


| | |
|---|--|
|  | UNIVERSIDAD DON BOSCO FACULTAD DE ESTUDIOS TECNOLOGICOS ESCUELA DE COMPUTACION |
| CICLO 1-2016 | CLASE Nº11 |
| | Tema: Vistas, Implementación de Procedimientos almacenados y desencadenadores Materia: Base de datos Docente: Blanca Iris Cañas |

Implementación de estructuras repetitivas en un procedimiento almacenado

1. Hacer uso de la base de datos **Northwind**
2. En los ejercicios de este punto cambiar en todos los ejercicios la palabra SuCarnet por su número de carnet
3. Crear el siguiente procedimiento almacenado:

```

CREATE PROCEDURE Mostrar_10_pedidos_SuCarnet
AS
DECLARE @contador int
DECLARE @num int

SET @contador=0
--Obteniendo el primer valor del campo OrderID de la tabla Orders
SET @num=(SELECT TOP 1 OrderID FROM Orders ORDER BY OrderID)

--Evalua si el contador es menor que 10, si la condicion se cumple
--realiza la instruccion SELECT
WHILE @contador<10
BEGIN
    SELECT OrderID, OrderDate FROM Orders WHERE OrderID=@num+@contador
    --Se incrementa el contador
    SET @contador=@contador+1
END

```

4. Ejecutamos el procedimiento almacenado

```

--Ejecutamos el procedimiento almacenado
EXEC Mostrar_10_pedidos_SuCarnet

```

5. En el ejemplo siguiente, si el precio de venta promedio de un producto es inferior a \$300, se realiza dentro del bucle WHILE la acción de duplicar los precios y, a continuación, selecciona el precio máximo. Si el precio máximo es menor o igual que \$500, el bucle WHILE se reinicia y se vuelve a duplicar los precios. El bucle realiza la operación hasta que el precio máximo sea mayor a \$500, después de lo cual finaliza el bucle WHILE.

```

CREATE PROCEDURE Actualizar_precio_SuCarnet
AS
    WHILE (SELECT AVG(UnitPrice) FROM Products) < 300
    BEGIN
        UPDATE Products
        SET UnitPrice = UnitPrice * 2
        SELECT MAX(UnitPrice) AS [Precio Máximo] FROM Products
        IF (SELECT MAX(UnitPrice) FROM Products) < 500
            BREAK
        --Sale del bucle más interno en una instrucción WHILE o una instrucción IF...ELSE
        --dentro de un bucle WHILE
        ELSE
            CONTINUE
        --Reinicia un bucle WHILE. Las instrucciones que se encuentren después de la
        --palabra clave CONTINUE se omiten
    END
    --Aquí terminar el código del procedimiento

```

6. Antes de ejecutar el procedimiento se debe verificar la información de la tabla Products, por ejemplo al ejecutar la consulta:

```

SELECT UnitPrice FROM Products
ORDER BY UnitPrice DESC

```

7. Se obtienen los siguientes resultados (estos pueden variar, tomar en cuenta los resultados obtenidos en la consulta)

| | UnitPrice |
|---|-----------|
| 1 | 316.20 |
| 2 | 123.79 |
| 3 | 97.00 |
| 4 | 81.00 |
| 5 | 62.50 |
| 6 | 55.20 |

8. Ahora ejecutar el procedimiento almacenado

```

--Ejecutamos el procedimiento
EXECUTE Actualizar_precio_SuCarnet

```

9. Ahora verifique los resultados, ejecutando la consulta del punto 5, si no hay cambios en los datos, modifique el procedimiento almacenado por la siguiente información:

```

WHILE (SELECT AVG(UnitPrice) FROM Products) < 500
BEGIN
    UPDATE Products
    SET UnitPrice = UnitPrice * 2
    SELECT MAX(UnitPrice) AS [Precio Máximo] FROM Products
    IF (SELECT MAX(UnitPrice) FROM Products) < 3000
        BREAK

```

10. Ejecute de nuevo el procedimiento almacenado y verifique los resultados
11. En el siguiente ejemplo se desea mostrar los nombres de los clientes que se encuentran en la tabla Customers

```
CREATE PROC Mostrar_Clientes_SuCarnet
AS
    DECLARE @Nombre NVARCHAR(40)
    --Se declara el cursor @cursor, el cual se utilizara para recorrer
    --cada resultado de la consulta SELECT
    DECLARE @cursor CURSOR

    --Se asigna el primer dato al cursor
    SET @cursor = CURSOR FOR
    SELECT CompanyName FROM Customers
    --Abrir el cursor
    OPEN @cursor
    --Recupera las filas del cursor
    FETCH NEXT
    FROM @cursor INTO @Nombre
    WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT 'El nombre del cliente es: ' + @Nombre
        --Se mueve al siguiente registro
        FETCH NEXT FROM @cursor INTO @Nombre
    END
    --Este comando hace desaparecer el puntero sobre el registro actual
    CLOSE @cursor
    DEALLOCATE @cursor
GO
```

