Ejemplos de Programación en Transact-SQL

Vamos a usar para los ejemplos la base de datos Gestion8.

Abre una nueva consulta, vamos a crear un script que contenga tanto las instrucciones para crear los procedimientos como para comprobar que funcionan.

Para asegurarnos que al ejecutar siempre se ejecute sobre Gestion 8 empezamos por un USE, escribe:

```
USE Gestion8;
```

1.- A continuación queremos crear un procedimiento que liste los clientes, será con un CREATE PROC.

Escribe:

```
-- 1.- Listar los datos de los clientes.
CREATE PROCEDURE ListarClientes
AS
      SELECT * FROM clientes;
```

Ejecuta el script.

Te sale un error, fíjate en el mensaje del error , te dice que CREATE PROCEDURE tiene que ser la primera instrucción de un lote. Tenemos pues que cerrar el lote abierto y abrir uno nuevo utilizando GO, y a continuación podremos escribir el CREATE PROCEDURE:

Añade entre el USE y el comentario un GO (el comentario no cuenta como instrucción):

```
USE Gestion8;
GO
-- 1.- Listar los datos de los clientes.
CREATE PROCEDURE ListarClientes
AS
      SELECT * FROM clientes;
```

Ejecuta ahora el script. Ya no da error.

Parece que no ha pasado nada pero se ha hecho lo que hemos pedido, en la base Gestion8 se ha creado el procedimiento ListarClientes.

Para comprobarlo, en el ***Explorador de objetos*** despliega ***Gestion8***, despliega la carpeta ***Programación***, dentro de esta la carpeta ***Procedimientos almacenados*** y verás el procedimiento que acabamos de crear. Se llama ***dbo.ListarClientes***, recuerda que dbo es el nombre del esquema donde se ha creado el procedimiento y donde tenemos también las tablas.
Dentro del procedimiento tienes una carpeta ***Parámetros*** que si la despliegas te muestra los parámetros, en este caso el procedimiento no tiene ninguno y sólo nos indica que el procedimiento devuelve un entero (lo que todos, si no lo recuerdas repasa el apartado sobre RETURN).

Vuelve a la consulta (el script) y ejecútala de nuevo.
Te sale otro error, te indica que ya hay un objeto con ese nombre. Cuando intenta ejecutar el CREATE PROC ya hay un procedimiento con ese nombre y da error.

Para que podamos ejecutar el script las veces que queramos vamos a añadir la indicación de que si el procedimiento ya existe lo elimine antes de hacer el CREATE. En seudocódigo sería:

"Si el procedimiento ListarClientes existe entonces elimínalo", que en SQL se tradiciría por:

```
IF object_id('ListarClientes', 'P') IS NOT NULL DROP PROC ListarClientes;
```

Recuerda, la función object_id (la vimos en el apartado del IF…ELSE) tiene dos parámetros de entrada, el nombre del objeto y su tipo y me devuelve el id (código) de

ese objeto dentro del servidor, y NULL si el objeto no existe. No nos interesa el código pero sí el valor NULL que indica que el objeto no existe.
El IF se leería: Si el ID del procedimiento ListarClientes no es nulo entonces elimina el procedimiento ListarClientes.

Añade esta línea delante del GO. Y ejecuta el script, verás que ahora no da error. Lo que hemos hecho hasta ahora no parece tener mucho sentido, si lo hemos creado, ¿para qué borrarlo y crearlo de nuevo?

Porque ahora voy a añadir las instrucciones necesarias para comprobar que el procedimiento funciona. Una cosa es que lo hayamos creado sin problemas y otra cosa es que lo que hayamos definido haga lo que queremos que haga.
Si al probar el procedimiento nos da error o no hace lo que queremos tendremos que rectificar el procedimiento sobre ese mismo script y volver a crearlo con las modificaciones incorporadas, por eso nos vendrá bien poder ejecutar varias veces el script y que no falle cuando el procedimiento ya exista.

