Integración de Instrucciones DDL y DML

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 Instrucciones DDL

Instrucción CREATE TABLE

Dentro de un **SP** podemos utilizar diferentes instrucciones DDL. Si queremos crear una tabla para almacenar datos temporales, debemos introducir la instrucción CREATE TABLE dentro para crear dicha tabla.

```SQL
DELIMITER $$
CREATE PROCEDURE sp_crear_tabla()
BEGIN
     CREATE TABLE nombre_tabla (
          id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
          descripcion VARCHAR(200));
END $$
```

```SQL
CALL sp_crear_tabla();
```

Instrucción ALTER TABLE

Si requerimos modificar una tabla porque su estructura cambia de forma frecuente, introducimos la instrucción ALTER TABLE dentro de un **SP**.

```sql
DELIMITER $$
CREATE PROCEDURE sp_modificar_tabla()
BEGIN
     ALTER TABLE nombre_tabla ADD COLUMN campo VARCHAR(50) NOT NULL;
END $$
```

```sql
CALL sp_modificar_tabla();
```

Instrucción DROP TABLE

Ahora, si necesitamos eliminar una tabla temporal, introducimos la instrucción DROP TABLE dentro de un **SP**.

```SQL
DELIMITER $$
CREATE PROCEDURE sp_eliminar_tabla()
BEGIN
     DROP TABLE IF EXISTS nombre_tabla;
END $$
```

```SQL
CALL sp_eliminar_tabla();
```

2 Instrucciones DML con parámetros

Instrucción INSERT

Dentro de un **SP** podemos utilizar diferentes instrucciones DML. Si queremos agregar un nuevo usuario llamado “DIEGO PEREZ”, debemos utilizar parámetros de entrada para que el **SP** reciba dichos datos. Estos datos, serán utilizados como valores en la instrucción INSERT.

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_agregar_usuario(
     IN nombre VARCHAR(30), IN apellido VARCHAR(30))
BEGIN
     INSERT INTO usuario (nombre, apellido) VALUES (nombre, apellido);
END $$
```

SQL

```sql
CALL sp_agregar_usuario('DIEGO', 'PEREZ');
```

Instrucción UPDATE

También, podemos modificar los datos de una tabla. Por ejemplo, se necesita cambiar el nombre de un usuario llamado “DIEGO” por “PABLO”. Para esto, se utilizan parámetros de entrada para que el **SP** reciba dichos datos. Estos datos, serán utilizados como valores en la instrucción UPDATE.

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_modificar_nombre_usuario(
     IN id INT, IN nombre VARCHAR(30))
BEGIN
     UPDATE usuario SET nombre = nombre WHERE id_usuario = id;
END $$
```

SQL

```sql
CALL sp_modificar_nombre_usuario(1,'PABLO');
```

Instrucción DELETE

Asimismo, podemos eliminar los datos de una tabla. Por ejemplo, se requiere borrar los datos de un usuario cuyo ID es 1. Entonces, se utilizan parámetros de entrada para que el **SP** reciba el valor del ID y tal valor será introducido en el WHERE de la instrucción DELETE.

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_eliminar_usuario(IN id INT)
BEGIN
     DELETE FROM usuario WHERE id_usuario = id;
END $$
```

SQL

```sql
CALL sp_eliminar_usuario(1);
```

Instrucción SELECT Con IN - OUT

La instrucción SELECT nos permite listar los datos de una tabla. Por ejemplo, se quiere saber cuál es el nombre del usuario con ID de valor 1. Para esto, se recibe el ID en un parámetro de entrada y el resultado se almacena en un parámetro de salida con la cláusula INTO.

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_dame_nombre_usuario(
     INOUT id INT, OUT nom VARCHAR(30))
BEGIN
     SELECT nombre INTO nom FROM usuario WHERE id_usuario = id;
END $$
```

SQL

```sql
CALL sp_dame_nombre_usuario(1,@nombre); -- Ejecución del SP y envía “1” como dato
SELECT @nombre; -- Muestra el resultado
```

Instrucción SELECT Con INOUT

Una variante del uso de esta instrucción podría ser utilizar un mismo parámetro para la entrada y la devolución del resultado. Por ejemplo, se quiere saber cuántos usuarios tienen en su nombre la letra “a”.

SQL

```sql
DELIMITER $$
CREATE PROCEDURE sp_dame_nombre_usuario(INOUT valor VARCHAR(30))
BEGIN
      SELECT COUNT(*) INTO valor FROM usuario
      WHERE nombre LIKE CONCAT('%', valor, '%');
END $$
```

SQL

```sql
SET @letra = 'a'; -- Declaración y asignación de una variable (dato)
CALL sp_dame_nombre_usuario(@letra); -- Ejecución del SP y envía “a” como dato
SELECT @letra; -- Muestra el resultado
```


DigitalHouse>
Coding School