Carrera: **Analista Programador**

Lenguajes de Última Generación

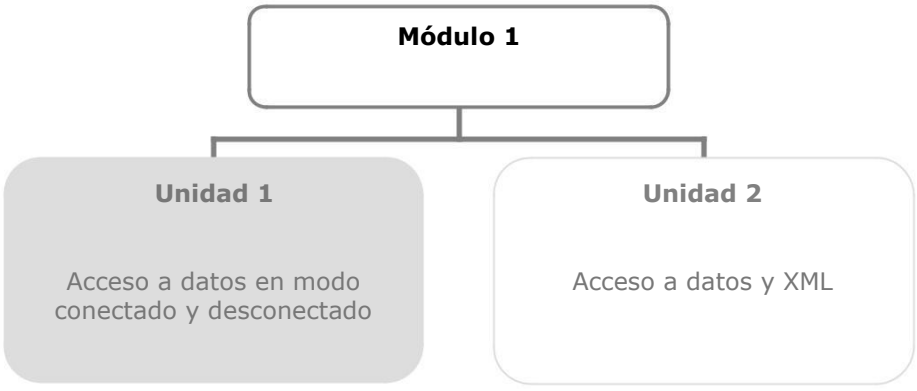
Módulo 1

Aprovechar el acceso a datos

Unidad 1

Acceso a datos en modo conectado y desconectado

Autor de contenidos: Ing. Mauricio Prinzo



Presentación

Cuando desarrollamos un sistema es requisito obligatorio resguardar toda la información.

Existen diversos tipos de sistemas que podemos clasificar rápidamente como educativos, lúdicos, de gestión administrativa, etc. En todos los sistemas la información es el recurso más valioso y es por ello que los sistemas de información se complementan con bases de datos.

No es el fin de este material profundizar en el concepto y uso de las base de datos, pero sí en los medios que ofrece el lenguaje de programación para interactuar con ella.

Esperamos que al terminar el modulo usted alcance la comprensión de los contenidos para poder:

Acceder rápidamente a una base de datos relacional.
Comprender las ventajas de trabajar en modo conectado o desconectado.
Comprender las estructuras de datos utilizadas en un lenguaje de programación
Trabajar con estructuras planas de datos que faciliten el intercambio de datos.

A continuación, le presentamos un detalle de los contenidos y actividades que integran esta unidad. Usted deberá ir avanzando en el estudio y profundización de los diferentes temas, realizando las lecturas requeridas y elaborando las actividades propuestas, algunas de desarrollo individual, otras para resolver en colaboración con otros estudiantes siempre acompañado con su profesor tutor.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Contenidos y Actividades

1. Conceptos generales

1.1. Base de datos relacional

1.2. Proveedores de datos

1.3. Arquitectura de ADO.net

1.4. Espacio de nombres ADO.net

2. Modo conectado

2.1. Objeto SqlConection

2.2. Objeto SqlCommand

2.3. Objeto sqlDataREader

3. Modo Desconectado

3.1. Objeto Dataset

3.2. Objeto SqlDataadapter

Lectura Requerida

- Deitel, Harvey M.; Deitel, Paul J.; Vidal Romero Elizondo, Alfonso(Traductor); y otros. **Cómo programar en C#.** 2a.ed.-- México, DF. Capítulo 20.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Trabajo colaborativo/Foro

Debate sobre los fundamentos planteados en el módulo.

Para el estudio de estos contenidos usted deberá consultar la bibliografía que aquí se menciona:

BIBLIOGRAFÍA OBLIGATORIA

Deitel, Harvey M.; Deitel, Paul J.; Vidal Romero Elizondo, Alfonso(Traductor); y otros. **Cómo programar en C#.** 2a.ed.-- México, DF..

Bibliografía Ampliatoria

Pearson Mayo, Joseph. **C# al descubierto**. Madrid : Pearson Educación, c2002. xxxi, 749 páginas

Links a temas de interés

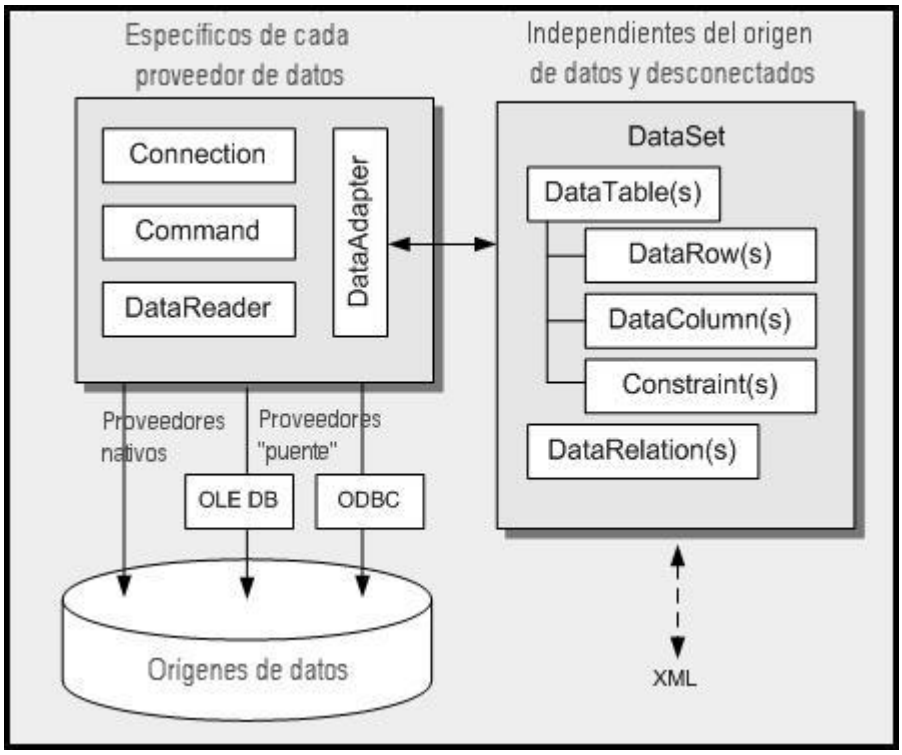
http://Biblioteca.vaneduc.edu.ar/

El portal de la biblioteca universitaria con material y links de interés, además del asesoramiento de un referencista especializado.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Organizador Gráfico

El siguiente esquema le permitirá visualizar la interrelación entre los conceptos que a continuación abordaremos.



