# STORED PROCEDURE

## 262MI – BANCO DE DADOS II

# O que é uma Stored Procedure?

- Conceito: Uma Stored Procedure (ou Procedimento Armazenado) é um conjunto de instruções SQL preparadas e salvas diretamente no banco de dados. Em vez de escrever a mesma consulta complexa várias vezes na sua aplicação, você a salva no banco e apenas a "chama" quando precisar.
- Analogia: Pense nisso como a receita de um bolo. Você não precisa inventar a receita toda vez que for assar; você apenas pega a receita (a Procedure), fornece os ingredientes como farinha e ovos (os Parâmetros), e o forno faz o trabalho de entregar o bolo pronto (o Resultado).

# Por que usar Procedures? (Vantagens)

- Performance: Como o código fica armazenado no servidor de banco de dados, ele reduz o tráfego de rede entre sua aplicação (ex: um site em PHP/Node.js) e o banco.
- Reuso de Código: A mesma lógica de negócio pode ser usada por múltiplas aplicações diferentes sem reescrever o SQL.
- Segurança: Você pode dar permissão para um usuário executar a Procedure, sem dar a ele permissão para ver ou alterar as tabelas diretamente.
- Manutenção Centralizada: Se a regra de negócio mudar, você altera apenas a Procedure no banco, e todas as aplicações conectadas já passam a usar a regra nova.

# Anatomia de uma Procedure

- Para criar uma procedure no MySQL, mudamos temporariamente o DELIMITER (delimitador) para que o banco não ache que o ponto e vírgula (;) no meio do código seja o fim do comando.
- Parâmetros Principais:
  - **IN**: Dados que entram na procedure (ex: ID do cliente).
  - **OUT**: Dados que a procedure devolve (ex: Total de vendas).
  - **INOUT**: Entra com um valor e sai com ele modificado.

Vamos criar uma procedure para buscar todos os filmes de um determinado ano de lançamento (release_year).

```
DELIMITER $$

CREATE PROCEDURE sp_filmes_por_ano(
    IN p_ano YEAR
)
BEGIN
    SELECT film_id, title, description, release_year
    FROM film
    WHERE release_year = p_ano;
END $$

DELIMITER ;

-- Como usar: CALL sp_filmes_por_ano(2006);
```

Vamos criar uma procedure para cadastrar um novo ator. Isso evita que a aplicação precise fazer o INSERT direto.

```
DELIMITER $$

CREATE PROCEDURE sp_cadastrar_ator(
    IN p_nome VARCHAR(45),
    IN p_sobrenome VARCHAR(45)
)
BEGIN
    INSERT INTO actor (first_name, last_name)
    VALUES (UPPER(p_nome), UPPER(p_sobrenome));

    SELECT LAST_INSERT_ID() AS novo_actor_id;
END $$

DELIMITER ;

-- Como usar: CALL sp_cadastrar_ator('Wagner', 'Moura');
```

Imagine que a diretoria da locadora decidiu aumentar o preço do aluguel (rental_rate) de todos os filmes de uma determinada classificação indicativa (rating).

```
DELIMITER $$

CREATE PROCEDURE sp_aumentar_preco_por_classificacao(
    IN p_rating ENUM('G','PG','PG-13','R','NC-17'),
    IN p_valor_aumento DECIMAL(4,2)
)
BEGIN
    UPDATE film
    SET rental_rate = rental_rate + p_valor_aumento
    WHERE rating = p_rating;
END $$

DELIMITER ;

-- Como usar: CALL sp_aumentar_preco_por_classificacao('PG-13', 0.50);
```

Uma procedure que conta quantos clientes ativos existem em uma loja específica, devolvendo o valor para uma variável.

```
DELIMITER $$

CREATE PROCEDURE sp_contar_clientes_ativos(
    IN p_store_id TINYINT,
    OUT p_total INT
)
BEGIN
    SELECT COUNT(*) INTO p_total
    FROM customer
    WHERE store_id = p_store_id AND active = TRUE;
END $$

DELIMITER ;

-- Como usar:
-- CALL sp_contar_clientes_ativos(1, @total_clientes);
-- SELECT @total_clientes;
```

Vamos registrar um novo aluguel de filme (rental), garantindo que a data do aluguel seja o momento exato da transação.

```
DELIMITER $$

CREATE PROCEDURE sp_registrar_aluguel(
    IN p_inventory_id MEDIUMINT,
    IN p_customer_id SMALLINT,
    IN p_staff_id TINYINT
)
BEGIN
    -- Insere o registro usando a data atual (NOW())
    INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id)
    VALUES (NOW(), p_inventory_id, p_customer_id, p_staff_id);

    -- Retorna sucesso e o ID gerado
    SELECT 'Aluguel registrado com sucesso!' AS status, LAST_INSERT_ID() AS rental_id;
END $$

DELIMITER ;
-- Como usar: CALL sp_registrar_aluguel(15, 300, 1);
```

# EXERCÍCIOS

1) Busca de Cliente: Crie uma procedure chamada sp_buscar_cliente_email que receba um endereço de e-mail como parâmetro IN e retorne o primeiro nome, sobrenome e ID da loja do cliente (customer).

2) Desativar Cliente: Crie uma procedure sp_desativar_cliente que receba um customer_id e atualize o campo active desse cliente para FALSE (0).

3) Filmes por Idioma: Crie uma procedure que receba o nome de um idioma (ex: 'English') na tabela language e faça um SELECT trazendo todos os filmes (film) daquele idioma. (Dica: precisará de um JOIN).

# EXERCÍCIOS

4. Atualizar Custo de Reposição: Crie uma procedure que receba o ID de um filme (film_id) e um novo valor para replacement_cost. A procedure deve atualizar o valor na tabela film.

5. Contar Filmes de um Ator (OUT): Crie uma procedure que receba o actor_id (IN) e use um parâmetro (OUT) para retornar a quantidade total de filmes em que esse ator participou, pesquisando na tabela film_actor.

6. Transferência de Funcionário: Crie uma procedure sp_transferir_staff que receba o ID do funcionário (staff_id) e o ID da nova loja (store_id), atualizando a lotação dele na tabela staff.

7. Relatório de Estoque da Loja: Crie uma procedure que receba o ID da loja (store_id) e liste o título dos filmes e a quantidade de cópias disponíveis no inventário (inventory) dessa loja específica.

# EXERCÍCIOS

8. Processar Pagamento: Crie uma procedure sp_novo_pagamento que receba o customer_id, staff_id, rental_id e o amount (valor). A procedure deve inserir o registro na tabela payment usando a data e hora atuais.

9. Exclusão Segura de Categoria: Crie uma procedure para deletar uma categoria (category). Ela só deve permitir o DELETE se não houver nenhum filme vinculado a essa categoria na tabela film_category. (Dica: Use IF e COUNT).

10. Resumo Financeiro do Cliente (INOUT): Crie uma procedure que receba o customer_id. A procedure deve calcular o valor total já pago pelo cliente na tabela payment e colocar esse resultado em um parâmetro OUT. Se o valor total for maior que 100 dólares, faça a procedure retornar também uma mensagem (via SELECT simples) dizendo "Cliente VIP".