



Universidad Nacional de la Patagonia San Juan Bosco
Facultad de Ingeniería

Cátedra: **Bases de datos I**

Stored Procedures, Triggers y Funciones

- **Stored Procedures – Introducción**

- Los Stored procedures (SP) esencialmente son funciones que se pueden crear en la BD para reutilizarlos posteriormente, permitiendo manejar parámetros de entrada y retornar valores como resultado, pero no regresan ningún valor directamente.
- Un SP es, por definición, una porción de código SQL que contiene sentencias declarativas o procedurales. Reside en el catálogo de la BD y se lo puede invocar a través de un **trigger** (disparador), otro SP o desde una aplicación cliente.
- Entonces, un SP puede contener cualquier sentencia SQL como:
 - ✓ INSERT
 - ✓ UPDATE
 - ✓ DELETE
- O incluso comandos de definición de datos, como ser:
 - ✓ CREATE TABLE
 - ✓ ALTER TABLE
 - ✓ Etc
- Adicionalmente se soportan estructuras clásicas como IF, WHILE, etc.


- **SP – Creación**

- Un SP se divide en tres partes:
 - ✓ Nombre del SP
 - ✓ Lista de parámetros
 - ✓ Cuerpo del SP
- Los SP se crean con el comando `CREATE PROCEDURE` y su sintaxis es la siguiente:

```
CREATE PROCEDURE sp_nombre_procedimiento
@param data_type = default_value
AS
-- Sentencias del procedimiento
```

- Ejemplo de un SP sencillo:

```
CREATE PROCEDURE sp_Borrar_Empleado
(@EmployeeId INT)
AS
BEGIN
    DELETE FROM Employees
    WHERE EmployeeId = @EmployeeId;
END
```



Un parámetro definido de
tipo INT

- SP – Invocación y manejo

- Para ejecutar un SP se utiliza el comando EXEC
- En el caso del ejemplo anterior, puede ser invocado de la siguiente manera:

```
EXEC sp_Borrar_Empleado 465
```

- El procedimiento anterior se lo usa para borrar el registro de un empleado, sin embargo, como vimos anteriormente, un SP puede ser usado para insertar registros en una tabla, para actualizar valores, puede realizar una consulta, devolviendo un conjunto de tuplas, etc.
- También es posible correr múltiples consultas, que retornen múltiples conjuntos de resultados con un único SP.

- **SP – Eliminación**

- Para borrar un Stored Procedure se utiliza el comando DROP
- Su sintaxis es:

```
DROP PROCEDURE sp_nombre_stored_procedure
```

- En el caso del ejemplo anterior quedaría de la siguiente manera:

```
DROP PROCEDURE sp_Borrar_Empleado
```

- SP – Ventajas

- **Velocidad:** Se usan para incrementar la performance de una aplicación. Al crear los SP's se compilan y luego se almacenan en el catálogo de la base de datos. Luego, cuando una aplicación cliente los invoca, se los ejecuta más rápido que una sentencia SQL que se envía desde una aplicación cliente, por ejemplo. Esto puede resultar en un **incremento significativo de la performance del sistema.**
- **Reducción de tráfico:** Se reduce significativamente el tráfico en la red entre el servidor de la base de datos y los clientes porque las aplicaciones no tienen que enviar sentencias SQL muy largas y sin compilar para obtener los datos requeridos, solamente se envía la llamada al SP que es significativamente menor.
- **Seguridad:** Se pueden usar los SP para atender aspectos relacionados con la seguridad de la BD porque cada SP puede tener designados sus propios permisos.
- **Reuso de código y abstracción:** La mayor ventaja reconocida de los SP es la posibilidad de reutilizar el código. Una vez creado, un SP puede ser usado nuevamente por varias aplicaciones cliente que así lo requieran.

- SP – Usos

- Los SP pueden ser particularmente útiles:
- Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la BD.
- Cuando la seguridad es muy importante. Los bancos, por ejemplo, usan SP para todas las operaciones comunes. Esto proporciona un entorno seguro y consistente, y los SP pueden asegurar que cada operación se loguea apropiadamente. En tal entorno, las aplicaciones y los usuarios no obtendrían ningún acceso directo a las tablas de la base de datos, sólo pueden ejecutar algunos procedimientos almacenados.

- Funciones

- Las funciones definidas son rutinas que aceptan parámetros, realizan una acción, p.e. un cálculo complejo, y devuelven el resultado de esa acción como un valor.
- Una función puede llamarse desde dentro de un comando como cualquier otra función (esto es, invocando el nombre de la función), y puede retornar un valor escalar o tablas.
- Al igual que los stored procedures, las funciones definidas por el usuario reducen el costo de compilación del código, permitiendo una ejecución más rápida.
- Pueden reducir el tráfico de red, ya que una operación que filtra datos basándose en restricciones complejas que no se puede expresar en una sola expresión escalar, se puede expresar como una función. La función se puede invocar en la cláusula `WHERE` para reducir el número de filas que se envían al cliente.

