

**UNIVERSIDAD TECNOLÓGICA NACIONAL  
FACULTAD REGIONAL GENERAL PACHECO**

**TÉCNICO SUPERIOR EN PROGRAMACIÓN  
PROGRAMACIÓN III**

**UNIDAD 3:**

***Acceso a base de datos***

**TSSI TAMARA HERRERA**

## **Contenidos de la unidad**

1. ¿Cómo trabajar con una base de datos desde visual?
2. Funcionamiento de las variables SQL
3. Procesos comunes a todas las conexiones
4. Aplicación 1: Cargar un GridView
  - Carga a partir de un DataReader
  - Carga a partir de un DataSet
5. Aplicación 2: Cargar un DropDownList
  - Carga a partir de un DataReader
  - Carga a partir de un DataSet
6. Aplicación 3: Recorrer elementos de un DataReader y de un DataSet
7. Aplicación 4: Procedimientos almacenados

## Introducción

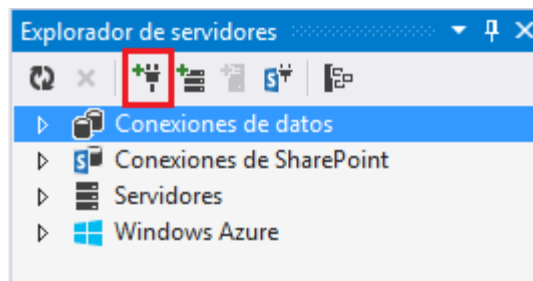
En esta unidad aprenderemos a realizar conexiones a bases de datos. Para esto es importante que importen la base de datos Neptuno al Management Studio de SQL. En el aula virtual se encuentra disponible un archivo que explica cómo realizar esta importación.


### ¿Cómo trabajar con una base de datos desde Visual?

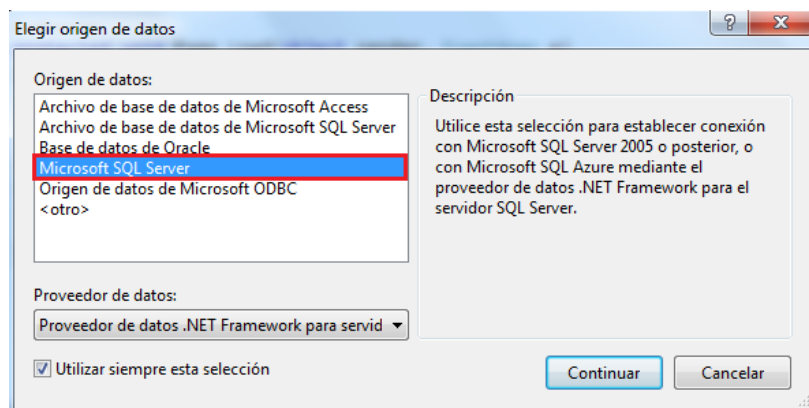
¿Es necesario incorporarla? No, no es necesario. Es una herramienta que trae Visual Studio para trabajar las bases de datos desde su entorno, pero si queremos podemos trabajar desde el Management Studio de SQL.

- Luego de crear un Sitio Web en Visual Studio podemos hacer lo siguiente:

*Ir a la barra de herramientas -> Ver Explorador de Servidores*



1. Clic sobre el botón  para agregar una base de datos.
2. Aparecerá la siguiente ventana.



✓ Seleccionar origen de datos: Microsoft SQL Server. Clic en continuar.

3. Luego aparecerá la siguiente ventana.

Agregar conexión

Especifique la información para establecer conexión con el origen de datos seleccionado o haga clic en "Cambiar" para elegir un origen y/o un proveedor de datos diferente.

Origen de datos: **A.**

Microsoft SQL Server (SqlClient) Cambiar...

Nombre del servidor: **B.**

localhost\\sqlexpress Actualizar **C.**

Conexión con el servidor

☒ Usar autenticación de Windows

☐ Usar autenticación de SQL Server

Nombre de usuario:

Contraseña:

☐ Guardar mi contraseña

Establecer conexión con una base de datos

☒ Seleccione o escriba el nombre de la base de datos: **D.**

Neptuno Examinar...

☐ Asociar con un archivo de base de datos:

Nombre lógico:

Avanzadas...

