Departamento de Lenguajes y Sistemas Informáticos

ADO.NET (2/2)

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escuela Politécnica Superior
Universidad de Alicante

1. Summary
2. Disconnected environment
3. Data controls
4. Concurrence
5. Desconected vs connected environment

1 Summary

ActiveX Data Objects

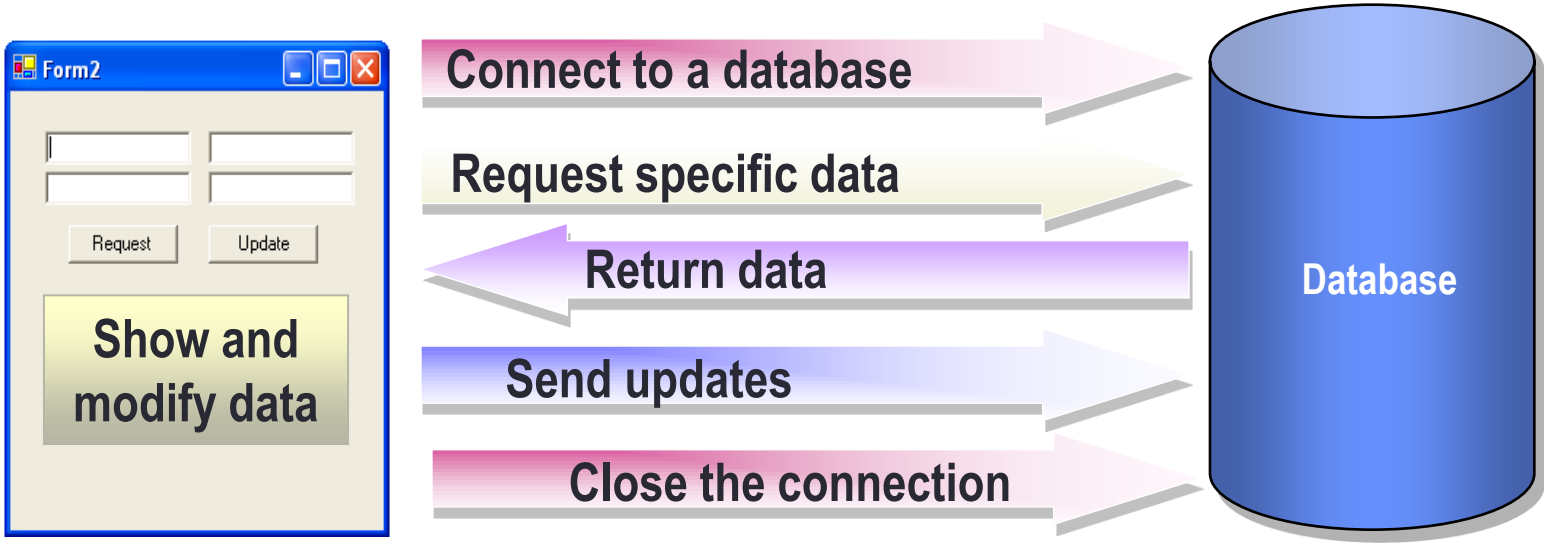
- ADO.NET is the technology used by the asp.net applications to communicate with the DB.
- Optimized for distributed applications (such as Web applications).
- Based on XML
- New set of model objects.
- **Connected and disconnected access to data.**

Connected environment

- A connected environment is that one in which the users are continuously connected to a data source
- Advantages:
  - The environment is easier to maintain
  - The concurrency is more easily controled
  - It is more probable that the data are updated that in other environments
- Disadvantages:
  - We need a constant network connection
  - Limited scalability

Connected environment(II)

