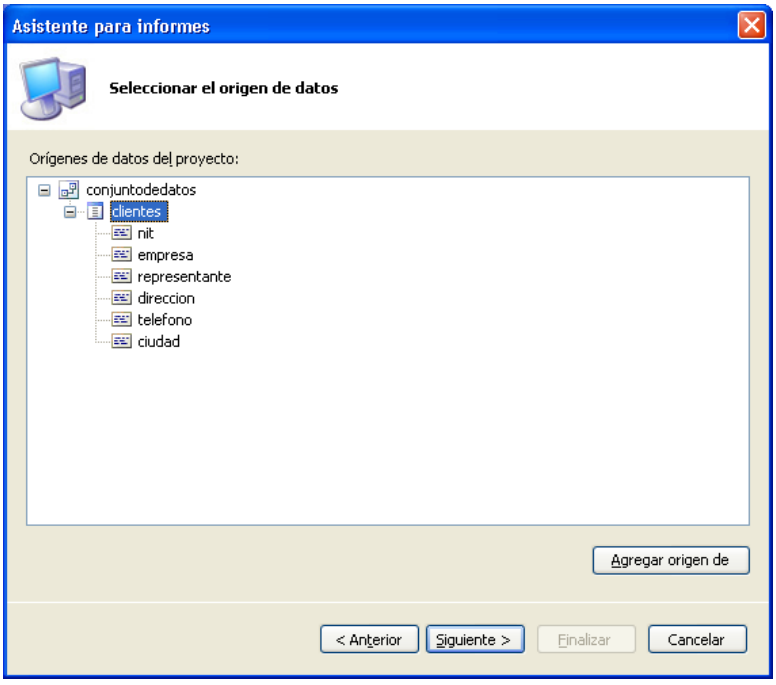
Cambie el nombre de la conexión que allí aparece por **conexion** y pulse el botón **Siguiente>,** se visualizara la siguiente figura:

Figura 12.55. Ventana de elección de objetos de la base de datos.



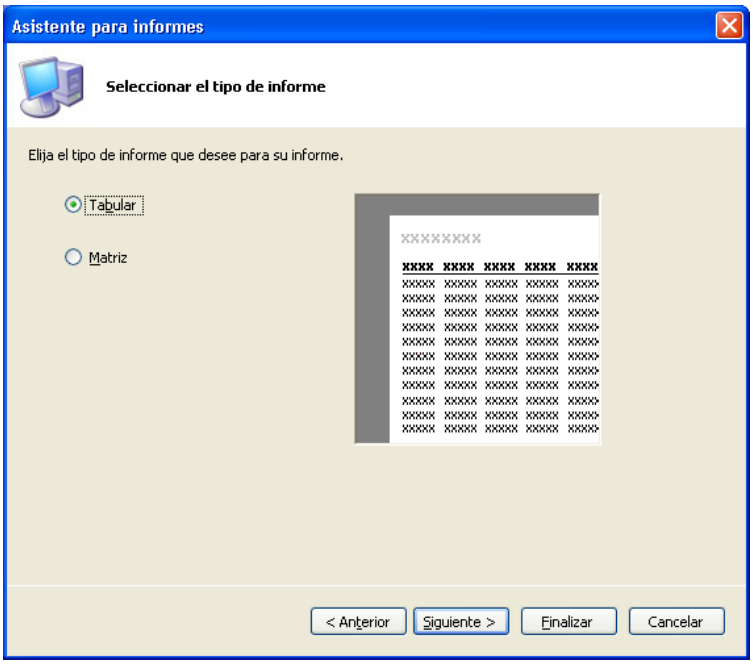
Pulse el signo (+) al lado de **Tablas** para desplegar las tablas de la base de datos y seleccione la tabla **clientes**. Por otro lado cambie el nombre del **DataSet** que allí aparece por **conjuntodedatos** y pulse el botón **Finalizar,** para visualizara la figura:

Figura 12.56. Asistente para informes – seleccionar el orígen de datos.



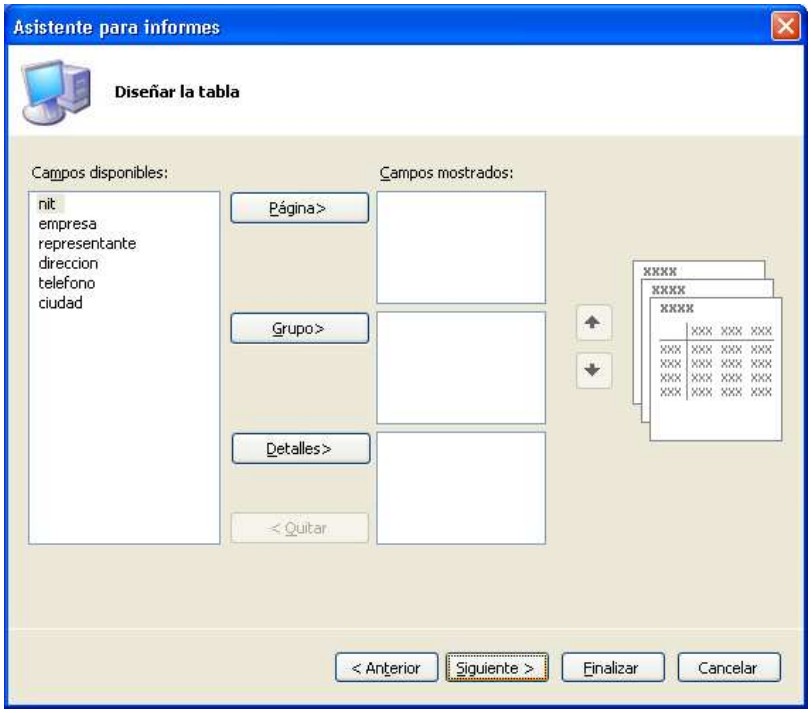
Pulse los signos (+) para desplegar la tabla. De clic en **Siguiente>** para visualizar la ventana de selección de tipo de informe.

