Procedimientos Almacenados y ADO.Net

¿Que es un Procedimiento Almacenado?

Un **procedimiento almacenado** (***stored procedure***) es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos. Generalmente son escritos en un lenguaje de bases de datos propietario como T-SQL para SQL Server o PL/SQL para Oracle database. **La ventaja de un procedimiento almacenado** es que al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y solo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

Transact-SQL

Transact-SQL (T-SQL) es el lenguage de programación del SQL Sever, a través de el podemos realizar muchas operaciones relacionadas con el SQL sin tener que volver a pasar por código ASP o VB, esto simplificará el código y ganará en rapidez dado que el T-SQL se ejecuta dentro del SQL Sever y es código compilado, se compila la primera vez que se ejecuta el Stored Procedure.

EL T-SQL se puede utilizar desde multitud de aplicaciones y desde diferentes lenguajes de progrmación :

- Desde Visual Basic
- Desde Visual C++
- Desde Active Server Pages (ASP)
- etc...

No se utiliza dentro de estos lenguajes sino en desde los llamados Stored Procedures (SP) que estan en la propia base de datos.

Ejemplo:

```
Alter Procedure pubs_listar_authors
As
Select * From authors
```

Este SP devuelve un conjunto de resultados de la base de datos PUBS, devuelve todos los registros de la tabla authors.

Pero esto es la parte sencilla del T-SQL, este no solo sirve para hacer consulta o insert, T-SQL es un potente lenguaje de programación orientado al SQL Server y como tal tiene :

- instrucciónes para el control de flujo,
- variables,
- tipos de datos
- Funciones matemática, de tratamiento de cadenas, de fecha y hora

Pero además incluye funciones propias del SQL Sever para trabajar con las bases de datos.

Tipos de datos

Como todo lenguaje de programación T-SQL posee una serie de tipos de datos, estos corresponden con los tipos de datos que pueden utilizarse en SQL Server al definir tablas, entre ellos podemos destacar :

- **int** Datos enteros (números enteros) comprendidos entre -2^31 (-2.147.483.648) y 2^31 - 1 (2.147.483.647).
- **decimal** Datos de precisión y escala numérica fijas comprendidos entre -1038 +1 y 1038 - 1.
- **numeric** Funcionalmente equivalente a decimal.
- **char** Datos de caracteres no Unicode de longitud fija con una longitud máxima de 8.000 caracteres.
- **varchar** Datos no Unicode de longitud variable con un máximo de 8.000 caracteres.
- **datetime** Datos de fecha y hora comprendidos entre el 1 de enero de 1753 y el 31 de diciembre de 9999, con una precisión de 3,33 milisegundos.

Tiene más tipos de datos pero no los voy a poner todos aqui porque los podeis encontrar en los books OnLine del SQL Sever.

Comentarios

Como todo lenguaje de programación en T-SQL también podemos comentar nuestro código para que éste pueda ser mas amigable y leerse con más comodidad. Los comentarios se identifican de la siguiente forma :

```
-- Comentario de una linea
/* comentario de
varias lineas */
```

Variables

Para declarar variables dentro del SP utilizaremos la palabra reservada Declare seguida de @ nombre de variable y tipo de datos, de la siguiente forma:

```
Declare @NombreVariable Varchar(40)
```

Para inicializarla utilizaremos la palabra reservada Set o Select :

```
Set @NombreVariable = 'PRUEBAS'
Select @NombreVariable = 'PRUEBAS'
```

Otra forma de utilizar variables es recibiendolas como parametros de entrada o de salida desde el SP, la forma de utilizarlas sería:

```
Create sp_Pruebas
     @Var1 numeric,
     @Var2 numeric = 0,
     @Var3 numeric output
As
```

En este ejemplo se crean tres tipos de variables, la primera recibirá un parametro desde el exterior ya sea desde ASP, VB, C++ o desde otro SP ... la segunda puede recibir parametros o no, si no los recibe asumirá como parametro por defecto el 0 que es con el que la hemos inicializado y la última recibirá un parametro y devolverá un valor al exterior.

Control del flujo del programa

Para controlar el flujo del programa disponemos de una serie de instrucciones:

Palabra clave definición

- **BEGIN**...END Define un conjunto de instrucciones.
- **BREAK** Sale de un bucle while.
- **CONTINUE** Continua un bucle while.
- **IF**...**ELSE** Define una ejecución condicional y, opcionalmente, una ejecución alternativa si la condición es FALSE.
- **RETURN** Sale del Stored Procedure si ejecutar nada más.
- **WAITFOR** Espera cierto tiempo a seguir con la ejecución de SP.
- **WHILE** Repite instrucciones mientras una condición específica sea TRUE.

Funciones

Bajo estas lineas pongo un ejemplo de diferentes funciones utiles en T-SQL.