CONNECTION OPEN

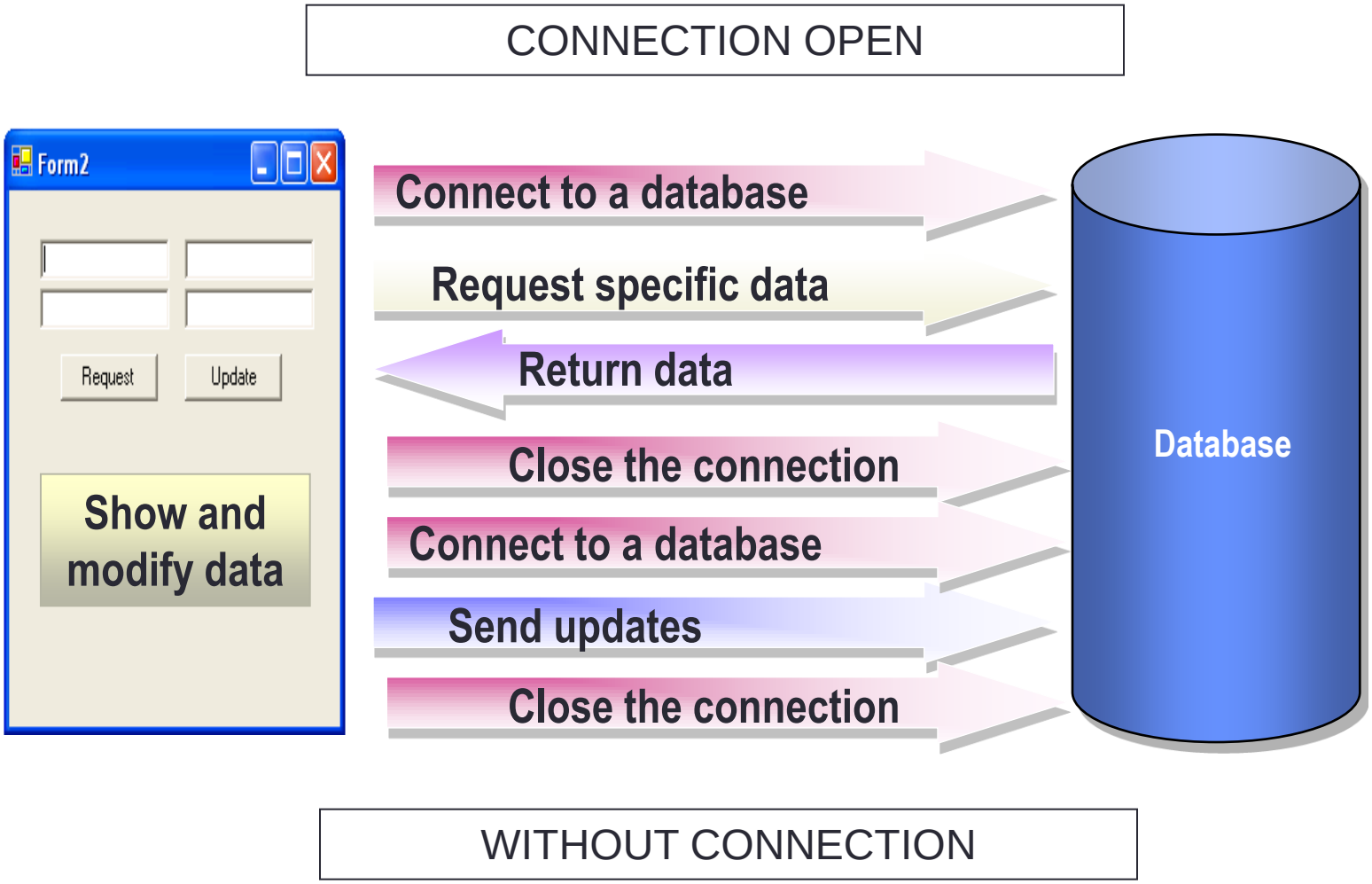


WITHOUT CONNECTION

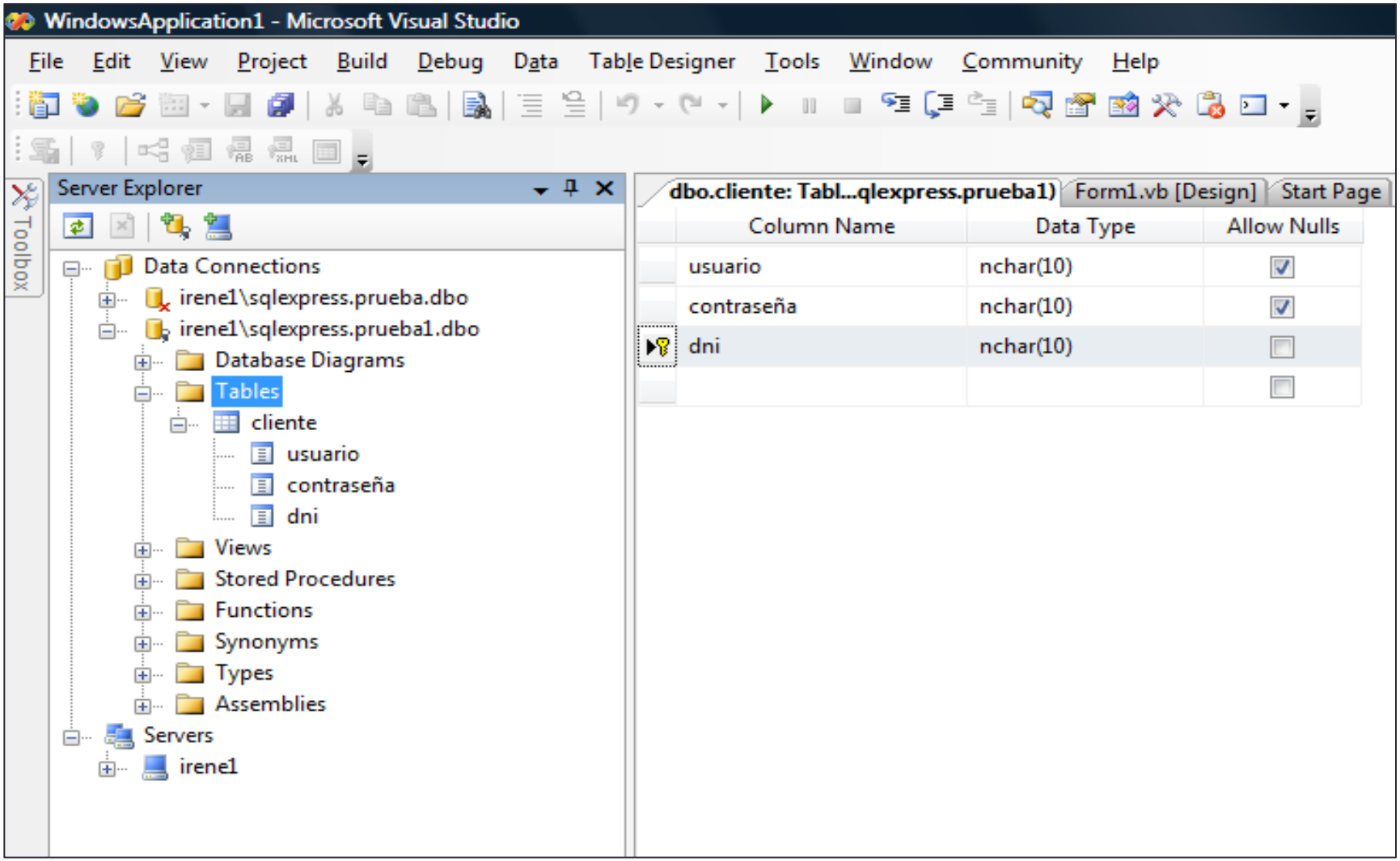
Disconnected environment

- In a disconnected environment data can be modified in an independent way and the updates are written in the database afterwards.
- Advantages:
  - Connections are used during the minimal time needed, allowing that less connections are used by more users
  - A disconnected environment improves the scalability and performance of the applications
- Disadvantages:
  - Data are not allways updated
  - There can be conflicts with updates that must be solved

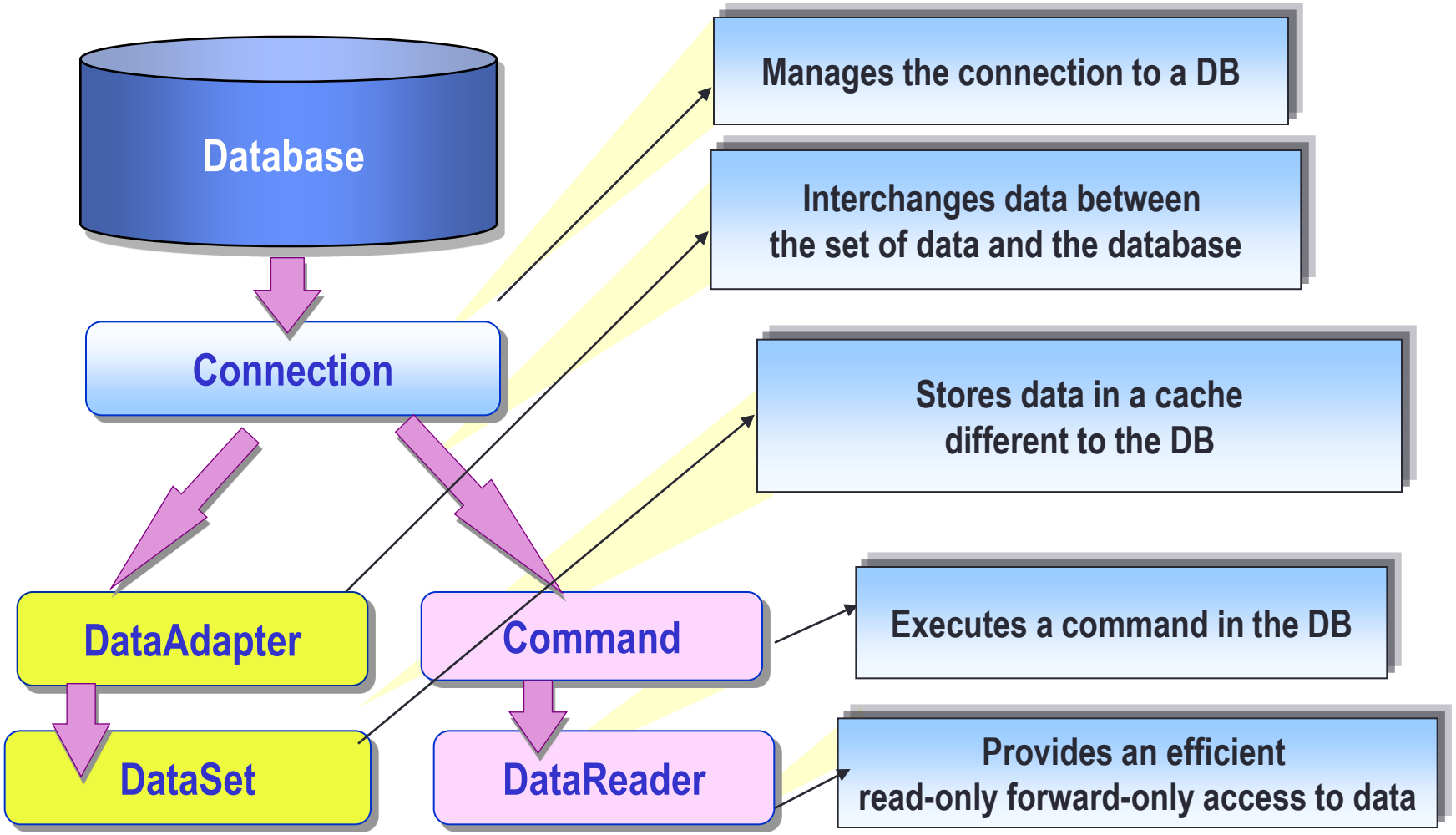
Disconnected environment (II)



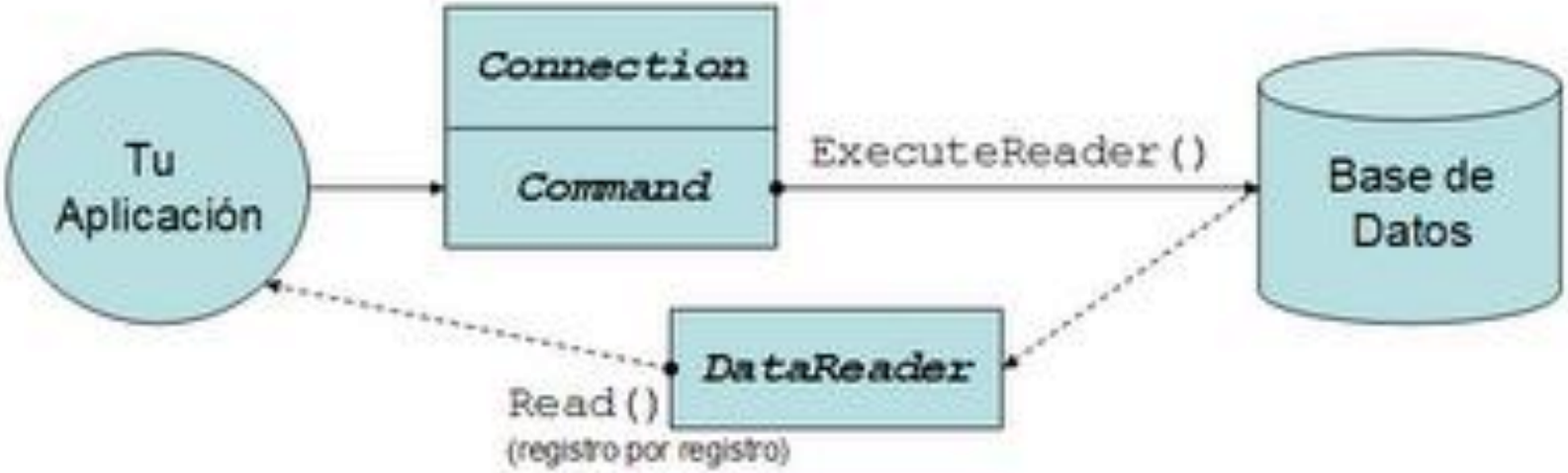
Client table…



ADO.NET Objects



Connected environment

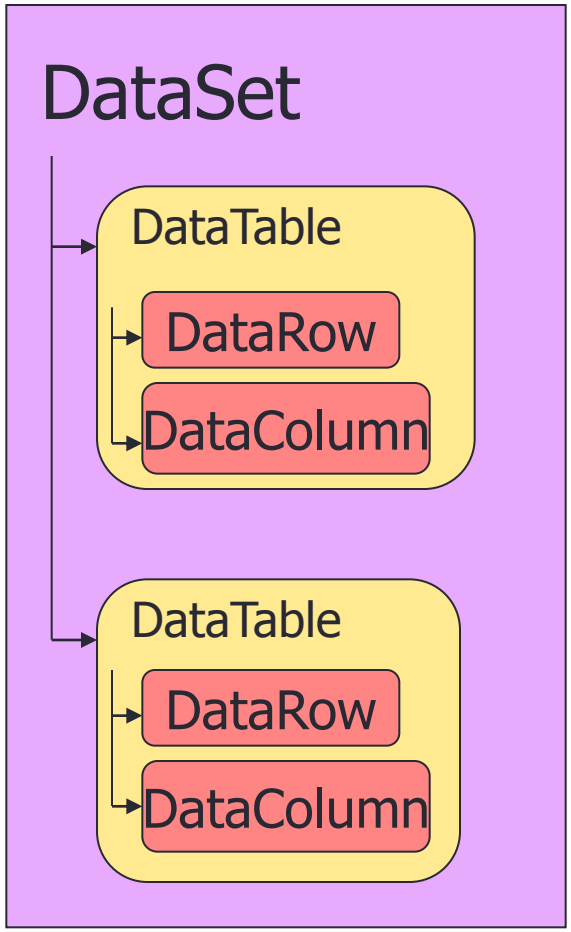


2 Disconnected environment

1. Connection Object

- Connection: used to stablish connections with the proper data provider (Open method).