12. Ejecutar el procedimiento almacenado

```
--Ejecutamos el procedimiento almacenado
EXEC Mostrar_Clientes_SuCarnet
```

Parte 3: Uso de desencadenadores

1. En la base de datos Norhtwind, se crearan los siguientes TRIGGERS o desencadenadores

```
CREATE TRIGGER Disp_SEGURIDAD_SuCarnet
ON DATABASE FOR DROP_TABLE, ALTER_TABLE
AS
    BEGIN
        --RAISERROR se usa para devolver mensajes a las aplicaciones con el
        --mismo formato que un error del sistema
        RAISERROR ('No está permitido borrar ni modificar tablas!' , 16, 1)
        --16 Severidad
        --1 Estado
        ROLLBACK TRANSACTION
    END
GO
```

Nota:

ROLLBACK TRANSACTION: Revierte una transacción explícita o implícita hasta el inicio de la transacción o hasta un punto de retorno dentro de la transacción.

2. Crear la siguiente tabla:

```
--Crear la tabla
CREATE TABLE prueba_SuCarnet
(campo1 int
)
```

3. Ejecutar la siguiente instrucción:

```
--Eliminar la tablas
DROP TABLE prueba_SuCarnet
```

4. Verifique los resultados, debe de mostrar el mensaje indicando de que no se puede eliminar ni modificar una tabla en la base de datos, el disparador se activa en el momento que un usuario desee: eliminar o modificar una tabla. Por eso es que no se puede eliminar la tabla prueba_SuCarnet

5. Crear el siguiente TRIGGER a la tabla prueba_SuCarnet

```
CREATE TRIGGER Mensaje_Insercion_SuCarnet
ON prueba_SuCarnet
AFTER INSERT
AS
    BEGIN
    PRINT 'Se agregó un nuevo registro a la tabla'
    ROLLBACK TRANSACTION
    END
```

6. Agregar un dato a la tabla

```
--Verificando el TRIGGER
INSERT INTO prueba_SuCarnet VALUES (1)
```

Cada vez que se agregue un dato a la tabla se mostrara el mensaje indicando que se ha agregado un nuevo dato a la tabla

7. Agregar un nuevo dato a la tabla Employees (el dato a ingresar son su apellido y nombre)

```
--Agregando un nuevo dato a la tabla Employees
INSERT INTO Employees(Lastname,Firstname) VALUES ('Cañas', 'Blanca')
```

8. Crear el siguiente TRIGGER

```

CREATE TRIGGER actualizar_emple_SuCarnet
ON Employees
AFTER UPDATE
AS
    IF UPDATE(Lastname)
    BEGIN
        PRINT 'Se realizo un cambio en el apellido de los empleados'
    END

```

9. Antes de verificar el TRIGGER, realice la consulta para conocer el código del empleado (EmployeeID) que se agregó en el punto 7

```

--Verificando los datos de los empleados
SELECT * FROM Employees

```

| | EmployeeID | LastName | FirstName | Title |
|----|------------|----------|-----------|-------|
| 11 | 11 | Cañas | Blanca | NULL |

10. Comprobar el TRIGGER realizando una actualización en el apellido del empleado donde el código del empleado sea igual al que obtuvo en la consulta anterior

```

--Actualizando el apellido del empleado
UPDATE Employees SET LastName='Abarca'
WHERE EmployeeID=11

```

11. Verificar los resultados de la consulta y del TRIGGER
12. Guardar los cambios
13. Agregar un nuevo empleado a la tabla Employees
14. Crear el siguiente procedimiento almacenado

```

CREATE TRIGGER eliminar_emple_SuCarnet
ON employees
FOR DELETE
AS
    IF(SELECT COUNT(*) FROM deleted)>1
    BEGIN
        PRINT 'No se puede borrar mas de un empleado al mismo tiempo'
        ROLLBACK TRANSACTION
    END

```

La tabla DELETED almacena copias de las filas afectadas por las instrucciones DELETE y UPDATE. Durante la ejecución de una instrucción DELETE o UPDATE, las filas se eliminan de la tabla del desencadenador (TRIGGER) y se transfieren a la tabla DELETED.

15. Ejecutar la siguiente consulta para eliminar más de un registro de la tabla Employees

```
--La instrucción DELETE activa el desencadenador y evita la transacción.  
DELETE FROM Employees WHERE EmployeeID > 10
```

16. Ahora ejecutar la siguiente consulta

```
--La instrucción DELETE activa el desencadenador y permite la transacción.  
DELETE FROM Employees WHERE EmployeeID = 11
```

17. Verificar los resultados de la consulta

Tomar un dato de la columna Employees, por ejemplo uno de los códigos de empleado que acaba de agregar

18. Guardar los cambios