Figura 12.57. Asistente para informes – seleccionar el tipo de informe.



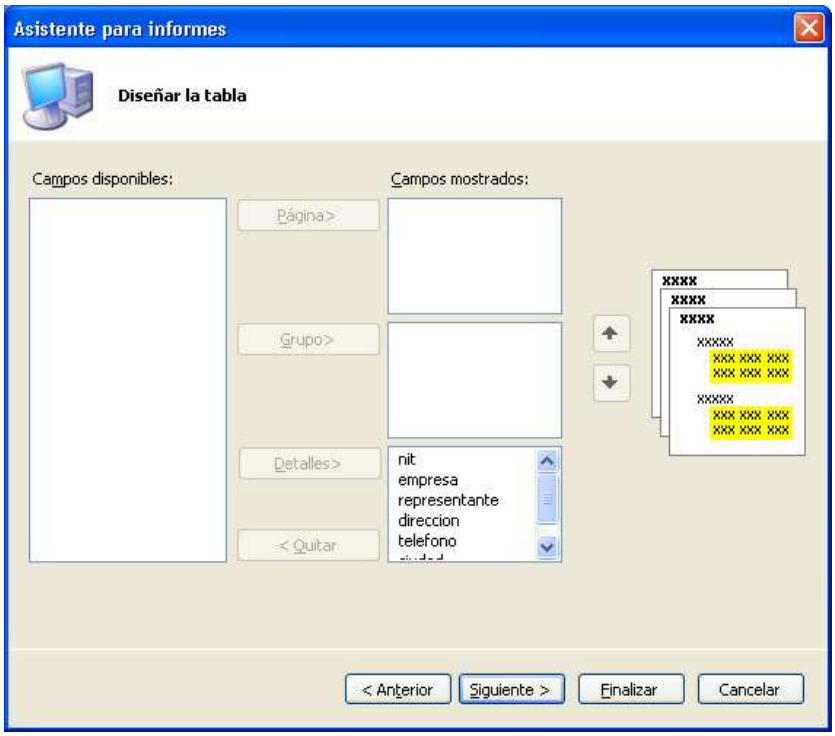
Seleccione la opción **Tabular** o **Matriz** y de clic en **Siguiente>** para visualizar la ventana de diseño de tabla.

Figura 12.58. Asistente para informes – diseñar la tabla.



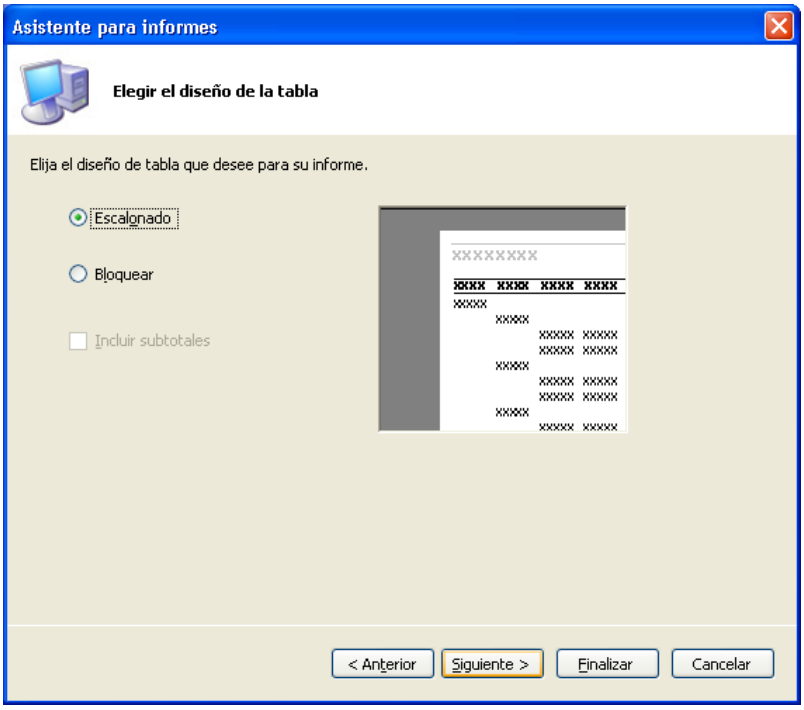
Seleccione cada campo y pulse el botón **Detalles>.** Se obtendra la siguiente figura:

Figura 12.59. Diseño de la tabla con los campos seleccionados.



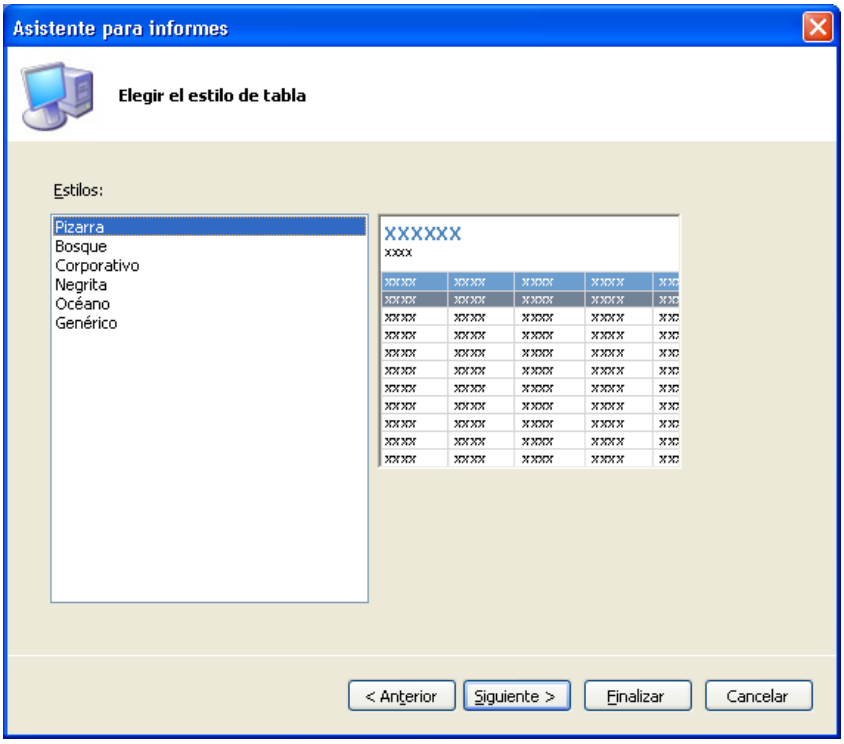
De clic en **Siguiente>** para visualizar la ventana de elección de diseño de la tabla.

