



CEUB

EDUCAÇÃO SUPERIOR

ceub.br



BANCO DE DADOS II

AULA 12

FUNÇÕES

Prof. Leonardo R. de Deus

1. INTRODUÇÃO

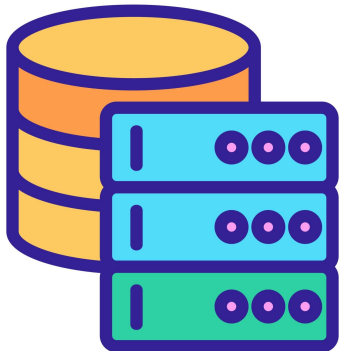
Vimo que:

O SQL nos permite interagir com o banco de dados, para obter respostas com base nos dados que temos armazenados.

1. INTRODUÇÃO

Vimo que:

O SQL nos permite interagir com o banco de dados, para obter respostas com base nos dados que temos armazenados.



- 📦 Armazena dados de forma estruturada: Tabelas, relacionamentos, índices, integridade.
- 🔍 Permite consultas inteligentes: Filtrar, agrupar e cruzar informações para gerar respostas.
- 🧩 Garante integridade e consistência: Chaves primárias, estrangeiras, restrições e validações.
- 🔒 Oferece segurança e controle de acesso: Usuários, permissões, criptografia e logs de auditoria.
- ⚙️ **Automatiza regras de negócio: Triggers, functions e views que aplicam lógicas internas.**
- 🚀 Suporta processamento e lógica interna: O banco pode “pensar” e tomar decisões com **SQL procedural**.

1. INTRODUÇÃO

Vimo que:

O SQL nos permite agir com o banco de dados para obter respostas com base nos dados que temos.

O banco de dados não é só um repositório.



É o motor lógico que sustenta o sistema.

...s, índices, integridade.

... para gerar respostas.

...as, restrições e validações.

...s, permissões, criptografia e logs de auditoria.

Triggers, funções e views que aplicam lógicas internas.

...porta procedimentos e lógica interna: O banco pode “pensar” e tomar decisões com **SQL** procedural.

1. INTRODUÇÃO

**Mas quando o banco
começa a pensar?**



1. INTRODUÇÃO

**Mas quando o banco
começa a pensar?**



**Quando existem regras de negócio
implementadas no banco de dados!**



Sem lógica (SQL declarativo)

O banco apenas responde consultas.

- Quais são os clientes do estado de São Paulo.”



Com lógica (SQL procedural)

O banco executa ações, faz verificações e automatiza decisões.

- Verifique se o cliente existe, calcule seu total de compras e atualize o estoque.”

1. INTRODUÇÃO

Mas quando o banco começa a pensar?



Quando existem regras de negócio implementadas no banco de dados!

Sem lógica (SQL declarativo)

O banco apenas responde consultas.

- Quais são os clientes do estado de

- **variáveis**
- **estruturas**
- **condicionais**
- **laços de repetição**
- **tratamento de erros**

Com lógica (SQL procedural)

O banco executa ações, faz verificações e automatiza decisões.

- Verifique se o cliente existe, calcule seu total de compras e atualize o estoque.”

1. INTRODUÇÃO

FUNÇÕES
PROCEDIMENTOS ARMAZENADOS
TRIGGERS



Quando existem regras de negócio implementadas no banco de dados!

Com lógica (SQL procedural)

Exemplo: verificar se o cliente é novo.

São os clientes do estado...

Com lógica (SQL procedural)

O banco executa ações, faz verificações e automatiza decisões.

- "Verifique se o cliente existe, calcule seu total de compras e atualize o estoque."

O QUE É UM FUNÇÃO (FUNCTION)

Uma função é um bloco de código reutilizável que executa uma tarefa específica e retorna um valor.

São usadas para:

- automatizar cálculos;
- formatar dados;
- centralizar regras de negócio;
- evitar repetição de código SQL;
- melhorar desempenho da aplicação

O QUE É UM FUNÇÃO (FUNCTION)

Uma função é um bloco de código reutilizável que executa uma tarefa específica e retorna um valor.

São usadas para:

- automatizar cálculos;
- formatar dados;
- centralizar regras de negócio;
- evitar repetição de código SQL;
- melhorar desempenho da aplicação

Importante:

- ★ função sempre retorna um valor (escalar ou tabular);
- ★ é chamada como uma consulta (select) ou dentro de outras consultas (select ou where);
- ★ como boa prática, não deve modificar dados do banco

3. ESTRUTURA DE UMA FUNÇÃO

CREATE FUNCTION nome_da_funcao(parâmetros tipo_parametro)

RETURNS tipo_retorno AS \$\$

BEGIN

-- bloco de código

END;

\$\$ **LANGUAGE** plpgsql;

3. ESTRUTURA DE UMA FUNÇÃO

CREATE FUNCTION nome_da_funcao(parâmetros tipo_parametro)

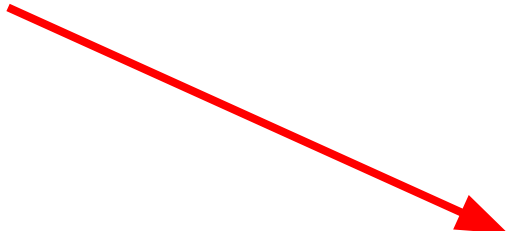
RETURNS tipo_retorno AS \$\$

BEGIN

-- bloco de código

END;