2. DataSet Object



- New object **DataSet**: represents a relational **database in memory**:
  - We do not need a continous connection

DataSet Object (II)

- We can work with a DB that is a copy of the parts with which we want to work from the real DB, leaving free the connection.

- If we want to reflect our modifications in the real DB, we have to confirm our DataSet object.

DataRow, DataColumn Objects

- DataRow
  - Represents a unique row of information of the table.
  - We can access the individual values by using the name of the field or an index.
- DataColumn
  - They do not contain any real value
  - They store data about the column (data type, predefined value, restrictions..)

DataRelation, DataView Objects

- DataRelation
  - It specifies a relation parent/child between two different tables in a DataSet object.
- DataView
  - It provides a view over a DataTable.
  - Each DataTable contains at least one DataView (in the DefaultView property), which is used for the data binding.
  - DataView shows the data of DataTable with no changes or with special filtered or sorting options.

3. Adapter object

- The Adapter object is responsible of managing the connection.
- It is used to insert data in a **DataSet object**.
- The **DataAdapter** object uses commands to update the data source after doing modifications in the **DataSet object**.

DataAdapter (II)

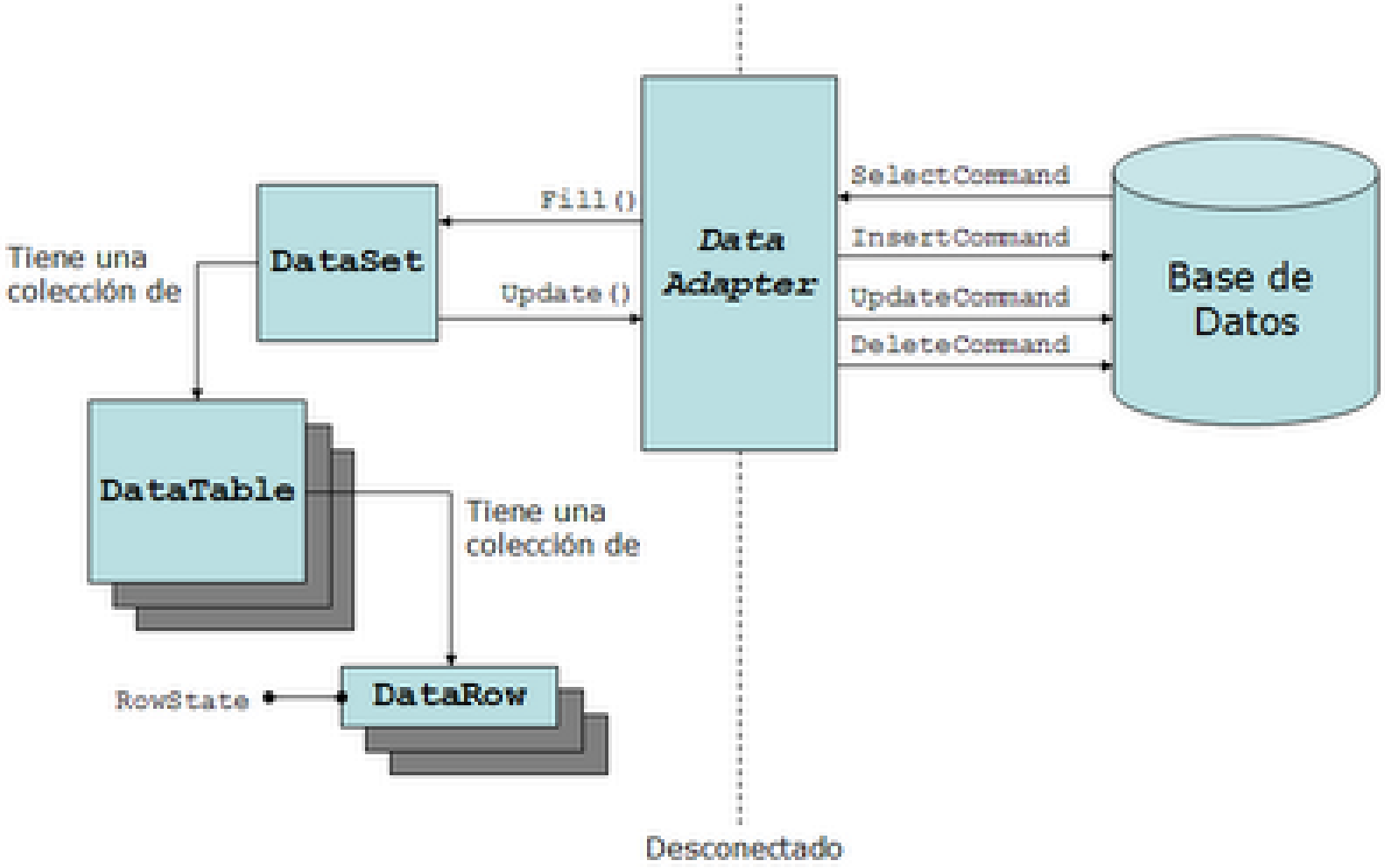
- An advantage of using a DataAdapter is that we **do not have to worry for openning and closing the connection**. It does it automatically when needed.

4. CommandBuilder Object

- This object (optional) is used by the **DataAdapter** for creating the needed SQL commands.

- We can also explicitly provide the SQL commands or by mean of stored procedures.

Disconnected environment



Remember…

- System.Data.OleDb and System.Data.SqlClient: responsible classes for the data access from SQL Sever and OLE DB sources.
- They include classes that when working with SQL will contain the Sql prefix and when using OleDB will contain the OleDb prefix:
  - SqlConnection and OleDbConnection
  - SqlDataAdapter and OleDbDataAdapter
  - SqlCommandBuilder and OleDbCommandBuilder
  - **BUT NOT IN DataSet (and dataRow, dataColumn…)**

EXAMPLE: data insertion

Usuario luis

Contraseña 1256

DNI 55555

Insertar usuario

Connection to a DB in Sql Server

Import namespaces

```
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.SqlTypes;
```

Create the connection

```
string s = "data source=.\\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|\\Database1.mdf;
User Instance=true";
SqlConnection c=new SqlConnection(s);
```

Definition of a Select Command

- For recovering the data we need:
  - ~~A SQL sentence which selects the desired information~~
  - ~~A Command object which performs the SQL sentence~~
  - ~~A DataReader object which captures the recovered entries~~

ATENTION: In the disconnected mode we always need to recover the data (SELECT) in order to locally work with it (INSERT, UPDATE, DELETE)

Definition of a Select Command

- For recovering the data we need:
  - ~~A SQL sentence which selects the desired information~~
  - ~~A Command object which performs the SQL sentence~~
  - ~~A DataReader object which captures the recovered entries~~

  - A **DataAdapter** object which performs the SQL sentence
  - A **DATASET** object in which storing the result of the SQL sentence

DataSet and DataAdapter Objects

- We create a virtual DB, with a DataSet object

```
DataSet bdvirtual = new DataSet();
```

- We fill it in with the tables that we want to work with:
  - Objeto DataAdapter
    - Método Fill()

...

```
SqlDataAdapter da = new
SqlDataAdapter("select * from Cliente", c);

 da.Fill(bdvirtual,"cliente");
```

Now we work locally

- Now in “bdvirtual” we have our local database.

For working locally

- We do it modifying rows and columns of the tables stored locally.
- In the dataset we hace stored the virtual db, we copy to a datatable the table to be modified.

```
DataTable t = new DataTable();
t = bdvirtual.Tables["cliente"];
```

Operations

- Obtaining a table:
  - DataTable t = new DataTable();
  - t = bdvirtual.Tables["cliente"];
- Accesing to the elements of the rows of that table (we can use a loop):
  - DataRow fila = t.Rows[0];
  - fila[0] = "Andrés";
  - **SAME AS**
  - t.Rows[0][0] = "Andrés";

First row, second column:

- t.Rows[0][1]
- Column information: (name, type)
  - t.Columns[0].ColumnName
  - t.Columns[0].DataType

We want to insert a new client

- This is equivalent to insert a new row in our local table…

```
DataRow nuevafila = t.NewRow();
nuevafila[0] = textBox1.Text;
nuevafila[1] = textBox2.Text;
nuevafila[2] = textBox3.Text;
t.Rows.Add(nuevafila);
```

Validating the changes

- DataAdapter object:
  - It has been used to fill in the table, we also use it to update the data in the real DB.
- Update method
  - The *DataAdapter* will analyze the changes done in the *DataSet* and will perform the proper commands to *insert, update or delete* in the real DB.