Ahora vamos a añadir la prueba del procedimiento para ver si funciona bien. Para eso hay que llamarlo y en este caso como no tiene parámetros la llamada será:

```
EXEC ListarClientes  -- Lo probamos
```

Añade la instrucción después de la SELECT y ejecuta el script.

No pasa nada. ¿Pero, no debería enseñar el resultado de la SELECT si le pido que ejecute el procedimiento?

Sí, pero todavía no le has pedido que lo ejecute. ¿Por qué?

Has escrito la llamada (EXEC) después del SELECT pero no habías indicado que la definición del procedimiento había acabado así que se ha interpretado como que el EXEC formaba parte de la definición del procedimiento. Como se estaba definiendo el procedimiento no se ha ejecutado el EXEC.

Ahora selecciona únicamente la línea EXEC ListarClientes y ejecuta.

¿Qué ocurre? Te empieza a sacar varias veces el resultado de la SELECT y al final te da otro error: “Se superó el nivel máximo de anidamiento de vistas, procedimientos almacenados, funciones o desencadenadores (límite: 32)”.

¿Qué ha pasado?

Como en la definición del procedimiento está incluída la llamada, cuando hemos ejecutado por primera vez el procedimiento se ha ejecuta la SELECT y el EXEC, con lo cual se ha vuelto a ejecutar la SELECT y el EXEC y así sucesivamente hemos entrado en un bucle infinito. Sería un bucle infinito porque no acabaría nunca, pero SQL Server tiene su protección contra este tipo de fallos, y al cabo de 32 llamadas recursivas (llamada a un procedimiento dentro del mismo) el sistema da un error y aborta el procedimiento.

Así que tenemos que indicar que el procedimiento acaba después de la SELECT, añade un GO después.

Ejecuta el script, ahora no da error y nos aparece en la pestaña resultados el resultado de la SELECT.

Con esto hemos llegado a tener en el script las instrucciones necesarias para crear el procedimiento, a la vez probar que funciona y el script funcionará siempre, exista o no previamente el procedimiento.

Nota. Alguien podría pensar (sobre todo si tiene experiencia previa en programación) y si encierro la SELECT entre un BEGIN y un END para delimitar la definición del procedimiento, el EXEC no estaría incluido en la definición del procedimiento. Pues no, el BEGIN… END permite delimitar bloques de código pero lo que cierra el procedimiento es el GO o el final del script.

2.- Ahora añade el código necesario para crear el procedimiento *ListarEmpleados* que liste los datos de los empleados, añadiendo también todo lo necesario para que se pueda ejecutar varias veces y las instrucciones para probar que funciona.

Intenta hacerlo sól@, al final de este archivo tienes la solución.

3.- Ahora vamos a añadir el código necesario para crear el procedimiento *PedidosCli* que liste los pedidos de un determinado cliente, añadiendo también todo lo necesario para que se pueda ejecutar varias veces y las instrucciones para probar que funciona.

Ahora nos aparece un parámetro de entrada para guardar y pasar al procedimiento el código del cliente "determinado".

La instrucción para listar los pedidos de un determinado cliente por ejemplo el 21003 sería: SELECT * FROM Pedidos WHERE clie=21003, luego la instrucción para listar los pedidos de un cliente cuyo código tenemos en la variable @cli (por ejemplo) sería: SELECT * FROM Pedidos WHERE clie=@cli, esa es la instrucción que escribiremos dentro del procedimiento y @cli será el parámetro de entrada.
Escribe:

```
-- 3.- Listar los pedidos de un determinado cliente

IF object_id('PedidosCli', 'P') IS NOT NULL DROP PROC PedidosCli;
GO
CREATE PROCEDURE PedidosCli @cli INT
AS
      SELECT * FROM pedidos WHERE clie=@cli;
GO
-- Lo probamos, cuando probamos un  programa debemos probarlo en distintas
situaciones para comprobar que funciona en todas.
EXEC PedidosCli 2103 -- Un cliente que tiene varios pedidos
EXEC PedidosCli 2105 -- Un cliente que no tiene pedidos
EXEC PedidosCli 2200 -- Un cliente que no existe
EXEC PedidosCli      -- Si no indicamos un valor para el parámetro da error
```

Aquí hemos puesto varios EXEC porque para probar si un programa funciona no basta con probarlo una vez para una situación concreta sino para todas las situaciones diferentes posibles (o al menos un máximo). Como ahora el procedimiento tiene un parámetro de entrada las situaciones distintas que se pueden producir son que el cliente tenga varios pedidos (para comprobar que saca todos los pedidos del cliente), para un cliente que no tenga pedidos (para ver qué pasa en este caso), para un cliente que no existe y si no se da ningún valor de parámetro en la llamada.

