Conectando aplicaciones C# con bases de datos

OBJETIVOS

Que el alumno sea capaz de:

- Describir el proceso para enlazarse a una base de datos
- Enumerar los componentes necesarios para enlazar una base de datos
- Identificar los diferentes proveedores para una conexión a BDDs disponibles en VB.Net, así como sus características

CONTENIDOS

- Interfaces con accesos a Bases de Datos
- Acceso a datos con ADO.NET
- Características de ADO.NET
  - Espacio de nombres y clases básicas de ADO.NET
  - Namespace para datos en .NET Framework
- El Proveedor de datos de .NET
  - Objeto conecction
  - Objeto Command
  - Objeto DataReader

Interfaces de acceso a bases de datos

Evolución histórica de los "estándares" propuestos por Microsoft:

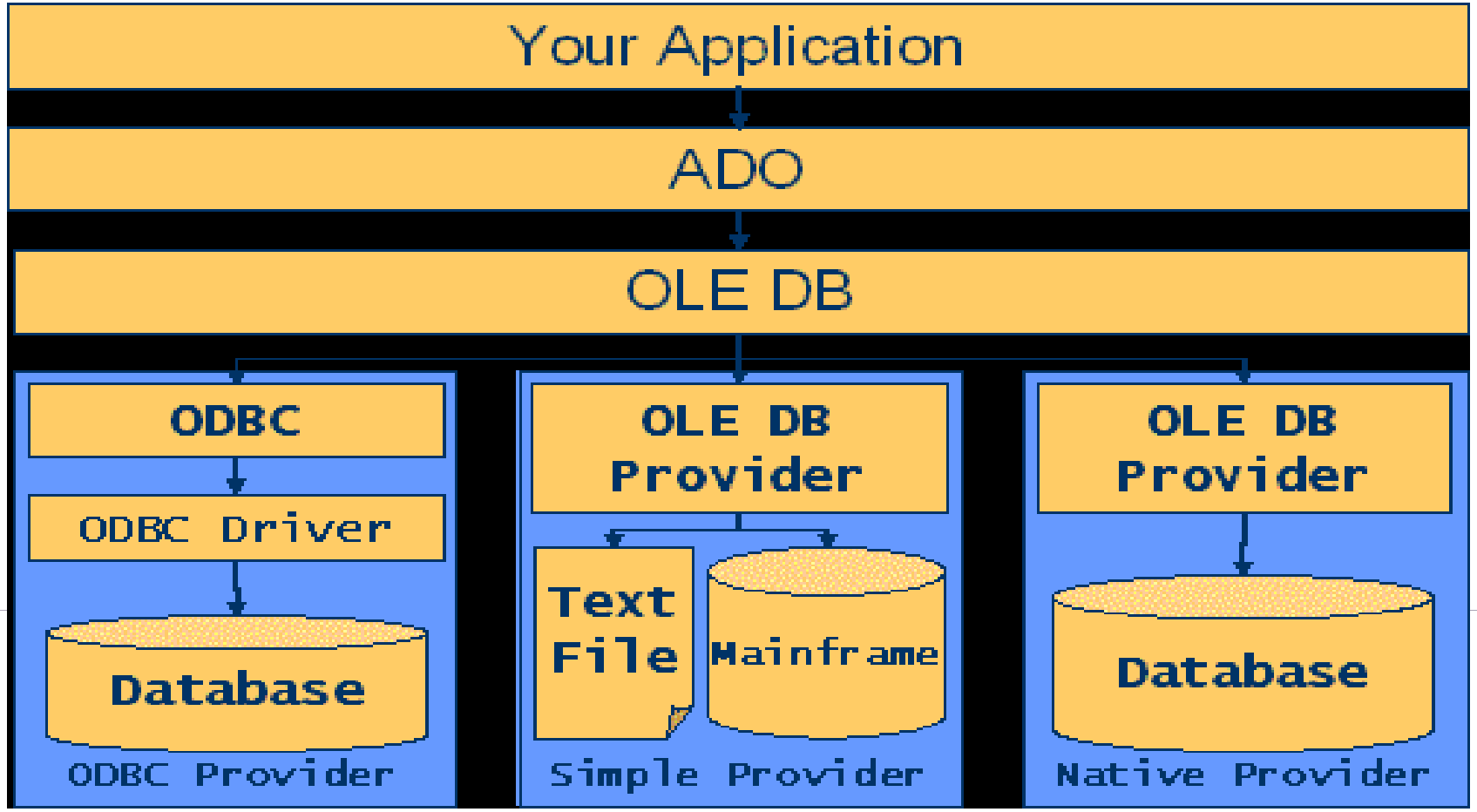
- **DBC (Open Database Connectivity)**: API estándar ampliamente utilizado, disponible para múltiples DBMSs, utiliza SQL para acceder a los datos.
- **DAO (Data Access Objects)**: Interfaz para programar con bases de datos JET/ISAM, utiliza automatización OLE y ActiveX.
- **RDO (Remote Data Objects)**: Fuertemente acoplado a ODBC, orientado al desarrollo de aplicaciones cliente/servidor.

