9- Procedimientos almacenados.

Objetivos:

- Crear procedimientos almacenados para ser usados en el desarrollo de software.

Recursos:

- Microsoft SQL Server Management Studio
- Guías prácticas.
- Base de datos de ejemplo: autos.

Introducción

Uno de los procedimientos más usados en el diseño de la base de datos, son los Procedimiento almacenados, pues estos permiten agilizar los procesos de consultas de datos, aumentar la seguridad, reutilizar código y permiten desarrollo de software más ágil evitar hacer más código.

Procedimientos almacenados.

Un procedimiento almacenado de SQL Server es un grupo de una o varias instrucciones Transact-SQL o una referencia a un método de Common Runtime Language (CLR) de Microsoft .NET Framework. Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.

- Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos

- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Ventajas de usar procedimientos almacenados.

Tráfico de red reducido entre el cliente y el servidor

Los comandos de un procedimiento se ejecutan en un único lote de código. Esto puede reducir significativamente el tráfico de red entre el servidor y el cliente porque únicamente se envía a través de la red la llamada que va a ejecutar el procedimiento.

Mayor seguridad

Varios usuarios y programas cliente pueden realizar operaciones en los objetos de base de datos subyacentes a través de un procedimiento, aunque los usuarios y los programas no tengan permisos directos sobre esos objetos subyacentes. El procedimiento controla qué procesos y actividades se llevan a cabo y protege los objetos de base de datos subyacentes. Esto elimina la necesidad de conceder permisos en cada nivel de objetos y simplifica los niveles de seguridad.

Reutilización del código

El código de cualquier operación de base de datos redundante resulta un candidato perfecto para la encapsulación de procedimientos. De este modo, se elimina la necesidad de escribir de nuevo el mismo código, se reducen las inconsistencias de código y se permite que cualquier usuario o aplicación que cuente con los permisos necesarios pueda acceder al código y ejecutarlo.

Mantenimiento más sencillo

Cuando las aplicaciones cliente llaman a procedimientos y mantienen las operaciones de base de datos en la capa de datos, solo deben actualizarse los cambios de los procesos en la base de datos subyacente. El nivel de aplicación permanece independiente y no tiene que tener conocimiento sobre los cambios realizados en los diseños, las relaciones o los procesos de la base de datos.

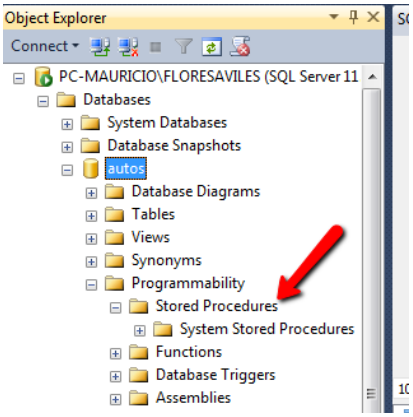
Rendimiento mejorado

De forma predeterminada, un procedimiento se compila la primera vez que se ejecuta y crea un plan de ejecución que vuelve a usarse en posteriores ejecuciones. Como el procesador de consultas no tiene que crear un nuevo plan, normalmente necesita menos tiempo para procesar el procedimiento.

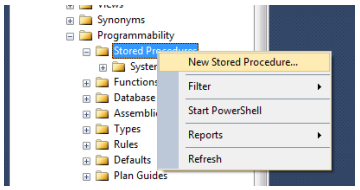
Ejemplo: Utilizando SQL Server Management Studio

Utilizando la base de datos “Autos”, crearemos un procedimiento almacenado para realizar búsquedas de repuestos por su nombre y que cumplan la condición que estén arriba de un precio dado.

En primer lugar vamos a buscar la base de datos “Autos” y la expandiremos, después buscaremos Programmability (programación) y la expandiremos, y nos quedara como lo muestra la siguiente figura.



Haga clic con el botón secundario en Procedimientos almacenados y, a continuación, haga clic en Nuevo procedimiento almacenado.



Este procedimiento nos devolverá la una pestaña de consulta, el siguiente código:

```
-- ================================================
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- ================================================
```