Command builder

- CommandBuilder Object:
  - Command builder
  - We pass to it as an argument the DataAdapter
  - It builds the needed SQL commands to act over the DB

```
• SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
• da.Update(bdvirtual, "cliente");
• label4.Text = "Changed";
```

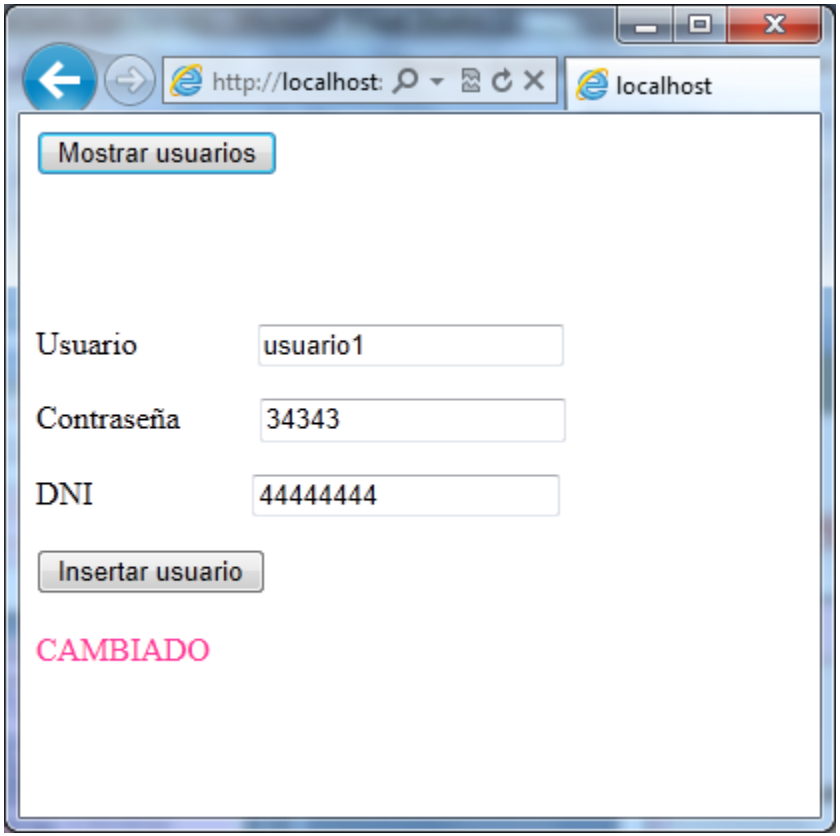
```
string s = "data source=.\\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|\\Database1.mdf;User
Instance=true";
SqlConnection c=new SqlConnection(s);
DataSet bdvirtual = new DataSet();
SqlDataAdapter da = new SqlDataAdapter("select * from Cliente", c);
da.Fill(bdvirtual, "cliente");

DataTable t = new DataTable();
t = bdvirtual.Tables["cliente"];
DataRow nuevafila = t.NewRow();
nuevafila[0] = textBox1.Text;
nuevafila[1] = textBox2.Text;
nuevafila[2] = textBox3.Text;
t.Rows.Add(nuevafila);

SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
da.Update(bdvirtual, "cliente");
label4.Text = "changed";
```

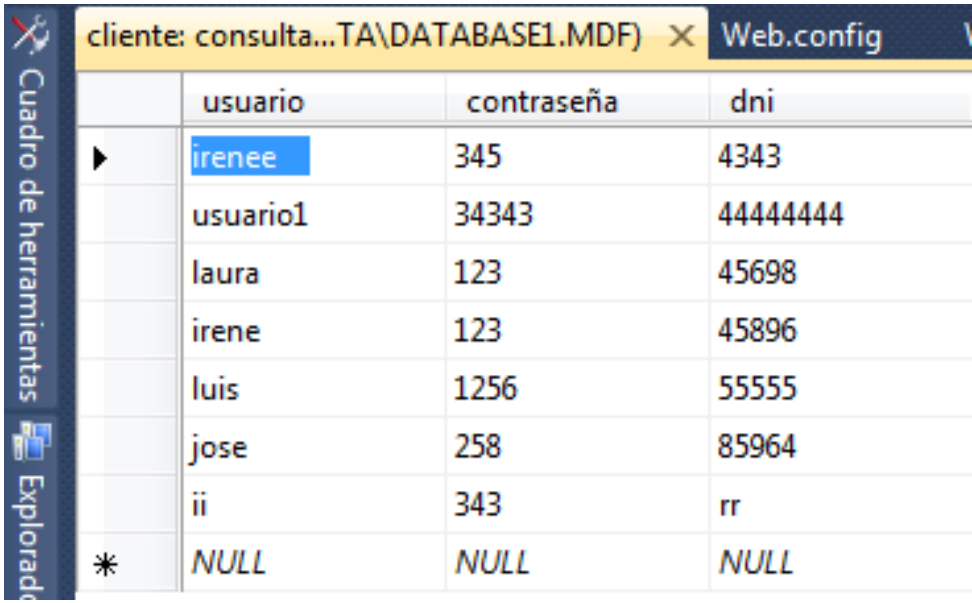
What is missing in this code?

In the 3 layers, we modify the CAD function

```
public bool InsertarCliente(ENCliente cli)     {
        bool cambiado;
        ENCliente cl = cli;
        DataSet bdvirtual = new DataSet() ;
        SqlConnection c = new SqlConnection(s);
    try     {
            SqlDataAdapter da = new SqlDataAdapter("select * from Cliente",c);
            da.Fill(bdvirtual,"cliente");
            DataTable t = new DataTable();
            t = bdvirtual.Tables["cliente"];
            DataRow nuevafila = t.NewRow();
            nuevafila[0] = cl.Usuario;
            nuevafila[1] = cl.Contraseña;
            nuevafila[2] = cl.Dni;
            t.Rows.Add(nuevafila);
            SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
            da.Update(bdvirtual, "cliente");
            cambiado = true;     }
        catch (Exception ex) {   cambiado = false;    }
        finally    {     c.Close();        }
        return cambiado;  }
```

Execution





usuario	contraseña	dni
irenee	345	4343
usuario1	34343	44444444
laura	123	45698
irene	123	45896
luis	1256	55555
jose	258	85964
ii	343	rr
NULL	NULL	NULL

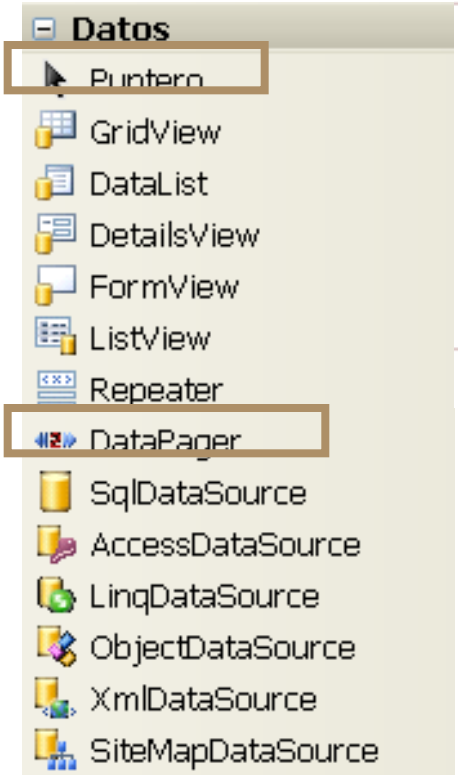
3 GridView Control

GridView Control

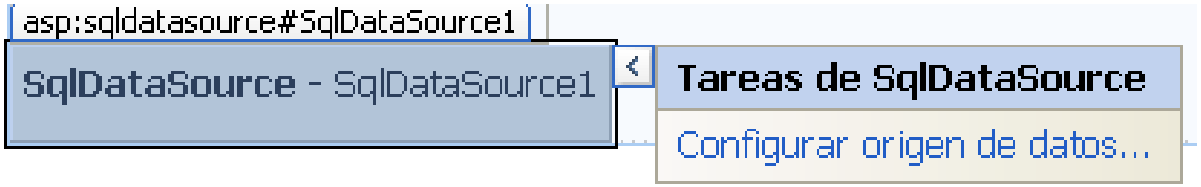
- Control for representing data in a table format (rows and columns)
- Properties:
  - Selection
  - Pagination
  - Sorting
  - Edition
  - Extensible by means of templates

- http://msdn.microsoft.com/es-es/library/cc295223(v=expression.40).aspx

GridView


Datos
Puntero
GridView
DataList
DetailsView
FormView
ListView
Repeater
DataPager
SqlDataSource
AccessDataSource
LinqDataSource
ObjectDataSource
XmlDataSource
SiteMapDataSource


asp:gridview#GridView1
Column0 Column1 Column2
abc abc abc
abc abc abc
abc abc abc
abc abc abc
abc abc abc
Tareas de GridView
Formato automático...
Elegir origen de datos:
(Ninguno)
Editar columnas...
Agregar nueva columna...
Editar plantillas


asp:sqldatasource#SqlDataSource1
SqlDataSource - SqlDataSource1
Tareas de SqlDataSource
Configurar origen de datos...

Goal:

We are showing the data of the client table in a gridview.