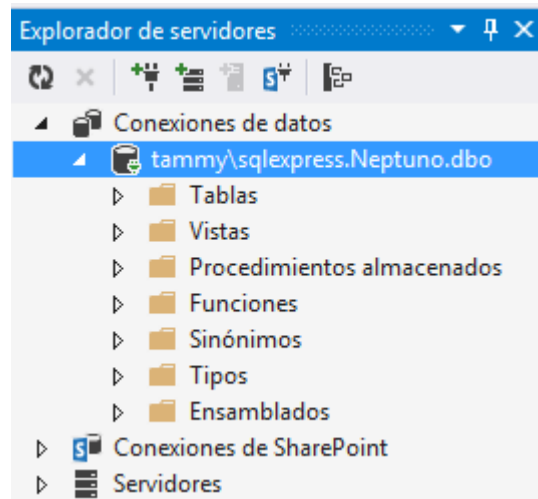
Probar conexión **E.** Aceptar Cancelar

A. Repite lo que seleccionamos en el cuadro anterior.

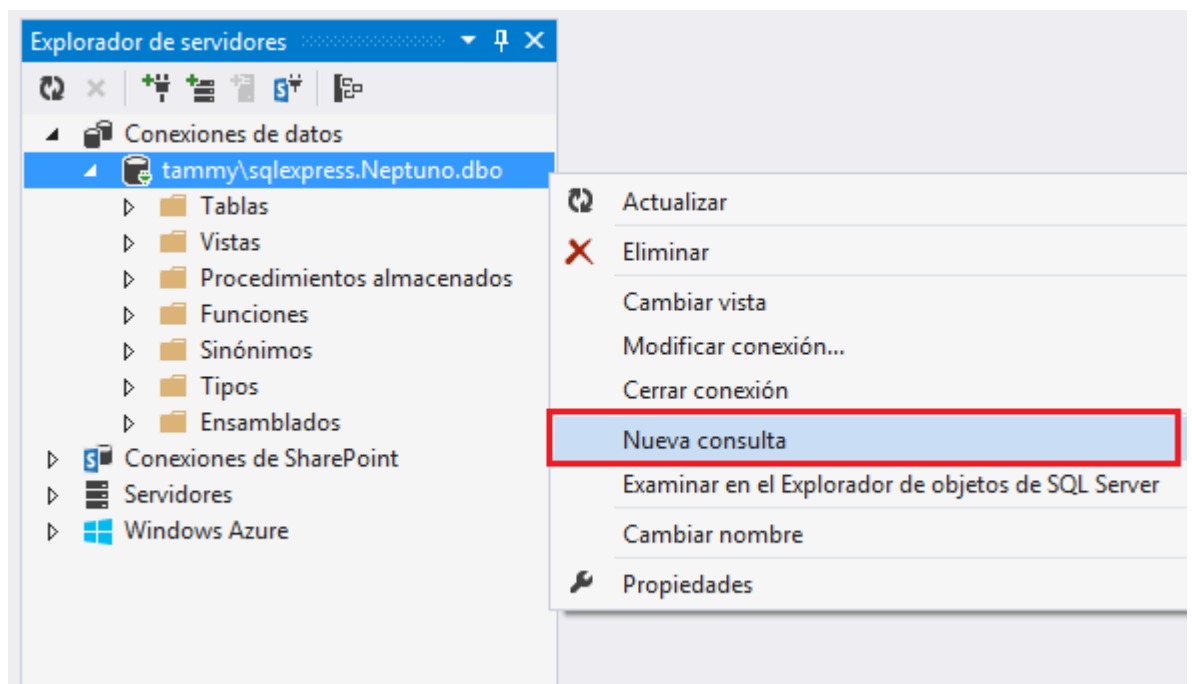
B. Elegir el nombre del servidor puede ser *localhost* o *localhost\\sqlexpress* depende de cual hayan instalado.

C. Clic en actualizar.

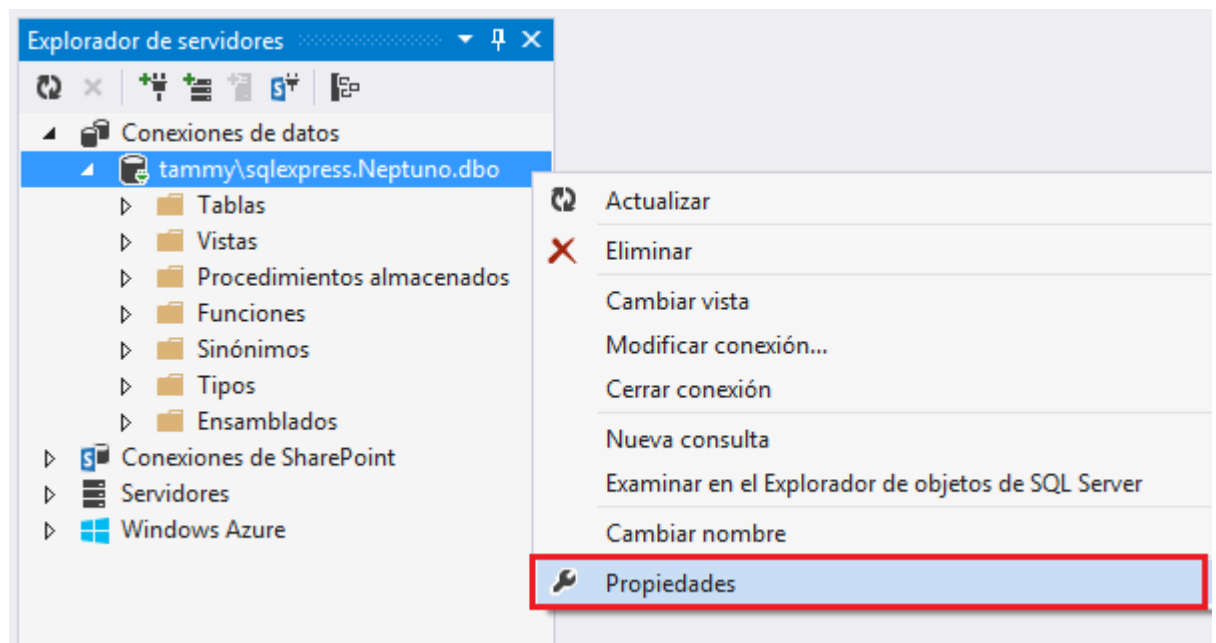
- D. Elegir el nombre de la base de datos, esta lista se tiene que completar automáticamente. En este caso, seleccionaremos la base de datos Neptuno.
  - E. Clic en aceptar.
4. Podrán observar la base de datos dentro del explorador de servidores.



