UNIVERSIDAD ICESI

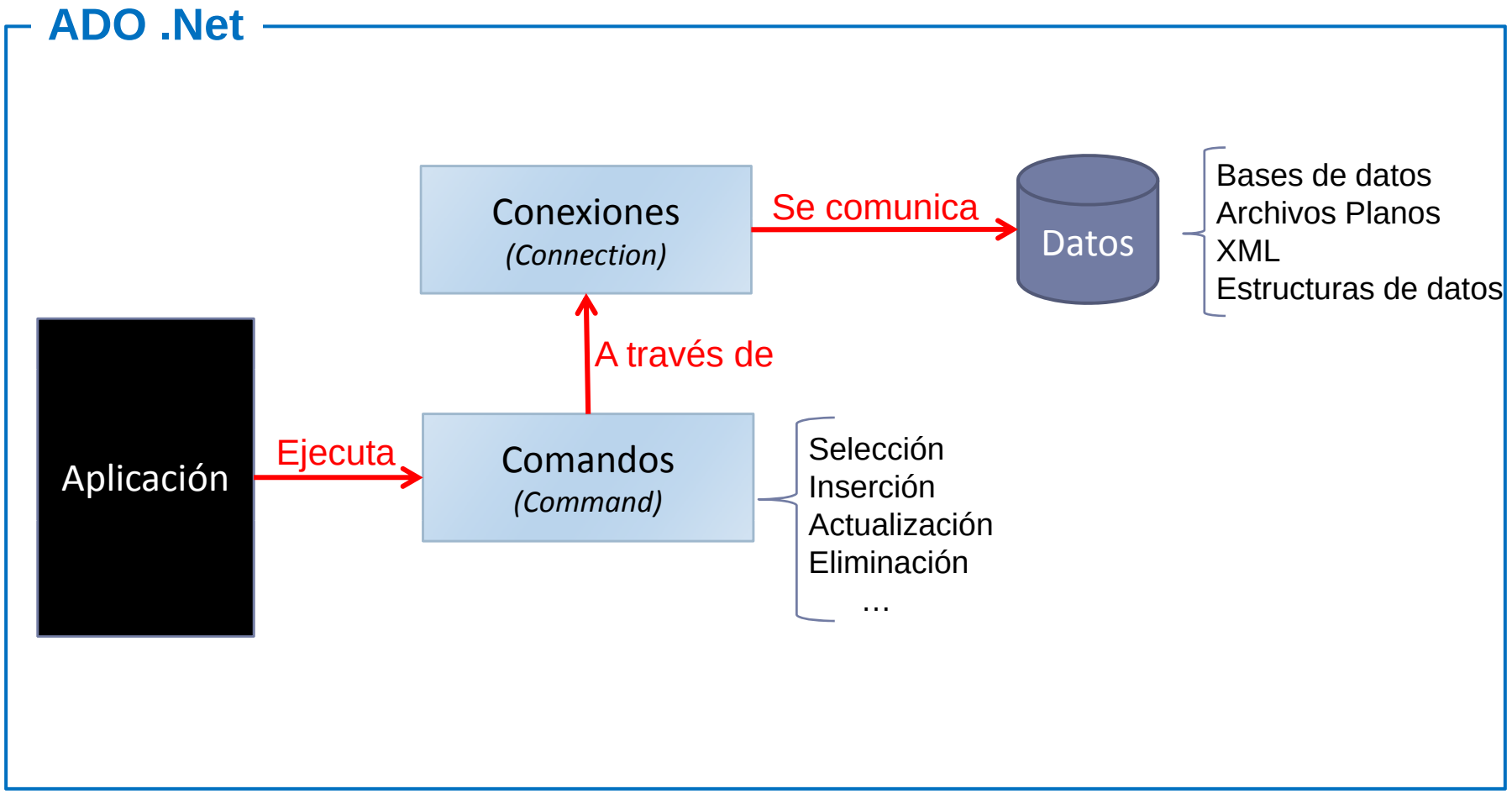
Acceso a Datos
ADO .NET

Laboratorio de Programación
Lorena Castañeda Bueno

Contenido

1. Acceso a Datos con ADO .Net
2. Objetos de ADO .Net
3. Modelo Desconectado

Estructura general



¿Qué es ADO.NET?