Funciones de Cadena Entre otras disponemos de:

- **ASCII** Devuelve el código ASCII del carácter más a la izquierda de una expresión de caracteres.
- **CHAR** Una función de cadena que convierte un código ASCII int en un carácter.
- **LEN** Devuelve el número de caracteres.
- **LTRIM** Devuelve una expresión de caracteres después de quitar los espacios en blanco a la izquierda.
- **REPLACE** Reemplaza por una tercera expresión todas las apariciones de la segunda expresión de cadena proporcionada en la primera expresión de cadena.
- **SUBSTRING** Devuelve parte de una expresión de caracteres.
- **UPPER** Devuelve una expresión de tipo carácter con datos de carácter en minúscula convertidos a mayúscula.

Funciones de fecha y hora Entre otras disponemos de:

- **DATEADD** Devuelve un valor datetime nuevo que se basa en la suma de un intervalo a la fecha especificada.

- **DATEDIFF** Devuelve el número de límites de fecha y hora que hay entre dos fechas especificadas.
- **DATEPART** Devuelve un entero que representa la parte de la fecha especificada de la fecha indicada.
- **GETDATE** Devuelve la fecha y hora actuales del sistema.

Además de estas tenemos también

Funciones de agregado
Funciomnes de configuración
Funciones de cursor
Funciones matematicas
Funciones de Seguridad
Funciones de sistemas
Funciones de texto e imagen

Además de lo explica en esta sección sobre la estructura del Transact-SQL, éste posee unas funciones y/o instrucciones propias del SQL Server para trabajar exclusivamente con las bases de datos, algunas conociadas y/o utilizadas y otras no tanto, entre ellas tenemos:

- **SELECT** Para realizar consultas a las bases de datos.
- **INSERT** Para insertar datos.
- **DELETE** Para borrar datos
- **UPDATE** Para modificar datos
- **DROP** Para borrar tablas, SP, Indices, Triggers ...
- **CREATE** Para crear tablas, SP, Indices, Triggers
- **ALTER** Para agregar Columnnas, Modificar Procedures ...

Uso de Procedimientos Almacenados en ADO.Net

Utilizar un procedimiento almacenado desde Visual Studio .Net no tiene casi ninguna diferencia con respecto de hacerlo utilizando queries en texto plano.
Al igual que con estos ultimos, utilizaremos un SQLCommand, al que le indicaremos que estamos hablando de un Stored Procedure, y obtendremos los datos mediante un DataAdapter, o directamente desde el Command, como lo hacemos siempre.

Ejemplo 1: Ejecutar un Procedimiento Almacenado, y guardar el resultado en un DataTable

El siguiente codigo consiste principalmente en usar un objeto **SqlDataAdapter** y en el comando de selección indicarle el nombre del procedimiento almacenado, además de asignarle el "tipo" de comando a usar, cuyo valor debe ser: **CommandType.StoredProcedure**.

Después asignamos el **DataTable** a la propiedad **DataSource** del control **DataGridView** y listo.

```
Try
Dim dt As New DataTable
Dim da As New SqlDataAdapter("sp_PorFechaAlta", My.Settings.ConnectionString)
da.SelectCommand.CommandType = CommandType.StoredProcedure
```

```
    da.SelectCommand.Parameters.Add("@FechaAlta", SqlDbType.DateTime)
da.SelectCommand.Parameters("@FechaAlta").Value = CDate(Me.txtFecha.Text)
    da.Fill(dt)

Me.dgvConsulta.DataSource = dt

    LabelStatus.Text = "Filas en el grid: " & dt.Rows.Count
Catch ex As Exception
Me.LabelNonQuery.Text = "Error: " & ex.Message
    LabelStatus.Text = "ERROR"
End Try
```

Ejemplo 2: Ejecutar un Procedimiento Almacenado de Update

El siguiente codigo consiste principalmente en usar un objeto **SqlCommand** para actualizar un registro en la base de datos.
Luego se le muestra un mensaje al usuario con el resultado de esta operacion.

```
Try
Dim dt As New DataTable
Dim cm As New SqlCommand("sp_ActFechaAlta", My.Settings.ConnectionString)
cm.CommandType = CommandType.StoredProcedure
    cm.Parameters.Add("@Id"), SqlDbType.Int)
    cm.Parameters("@Id").Value = CInt(Me.txtIdentificacion.Text)
    cm.Parameters.Add("@FechaAlta", SqlDbType.DateTime)
cm.Parameters("@FechaAlta").Value = CDate(Me.txtFecha.Text)

    If cm.ExecuteNonQuery() Then
        MsgBox("Los datos se modificaron correctamente")
        LabelStatus.Text = "Actualizacion OK"
    Else
        MsgBox("Los datos se modificaron correctamente")
        LabelStatus.Text = "ERROR"
    End If

Catch ex As Exception
Me.LabelNonQuery.Text = "Error: " & ex.Message
    LabelStatus.Text = "ERROR"
End Try
```

Bibliografia Utilizada:

- **Programacion Avanzada con Microsoft Visual Basic .Net - Francesco Balena**
- **http://www.netveloper.com/contenido2.aspx?IDC=47_0_**

Integrantes:

- Hanglin, Maximiliano
- Dias Tasara, Pablo
- Brighina, Lucas
- Maidana, Matias
- Iglesias, Mauro
- Sanchez, Carlos