\$\$ **LANGUAGE** plpgsql;

- 
- int
 - double
 - varchar
 - text
 - table
 - void
 -
 -
 -

3. ESTRUTURA DE UMA FUNÇÃO

Nossa primeira função

```
CREATE FUNCTION loja.fn_soma(num1 int, num2 int)
```

```
RETURNS INT AS $$
```

```
BEGIN
```

```
    RETURN num1 + num2;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

3. ESTRUTURA DE UMA FUNÇÃO

Alterando a função

CREATE OR REPLACE FUNCTION loja.fn_soma(num1 int, num2 int, num3 int)

RETURNS TEXT AS \$\$

BEGIN

RETURN num1 + num2 + num3;

END;

\$\$ LANGUAGE plpgsql;

3. ESTRUTURA DE UMA FUNÇÃO

Alterando a função, mudando o tipo de retorno

CREATE OR REPLACE FUNCTION loja.fn_soma(num1 int, num2 int)

RETURNS TEXT AS \$\$

BEGIN

RETURN 'A somas dos números fornecidos é: ' || num1 + num2;

END;

\$\$ LANGUAGE plpgsql;

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 1:

Em vários relatórios, precisamos calcular o valor total de um item (quantidade * preço).

```
SELECT
    id_item_pedido,
    id_pedido,
    quantidade,
    valor_unitario,
    (quantidade * valor_unitario) AS valor_total
FROM
    loja.tb05_item_pedido as ip
WHERE id_pedido = 31;
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 1:

Em vários relatórios, precisamos calcular o valor total de um item (quantidade * preço).

```
CREATE OR REPLACE FUNCTION loja.fn_calcular_valor_total_item(  
    p_quantidade INT,  
    p_preco_unitario DECIMAL  
)  
RETURNS DECIMAL AS $$  
BEGIN  
  
    RETURN p_quantidade * p_preco_unitario;  
  
END;  
$$ LANGUAGE plpgsql;
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 1:

Usando a função

```
select loja.fn_calcular_valor_total_item(quantidade, valor_unitario)  
from loja.tb05_item_pedido
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 1:

Usando a função

```
SELECT
    id_item_pedido,
    id_pedido,
    quantidade,
    valor_unitario,
    loja.fn_calcular_valor_total_item(quantidade, valor_unitario) AS valor_total
FROM loja.tb05_item_pedido
WHERE id_pedido = 31;
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 2:

A equipe de logística precisa do endereço completo dos clientes para gerar etiquetas de envio

```
SELECT
    nome_cliente,
    endereco || ', ' || cidade || ' - ' || estado || ', CEP: ' || cep as endereço
FROM
    loja.tb01_cliente
WHERE
    id_cliente = 1;
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 2:

A equipe de logística precisa do endereço completo dos clientes para gerar etiquetas de envio

CREATE OR REPLACE FUNCTION

```
loja.fn_obter_endereco_completo(p_id_cliente INT)
RETURNS TEXT AS $$
DECLARE
    v_endereco_formatado TEXT;
BEGIN
    SELECT
        endereco || ', ' || cidade || ' - ' || estado || ', CEP: ' || cep
    INTO
        v_endereco_formatado
    FROM
        loja.tb01_cliente
    WHERE
        id_cliente = p_id_cliente;

    RETURN v_endereco_formatado;
END;
$$ LANGUAGE plpgsql;
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 2:

Usando a função

```
select loja.fn_obter_endereco_completo(15)
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 2:

Usando a função

```
SELECT
    nome_cliente,
    loja.fn_obter_endereco_completo(id_cliente) AS endereço
FROM
    loja.tb01_cliente
WHERE
    id_cliente = 15;
```


4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 3:

Precisamos de uma forma rápida de classificar o status do estoque de um produto em relatórios, em vez de apenas ver o número.

CREATE OR REPLACE FUNCTION

```
loja.fn_verificar_status_estoque(p_id_produto INT)
RETURNS TEXT AS $$
DECLARE
    v_estoque_atual INT;
BEGIN
    SELECT estoque INTO v_estoque_atual
    FROM loja.tb02_produto
    WHERE id_produto = p_id_produto;

    IF v_estoque_atual = 0 THEN
        RETURN 'Indisponível';
    ELSIF v_estoque_atual <= 5 THEN
        RETURN 'Estoque Baixo';
    ELSIF v_estoque_atual <= 20 THEN
        RETURN 'Nível Atenção';
    ELSE
        RETURN 'Disponível';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 3:

Usando a função

```
select loja.fn_verificar_status_estoque(1)
```

```
update loja.tb02_produto  
set estoque = 2  
where id_produto = 1
```

4. CRIANDO UMA FUNÇÃO NO BANCO ECOMMERCE

Problema 3:

Usando a função

```
SELECT
    nome_produto,
    categoria,
    estoque,
    loja.fn_verificar_status_estoque(id_produto) AS status
FROM
    loja.tb02_produto
ORDER BY
    estoque ASC;
```

5. ATIVIDADE

O que fazer:

Crie uma função no banco Ecommerce, que execute alguma ação com base nas tabelas e dados que temos no nosso banco.

O que entregar:

O script da função, funcionando, em formato pdf, na tarefa da sala online.

**OBRIGADO
A TODOS!**