Interfaces de acceso a bases de datos

- **OLE DB**: Construido sobre COM, permite acceder a bases de datos tanto relacionales como no relacionales (no está restringido a SQL). Se puede emplear con controladores ODBC y proporciona un interfaz a bajo nivel en C++.

- **ADO (ActiveX Data Objects)**: Ofrece un interfaz orientado a objetos y proporciona un modelo de programación para OLE DB accesible desde lenguajes distintos a C++ (p.ej. Visual Basic).

Interfaces de acceso a bases de datos



ADO.NET (1)

ADO (ActiveX Data Object).NET

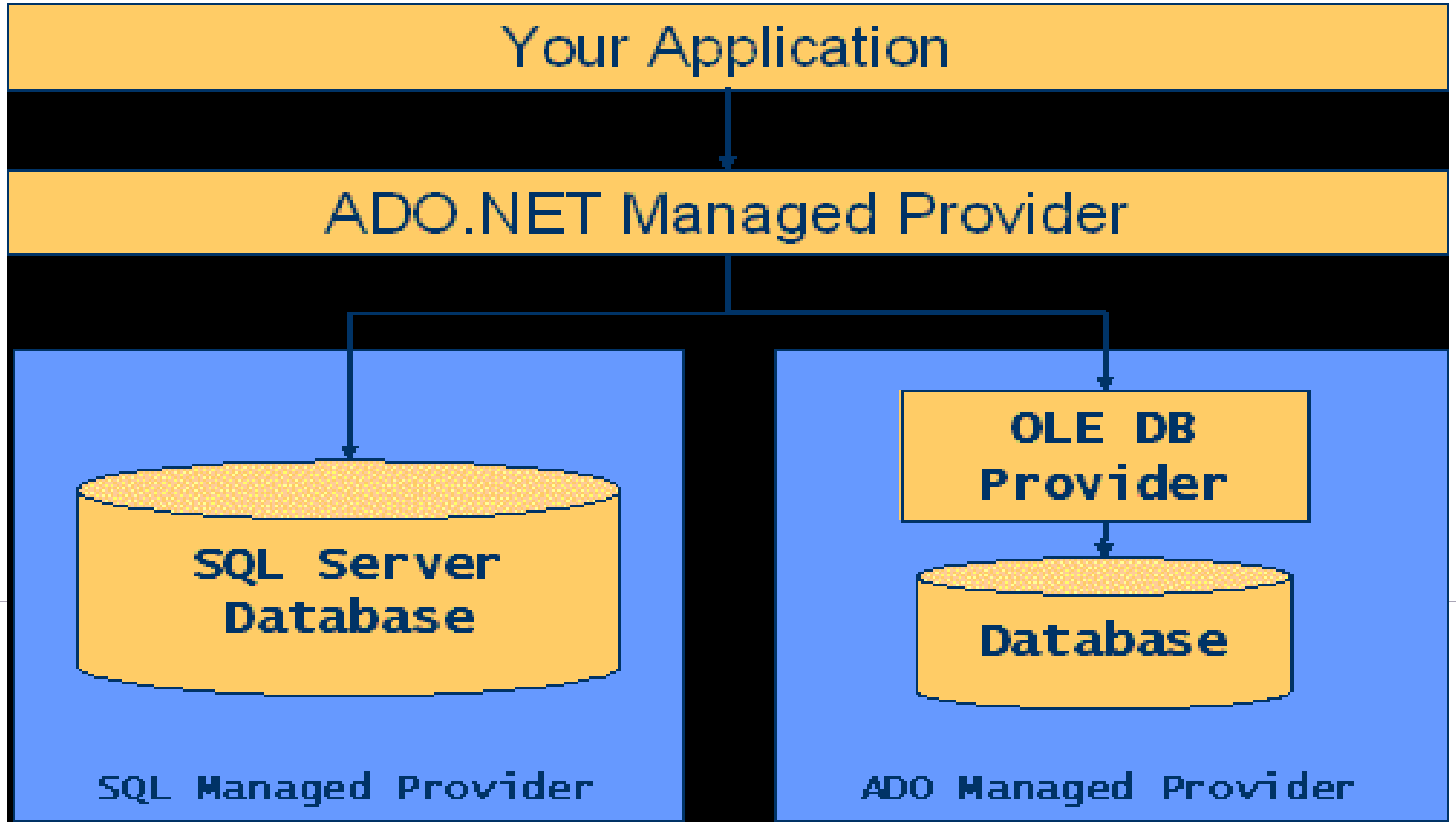
- Es la tecnología de acceso a datos utilizada por Visual Studio .Net y por el resto de aplicaciones del .Net Framework.
- Esta integrado en .Net Framework y proporciona acceso a datos.
- Consta de un conjunto de clases que se utilizan para acceder y manipular orígenes de datos, como por ejemplo: una base de datos o una planilla en excel, etc

ADO.NET (2)

- **ADO.NET** es una evolución del modelo de acceso a datos de ADO, que controla directamente los requisitos del usuario para programar aplicaciones escalables.
- Se diseñó específicamente para la Web, teniendo en cuenta la escalabilidad, la independencia y el estándar XML.