Lo/a invitamos ahora a comenzar con el estudio de los contenidos que conforman esta unidad.

1. Conceptos generales

Con frecuencia las aplicaciones que se desarrollan necesitan *persistir* sus datos en el tiempo para poder ser reutilizados. Las **bases de datos relacionales** cumplen ese rol.

La aparición de ADO.NET produjo un cambio en la manera de pensar la conexión con las fuentes de datos y, cómo utilizarlo es lo que intentaremos desentrañar en este módulo.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

1.1. Base de datos relacional

Las bases de datos relacionales son los motores que cumplen con el modelo relacional - actualmente uno de los modelos más utilizados. Es importante señalar que aunque existen bases de datos orientadas a objetos aún no se encuentran aceptadas completamente por las empresas.

Este modelo se conforma por entidades y sus interrelaciones. Entre ellas aseguran la formación de un modelo normalizado que impide la perdida de información y asegura su recuperación en forma precisa y rápida.

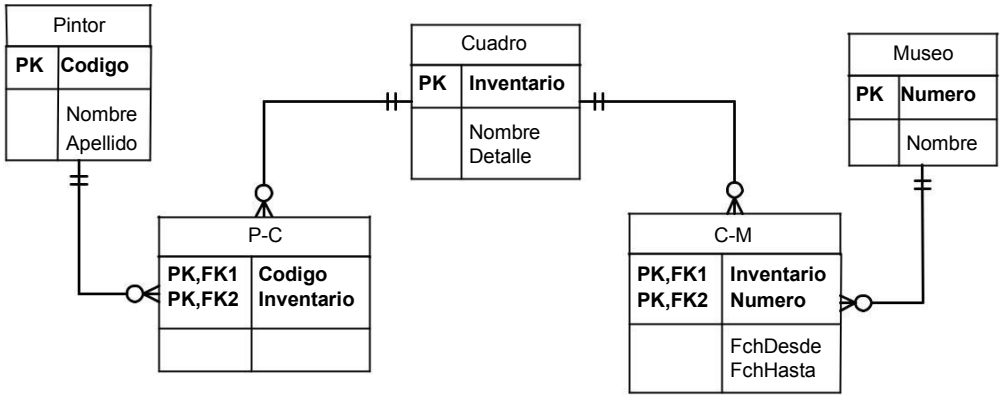
Las **entidades** son el conjunto de campos que conforman una tabla de la base de datos, generalmente expresadas con un nombre en singular como por ejemplo "cliente".

Los **campos** son los descriptores de dicha entidad, generalmente representados por un nombre y un tipo de dato un ejemplo de ello sería CUIL de tipo texto.

El objetivo primario de las bases de datos es evitar la duplicación innecesaria de la información razón por la cual no toda la información puede estar en solo una entidad.

Las interrelaciones son las asociaciones que existen entre las entidades y expresan la vinculación de los datos.

Un ejemplo de bases de datos podría ser el siguiente:



Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

En el diagrama, que llamaremos DER (Diagrama entidad relación), podemos reconocer cuatro entidades: Pintor, P-C, Cuadro, C-M, Museo. Y sus campos como código, nombre etc.

En la base de datos las entidades o tablas se pueden representar como una matriz de doble entradas, tal como se muestra en la figura siguiente:

Las filas representan los registros o *tuplas*, y las columnas los campos. Por cada pintor que se carga en la tabla, se reserva un registro completo.

Para acceder a los datos en una tabla se utiliza un lenguaje estándar que se llama SQL (Stándar Query Language). El SQL se forma por tres grupos básicos:

1) la selección de los campos
2) las tablas que participan
3) las condiciones de filtrado.

Un caso posible de SQL para obtener todos los campos de la tabla pintor sería como SELECT * FROM pintor.

Por supuesto, este tema es mucho más complejo y merece un tratamiento profundo y detallado que excede nuestra asignatura. No obstante, le proponemos la realización del siguiente ejercicio.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Actividades para la facilitación de los aprendizajes

Actividad 1: Conocer SQL, crear Tablas y ejecutar qquery

Ejercicio 1

a) Cree una Base de Datos con 3 tablas (socio, pais, provincia)
b) La tabla socio posee los campos Id_Socio, Nombre, Apellido, email, Id_Pais, Id_provincia
c) La tabla Pais posee los campos Id_pais, Nombre
d) La tabla Provincia posee los campos Id_Provincia, Nombre
e) Una vez creda la Base de Datos mostrar el grafico y las relaciones de las mismas

Ejercicio 2

a) Realice la consulta de traer todos los datos de la tabla socios y otra de la tabla país y provincia
b) Realicer una consulta que traiga solo los registros Id_socio, nombre, email de los socios de la provincia de Bs AS
c) Realice una consulta que traiga solos los registros Id_socio, nombre, email de los socios llamados Pablo
d) Saque la cuenta del total de socios, otra para provincias.

Si tiene alguna duda, por favor, compártala con pares y tutor.

Si lo desea, puede ampliar estos contenidos realizando búsquedas personales o consultando a su tutor.

