Aula sobre Stored Procedures

1. Conceito

Stored Procedures são conjuntos de instruções SQL armazenadas no banco de dados que podem ser executadas como uma unidade. Elas permitem encapsular a lógica de negócios, facilitando a reutilização de código e melhorando a manutenção e segurança das operações no banco de dados.

2. Vantagens das Stored Procedures

- **Reusabilidade**: Escreva a lógica uma vez e use-a em diferentes partes do sistema.
- **Segurança**: Controle o acesso aos dados e operações, evitando SQL Injection.
- **Desempenho**: Reduza o tráfego entre a aplicação e o banco de dados.
- **Manutenção**: Centralize a lógica do banco de dados, facilitando as alterações.

3. Sintaxe Básica

A sintaxe básica para criar uma stored procedure em MySQL é a seguinte:

sql
Copiar código

```
DELIMITER //

CREATE PROCEDURE nome_da_procedura (parametros)
BEGIN
    -- corpo da stored procedure
END //

DELIMITER ;
```

- **DELIMITER**: Altera o delimitador padrão (;) para permitir a definição de blocos de código mais complexos.
- **CREATE PROCEDURE**: Inicia a criação da stored procedure.
- **parametros**: Lista de parâmetros (se houver).
- **BEGIN ... END**: Delimita o corpo da procedure.

4. Exemplos de Stored Procedures

Vamos criar uma stored procedure para manipular a tabela `Vendas` usada anteriormente.

4.1. Estrutura da Tabela `Vendas`

Vamos criar uma tabela `Vendas` com os seguintes comandos:

sql
Copiar código
```sql
CREATE TABLE Vendas (
    id_venda INT AUTO_INCREMENT PRIMARY KEY,
    produto VARCHAR(100),
    valor DECIMAL(10, 2),
    data_venda DATE
);
```

4.2. Inserir Dados na Tabela `Vendas`

sql
Copiar código
```sql
INSERT INTO Vendas (produto, valor, data_venda)
VALUES
('Produto A', 150.00, '2023-01-15'),
('Produto B', 200.00, '2023-02-20'),
('Produto C', 250.00, '2023-03-05');
```

4.3. Criar uma Stored Procedure para Adicionar uma Venda

sql
Copiar código
```sql
DELIMITER //

CREATE PROCEDURE AdicionarVenda(
    IN p_produto VARCHAR(100),
    IN p_valor DECIMAL(10, 2),
    IN p_data_venda DATE
)
BEGIN
    INSERT INTO Vendas (produto, valor, data_venda)
    VALUES (p_produto, p_valor, p_data_venda);
END //

DELIMITER ;
```

Explicação:

- **IN p_produto, IN p_valor, IN p_data_venda**: Parâmetros de entrada.
- **INSERT INTO Vendas ... VALUES ...**: Insere uma nova venda na tabela `Vendas`.

4.4. Criar uma Stored Procedure para Listar Vendas

sql
Copiar código
```sql
DELIMITER //

CREATE PROCEDURE ListarVendas()
BEGIN
    SELECT * FROM Vendas;
END //

DELIMITER ;
```

Explicação:

- **SELECT * FROM Vendas**: Seleciona todos os registros da tabela `Vendas`.

4.5. Criar uma Stored Procedure com Parâmetro de Filtro

sql
Copiar código
```sql
DELIMITER //

CREATE PROCEDURE ListarVendasPorProduto(
    IN p_produto VARCHAR(100)
)
BEGIN
    SELECT * FROM Vendas
    WHERE produto = p_produto;
END //

DELIMITER ;
```

Explicação:

- **IN p_produto**: Parâmetro de entrada para filtrar as vendas por produto.
- **SELECT * FROM Vendas WHERE produto = p_produto**: Seleciona registros filtrados pelo nome do produto.

5. Executar Stored Procedures

Para executar uma stored procedure, use o comando `CALL`:

sql
Copiar código
```sql
CALL AdicionarVenda('Produto D', 300.00, '2023-04-01');
```

```
CALL ListarVendas();
CALL ListarVendasPorProduto('Produto A');
```

6. Alterar e Excluir Stored Procedures

6.1. Alterar uma Stored Procedure

Para alterar uma stored procedure, você precisa redefini-la completamente:

sql
Copiar código

```
DELIMITER //

CREATE PROCEDURE AdicionarVenda(
    IN p_produto VARCHAR(100),
    IN p_valor DECIMAL(10, 2),
    IN p_data_venda DATE,
    IN p_quantidade INT
)
BEGIN
    INSERT INTO Vendas (produto, valor, data_venda)
    VALUES (p_produto, p_valor * p_quantidade, p_data_venda);
END //

DELIMITER ;
```

6.2. Excluir uma Stored Procedure

Para excluir uma stored procedure, use o comando `DROP`:

sql
Copiar código

```
DROP PROCEDURE AdicionarVenda;
```

7. Boas Práticas

- **Nome Descritivo**: Use nomes claros e descritivos para suas stored procedures.
- **Documentação**: Comente seu código para explicar a lógica da procedure.
- **Tratamento de Erros**: Adicione tratamento de erros quando necessário.
- **Segurança**: Limite o acesso às stored procedures para garantir a segurança dos dados.
- **Testes**: Teste suas stored procedures com dados reais e cenários diferentes.

Resumo

- **Stored Procedures** encapsulam lógica SQL, oferecendo reusabilidade, segurança e melhor desempenho.
- A criação de stored procedures envolve definir parâmetros e comandos SQL encapsulados em um bloco `BEGIN ... END`.
- Stored Procedures podem ser criadas, alteradas, e excluídas usando os comandos SQL apropriados.

Esses conceitos e exemplos fornecem uma base sólida para entender e trabalhar com stored procedures no MySQL.