- Tecnología .Net para el acceso y manipulación de datos.
- Es especialmente diseñado para trabajar sobre ambientes desconectados.
- El espacio de nombres es System.Data

Objetos de Ado. Net (I)

- CONNECTION

Permite que se establezca un canal de comunicación con la fuente de datos

- COMMAND

Permite que se ejecuten las instrucciones sobre la fuente de datos

Las clases exactas de CONNECTION y COMMAND dependerán del proveedor de datos que se utilice y éste a su vez depende del tipo de fuente de datos a la que se quiera conectar

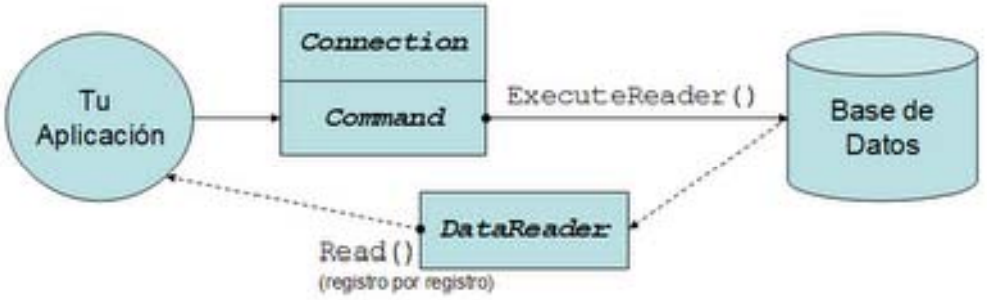
Sin importar cuál sea los pasos para utilizarlo siempre son los mismos:

- Abrir la conexión
- Ejecutas los comandos
- Cerrar la conexión

Objetos de Ado. Net (II)

Para leer los datos hay dos modos:

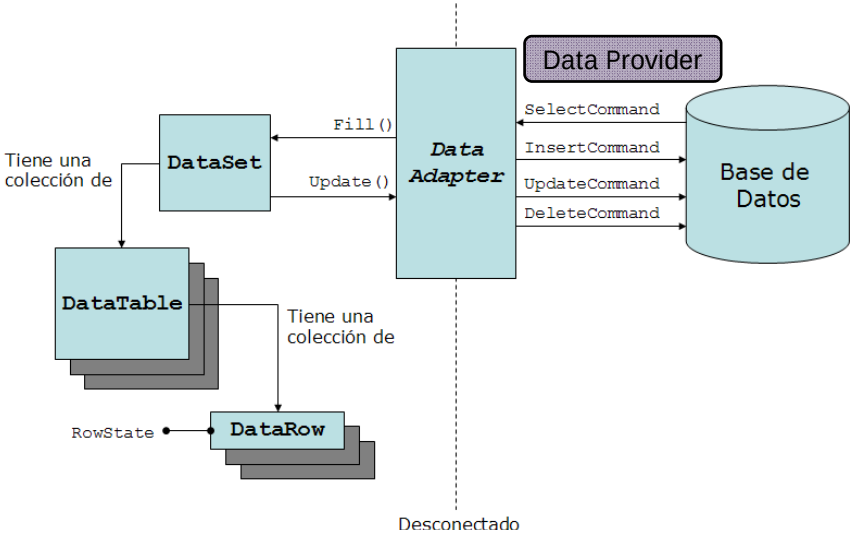
1. DATA READER
   - Es un objeto ligero y rápido que permite leer UN registro a la vez, de modo secuencial.
   - Es un objeto de solo lectura
   - No da información del tipo de dato que está leyendo, el desarrollador debe conocerlos y también el orden de las columnas