1.2. Proveedores de datos

Los **proveedores de datos** facilitan la comunicación con los motores de base de datos. Existe en el mercado un proveedor para cada base de datos.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Los proveedores de datos brindan los medios de comunicación necesarios para que dos aplicaciones puedan convivir en escenarios totalmente diferentes. En este caso, su aplicación y el motor de base de datos.

Los proveedores, tienen la inteligencia para sumar a su aplicación los conocimientos necesarios para que interactuar con la base de datos sea una tarea sencilla.

ADO.NET propone tres proveedores distintos:

1- **OLEDB**: aprovecha los proveedores OLEDB que existen en su sistema operativo actual. Generalmente se sincroniza con código no administrado. Recuerde que en otras asignaturas usted ha estudiado que en .NET se puede trabajar con código administrado o no administrado.

2- **SQL server:** Es un proveedor escrito exclusivamente para base de datos SQL SERVER. Ofrece mejores prestaciones que OLEDB y la comunicación es directa haciendo mucho más performante el intercambio de datos entre las dos aplicaciones, es decir entre su programa y la base de datos

3- **ODBC es un** proveedor limitado, pues no brinda muchos servicios para realizar la comunicación con la base de datos. Permite conectarse con Access y Oracle entre otros motores de base de datos

1.3. Arquitectura ADO.NET

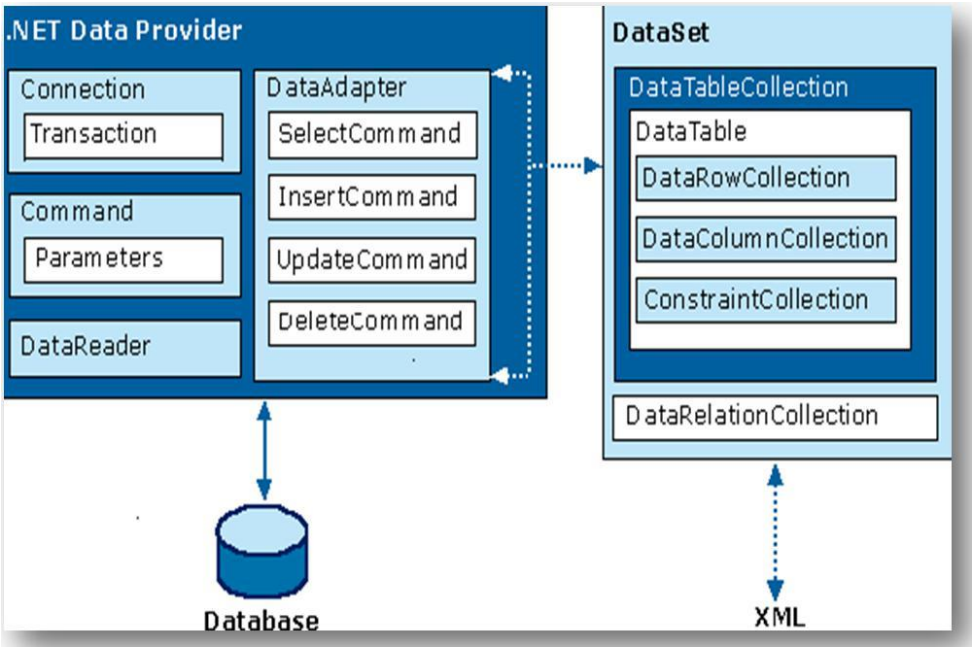
ADO.NET representa la relación entre todos los objetos que permiten la conexión a las bases datos. Está constituido por algunos objetos que conforman la arquitectura y otros que, fuera de ella, son utilizados para fines determinados.

Dentro de ADO.NET tenemos los objetos **Conection**, **Command**, **Adapter**, **dataReader**.

Si trabajamos con SQL SERVER, los renombramos a **sqlConection**, **sqlCommand**, **sqlDataAdapter**, **sqldataReader**.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Fuera del modelo tenemos el **dataset**.



Vamos a detenernos un momento para intentar ordenar la información que explicamos hasta aquí y ordenar los conocimientos aprendidos.

Las base de datos son aplicaciones externas que nos permiten resguardar toda la información que circula por nuestros procesos, y nos ofrece facilidades para leer, guardar, modificar o recuperar los datos que guardamos. Para que nuestro sistema pueda interactuar con las bases de datos necesita comunicarse con ella, y esto puede lograrse usando distintos proveedores de servicios.

Pero esto no termina allí, .Net tiene un conjunto de objetos llamado en ADO.NET que apoyado el proveedor puede conectarse a la base de datos y cumplir con nuestro objetivo: poder manipular los datos almacenados.

1.4. Espacio de nombres ADO.NET

El modelo de acceso de datos se encuentra dentro del framework.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

El framework es un set de funciones y servicios a los que se accede por los **espacios de nombre** (namespace).

Los espacios de nombre utilizados por ADO.NET son:

System.Data
System.Data.SqlClient
System.Data.OleDB
System.Data.Odbc

2. Modo Conectado

Existen dos modos de objetos para acceder a Datos:

1 Modo conectado
2 Modo desconectado

Veremos el primero.

2.1. Modo conectado

Decimos que un entorno está conectado cuando el modelo de datos está conectado continuamente al motor de la base de datos.

Esta estrategia es más fácil de mantener usando simplemente los objetos que veremos a continuación, de este modo los conflictos con la concurrencia se controlan mejor.