- Incluye una serie de **proveedores** que actúan como intermediarios entre la base de datos y la aplicación:

Acceso a los datos (2)



ADO.NET: Fundamentos (1)

- El modelo de objetos de accesos de datos tradicional se basa en establecer una conexión con la base de datos y mantenerla abierta mientras la aplicación realiza las diferentes operaciones con datos.
- Este modelo requiere mantener conexiones abiertas durante un largo periodo de tiempo, ocasionando un deterioro del rendimiento de la aplicación (las conexiones abiertas consumen muchos recursos del servidor) y puede bloquear acceso a otros usuarios, pues los servidores de datos solo pueden proporcionar un numero limitado de conexiones.
- Este problema puede ser especialmente grave en aplicaciones Web, donde además de haber un elevado numero de usuarios accediendo a la misma base de datos, las conexiones entre la aplicación y el servidor de datos se realizan a través de una serie de redes.

ADO.NET: Fundamentos (2)

- Dentro de la librería ADO.NET encontramos clases específicamente creadas para el trabajo en modo desconectado
- Ademas, proporciona la posibilidad de actualizar el origen de datos con el cambio realizado sobre una copia local.

- ADO.NET proporciona a las aplicaciones 2 modos de trabajo con un origen de datos:
  - **Modo conectado**
  - **Modo desconectado**

ADO.NET: Fundamentos (3)

Acceso a datos en modo conectado

- Accede a los datos reales existentes en una base de datos a través de una conexión establecida con la misma.

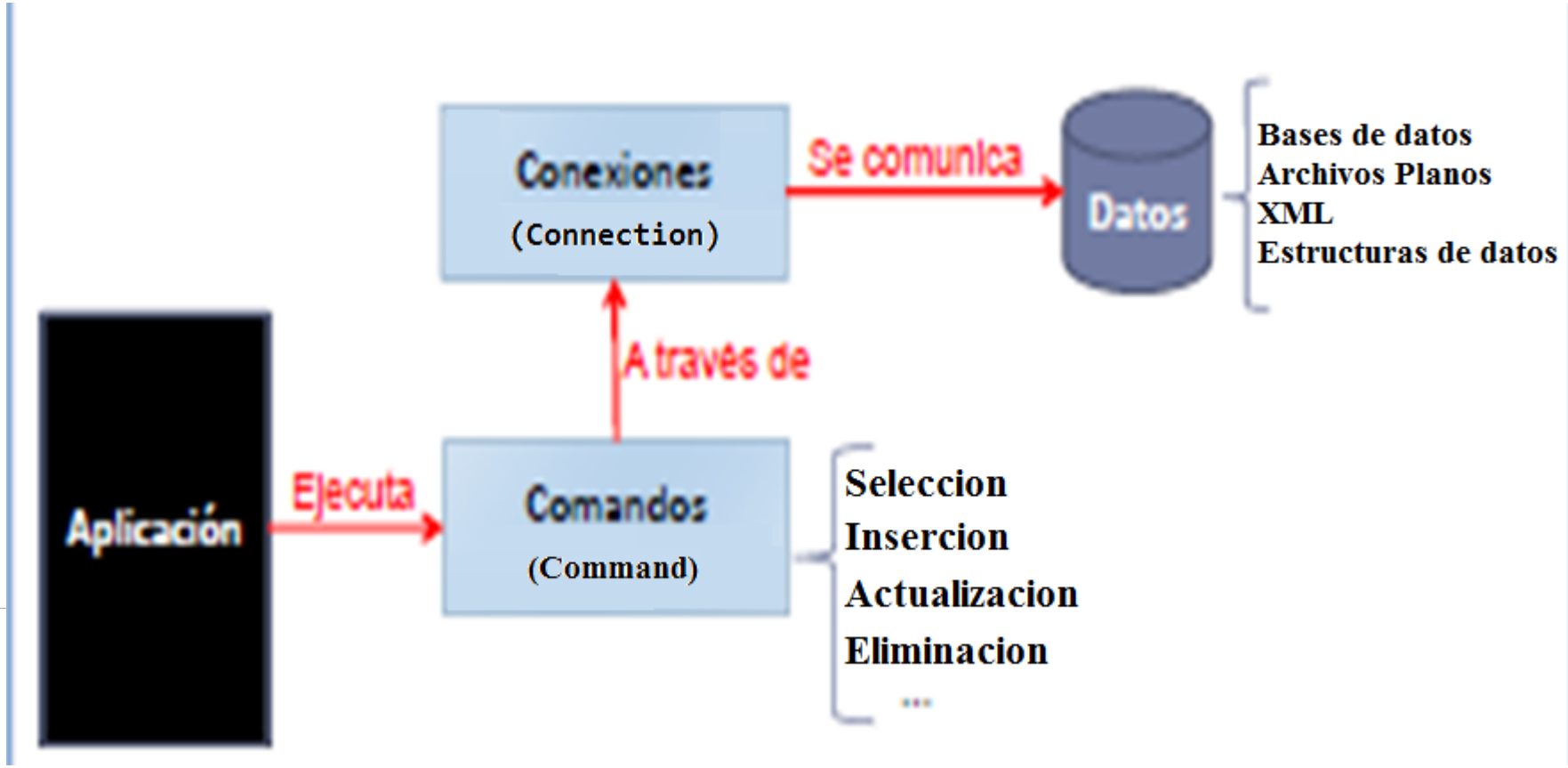
- Una aplicación recurrirá a este modo de acceso cuando necesite realizar alguna operación puntual sobre la base de datos, como la inserción, actualización, localización y eliminación de un registro o la recuperación de un conjunto de registros para ser mostrados en un pagina web, siendo en estos casos la solución que ofrece mayor rapidez y eficiencia.

