Ejercicios 3 de Programación en Transact-SQL

Vamos a usar para los ejercicios la base de datos Gestion8. Abre una nueva consulta y escribe:

```
USE Gestion8
```

Empezaremos por redactar un procedimiento (CuantosPedRep3_1) que nos devuelva cuántos pedidos ha realizado un determinado representante. Si el representante no existe devolverá 0.

Hasta ahora, en los ejercicios que hemos realizado la salida se efectuaba a través de la pestaña Mensajes o Resultados pero el procedimiento no devolvía nada al código que le había llamado, ahora sí. Nos piden que el procedimiento devuelva algún (o algunos) resultado, por lo que deberemos utilizar parámetros de salida o el RETURN según el caso.
En este caso lo que hay que devolver es un número entero (cuántos pedidos) por lo que podemos utilizar las dos formas. Vamos a hacerlo primero utilizando un parámetro de salida y en el siguiente ejercicio lo haremos utilizando el RETURN.

Empezaremos el script como en los ejercicios anteriores eliminando el procedimiento si ya existe para luego crearlo. Escribe:

```
-- 1.- Redactar un procedimiento que nos devuelva cuántos pedidos ha realizado un
determinado representante, si el representante no existe devolverá 0.
IF object_id('CuantosPedRep3_1', 'P') IS NOT NULL DROP PROC CuantosPedRep3_1;
GO
```

El procedimiento tiene un parámetro de entrada (para recoger el valor del representante del cual se quiere saber cuántos pedidos tiene), y uno de salida para recoger lo que devuelve el procedimiento. Los dos son enteros.
Empezamos por indicar el nombre del procedimiento y sus parámetros:

```
CREATE PROCEDURE CuantosPedRep3_1 @rep INT, @cuantos INT OUTPUT
```

A continuación definimos el cuerpo del procedimiento:

```
AS
      SET @Cuantos = (SELECT count(*) FROM pedidos WHERE rep=@rep);
```

Asignamos al parámetro de salida el resultado de la SELECT. Y terminamos la definición con un GO, escribe:

```
GO
```

Después de definir el procedimiento ahora lo vamos a probar. ¿Qué situaciones se pueden presentar?

- Un representante que no existe
- Un representante sin pedidos
- Un representante con pedidos (es mejor elegir uno que tenga varios pedidos).

Buscamos en las tablas esos representantes y esos códigos serán los que usaremos para probar el procedimiento.

```
-- Ahora vamos a probar el procedimiento:
PRINT 'Prueba del ejercicio 1'  -- Esto lo ponemos para saber por dónde vamos si
                                   ejecutamos el script completo
```

Como el procedimiento lleva un parámetro de salida, antes de llamarlo tenemos que tener a nivel de script una variable donde recoger lo que nos devuelve el proc.

```
DECLARE @resul INT
```

Ahora podemos llamar al procedimiento:

```
EXEC CuantosPedRep3_1 101, @resul OUTPUT  -- Un representante que tiene varios
pedidos
Para poder ver el resultado devuelto por el procedimiento sacamos un mensaje con el
valor recogido en la llamada:
PRINT @resul
```

Y comprobamos que el número devuelto corresponde con el número de pedidos del empleado 101.

Después hacemos la prueba con un empleado que no tiene pedidos, para recoger el valor devuelto podemos utilizar la misma variable de antes porque el valor anterior ya no nos sirve.

```
EXEC CuantosPedRep3_1 104, @resul OUTPUT -- Un representante que no tiene pedidos
```

Para visualizar mejor el resultado en este caso vamos a mejorar el PRINT del resultado, sacaremos un mensaje más explícito:

```
PRINT 'El representante 104 tiene ' + STR(@resul,3) + ' pedidos'
```

Así hemos practicado la concatenación de texto fijo con variables. Comprobamos que el resultado devuelto es cero.

Por último probamos el procedimiento con un empleado que no existe:

```
EXEC CuantosPedRep3_1 200, @resul OUTPUT -- Un representante que no existe
PRINT 'El representante 200 tiene ' + STR(@resul,3) + ' pedidos'
```

En este caso, cuando el representante no tiene pedidos quedaría mejor que el mensaje fuese diferente según el caso. Sustituye el anterior PRINT por:

```
IF @resul = 0 PRINT 'El representante 200 no tiene pedidos'
   ELSE  PRINT 'El representante 200 tiene ' + STR(@resul,3) + ' pedidos'
```

Ejecuta el script.

El PRINT indicado después de cada EXEC varía para que veas diferentes maneras de presentar el resultado. Puedes también cambiar los PRINTs asociados a los demás representantes para comprobar que el IF funciona en todos los casos. Recuerda sustituir el código del representante (200) por el correspondiente en cada caso.

```
-- 2.- Como el valor devuelto es un entero podríamos haber utilizado el RETURN en vez de un parámetro de salida:
-- En este caso no definimos parámetro de salida (OUTPUT), sólo el de entrada.
IF object_id('CuantosPedRep3_2', 'P') IS NOT NULL DROP PROC CuantosPedRep3_2;
GO
CREATE PROCEDURE CuantosPedRep3_2 @rep INT
AS
-- La asignación del valor al RETURN también es diferente:
      RETURN (SELECT count(*) FROM pedidos WHERE rep=@rep);
GO
```

Y la llamada al procedimiento también va a ser diferente:

```
PRINT 'Prueba del ejercicio 2' – lo ponemos para que cuando ejecutemos el script completo sepamos por donde vamos.
```

Seguimos necesitando una variable a nivel de script para recoger el resultado devuelto.

```
DECLARE @resul INT
```

La llamada es diferente, es más parecida a la llamada a una función (asignamos la función a una variable que almacena el resultado) pero sin paréntesis y con la palabra EXEC delante.

```
EXEC @resul = CuantosPedRep3_2 101 -- Un representante que tiene varios pedidos

PRINT @resul

EXEC @resul = CuantosPedRep3_2 104 -- Un representante que no tiene pedidos

PRINT 'El representante 104 tiene ' + STR(@resul,3) + ' pedidos'

EXEC @resul = CuantosPedRep3_2 200 -- Un representante que no existe

IF @resul = 0 PRINT 'El representante 200 no tiene pedidos'
   ELSE  PRINT 'El representante 200 tiene ' + STR(@resul,3) + ' pedidos'
```