Unidad 4

Acceso a Datos

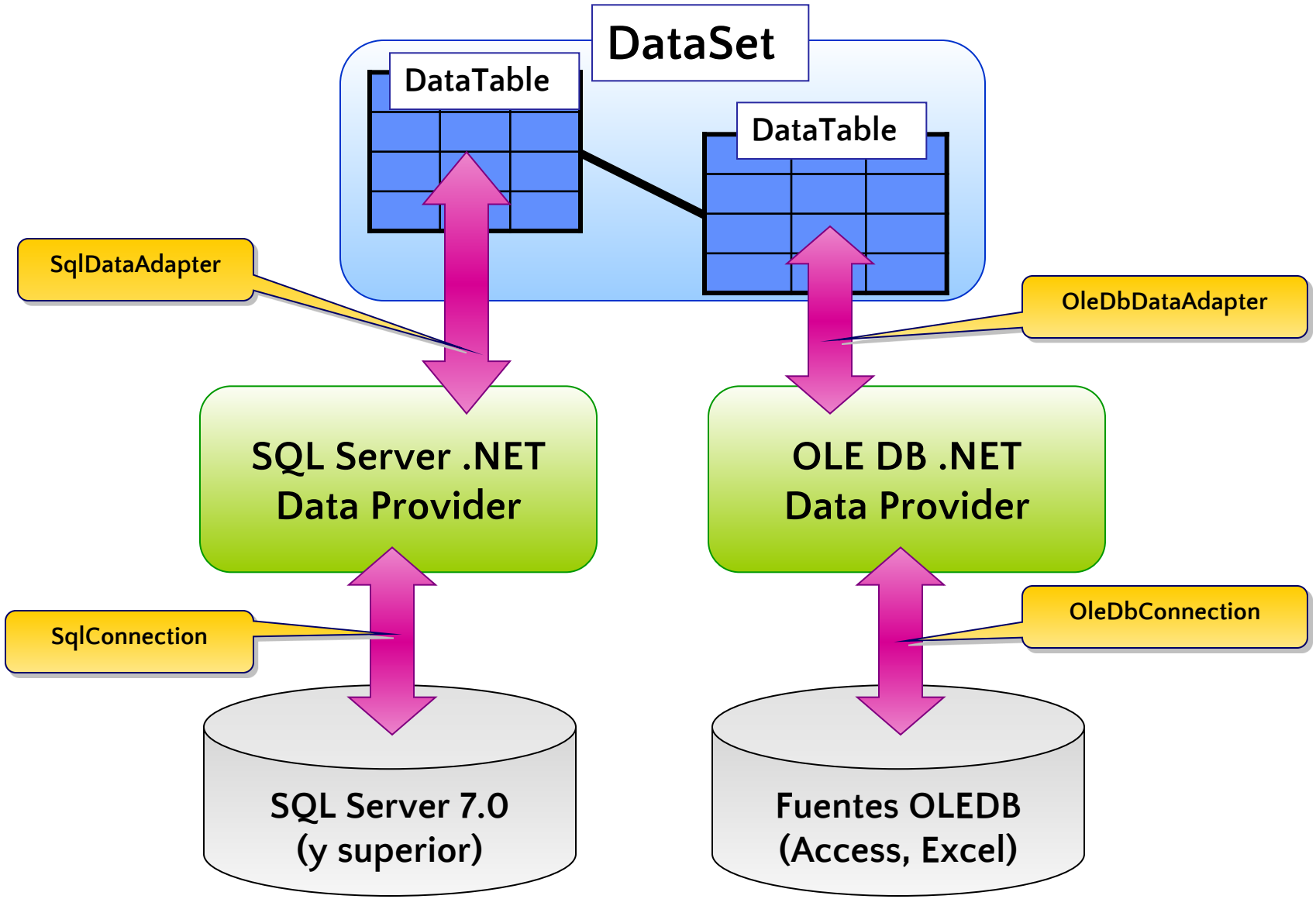
Contenidos

- Que es ADO.NET?
- Namespaces
- Connection
- Commands
- DataReader
- DataSet, DataTable y DataAdapter
- DataSet Tipados (Typed DataSet)
- ASP.NET Data Binding
- Data Source Components

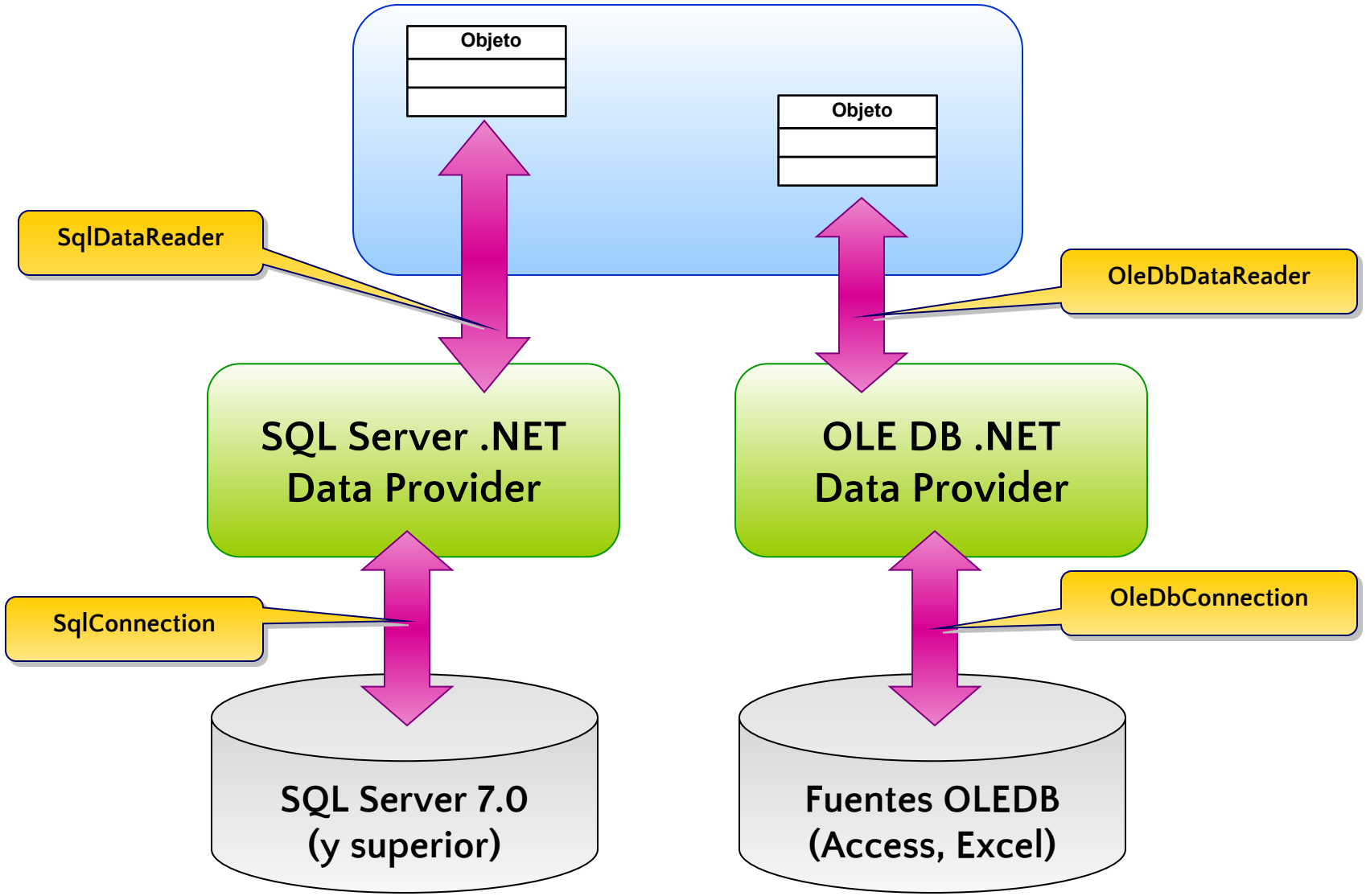
Que es ADO.NET?

