Trabajar en modo desconectado

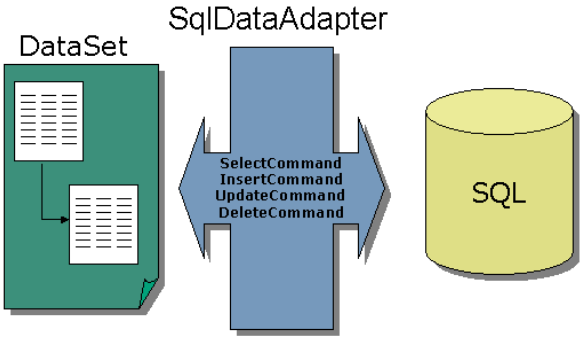
Recuerda que tenemos dos modos de trabajo:

- Trabajar cargando un conjunto de datos en memoria *(Clases desconectadas)*
- Trabajar directamente con la base de datos *(Clases conectadas)*.

Recordamos asimismo que en modo desconectado:

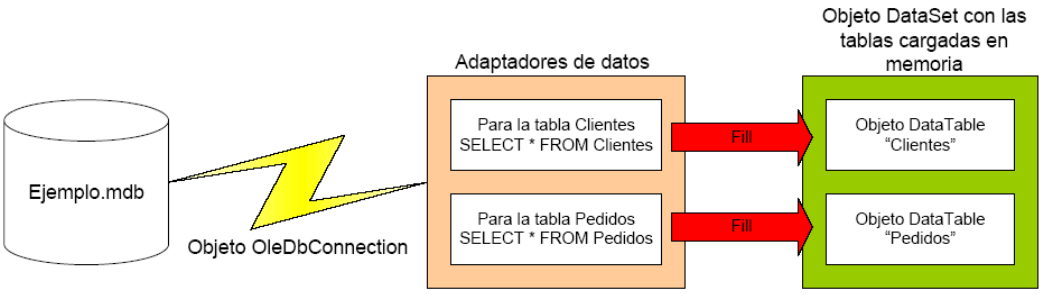
- La conexión sólo es necesario establecerla cuando se descarga información del origen de datos.
- El método **Fill** se encargará de cargar una tabla en el DataSet.
- Cada tabla del DataSet precisa de un objeto DataAdapter.
- Los datos almacenados en el DataSet se pueden modificar.
- El método **Update** del objeto DataAdapter permite actualizar el origen de datos.
- La información entre el cliente y el servidor se transmite en forma de datos XML

En modo desconectado los adaptadores de datos se utilizan para relacionar una conexión con un conjunto de datos (DataSet):



- Adapta los datos del formato nativo del gestor de bases de datos para que puedan ser utilizados en el *DataSet* (XML).
  - Carga las diferentes tablas en el *DataSet.*
  - Actualiza las modificaciones del *DataSet* en el origen de datos.

- Normalmente se utilizará un adaptador de datos por cada tabla que queramos recuperar del origen de datos.



En función del proveedor de datos con el que vayamos a trabajar, tenemos un determinado objeto DataAdapter:

Proveedor	Objeto DataAdapter
OLE DB	OleDbDataAdapter
SQL Server	SqlDataAdapter
ODBC	OdbcDataAdapter
Oracle	OracleDataAdapter

Cuando utilizamos objetos DataAdapter para intercambiar datos entre un objeto DataSet y una fuente de datos, podemos especificar las acciones que deseamos realizar utilizando una de las cuatro propiedades del DataAdapter. Estas propiedades ejecutan una instrucción SQL o invocan un procedimiento almacenado:

Propiedad	Descripción
SelectCommand	Su función es recoger la información requerida de una base de datos para poblar o llenar un DataSet.
UpdateCommand, DeleteCommand, InsertCommand	Funciones de mantenimiento de la base de datos, en función de qué tipo de mantenimientos estemos realizando: (actualizaciones, insertar registros, eliminar registros, utilizaremos la propiedad determinada).

- La propiedad SelectCommand del DataAdapter es un objeto Command que recupera datos del origen de datos.