- *Using the wizard*
- *Writing code*

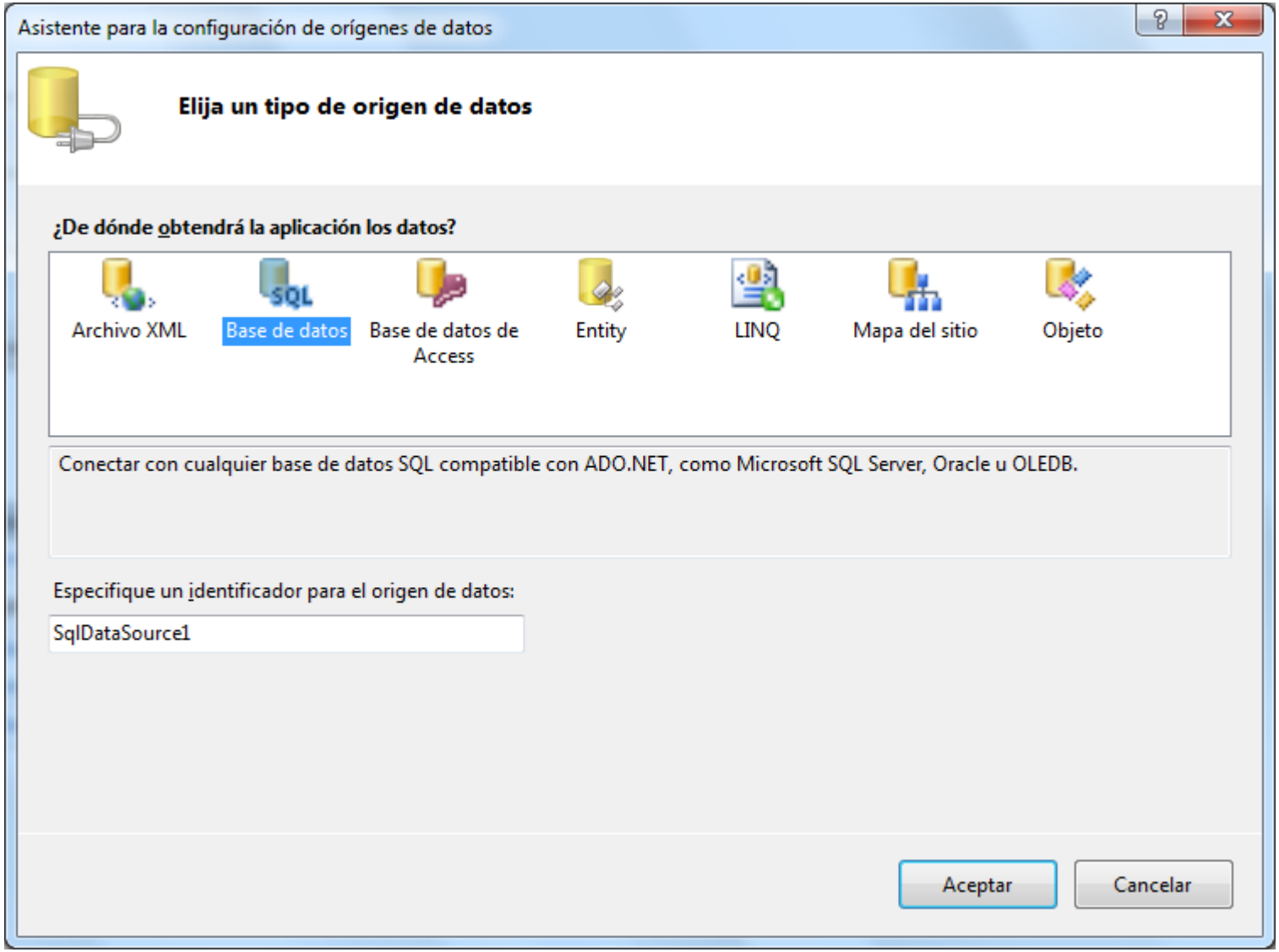
GridView → DataSource

- We add a GridView control and we select as a datasource the db.

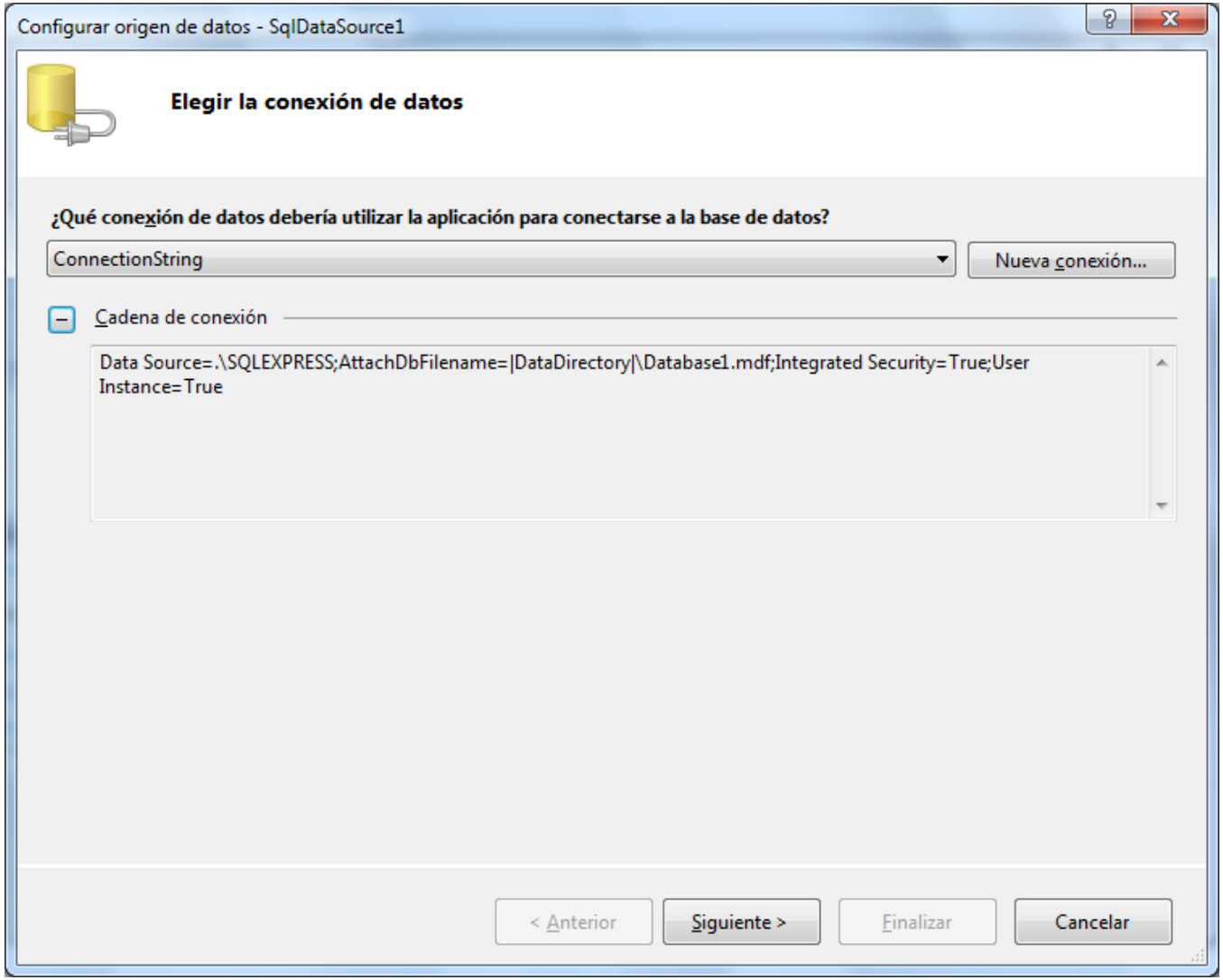
GridView control

- We can assign a table of the DB as datasource, or the entries obtained as a result of a SQL sentence.