Figura 12.60. Asistente para informes – elegir el diseño de la tabla.



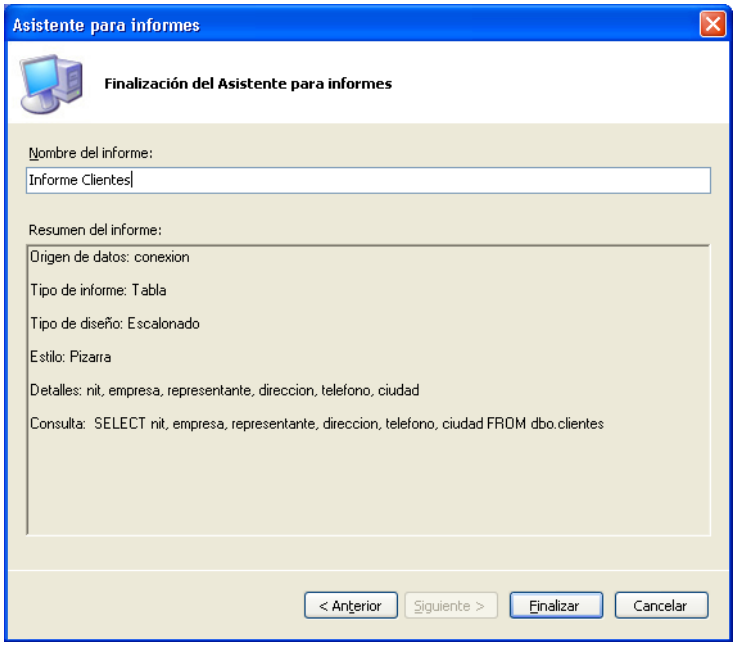
Seleccione la opción **Escalonado** o **Bloquear** y de clic en **Siguiente>** para visualizar la ventana de estilo de la tabla.

Figura 12.61. Asistente para informes – elegir el estilo de tabla.



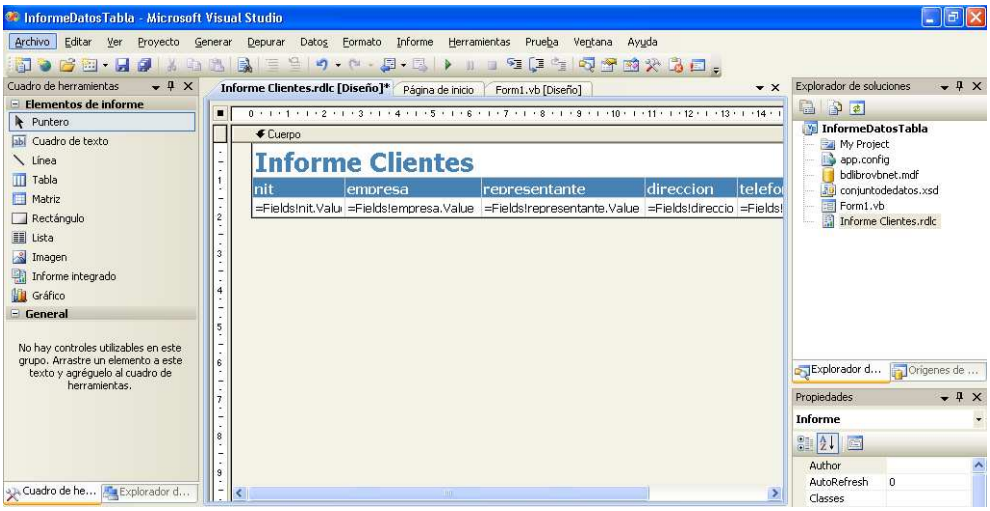
Seleccione la opción con el estilo que desee y de clic en **Siguiente>** para visualizar la ventana de finalización del asistente de informes.

Figura 12.62. Ventana de finalización del asistente de informes.



Donde aparece el nombre del informe cámbielo por **Informe Clientes** y de clic en **Finalizar**> para visualizar el diseño final del informe.

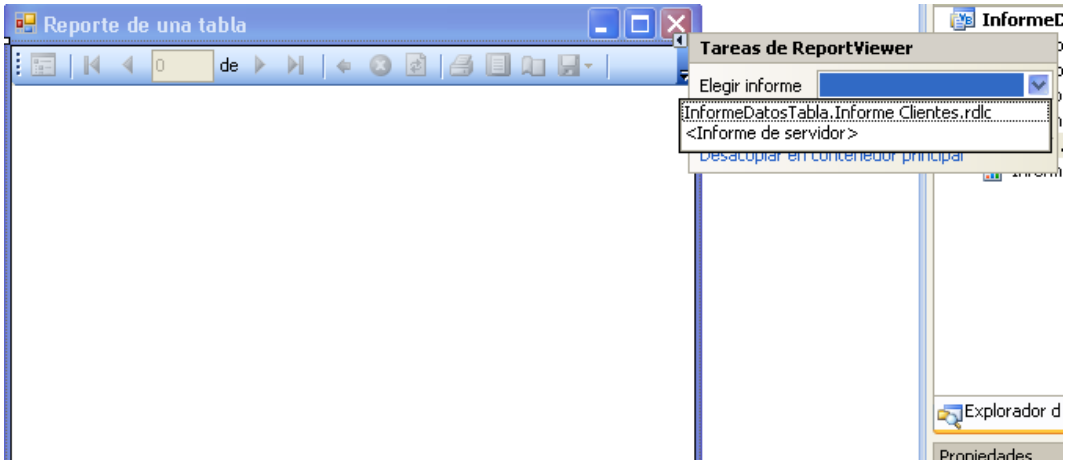
Figura 12.63. Diseño final del informe.