- ADO.NET es un conjunto de clases para conectar y manipular fuentes de datos
- Características
  - Diseñado para trabajar en entornos desconectados
  - Modelo de programación con soporte para XML
  - Brinda un conjunto de clases, estructuras, interfaces y enumeraciones para la administración del acceso a datos
  - Basado en Data Providers
  - Diseñado para trabajar con cualquier origen de datos (no solo base de datos relacionales)

Modelo de objetos de ADO.NET (usando DataSets)



Modelo de objetos de ADO.NET (usando Objetos)



ADO.NET - Namespaces

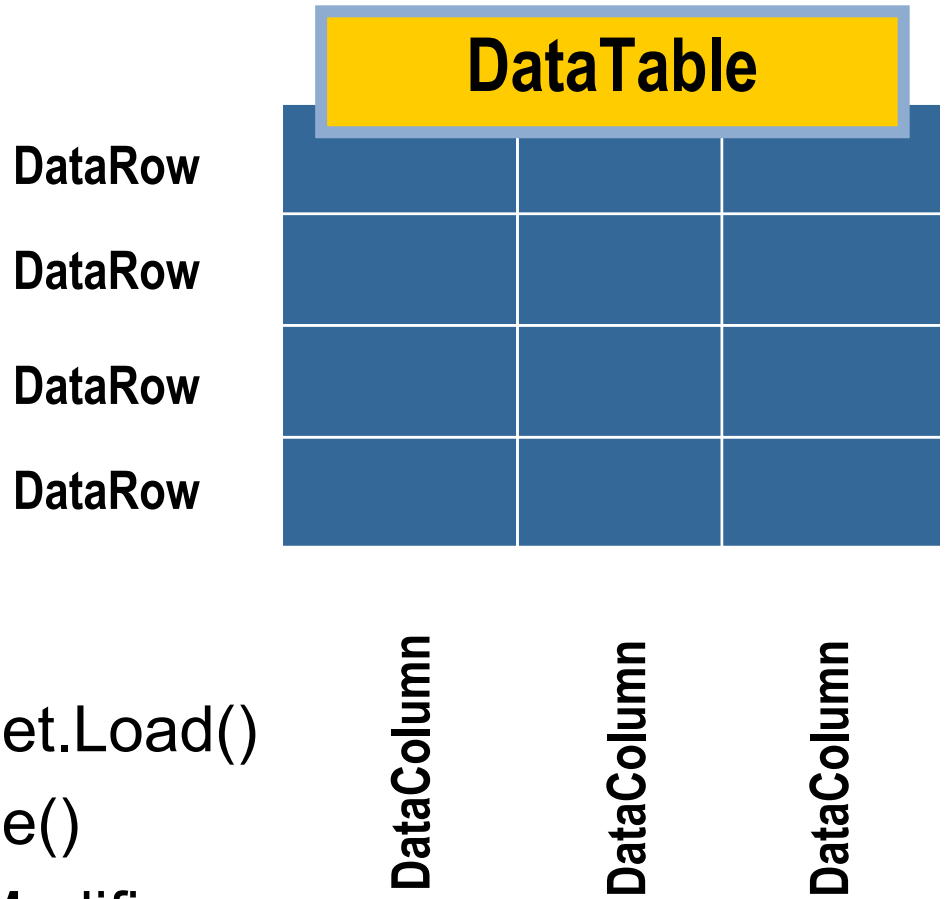
- Use la instrucción using para importar namespaces:
  - System.Data
  - System.Data.Common
- Depende del origen de datos a usar (Data Provider):
  - System.Data.SqlClient
  - System.Data.OleDB
  - System.Data.ODBC
  - System.Data.OracleClient

DataTable

- Es el objeto central de la biblioteca ADO.NET para manejar datos (sino se usan objetos)
- El esquema esta definido por DataColumnCollection.
- Mantiene la integridad de los datos por medio de Constraints
- Por medio de sus eventos podemos controlar los diferentes estados de los registros.
- Desconoce su origen de datos, por lo que funciona como una entidad independiente

DataTable

- System.Data.DataTable
- System.Data.DataRow
- System.Data.DataColumn



- Es iXMLSerializable
- Método DataTable/DataSet.Load()
- Método DataView.ToTable()
- RowState.SetAdded/SetModifie

DataTable - Miembros

.NewRow	Devuelve un objeto DataRow vacio con el esquema del DataTable
.ReadXMLSchema	Establece el Esquema del DataTable en base al contenido de un archivo XML
.ReadXML	Carga el contenido del DataTable en base a un archivo XML o Objetos Stream, Objets, etc
.Rows	Colección de Rows contenidos dentro del DataTable
.Select	Método del cual podemos por medio de expresiones realizar consultas sobre los datarows cargados.
.WriteXML	Escribe un archivo .xml con el contenido del DataTable