Select DB

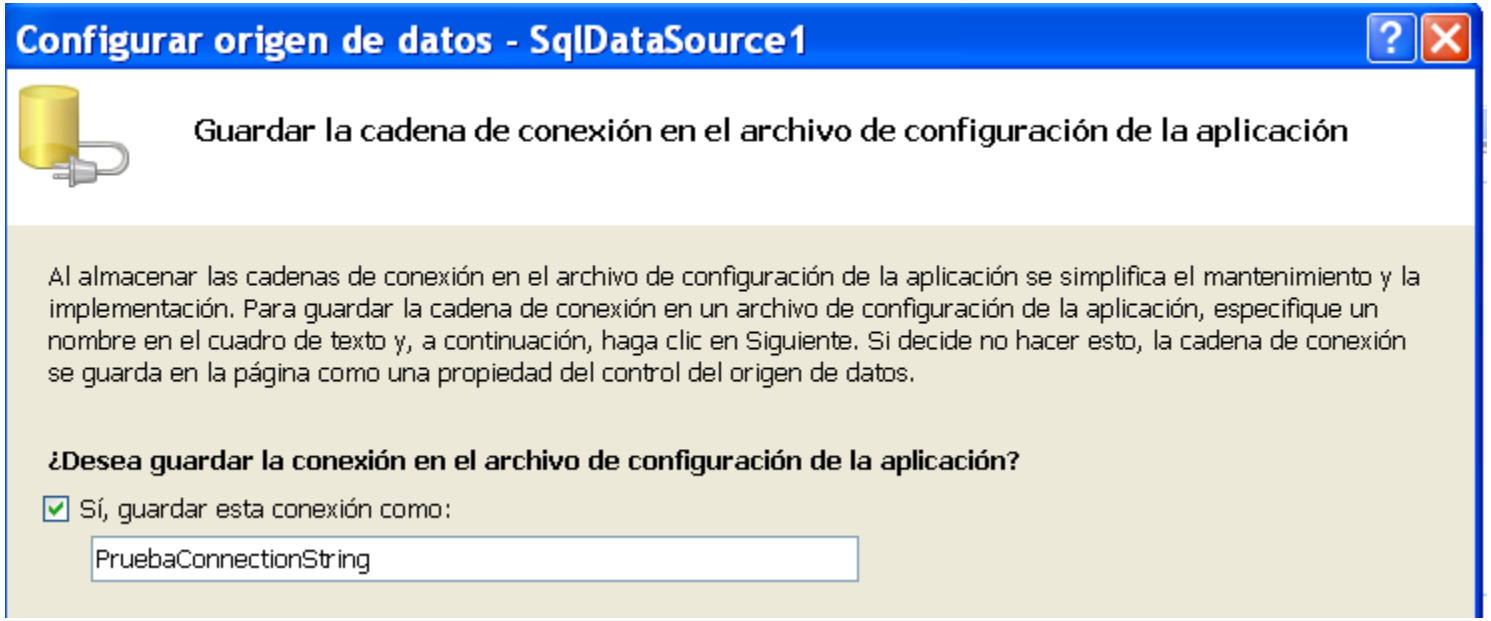

Asistente para la configuración de orígenes de datos
Elija un tipo de origen de datos
¿De dónde obtendrá la aplicación los datos?
Archivo XML
Base de datos
Base de datos de Access
Entity
LINQ
Mapa del sitio
Objeto
Conectar con cualquier base de datos SQL compatible con ADO.NET, como Microsoft SQL Server, Oracle u OLEDB.
Especifique un identificador para el origen de datos:
SqlDataSource1
Aceptar
Cancelar

Select data connection



GridView

- We store the connection string in the Web.config file



Configurar origen de datos - SqlDataSource1

Configurar la instrucción Select

¿Cómo desea recuperar los datos de la base de datos?

- ○ Especificar una instrucción SQL o un procedimiento almacenado personalizado
- ◉ Especificar columnas de una tabla o vista

Nombre:

cliente

Columnas:

- [x] *
- [ ] usuario
- [ ] contraseña
- [ ] dni

[ ] Devolver sólo filas únicas

WHERE...

ORDER BY...

Avanzadas...

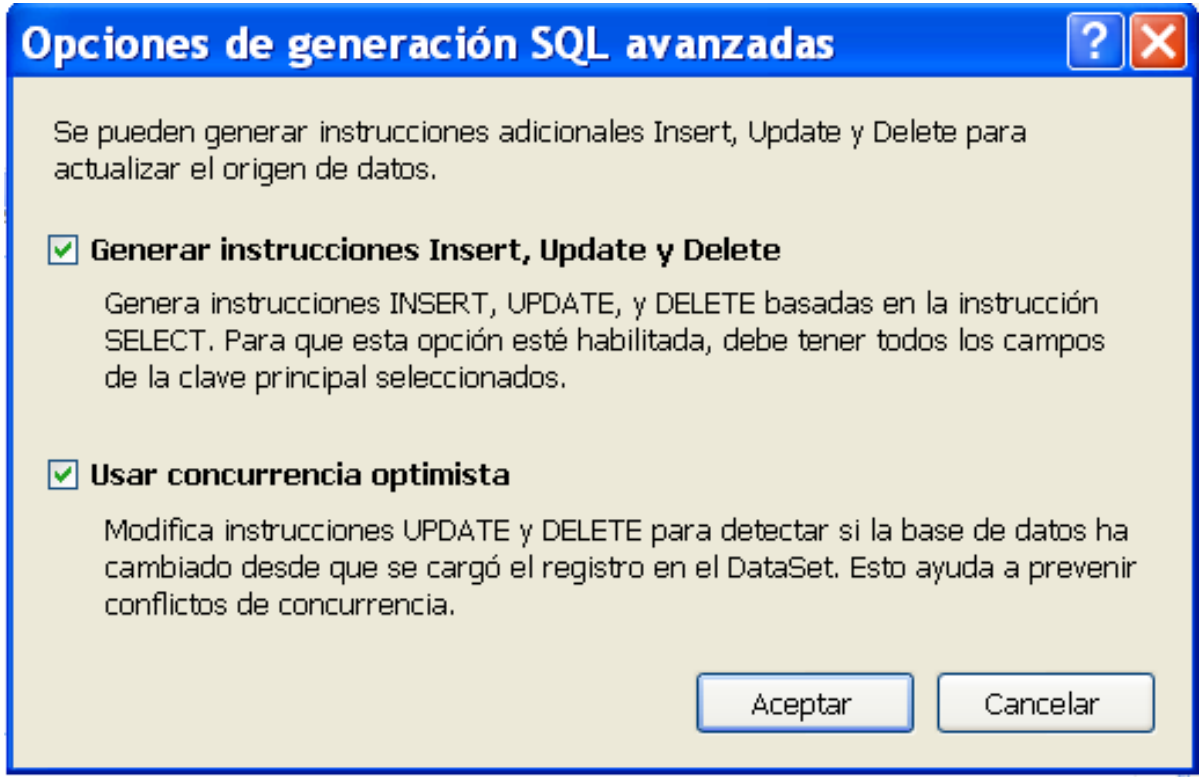
Instrucción SELECT:

```
SELECT * FROM [cliente]
```

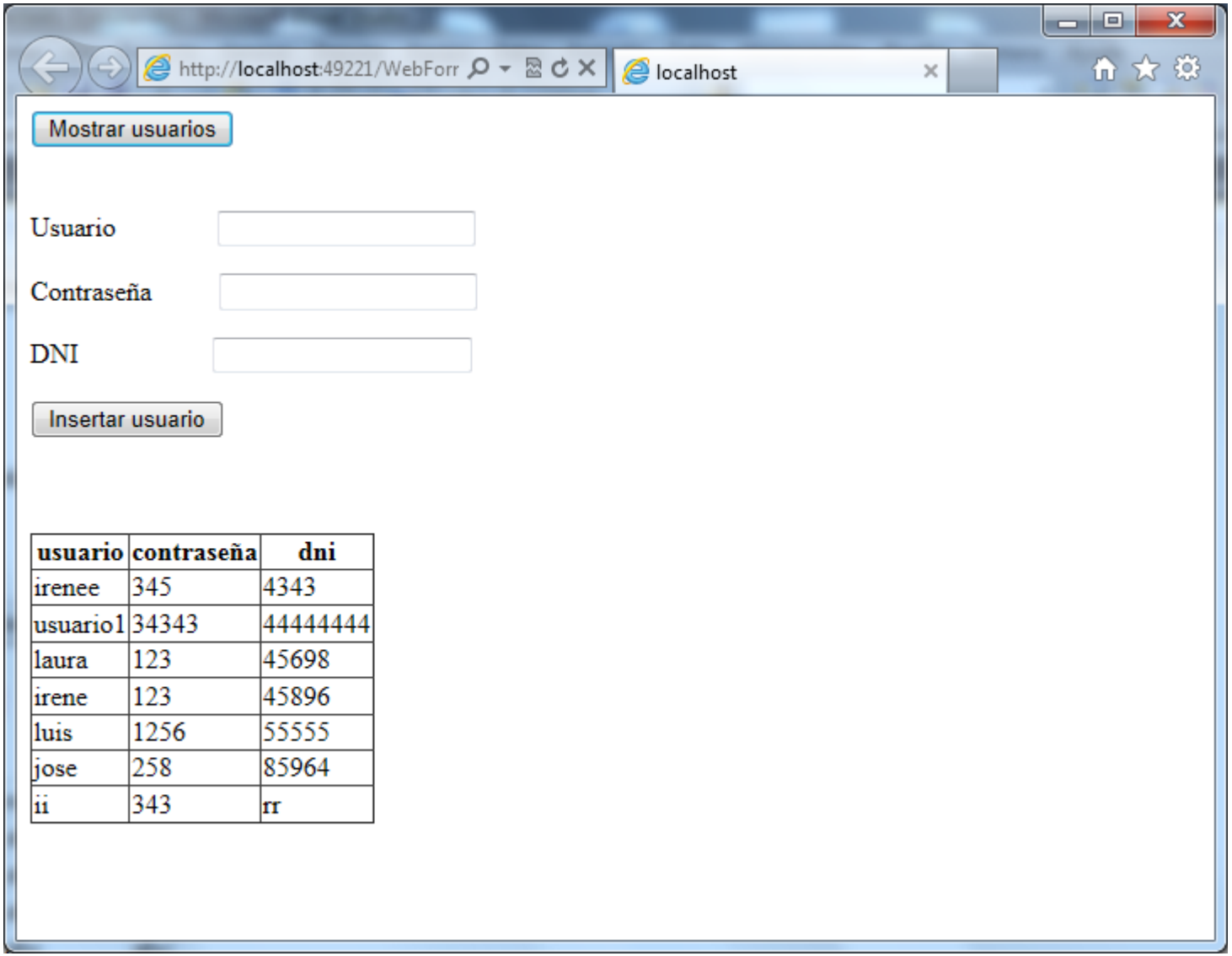
< Anterior | Siguiente > | Finalizar | Cancelar

GridView (wizard)

- Advanced
  - We can generate the Insert, Update and Delete instructions



Execution…



usuario	contraseña	dni
irenee	345	4343
usuario1	34343	44444444
laura	123	45698
irene	123	45896
luis	1256	55555
jose	258	85964
ii	343	rr