En este momento se podrán modificar los textos de las etiquetas de los campos, como también, la longitud de los campos. Además en el **explorador de soluciones** se visualizara el informe agregado al proyecto.

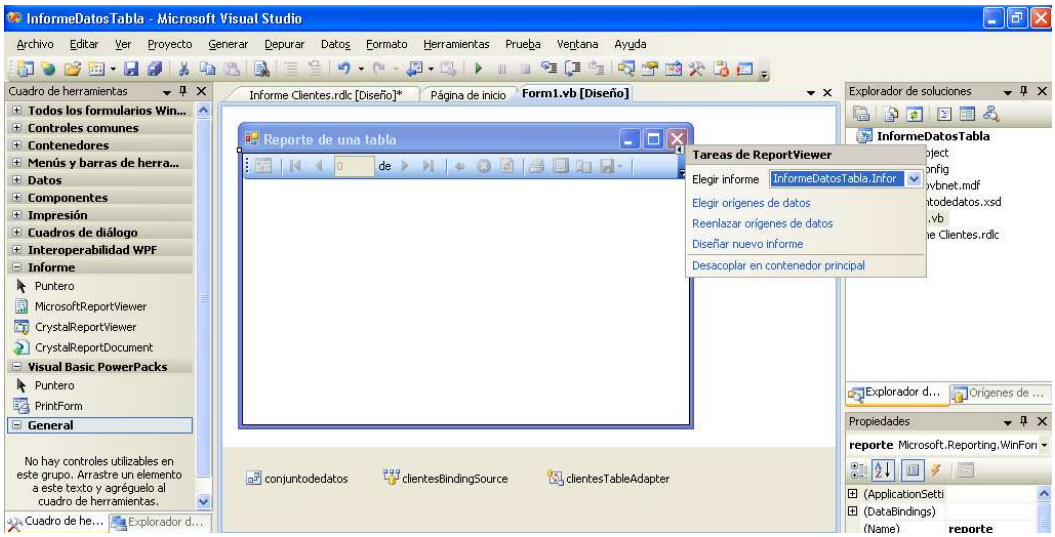
Elija la ficha de diseño del formulario y de clic sobre la flecha del objeto **reporte** y seleccione la opción **Elegir informe**.

Figura 12.64. Seleccionar opción Elegir informe.



Allí escoja **InformeDatosTabla.Informe Cliente.rdlc**, para visualizar el formulario con los objetos **conjuntodedatos**, **clientesBindingSource** y **clientesTableAdapter**.

Figura 12.65. Formulario con los objetos de manejo de datos.



- **Ejecutar el proyecto**

Al ejecutarse el proyecto, se visualizara la figura 4.56. Por medio de la barra de herramientas se podrá desplazar hacia las diferentes hojas del reporte, así como, imprimir el reporte, modificar el diseño del informe, configurar la página, cambiar el tamaño de visualización del informe y buscar texto dentro del informe.

Figura 12.66. Formulario con el reporte de los registros de la tabla clientes.

12.5.7. Relación de tablas.

Realizar un proyecto llamado **RelacionConComboBox,** que permita a un usuario seleccionar desde una lista desplegable el nombre de una empresa y visualizar en una cuadricula los pedidos que dicha empresa ha realizado.

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **Label,** 1 **ComboBox**, 1 **Button**, 1 **DataGridView**.

- **Establecer las propiedades de los objetos de la interfaz de usuario.**

Establezca las siguientes modificaciones a los controles:

Tabla 12.16. Propiedades de controles proyecto RelacionConComboBox.

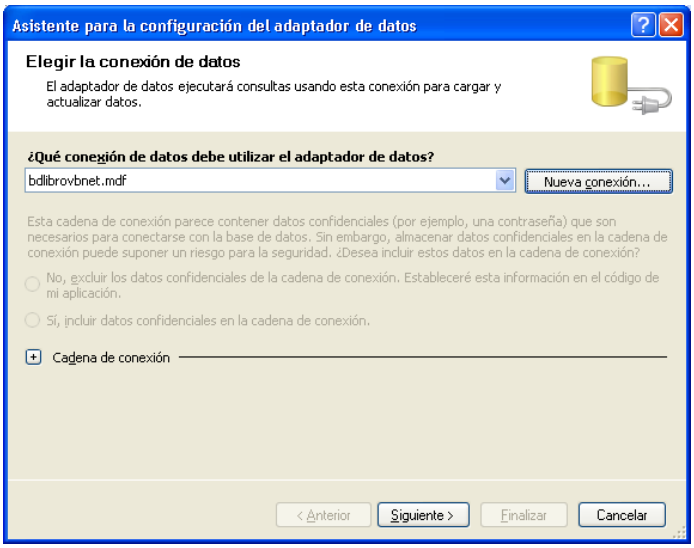
Control	Propiedad	Valor
Label1	Text	Seleccione empresa:
	name	etiquetaclientes
Button1	Text	Ver Pedidos
	name	boton
DataGridView1	Name	tablapedidos
ComboBox1	Name	lista
Form1	Name	formulario
	Text	Relación de tablas con un ComboBox.

Figura 12.67. Interfaz de usuario (RelacionConComboBox).