DataTable - Miembros

.WriteXMLSchema	Escribe en un archivo .xml con el esquema utilizado en el DataTable
.Columns	Colección de objetos DataColumn
.Columns.Add	Insertamos un Objeto DataColumn o bien indicamos el nombre y el tipo
.Columns.Remove	Eliminamos un objeto DataColumn del DataTable
.Load	Carga de datarows en base a un origen especificado (DataReader,…)
.Merge	Combina los rows entre múltiples DataTables

DataTable – Ejemplo C#

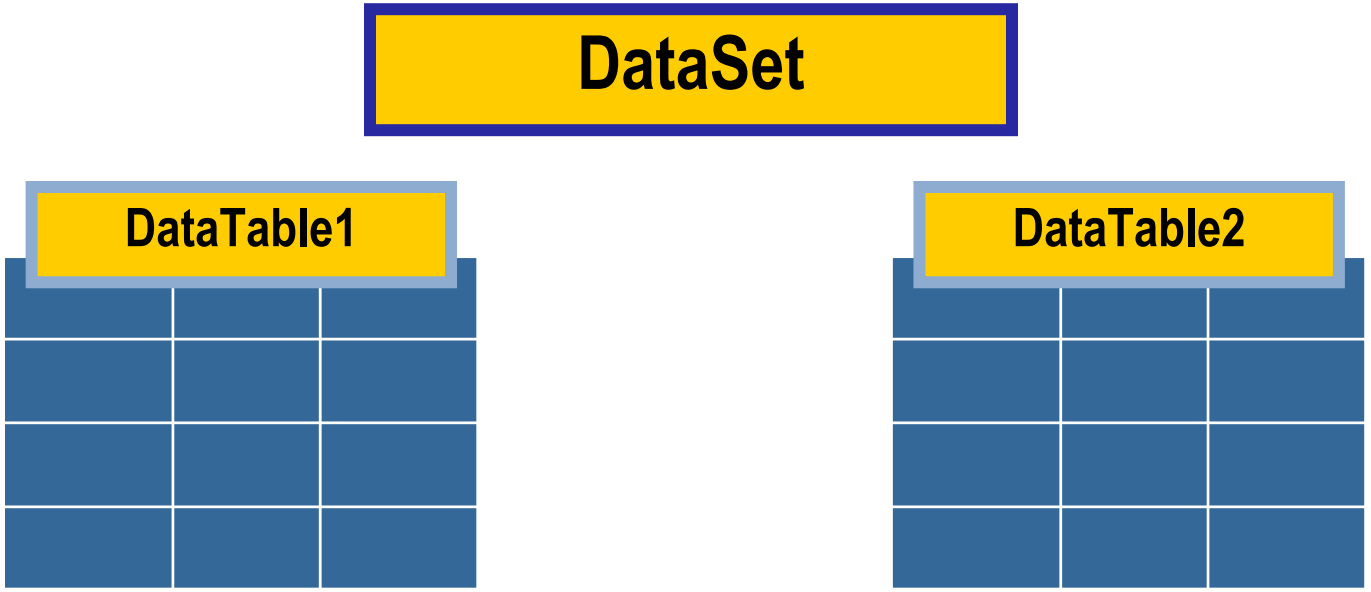
```
//Declaro el objeto DataTable y lo instancio
System.Data.DataTable tblTable = new System.Data.DataTable("MiTabla");

//Declaro un objeto DataColumn, le especicifico el nombre y el tipo de
//datos que almacenerá
System.Data.DataColumn colNombre =
    new System.Data.DataColumn("Nombre", typeof(System.Data.SqlTypes.SqlString));

//Adjunto a mi objeto Tabla el objeto Column que eh creao
tblTable.Columns.Add(colNombre);

//Declaro un objeto DataRow y le asigno un valor al campo Nombre
System.Data.DataRow rowData = tblTable.NewRow();
rowData["Nombre"] = "Jaime";

//Otra manera de asignar valores es indicando el indice del campo
System.Data.DataRow rowData2 = tblTable.NewRow();
rowData2[0] = "Christian";

//Adjunto los 2 Rows creados a la tabla
tblTable.Rows.Add(rowData2);
tblTable.Rows.Add(rowData);
```

DataTable – Ejemplo VB.net

```
'Declaro el objeto DataTable y lo Instancio
Dim tblTable As New DataTable("MiTabla")

'Declaro un objeto DataColumn, le especifico el nombre y el tipo de
'datos que almacenará
Dim colNombre As New DataColumn("Nombre", Type.GetType("string"))

'Adjunto a mi objeto Tabla el objeto Column que eh creado
tblTable.Columns.Add(colNombre)

'Declaro un objeto DataRow y le asigno un valor al campo Nombre
Dim rowData = tblTable.NewRow()
rowData("Nombre") = "Jaime"

'Otra manera de asignar valores es indicando el indice del campo
Dim rowData2 = tblTable.NewRow()
rowData2(0) = "Christian"

'Adjunto los 2 DataRows al DataTable
tblTable.Rows.Add(rowData)
tblTable.Rows.Add(rowData2)
```

DataSet

- Representación de datos en memoria
- Consiste en una Colección de objetos DataTables
- Mantiene la integridad entre los DataTables por medio del objeto DataRelation
- Desconoce el origen de los datos



DataSet

- DataSet/DataTable.Load
- Cargar un DataTable/DataSet desde un DataReader
  - Cargue desde dbDataReaders
    - OLEDbDataReader
    - SqlDataReader
    - DataTableReader*
    - Etc.
- Permite un control más especifico de los datos

DataSet – Ejemplo C#

```
//Creo 2 tablas
DataTable tblAlumnos = new DataTable("Alumnos");
tblAlumnos.Columns.Add("IDAlumno", typeof(int));
tblAlumnos.Columns.Add("Nombre", typeof(string));

DataTable tblCursos = new DataTable("Alumno_Cursos");
tblCursos.Columns.Add("IDAlumno", typeof(int));
tblCursos.Columns.Add("Curso", typeof(string));

//Creo un DataSet que contendrá las dos tablas
DataSet dsUniversidad = new DataSet("UniversidadAcme");
dsUniversidad.Tables.Add(tblAlumnos);
dsUniversidad.Tables.Add(tblCursos);

//Creo la relacion entre las tablas
DataRelation drRelacion =
    new DataRelation("AlumnCourse",
                     tblAlumnos.Columns["IDAlumno"],
                     tblCursos.Columns["IDAlumno"]);
dsUniversidad.Relations.Add(drRelacion);
```

