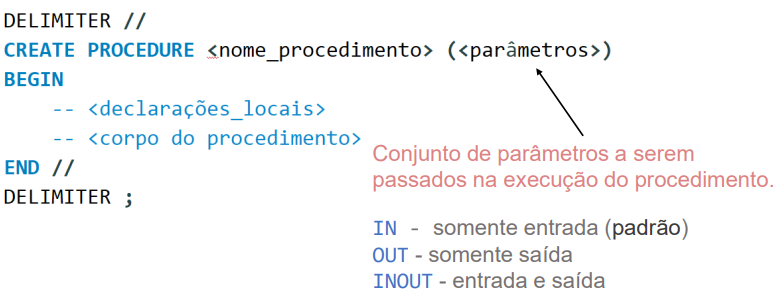
**BD2 – Prova - Resumo**

**Stored procedures / Função**

* Stored Procedure (ou Procedimento Armazenado) é:
* Um conjunto de comandos SQL (tipo SELECT, INSERT, UPDATE...) que você grava no Banco de Dados.
* Depois você executa a procedure quantas vezes quiser, sem precisar reescrever o código.

Vantagens:

* Evita repetição de código.
* Torna as operações mais rápidas e organizadas.
* Útil para tarefas complexas ou repetitivas.



Existem 3 tipos principais de parâmetros

| Tipo de Parâmetro | Para que serve | Exemplo Rápido |
| --- | --- | --- |
| IN | Enviar um valor para a procedure usar | IN nome VARCHAR(50) |
| OUT | Receber um valor de dentro da procedure | OUT resultado INT |
| INOUT | Enviar e receber valor (entrada + saída) | INOUT contador INT |

Parâmetro IN

Só envia valor para a procedure.

A procedure não altera o valor que foi passado.

CREATE PROCEDURE buscar\_pessoa(IN nome\_busca VARCHAR(100))

BEGIN

SELECT \* FROM pessoas

WHERE nome = nome\_busca;

END;

Chamada:

CALL buscar\_pessoa('Ana');

(Manda "Ana" para dentro da procedure.)

Parâmetro OUT

A procedure cria/modifica o valor, e você pega o valor de volta.

CREATE PROCEDURE contar\_pessoas(OUT total INT)

BEGIN

SELECT COUNT(\*) INTO total FROM pessoas;

END;

Chamada:

-- Criar uma variável para receber o resultado

SET @resultado = 0;

CALL contar\_pessoas(@resultado);

SELECT @resultado;

(Vai contar quantas pessoas existem e guardar o valor na variável @resultado.)

Parâmetro INOUT

Envia um valor para a procedure e depois recebe o valor alterado.

CREATE PROCEDURE aumentar\_valor(INOUT numero INT)

BEGIN

SET numero = numero + 10;

END;

Chamada:

SET @meu\_numero = 5;

CALL aumentar\_valor(@meu\_numero);

SELECT @meu\_numero;

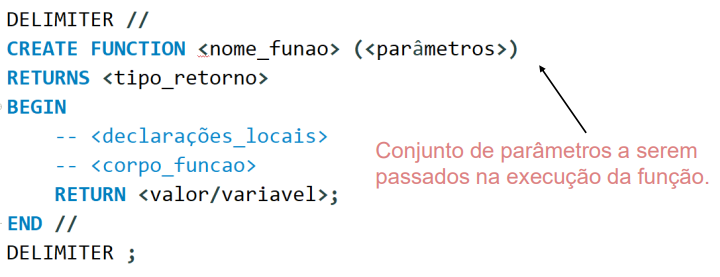
(Manda o número 5, a procedure soma 10, e você recebe 15 de volta.)

**Function**

* Função sempre devolve um único valor (como um número, um texto, etc).
* Você pode usar funções dentro de SELECTs normalmente.

Vantagens

* Fazer pequenos cálculos.
* Transformar ou formatar dados.
* Facilitar consultas que sempre usam a mesma lógica.