Características de ADO.NET

- ADO.Net utiliza el acceso a datos para entornos conectados; es decir que se conecta, hace lo que tiene que hacer y se desconecta.

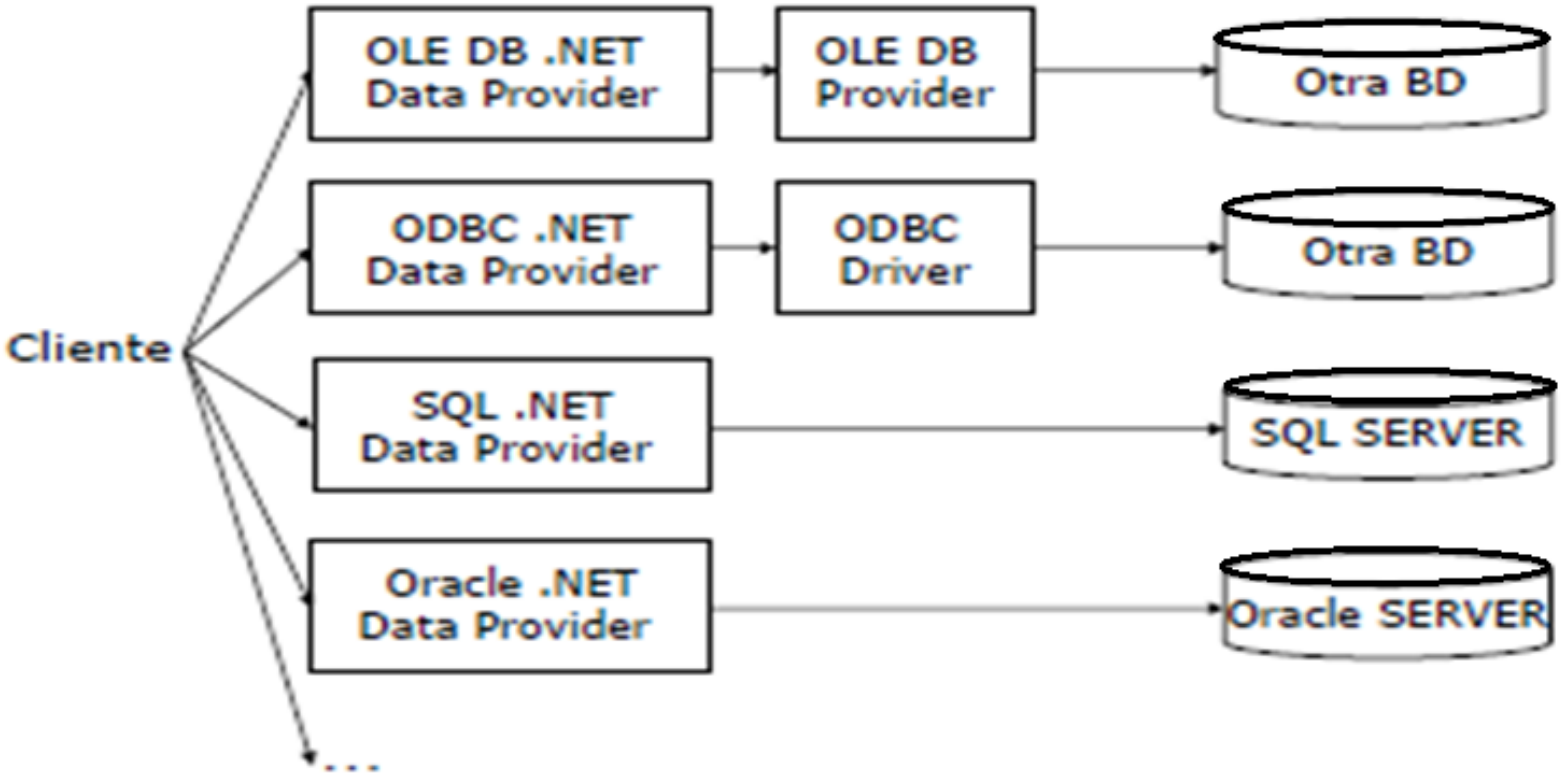
- Su arquitectura permite crear componentes que administran eficientemente datos procedentes de múltiples orígenes, y también proporciona las herramientas necesarias para solicitar y reconciliar datos entres grupos de aplicaciones.