- **Establecer la conexión**

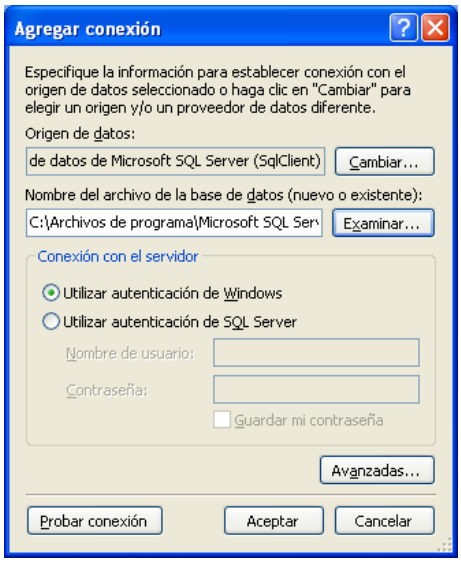
Desde la ficha **Datos** del cuadro de herramienta arrastre hacia el formulario el adaptador de datos **SqlDataAdapter** (Si no aparece el control, de clic derecho sobre la ficha **Datos** y ejecute la orden **Elegir elementos**, busque el control, selecciónelo en el cuadro de verificación y pulse el botón **Aceptar**.). Se visualizara la siguiente figura:

Figura 12.68. Ventana Elegir la conexión de datos



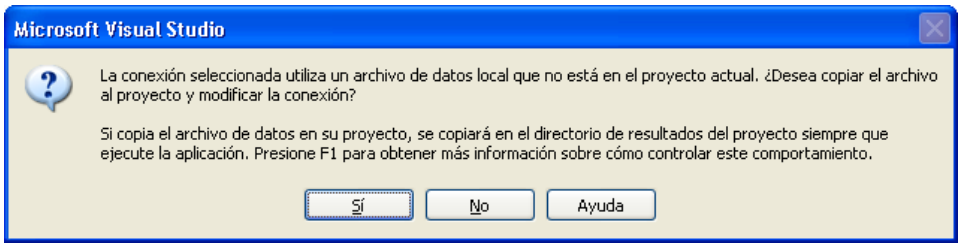
De clic en el botón **Nueva Conexión**. En la ventana que se visualiza elija como orígen de datos **Microsoft SQL Server** y la base de datos **bdlibrovbnet.**

Figura 12.69. Ventana Agregar conexión



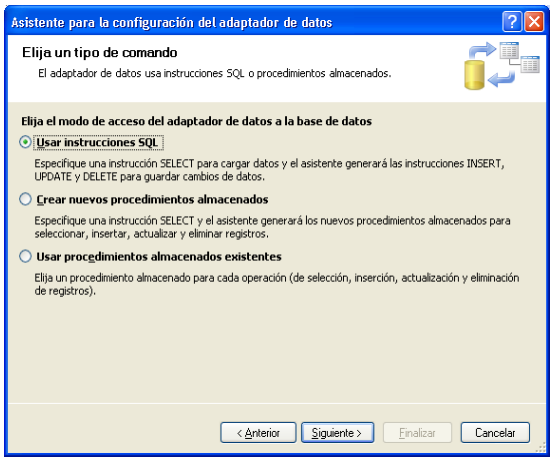
Pulse el botón **Aceptar**, para visualizar nuevamente la ventana de **Elegir la conexión de datos**. Allí pulse **Siguiente>** para visualizar el siguiente mensaje:

Figura 12.70. Ventana Microsoft Visual Studio



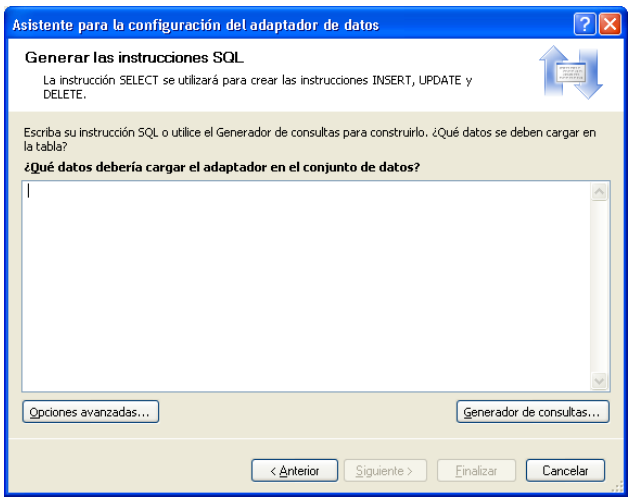
Pulse el botón **Sí** para copiar el archivo de datos en la carpeta donde guardo el proyecto. Se visualizara la ventana de elección de tipo de comando:

Figura 12.71. Ventana elección tipo de comando.



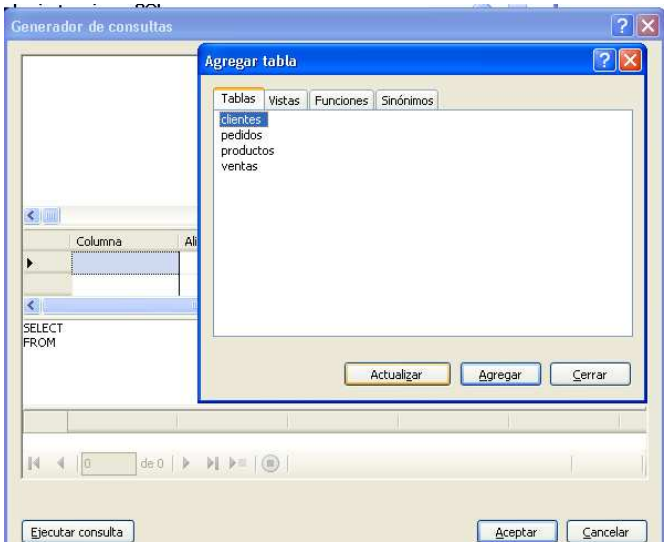
Por omisión esta seleccionada la opción **Usar instrucciones SQL**. Pulse **Siguiente>** para visualizar la ventana de generación de instrucciones SQL.

Figura 12.72. Ventana Generar las instrucciones SQL.