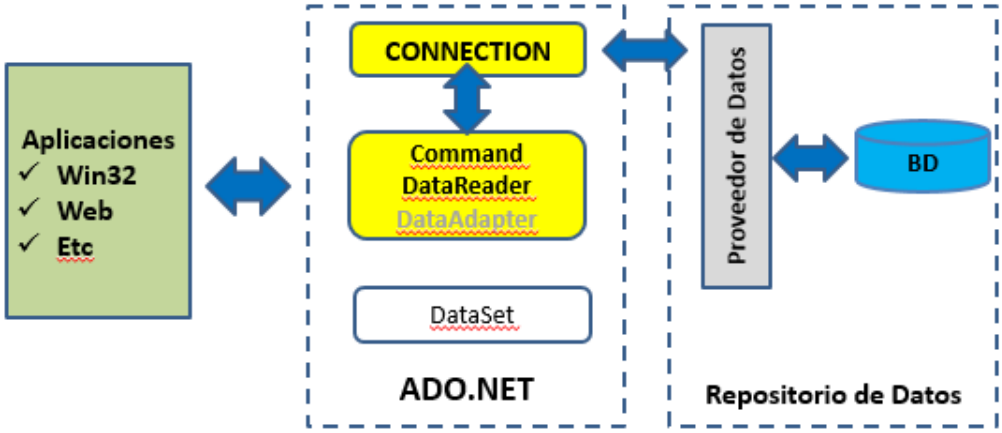
Esto último es simple de comprobar, pues si está conectado todo el tiempo las estrategias de sincronización que ofrece el objeto evita el conflicto que suscita la persistencia simultánea de datos. Como está siempre conectado, los datos van a estar actualizados en todo momento.

A continuación, detallaremos el uso de los objetos que componen el modelo ADO. Básicamente podemos hablar de tres: **sqlConnection** que se encarga de conectarse a la base de datos, el **sqlCommand** que ejecuta la acción sobre la base de datos sea leer, borrar, modificar o consultar. El **sqldataReader** que representa la tabla en nuestra aplicación.

Analizaremos cada uno.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Un esquema del modelo conectado



Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

2.1. Objeto SqlConection

El objeto **SqlConection** se encuentra dentro del espacio de nombre **System.data.SqlClient** y permite la conexión a la base de datos. Necesita lo que llamamos string de conexión, es decir, una estructura de datos que contiene toda la información que usará el proveedor de datos para conectarse a la base de datos

Un string de conexión puede obtenerse por distintos caminos, nosotros le presentamos uno de ellos.

```
Data Source=[servidor];Initial Catalog=[Nombre de BD];Integrated
Security=True;
```

No hay duda que este es el objeto que más se necesita si vamos a utilizar el modo conectado ya que determina la conexión. Luego, el resto de los objetos usarán la conexión para realizar sus tareas.

Para trabajar en el modo conectado, utilizaremos básicamente una propiedad y dos métodos del objeto - aunque no son los únicos.

- La propiedad es **ConnectionString**, a través de ella se le carga el string de conexión que detallamos anteriormente.

- El método **Open** es la acción que dispara el objeto para conectarse a la base de datos. Tiene un constructor sobrecargado, por eso se puede agregar como argumento el string de conexión.

- El método **Close**, muy importante porque debemos cerrar la conexión de la base de datos cuando ya no la usamos para liberar el recurso y compartirlo con otros usuarios.

Un ejemplo del uso de este objeto es el siguiente:

```
Using System.Data;
Using System.Data.SqlClient;

SqlConnection oCn = new SqlConnection();
      oCn.ConnectionString = "……";
      oCn.Open();

        (……)
      oCn.Close();
```


This document is available free of charge on studocu

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

En muchos textos referidos a este entorno de programación podemos encontrar la declaración de la variable y la creación de la instancia en la misma línea:

```
SqlConnection oCn = new SqlConnection();
```

Como mencionamos, se pueden utilizar el constructor sobrecargado del objeto Connection y agregar el ConnectionString.

```
SqlConnection oCn = new SqlConnection(@"Data Source=.\SQL_UAI; Initial Catalog=base;
Integrated Security=True");
```

Aunque el efecto es el mismo, recomendamos crear la instancia de la clase lo más tarde posible y cerrarla lo más temprano posible. Esta afirmación responde a un saber informal que se puede explicar técnicamente pero para simplificarlo, diremos que la intención es reservar los recursos de la computadora la menor cantidad de tiempo posible.

Se puede usar otra propiedad importante llamada **state** que nos indica el estado de la conexión: *conecting*, *closed*, *open* entre otras.

```
if (oCn.State == ConnectionState.Connecting)
        { oCn.Close();}
```

Otra manera se puede utilizar en métodos para abrir y cerrar la instancia de conexión a la BD.

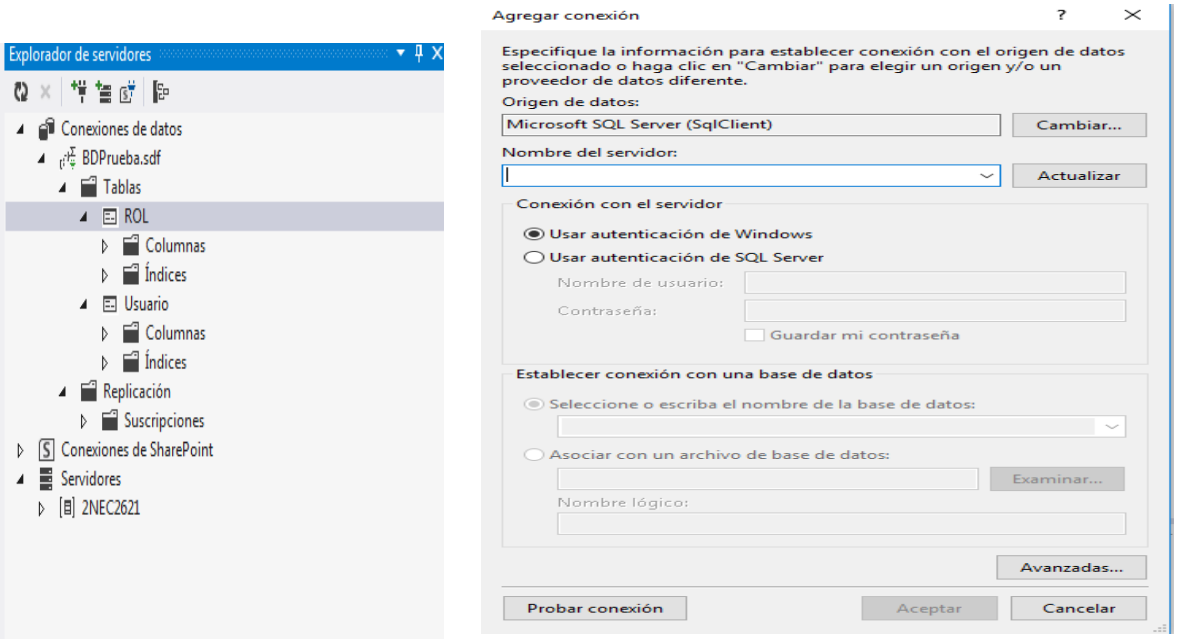
```
private SqlConnection conexion;

       public void Abrir()
      {
         conexion = new SqlConnection();
         conexion.ConnectionString = @"Data Source=.\SQL_UAI; Initial Catalog=base;
Integrated Security=True";
         conexion.Open();
      }
      public void Cerrar()
      {
         conexion.Close();
         conexion.Dispose();
         conexion = null;
         GC.Collect();
      }
```


Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

¿Cómo obtengo el ConnectionString?

Los pasos a seguir para obtenerlo dentro del framework son los siguientes:

Paso 1) Menú: Herramientas → Conectar con la BD

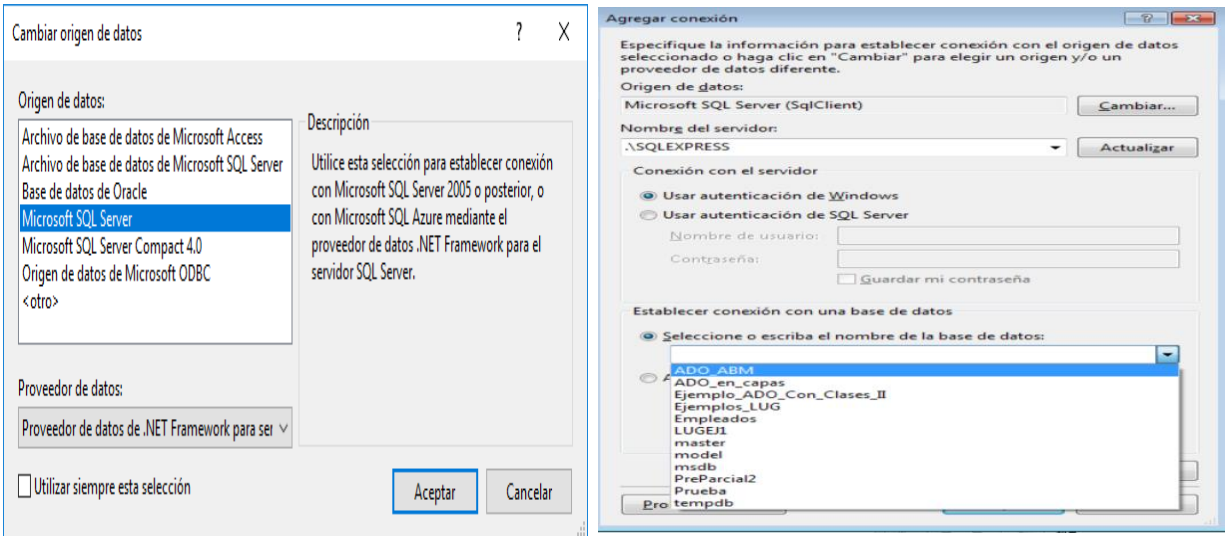


Paso 2) Seleccionar el Origen de datos, server y BD

Origen de datos: la base de datos que se va utilizar

Nombre del servidor: El nombre del servidor, en este caso es SQL, y el nombre es .\SQLEXPRESS

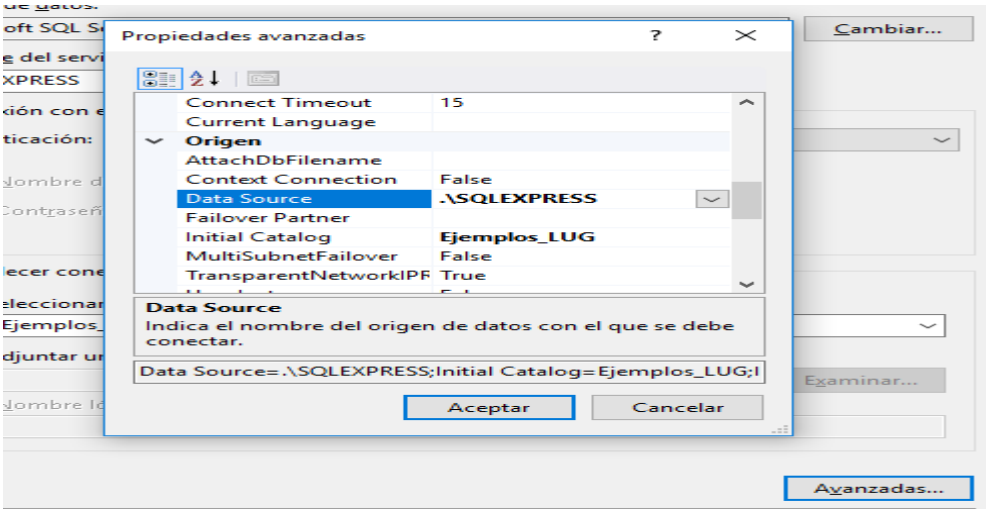
El nombre de la BD, les va a traer todas las BD que se encuentren en el servidor.