Estructura General de ADO.NET



Proveedores de Acceso Datos (1)

- Son un conjunto de clases adaptadas para trabajar con un origen de datos especifico.
- Si una aplicación necesita acceder a un determinado origen de datos, debe utilizar el proveedor correspondiente a este tipo de origen de datos.



Proveedores de Acceso Datos (2)

ADO.NET

- **OLE DB**
  - **Es un proveedor generico**
  - **Acceso vía protocolo OLE DB a cualquier fuente de datos que lo soporte**
- **ODBC**
  - **Acceso vía protocolo ODBC a cualquier fuente de datos que lo soporte**

Proveedores de Acceso Datos (3)

ADO.NET

- **SQL Server**
  - **Acceso nativo a MS SQL Server 7.0 o superior y MS Access**
- **Oracle**
  - **Acceso nativo a Oracle Server**

Otros provistos por terceros

- **MySQL, PostgreSQL, etc.**

Espacios de Nombre (1)

- Las clases de cada Proveedor de Datos de ADO.NET están definidas en el espacio de nombres System.Data

- La jerarquia de espacios de nombres esta organizada estructuralmente de la siguiente manera:

a. Objetos de System.Data
b. Proveedores de acceso a datos.

Espacios de Nombre (2)

Jerarquía de espacio de Nombres (namespace)

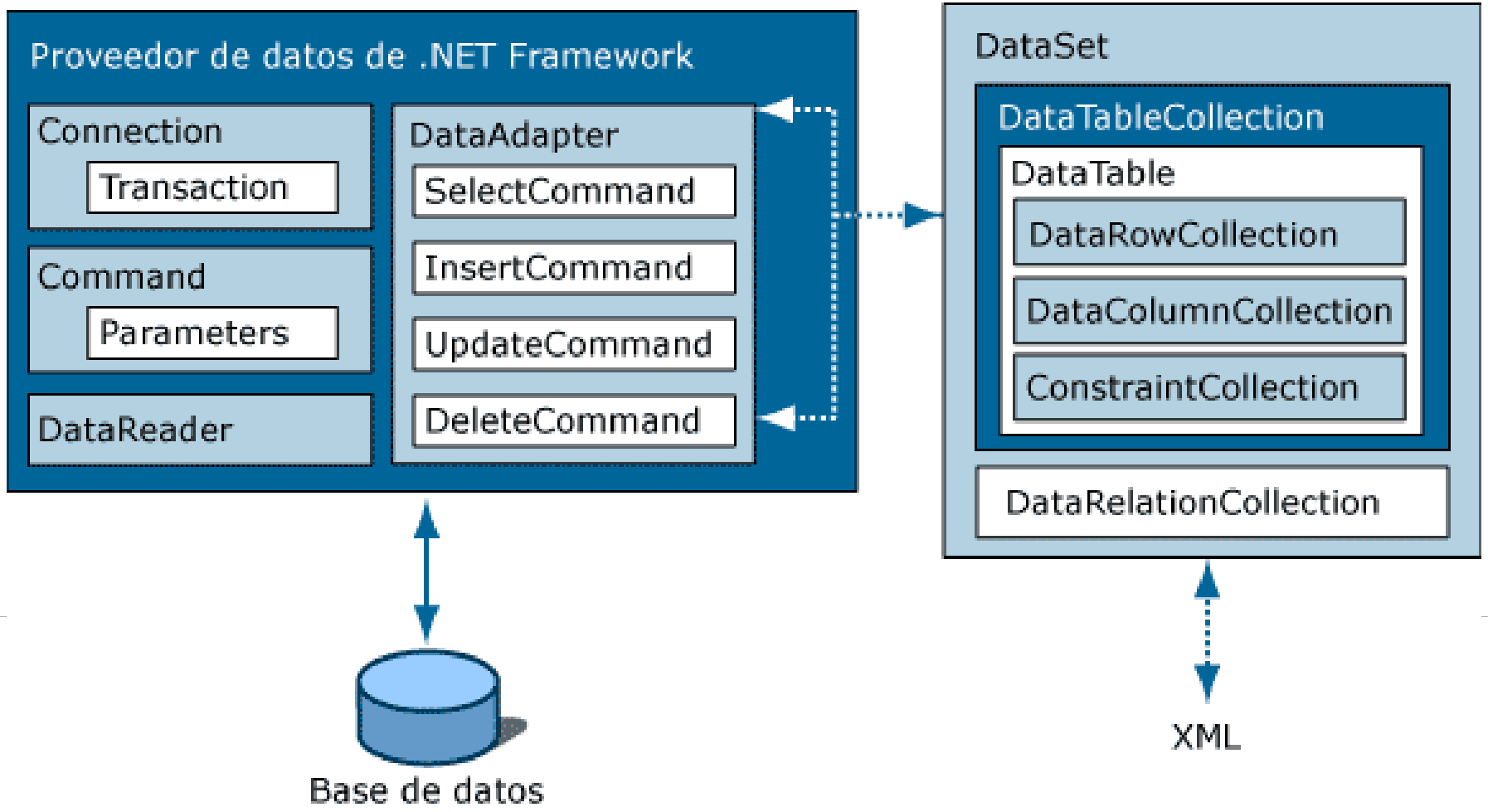
- **System.Data**
- **System.Data.Common**
- **System.Data.OleDb**
- **System.Data.SqlClient**
- **System.Data.OracleClient**
- **System.Data.Odbc**