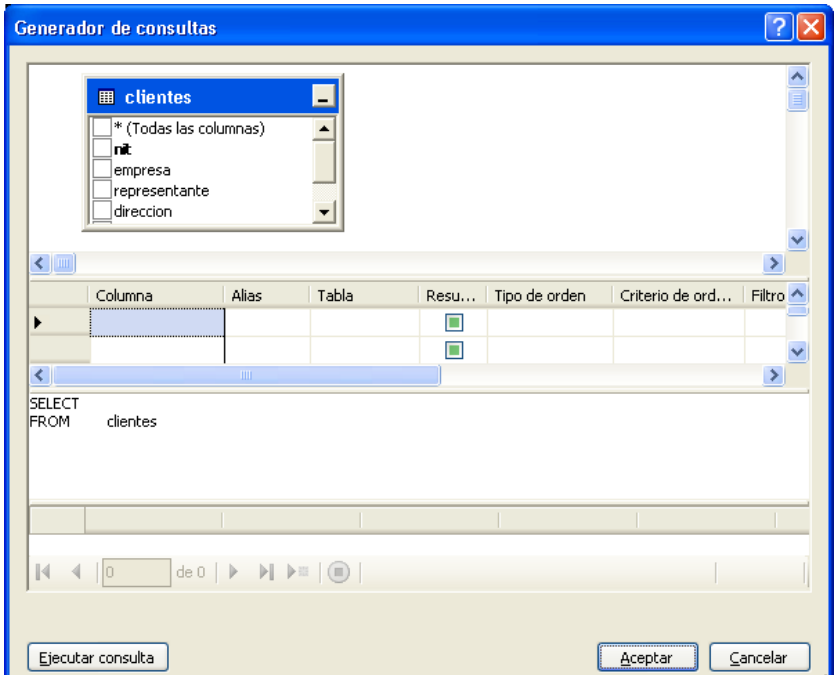
Aquí puede realizar la selección de los datos que debería cargar el adaptador de datos. Una forma es escribiendo en la ventana instrucciones SQL y la otra es pulsar el botón **Generador de consultas**. Para el ejemplo se pulsara el botón para visualizar la siguiente figura:

Figura 12.73. Ventana Generador de consultas.



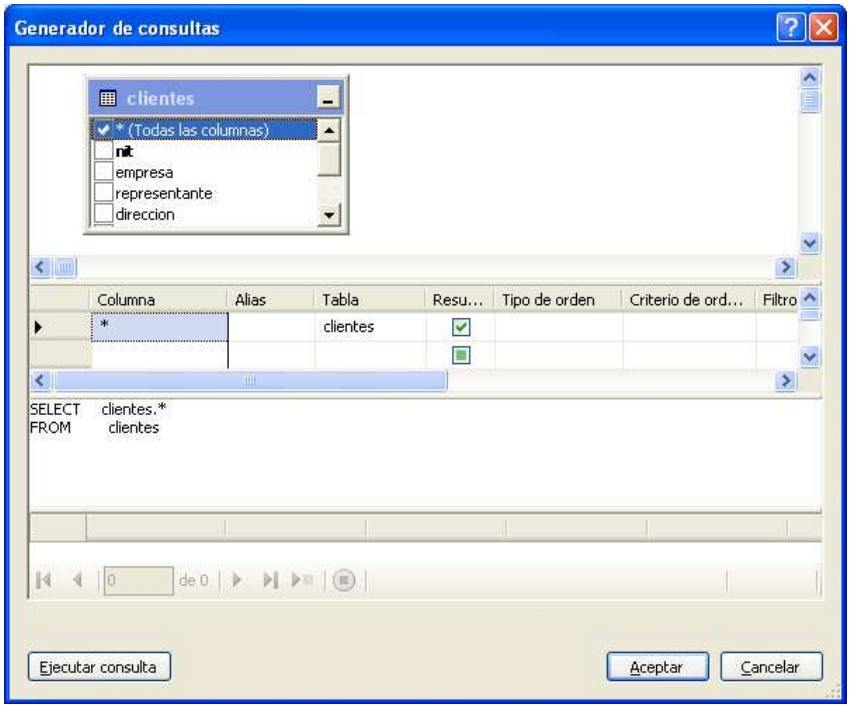
Seleccione la tabla **clientes**, pulse el botón **Agregar** y luego el botón **Cerrar**, se visualizara la siguiente figura:

Figura 12.74. Generador de consultas con la tabla clientes.



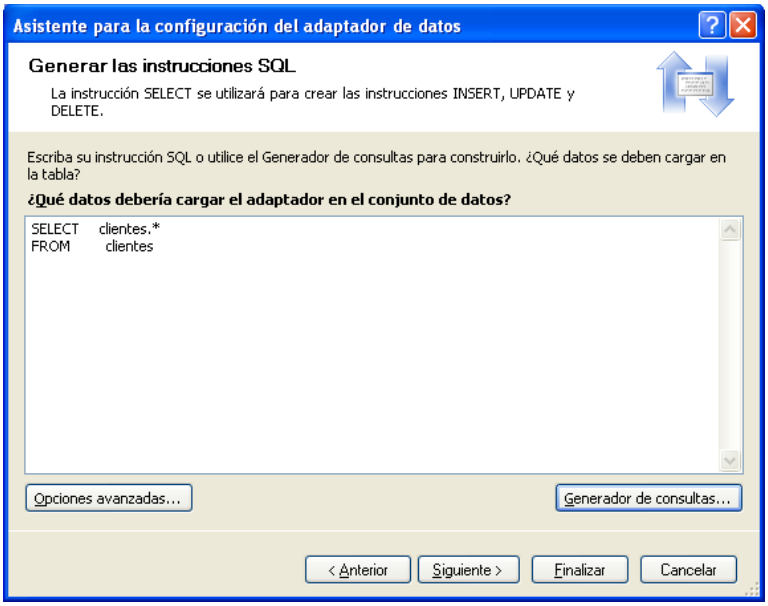
En esta nueva ventana seleccione la opción *** (todas las columnas)** o los campos que desea visualizar (Si desea visualizar la consulta seleccionada pulse el botón **Ejecutar consulta)**. Se obtendrá la siguiente figura:

Figura 12.75. Generador de consultas con los campos seleccionados (tabla clientes).