This document is available free of charge on studocu

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Paso 3) La opción de Avanzadas



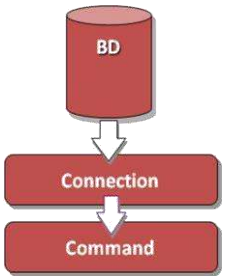
2.2. Objeto sqlCommand

Este objeto se complementa con el objeto de conexión y tiene la responsabilidad de ejecutar todas las órdenes necesarias para tomar los datos de la base. Puede hacer operaciones de agregar, borrar, modificar o seleccionar.

Generalmente se usa con sintaxis SQL pero también puede ejecutar, directamente, procedimientos almacenados.

Los **procedimientos almacenados** son funciones precompiladas en la base de datos. Se utilizan con más frecuencia que las órdenes de SQL porque mitigan las amenazas de ataques por SQL injection.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)



Para ejecutar un **command** debe respetar los siguientes pasos:

Para ejecutar un **command** debe respetar los siguientes pasos:

1- Relacione el comando con la conexión
2- Especifique el tipo de comando (texto, procedimiento almacenado)
3- Facilitar el comando a ejecutar

Los pasos anteriores expresados en código serían los siguientes.

```
SqlCommand oCmd = new SqlCommand();
      oCmd.Connection = oCn;
      oCmd.CommandType = CommandType.Text:
      oCmd.CommandText = "select * from Pintor";
```

Luego, debemos ejecutar el comando. La ejecución puede ser de tres formas diferentes:

1) Ejecutar el comando y devolver un conjunto de datos
2) Ejecutar el comando y devolver un escalar
3) Ejecutar el comando y no devolver nada

```
1. oCmd.ExecuteReader();
2. oCmd.ExecuteScalar();
3. oCmd.ExecuteNonQuery();
```

La decisión de elegir alguno de los tres está directamente ligada a la estrategia de recuperación de datos que decida usted como programador.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Un ejemplo completo se podría ver como el siguiente:

```
SqlConnection oCn = new SqlConnection();
oCn.ConnectionString = "…";
oCn.Open():

SqlCommand() oCmd = new SqlCommand();
oCmd.Connection = oCn;
oCmd.CommandType = CommandType.Text;
oCmd.CommandText = "insert into pintor value(1,'Angela','Fernandez')";

oCmd.ExecuteNonQuery();
(…)
if (oCn.State == ConnectionState.Connecting)
    {       oCn.Close(); }
```

Cuando utilizamos consultas de SQL o llamadas a procedimientos almacenados, solemos necesitar **parámetros**.

El comando posee una lista de parámetros llamada **Parameters**. El uso es similar a una lista en la medida en que se agregan o se sacan parámetros de la misma manera.

Supongamos que debemos insertar un nuevo registro en la tabla pintor, el código sería:

```
SqlCommand oCmd = new SqlCommand();
oCmd.Connection = oCn;
oCmd.CommandType = CommandType.Text;
oCmd.CommandText = "insert into pintor
values(@Codigo,@Nombre,@Apellido)";
SqlParameter pCodigo = new SqlParameter();
pCodigo.ParameterName = "@Codigo";
pCodigo.Value = 1;
oCmd.Parameters.Add(pCodigo);
(reptirlo para cada parametro)

oCmd.ExecuteScalar();
```

Cuando trabajamos con procedimientos almacenados, crear la lista de parámetros puede ser una tarea engorrosa. Existe un comando que

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

permite recrear la lista de parámetros automáticamente tomando la referencia del procedimiento almacenado.

```
oCmd.Connection = oCnn;
oCmd.CommandType = CommandType.StoredProcedure;
oCmd.CommandText = SP;
SqlCommandBuilder.DeriveParameters(oCmd);
```

El objeto **sqlComandoBuilder** recupera los parámetros del procedimiento almacenado y genera en forma automática la lista de parámetros en el comando.

Actividades para la facilitación de los aprendizajes

Actividad 2: Creando y ejecutando procedimientos almacenados

Tome las consultas de la **Actividad 1** y creelas como procedimientos almacenados.

Si tiene dudas sobre la resolución de este ejercicio, consulte a su tutor.

2.3. Objeto sqlDataReader

El objeto dataReader representa una tabla y como usted sabe, es la única manera para usar tablas en un programa.

Si bien es muy rápido tiene algunas limitaciones, entre ellas, puede recorrerse en una única dirección. Es decir que si llegamos al registro 5 y necesitamos leer el registro 4 estamos obligados a volver al registro 0 y comenzar otra vez.

El objeto no es más que una colección que se recorre registro por registro. Al igual que la lista en otros lenguajes tiene un método llamado **Read** que lee el registro y pasa al puntero siguiente.

This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

```
SqlDataReader  oDr= cmd.ExecuteReader();
while (oDr.Read())
{   ….
}
oDr.Close();
```

El **dataReader** necesita del **SqlComando**, es decir, al ejecutar el método **ExecuteReader** del objeto **sqlComand** obtenemos un objeto **dataReader**.