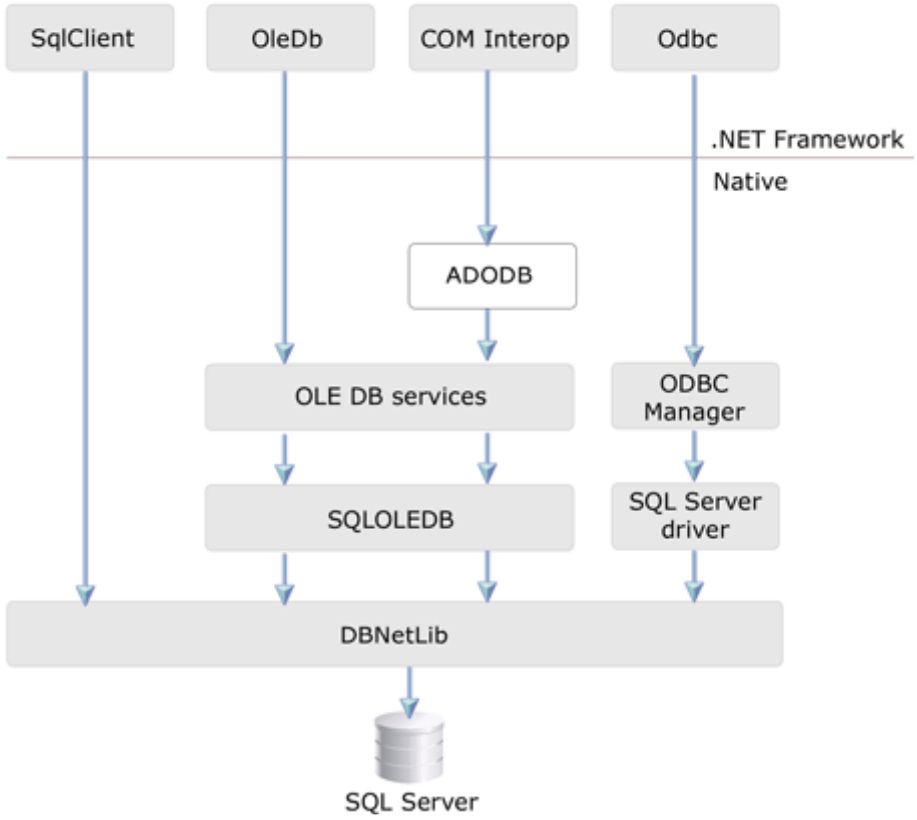
Objetos de Ado. Net (II)

2. DATA SET
   - Es un objeto que a su vez contiene otros sub-objetos que lo hacen más lento que el DataReader pero que tiene más flexibilidad para el uso de los datos.
   - El papel del DataAdapter es el de “sincronizador de datos” por medio de sus dos métodos principales, Fill() y Update().

- La forma de trabajar sería:
  1. Llenar el DataSet
  2. Realizar las acciones
  3. Actualizar al fuente de datos