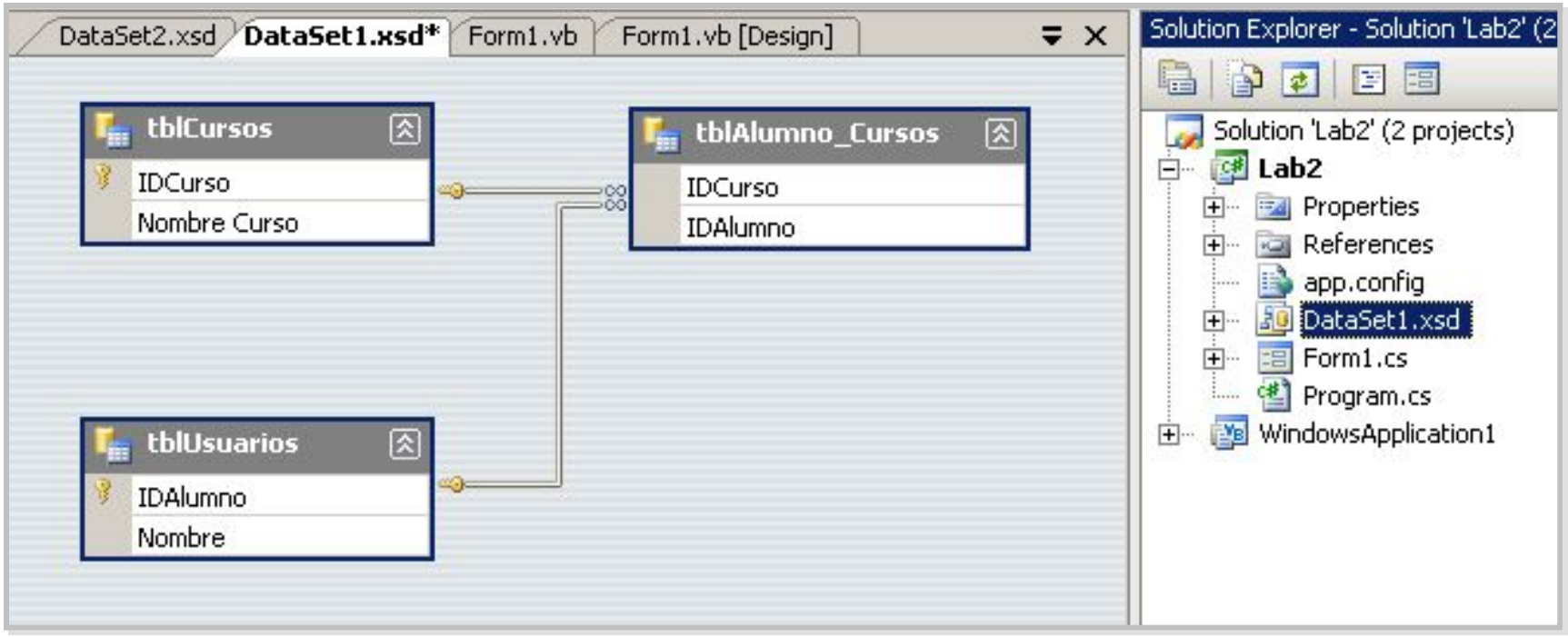
DataSet – Ejemplo

```
/*Por medio del método Select me traigo una colección de
 * DataRows del cual solo utilizo uno solo (precisamente porque
 * se que IDAlumno es un valor único)
*/
DataRow rowAlumno = tblAlumnos.Select("IDAlumno = 1")[0];
Console.Write(rowAlumno["Nombre"]);

/*
 * Me traigo la lista de cursos del alumno seleccionado
 * y recorro la colección de objetos DataRows obtenidos para
 * representarlos en pantalla mostrando el curso al cual se
 * encuentra
 */
DataRow[] rowCursos = rowAlumno.GetChildRows("AlumnoCursos");
foreach (DataRow rowCurso in rowCursos)
{
    Console.Write(rowCurso["Curso"]);
}
```

