Repaso: Stored procedure

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Estructura de un stored procedure

1. **DELIMITER:** Se escribe esta cláusula seguida de una combinación de símbolos que no serán utilizados en el interior del SP.
2. **CREATE PROCEDURE:** Se escribe este comando seguido del nombre que identificará al SP.
3. **BEGIN:** Esta cláusula se utiliza para indicar el inicio del código SQL.
4. **Bloque de instrucciones SQL**.
5. **END:** Se escribe esta cláusula seguida de la combinación de símbolos definidos en DELIMITER y se utiliza para indicar el final del código SQL.

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_nombre_procedimiento()
BEGIN
    -- Bloque de instrucciones SQL;
END $$
```

¿Qué es un parámetro?

- Los parámetros son variables por donde se envían y reciben datos de programas clientes.
- Se definen dentro de la cláusula CREATE.
- Los **SP** pueden tener uno, varios o ningún parámetro de entrada y asimismo, pueden tener uno, varios o ningún parámetro de salida.
- Existen 3 tipos de parámetros:

Parámetro	Tipo	Función
IN	Entrada	Recibe datos
OUT	Salida	Devuelve datos
INOUT	Entrada-Salida	Recibe y devuelve datos

- Un SP puede tener **parámetros**. Los parámetros representan la forma en que un SP puede recibir valores, devolver valores, o ambos.
- Existen parámetros de entrada (IN), de salida (OUT) y de entrada/salida (INOUT).

Ejemplo:

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_productos(IN filtro_categoria VARCHAR(15))
BEGIN
    SELECT ProductoNombre, PrecioUnitario FROM productos p
    JOIN categorias C ON p.CategoriaID = c.CategoriaID
    WHERE CategoriaNombre = filtro_categoria;
END $$
```

SQL

```sql
CALL sp_productos(CarnesRojas);
```

- Dentro de un **SP** se permite declarar y asignar valores a una **variable** usando **SET** o dentro de una sentencia SELECT utilizando **INTO**.
- Fuera del SP usamos las variables anteponiendo el símbolo **@**.

Ejemplo:

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_cantidad_productos(IN filtro_categoria VARCHAR(15), OUT cantidad INT)
BEGIN
    SELECT count(*) INTO cantidad FROM productos p
    JOIN categorias C ON p.CategoriaID = c.CategoriaID
    WHERE CategoriaNombre = filtro_categoria;
END $$
```

SQL

```sql
CALL sp_cantidad_productos('Seafood', @cant_seafood);
SELECT @cant_seafood;
```

Declaración del parámetro INOUT

Es un mismo parámetro que utiliza para la entrada y salida de datos. Puede recibir valores y devolverlos los resultados en la misma variable.

Sintaxis:

```sql
CREATE PROCEDURE sp_nombre_procedimiento(INOUT param1 TIPO_DE_DATO, INOUT param2 TIPO_DE_DATO);
```

Ejemplo:

```sql
DELIMITER $$
CREATE PROCEDURE sp_aumentar(INOUT aumento FLOAT) 2.000
BEGIN
     SET aumento = aumento + 26700.50;
END $$
```

Ejecución:

```sql
SET @salario = 2000.00; -- Declaración y asignación de variable (dato)
CALL sp_aumentar(@salario); -- Ejecución y envío de dato (2000.00)
SELECT @salario; -- Muestra el resultado
```

Ventajas del stored procedure

- **Gran velocidad de respuesta**: Todo se procesa dentro del servidor.
- **Mayor seguridad**: Se limita e impide el acceso directo a las tablas donde están almacenados los datos, evitando la manipulación directa por parte de las aplicaciones clientes.
- **Independencia**: Todo el código está dentro de la base de datos y no depende de archivos externos.
- **Reutilización del código**: se elimina la necesidad de escribir nuevamente un conjunto de instrucciones.
- **Mantenimiento más sencillo**: Disminuye el costo de modificación cuando cambian las reglas de negocio.

Desventajas del stored procedure

- **Difícil modificación:** Si se requiere modificarlo, su definición tiene que ser reemplazada totalmente. En bases de datos muy complejas, la modificación puede afectar a las demás piezas de software que directa o indirectamente se refieran a este.
- **Aumentan el uso de la memoria:** Si usamos muchos procedimientos almacenados, el uso de la memoria de cada conexión que utiliza esos procedimientos se incrementará sustancialmente.
- **Restringidos para una lógica de negocios compleja:** En realidad, las construcciones de procedimientos almacenados no están diseñadas para desarrollar una lógica de negocios compleja y flexible.


DigitalHouse>
Coding School