Espacios de Nombre (3)

Namespace **System.Data**:

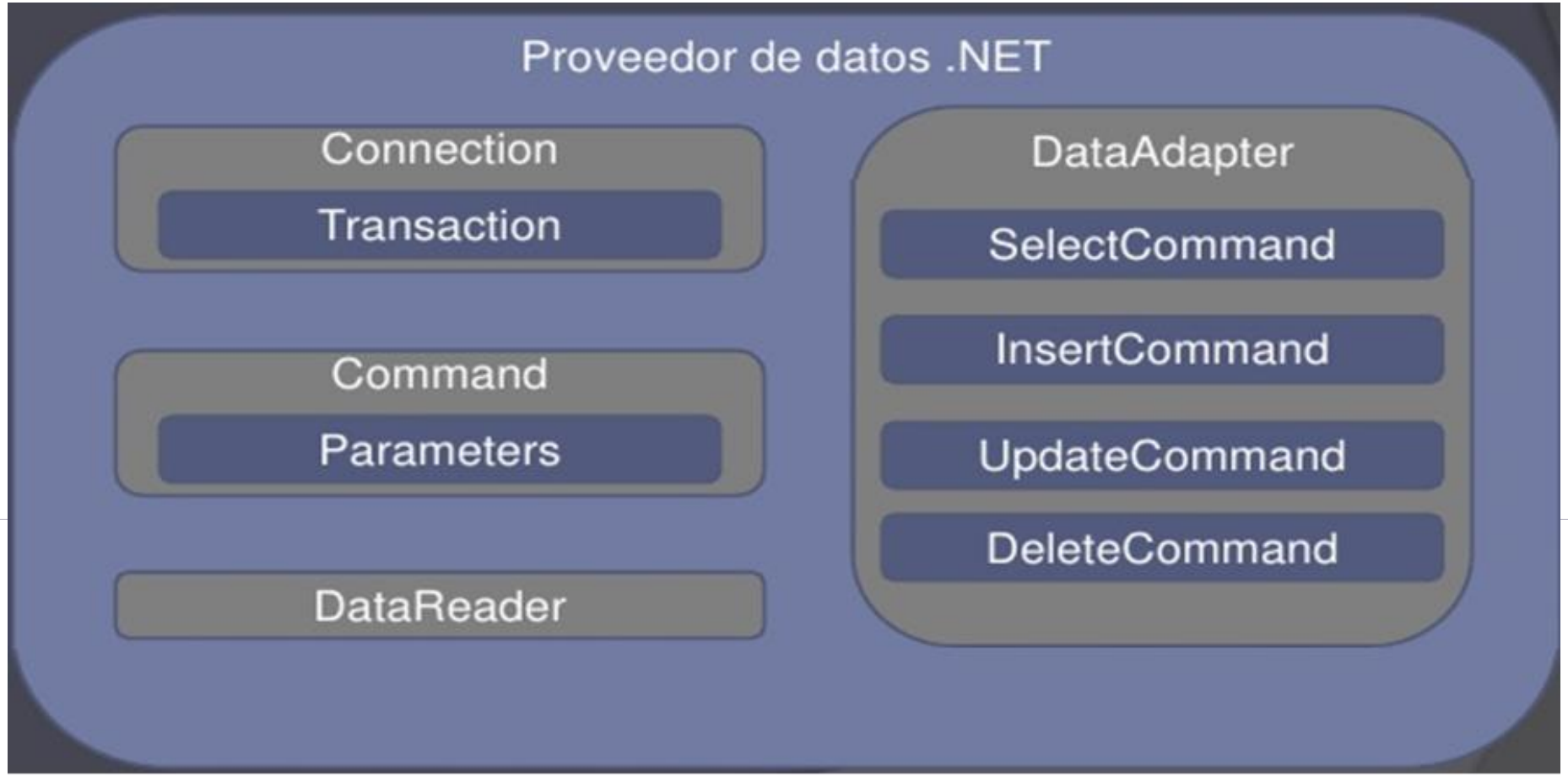
- Contiene la base de ADO.NET
- Proporciona el modo de trabajo sobre datos
  - Clases y métodos para la manipulación de datos
  - Posibilidad de creación de vistas
  - Representación lógica de los datos
  - Utilización de XML para visualizar, compartir y almacenar datos

