CEUB

EDUCAÇÃO SUPERIOR

ceub.br

BANCO DE DADOS II

AULA 13

PROCEDURES

Prof. Leonardo R. de Deus

1. INTRODUÇÃO

Vimos 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 [illegible]gir com o[illegible]ra obt[illegible]espostas com base nos dados que temos [illegible]

O banco de dados não é só um repositório.

É o motor lógico que sustenta o sistema.

[illegible]s, índices, integridade.

[illegible]s para gerar respostas.

[illegible]iras, restrições e validações.

[illegible]s, permissões, criptografia e logs de auditoria.

[illegible]Triggers, funct[illegible]ns e views que aplicam lógicas internas.

[illegible]porta pro[illegible]amento e lógica interna: O banco pode “pensar” e tomar decisões com **SQL [illegible]cedural.**

2. SQL PROCEDURAL

🧠 **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.”

Quando existem regras de negócio implementadas no banco de dados!

PROCEDIMENTOS ARMAZENADOS

- consistem em um conjunto de instruções SQL que são armazenadas no banco de dados e podem ser executadas repetidamente;

- são como programas que residem no banco de dados e podem ser chamados por nome para realizar uma tarefa específica.

2. SQL PROCEDURAL

PROCEDIMENTOS ARMAZENADOS

- consistem em um conjunto de instruções SQL que são armazenadas no banco de dados e podem ser executadas repetidamente;

- são como programas que residem no banco de dados e podem ser chamados por nome para realizar uma tarefa específica.

- **FUNCTIONS**
- **PROCEDURES**
- **TRIGGERS**

FUNCTIONS

São blocos de código SQL que retornam um valor ou um conjunto de valores.

São utilizadas para realizar cálculos ou operações específicas e **podem ser utilizadas em expressões SQL**.

Podem ter parâmetros de entrada, mas não podem ter parâmetros de saída.

Regra de Ouro: Functions são para realizar cálculos e retornar valores.

2. SQL PROCEDURAL

FUNCTIONS

São blocos de código SQL que retornam um valor ou um conjunto de valores.

São utilizadas para realizar cálculos ou operações específicas e **podem ser utilizadas em expressões SQL**.

Podem ter parâmetros de entrada, mas não podem ter parâmetros de saída.

Regra de Ouro: Functions são para realizar cálculos e retornar valores.

VS

PROCEDURES

São conjuntos de instruções SQL que são armazenadas no banco e podem ser executadas repetidamente.

Podem realizar operações de CRUD em tabelas, realizar cálculos, enviar mensagem, etc.

Podem ter parâmetros de entrada e saída, **mas não podem ser utilizados em expressões SQL**.

Regra de Ouro: Procedures são para executar ações e modificar dados.

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação		
Retorno		
Modifica dados?		
Como é chamada?		
Forma de usar		
Analogia		

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação	cuidam de consultas e cálculos (resultados e relatórios)	cuidam de rotinas de processo (atualizações e automatizações)
Retorno		
Modifica dados?		
Como é chamada?		
Forma de usar		
Analogia		

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação	cuidam de consultas e cálculos (resultados e relatórios)	cuidam de rotinas de processo (atualizações e automatizações)
Retorno	obrigatório (sempre retorna um valor)	opcional
Modifica dados?		
Como é chamada?		
Forma de usar		
Analogia		

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação	cuidam de consultas e cálculos (resultados e relatórios)	cuidam de rotinas de processo (atualizações e automatizações)
Retorno	obrigatório (sempre retorna um valor)	opcional
Modifica dados?	não (por boa prática)	sim (sendo o objetivo principal)
Como é chamada?		
Forma de usar		
Analogia		

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação	cuidam de consultas e cálculos (resultados e relatórios)	cuidam de rotinas de processo (atualizações e automatizações)
Retorno	obrigatório (sempre retorna um valor)	opcional
Modifica dados?	não (por boa prática)	sim (sendo o objetivo principal)
Como é chamada?	dentro de um Select, From, ou Where	utiliza o comando CALL
Forma de usar		
Analogia		