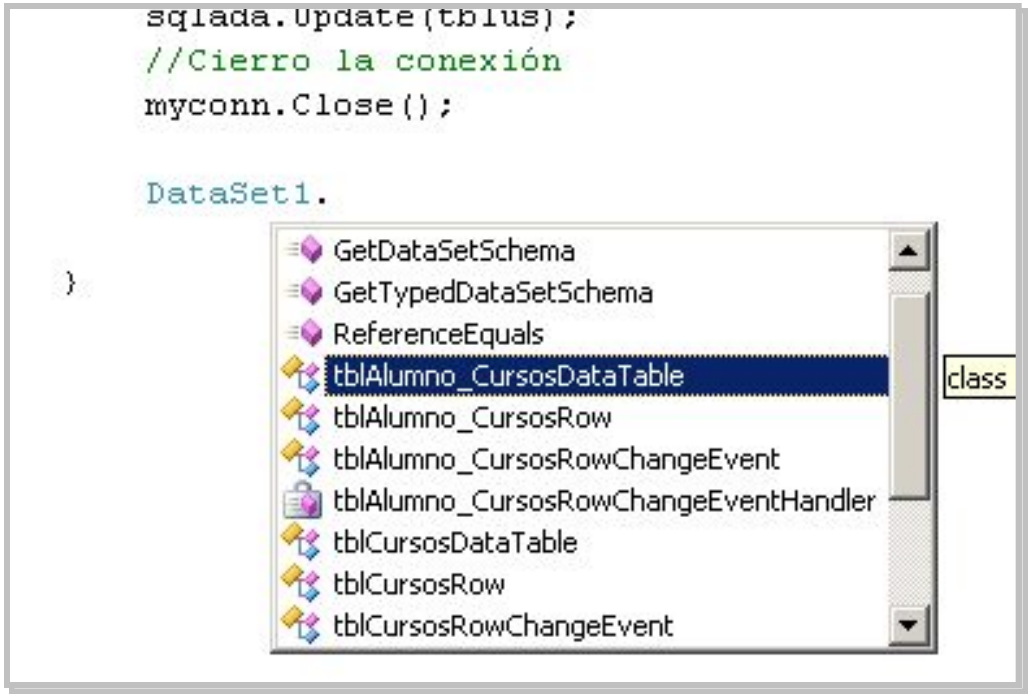
DataSet Tipado

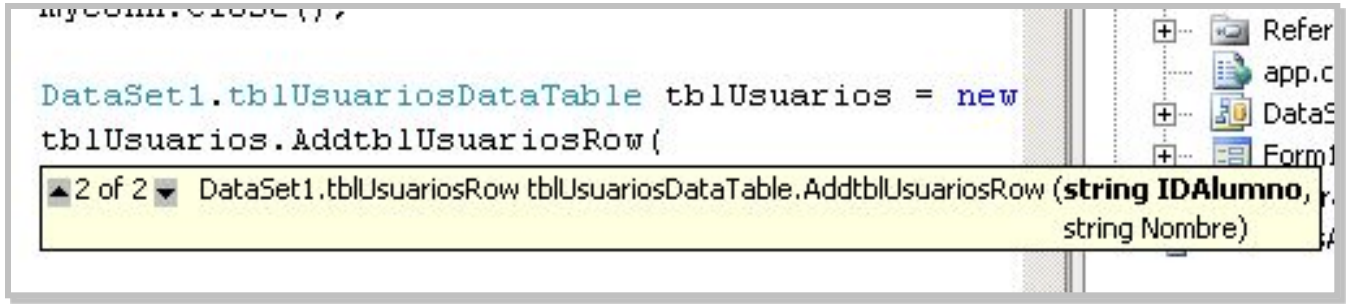
- Clase generada en Tiempo de Diseño
  - Hereda del DataSet
  - Schema codificado en la clase
- Beneficios
  - Completado de sentencias
  - Comprobación de tipos en compilación
  - Código conciso y legible

DataSet Tipado – Ejemplo (Diseñador)



DataSet Tipado – Ejemplo





Objeto Connection

- Permite establecer la comunicación fisica entre la aplicación y la base de datos
- Representa una conexión al Data Source
- Una conexión permite:
  - Personalizar la conexión a la base de datos
  - Begin, commit, y abortar transacciones

Objeto Connection - Providers

- System.Data.SqlClient.SqlConnection
- System.Data.ODBC.ODBCConnection
- System.Data.OleDB.OLEDBConnection
- System.Data.OracleClient.OracleConnection

```
string sCnn = "data source=localhost; " +
  "initial catalog=northwind; integrated security=true";
SqlConnection conn = new SqlConnection(sCnn);
```

Objeto Connection – Propiedades y Métodos

- Clases xxxConnection heredan de System.Data.Common.DbConnection

- Propiedades:
  - ConnectionString: Cadena de conexión

- Métodos:
  - Open: Abre la conexión con el origen especificado
  - Close: Cierra la conexión
  - BeginTransaction: Inicia una transacción con el origen

Objeto Connection - ConnectionString

- Parámetro critico en una aplicación que puede contener datos “sensibles” y requerir algún tipo de protección
- Especificado con entrada en el Web.config:

```
<connectionStrings>
    <add name=“AcademiaSQL“ connectionString="SERVER=(local);
DATABASE=Academia;UID=net; Password=net" />
    <add name=“AcademiaAccess“ connectionString="~\Academia.mdb" />
</connectionStrings>
```

- Accedido por código:

```
System.Configuration.ConfigurationManager.Con
nectionStrings[“NWind”].ConnectionString;
```

Objeto Connection - ConnectionString

- Parámetros
  - Data Source
  - Initial Catalog
  - Integrated Security
  - Connection Timeout
  - User ID
  - Password
  - Provider (usado para OleDbConnection)
- www.connectionstrings.com

Objeto Connection - Ejemplo

```
System.Data.SqlClient.SqlConnection myconn =
    new System.Data.SqlClient.SqlConnection();
myconn.ConnectionString =
    "Data Source=MiPC;Initial Catalog=Northwind;User ID=sa;Password=123";
myconn.Open();
//
//Realizo las operaciones necesarias
//
myconn.Close();
//Por último me DESCONECTO
```

```
Dim myconn As New System.Data.SqlClient.SqlConnection()
myconn.ConnectionString = _
    "Data Source=MiPC;Initial Catalog=Northwind;User ID=sa;Password=123"
myconn.Open()
'
'Realizo las operaciones necesarias
'
myconn.Close()
'Por último me DESCONECTO
```

Objeto Command

- Representa una Instrucción SQL o un procedimiento almacenado que se ejecuta en un origen de datos
- Requiere de un objeto Connection
- Soporta Parámetros
- Especifico para cada proveedor (XxxCommand):
  - SQLCommand
  - ODBCCommand
  - OLEDBCommand
  - OracleCommand
- Puede ser usado de forma individual o a través de DataAdapters
- Expone 4 métodos importantes para devolver datos (próximo slide)

Objeto Command – Propiedades y Metodos

- Propiedades (a configurar):
  - CommandType
  - CommandText
  - Connection
  - Parameters
- Métodos (a ejecutar):
  - ExecuteReader()
  - ExecuteScalar()
  - ExecuteNonQuery()
  - ExecuteXMLReader()

Objeto Command – Parámetro de Entrada (INPUT)

- Crear parámetro
- Definir dirección y tipo de dato
- Asignar valor
- Agregar a la colección Parameters

```
SqlParameter param =
            new SqlParameter("@FechaVto",
SQLDbType.DateTime);
param.Direction = ParameterDirection.Input;
param.Value = (DateTime)(txtFechaVto.Text);
cmd.Parameters.Add(param);
```

Objeto Command – Parámetro de Salida (OUTPUT)

- Crear parámetro
- Definir dirección
- Agregar a la colección Parameters

```
SqlParameter param =
                    new SqlParameter("@Cantidad",
SQLDbType.Int);
param.Direction = ParameterDirection.Output;
cmd.Parameters.Add(param);
```

- Llamar al procedimiento almacenado

```
cmd.ExecuteNonQuery();
```

- Leer el parámetro de salida

```
int cantComprada = (int)cmd.Parameters["@Cantidad"].Value;
```

