TRANSACT SQL

PROCEDIMIENTOS ALMACENADOS

PROCEDIMENTOS ALMACENADOS

Un procedimiento almacenado (STORE PROCEDURE) está formado por un conjunto de instrucciones en Transact-SQL que definen un determinado proceso, dicho procedimiento puede aceptar parámetros de entrada y devolver un valor o conjunto de resultados. Este procedimiento se guarda en el servidor y puede ser ejecutado en cualquier momento.

Los procedimientos almacenados se diferencian de las instrucciones SQL ordinarias y de los lotes de instrucciones SQL en que están pre-compilados. La primera vez que se ejecuta un procedimiento, el procesador de consultas de SQL server lo analiza y prepara un plan de ejecución que se almacena en una tabla del sistema. De esta manera los procedimientos almacenados se ejecutan de una forma casi instantánea.

Procedimientos almacenados con parámetros en SQL Server

Sintaxis:

Create Procedure NombreProcedimiento [@NombreParametro1 TipoDato1, …,

NombreParametroN TipoDatoN] **As**

Instrucciones

Posteriormente lo ejecutamos introduciendo un valor/res correspondiente al parámetro/os introducido/s.

Exec Procedimiento Valor1, .., ValorN

Ejemplo 1

```
CREATE PROCEDURE NUMEMPDEPT @NDEP SMALLINT
AS
SELECT COUNT(*) AS [NUM EMPLEADOS], DEPT_NO
FROM EMP
WHERE DEPT_NO = @NDEP
GROUP BY DEPT_NO
```

Crea el procedimiento numempdept con el parámetro ndep que es tipo smallint. El procedimiento hace una select que realiza grupos por número de deparamento y cuenta cuantos empleados hay en cada departamento.

```
EXEC NUMEMPDEPT 20
```

Ejecuta el procedimiento introduciendo el valor 20 para el parámetro ndep, con lo que haría la select sobre el departamento 20.

Ejemplo 2

```
CREATE PROCEDURE BUSCAEMP,@NOFICIO NVARCHAR(19))
AS
SELECT * FROM EMP
WHERE DEPT_NO = @NDEP
AND OFICIO = @NOFICI

EXEC BUSCAEMP 20,'Empleado'
```

MODIFICAR UN PROCEDIMIENTO ALMACENADO

Sintaxis:

ALTER Procedure NombreProcedimiento [@NombreParametro1 TipoDato1, …, NombreParametroN TipoDatoN] **As**

Instrucciones

Posteriormente lo ejecutamos introduciendo un valor/res correspondiente al parámetro/os introducido/s.

Exec Procedimiento Valor1, .., ValorN

ELIMINAR UN PROCEDIMIENTO ALMACENADO

DROP PROCEDURE NombreProcedimiento

Si quisiéramos eliminar más de un procedimiento de una vez, lo haríamos listando los nombres de los procedimientos a borrar separados por comas.

```
-- ELIMINAR EL PROCEDIMIENTO ALMACENADO
DROP PROCEDURE BORRAREMP
```

NOTA: Transact-SQL permite abreviar la palabra reservada **PROCEDURE** por **PROC**

Obviamente, no podemos crear un procedimiento o modificarlo y ejecutarlo en el mismo script.

PROCEDIMIENTOS ALMACENADOS CON PARÁMETROS CON VALORES POR DEFECTO

Sintaxis:

Create Procedure Nombre @Variable **tipo** = Valor
As
Instrucciones

Donde Valor es el valor que le damos por defecto, este valor puede almacenar comodines (%).

Ejemplo 1

```
-- Crea un procedimiento con el parámetro ndept y le da por defecto valor 10.
CREATE PROCEDURE NUMEMP @NDEPT SMALLINT = 10
AS
SELECT DEPT_NO, COUNT(*) AS [NUM EMPLEADOS]
FROM EMP WHERE DEPT_NO = @NDEPT
GROUP BY DEPT_NO

EXEC NUMEMP
```

Ejemplo 2

Debe devolver salario, oficio y comisión y del empleado que le pasamos el apellido.

```
CREATE PROCEDURE SALARIOOFICIO @PAPELLIDO VARCHAR(20) = 'REY'
AS
SELECT OFICIO, SALARIO, COMISION
FROM EMP WHERE APELLIDO = @PAPELLIDO

EXEC SALARIOOFICIO 'GARCIA'
```

Sacaría el salario, el oficio y la comisión de todos los que tengan apellido Garcia, sino pusiésemos parámetro, por defecto sacaría los que tuviesen apellido Rey.