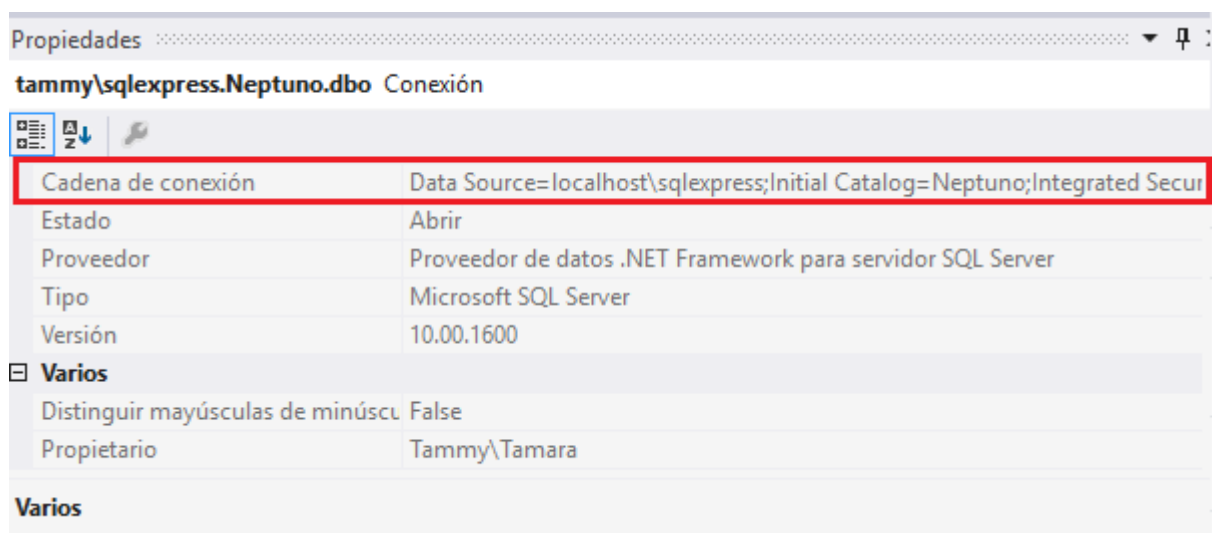
**Haciendo clic derecho sobre la base de datos podemos realizar consultas**



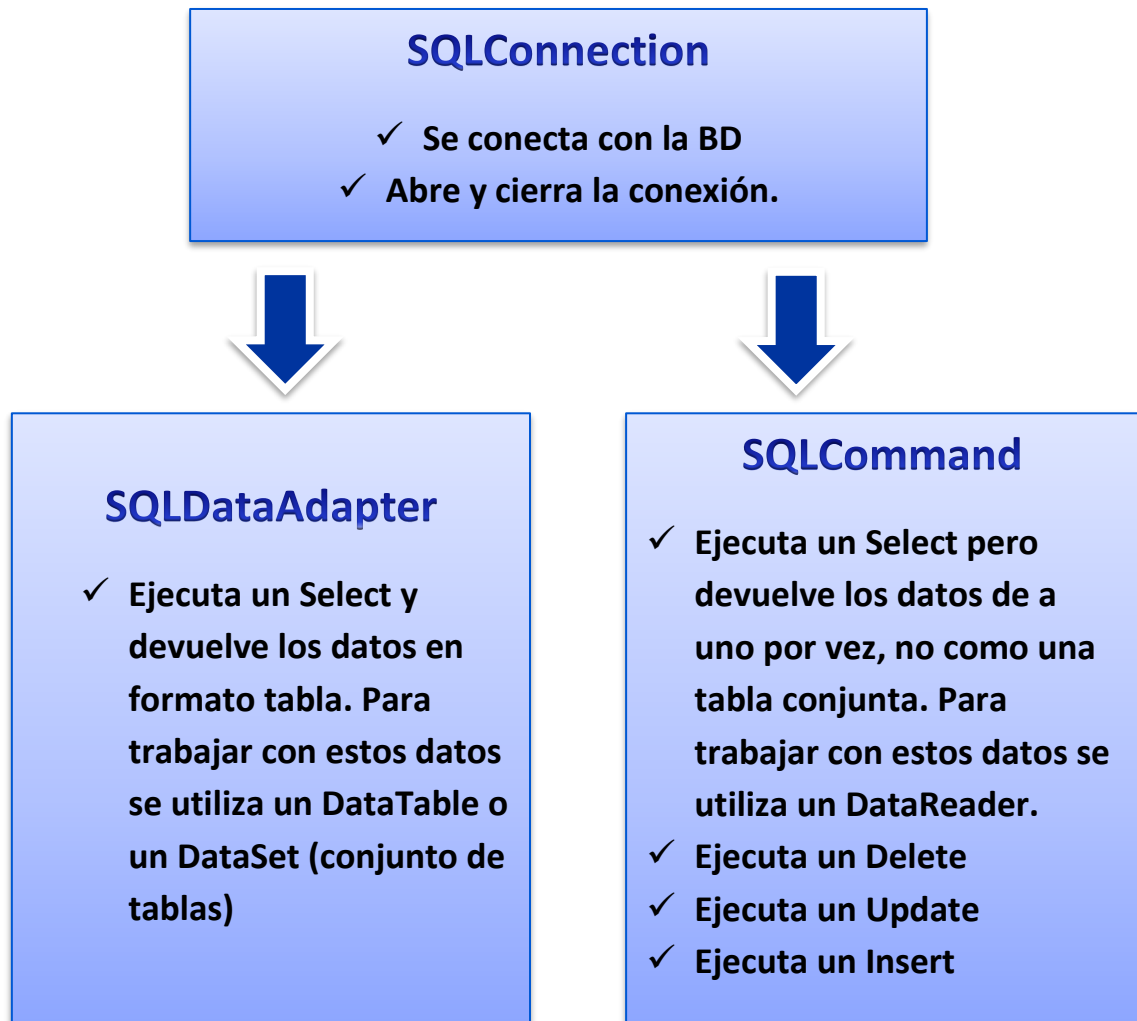
**Haciendo clic derecho sobre la base de datos y seleccionando propiedades puedo obtener la cadena de conexión.**



Dentro de las propiedades de la base de datos podemos observar la Cadena de Conexión. La cadena de conexión contiene la ruta para conectarnos a esa base de datos.