Objeto Command – Ejemplo I SQLCommand

```
//Creo un objeto SQLCommand
System.Data.SqlClient.SqlCommand myComand = new SqlCommand();
//Indico la conección a utilizar
myComand.Connection = myconn;
//Escribo la consulta T-SQL que me devolverá la cantidad de
//regitros en la tabla Customers
myComand.CommandText = "SELECT Count(*) FROM Customers";
myconn.Open();
//El valor devuelto lo guardo en una variable tipo Entero
int CantFilas = Convert.ToInt32(myComand.ExecuteScalar());
//Cierro la Conecció
myconn.Close();
Console.WriteLine("Cantidad de Registros: " + CantFilas.ToString());
```

Objeto Command – Ejemplo II SQLCommand

```
//Creo un objeto SQLCommand
System.Data.SqlClient.SqlCommand myUpdate = new SqlCommand();
//Indico la conección a utilizar
myUpdate.Connection = myconn;
//Indico el tipo de llamada que realizaré (por default es Text)
myUpdate.CommandType = CommandType.StoredProcedure;
//Agrego los parámetros que necesita el StoreProcedure
myUpdate.Parameters.Add("@ID", SqlDbType.Int).Value = "1";
myUpdate.Parameters.Add("@Nombre", SqlDbType.Text).Value = "Juan";
//Indico el nombre del StoreProcedure
myUpdate.CommandText = "UpdateAlumnos";
//Abro la conección
myconn.Open();
//Ejecuto la operación
myUpdate.ExecuteNonQuery();
//Cierro la conección
myconn.Close();
```

Objeto DataReader

- Forward-only / Read-only
- Acceso rápido a los datos
- Conectado al origen
- La conexión la maneja usted mismo
- Los datos se manejan por código o a través de controles enlazados
- Usa pocos recursos

Objeto DataReader – Ejemplo I

```
//Declaro e instancio un objeto SqlCommand
System.Data.SqlClient.SqlCommand comand =
    new System.Data.SqlClient.SqlCommand();
//Indico el objeto Conextion que utilizará
comand.Connection = myconn;
//Indico la cadena TSQL utilizando la propiedad CommandText
comand.CommandText = "SELECT * FROM Customers";
//Abro la conección
myconn.Open();
//Declaro un objeto SqlDataReader e invoco el método
//ExecuteReader del objeto SqlCommand
System.Data.SqlClient.SqlDataReader dr = comand.ExecuteReader();
//Recorro la los registros moviendo el cursor a la siguiente posición
while (dr.Read())
{
    Console.WriteLine(dr["CustomerName"].ToString());
}
//Cierro la coneccion
myconn.Close();
```

Objeto DataReader – Ejemplo II

```
//Creo un objeto DataTable Tipado
DataSet1.tblAlumno_CursosDataTable tblAlumnos =
    new DataSet1.tblAlumno_CursosDataTable();

//Cargo el DataTable con el contenido obtenido
//de la base de datos
myconn.Open();
tblAlumnos.Load(dr);
myconn.Close();

//Creo otro objeto DataReader especifico para recorrer el contenido
//de un DataTable
System.Data.DataTableReader drAlumno = tblAlumnos.CreateDataReader();
while (drAlumno.Read())
{
    //.....
    Console.WriteLine(drAlumno["CustomerName"].ToString());
    //....
}
```

Objeto DataAdapter

- Gestiona el intercambio de datos entre DataTables y un Data Source
  - **.Fill** (DataSet o DataTable)
  - **.Update** (DataSet o DataTable)
- Provee relaciones entre tablas y columnas
- El usuario puede saltarse los comandos Insert/Update/Delete

