Stored Procedures

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 Concepto, estructura y definición

¿Qué es un stored procedure?

Los **SP** (stored procedure) son un conjunto de instrucciones en formato SQL que se almacenan, compilan y ejecutan dentro del servidor de bases de datos.

Pueden incluir parámetros de entrada y salida, devolver resultados tabulares o escalares, mensajes para el cliente e invocar instrucciones DDL y DML.

Por lo general, se los utiliza para definir la lógica del negocio dentro de la base de datos y reducir la necesidad de codificar dicha lógica en programas clientes.

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

Definición de un stored procedure

- **CREATE PROCEDURE**: Crea un procedimiento almacenado.

```sql
CREATE PROCEDURE sp_nombre_procedimiento()
```

- **DROP PROCEDURE**: Elimina un procedimiento almacenado. Se requiere del privilegio de ALTER ROUTINE.

```sql
DROP PROCEDURE [IF EXISTS] sp_nombre_procedimiento();
```

2 | Variables

Declaración de variables

- Dentro de un **SP** se permite declarar variables que son elementos que almacenan datos que pueden ir cambiando a lo largo de la ejecución.
- La declaración de variables se coloca después de la cláusula BEGIN y antes del bloque de instrucciones SQL.
- Opcionalmente, se puede definir un valor inicial mediante la cláusula DEFAULT.

Sintaxis:

```sql
DECLARE nombre_variable TIPO_DE_DATO [DEFAULT valor];
```

Ejemplo:

```sql
DECLARE salario FLOAT DEFAULT 1000.00;
```

Asignación de valores a variables

- Para asignar un valor a una variable se utiliza la cláusula SET.
- Las variables solo pueden contener valores escalares. Es decir, un solo valor.

Sintaxis:

```sql
SET nombre_variable = expresión;
```

Ejemplo:

```sql
DELIMITER $$
CREATE PROCEDURE sp_nombre_procedimiento()
BEGIN
     DECLARE salario FLOAT DEFAULT 1000.00;
     SET salario = 25700.50;
END $$
```

3 | Parámetros

¿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

Declaración del parámetro IN

Es un parámetro de entrada de datos y se utiliza para recibir valores. Este parámetro viene definido por defecto cuando no se especifica su tipo.

Sintaxis:

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

Ejemplo:

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

Ejecución:

```sql
CALL sp_nombre_procedimiento(11);
```

Declaración del parámetro OUT

Es un parámetro de salida de datos y se utiliza para devolver valores.

Sintaxis:

```SQL
CREATE PROCEDURE sp_nombre_procedimiento(OUT param1 TIPO_DE_DATO, OUT param2 TIPO_DE_DATO);
```

Ejemplo:

```SQL
DELIMITER $$
CREATE PROCEDURE sp_nombre_procedimiento(OUT salario FLOAT)
BEGIN
     SET salario = 25700.50;
END $$
```

Ejecución:

```SQL
CALL sp_nombre_procedimiento(@salario);
SELECT @salario; -- Bloque de instrucciones SQL
```

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 $$
CREATE PROCEDURE sp_nombre_procedimiento(INOUT aumento FLOAT)
BEGIN
     SET aumento = aumento + 25700.50;
END $$
```

Ejecución:

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

4 Ventajas y desventajas

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