- Las propiedades InsertCommand, UpdateCommand y DeleteCommand de DataAdapter son objetos Command que permiten administrar las actualizaciones de los datos en el origen de datos para reflejar las modificaciones efectuadas en el DataSet. *Estas propiedades las describiremos más detalladamente cuando veamos cómo Actualizar orígenes de datos con DataAdapters.*

Cargar las tablas desde el origen de datos en el DataSet

El método Fill ejecuta implícitamente una consulta SQL en la propiedad SelectCommand del objeto DataAdapter. Los resultados se utilizan para definir la estructura del objeto DataTable y poblar la tabla con datos. El procedimiento habitual será:

1. Crear la instancia del adaptador de datos:

```
Dim nombreAdaptador As xxxDataAdapter
nombreAdaptador = New xxxDataAdapter(selectSQL, cn)
```

- ✓ xxxDataAdapter es alguna de las clases SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter y OracleDataAdapter.
- ✓ selectSQL es una cadena con la instrucción SQL que recuperará los datos de la tabla que se añadirá al *DataSet*.
- ✓ cn es un objeto Connection.

2. Rellenar el conjunto de datos
   - El adaptador de datos permite cargar las tablas recuperadas a partir de la sentencia SQL en objetos de tipo *DataTable* del conjunto de datos.
   - El método *Fill* permite realizar la carga de datos:

     objetoDataAdapter.Fill(objetoDataSet,nombreObjetoDataTable)

     ✓ Precisa la existencia de un objeto de la clase DataSet.
     ✓ *nombreObjetoDataTable* es una expresión de cadena que identificará la tabla dentro del conjunto de datos.
     ✓ Los datos que se cargarán en la tabla serán los que se recuperen mediante la orden SQL del constructor del adaptador de datos.

El siguiente código muestra cómo crear un objeto SqlDataAdapter, y a continuación invoca al método Fill para almacenar los datos en el objeto DataSet:

```
'Crear la conexión
Dim cn As New SqlConnection _
("data source=localhost;initial catalog=pubs;" & _
"integrated security=SSPI;persist security info=True;")

'Crear el DataSet
Dim ds As New DataSet()

'Crear el DataAdapter
Dim da As New SqlDataAdapter ("select * from Authors", cn)

'Llenar el DataSet
da.Fill(ds, "Authors")

'Se puede utilizar y manipular la información de la tabla sin conexión
…….
……
'Cuando los datos necesiten ser actualizados contra la fuente de datos, se utilizará
'el objeto DataAdapter para volver a conectarse y actualizar la fuente de datos
da.Update(ds, "Authors")
```

Atención!, el código no abre o cierra el objeto Connection de manera explícita. El método Fill abre de forma implícita el objeto Connection que utiliza el DataAdapter cuando determina que esta conexión no está abierta. Si el método Fill ha abierto la conexión, el mismo método la cierra cuando termina de utilizarla. Este hecho simplifica el código cuando se trata de una operación única, como un método Fill o Update. Sin embargo, en el caso de que se estén realizando varias operaciones que necesiten tener abierta una conexión, puede mejorar el rendimiento de la aplicación llamar explícitamente al método Open de Connection, realizar a continuación las operaciones en el origen de datos y, finalmente, llamar al método Close.

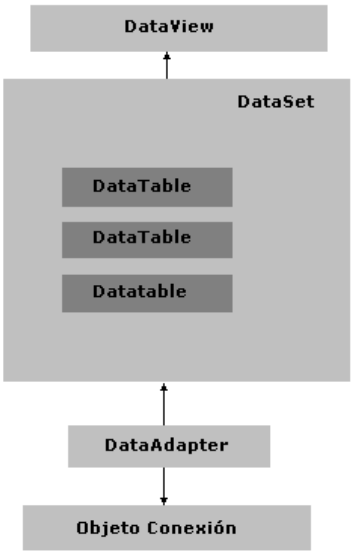
Es conveniente mantener abiertas las conexiones del origen de datos lo más brevemente posible con el fin de liberar recursos, de manera que estén disponibles para otras aplicaciones cliente.

La clase DataSet

Almacena en la memoria caché del cliente los resultados de la consulta SQL establecida en un adaptador de datos.

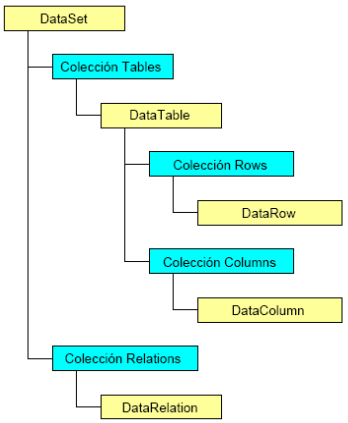
- ✓ Los datos están disponibles en modo desconectado.
- ✓ Forman una pequeña base de datos con la información necesaria para la aplicación.