## Funcionamiento de variables SQL



- ✓ Entonces, como conclusión podemos decir que siempre vamos a utilizar la variable SqlConnection tanto para abrir como para cerrar la conexión.
- ✓ Para leer datos podemos utilizar un SqlDataAdapter o un SqlCommand, más adelante veremos las diferencias.
- ✓ Tanto para realizar Delete, Update e Insert, utilizaremos un SqlCommand.

## Procesos comunes a todas las conexiones

### Incorporar la librería SQL

Para declarar de forma más acotada las variables, deberemos incluir el atajo a la librería en el encabezado del formulario:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.Data.SqlClient;
```



### Abrir la conexión

1. Para comenzar puedo guardar la cadena de conexión de la BD en una variable String. Esta cadena de conexión la puedo obtener desde el explorador de servidores (como vimos anteriormente) o simplemente escribiéndola a través de código.

```
string rutaBD;  
rutaBD="Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
```

- ✓ Data Source: Nombre del servidor
- ✓ Initial Catalog: Nombre de la base de datos

2. Ahora lo que debemos hacer es asociar una variable SqlConnection a ese String.

#### Primera forma posible de inicializar una conexión

```
SqlConnection cn = new SqlConnection(rutaBD);  
cn.Open();
```

#### Segunda forma posible de inicializar una conexión

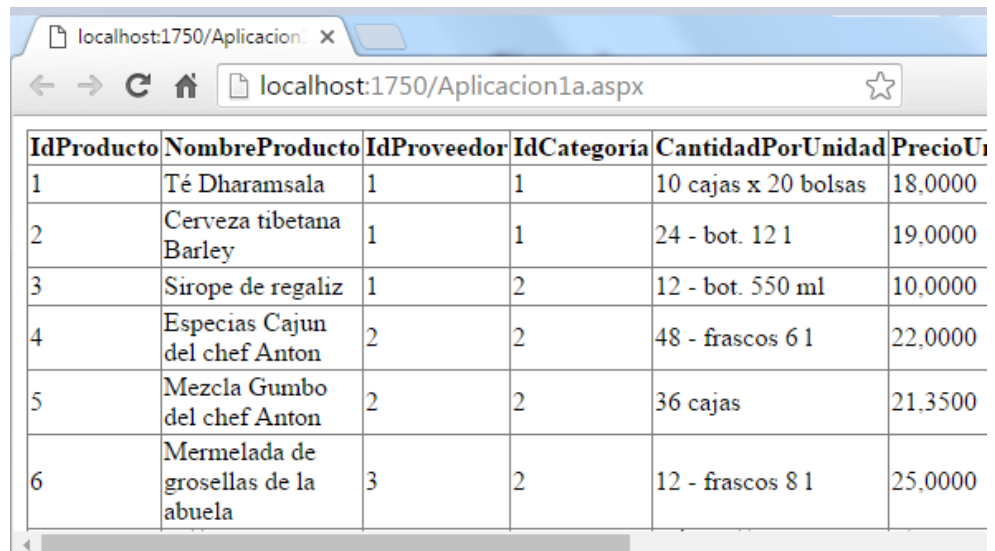
```
SqlConnection cn = new SqlConnection();  
cn.ConnectionString = rutaBD;  
cn.Open();
```



## Carga de un GridView

### ➤ Aplicación 1a: Carga de un GridView a partir de un DataSet

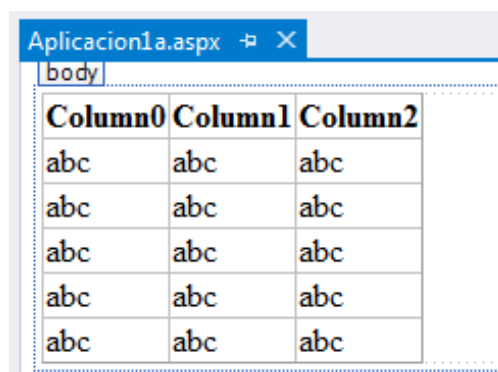
El usuario al abrir la página podrá observar un GridView cargado con los productos de la base de datos Neptuno, no ingresará nada.



IdProducto	NombreProducto	IdProveedor	IdCategoria	CantidadPorUnidad	PrecioU
1	Té Dharamsala	1	1	10 cajas x 20 bolsas	18,0000
2	Cerveza tibetana Barley	1	1	24 - bot. 12 l	19,0000
3	Sirope de regaliz	1	2	12 - bot. 550 ml	10,0000
4	Especias Cajun del chef Anton	2	2	48 - frascos 6 l	22,0000
5	Mezcla Gumbo del chef Anton	2	2	36 cajas	21,3500
6	Mermelada de grosellas de la abuela	3	2	12 - frascos 8 l	25,0000

#### Pasos a seguir:

1. Crear un WebForms llamado: **Aplicación1a.aspx**
2. La vista diseño solo está compuesta por un GridView cuya propiedad ID es **grdProductos**. Tiene que quedar de la siguiente manera:



Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

3. Agregar lo siguiente en el encabezado:

```
using System.Data.SqlClient;
```

## 4. Agregar el siguiente código en el Page Load:

```
protected void Page_Load(object sender, EventArgs e)
{
    string rutaNeptuno =
    "Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
    string consultasql = "select * from productos";

    if (!IsPostBack)
    {
        SqlConnection cnNeptuno = new SqlConnection(rutaNeptuno);
        SqlCommand cmd = new SqlCommand(consultasql, cnNeptuno);
        cnNeptuno.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        grdProductos.DataSource = dr;
        grdProductos.DataBind();
        cnNeptuno.Close();
    }
}
```

Código	Explicación
<pre>string rutaNeptuno = "Data Source=localhost\\s string consultasql = "select * from productos"</pre>	Guardo en dos variables string, la consulta y la cadena de conexión.
<pre>SqlConnection cnNeptuno = new SqlConnection(rutaNeptuno); cnNeptuno.Open();</pre>	Abro la conexión utilizando la variable que tiene la cadena de conexión.
<pre>SqlCommand cmd = new SqlCommand(consultasql, cnNeptuno);</pre>	Cargo la consulta y la conexión a un SqlCommand.
<pre>SqlDataReader dr = cmd.ExecuteReader();</pre>	Ejecuto el SqlCommand y los datos de esa consulta los guardo en un DataReader.
<pre>grdProductos.DataSource = dr; grdProductos.DataBind();</pre>	El DataReader lo vuelco en la grilla y realizó el DataBind que fuerza que se carguen los elementos, sin este no se cargaría la grilla.
<pre>cnNeptuno.Close();</pre>	Cierro la conexión.

## ¿Cuál es el aspecto negativo de guardar los datos en un DataReader?

La forma de trabajar un DataReader es la siguiente, va leyendo fila a fila los datos que obtuvo desde la base de datos y una vez que los leyó, ya no podremos consultarlos otra vez. Con esto me refiero a lo siguiente:

Después de realizar esta línea:

```
grdProductos.DataSource = dr;
```

El DataReader quedará vacío por lo que ya no lo deberíamos utilizar.

Si quisiéramos hacer lo siguiente:

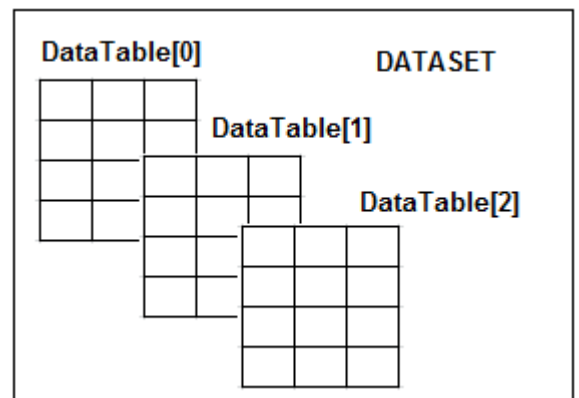
```
grdProductos.DataSource = dr;  
grdArticulos.DataSource = dr;
```

El GridView grdProductos quedaría cargado y el GridView grdArticulos quedaría vacío.

## ¿Hay alguna forma de consultar datos de una base de datos y que me queden guardados?

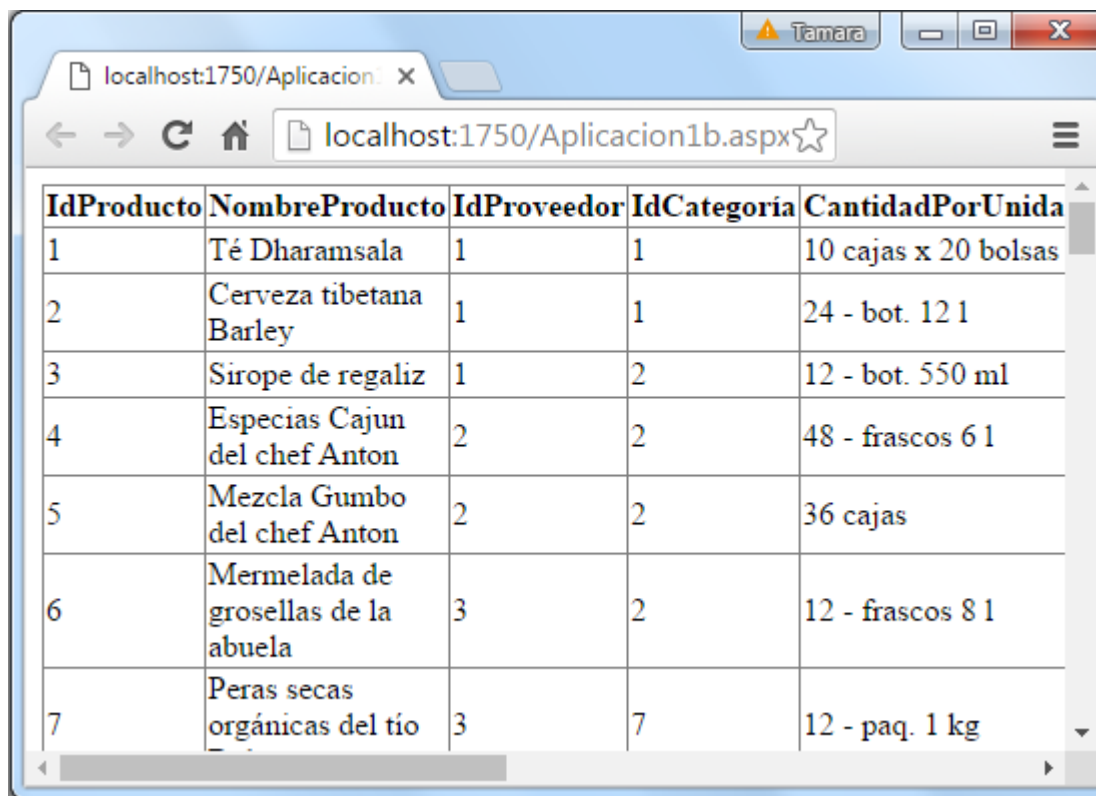
¡Sí!, lo que podemos hacer es consultar los datos de la base de datos y guardarlos dentro de una variable en mi programa. Esta variable que vamos a utilizar es un DataSet. Un DataSet lo podemos leer varias veces y no borra la información.

Un DataSet está compuesto por un conjunto de tablas (DataTable). Cada consulta que realizaremos la guardaremos dentro de una tabla de ese DataSet. Para realizar esto utilizaremos un SqlDataAdapter.



### ➤ Aplicación 1b: Carga de un GridView a partir de un DataSet

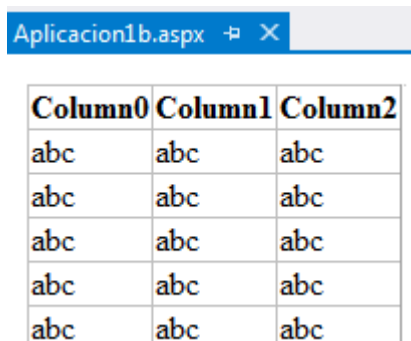
El usuario al abrir la página podrá observar un GridView cargado con los productos de la base de datos Neptuno, no ingresará nada.



IdProducto	NombreProducto	IdProveedor	IdCategoría	CantidadPorUnidad
1	Té Dharamsala	1	1	10 cajas x 20 bolsas
2	Cerveza tibetana Barley	1	1	24 - bot. 12 l
3	Sirope de regaliz	1	2	12 - bot. 550 ml
4	Especias Cajun del chef Anton	2	2	48 - frascos 6 l
5	Mezcla Gumbo del chef Anton	2	2	36 cajas
6	Mermelada de grosellas de la abuela	3	2	12 - frascos 8 l
7	Peras secas orgánicas del tío	3	7	12 - paq. 1 kg

#### Pasos a seguir:

1. Crear un WebForms llamado: **Aplicación1b.aspx**
2. La vista diseño tiene que quedar con solo un gridview llamado **grdProductos**



Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

3. Agregar lo siguiente en el encabezado:

```
using System.Data;
using System.Data.SqlClient;
```

4. Código a incorporar en el evento Page load:

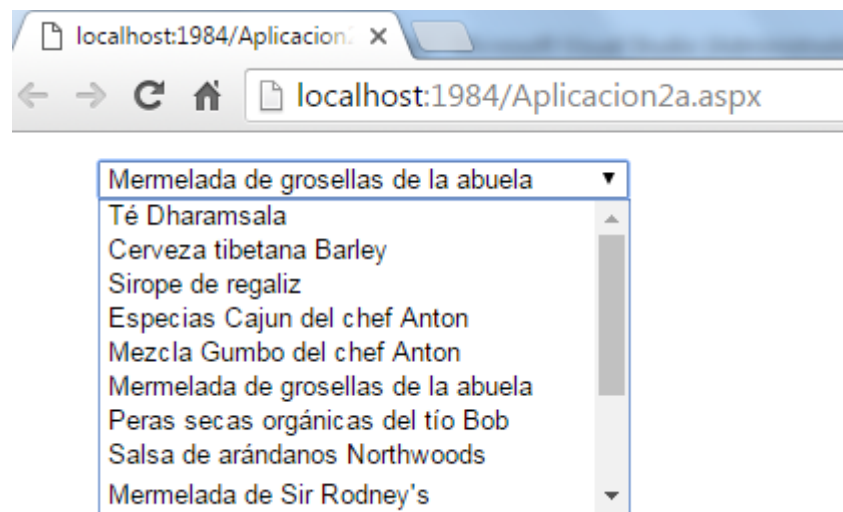
```
string rutaBD=
"Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
String consulta = "Select * from productos";

if (!IsPostBack)
{
    DataSet ds = new DataSet();
    SqlConnection cn = new SqlConnection(rutaBD);
    cn.Open();
    SqlDataAdapter adaptador = new SqlDataAdapter(consulta,cn);
    adaptador.Fill(ds,"Productos");
    grdProductos.DataSource = ds.Tables["Productos"];
    grdProductos.DataBind();
    cn.Close();
}
```

Código	Explicación
<pre>string rutaNeptuno = "Data Source=localhost\\s string consultasql = "select * from productos" DataSet ds = new DataSet();</pre>	Declaración de variables.
<pre>SqlConnection cnNeptuno = new SqlConnection(rutaNeptuno); cnNeptuno.Open();</pre>	Abro la conexión.
<pre>SqlDataAdapter adaptador = new SqlDataAdapter(consulta,cn);</pre>	Cargo la consulta y la conexión a un sqlDataAdapter.
<pre>adaptador.Fill(ds,"Productos");</pre>	El Fill se encarga de ejecutar la transacción y guardarla dentro del DataSet generando una tabla que se llama "Productos"
<pre>grdProductos.DataSource = ds.Tables["Productos"]; grdProductos.DataBind();</pre>	Vuelco la tabla a la grilla
<pre>cnNeptuno.Close();</pre>	Cierro la conexión.

## Aplicación 2: Cargar un DropDownList

El usuario al abrir la página podrá observar un DropDownList con los nombres de los productos. La vista diseño tiene que tener solo un DropDownList llamado **ddlProductos**



A continuación vamos a explicar dos diferentes formas de cargarlo:

En la aplicación2a.aspx, se carga el DropDownList mediante un DataReader.