La lectura de valores en un Data Reader es por su ubicación en la colección o por su nombre

```
dr.item[“Nombre”];
```

Un ejemplo sencillo lo vemos en el siguiente ejemplo:

```
SqlConnection  oCn = new System.Data.SqlClient.SqlConnection();
oCn.ConnectionString = "…";
oCn.Open();

 SqlCommand oCmd = new System.Data.SqlClient.SqlCommand();
 oCmd.Connection = oCn;
 oCmd.CommandType = CommandType.Text;
 oCmd.CommandText = "select * from pintor" ;
 SqlDataReader oDr   = oCmd.ExecuteScalar();
 If (oCn.State == ConnectionState.Connecting)
            { oCn.Close(); }
```

Se puede crear una clase que permita el uso de objetos, de esa forma lograríamos un poder de abstracción en las aplicaciones para darle más escalabilidad.

```
public class SQLSERVERX
{
   private SqlConnection oCnn;
   public int Ejecutar(string SP, params string[] Datos)
   {
      sqlDataReader rs = new sqlDataReader();
      SqlCommand oCmd = new SqlCommand();
      SqlDataAdapter da = new SqlDataAdapter();
      int Errores = 0;

      int i;
      try
      {
         oCmd.Transaction = TX;
         oCmd.Connection = oCnn;
         oCmd.CommandType = CommandType.StoredProcedure;
oCmd.CommandText = SP;
```

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

```
            SqlCommandBuilder.DeriveParameters(oCmd);
            i = 0;
            bool ok = true;
            if (Datos[0].Trim() != "" && Datos.Length - 1 >= 0)
            {
                while (ok && i <= Datos.Length - 1)
                {
                    oCmd.Parameters(i + 1).Value = Datos[i];
                    i = i + 1;
                }
            }
            Errores = oCmd.ExecuteReader();
        }
        catch (SqlException ex)
        {
            VolverTrasaccion();
            Cerrarconexion();
            throw new Exception(ex.Message.Trim);
            Errores = -1;
        }
        catch (Exception ex)
        {
            VolverTrasaccion();
            Cerrarconexion();
            throw new Exception(ex.Message.Trim());
            Errores = -1;
        }

        da.Dispose();
        oCmd.Dispose();
        return Errores;
    }
```

Esta actividad puede resultar útil para promover los aprendizajes esperados y para que usted evalúe la comprensión de los contenidos desarrollados hasta ahora.

Actividades para la facilitación de los aprendizajes

Le proponemos llenar un **combobox** con los datos obtenidos de un **datareader**

Esto le permitirá desarrollar clase de utilidades que conformarán capas de datos en la arquitectura futura de sus aplicaciones.

This document is available free of charge on studocu

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Actividad 3

También, le proponemos, la realización de la siguiente ejercitación:

Ejercicio 1: Con la Base de Datos creada en la actividad anterior cree un formulario MDI, que contenga un formulario para socios, otro para país y otro para Provincias. Debe mostrar el contenido de la tabla país y el ABM de la misma.

Ejercicio 2: Debe realizar un sistema que permita administrar todos los productos de una empresa, y las categorías de los mismos. Toda la información debe persistir en Base de Datos.

Ejercicio 3: En el ejercicio anterior por cada operación que impacte en el sistema (ABM, Loguin, logout) cree una tabla más llamada Log, la cual debe contener Id, Fecha, operación y Usuario. Registre cada operación.

Cree un formulario que muestre las actividades realizadas

Puede realizar los ejercicios con query embebidas y también con procedimientos almacenados.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Si tiene dudas o dificultades, por favor, comuníquese con pares y/o tutor. Entre todos/as intentaremos resolverla.

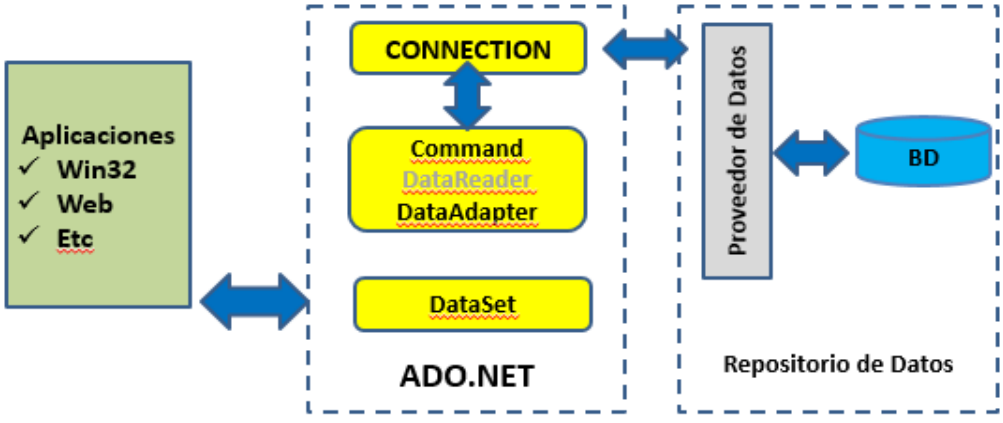
3. Modo desconectado

Nos encontramos en un entorno desconectado cuando el modelo de datos está conectado para obtener los datos y luego se puede desconectar sin perder los datos.

Uno de los principales problemas que tiene es la **sincronización** de los datos, puesto que si alguien cambió los datos mientras se estaba usando no sabremos cual es el valor definitivo.

Es decir, al trabajar desconectado dos usuarios se llevan una copia de los datos, hacen modificaciones por separado y luego deben actualizar la base de datos. La pregunta que seguramente se harán es qué pasa si un usuario cambia el precio de $10 a $12 y otro hace un cambio de $10 a $14, ¿cuál es el valor final? En esta asignatura no trabajaremos en la resolución de este tipo de problemas pero debe usted saber que este se presenta cuando trabajamos en un entorno desconectado.

Esquema modo Desconectado



This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

3.1. Objeto DataSet

El objeto **DataSet** no forma parte del modelo ADO.net pero se complementan ya que puede funcionar como una base de datos temporal. Es una evolución del viejo *recordset* de la versión anterior incorporando la posibilidad de contener varias tablas y relaciones entre dichas tablas.

Contiene **datatables** que son tablas vinculadas al modelo de datos y **dataRelation** que son las relaciones entre las tablas.

A su vez, el **datable** está formado por **datarows** y estas a su vez por **datacolumns**.