En la siguiente figura puedes ver el modelo de objetos de un DataSet:



Esta figura nos muestra cómo un DataSet es una copia en memoria de nuestro modelo de datos, por lo tanto tenemos tablas, con sus filas y columnas, relaciones y propiedades extendidas.

- ➢ Tables. Se compone de un objeto DataTableCollection el cual puede formarse por una o más DataTable. Un objeto DataTable representa una de las tablas de nuestra fuente de datos. Un DataTable está formado a su vez por colleciones de filas (Rows) y columnas (Columns) que podrán contener una o más filas (DataRow) o columnas (DataColumn).
- ➢ Relaciones. Se trata de colecciones de relaciones que pueden asociar tablas con claves foráneas.
- ➢ Propiedades extendidas. Hace referencia a colecciones de propiedades que pueden ser asignadas al DataSet en el momento de ser creado.



Tenemos tres modos diferentes de trabajar con los objetos DataSet:

- *Crear objetos DataTable dentro del DataSet para almacenar los datos mediante programación.*
- *Recoger la información de una base de datos, de modo que llenemos el DataSet con estos datos mediante un objeto "puente" entre el DataSet y la base de datos que recibe el nombre de DataAdapter.*
- *Trabajar con la información mediante fuentes de datos XML.*

Accediendo a los datos contenidos en el DataSet:

Una vez insertados los datos en un objeto DataSet, podemos acceder mediante código a los datos:

- Cada objeto DataSet está formado por uno o más objetos DataTable a los que podemos hacer referencia por su nombre ***ds.Tables("Clientes")*** o posición ordinal ***ds.Tables(0)***.

- Las clases DataRow y DataColumn son los componentes principales de una clase DataTable. Utilizaríamos un objeto DataRow con sus propiedades y métodos para recuperar y evaluar los valores de un objeto DataTable.

- DataRowCollection representa los objetos DataRow reales que se encuentran en el objeto DataTable, y DataColumnCollection contiene los objetos DataColumn que describen el esquema del objeto DataTable. La ***propiedad Rows*** del objeto DataTable proporciona acceso por código a DataRowCollection. La ***propiedad Columns*** del objeto DataTable proporciona acceso programático a DataColumnCollection.

  El siguiente código de ejemplo agrega los nombres de columnas de un objeto DataSet al control ListBox denominado lista:

```
Dim col As DataColumn
For Each col In ds.Tables(0).Columns
        lista.Items.Add(col.ColumnName)
Next
```

- Tanto el objeto DataRowCollection como el objeto DataColumnCollection tienen una ***propiedad Count*** que nos permite determinar el número de filas o columnas de un objeto DataTable, como muestra el siguiente código de ejemplo:

```
ds.Tables("Authors").Rows.Count
ds.Tables("Authors").Columns.Count
```

- Contar las filas y columnas del objeto DataTable nos permite acceder a campos individuales del objeto DataTable. Podemos acceder a campos por posición ordinal (basada en 0) o por el nombre. En el siguiente código, x es el índice de la fila de datos a la que deseamos acceder:

```
DataSet.Tables(0).Rows(x)(1)
DataSet.Tables(0).Rows(x)("fieldname")
```

Llenar un DataSet desde múltiples DataAdapter:

- Se puede utilizar cualquier cantidad de objetos DataAdapter con un DataSet.
- Cada DataAdapter se puede utilizar para llenar uno o varios objetos DataTable y para reflejar en el origen de datos correspondiente las actualizaciones que sean necesarias.
- Se pueden agregar de forma local objetos DataRelation y Constraint al DataSet, de manera que se pueden relacionar datos procedentes de varios orígenes distintos.

Por ejemplo, un DataSet puede contener datos de una base de datos de Microsoft SQL Server, una base de datos de IBM DB2 expuesta mediante OLE DB y un origen de datos que genera secuencias XML. Para ocuparse de la comunicación con cada origen de datos se pueden usar uno o varios objetos DataAdapter.