4.- Crea un procedimiento PedidosRep para listar los pedidos de un determinado empleado. Este, puedes intentar hacerlo tú sól@.

Para probar el procedimiento piensa en las posibles situaciones que podemos tener y busca en las tablas empleados que cumplan esas situaciones para probar el procedimiento.
Intenta hacerlo sól@, al final de este archivo tienes la solución.

5.- Crea un procedimiento OficinasReg para listar las oficinas de una determinada region. Ahora lo que varía es el tipo de datos del parámetro, la región es cadena.
Intenta hacerlo sól@, al final de este archivo tienes la solución.

6.- Crea un procedimiento ClientesRep para listar los datos de los clientes asignados a un determinado representante.
Hazlo tú sól@. Al final de este archivo tienes la solución.

7.- Crea un procedimiento PedidosCliRep para listar los pedidos de un determinado cliente realizados por un determinado representante.
Ahora tenemos dos parámetros, uno para indicar el cliente y otro para indicar el representante.

Intenta hacerlo sól@, al final de este archivo tienes la solución.

8.- Crea un procedimiento PedidosCliFab para listar los pedidos en los que un determinado cliente ha comprado productos de un determinado fabricante.

Cuidado con el tipo de datos del fabricante, es texto.
Hazlo tú sól@, al final de este archivo tienes la solución.

9.- Crea un procedimiento Subordinados para listar los subordinados de un empleado determinado, si no se indica el empleado se entenderá el 104.
Los subordinados del 104 son los empleados que lo tienen como jefe.

Ahora nos aparece un parámetro opcional y el valor por defecto es 104 porque nos dicen “si no se indica el empleado se entenderá el 104”.

Intenta hacerlo sól@, al final de este archivo tienes la solución.

10.- Crea un procedimiento EmpleadosOficina para listar los empleados de una determinada oficina, si no se indica la oficina se entenderá la 12.

Hazlo tú sól@, al final de este archivo tienes la solución.