Mejor rendimiento con el proveedor adecuado



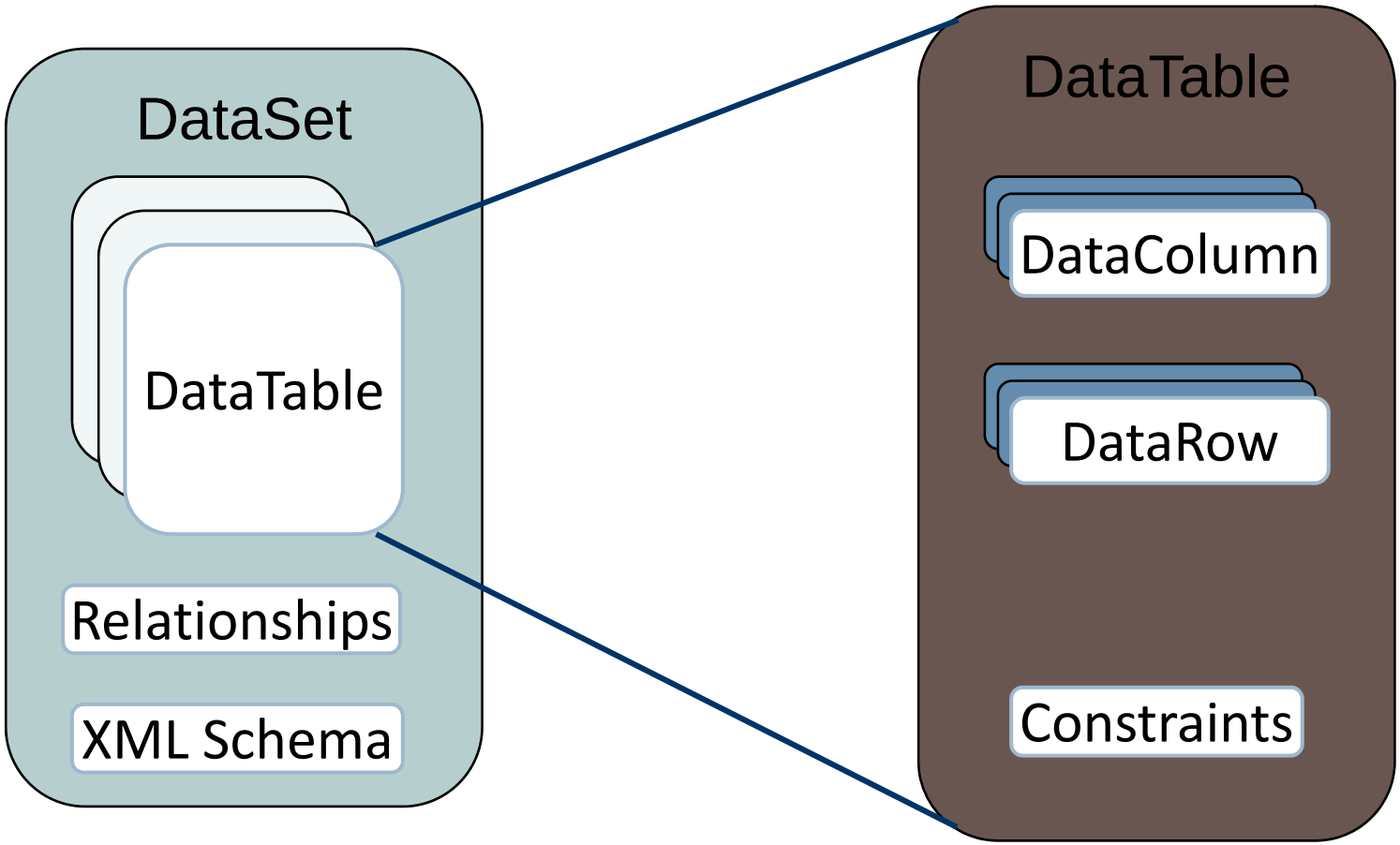
Dependiendo del proveedor se pueden requerir o no capas adicionales

Modelo desconectado

¿Qué es un DataSet?

- Es una representación de datos que reside en memoria.
- Un modelo de programación relacional coherente, independiente del origen de datos al que se encuentra asociado.
- Representa un conjunto completo de datos, incluyendo las tablas, relaciones y restricciones.
- Los cambios hechos en el DataSet no afectan inmediatamente la base de datos.
- Los datos pueden seguir siendo manipulados sin necesidad de tener la conexión establecida con la fuente de datos.
- Permite el ordenamiento, búsqueda y filtrado de datos de una forma sencilla.
- Permite trabajar con estructuras XML.

Representación gráfica



La Colección DataTable

- Cero o más tablas representadas por objetos DataTable.
- Cada DataTable contiene:
  - Una colección de columnas representadas por una DataColumnCollection.
  - Un conjunto de constraints representado en la ConstraintCollection.
  - Un conjunto de filas en la DataRowCollection, que representa los datos de la tabla.

DataColumns

```
// Crear un DataTable.
System.Data.DataTable tablaPadre= new DataTable("TablaPadre");
// Crear el objeto DataColumn para
DataColumn columna = new DataColumn();
// Configurar las propiedades de la columna
columna.DataType = System.Type.GetType("System.Int32");
columna.ColumnName = "id";
columna.ReadOnly = true;
// Propiedad Unique: si se duplican datos, se lanza ConstraintException
columna.Unique = true;
// Adicionar la columna a la DataColumnCollection.
tablaPadre.Columns.Add(column);
```

- Otras propiedades:
  - AllowDBNull– Si acepta o no valores nulos.
  - MaxLength– Tamaño de la columna.
  - Unique – Si se duplican datos, se lanza ConstraintException.