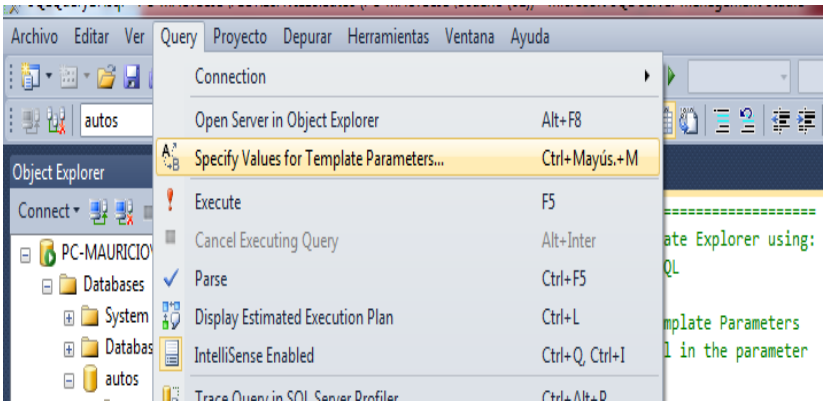
```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:              <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
        -- Add the parameters for the stored procedure here
        <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
        <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

    -- Insert statements for procedure here
        SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

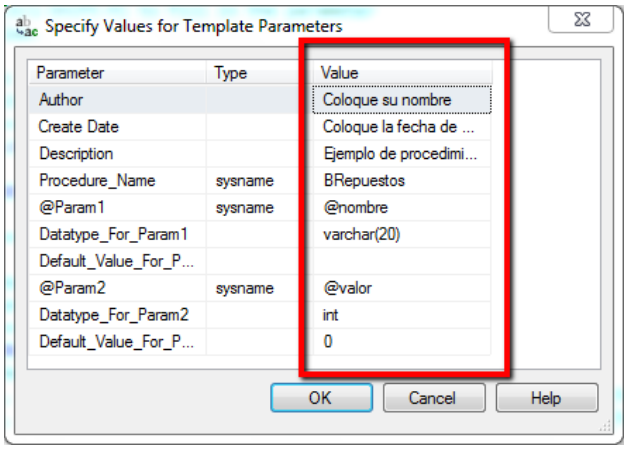
Es importante hacer ver que por default me presenta para realizar un procedimiento con dos parámetros, pero si necesito más o menos, se puede borrar o aumentar según sea conveniente para la consulta que queremos utilizar.

Ademas se presentan opciones para poner valores por ***"Default"*** a las variables, estos no son obligatorios y pueden usarse según sea conveniente o borrarse.

En el menú Query (Consulta), haga clic en Specify Values for Template Parameters (Especificar valores para parámetros de plantilla), como lo muestra la siguiente figura.



Esta opción me mostrar un cuadro de dialogo, en cual usted debe especificar valores para los parámetros de plantilla, especifique los siguientes valores para los parámetros mostrados.



Después en el Editor de consultas, reemplace la instrucción SELECT por la siguiente instrucción:

```
SELECT repuestos.nombre,repuestos.precio,repuestos.descuento FROM repuestos
        WHERE repuestos.nombre like @Nombre and precio > @valor
```

Es importante corregir una línea donde se encuentra los paramentos

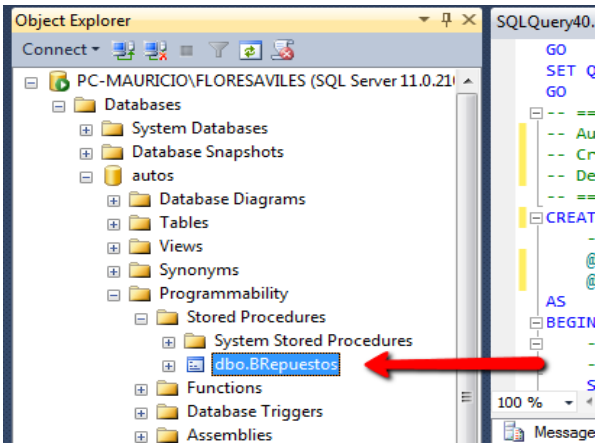
```
CREATE PROCEDURE BRepuestos
    -- Add the parameters for the stored procedure here
    @nombre varchar(20) = ,
    @valor int = 0
AS
```

Debemos borrar el signo de igual, y dejar la instrucción con la figura siguiente.

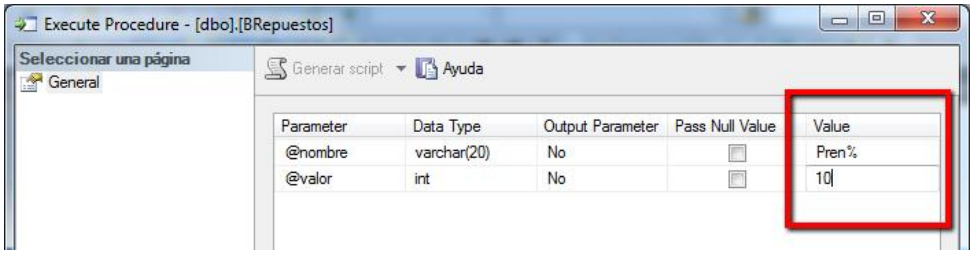
```
CREATE PROCEDURE BRepuestos
    -- Add the parameters for the stored procedure here
    @nombre varchar(20),
    @valor int = 0
AS
```

Para probar la sintaxis, en el menú Query (Consulta), haga clic en Parse (Analizar). Si se devuelve un mensaje de error, compare las instrucciones con la información anterior y corrija lo que sea necesario.

Una vez todo listo proceda a Execute (Ejecutar). El procedimiento se crea como un objeto de la base de datos, tal como lo muestra la figura.

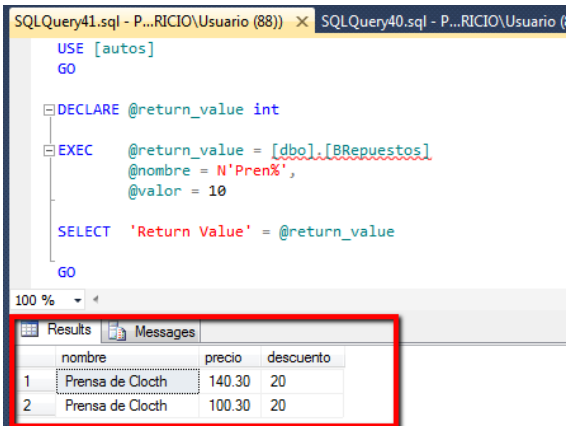


Para ejecutar el procedimiento, en el Explorador de objetos, haga clic con el botón derecho en el nombre del procedimiento almacenado BRepuestos y seleccione Execute Store Procedure (Ejecutar procedimiento almacenado)



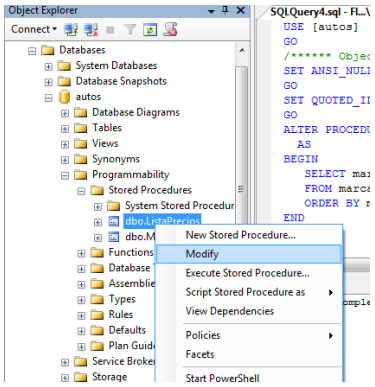
En esta opción veremos que la ventana nos pide el valor de los dos parámetros, para el ejemplo hacemos que busque todos los repuestos que empiecen con ***"Pren"*** agregamos el comodin ***"%"***, y en el valor ponemos ***10***.

Ahora para ver el resultado presionamos aceptar, y nos mostrara los siguientes resultados.

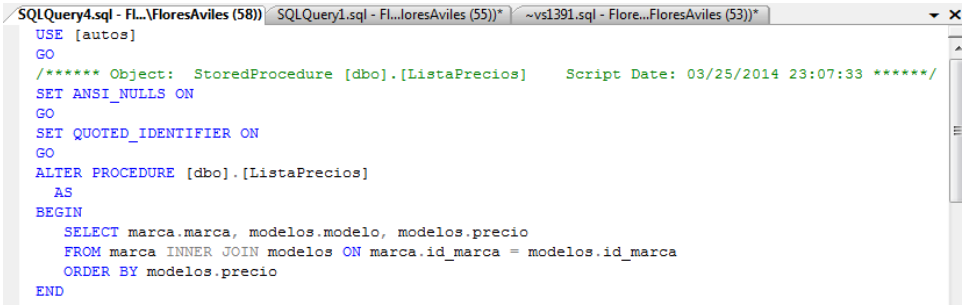


Modificar procedimientos almacenados.

Para modificar un procedimiento almacenado lo primero es buscarlo en el explorador de objetos y después clic derecho sobre el, como lo muestra la figura.



Lo cual nos colocara el código para ser editado en el manejador de consultas, como se muestra en la siguiente imagen.



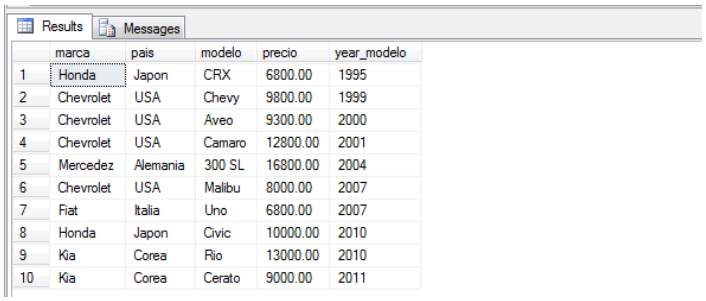
Cambiaremos un poco la consulta, aumentándole dos campos y cambiando el orden de ella, quedando la consulta de esta manera:

```
SELECT marca.marca,marca.pais,modelos.modelo, modelos.precio,modelos.year_modelo
   FROM marca INNER JOIN modelos ON marca.id_marca = modelos.id_marca
   ORDER BY modelos.year_modelo
```

Una vez efectuados los cambios, ejecutamos el procedimiento almacenado, y ya podemos utilizarlo.

