UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

LABORATORIO DE COMPUTACIÓN II

Primera Clase

Introducción a los Procedimientos Almacenados

1° Año – 2° Cuatrimestre

Índice

Introducción a los Procedimientos Almacenados

Un procedimiento almacenado es un conjunto de instrucciones a las que se les da un nombre, que se almacena en el servidor. Permiten encapsular tareas repetitivas. SQL Server permite los siguientes tipos de procedimientos almacenados:

- **Del sistema**: están almacenados en la base de datos "master" y llevan el prefijo "sp_".
- **Locales**: los crea el usuario
- **Temporales**: pueden ser locales, cuyos nombres comienzan con un signo numeral (#), o globales, cuyos nombres comienzan con 2 signos numeral (##). Los procedimientos almacenados temporales locales están disponibles en la sesión de un solo usuario y se eliminan automáticamente al finalizar la sesión; los globales están disponibles en las sesiones de todos los usuarios.
- **Remotos** son una característica anterior de SQL Server.
- **Extendidos**: se implementan como bibliotecas de vínculos dinámicos (DLL, Dynamic-Link Libraries), se ejecutan fuera del entorno de SQL Server. Generalmente llevan el prefijo "xp_".

Un procedimiento almacenado puede hacer referencia a objetos que no existen al momento de crearlo. Los objetos deben existir cuando se ejecute el procedimiento almacenado.

Los procedimientos almacenados en SQL Server pueden:

- Contener instrucciones que realizan operaciones en la base de datos y llamar a otros procedimientos almacenados.
- Aceptar parámetros de entrada.
- Devolver un valor de estado a un procedimiento.
- Devolver varios parámetros de salida.

Ventajas:

- Comparten la lógica de la aplicación con las otras aplicaciones, con lo cual el acceso y las modificaciones de los datos se hacen en un solo sitio.
- Permiten realizar todas las operaciones que los usuarios necesitan evitando que tengan acceso directo a las tablas.
- Reducen el tráfico de red; en vez de enviar muchas instrucciones, los usuarios realizan operaciones enviando una única instrucción.

- Pueden encapsular la funcionalidad del negocio. Las reglas o directivas empresariales encapsuladas en los procedimientos almacenados se pueden cambiar en una sola ubicación.
- Apartar a los usuarios de la exposición de los detalles de las tablas de la base de datos.
- Proporcionar mecanismos de seguridad: los usuarios pueden obtener permiso para ejecutar un procedimiento almacenado incluso si no tienen permiso de acceso a las tablas o vistas a las que hace referencia.

Los procedimientos almacenados pueden hacer referencia a tablas, vistas, a funciones definidas por el usuario, a otros procedimientos almacenados y a tablas temporales.

Un procedimiento almacenado puede incluir cualquier cantidad y tipo de instrucciones, excepto: *create default, create procedure, create rule, create trigger y create view.*

Si un procedimiento almacenado crea una tabla temporal, dicha tabla sólo existe dentro del procedimiento y desaparece al finalizar el mismo. Lo mismo sucede con las variables.

Creación y ejecución de Procedimientos Almacenados

Para crear un procedimiento almacenado se emplea la instrucción "create procedure". La sintaxis básica parcial es:

CREATE PROCEDURE nombreprocedimiento
AS instrucciones;

Con las siguientes instrucciones se crea un procedimiento almacenado llamado "pa_articulos_precios_mas100" que muestra todos los artículos de la librería con precios mayores a 100:

```
CREATE PROC pa_ articulos_precios_mas100
 AS
  SELECT *
  FROM   articulos
  WHERE  pre_unitario >100;
```

El procedimiento se ejecuta colocando "execute" (o "exec") seguido del nombre del procedimiento almacenado creado anteriormente como se muestra en la siguiente sentencia:

```
EXEC pa_articulos_precios_mas100;
```

Modificar procedimientos almacenados

Los procedimientos almacenados pueden modificarse, por necesidad de los usuarios o por cambios en la estructura de las tablas que referencia. Sintaxis:

ALTER PROCEDURE nombreprocedimiento
@parametro TIPO = valorpredeterminado
AS sentencias;

Modificar el procedimiento almacenado anterior para que muestre determinadas columnas:

```
ALTER PROC pa_articulos_precios_mas100
 AS
  SELECT descripcion, pre_unitario,observaciones,stock
  FROM   articulos
  WHERE  pre_unitario >100;
```