Arquitectura de ADO.NET (1)



Arquitectura de ADO.NET (2)

Jerarquia de clases de un Proveedor de datos .NET



ADO.NET: Clases (1)

- **Connection**: Permite establecer un canal de comunicación con una fuente de datos.
- Los objetos connection representan conexiones activas con la base de datos y deben permanecer abiertas para poder transferir información entre esta y la aplicación.
- Ejemplos de clases:
  - OleDBConnection
  - OracleConnection
  - SqlConnection

ADO.NET: Clases (2)

- **Command**: Ejecutan comandos de consultas SQL a una fuente de datos a través de una conexión. También pueden ejecutar procedimientos almacenados.
- Ejemplos de clases
  - OleDBCommand
  - OracleCommand
  - SqlCommand

ADO.NET: Clases (3)

- **DataReader**: Permite acceder a los registros retornados de una consulta SQL (tipo select) o procedimiento almacenado.
- Es un objeto ligero y rápido que permite leer "Un registro a la vez", de modo secuencial.
- Es un objeto de solo lectura
- No da información del tipo de dato que está leyendo, el desarrollador debe conocerlos y también el orden de las columnas.
- Ejemplos de clases.
  - OleDBDataReader
  - OracleDateReader
  - SqlDataReader

ADO.NET: Clases (4)

- **Parameter**: Este objeto contiene información relativa a un parámetro de un comando, que va a ser enviado a la base de datos para su ejecucion.

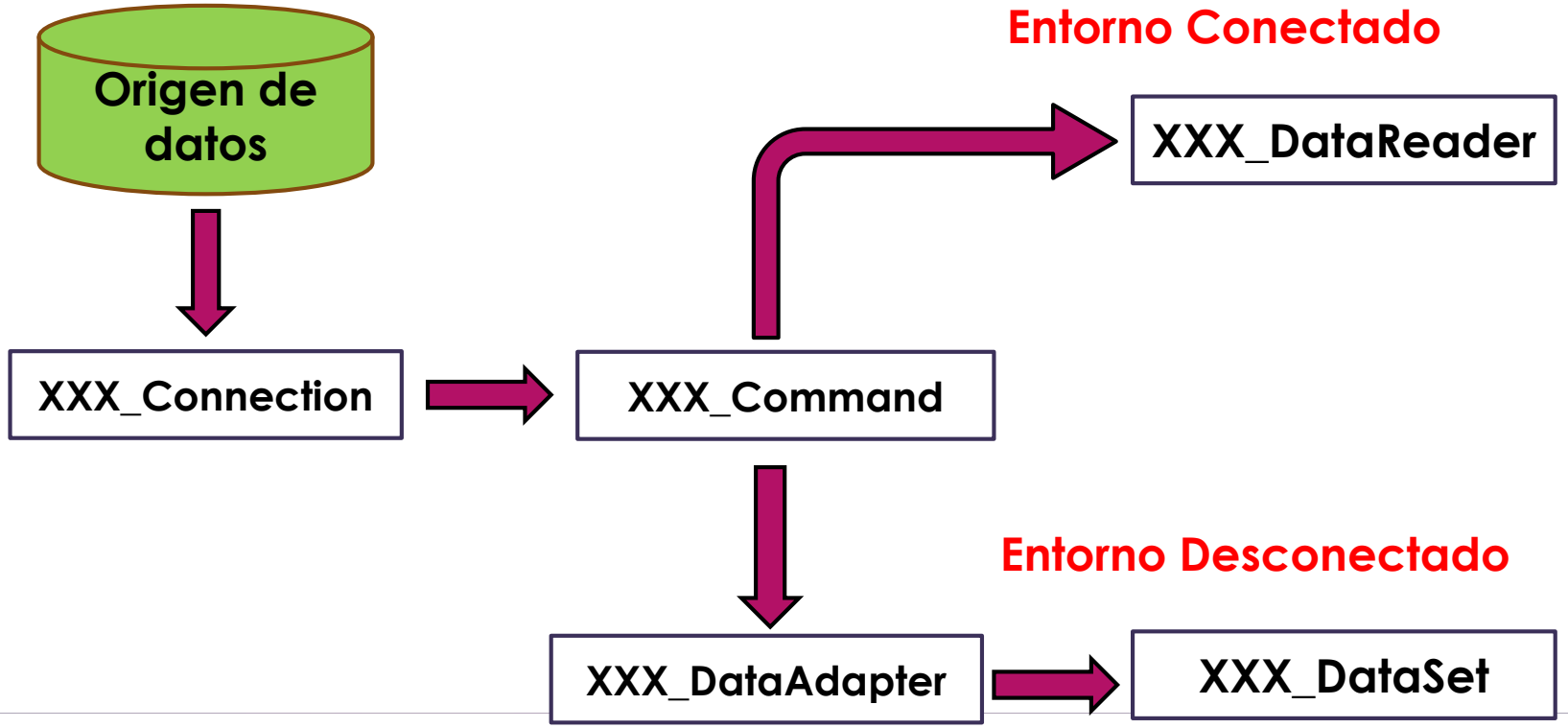
- **Transaction**: Permite agrupar operaciones sobre la base de datos dentro de una transacción, a fin de que estas sean consideradas conjuntamente como una única unidad de ejecución.