```
string rutaNeptuno=
"Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
string consultasql = "select * from productos";
if (!IsPostBack)
{
    SqlConnection cnNeptuno = new SqlConnection(rutaNeptuno);
    SqlCommand cmd = new SqlCommand(consultasql, cnNeptuno);
    cnNeptuno.Open();
    SqlDataReader dr = cmd.ExecuteReader();
    ddlProductos.DataSource = dr;
    ddlProductos.DataTextField = "NombreProducto";
    ddlProductos.DataValueField = "IdProducto";
    ddlProductos.DataBind();
    cnNeptuno.Close();
}
```

Observaciones:

Código	Explicación
<code>SqlDataReader dr = cmd.ExecuteReader(); ddlProductos.DataSource = dr;</code>	Ejecuto la consulta y vuelco los datos del DataReader en el dropdownlist.
<code>ddlProductos.DataTextField = "NombreProducto";</code>	Seteo que columna de la base de datos quiero que muestre
<code>ddlProductos.DataValueField = "IdProducto";</code>	Seteo que columna de la base de datos quiero que valga internamente.
<code>ddlProductos.DataBind();</code>	Realizo la vinculación para que se pueda visualizar el contenido

**En la aplicación2b.aspx, se carga el DropDownList mediante un DataSet.**

```
string consulta="Select * from productos";
string rutaBD=
"Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
if (!IsPostBack)
{
    DataSet ds = new DataSet();
    SqlConnection cn = new SqlConnection(rutaBD);
    cn.Open();
    SqlDataAdapter adaptador = new SqlDataAdapter(consulta, cn);
    adaptador.Fill(ds, "Productos");
    ddlProductos.DataSource = ds.Tables["Productos"];
    ddlProductos.DataTextField = "NombreProducto";
    ddlProductos.DataValueField = "IdProducto";
    ddlProductos.DataBind();
    cn.Close();
}
```

Observaciones:

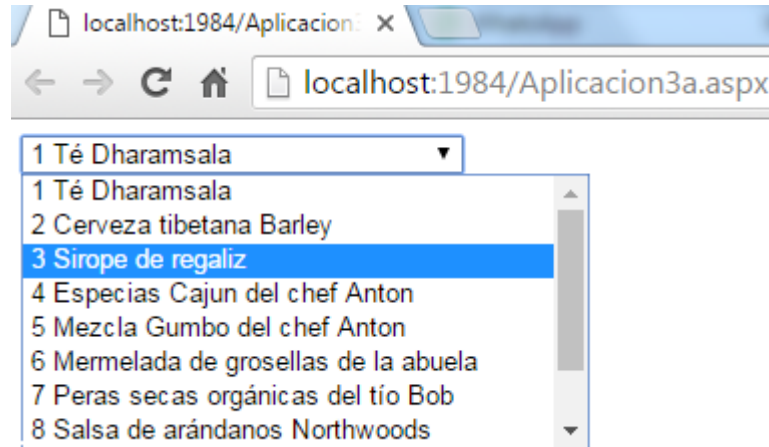
Fijarse que en las dos versiones se realiza casi el mismo código. La única diferencia se observa en el DataSource, en una versión lo igualo a un DataReader y en la otra, lo igualo a un DataTable.

### ¿Aspectos negativos de esta forma de cargar?

En estas versiones, solo puedo indicarle un valor al DataTextField y al DataValueField, por ende si quisiera mostrar más elementos, no podría.

## Aplicación 3: Cargar un DropDownList

El usuario al abrir la página podrá observar un DropDownList con los ID y nombres de los productos. La vista diseño tiene solo un DropDownList llamado **ddlProductos**



A continuación vamos a cargar de dos maneras el DropDownList. Recordar que no lo podemos hacer como el ejemplo anterior porque en este caso tenemos que setear dos valores para que se vean. Vamos a resolver este ejercicio primero mediante un DataReader y luego mediante un DataSet.

### ➤ Aplicación3a.aspx

#### Código a incorporar en el load

```
string rutaNeptuno=
"Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
string consultasql = "select * from productos";
if (!IsPostBack)
{
    SqlConnection cnNeptuno = new SqlConnection(rutaNeptuno);
    SqlCommand cmd = new SqlCommand(consultasql, cnNeptuno);
    cnNeptuno.Open();
    SqlDataReader dr = cmd.ExecuteReader();
    while (dr.Read())
    {
        ddlProductos.Items.Add(dr["IdProducto"] + " " + dr["NombreProducto"]);
    }
    cnNeptuno.Close();
}
```



### Observaciones

En este caso, voy a leer los datos del DataReader y los voy a ir agregando uno a uno al DropDownList. Cada vez que realizó un *dr.Read()*, leo una fila. Para obtener una columna de esa fila, lo hago con corchetes ["Nombre del campo de la base de datos"].

### ➤ **Aplicación3b.aspx**

#### **Código a incorporar en el load**