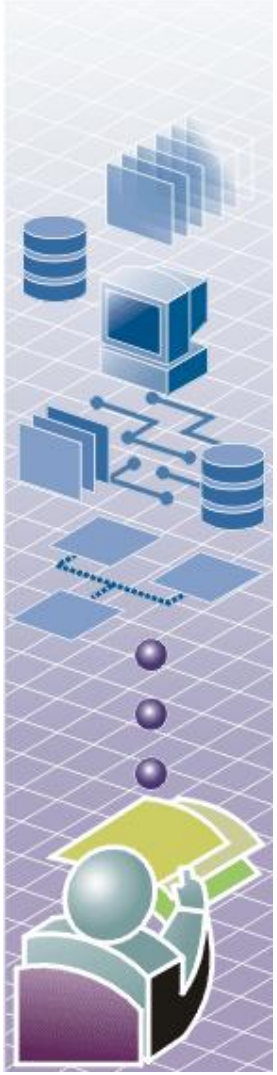
Code to show data from a dataset

```
ENCliente enl = new ENCliente();
DataSet d = new DataSet();

        protected void Page_Load(object sender, EventArgs e)
        {

            if (!Page.IsPostBack)
            {
                d = enl.listarClientesD();

                GridView1.DataSource = d;
                GridView1.DataBind();
            }

        }
```

Where in the CAD…

```
public DataSet ListarClientesD()
        {
            DataSet bdvirtual = new DataSet();

                SqlConnection c = new SqlConnection(s);
                SqlDataAdapter da = new SqlDataAdapter("select * from Cliente", c);
                da.Fill(bdvirtual, "cliente");
             return bdvirtual;
}
```

Exercise



Exercise

- Modify the previous example for editing the data of a client in the DB.

5 min

AutoGenerateSelectButton=true

Usuario

Contraseña

DNI

Insertar usuario Editar usuario

	usuario	contraseña	dni
Seleccionar	laura	1237	44444444
Seleccionar	laura2	1239	4569
Seleccionar	irene	123	45896
Seleccionar	luis	1256	55555
Seleccionar	jose	258	85964
1 2			

Event SelectedIndexChanged

```
protected void GridView2_SelectedIndexChanged(object sender,
 EventArgs e)
    {
        TextBox1.Text = GridView2.SelectedRow.Cells[1].Text;
        TextBox2.Text = GridView2.SelectedRow.Cells[2].Text;
        TextBox3.Text = GridView2.SelectedRow.Cells[3].Text;
        TextBox3.Enabled = false;
    }
```

Edit button (click)

```
{

 ENCliente en = new ENCliente();
 en.Usuario = TextBox1.Text;
 en.Contraseña = TextBox2.Text;
  d = en.ModificarCliente(GridView2.SelectedIndex);

 GridView2.DataSource = d;
 GridView2.DataBind();
}
```

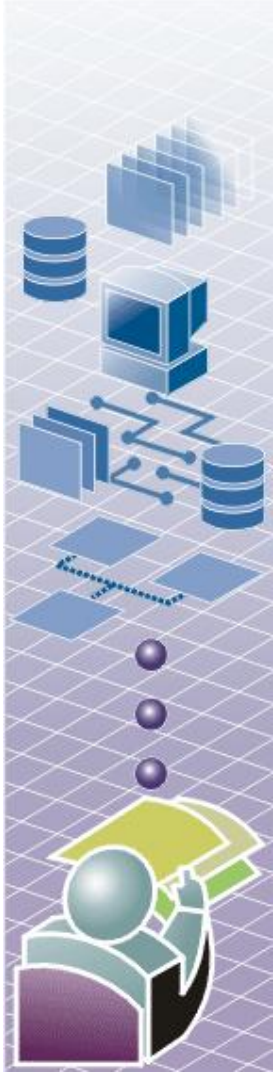
EN

```
public DataSet ModificarCliente(int i)
{
    CADcliente c = new CADcliente();
    DataSet a = c.ModificarCliente(this,i);
    return a;
}
```

CAD (simplified)

```
public DataSet ModificarCliente(ENCliente cli, int i)
  {

      ENCliente cl = cli;
      DataSet bdvirtual = new DataSet();
      SqlConnection c = new SqlConnection(s); SqlDataAdapter da = new
      SqlDataAdapter("select * from Cliente", c);
         da.Fill(bdvirtual, "cliente");
         DataTable t = new DataTable();
         t = bdvirtual.Tables["cliente"];


         t.Rows[i]["usuario"]=cl.Usuario;
         t.Rows[i]["contraseña"] = cl.Contraseña;

         SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
         da.Update(bdvirtual, "cliente");

          return bdvirtual;
}
```

Exercise



Exercise

- Modify the previous example for deleting the data of a client in the DB.

5 min

AutoGenerateDeleteButton=true

```
protected void GridView2_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        ENCliente en = new ENCliente();

        d = en.BorrarCliente(e.RowIndex);



        GridView2.DataSource = d;
        GridView2.DataBind();

    }
```

EN

```
public DataSet BorrarCliente(int i)
    {
        CADcliente c = new CADcliente();
        DataSet a = c.BorrarCliente(this, i);
        return a;
    }
```

CAD

```
public DataSet BorrarCliente(ENCliente cli, int i)
    {
        ENCliente cl = cli;
        DataSet bdvirtual = new DataSet();
        SqlConnection c = new SqlConnection(s);

            SqlDataAdapter da = new SqlDataAdapter("select * from Cliente", c);
            da.Fill(bdvirtual, "cliente");
            DataTable t = new DataTable();
            t = bdvirtual.Tables["cliente"];

            t.Rows[i].Delete();

            SqlCommandBuilder cbuilder = new SqlCommandBuilder(da);
            da.Update(bdvirtual, "cliente");

        return bdvirtual;       }
```

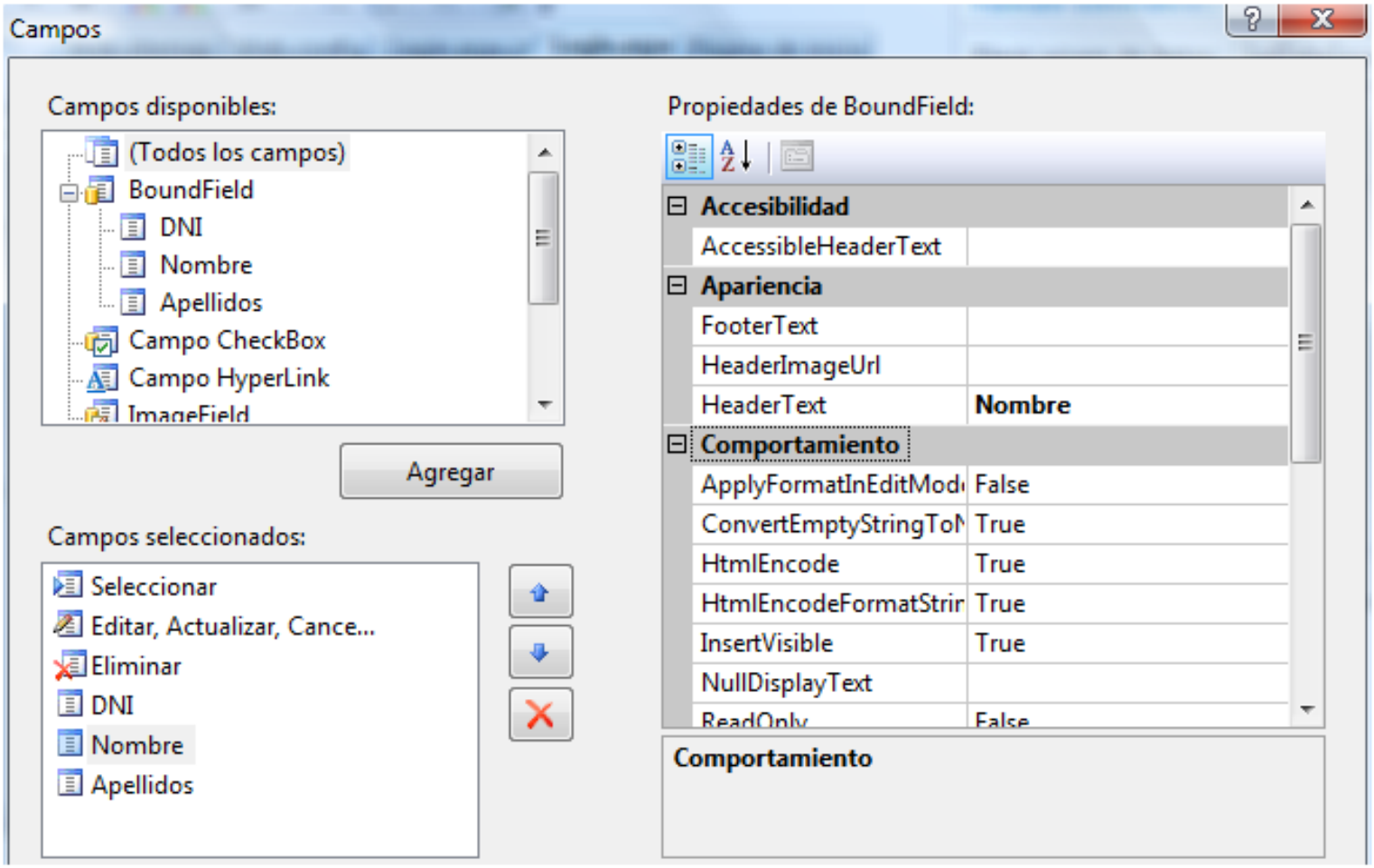
Pagination in GridView

- Properties
  - AllowPaging = true
  - PageSize = 5 (number of elements in a page)