SOLUCIONES

```
-- 2.- Listar los datos de los empleados.
IF object_id('ListarEmpleados', 'P') IS NOT NULL DROP PROC ListarEmpleados;
GO
CREATE PROCEDURE ListarEmpleados
AS
      SELECT * FROM empleados;
GO
EXEC ListarEmpleados  -- Lo probamos

-- 4.- Listar los pedidos de un determinado representante
El lo mismo que el ejercicio anterior pero ahora seleccionamos por rep.
IF object_id('PedidosRep', 'P') IS NOT NULL DROP PROC PedidosRep;
GO
CREATE PROCEDURE PedidosRep @emple INT
AS
      SELECT * FROM pedidos WHERE rep=@emple;
GO
-- Lo probamos.
EXEC PedidosRep 101 -- Un empleado que tiene varios pedidos
EXEC PedidosRep 111 -- Un empleado que no tiene pedidos
EXEC PedidosRep 200 -- Un empleado que no existe

-- 5.- Listar las oficinas de una determinada region
IF object_id('oficinasReg', 'P') IS NOT NULL DROP PROC OficinasReg;
GO
CREATE PROCEDURE OficinasReg @reg CHAR(15)
AS
      SELECT * FROM Oficinas WHERE region=@reg;
GO
-- Lo probamos.
EXEC OficinasReg 'Este'     -- Una región que tiene varias oficinas
EXEC OficinasReg 'NARTE'    -- Una región que no existe

-- 6.- Listar los datos de los clientes de un determinado representante.
IF object_id('Clientesrep', 'P') IS NOT NULL DROP PROC Clientesrep;
GO
CREATE PROCEDURE Clientesrep @rep INT
AS
      SELECT * FROM clientes WHERE repclie=@rep;
GO
EXEC Clientesrep 101 -- Un representante que tiene varios clientes
EXEC Clientesrep 112 -- Un representante que no tiene clientes
EXEC Clientesrep 300 -- Un representante que no existe
```

```
-- 7.- Listar los pedidos de un determinado cliente realizados por un determinado
representante.
-- Ahora tenemos dos parámetros, uno para indicar el cliente y otro para indicar el
representante
IF object_id('PedidosCliRep', 'P') IS NOT NULL DROP PROC PedidosCliRep;
GO
CREATE PROCEDURE PedidosCliRep @cli INT, @rep INT
AS
      SELECT * FROM pedidos WHERE clie=@cli AND rep=@rep;
GO
EXEC PedidosCliRep 2108, 109  -- Un cliente que tiene varios pedidos, del
representante y otros pedidos de otros rep.
EXEC PedidosCliRep 2103, 109 -- Un cliente que no tiene pedidos del representante
pero sí de otros
EXEC PedidosCliRep 2105, 109 -- Un cliente que no tiene pedidos
EXEC PedidosCliRep 2200, 109 -- Un cliente que no existe
-- Aquí habría más variantes, pero con estas pruebas es suficiente.

-- 8.- Listar los pedidos en los que un determinado cliente ha comprado productos de
un determinado fabricante.
-- Cuidado con el tipo de datos del fabricante, es texto

IF object_id('PedidosCliFab', 'P') IS NOT NULL DROP PROC PedidosCliFab;
GO
CREATE PROCEDURE PedidosCliFab @cli INT, @fab CHAR(5)
AS
      SELECT * FROM pedidos WHERE clie=@cli AND fab=@fab;
GO
EXEC PedidosCliFab 2111, 'aci'  -- Un cliente que tiene varios pedidos, del
fabricante y otros pedidos de otros fabricantes.
EXEC PedidosCliFab 2112, 'aci'  -- Un cliente que no tiene pedidos del fabricante
pero sí de otros
EXEC PedidosCliFab 2105, 'aci'  -- Un cliente que no tiene pedidos
EXEC PedidosCliFab 2200, 'aci'  -- Un cliente que no existe

-- 9.- Listar los subordinados de un empleado determinado, si no se indica el
empleado se entenderá el 104.
-- Los subordinados del 104 son los empleados que lo tienen como jefe.
-- Ahora nos aparece un parámetro opcional y el valor por defecto es 104 porque nos
dicen “si no se indica el empleado se entenderá el 104”.

IF object_id('Subordinados', 'P') IS NOT NULL DROP PROC Subordinados;
GO
CREATE PROCEDURE Subordinados @rep INT = 104
AS
      SELECT * FROM empleados WHERE jefe=@rep;
GO
EXEC Subordinados   -- Para comprobar que funciona sin parámetros, comprueba que los
que salen tienen como jefe el 104 (el valor `por defecto)
EXEC Subordinados 106 -- Un empleado que tiene varios subordinados
EXEC Subordinados 105 -- Un empleado que no tiene subordinados
EXEC Subordinados 300 -- Un empleado que no existe

-- 10.- Listar los empleados de una determinada oficina, si no se indica la oficina
se entenderá la 12.
IF object_id('EmpleadosOficina', 'P') IS NOT NULL DROP PROC EmpleadosOficina;
GO
CREATE PROCEDURE EmpleadosOficina @ofi INT = 12
AS
      SELECT * FROM empleados WHERE oficina=@ofi;
GO
EXEC EmpleadosOficina     -- Para comprobar que funciona sin parámetros
EXEC EmpleadosOficina 21  -- Una oficina que tiene varios empleados
EXEC EmpleadosOficina 24  -- Una oficina que no tiene empleados
EXEC EmpleadosOficina 300 -- Una oficina que no existe
```