La colección de Constraints (Restricciones)

- PrimaryKey – Llave primaria.

```
// Convertir  la columna anterior en clave primaria.
DataColumn[] llavesPrimarias= new DataColumn[1];
llavesPrimarias[0] = table.Columns["id"];
tablaPadre.PrimaryKey = llavesPrimarias;
```

- ForeignKeyConstraint – Llave foránea
  - Se crea cuando se agrega una relación entre dos tablas

```
ForeignKeyConstraint custOrderFK = new ForeignKeyConstraint("OrdenesClienteFK",
                                    custDS.Tables["Clientes"].Columns["ClienteID"],
                                    custDS.Tables["Ordenes"].Columns["ClienteID"]);
custOrderFK.DeleteRule = Rule.None;
// No puede borrar un cliente que ya esté asociado a una orden
custDS.Tables["Ordenes"].Constraints.Add(custOrderFK);
```

La colección DataRelationCollection

- Objetos DataRelation que asocian las filas de un DataTable con las de otro DataTable.
- Cada DataRelation consta de el nombre de la relación, el nombre de las tablas relacionadas y las columnas relacionadas de cada tabla.

```
private void crearRelacion() {
        // Obtienen los objetos DataColum de las dos tablas que se quieren relacionar
        DataColumn colPadre = dataSet.Tables["Clientes"].Columns[«ClienteID"];
        DataColumn colHijo= dataSet.Tables["Ordenes"].Columns["ClienteID"];
        // Crear el DataRelation.
        DataRelation relCustOrder= new DataRelation("OrdenesPorCliente", colPadre, colHijo);
        // Adicionar la relación al DataSet.
        dataSet.Relations.Add(relCustOrder);
}
```

- Pueden existir relaciones con más de una columna por tabla.
- Las relaciones pueden tener restricciones de integridad: UniqueKeyConstraint y ForeignKeyConstraint.

Refrescando datos a través de un DataSet

- Se utiliza el método DataAdapter.Fill.
- Si el DataTable tiene llaves primarias definidas se refrescan las filas teniendo en cuenta la restricción de pk.
- A través de la propiedad RowState de las filas pueden identificarse aquellas que cambiaron.
- Si se quiere refrescar conservando lo nuevo del servidor y lo nuevo en memoria:
  - En una nueva tabla recuperar los datos con el método fill del DataAdapter.
  - Luego ejecutar un Merge con la propiedad PreserveChanges en true.

Buscando datos en el DataSet

- La idea es hacerlo de forma óptima utilizando los índices de las tablas en caso de que se tengan.
- Los índices son creados cuando se crea una llave primaria o una vista asociada a una tabla.
- Si la consulta se hace sobre el PK de la tabla:
  - Usar el método DataTable.Rows.Find.
- Para consultas que no involucran el PK:
  - Construir una vista (DataView), ordenarla para que se construya el índice y utilizar los métodos Find y FindRows de la vista.

Elegir un DataReader o un DataSet

- Use un DataSet para:
  - Usar datos de forma remota entre un nivel y otro o desde un servicio web XML.
  - Interactuar con datos dinámicamente, enlazar con un control de Windows o web Forms.
  - Para combinar y relacionar datos procedentes de diferentes orígenes.
  - Almacenar datos en memoria caché dentro de la aplicación.
  - Proporcionar una vista XML de los datos.
  - Procesamiento de datos sin necesidad de tener una conexión abierta con el origen.
- Utilizar el DataReader sino necesita la funcionalidad de modo desconectado que proporciona el DataSet. El rendimiento es mejor por no requerir la memoria y el procesamiento que se requiere con el DataSet.

Referencias

- Sitio oficial de ADO. Net http://msdn.microsoft.com/es-es/library/e80y5yhx(VS.80).aspx