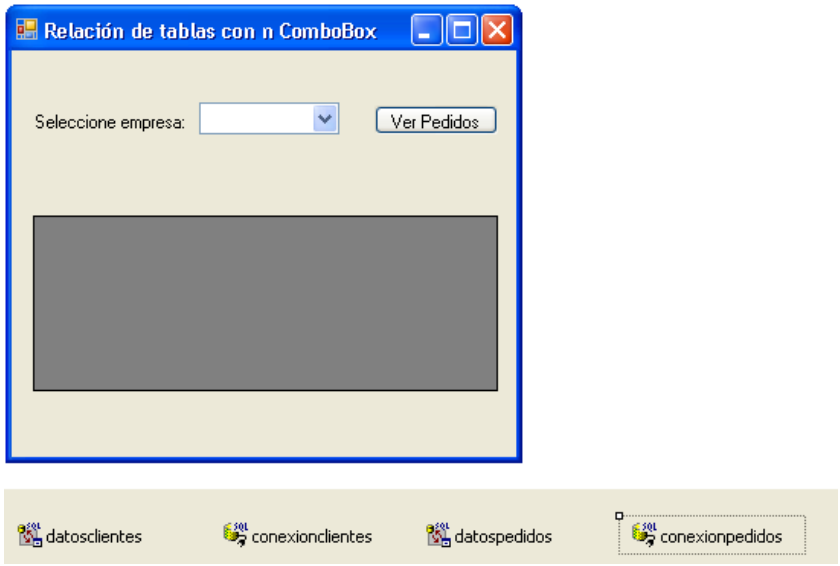
Pulse el botón **Aceptar**, y se visualizara la siguiente figura:

Figura 12.76. Generador de consultas con la instrucción SQL.



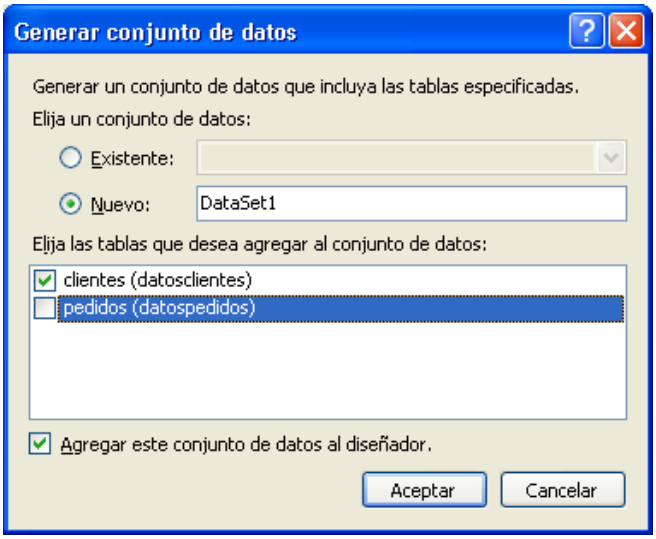
Al pulsar el botón **Finalizar,** se crearan los objetos: **SqlDataAdapter1** y **SqlConnection1.** Cambie el nombre de dichos objetos por **datosclientes y conexioncliente** respectivamente. Ahora seleccione nuevamente el control **SqlDataAdapter** de la ficha **Datos** y realice los mismos pasos que se realizaron anteriormente, pero esta vez seleccione la tabla **pedidos** y cambie los nombres de los objetos **SqlDataAdapter1** y **sqlConnection1** por **datospedidos** y **conexionpedidos**. Se visualizara la siguiente figura:

Figura 12.77. Formulario con los objetos SqlDataAdapter y sqlConnection.



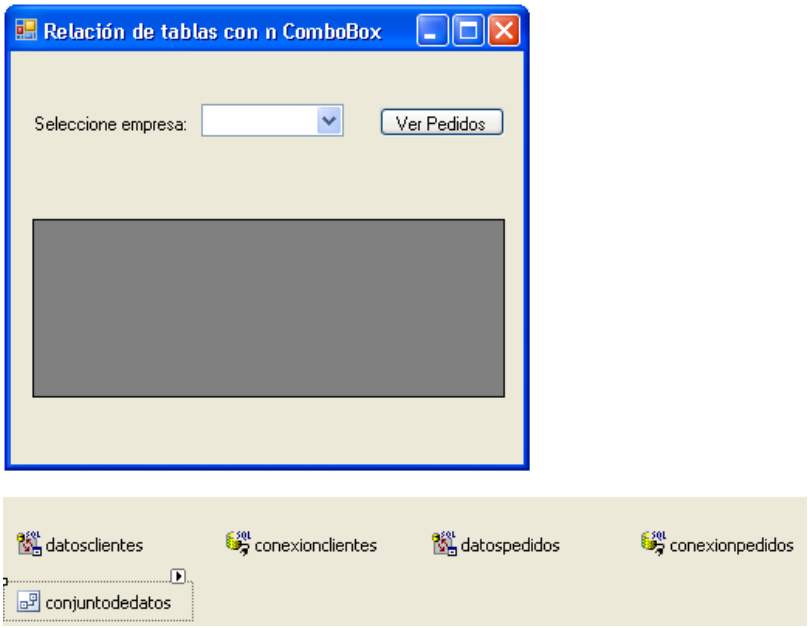
Seleccione la opción **Generar Conjunto de Datos** del menú **Datos** y se mostrara el siguiente cuadro de dialogo:

Figura 12.78. Ventana Generar conjunto de datos.



Elija la opción **Nuevo** y cambie el nombre **DataSet1** por **conjuntodedatos,** deshabilite el cuadro de verificación **pedidos (datospedidos)** y pulse el botón **Aceptar** para agregar el conjunto de datos a la aplicación, como lo muestra la figura:

Figura 12.79. Formulario con el objeto conjuntodedatos.