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação	cuidam de consultas e cálculos (resultados e relatórios)	cuidam de rotinas de processo (atualizações e automatizações)
Retorno	obrigatório (sempre retorna um valor)	opcional
Modifica dados?	não (por boa prática)	sim (sendo o objetivo principal)
Como é chamada?	dentro de um Select, From, ou Where	utiliza o comando CALL
Forma de usar	Select fn_calculavalor(...) from …;	Call sp_uptable(...);
Analogia		

2. SQL PROCEDURAL

CARACTERÍSTICA	FUNCTION	PROCEDURE
Propósito Principal	calcular e retornar um valor	executar uma sequência de ações
Aplicação	cuidam de consultas e cálculos (resultados e relatórios)	cuidam de rotinas de processo (atualizações e automatizações)
Retorno	obrigatório (sempre retorna um valor)	opcional
Modifica dados?	não (por boa prática)	sim (sendo o objetivo principal)
Como é chamada?	dentro de um Select, From, ou Where	utiliza o comando CALL
Forma de usar	Select fn_calculavalor(...) from …;	Call sp_uptable(...);
Analogia	uma calculadora (funções respondem perguntas)	uma receita/checklist (procedures executam tarefas)

3. PROCEDURE (RESUMO)

Uma **stored procedure** (ou simplesmente *procedure*) é um bloco de código armazenado no banco de dados que:

- executa instruções SQL sequenciais;
- não retorna valor diretamente, mas pode alterar dados, chamar outras funções, controlar fluxo de execução e transações e retornar parâmetros de saída;
- diferem das funções porque não retornam valor direto

4. CRIANDO NOSSA PRIMEIRA PROCEDURE

Para exemplificar, inicialmente, o funcionamento de uma procedure, vamos criar um exemplo simples de transferência bancária.

4. CRIANDO NOSSA PRIMEIRA PROCEDURE

Para exemplificar, inicialmente, o funcionamento de uma procedure, vamos criar um exemplo simples de transferência bancária.

1. Criar uma tabela de contas

```
CREATE TABLE loja.tb_conta (
    id_conta SERIAL PRIMARY KEY,
    nome_titular VARCHAR(100),
    conta VARCHAR(10),
    saldo DECIMAL NOT NULL DEFAULT 0
);
```

4. CRIANDO NOSSA PRIMEIRA PROCEDURE

Para exemplificar, inicialmente, o funcionamento de uma procedure, vamos criar um exemplo simples de transferência bancária.

2. Inserindo dados da tabela

```
INSERT INTO loja.tb_conta (nome_titular, conta, saldo)
VALUES
('João Gomes', '136-5241/1', 1000.00),
('Bruna Santos', '148-3417/2', 500.00);
```

4. CRIANDO NOSSA PRIMEIRA PROCEDURE

Para exemplificar, inicialmente, o funcionamento de uma procedure, vamos criar um exemplo simples de transferência bancária.

3. Criando a Procedure para efetivar a transferência entre as contas

```
CREATE PROCEDURE loja.sp_tranfere_valor(
        p_conta_origem VARCHAR(10),
        p_conta_destino VARCHAR(10),
        p_valor DECIMAL
)
LANGUAGE SQL
AS $$

        --debitando o valor desejado da conta de origem
        UPDATE loja.tb_conta
        SET saldo = saldo - p_valor
        WHERE conta = p_conta_origem;

        -- creditando o valor desejado na conta de destino
        UPDATE loja.tb_conta
        SET saldo = saldo + p_valor
        WHERE conta = p_conta_destino;

$$;
```