El siguiente ejemplo carga en el mismo DataSet dos tablas procedentes de una misma bases de datos:

```
'Establece los espacios de nombre
Imports System.Data
Imports System.Data.OleDb
...
'Declaración de variables. Dependiendo de su alcance pueden llevar otros modificadores
Dim cn As New OleDbConnection
Dim daClientes As OleDbDataAdapter
Dim daPedidos As OleDbDataAdapter
Dim ds As New DataSet

'Cadena de conexión para el proveedor OLEDB para bases de datos Access
cn.ConnectionString = "Provider=Microsoft.Jet.OleDB.4.0;" & _
"Data Source=c:\BBDD\Ejemplo.mdb"

'Crear los adaptadores de datos
daClientes = New OleDbDataAdapter("SELECT * FROM Clientes", cn)
daPedidos = New OleDbDataAdapter("SELECT * FROM Pedidos", cn)

'Rellenar el Dataset
daClientes.Fill(ds, "Clientes")
daPedidos.Fill(ds, "Pedidos")
```

Y el siguiente, carga en el mismo dataset, dos tablas procedentes de distintas bases de datos: se llena una lista de clientes a partir de la base de datos Northwind de Microsoft SQL Server 2000 y una lista de pedidos a partir de la base de datos Northwind almacenada en Microsoft Access 2000. Además, las tablas de datos llenas se relacionan entre sí mediante DataRelation, con lo que se puede mostrar una lista de clientes con los pedidos que ha realizado cada uno. Y todo ello desconectado del origen de datos:

```
Dim custAdapter As SqlDataAdapter = New SqlDataAdapter( _
  "SELECT * FROM Customers", cn1)

Dim ordAdapter As OleDbDataAdapter = New OleDbDataAdapter( _
  "SELECT * FROM Orders", cn2)

Dim customerOrders As DataSet = New DataSet()
custAdapter.Fill(ds, "Clientes")
ordAdapter.Fill(ds, "Pedidos")

Dim relation As DataRelation = _
  customerOrders.Relations.Add("Clientes", _
  customerOrders.Tables("Clientes").Columns("CustomerID"), _
  customerOrders.Tables("Pedidos").Columns("CustomerID"))

Dim pRow, cRow As DataRow

For Each pRow In customerOrders.Tables("Clientes").Rows
  Console.WriteLine(pRow("CustomerID").ToString())

  For Each cRow In pRow.GetChildRows(relation)
    Console.WriteLine(vbTab & cRow("OrderID").ToString())
  Next
Next
```

Simplemente hemos añadido una relación a nuestro DataSet, indicando el nombre de la relación, la clave principal del DataTable "Clientes" y la clave foránea de la tabla "Pedidos". Recuerda que aquí se utilizan los nombres de los DataTable que no tienen que coincidir con el nombre de las tablas en la base de datos, como es este caso. Pero es muy frecuente encontrar que los DataTable se nombran igual que las tablas de la base de datos. Aquí los hemos llamado de diferente modo para aclarar que no son las tablas de la base de datos con lo que estamos trabajando, sino con los DataTable.

Con los conocimientos adquiridos en la unidad donde hemos aprendido SQL, puedes hacerte la siguiente pregunta: si las tablas se encuentran relacionadas ya en una de las dos bases de datos (Northwind de Microsoft SQL Server o Northwind de Access), ¿por qué no puedo hacer una única consulta con INNER JOIN y llenar un único DataTable, ahorrando así la creación de relaciones?

Si te has hecho esta pregunta, enhorabuena, tienes toda la razón, podríamos haber hecho lo mismo con una única consulta aprovechando la relación existente entre las dos tablas en una de las dos bases de datos. Hemos realizado el ejercicio de este modo para que puedas comprobar cómo enlazar DataTables en un DataSet. Ahora imagina, que quieres hacer la relación entre dos tablas independientes tales que una de ellas te la han pasado en una base de datos X de SQL Server y otra en una base de datos Y de Access. No tienes posibilidad de hacer ninguna consulta con las dos tablas... Con ADO.net puedes crear esta relación en memoria tal y como se muestra en el ejemplo, incluso si las tablas pertenecen a distintas bases de datos, o más complicado todavía, si las tablas las obtenemos a partir de diferentes proveedores de Orígenes de datos.

Aquí queda demostrada plenamente la **independencia de los DataSet con el origen de datos**. Nos permite trabajar con diferentes proveedores: el abanico de posibilidades se abre considerablemente con esta característica.