Ejemplo 3

```
--  Procedimiento ejemplo del uso de caracteres comodín %
CREATE PROCEDURE SALARIOOFICIOLIKE (@PAPELLIDO VARCHAR(20) = 'REY' )
AS
SELECT OFICIO, SALARIO, COMISION
FROM EMP
WHERE APELLIDO LIKE '%' + @PAPELLIDO + '%'

EXEC SALARIOOFICIOLIKE 's'
```

Mostraría oficio, salario y comisión de los empleados que tuviesen una ‘s’ en su apellido.

Ejemplo 4

Procedimiento que al introducir oficio y salario, debe sacar el oficio de los empleados que tengan el mismo oficio pasado como parámetro y además ganen más del salario indicado. Debemos hacer que sino introduce nada saque todos los registros.

```
CREATE PROCEDURE DOSPARAMETROS2(
@OFICIO VARCHAR(12) = '%',
@SALARIO INTEGER = 0)
AS
SELECT APELLIDO FROM EMP WHERE OFICIO LIKE @OFICIO AND SALARIO > @SALARIO

EXEC DOSPARAMETROS2 'Empleado',1000
```

Ejemplo 5

Sacar todos los empleados que se dieron de alta entre una determinada fecha inicial y fecha final y que pertenecen a un determinado departamento.

```
CREATE PROCEDURE TRESPARAMETROS(
@FINICIAL DATETIME = '01-01-1980',
@FFINAL SMALLDATETIME = '12-07-2002' ,
@DEPT_NO NVARCHAR(10) = '%'
)
AS
SELECT * FROM EMP WHERE FECHA_ALT BETWEEN @FINICIAL AND @FFINAL AND CAST(DEPT_NO AS
NVARCHAR(10)) LIKE @DEPT_NO

EXEC TRESPARAMETROS '01-01-1980','12-07-2002',’30’
```

Ejemplo 6

Crear procedimiento que inserte un empleado. Crear otro procedimiento que borre un empleado que coincida con los parámetros indicados (los parámetros serán todos los campos de la tabla empleado)

```
CREATE PROCEDURE [INSERTA EMPLEADO](
@EMP_NO INT, @APELLIDO NVARCHAR(20), @OFICIO NVARCHAR(20), @DIR INT, @FECHA_ALT
SMALLDATETIME, @SALARIO INT, @COMISION INT, @DEPT_NO INT)
AS
INSERT INTO EMP VALUES (@EMP_NO, @APELLIDO, @OFICIO, @DIR, @FECHA_ALT, @SALARIO,
@COMISION, @DEPT_NO)


EXEC [INSERTA EMPLEADO] 7855,'ALIAGA','EMPLEADO',7782,'23/05/2015',45000,0,30


CREATE PROCEDURE [BORRA EMPLEADO](
@EMP_NO INT, @APELLIDO NVARCHAR(20), @OFICIO NVARCHAR(20), @DIR INT, @FECHA_ALT
SMALLDATETIME, @SALARIO INT, @COMISION INT, @DEPT_NO INT)
AS
DELETE FROM EMP WHERE EMP_NO = @EMP_NO AND APELLIDO = @APELLIDO AND OFICIO = @OFICIO
AND DIR = @DIR AND FECHA_ALT = @FECHA_ALT AND SALARIO = @SALARIO AND COMISION =
@COMISION AND DEPT_NO = @DEPT_NO

EXEC [BORRA EMPLEADO] 7855,'ALIAGA','EMPLEADO',7782,'23/05/2015',45000,0,30
```

Variables en procedimientos almacenados de SQL Server

Siempre que necesitemos alguna variable dentro de un procedimiento almacenado tendremos que declararla, en este sentido, la declaración y asignación de valores a variables en procedimientos es exactamente igual que cuando hacemos scripts. Recordemos

Sintaxis de declaración

Declare @nombre tipo_dato

Sintaxis de asignación directa

Set @nombre = valor

Sintaxis de asignación mediante consulta

Select @nombre = campo from tabla

VARIABLES DE SALIDA EN PROCEDIMIENTOS ALMACENADOS

Hasta ahora habíamos utilizado solamente variables de entrada como parámetros en los procedimientos almacenados, ahora aprenderemos a utilizar variables de salida, podemos usar tantas como queramos.