- **Escribir código**

a) Antes de la apertura de la clase **formulario** se debe importar el siguiente espacio de nombres:

```
Imports System.Data.SqlClient
Public Class formulario
      ……
End Class
```

b) Después de la apertura de la clase **formulario** y antes de los procedimientos **sub**, inicialice las siguientes variables u objetos globales:

```
Public class formulario
   Dim vistapedidos As DataView
         ……..
         ……..
End class
```

Se inicializa un objeto global llamado **vistapedidos** de tipo **DataView** para obtener una vista de los datos.

c) De doble clic sobre el formulario para abrir el editor de código del procedimiento **formulario_ Load** y escriba el siguiente código:

```
datosclientes.Fill(conjuntodedatos, "clientes")
```

```
datospedidos.Fill(conjuntodedatos, "pedidos")
lista.DataSource = conjuntodedatos.Tables("clientes")
lista.DisplayMember = conjuntodedatos.Tables("clientes").Columns(1).ToString
lista.ValueMember = conjuntodedatos.Tables("clientes").Columns(0).ToString
vistapedidos = conjuntodedatos.Tables("pedidos").DefaultView
```

Se rellenan los objetos **datosclientes** y **datospedidos** con el conjunto de datos y su respectiva tabla. Al objeto **lista** en su propiedad **DataSource** se le asigna la tabla **clientes** del conjunto de datos, a la propiedad **DisplayMember** se le asigna el valor del segundo campo de la tabla **clientes** (columna uno (1) – empresa) y a la propiedad **ValueMember** se le asigna el valor del primer campo de la tabla **clientes** (columna cero (0) - nit). Por último al objeto **vistapedidos** se le asigna la tabla **pedidos** del conjunto de datos y utilizando la propiedad **DefaultView** se personaliza la vista de los datos.

d) De doble clic sobre el objeto **boton** para abrir el editor de código y escriba el siguiente código:

```
Dim datoempresa As String = lista.SelectedValue
vistapedidos.RowFilter = "nit = '" & datoempresa & "'"
tablapedidos.DataSource = vistapedidos
```

Se inicializa una variable llamada **datoempresa** de tipo **String** que almacenara lo seleccionado en el objeto **lista** por intermedio de la propiedad **SelectedValue**. Se utiliza la propiedad **RowFilter** del objeto **vistapedidos** para filtrar los registros por el campo **nit** de acuerdo al valor de la variable **datoempresa**. Por último se asigna a la propiedad **DataSource** del objeto **tablapedidos** el contenido del objeto **vistapedidos**.

e) De doble clic sobre el objeto **lista** para abrir el editor de código del procedimiento **lista_SelectedIndexChanged** y escriba el siguiente código:

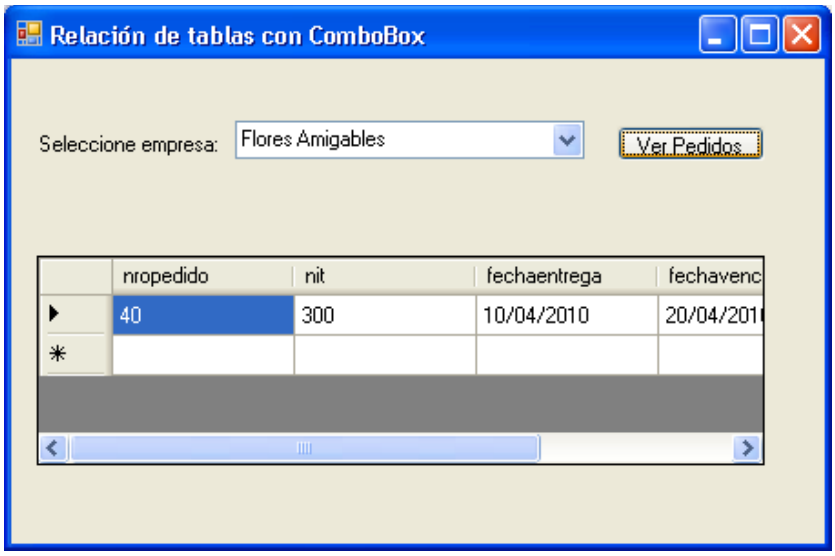
```
tablapedidos.DataSource = Nothing
```

Utilizando la palabra clave **nothing** se libera en memoria el contenido que tenga la propiedad **DataSource** del objeto **tablapedidos.**

- **Ejecutar el proyecto**

Al ejecutarse el proyecto, se visualizara la figura 12.67., mostrándose en el objeto **lista** el primer nombre de empresa de la tabla **clientes** (Si se desea ver los nombres de empresas ordenados alfabéticamente cambie el valor de la propiedad **sorted** del objeto **lista** por **True**). Si selecciona un nombre de empresa y se pulsa el botón **Ver Pedidos**, se visualizara en la cuadricula todos los registros que estén relacionados con la tabla **pedidos.**

Figura 12.80. Relación de las tablas clientes – pedidos con DataGridView.



12.6. Ejercicios bases de datos.

1. Crear un programa que permita realizar una conexión a una base de datos y mostrar datos de una tabla en un control ListView
2. Elaborar un programa que permita realizar una conexión a una base de datos y eliminar registros.
3. Escribir un programa que permita visualizar los registros de una tabla en campos de texto. El usuario deberá poderse desplazar por cada registro de la tabla utilizando los botones: Primero, Siguiente, Anterior y Último.
4. Diseñar un programa que permita visualizar los registros de una tabla en un control DataGridView. El usuario deberá poderse desplazar por cada registro de la tabla utilizando los botones: Primero, Siguiente, Anterior y Último.
5. Hacer un programa que permita escribir sentencias SQL en un campo de texto y visualizar los resultados de dicha sentencia en una cuadricula.
6. Realizar un programa que permita visualizar un reporte los registros relacionados cuyos nit sean mayores de 500.
7. Hacer un programa que permita visualizar un reporte los registros relacionados cuyas empresas empiecen por la letra C.
8. Realizar un programa que permita hacer una conexión a una base de datos y visualizar los registros cuyo nit sea menor o igual a 500 en un ListView.
9. Realizar un programa que permita realizar una relación entre tres tablas de una base de datos en SQL Server.
10. Hacer un programa que permita visualizar en un reporte los registros de una relación de dos tablas.