**Curso: ADS – Análise e Desenvolvimento de Sistema**

**Ano atualização: 2024/2**

**Aluno: Murilo César Ferreira**

**Nome da Trilha: OT7 – PROCEDURES**

**FICHAMENTO**

Para o desenvolvimento deste fichamento, você deverá trazer além dos conceitos exemplos práticos sobre Procedures.

1. **O que são PROCEDURES ?**

Procedures (ou Procedimentos Armazenados) são blocos de código *SQL* que são armazenados no banco de dados e podem ser executados sob demanda. Elas permitem encapsular uma sequência de instruções *SQL*, tornando-as reutilizáveis e simplificando operações repetitivas.

Uma procedure pode aceitar parâmetros de entrada, gerar saídas e realizar operações complexas como cálculos, manipulação de dados e controle de fluxo dentro do banco de dados.

1. **Explique quando usar e quando se deve aplicar os PROCEDURES;**

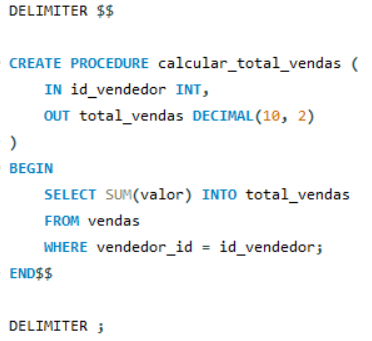
As procedures são especialmente úteis em cenários onde operações no banco de dados precisam ser executadas repetidamente ou em casos que demandem a execução de uma lógica mais elaborada diretamente no servidor de banco de dados. A aplicação de procedures é apropriada quando se busca automatizar tarefas que envolvem manipulação de dados, cálculos ou agregações que devem ser realizados no lado do banco, reduzindo a quantidade de chamadas entre a aplicação e o banco de dados e, consequentemente, melhorando o desempenho geral.

Por outro lado, seu uso deve ser avaliado com cautela. Procedures não são ideais para cenários onde a lógica de negócios da aplicação muda frequentemente, uma vez que alterações exigem modificações diretamente no banco de dados. Além disso, em arquiteturas onde se busca concentrar a lógica de negócios na camada de aplicação, o uso de procedures pode conflitar com o design pretendido.

1. **Como criar uma PROCEDURE , estrutura;**

A criação de uma procedure no MySQL envolve a utilização de comandos específicos que definem seu nome, parâmetros e o conjunto de instruções que serão executadas. Primeiramente, é necessário configurar um delimitador alternativo, geralmente $$, para evitar conflitos com o terminador padrão de comandos SQL (;). Após a definição da estrutura, parâmetros podem ser declarados com os tipos IN, OUT ou INOUT, dependendo se são usados para entrada, saída ou ambas.

Abaixo, um exemplo de procedure:



No exemplo, a procedure calcular\_total\_vendas recebe um identificador de vendedor como entrada e retorna o total das vendas associadas a ele. A lógica da procedure é encapsulada entre as palavras-chave *BEGIN* e *END*.

1. **É seguro usar as PROCEDURES ? Porquê ?**

O uso de procedures é amplamente considerado seguro, desde que seja aplicado de forma correta e consistente com as melhores práticas de desenvolvimento e administração de banco de dados. Uma de suas principais vantagens é a proteção contra ataques de injeção SQL, já que as consultas dentro das procedures são pré-compiladas e armazenadas no banco de dados, eliminando a necessidade de interpretação dinâmica de strings SQL. Além disso, procedures podem ser configuradas para operar com permissões específicas, permitindo que usuários executem operações complexas sem acesso direto às tabelas subjacentes.

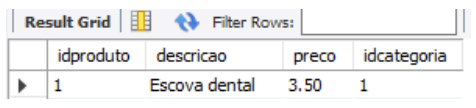
Outro aspecto que reforça sua segurança é a possibilidade de centralizar e validar operações críticas diretamente no banco de dados, o que reduz a margem de erro causada por inconsistências no lado da aplicação. Contudo, é necessário cautela, pois procedures mal projetadas podem comprometer o desempenho do banco, especialmente se utilizarem muitos recursos ou lógica excessivamente complexa. Assim, o uso de procedures deve ser equilibrado e alinhado com os objetivos e a arquitetura do sistema.