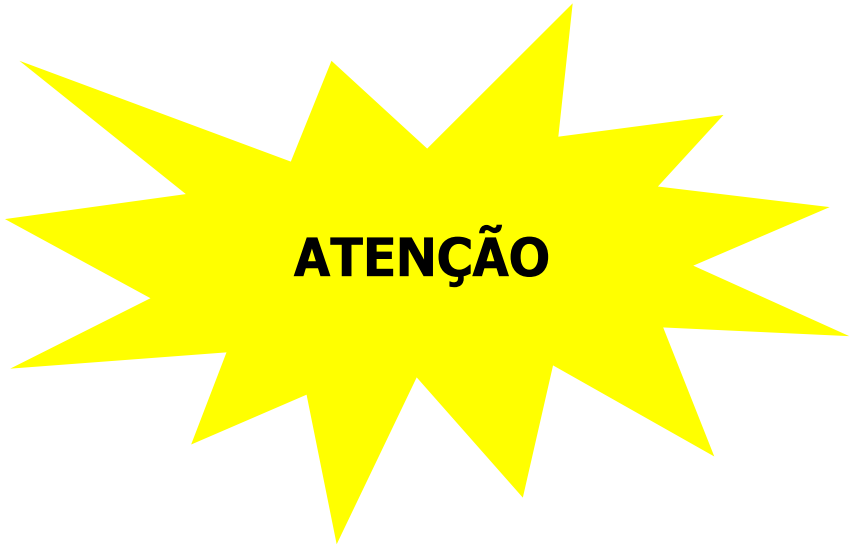
4. Executando a procedure

```
SELECT * FROM loja.tb_conta
```

```
CALL loja.sp_tranfere_valor('136-5241/1','148-3417/2', 200);
```

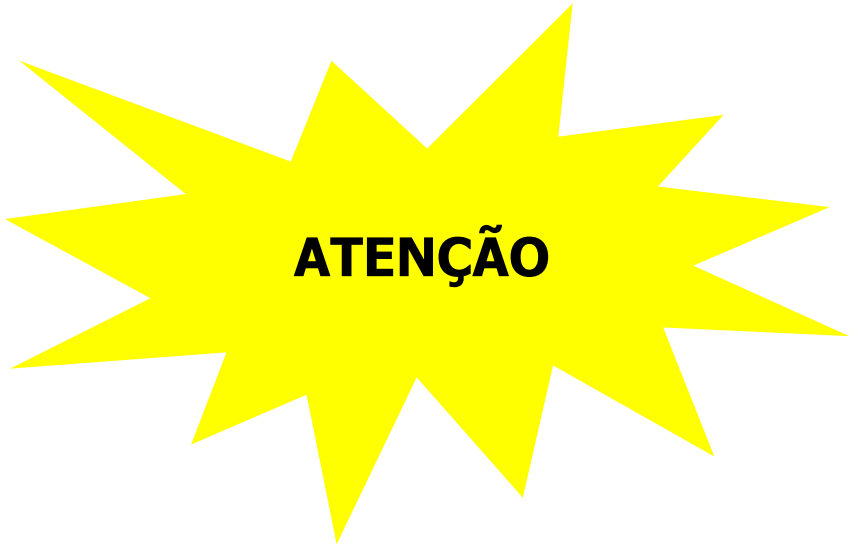
4. Executando a procedure

E se a conta de origem não tiver saldo suficiente para fazer a operação?



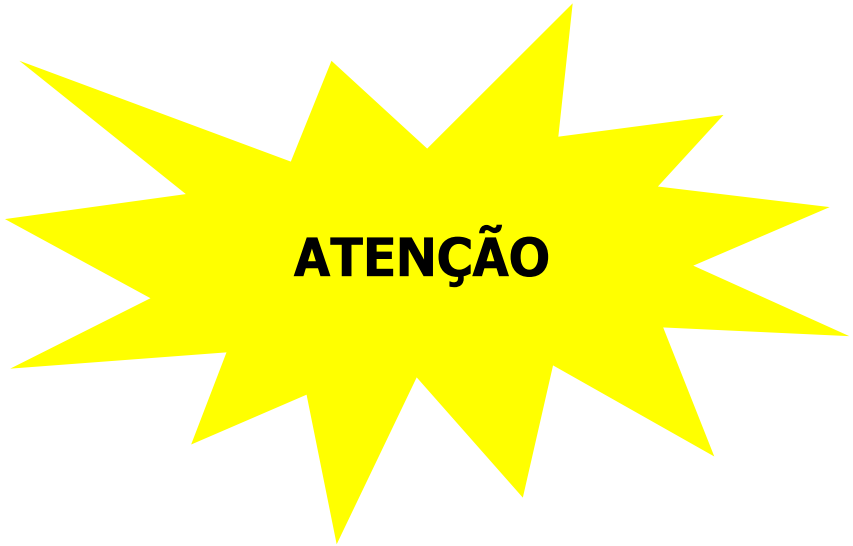
4. Executando a procedure

E se a conta de origem não existir?