Procesos de acceso a los datos (1)

- Cuando una aplicación .NET necesita acceder a una base de datos (utilizando el modo conectado de ADO.NET), debe seguir los siguientes pasos en el orden indicado:

1. Establecimiento de una conexión
2. Ejecución de la consulta SQL
3. Manipulación de resultados (si procede)
4. Cierre de la conexión

Procesos de acceso a los datos (2)



Nota:
XXX_ Identifica al nombre del proveedor de ADO.NET seleccionado para acceder al Origen de datos

Ejemplo de configuración de conexión sqlserver

Form1.cs [Diseño]*

_SEMANA 8 | PRUEBA_SEMANA_8.Form1 | Form1()

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Data.SqlClient;
namespace PRUEBA_SEMANA_8
{
    3 referencias
    public partial class Form1 : Form
    {
        1 referencia
        public Form1()
        {
            InitializeComponent();
        }
        1 referencia
        private void button1_Click(object sender, EventArgs e)
        {

            SqlConnection conexion = new SqlConnection("server= localhost; database= AdventureWorks2012; integrated security = true");
            conexion.Open();
            MessageBox.Show("Se abrió la conexión");
            conexion.Close();
            MessageBox.Show("Se cerró la conexión");
        }
    }
}
```

No se encontraron problemas. Línea: 6

Ejemplo de configuración de conexión sqlserver

Form1.cs [Diseño]*

_SEMANA 8 | PRUEBA_SEMANA_8.Form1 | Form1()

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Data.SqlClient;
namespace PRUEBA_SEMANA_8
{
    3 referencias
    public partial class Form1 : Form
    {
        1 referencia
        public Form1()
        {
            InitializeComponent();
        }
        1 referencia
        private void button1_Click(object sender, EventArgs e)
        {

            SqlConnection conexion = new SqlConnection("server= localhost; database= AdventureWorks2012; integrated security = true");
            conexion.Open();
            MessageBox.Show("Se abrió la conexión");
            conexion.Close();
            MessageBox.Show("Se cerró la conexión");
        }
    }
}
```

No se encontraron problemas. Línea: 6

Ejecución de una consulta sql

- Desde el momento en que se abre la conexión ya es posible ejecutar consultas SQL sobre ella.
- En ADO.NET las instrucciones SQL y las llamadas procedimientos almacenados son enviadas desde la aplicación a la base de datos utilizando objetos de la clase command.
- **Creación de objetos Command**

Ej.

```
String cadenaSQL;

cadenaSQL= “Insert into clientes”;

cadenaSQL +=“values ”+ “(‘miuse’,’mipass’)”;

Sql Command cmd=new sqlCommand(cadenaSql, cn);
```

Ejecución de una consulta sql (2)

Ejecución de consulta:

La clase command dispone de los siguientes metodos para ejecutar una consulta:

- **ExecuteNonQuery()** Se utiliza para ejecutar consultas de acción que no devuelven resultados.
- **ExecuteReader()** Se utiliza para ejecutar consultas de selección de registros. Como resultado, devuelve un objeto DataReader que permite acceder a los datos de la consulta.

En el ejemplo anterior, la ejecución de la consulta se llevaría acabo con el primer método.

```
cmd.ExecuteNonQuery
```

Cierre de una conexion

- Una vez finalizadas las operaciones sobre la base de datos, a fin de liberar recursos en el servidor, debemos proceder al cierre de la conexión tan pronto como sea posible.
- Esta operación ser realiza invocando al método close( ) del objeto connection:

```
cn.close( );
```

ENLACE

https://franklintutoriales.blogspot.com/2013/07/aplicacion-en-c-y-sql-server.html