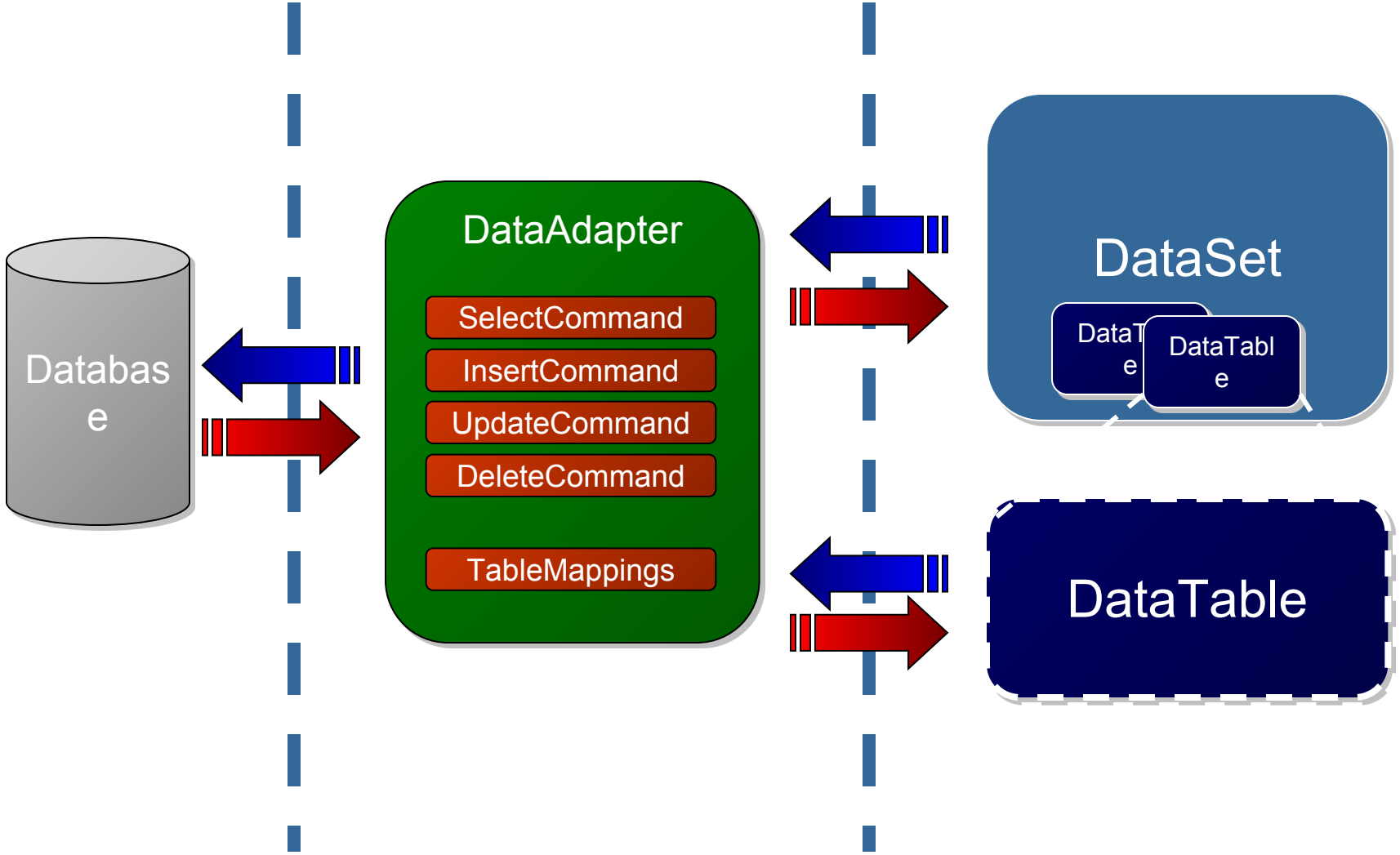
Objeto DataAdapter – Ejemplo Fill

- DataAdapter.**Fill**(DataTable)

```
System.Data.SqlClient.SqlDataAdapter sqlada =
    new System.Data.SqlClient.SqlDataAdapter("SELECT * ...", myconn);
DataSet1.tblUsuariosDataTable tblus = new DataSet1.tblUsuariosDataTable();

myconn.Open();
sqlada.Fill(tblus);//Relleno el DataTable con los resultados
myconn.Close();
//Por último me DESCONECTO
```

```
Dim sqlada As New SqlClient.SqlDataAdapter("SELECT....", myconn)
Dim tblus As New DataSet1.tblUsuariosDataTable()
myconn.Open()
sqlada.Fill(tblus) 'Relleno el DataTable con los resultados
myconn.Close()
'Por último me DESCONECTO
```

Objeto DataAdapter



Objeto DataAdapter – Ejemplo Update

- DataAdapter.**Update**(DataTable)

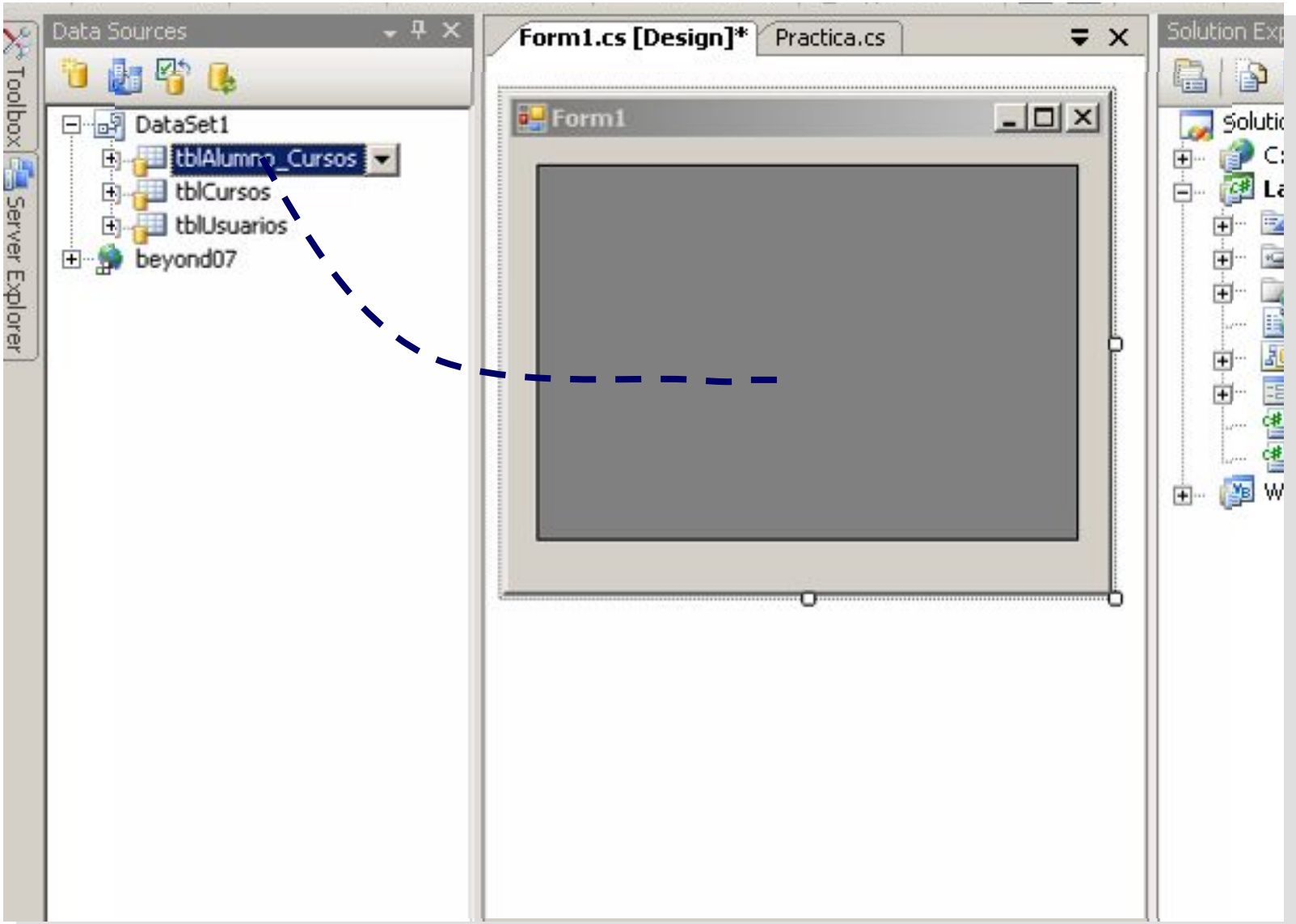
```
//Creo un objeto DataRow y le asigno un nombre
DataRow rowUsuario = tblus.NewRow();
rowUsuario["Nombre"] = "Mario";
//Adjunto el DataRow dentro del DataTable
tblus.Rows.Add(rowUsuario);
//Abro La Coneccion
myconn.Open();
//Actualizo todos los cambios realizados
//en el DataTable en la base de datos
sqlada.Update(tblus);
//Cierro la conexión
myconn.Close();
```

```
'Creo un objeto DataRow y le asigno un nombre
Dim rowUsuario = tblus.NewRow()
rowUsuario("Nombre") = "Mario"
'Adjunto el DataRow dentro del DataTable
tblus.Rows.Add(rowUsuario)

'Abro La Coneccion
myconn.Open()
'Actualizo todos los cambios realizados
'en el DataTable en la base de datos
sqlada.Update(tblus)
'Cierro la conexión
myconn.Close()
```