4. Executando a procedure

E se a conta de destino não existir?



O que é uma TRANSAÇÃO?

Conjunto de operação que o banco de dados trata como uma única unidade de trabalho.

Ou acontece tudo, ou não acontece nada.

O que é uma TRANSAÇÃO?

Conjunto de operação que o banco de dados trata como uma única unidade de trabalho.

Ou acontece tudo, ou não acontece nada.

Considerando nosso exemplo:

Em **transferência bancária** — se debitar de uma conta, precisa **obrigatoriamente** creditar na outra.

Se der erro no meio do caminho, **nenhuma das duas operações deve ser confirmada.**

TRANSAÇÃO IMPLÍCITA

```
UPDATE loja.tb_conta
      SET saldo = saldo - p_valor
      WHERE conta = p_conta_origem;
```

5. INTRODUÇÃO A TRANSAÇÃO

TRANSAÇÃO IMPLÍCITA

```
UPDATE loja.tb_conta
      SET saldo = 1000
      WHERE conta = '136-5241/1';
```

TRANSAÇÃO EXPLÍCITA

```
BEGIN;

UPDATE loja.tb_conta
      SET saldo = 1000
      WHERE conta = '136-5241/1';

COMMIT;
```

5. Alterando a procedure para usar o poder do SQL Procedural e verificar dados antes de executar a transação

```
CREATE OR REPLACE PROCEDURE
loja.sp_tranfere_valor(
        p_conta_origem VARCHAR(10),
        p_conta_destino VARCHAR(10),
        p_valor DECIMAL
)
LANGUAGE plpgsql
AS $$
DECLARE
        v_saldo_origem DECIMAL;
        v_conta INT;
```

```
BEGIN

        --verificando se a conta de origem existe
        SELECT COUNT(*) INTO v_conta
        FROM loja.tb_conta
        WHERE conta = p_conta_origem;

        IF v_conta = 0 THEN
                RAISE EXCEPTION 'Conta de origem não
encontrada.';
        ELSE
                RAISE NOTICE 'Conta de origem identificada:
%', p_conta_origem;
        END IF;

        --obtendo saldo da conta de origem
        SELECT saldo INTO v_saldo_origem
        FROM loja.tb_conta
        WHERE conta = p_conta_origem;

        --verificando se a conta possui saldo para realizar a
operação
        IF v_saldo_origem < p_valor THEN
                RAISE EXCEPTION 'Saldo insuficiente para
realizar a transferência.';
        END IF;

        --debitando o valor desejado da conta de origem
        UPDATE loja.tb_conta
        SET saldo = saldo - p_valor
        WHERE conta = p_conta_origem;

        --verificando se a conta de destino existe
        SELECT COUNT(*) INTO v_conta
        FROM loja.tb_conta
        WHERE conta = p_conta_destino;

        IF v_conta = 0 THEN
                RAISE EXCEPTION 'Conta de destino não
encontrada.';
        ELSE
                RAISE NOTICE 'Conta de destino identificada: %',
p_conta_destino;
        END IF;

        -- creditando o valor desejado na conta de destino
        UPDATE loja.tb_conta
        SET saldo = saldo + p_valor
        WHERE conta = p_conta_destino;

        RAISE NOTICE 'Transferência concluída com sucesso.';

END;
$$;
```

7. ATIVIDADE

O que fazer:

Crie um exemplo de procedure no banco Ecommerce, que execute alguma ação com base nas tabelas e dados que temos no nosso banco.

O que entregar:

Descrição do que a procedure faz, script da procedure, funcionando.

Entregar em formato pdf, na tarefa da sala online.

OBRIGADO
A TODOS!

CEUB

EDUCAÇÃO SUPERIOR