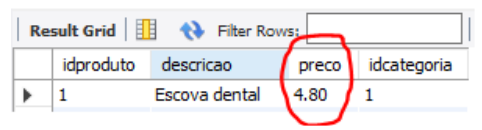
1. **Quais as principais instruções para usar os PROCEDURES.**

 Para trabalhar com procedures no MySQL, utilizam-se comandos como *CREATE PROCEDURE* para criar um novo procedimento armazenado e *CALL* para executá-lo. Por exemplo, para chamar uma procedure com parâmetros de entrada e saída, pode-se usar:

Após a execução, o valor de saída será armazenado na variável *@resultado*, que pode ser verificada com a instrução *SELECT*. Além disso, comandos como *SHOW PROCEDURE STATUS* permitem listar as procedures existentes em um banco de dados, enquanto *DROP PROCEDURE* é utilizado para excluí-las. A manutenção e documentação clara das procedures são essenciais para evitar problemas relacionados à sua evolução ou depuração. Em suma, as procedures oferecem grande flexibilidade e eficiência quando bem planejadas, tornando-se uma ferramenta valiosa em cenários de alta complexidade ou grande volume de operações no banco de dados.

**ATIVIDADES**

1. **Crie uma procedure para atualizar o preço de um produto. Para isso, a procedure deve receber o id do produto e alterar para um novo preço. Ex: Produto 1.**

**** **Executando a procedure para alteração de valor do produto o resultado será, por exemplo:**

DELIMITER $$

CREATE PROCEDURE atualizar\_preco (

IN id\_produto INT,

IN novo\_preco decimal(10,2))

BEGIN

UPDATE produto SET preco = novo\_preco WHERE idproduto=id\_produto;

IF ROW\_COUNT() = 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Nenhum produto foi atualizado. Verifique o ID do produto.';

END IF;

SELECT \* FROM produto;

END$$

DELIMITER ;

1. **Supondo que o gerente do Supermercado precisou repassar um aumento de preço dos produtos em 10% e precisa atualizar a base de dados dos produtos. Crie uma procedure onde aumente o valor de todos os produtos em 10%.**

DELIMITER $$

CREATE PROCEDURE reajustar\_precos (

IN porcent\_reajuste DECIMAL(10,2)

)

BEGIN

IF porcent\_reajuste <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'O percentual de reajuste deve ser maior que zero.';

END IF;

UPDATE produto

SET preco = preco \* (1 + porcent\_reajuste / 100);

IF ROW\_COUNT() = 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Nenhum produto foi atualizado. Os preços podem já estar ajustados.';

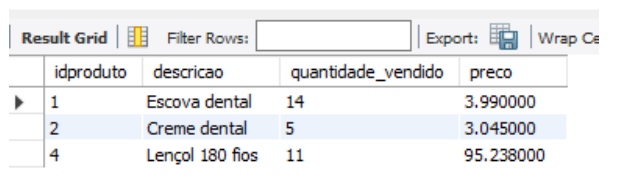
END IF;

SELECT \* FROM produto;

END$$

DELIMITER ;

1. **Crie uma procedure onde some a quantidade vendida de um determinado produto e aumente o valor do mesmo utilizando a soma da quantidade como porcentagem, ou seja, se foram vendidas 14 escovas de dentes o valor de escova de dente deve aumentar em 14%; Valor = %. Resultado:**

****

DELIMITER $$

CREATE PROCEDURE ajustar\_preco\_por\_vendas (

IN id\_produto INT -- Identificador do produto

)

BEGIN

DECLARE soma\_vendas INT;

SELECT SUM(quantidade) INTO soma\_vendas

FROM vendas

WHERE idproduto = id\_produto;

IF soma\_vendas IS NULL OR soma\_vendas = 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Nenhuma venda encontrada para o produto especificado.';

END IF;

UPDATE produto

SET preco = preco \* (1 + soma\_vendas / 100)

WHERE idproduto = id\_produto;

SELECT \* FROM produto WHERE idproduto = id\_produto;

END$$

DELIMITER ;