- When showing the data using the wizard we do not need code for pagination, however when we bind the data ourselves we need to write some code for the event

```
protected void GridView2_PageIndexChanging(object sender,
  GridViewPageEventArgs e)
      {
         d = enl.listarClientesD();
         GridView2.PageIndex = e.NewPageIndex;
          GridView2.DataSource = d;
          GridView2.DataBind();
      }
```

GridView: edit columns

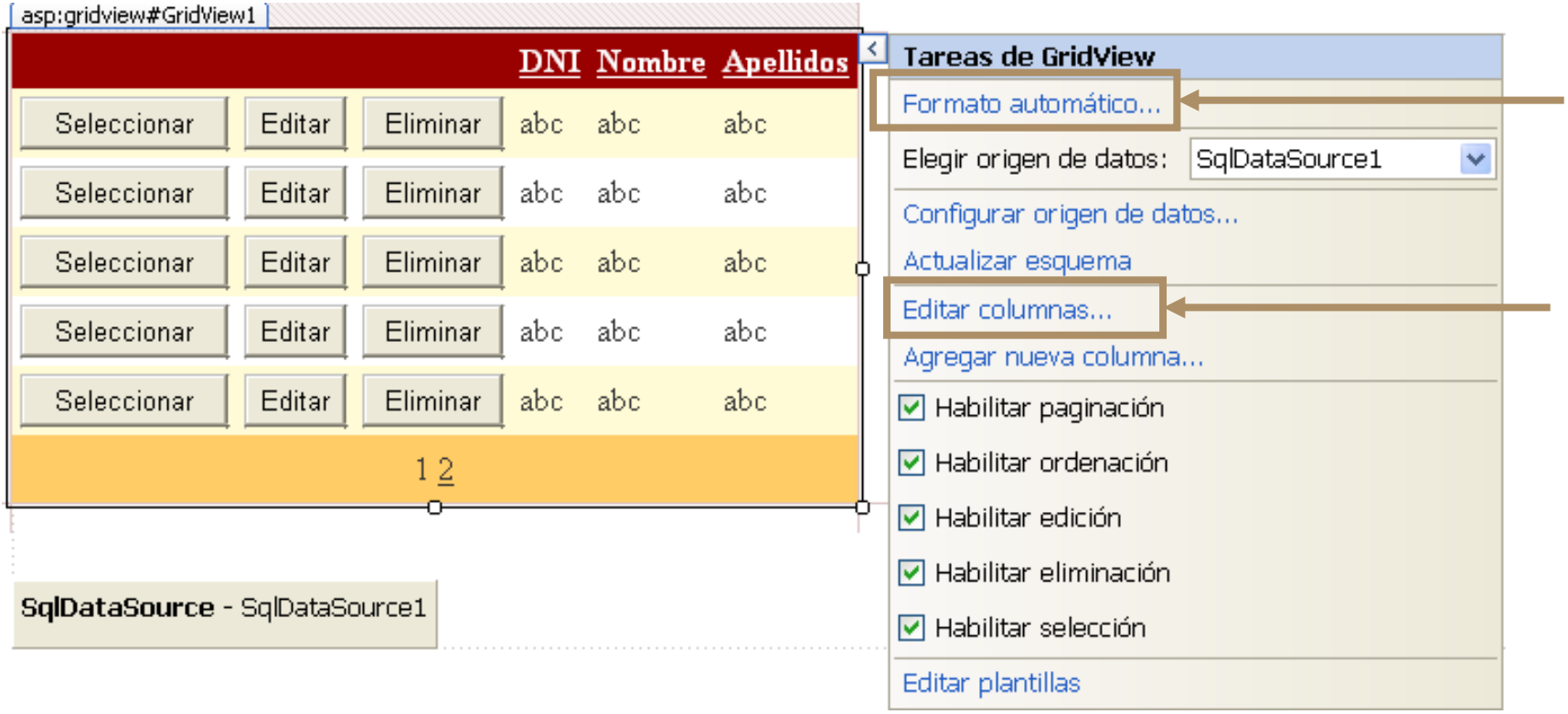


GridView

- Types of columns:
  - BoundField: Displays the text of a field of the DB
  - ButtonField: Displays a button for each item
  - CheckBoxField: Displays a checkbox for each item
  - CommandField: Provides selection, edition and deletion functions
  - HyperLinkField: Displays the text of a field of the DB as an hyperlink
  - ImageField: Displays an image
  - TemplateField: Allows specifying multiple fields and personalized controls

GridView

- We can modify the look and feel of the GridView



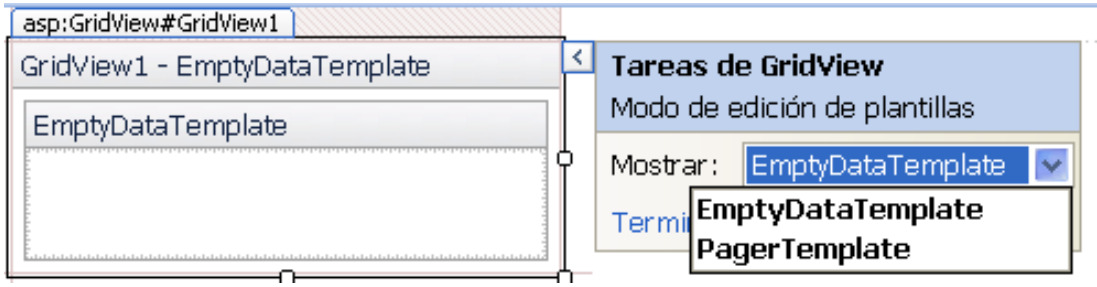
GridView

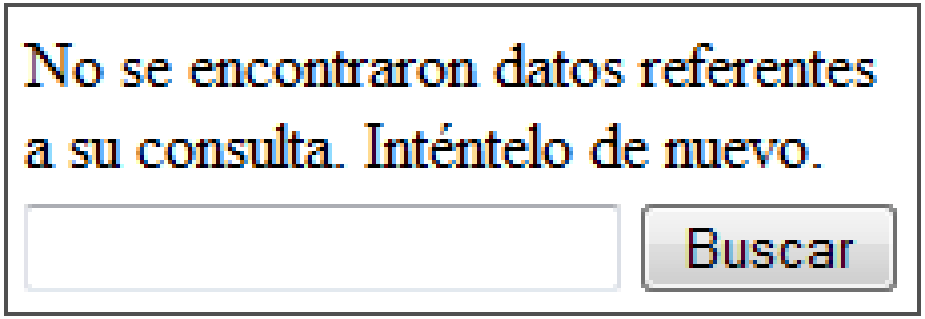
- Final result

				DNI	Nombre	Apellidos
	Actualizar	Cancelar		11111111	Laura	Sanchez
Seleccionar	Editar		Eliminar	22222222	Alberto	Lopez
Seleccionar	Editar		Eliminar	44444444	Juan	Perez
Seleccionar	Editar		Eliminar	55555555	Sara	Jover
Seleccionar	Editar		Eliminar	66666666	Berta	Belda
				1 2		

Edit templates

- **EmptyDataText**
  - It is used for showing a message when there is no data to be displayed in the GridView
- **EmptyDataTemplate**
  - We can personalize the message shown when the GridView is empty.





4

Concurrence

Disconnected environment: conflicts

- In a disconnected environment, several users can modify the data in the same files at the same time.
- **Ways of managing the conflict:**
  - Pessimistic.
  - Positive concurrence.
  - Last Win.
  - Write the code for managing the conflict.

Concurrence

- Pessimistic concurrence: When a row is read, this is blocked for reading for any other user, until the user that owns it, releases it.

Concurrence (I)

- Positive concurrence: Rows are always available for being read, different users can read the rows at the same time.

- When someone tries to modify a row that has been already modified, we get an error and it is not modified.

Concurrence (II)

- “Last win”: this technique implies that it does not exist any control. The last change done is the one that stays.

ADO.NET: Positive concurrence

- The DataSet object maintains two versions of the rows we read:
  - Original version, same as we read in the DB
  - Updated version, which represents the changes done by the user
- When the row is updated, the original values are compared with the real row in the DB, to check if it has been modified.
  - If it has been modified, we have to capture an exception
  - Otherwise, the update is done

RowUpdated Event

- We can write application code that allows the users to determine which changes should be kept. The specific solutions can vary depending on the bussiness requirements of a certain application.

- RowUpdated Event:
  - When updating a row: after every operation but before triggering any exception
  - We can examine the results and avoid that an exception is triggered

5

Connected vs Disconnected

Connected vs Disconnected

- Connected access to data (live connection)
  - DataReader
  - We can rapidly recover all the results.
  - We use a live connection. Lighter and faster than DataSet
  - We can only access the result in a read-only forward-only way.
  - Better performance than DataSet, so it is a better choice for simple data access.

- Disconnected access to data
  - DataSet

Connected vs Disconnected

- For read-only consults that you just have to do them once (we will not have to go to previous rows) the recommendad object is **DataReader**.
- *For example, to check if a product is in a table that stores the list of items of the stock list of a store, we just need a unique read-only query.*

- However, if we want to do a more complex access to data, such as a query for all the products of different type that belongs to a provider, the correct choice would be using **DataSet.**

Connected vs Disconnected

- **Acces to data.**
  - As we have said, if we want to get and store data, we use DataSet, since DataReader only allows reading the data.
- **Working with more than one table or database.**
  - If the function we are developing requires information located in several tables of the same database or several DB, we will use the *DataSet object*. With *DataReader* we can only build SQL queries that access to one DB.