Procedimientos almacenados o Stored Procedures:

Conjunto de sentencias almacenadas en el servidor y nos permite realizar tareas repetitivas de forma más sencilla. No pueden contener:

* Create procedure
* Create default
* Rule
* Triggers
* Views

Existen varios tipos:

* Del sistema: almacenados en la base de datos "master" y llevan el prefijo "sp\_";
* Locales: los crea el usuario;
* Temporales: pueden ser locales, su nombre comienza con "#";
* Extendidos: se implementan como bibliotecas de vínculos dinámicos (DLL: Dinamic Link Libraries), pueden ejecutarse fuera del entorno de SQL server.

# Proceso de Creación:

Al crear un Procedimiento, las instrucciones que contiene se analizan para que no haya errores en ellas, una vez analizadas y de no tener errores, SQL guarda el nombre del procedimiento en la tabla de "Sysobject" y su contenido en "Syscomment" en la base de datos activa. Si se encuentran errores el procedimiento no se crea.

## Ventajas:

Entre otras:

* Comparte la lógica de aplicación con las otras aplicaciones, por lo que la modificación de datos se puede hacer desde un solo sitio;
* Permite realizar las operaciones que el usuario necesita, evitando que tengan acceso directo a las tablas (se muestra la información que el desarrollador considera necesaria, protegiendo los datos privados);
* Reducción del tráfico de red; se envía el Procedimiento lo cual ahorra enviar un conjunto de sentencias declaradas independientes entre ellas.

## Creación:

Primero se crea la tabla sobre la que se va a trabajar, en este caso tabla Alumnos, luego se crea el Procedimientos en una sentencia con las palabras reservadas: "CREATE PROCEDURE" o "CREATE PROC"; luego el nombre que decidimos darle al mismo, "AS" y las sentencias que queremos q estén dentro de ese Procedimiento, tareas repetitivas que, para evitar estar escribiéndolas varias veces.

## Tabla:

CREATE TABLE Alumnos (

Id INT NOT NULL UNIQUE,

Legajo INT,

Nombre VARCHAR(20),

Apellido VARCHAR(20),

Facultad VARCHAR(20),

CONSTRAINT PK\_Alumno PRIMARY KEY (ID))

GO

## Procedimiento:

CREATE PROC InsertData

@Id INT,

@Legajo INT,

@Nombre VARCHAR(20),

@Apellido VARCHAR(20),

@Facultad VARCHAR(20)

AS

INSERT INTO Alumnos (Id, Legajo, Nombre, Apellido, Facultad)

VALUES (@Id, @Legajo, @Nombre, @Apellido, @Facultad)

GO

Este Procedimiento nos permite ingresar datos en la tabla, para ejecutarlo:

EXEC InsertData 1, 1234, 'Pedro', 'Gonzalez', 'Arquitectura'

GO

Una vez ejecutado, de no tener errores, cumple con su función y el usuario no sabe si esta tabla tiene más datos. Es importante recordar añadir el prefijo junto con el nombre del procedure creado, para poder identificar de que tipo son.

# Parámetros de Entrada:

En los Procedimientos Dinámicos se utilizan los Parametros de Entrada, para condicionar el resultado. La sintaxis es:

CREATE PROCEDURE SelectAllAlumnos @Apellido VARCHAR(20), @Facultad VARCHAR(20)

AS

SELECT \* FROM Alumnos WHERE Apellido = @Apellido AND Facultad = @Facultad

GO

Se utiliza la misma sintaxis para crear un Procedimiento y se le agrega luego del nombre "@" junto con el parámetro y el tipo)(cuando hay más de uno se separan con una coma); la sentencia sigue el curso normal. Esta instrucción crea un Procedimiento que selecciona alumnos que tengan determinado apellido y concurran a determinada facultad. Se ejecuta de la siguiente manera:

EXEC SelectAllAlumnos 'Gonzalez', 'Arquitectura'

GO

También se le pueden asignar valores por defecto junto a los parámetros, al momento de la ejecución de no pasarle ningún parámetro toma por defecto los valores que se asignaron al momento de la creación:

CREATE PROCEDURE SelectAlumnos @Nombre VARCHAR(20) = 'Juan'

AS

SELECT \* FROM Alumnos WHERE Nombre = @Nombre

GO

Al ejecutarlo:

EXEC SelectAlumnos

GO

# Parámetros de Salida:

Al ejecutarse el Procedimiento nos devuelve una variable junto con el resultado. La sintaxis es la misma que de los parámetros de entrada, la única diferencia es que junto a los parámetros y su tipo se le agrega la palabra reservada "OUTPUT"; se pueden crear Procedimientos con parámetros de entrada y salida, juntos:

CREATE PROCEDURE SelectAllAlumnos2 @Apellido VARCHAR(20), @Facultad VARCHAR(20), @Count INT OUTPUT

AS

SET @Count = (SELECT count(@Apellido) FROM Alumnos WHERE Apellido = @Apellido AND Facultad = @Facultad)

GO

Para ejecutarlo declaramos una variable nueva, que nos va a mostrar el total de los alumnos de la lista que coincidan con los parámetros que le pasamos, después de eso la sintaxis es la misma:

DECLARE @Total INT

EXEC SelectAllAlumnos2 'Gonzalez', 'Arquitectura', @Total OUTPUT

SELECT @Total

GO

De no colocar "OUTPUT" la variable total nos devuelve "NULL".

# Return:

Al ejecutar un Procedimiento, al llegar a la línea donde está la palabra reservada "RETURN" automáticamente ignora todas las líneas que vienen adelante. En la sintaxis puede o no contener parámetros de entrada o salida, así como también puede tener estructuras de control, dependiendo de las circunstancias que deseamos evaluar:

CREATE PROCEDURE SelectAllAlumnos3 @Apellido VARCHAR(20), @Facultad VARCHAR(20)

AS

IF (@Apellido is null) AND (@Facultad is null)

RETURN 0

ELSE

RETURN 1

GO

Para ejecutar declaramos la variable que queremos ver en el resultado, luego, la sintaxis cambia en lo siguiente:

DECLARE @Retorno INT

EXEC @Retorno = SelectAllAlumnos3 null, null

SELECT @Retorno

GO

# Manejo de Excepciones:

La sintaxis es la siguiente:

CREATE PROC InsertNewData

@Id INT,

@Legajo INT,

@Nombre VARCHAR(20),

@Apellido VARCHAR(20),

@Facultad VARCHAR(20)

AS

BEGIN TRY

INSERT INTO Alumnos(Id, Legajo, Nombre, Apellido, Facultad)

VALUES (@Id, @Legajo, @Nombre, @Apellido, @Facultad)

END TRY

BEGIN CATCH

PRINT 'No se puede ingresar mas alumnos'

END CATCH

GO

Este Procedimiento intenta insertar valores en una tabla, con manejo de excepciones. Si al momento de ejecutarse encuentra algún tipo de error muestra un mensaje por consola de que "no se pueden ingresar más alumnos" en nuestra tabla de Alumnos. Para ejecutarlo, la sintaxis es la misma:

EXEC InsertNewData 4, 7890, 'Douglas', 'Douglin', 'Derecho'

GO

# Anidación:

Se ejecuta un Procedimiento dentro de otro, primero se crea un Procedimiento, para luego llamarlo desde otro ya almacenado en nuestra base de datos:

CREATE PROCEDURE Total @Resultado INT OUTPUT

AS

SET @Resultado = (SELECT COUNT(Nombre) FROM Alumnos)

GO

Creamos el segundo que llama al primero, el cual solo nos devolvía un solo resultado (tota de nombres de los alumnos):

CREATE PROCEDURE TotalFinal @Total INT OUTPUT

AS

BEGIN

DECLARE @Numero INT

EXEC Total @Numero OUTPUT

SET @Total = @Numero

END

GO

Al ejecutar el segundo procedimiento a la variable en el segundo se le asigna el valor del resultado que devuelve el primer procedimiento, por lo que el segundo nos devuelve la cantidad de nombres que hay en nuestra tabla. La sintaxis de ejecución es la siguiente:

DECLARE @Num INT

EXEC TotalFinal @Num OUTPUT

SELECT @Num

GO

## Eliminar Procedimientos:

Estos Procedimientos se guardan en la base de datos, en la carpeta de Procedimientos Almacenados(Base de Datos/Programación/Procedimientos/Procedimientos Almacenados). Para eliminarlos hay tres formas:

* Se utiliza la sentencia drop más el nombre del procedimiento que se desea eliminar.

DROP PROCEDURE InsertData

GO

* Para evitar que se generen errores al eliminar Procedimientos(eliminar procedimientos que ya no existen), se pueden colocar estructuras de control:

DROP PROCEDURE IF EXISTS InsertData

GO

* Directamente desde la carpeta de procedimientos en la base de datos