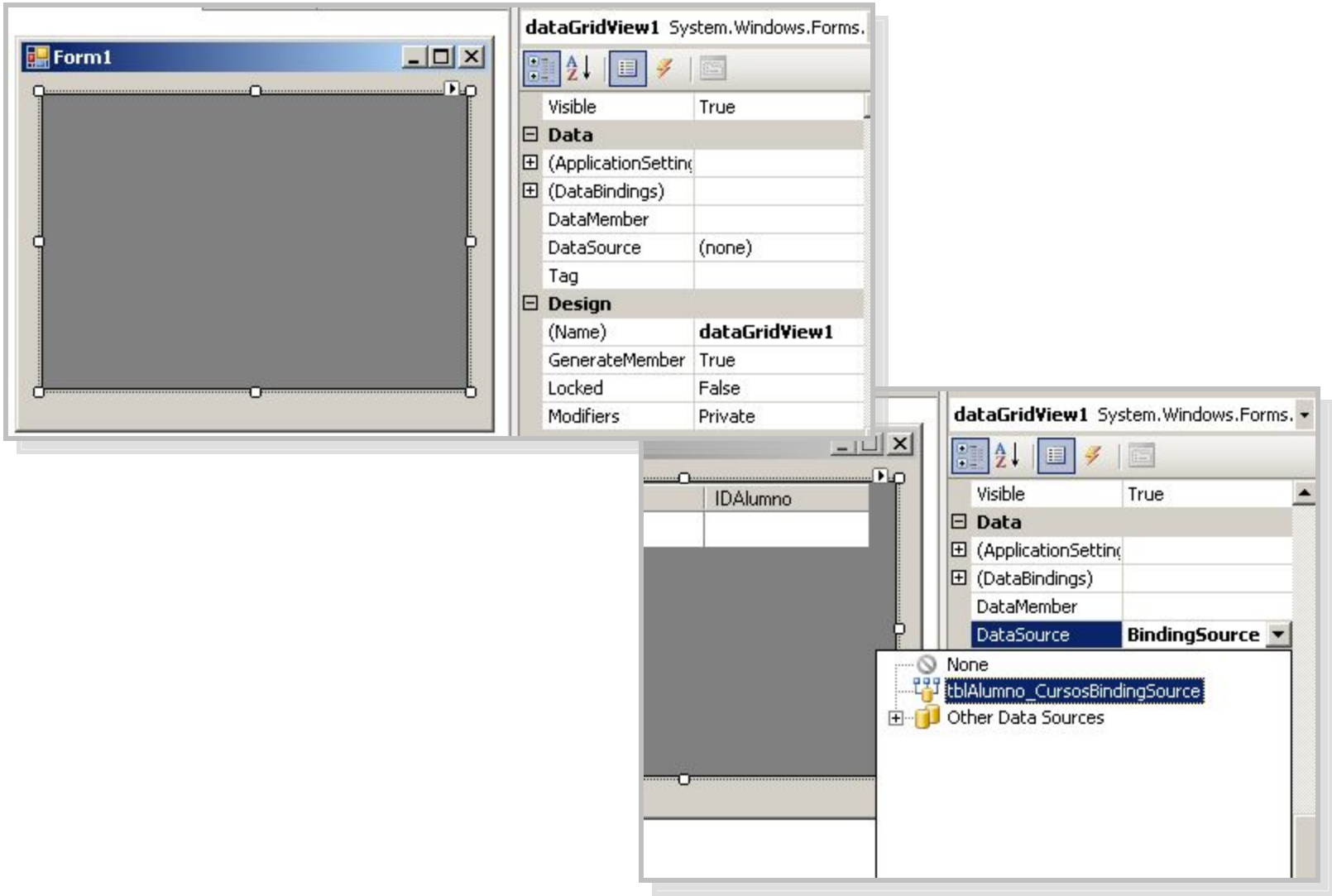
DataBinding (Data Sources)

- Mecanismo de Enlaces entre Objetos contenedores de datos (DataSet, WebService, DataBase) y los Controles WinForms
- Operaciones automatizadas
- No requiere escribir código
- Permite Navegación, Edición de registros
- Múltiples Origenes de Datos:
  - DataSet
  - Web Service’s
  - DataBase
- Se Establece en tiempo de Diseño

DataBinding (Data Sources) - Ejemplo



DataBinding (Data Sources) - Ejemplo



Objeto TransactionScope

- Agrupas operaciones combinadas en una unidad lógica de trabajo
- Controla y conserva la coherencia e integridad de todas las acciones
- Permite contener transacciones locales o distribuidas
- Método .Complete
- Encerrar en funcion Using()
- Agregar Referencia: System.Transaction
- NameSpace: System.Transaction
- Clase: TransactionScope

Objeto TransactionScope - Ejemplo

```
using (System.Transactions.TransactionScope tran = new
    System.Transactions.TransactionScope())//Declaro la Transacción
{
    try
    {
        System.Data.SqlClient.SqlCommand myUpdate = new SqlCommand();
        myUpdate.Connection = myconn;
        myUpdate.CommandType = CommandType.StoredProcedure;
        myUpdate.Parameters.Add("@ID", SqlDbType.Int).Value = "1";
        myUpdate.Parameters.Add("@Nombre", SqlDbType.Text).Value = "Juan";
        myUpdate.CommandText = "UpdateAlumnos";
        myconn.Open();
        myUpdate.ExecuteNonQuery();
        myconn.Close();
        tran.Complete = true; //Commit de la transacción
    }
    catch (Exception ex)
    {
        tran.Complete = false;//Rechazo todos los cambios realizados
        throw ex;
    }
}
```

Laboratorios

Laboratorio 1

Archivos TXT y XML

Laboratorio 2

Archivos TXT y XML con ADO.NET

Laboratorio 3

ADO.NET SQL y DataBinding