Eliminar procedimientos almacenados

Los procedimientos almacenados se eliminan con "DROP PROCEDURE". Sintaxis:

DROP PROCEDURE nombreprocedimiento;

Eliminar el procedimiento almacenado llamado "pa_articulosA":

```
DROP PROCEDURE pa_articulosA;
```

Procedimientos almacenados con parámetros de entrada

Los parámetros de entrada posibilitan pasar información a un procedimiento. Para que un procedimiento almacenado admita parámetros de entrada se deben declarar variables como parámetros al crearlo. La sintaxis es:

CREATE PROC nombreprocedimiento
@nombreparametro tipo =valorpordefecto
AS sentencias;

Los parámetros se definen luego del nombre del procedimiento, comenzando con un signo arroba (@). Los parámetros son locales al procedimiento, es decir, existen solamente dentro del mismo. Pueden declararse varios parámetros por procedimiento, se separan por comas.

Cuando el procedimiento es ejecutado, deben explicitarse valores para cada uno de los parámetros (en el orden que fueron definidos), a menos que se haya definido un valor por defecto, en tal caso, pueden omitirse.

Luego de definir un parámetro y su tipo, se puede especificar un valor por defecto; tal valor es el que asume el procedimiento al ser ejecutado si no recibe parámetros. El valor por defecto puede ser "null" o una constante, también puede incluir comodines si el procedimiento emplea "like".

Como ejemplo se crea un procedimiento que recibe dos precios unitarios para mostrar todos aquellos que se encuentren entre dichos valores:

```
create procedure pa_articulos_precios
 @precio1 money,
 @precio2 money,
AS
 select  descripcion,pre_unitario,observaciones
  from   articulos
  where  pre_unitario between @precio1 and @precio2;
```

El procedimiento se ejecuta colocando "exec" seguido del nombre del procedimiento y los valores para los parámetros separados por comas:

```
exec pa_articulos_precios 100,200;
```

El ejemplo anterior ejecuta el procedimiento pasando valores a los parámetros por posición. También podemos emplear la otra sintaxis en la cual pasamos valores a los parámetros por su nombre:

```
exec pa_articulos_precios @precio1=100, @precio2=200;
```

En este caso, cuando los valores son pasados con el nombre del parámetro, el orden en que se colocan puede alterarse.

```
exec pa_articulos_precios @precio2=200, @precio1=100;
```

Si se quiere ejecutar un procedimiento que permita omitir los valores para los parámetros se debe, al crear el procedimiento, definir valores por defecto para cada parámetro:

```
create procedure pa_articulos_precios
  @precio1 money=100,
  @precio2 money=200,
 AS
  select descripcion,pre_unitario,observaciones
   from articulos
   where pre_unitario between @precio1 and @precio2;


exec pa_articulos_precios;
```

Si se envía un solo parámetro a un procedimiento que tiene definido más de un parámetro sin especificar a qué parámetro corresponde (valor por posición), asume que es el primero.

Se puede emplear patrones de búsqueda en la consulta que define el procedimiento almacenado y utilizar comodines como valores por defecto:

```
 create procedure pa_articulos_descripcion
  @descripcion varchar(100)= '%'
AS
  select descripcion,pre_unitario,observaciones
   from articulos
   where descripción like @descripcion;
```

```
exec pa_articulos_descripcion;
```

Procedimientos almacenados con parámetros de salida

Dijimos que los procedimientos almacenados pueden devolver información; para ello se emplean parámetros de salida. El valor se retorna a quien realizó la llamada con parámetros de salida.

Para que un procedimiento almacenado devuelva un valor se debe declarar una variable con la palabra clave "output" al crear el procedimiento:

```
CREATE PROCEDURE nombreprocedimiento
  @parametroentrada tipo =valorpordefecto,
  @parametrosalida tipo=valorpordefecto output
 AS
   sentencias
   select @parametrosalida=sentencias;
```