```
EXEC ListaPrecios
```

Lo cual me presenta el siguiente resultado:



Results | Messages

	marca	pais	modelo	precio	year_modelo
1	Honda	Japon	CRX	6800.00	1995
2	Chevrolet	USA	Chevy	9800.00	1999
3	Chevrolet	USA	Aveo	9300.00	2000
4	Chevrolet	USA	Camaro	12800.00	2001
5	Mercedez	Alemania	300 SL	16800.00	2004
6	Chevrolet	USA	Malibu	8000.00	2007
7	Fiat	Italia	Uno	6800.00	2007
8	Honda	Japon	Civic	10000.00	2010
9	Kia	Corea	Rio	13000.00	2010
10	Kia	Corea	Cerato	9000.00	2011

Ejemplo: Utilizando Transact SQL

También puede escribir directamente el código y crear un procedimiento almacenado directamente, un ejemplo seria el siguiente código:

```
USE autos
GO

CREATE PROCEDURE ModelosXMarca
   @busca varchar(15)

AS
BEGIN
   SELECT marca.marca, modelos.modelo, modelos.precio
   FROM marca INNER JOIN modelos ON marca.id_marca = modelos.id_marca
   WHERE marca.marca like @busca
END
```

Este procedimiento muestra los modelos de una marca de carros, a partir de nombre de marca de búsqueda.

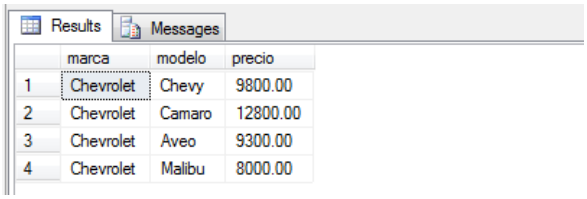
Para probar este procedimiento, lo ejecutamos usando la palabra ***"Execute"*** o ***"Exce",*** por ejemplo:

```
EXEC ModelosXMarca 'Che%'
```

O

```
EXECUTE ModelosXMarca 'Che%'
```

El resultado será el siguiente:



Results | Messages

	marca	modelo	precio
1	Chevrolet	Chevy	9800.00
2	Chevrolet	Camaro	12800.00
3	Chevrolet	Aveo	9300.00
4	Chevrolet	Malibu	8000.00

También puedo crear procedimientos almacenados sin parámetro, un ejemplo seria el

siguiente:

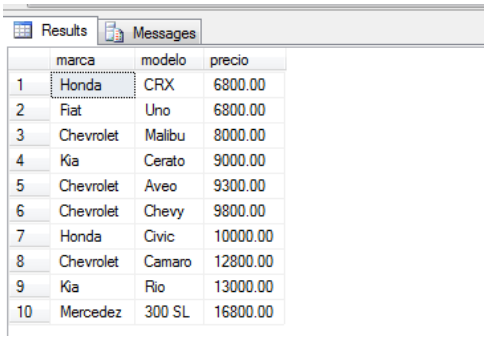
```
USE autos
GO

CREATE PROCEDURE ListaPrecios
  AS
BEGIN
   SELECT marca.marca, modelos.modelo, modelos.precio
   FROM marca INNER JOIN modelos ON marca.id_marca = modelos.id_marca
   ORDER BY modelos.precio
END
```

Lo probamos ejecutando el siguiente código:

```
EXEC ListaPrecios
```

Y el resultado es el siguiente:



	marca	modelo	precio
1	Honda	CRX	6800.00
2	Fiat	Uno	6800.00
3	Chevrolet	Malibu	8000.00
4	Kia	Cerato	9000.00
5	Chevrolet	Aveo	9300.00
6	Chevrolet	Chevy	9800.00
7	Honda	Civic	10000.00
8	Chevrolet	Camaro	12800.00
9	Kia	Rio	13000.00
10	Mercedez	300 SL	16800.00

Ejercicios:

- Cree un procedimiento almacenado para mostrarme una lista de repuestos con su nombre, precio, porcentaje de descuento y el valor que tuviera si se aplica dicho descuento.
- Elabore otro que tenga un parámetro que me pida el modelo del auto, y que me muestre todos los repuestos que pertenecen a ese modelo de auto.
- Ahora elabore uno que muestre la marca, el país, el nombre del modelo y el precio, pero que me pida dos parámetros, país y precio, para usarlo en la búsqueda.
- También elabore uno que me muestre los datos del repuesto, modelo y marca, a partir de que el precio de los repuestos, este entre dos valores.
- Cree una lista de modelos y que muestre la cantidad de repuestos que hay por cada modelo.
- Cada instructor asignará ejercicios adicionales.