La ventaja del **DatSet** es que está formado por colecciones enlazadas que se pueden recorrer fácilmente con la estructura **foreach**.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

```
foreach (DataRow item in Ds.tables[0].Rows)
{ .. }
```

El **DataSet** que funciona desconectado puede crearse como un modelo desvinculado de la base de datos.

```
DataSet Ds = new DataSet();
DataTable Dt = new DataTable();
Dt.TableName=”Persona”;
//Creo la columna Id
   DataColumn ColId ;

//Completo la columna Id
      ColId = new DataColumn("ID", typeof(int));

//Creo la columna Nombre
      DataColumn ColNombre = new As DataColumn();
//Completo la columna ColNombre
ColNombre.ColumnName =”Nombre”;
ColNombre.DataType= typeof(string);
COlNombre.MaxLength = 50;

//Agrego las columnas a la tabla
      Dt.Columns.Add(ColId);
      Dt.Columns.Add(ColNombre);

//Creo una fila con los datos en las columnas
      DataRow Dr;
      Dr = Dt.NewRow();
      Dr["Id"] = 1:
      Dr["Nombre"] = "Juan";
      Dt.Rows.Add(Dr);

      Dr = Dt.NewRow();
      Dr["ID"] = 2:
      Dr["Nombre"] = "Pedro";
      Dt.Rows.Add(Dr);

//Determino la clave primaria

    Dt.PrimaryKey = new DataColumn[] { Id };
    Ds.Tables.Add(Dt);

      this.cmbDataset.DataSource = Ds.Tables[0];
      this.cmbDataset.DisplayMember = "Nombre" ;
      this.cmbDataset.ValueMember = "Id";
```


This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Notemos que las últimas tres líneas muestran como desde un **Dataset** se puede popular un combobox sin necesidad de recorrer toda la tabla. Algo similar se puede lograr si utilizamos una Datagrid.

Resumiendo, los pasos para crear un dataset sin una base de datos son:

Resumiendo, los pasos para crear un **Dataset** sin una base de datos son:

1- Crear un objeto **Dataset**
2- Crear un **Datatable**
3- Crear el o los objetos **datacolumn** que se necesiten determinando el nombre del campo y el tipo de datos.
4- Agregar al **Datatable**
5- Llenar la colección de claves de la tabla con los campos que la componen
6- Agregar el objeto **Datatable** al **DataSet**
7- De existir, crear todas las relaciones

3.2. Objeto sqlDataAdapter

Este objeto es el nexo entre el modelo ADO.net con el objeto **Dataset**. Y por supuesto hay un tipo **DataAdapter** para cada proveedor de datos. si bien es frecuente nuestra referencia al objeto sqlDataAdapter existe también un **oldbdataadapter**Seguramente, en otras asignaturas habrá estudiado acerca de los constructores de los objetos. ¿Recuerdar el método **new**?. El **dataAdapter**, tiene una sobrecarga en su constructor, entre sus opciones puede no tener argumentos o agregar argumentos. Veremos cómo aprovechar los comandos en modo desconectado en los siguientes ejemplos.

Una rutina donde utilizamos el **SqlDataAdapter** para llenar un **dataset**

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

```
DataSet Ds = new DataSet();
SqlCommand oCmd = new SqlCommand():
SqlDataAdapter oDa = new SqlDataAdapter();
oCmd.Transaction = TX;
oCmd.Connection = oCnn;
oCmd.CommandType = CommandType.Text;
oCmd.CommandText = Query;
oDa.SelectCommand = oCmd;
oDa.Fill(Ds);
oDa.Dispose();
oCmd.Dispose();
Return Ds;
```

Otro ejemplo

```
DataSet ds = new DataSet();
SqlCommand oCmd = new SqlCommand();
oCmd.Transaction = TX;
oCmd.Connection = oCnn;
oCmd.CommandType = CommandType.Text;
oCmd.CommandText = Query;
ds = new DataSet();
SqlDataAdapter da = new SqlDataAdapter(oCmd);
//con argumentos
da.Fill(ds);
da.Dispose();
oCmd.Dispose();
return ds;
```

En el código, las líneas nuevas e importantes son:

```
SqlDataAdapter oDa = new SqlDataAdapter();
oDa.SelectCommand = oCmd;
```

Crea el **sqldataadapter** y lo relaciona con el comando. Esto mismo se podría escribir de la misma manera pero usando el constructor sobrecargado del objeto.

```
sqlDataAdapter Da = new sqlDataAdapter(Ocmd);
```


This document is available free of charge on studocu
Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Otra propuesta de trabajo.

Actividades para la facilitación de los aprendizajes

Actividad 4

Ejercicio 1: Llene un **datagrid** con los datos obtenidos de una **dataset**

Ejercicio 2: A los ejercicios de la **actividad 3** agréguele transacciones y que al realizar los ABM muestre la actualización de los datos en un datagrid.

Esto le permitirá desarrollar clase de utilidades que conformarán capas de datos en la arquitectura futura de sus aplicaciones

Conclusión

No existe aplicación empresarial que reniegue de una base de datos, la reinversión del conocimiento utilizado en la organización es indispensable como insumo a la hora de tomar decisiones. No utilizar una base de datos impediría el rendimiento de la empresa en un mercado competitivo como el actual.

ADO.net es un servicio que brinda Visual Studio para facilitar esta tarea y reutilizar todos los elementos persistidos en la base de datos.

Le solicitamos que profundice sus conocimientos acerca de ADO.net, con la lectura del siguiente material.

Lectura requerida

- Deitel, Harvey M.; Deitel, Paul J.; Vidal Romero Elizondo, Alfonso(Traductor); y otros. **Cómo programar en C#.** 2a.ed.-- México, DF. Capítulo 20.

Cierre de la unidad

A modo de cierre de la unidad y de inicio de futuros debates, lo/a invitamos a participar en el siguiente espacio de intercambio y discusión.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

Trabajo Colaborativo

Debate sobre los pilares fundamentales de la programación orientada a objetos

This document is available free of charge on studocu

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)