- Funciones (Tipos)

- **Función escalar:** Devuelven un único valor de datos del tipo definido en la cláusula RETURNS.

- **Funciones con valores de tabla:** Devuelven un tipo de datos table . Las funciones insertada con valores de tabla no tienen cuerpo; la tabla es el conjunto de resultados de una sola instrucción SELECT.

Funciones del sistema: Los SGBD generalmente proporcionan numerosas funciones del sistema que se pueden usar para realizar diversas operaciones. Las mismas no se pueden modificar. Por ejemplo en SQL-SERVER tenemos la función **GETDATE()**, para retornar la fecha y hora del motor, así como en POSTGRESQL existe la función equivalente **NOW()**

De similar manera que los SP, las funciones se crean con el comando `CREATE FUNCTION`, pueden especificarse cambios con `ALTER FUNCTION` y eliminarse una función con `DROP FUNCTION`.

- Funciones (Ejemplo Scalar)

CREATE FUNCTION dbo.fn_Calcular_Edad(@id_persona int,
@fecha_evaluacion date)

RETURNS int

AS

-- Calcula la edad de una persona utilizando funcion DATEDIFF

BEGIN

DECLARE @edad int;

SELECT @edad =

ABS(DATEDIFF(YEAR, Fecha_Nacimiento, @Fecha_evaluacion))

FROM Personas

WHERE id_persona = @id_persona

IF (@edad IS NULL)

SET @edad = 0;

RETURN @edad;

END;

- Funciones (Ejemplo Tabla)

CREATE FUNCTION dbo.fn_Usuarios_Activos (@id_usuario int, @fecha date)

RETURNS TABLE

AS

RETURN

(
 SELECT DISTINCT usr.Id_Usuario, usr.Descripcion_Usuario, usr.Usuario
 FROM Usuarios_Logins ul **INNER JOIN** Usuarios usr **ON** ul.id_usuario =
 usr.id_usuario
 WHERE @fecha **BETWEEN**
 ul.Fecha_Entrada **AND** ISNULL(ul.Fecha_Salida,'99991231') **AND**
 ul.id_usuario = @id_usuario
);



Función de Sistema

Funciones (Ejemplos Uso)

Scalar

```
SELECT Id_Persona, Nombre, Genero_Sexual,  
       dbo.fn_Calcular_Edad(id_persona, GETDATE()) AS Edad_Actual  
FROM Personas
```

Tabla

```
SELECT p.Id_Proveedor, p.Nombre, p.Direccion, usr.Usuario  
FROM Proveedores p  
     CROSS APPLY dbo.fn_Usuarios_Activos(p.id_usuario, GETDATE()) usr  
WHERE
```

- Triggers

- Un trigger o disparador es un objeto que se asocia con tablas y se almacena en la base de datos.
- Se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado.
- Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

Estructura básica de un "trigger" (disparador):

Llamada de activación: es la sentencia que permite "disparar" el código a ejecutar.

Restricción: es la condición necesaria para realizar el código. Esta restricción puede ser de tipo condicional o de tipo nulidad.

Acción a ejecutar: es la secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iniciales.

- Triggers

- Un trigger o disparador es un objeto que se asocia con tablas y se almacena en la base de datos.
- Se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado.
- Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

Estructura básica de un "trigger" (disparador):

Llamada de activación: es la sentencia que permite "disparar" el código a ejecutar.

Restricción: es la condición necesaria para realizar el código. Esta restricción puede ser de tipo condicional o de tipo nulidad.

Acción a ejecutar: es la secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iniciales.

- Triggers (Tipos)

La acción del trigger, siempre que no se viole la restricción del trigger se ejecuta dependiendo de la combinación de tipos de trigger:

Before statement: Antes de ejecutar la sentencia de disparo.

Before row: Antes de modificar cada fila afectada por la sentencia de disparo, y antes de chequear las restricciones de integridad apropiadas .

After statement: Después de ejecutar la sentencia de disparo, y después de chequear las restricciones de integridad apropiadas.

After row: Después de modificar cada fila afectada por la sentencia de disparo y posiblemente aplicando las restricciones de integridad apropiadas .

- Triggers (Características, y Ejemplo)

- No aceptan parámetros o argumentos (pero podrían almacenar los datos afectados en tablas temporales)
- No pueden ejecutar las operaciones COMMIT o ROLLBACK porque estas son parte de la sentencia SQL del disparador (únicamente a través de transacciones autónomas)
- Pueden causar errores de mutaciones en las tablas, si se han escrito de manera deficiente.

Ejemplo:

```
CREATE TRIGGER TRG_CLIENTE_MOD ON CLIENTES AFTER  
UPDATE AS  
    BEGIN  
        INSERT INTO HISTORICO_CLIENTE  
            (ID_CLIENTE, FECHA_MODIFICACION)  
        SELECT ID_CLIENTE, GETDATE()  
        FROM INSERTED  
    END
```