Los parámetros de salida pueden ser de cualquier tipo de datos, excepto text, ntext e image. Se crea un procedimiento almacenado que muestre los descripción, precio y observaciones de los articulos que comiencen con una (o más) letra/s determinada/s (enviada como parámetro de entrada) y retorne la suma y el promedio de los precios de todos los artículos que cumplan con la condición de búsqueda:

```
create procedure pa_articulos_sumaypromedio
 @descripcion varchar(100)='%',
 @suma decimal(10,2) output,
 @promedio decimal(8,2) output
 as
  select descripcion,pre_unitario,observaciones
   from  articulos
   where descripción like @descripcion
 select @suma=sum(precio)
  from  articulos
  where descripción like @descripcion
 select @promedio=avg(precio)
  from  articulos
  where descripción like @descripcion;
```

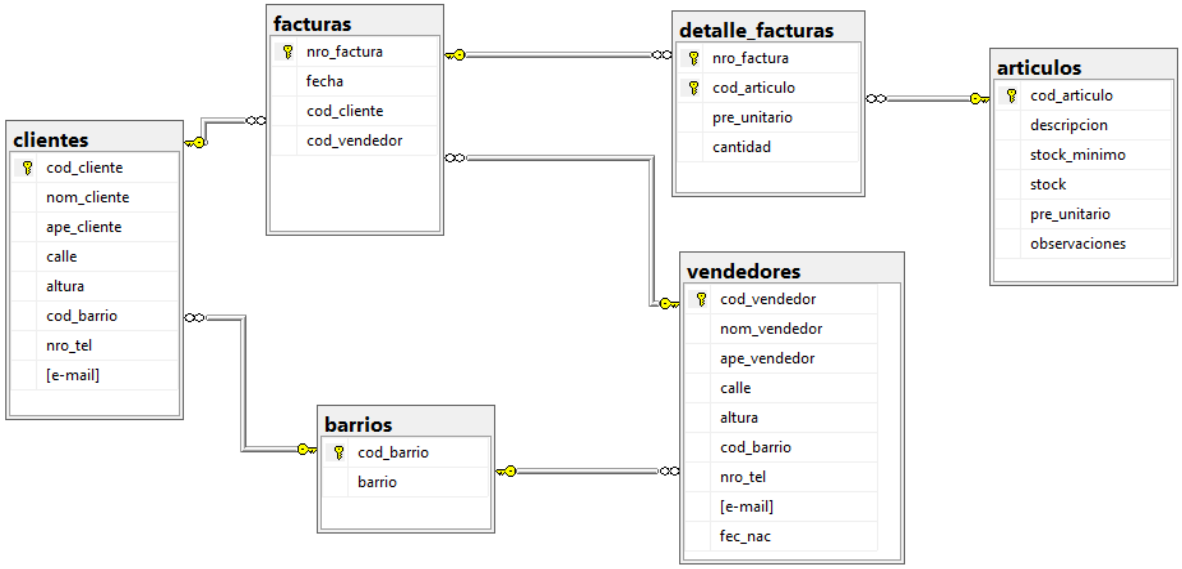
Al ejecutar el procedimiento se puede ver el contenido de las variables en las que se almacenan los parámetros de salida del procedimiento (Al ejecutarlo también debe emplearse "output"):

```
declare @s decimal(10,2), @p decimal(8,2)
execute pa_autor_sumaypromedio 'lápiz%', @s output, @p output
select @s as total, @p as promedio;
```

Problema:

Desde el área de programación del sistema de información de la librería nos solicitan la creación de objetos en la base de datos que permitan ingresar parámetros en el momento de la ejecución, es decir crear Procedimientos Almacenados (SP):

Antes de comenzar con la resolución de este problema, tenga en cuenta la estructura de la base de datos Librería que se presenta en la imagen siguiente:



Por otro lado, si está trabajando en las computadores de la Facultad, al crear el SP, incluya en el nombre, su legajo, para identificar el creado por usted respecto a los de sus compañeros; recuerde que está en un entorno cliente/servidor, la base de datos se encuentra en un servidor y es la misma para todos los clientes/usuarios. También, si el legajo está al principio del nombre del SP, debe escribir dicho nombre entre comillas ya que todo nombre de objeto en SQL Server, debe comenzar con caracter alfanumérico.