```
string consulta = "Select * from productos";
string rutaBD =
"Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True";
if (!IsPostBack)
{
    DataSet ds = new DataSet();
    SqlConnection cn = new SqlConnection(rutaBD);
    cn.Open();
    SqlDataAdapter adaptador = new SqlDataAdapter(consulta, cn);
    adaptador.Fill(ds, "Productos");
    foreach(DataRow dr in ds.Tables["Productos"].Rows)
    {
        ddlProductos.Items.Add(dr["IdProducto"] + " " + dr["NombreProducto"]);
    }
    cn.Close();
}
```

### Observaciones

En este caso, recorro la tabla que se generó en el DataSet, esa tabla se llama Productos. Al igual que la versión anterior recorro las filas y a través del corchete obtengo la columna.

## Procedimientos Almacenados

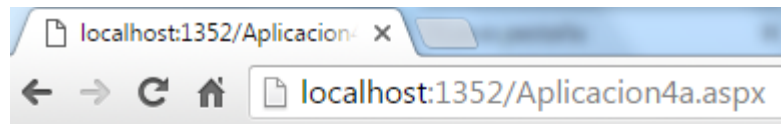
- Para ejecutar un procedimiento, vamos a utilizar un SqlCommand.

Vamos a cargar el siguiente procedimiento sobre la base de datos Neptuno

```
CREATE PROCEDURE spInsertarCategoria
(
    @IDCATEGORIA INT,
    @NOMBRECATEGORIA NVARCHAR(15),
    @DESCRIPCION NVARCHAR(40)
)
AS
INSERT INTO Categorías
(
    IDCATEGORÍA,
    NOMBRECATEGORÍA,
    DESCRIPCIÓN
)
VALUES
(
    @IDCATEGORIA,
    @NOMBRECATEGORIA,
    @DESCRIPCION
) RETURN
```

### Aplicación 4: Insertar datos

El botón guardará en la base de datos los datos ingresados por el usuario y en caso de que se haya efectuado la operación con éxito, mostrará un mensaje en la pantalla.



Id Categoría:	<input type="text" value="300"/>
Nombre Categoría:	<input type="text" value="Juguetería"/>
Descripción:	<input type="text" value="Edad 3 a 5 años"/>
	<input type="button" value="Guardar"/>

Operación realizada con éxito

*Para esto necesitamos, tres TextBox, un Button y un Label.*

- ID TextBox: txtIdCat, txtNombreCat, txtDescripciónCat
- ID Botón: btnAceptar.
- ID Label: lblMensaje

*Vamos a programar el siguiente código en el evento Clic del botón*

```
protected void Button1_Click(object sender, EventArgs e)
{
    int filasAfectadas;

    SqlConnection cn = new SqlConnection(rutaBD);
    cn.Open();

    SqlCommand cmd = new SqlCommand();
    cmd.Connection = cn;

    SqlParameter SqlParameteros = new SqlParameter();
    SqlParameteros = cmd.Parameters.Add("@IDCATEGORIA", SqlDbType.Int);
    SqlParameteros.Value = Convert.ToInt32(txtIdCat.Text);

    SqlParameteros = cmd.Parameters.Add("@NOMBRECATEGORIA", SqlDbType.VarChar);
    SqlParameteros.Value = txtNombreCat.Text;

    SqlParameteros = cmd.Parameters.Add("@DESCRIPCION", SqlDbType.VarChar);
    SqlParameteros.Value = txtDescripcionCat.Text;

    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "spInsertarCategoria";

    filasAfectadas = cmd.ExecuteNonQuery();

    if (filasAfectadas == 1)
    {
        lblMensaje.Text = "Operación realizada con éxito";
        txtDescripcionCat.Text = "";
        txtIdCat.Text = "";
        txtNombreCat.Text = "";
    }
}
```

Explicación del código:

Código	Explicación
<pre>string rutaBD = "Data Source=localhost\\sqlexpress;Initial Catalog=Neptuno;Integrated Security=True"; int filasAfectadas;</pre>	Declaro variables. La ruta en ese caso la declare por fuera de los eventos.
<pre>SqlConnection cn = new SqlConnection(rutaBD); cn.Open();</pre>	Creamos una variable sqlConnection y la asociamos a la ruta.
<pre>SqlCommand cmd = new SqlCommand(); cmd.Connection = cn;</pre>	Creamos una variable sqlCommand y la vinculamos con la variable sqlConnection.
<pre>SqlParameter SqlParametros = new SqlParameter(); SqlParametros = cmd.Parameters.Add("@IDCATEGORIA", SqlDbType.Int); SqlParametros.Value = Convert.ToInt32(txtIdCat.Text);</pre>	Le agregamos al SqlCommand, los parametros que debe recibir el procedimiento almacenado.
<pre>cmd.CommandType = CommandType.StoredProcedure; cmd.CommandText = "spInsertarCategoria";</pre>	Seteamos en ese SqlCommand, que va a ejecutar un procedimiento almacenado y cargamos el nombre del procedimiento.
<pre>filasAfectadas = cmd.ExecuteNonQuery();</pre>	ExecuteNonQuery, ejecuta la consulta y devuelve la cantidad de filas afectadas.
<pre>if (filasAfectadas == 1) {     lblMensaje.Text = "Operación realizada con éxito";     txtDescripcionCat.Text = "";     txtIdCat.Text = "";     txtNombreCat.Text = ""; }</pre>	Si hubo una fila afectada, entonces se agrego bien la categoría y borró los datos anteriores.