Integração de instruções DDL e DML

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Temas

1 Instruções DDL

2 Instruções DML com parâmetros

1 | Instruções DDL

Instrução CREATE TABLE

Dentro de uma SP, podemos usar diferentes instruções DDL. Se quisermos criar uma tabela para armazenar dados temporários, devemos introduzir a instrução CREATE TABLE para criar essa tabela.

SQL
```sql
DELIMITER $$
CREATE PROCEDURE sp_criar_tabela()
BEGIN
     CREATE TABLE nome_tabela (
          id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
          descricao VARCHAR(200));
END $$
```

SQL
```sql
CALL sp_criar_tabela();
```

Instrução ALTER TABLE

Se precisarmos modificar uma tabela porque sua estrutura muda com frequência, apresentamos a instrução ALTER TABLE em uma SP.

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

```sql
CALL sp_modificar_tabela();
```

Instrução DROP TABLE

Agora, se precisarmos eliminar uma tabela temporária, colocamos a instrução DROP TABLE dentro de uma SP.

```sql
DELIMITER $$
CREATE PROCEDURE sp_eliminar_tabela()
BEGIN
     DROP TABLE IF EXISTS nome_tabela;
END $$
```

```sql
CALL sp_eliminar_tabela();
```

2 | Instruções DML com parâmetros

Instrução INSERT

Dentro de uma SP, podemos usar diferentes instruções DML. Se quisermos adicionar um novo usuário denominado "DIEGO ROCHA", devemos utilizar parâmetros de entrada para que a SP receba esses dados. Esses dados serão usados como valores na instrução INSERT.

SQL
```sql
DELIMITER $$
CREATE PROCEDURE sp_adicionar_usuario(
     IN nome VARCHAR(30), IN sobrenome VARCHAR(30))
BEGIN
     INSERT INTO usuario (nome, sobrenome) VALUES (nome, sobrenome);
END $$
```

SQL
```sql
CALL sp_adicionar_usuario('DIEGO', 'ROCHA');
```

Instrução UPDATE

Além disso, podemos modificar os dados de uma tabela. Por exemplo, você precisa alterar o nome de um usuário chamado “DIEGO” para “DOUGLAS”. Para isso, são utilizados parâmetros de entrada para que a SP receba esses dados. Esses dados serão usados como valores na instrução UPDATE.

SQL

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

SQL

```sql
CALL sp_modificar_nome_usuario(1,'DOUGLAS');
```

Instrução DELETE

Da mesma forma, podemos remover os dados de uma tabela. Por exemplo, é necessário deletar os dados de um usuário cujo ID é 1. Em seguida, os parâmetros de entrada são usados para que a SP receba o valor do ID e tal valor seja inserido no WHERE da instrução DELETE.

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

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

Instrução SELECT Com IN - OUT

A instrução **SELECT** nos permite listar os dados de uma tabela. Por exemplo, você deseja saber o nome do usuário com ID de valor 1. Para isso, o ID é recebido em um parâmetro de entrada e o resultado é armazenado em um parâmetro de saída com a cláusula INTO.

SQL

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

SQL

```sql
CALL sp_dame_nome_usuario(1,@nome); -- Executa a SP e envia o  “1” como dado
SELECT @nome; -- Exibe o resultado
```

Instrução SELECT Com INOUT

Uma variante do uso desta instrução poderia ser usar o mesmo parâmetro para a entrada e o retorno do resultado. Por exemplo, você deseja saber quantos usuários possuem a letra "a" em seus nomes.

SQL

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

SQL

```sql
SET @letra = 'a'; -- Declaração e atribuição de uma variável (dado)
CALL sp_exibe_nome_usuario(@letra); --Executa a SP e envia “a” como dado
SELECT @letra; -- Exibe o resultado
```


DigitalHouse>
Coding School