Sintaxis

Create Procedure Nombre @Variable tipodedato **Output**
as
Instrucciones
Print @Variable

Ejemplo 1:

```
CREATE PROCEDURE TOTALSALCOM @APELLIDO NVARCHAR(25), @TOTAL INT OUTPUT
--CREAMOS EL PROCEDIMIENTO CON UN PARÁMETRO Y UNA VARIABLE DE SALIDA
AS
--DECLARAMOS DOS VARIABLES PARA ALMACENAR VALORES
DECLARE @SAL INT
DECLARE @COM INT
--ASIGNAMOS LOS VALORES CORRESPONDIENTES A LAS VARIABLES Y DESPUÉS LAS SUMAMOS Y LAS GUARDAMOS EN LA VARIABLE DE SALIDA.
SELECT @SAL = SALARIO FROM EMP WHERE APELLIDO = @APELLIDO
SELECT @COM = COMISION FROM EMP WHERE APELLIDO = @APELLIDO
SET @TOTAL = @SAL + @COM
--DEVOLVEMOS EL VALOR DE LA VARIABLE QUE QUEREMOS

PRINT @TOTAL
```

Para ejecutarlo en el analizador de consultas, hemos de declarar la variable que vamos a recuperar también, para almacenar el valor que nos pasa el procedimiento almacenado en ella. Es decir, en este caso @total almacenará el valor pasado por la variable del mismo nombre que hay en el procedimiento:

```
DECLARE @TOTAL INT
EXEC TOTALSALCOM 'JIMENEZ', @TOTAL OUTPUT
```

Ejemplo 2:

```
CREATE PROCEDURE TOTALES
@NDEPT INT = NULL, @TOTAL INT OUTPUT
AS
IF @NDEPT IS NULL
SELECT @TOTAL = SUM(SALARIO) FROM EMP
ELSE
SELECT @TOTAL = SUM(SALARIO) FROM EMP WHERE DEPT_NO = @NDEPT
SELECT @TOTAL
-- Llamada al procedimiento y ejecución
DECLARE @TOTAL INT
EXEC TOTALES 10, @TOTAL OUTPUT
```

NOTA:

Siempre es deseable que las instrucciones del procedimiento estén dentro de un bloque **TRY** CATCH y controladas por una transacción, especialmente en los procedimientos que modifican datos de la BD.

Ejemplo 6 con TRY CATCH y TRANSACCIONES

Ahora modificaos el procedimiento del ejercicio 6 que inserta un empleado. También modificamos el otro procedimiento que borra un empleado que coincida con los parámetros indicados (los parámetros serán todos los campos de la tabla empleado)

```
ALTER PROCEDURE [INSERTA EMPLEADO](
@EMP_NO INT, @APELLIDO NVARCHAR(20), @OFICIO NVARCHAR(20), @DIR INT, @FECHA_ALT
SMALLDATETIME, @SALARIO INT, @COMISION INT, @DEPT_NO INT)
AS
BEGIN TRY
        BEGIN TRAN
                INSERT INTO EMP VALUES (@EMP_NO, @APELLIDO, @OFICIO, @DIR, @FECHA_ALT,
        @SALARIO, @COMISION, @DEPT_NO)
        COMMIT
END TRY
BEGIN CATCH
        ROLLBACK
        PRINT ERROR_MESSAGE()
END CATCH
```

```
EXEC [INSERTA EMPLEADO] 7855,'ALIAGA','EMPLEADO',7782,'23/05/2015',45000,0,30
```

```
CREATE PROCEDURE [BORRA EMPLEADO](
@EMP_NO INT, @APELLIDO NVARCHAR(20), @OFICIO NVARCHAR(20), @DIR INT, @FECHA_ALT
SMALLDATETIME, @SALARIO INT, @COMISION INT, @DEPT_NO INT)
AS
BEGIN TRY
        BEGIN TRAN
                DELETE FROM EMP WHERE EMP_NO = @EMP_NO AND APELLIDO = @APELLIDO AND
        OFICIO = @OFICIO AND DIR = @DIR AND FECHA_ALT = @FECHA_ALT AND SALARIO = @SALARIO
        AND COMISION = @COMISION AND DEPT_NO = @DEPT_NO
        COMMIT
END TRY
BEGIN CATCH
        ROLLBACK
        PRINT ERROR_MESSAGE()
END CATCH


EXEC [BORRA EMPLEADO] 7855,'ALIAGA','EMPLEADO',7782,'23/05/2015',45000,0,30
```