1. **Crear un SP llamado "legajo_pa_articulos_precios" que muestra todos los artículos cuyo precio unitario es menor a 100:**

```
CREATE PROC "123456_pa_articulos_precios
 AS
  SELECT * FROM articulos
```

```
  WHERE pre_unitario <100;
```

Para ejecutar el procedimiento almacenado creado anteriormente tipeamos:

```
EXEC "123456_pa_articulos_precios";
```

Para eliminar el procedimiento almacenado llamado " pa_articulos_precios ":

```
DROP PROCEDURE "123456_pa_articulos_precios";
```

2. **Crear un SP que muestre los artículos cuyo precio unitario sea menor a un precio que se ingresará en el momento en que se ejecute.**

```
CREATE PROC pa_articulos_precios2
@precio money
  AS
   SELECT * FROM articulos
   WHERE pre_unitario <@precio
```

Luego se ejecuta con la siguiente sentencia:

```
exec pa_articulos_precios2 100';
```

3. **Si se quiere listar los artículos cuyo precio esté entre dos valores.**

```
CREATE PROC pa_articulos_precios3
@precio1 money,
@precio2 money
  AS
   SELECT * FROM articulos
   WHERE pre_unitario between @precio1 and @precio2

exec pa_ articulos_precios3 @precio1=50, @precio2=100;
```

4. **Crear un procedimiento almacenado que muestre la descripción de un artículo de código determinado (enviado como parámetro de entrada) y nos retorne el total facturado para ese artículo y el promedio ponderado de los precios de venta de ese artículo**

```
Create proc pa_ventas_articulo
@codigo int,
@total decimal(10,2) output,
@precioprom decimal(6,2) output
as
 select descripcion from articulos
```

```
where cod_articulo=@codigo
select @total=sum(cantidad*pre_unitario)
from detalle_facturas
where cod_articulo=@codigo
select @precioprom=sum(pre_unitario)/sum(cantidad)
from detalle_facturas
where cod_articulo=@codigo
```

Ejecutar el procedimiento:

```
 declare @s decimal(12,2), @p decimal(10,2)
execute pa_ventas_articulo 5, @s output, @p output
select @s total, @p  'Precio promedio'
```

5. **Crear un procedimiento que muestre todos los articulos menores a un precio que se ingresa como parámetro**

```
create procedure pa_articulos_precios
 @precio decimal(12,2)=null
as
if @precio is null
begin
 select 'Debe indicar un precio'
 return
end;

select descripcion from  articulos where precio< @precio
```

6. **Crear un procedimiento almacenado que ingresa registros en la tabla "articulos". Los parámetros correspondientes al pre_unitario DEBEN ingresarse con un valor distinto de "null", los demás son opcionales. El procedimiento retorna "1" si la inserción se realiza, es decir, si se ingresan valores para pre_unitario y "0", en caso que pre_unitario sea nulo.**

```
create procedure pa_articulos_ingreso
@descripcion nvarchar (50) NULL ,
@stock_minimo smallint NULL ,
@pre_unitario decimal(10, 2) NOT NULL ,
@observaciones nvarchar (50)=null,
as
 if (@pre_unitario is null)
  return 0
 else
 begin
  insert into articulos
values(@descripcion,@stock_minimo,@pre_unitario,@observaciones)
  return 1
 end;
```

Para ver el resultado, debemos declarar una variable en la cual se almacene el valor devuelto por el procedimiento; luego, ejecutar el procedimiento asignándole el valor devuelto a la variable, finalmente mostramos el contenido de la variable:

```
declare @retorno int
exec     @retorno=pa_articulos_ingreso     @descripcion     ='Regla
50cm',@pre_unitario=75
select 'Ingreso realizado=1' = @retorno

exec @retorno=pa_articulos_ingreso
select 'Ingreso realizado=1' = @retorno;
```

7. **Legajo_Detalle_Ventas: liste la fecha, la factura, el vendedor, el cliente, el artículo, cantidad, precio e importe. Entre dos fechas que se ingresen por parámetro.**
8. **Legajo_Antiguedad_Vend: Este SP lista los vendedores con una antigüedad mayor a la que se ingresa en el momento de ejecutar el SP**
9. **Legajo_INS_Vendedor: Cree un SP que le permita insertar registros en la tabla vendedores.**
10. **Legajo_UPD_Vendedor: cree un SP que le permita modificar un vendedor cargado.**
11. **Legajo_DEL_Vendedor: cree un SP que le permita eliminar un vendedor ingresado.**
12. **Listar las ventas de los clientes cuyos nombres y/o apellidos comiencen con la o las letras que se ingresen en el momento de la ejecución**

"El trabajo ocupará una gran parte de tu vida, la mejor forma de lidiar con ello, es encontrar algo que realmente ames."
Steve Jobs