Criar uma função que soma dois números:

CREATE FUNCTION soma(a INT, b INT)

RETURNS INT

DETERMINISTIC

BEGIN

RETURN a + b;

END;

Depois para usar:

SELECT soma(10, 5);

(Vai retornar 15.)

**OBS: VER?? Permite lógica condicional e loops: IF-THEN, CASE, WHILE, LOOP.**

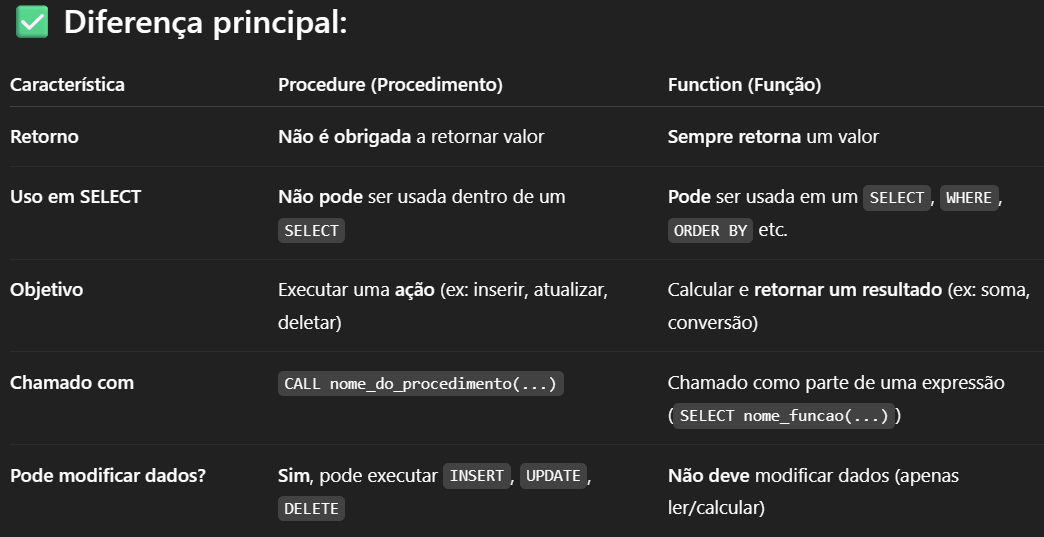
**Diferença Rápida: Stored Procedure x Função**

| Stored Procedure | Função |
| --- | --- |
| Pode fazer várias operações (INSERT, UPDATE, DELETE, SELECT) | Só devolve 1 valor |
| É chamada com CALL | É usada dentro de SELECT |
| Pode ou não devolver um resultado | Sempre devolve um único resultado |
| Usada para operações mais complexas | Usada para pequenos cálculos |

Stored Procedure: script gravado no banco que pode fazer várias operações de uma vez

Função: script gravado que devolve um único valor e é usada dentro de SELECTs

* Use PROCEDURE → quando quiser executar ações ou múltiplas etapas (ex: atualizar salário, inserir dados).
* Use FUNCTION → quando quiser calcular e retornar um valor simples que pode ser usado em uma consulta.



**Principais usos do SET:**

1. Definir valor de variáveis (em procedimentos, funções ou scripts):

DECLARE salario\_atual INT;

SET salario\_atual = 50000;

2. Atualizar colunas de uma tabela:

UPDATE employees

SET last\_name = 'Silva'

WHERE emp\_no = 10001;

3. Definir mensagens em erros personalizados (SIGNAL):

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Salário inválido';

4. Alterar variáveis de sistema ou sessão (mais avançado):

SET @meu\_nome = 'Maria'; -- variável do usuário

SET sql\_mode = 'STRICT\_ALL\_TABLES'; -- variável do sistema

Importante:

* O SET não executa nenhuma lógica sozinho, ele só define valores.
* É um comando fundamental em scripts SQL, especialmente dentro de procedimentos e funções.