Banco de Dados

Aula 15 - Linguagem SQL – Stored Procedures

Apresentação

Na aula anterior, discutimos como analisar dados armazenados em várias tabelas de um modo mais simplificado usando o conceito de visões. Nesta aula, estudaremos o conceito de *Stored Procedures*, ou **procedimentos armazenados**, usados para definir operações extremamente úteis para manipulação e acesso de dados em um sistema de banco de dados. Aprenderemos a criar, executar e apagar essas estruturas, bem como utilizá-las usando o recurso de passagem de parâmetros. E, por fim, discutiremos como utilizar estruturas de controle de fluxo de dados, comuns em linguagem de programação, dentro desses procedimentos, aumentando assim a versatilidade de uso dessas estruturas.

Vídeo 01 - Apresentação

Objetivos

- Criar procedimentos armazenados com ou sem parâmetros no sistema MySQL.
- Utilizar as estruturas de controle de fluxo de dados, como estruturas IF e WHILE, na criação de procedimentos armazenados.

Stored procedures

No dia a dia do trabalho com sistemas de banco de dados, surgem problemas relacionados à necessidade de se realizar operações complexas que envolvem a realização de um ou mais comandos de consulta a tabelas em conjunto com comandos de inclusão e/ou alteração de dados. Em outros casos, deseja-se realizar operações com base em parâmetros definidos pelo usuário.

Nessas situações, o ideal é que essas operações não sejam definidas na aplicação que vai fazer uso do banco de dados, mas no próprio servidor do banco de dados. Isso garante a segurança do banco de dados, uma vez que o mesmo pode controlar o acesso a essas operações especiais, evitando o conhecimento excessivo de sua estrutura pelos desenvolvedores de aplicações. Desse modo, as *Stored procedures*, traduzidas do inglês como procedimentos armazenados, são definidas como um conjunto de comandos da linguagem SQL definidos no servidor do banco de dados e acionados por eventuais chamadas de qualquer usuário que tenha autorização para sua execução.

Além da questão de segurança do banco de dados, já que o usuário só poderá executar os procedimentos a que tiver acesso, os procedimentos armazenados ajudam a diminuir o tráfego de informações entre o usuário e o servidor, tornando mais simples o processamento das informações. Por exemplo, suponha que diversos usuários desejem fazer a consulta especificada no destaque a seguir. Essa consulta retorna uma lista de produtos vendidos e se refere ao banco de dados **sistvendas** apresentado na **Aula 13**.

```
mysql> SELECT  prod_nome
   FROM produtos, compras
   WHERE prod_codigo = comp_codproduto
   GROUP BY prod_nome;
```

A execução desse conjunto de comandos por diversos usuários simultaneamente vai gerar um grande tráfego de informações pela rede, o que não ocorre se esse conjunto de comandos for definido usando um procedimento armazenado no próprio servidor do banco de dados e o usuário solicite apenas sua

execução e o retorno do resultado, usando a sintaxe descrita no quadro a seguir. Desse modo, o usuário solicita ao servidor que execute a operação e lhe devolva o resultado, não se preocupando com a estrutura do banco de dados.

```
mysql> CALL nome_do_procedimento();
```

É possível também na criação dos procedimentos utilizar comandos comuns às linguagens de programação (que você já viu em disciplinas anteriores), como IF, WHILE, RETURN etc. Outra observação que deve ser feita é que visões não podem ser criadas em procedimentos armazenados.

Atividade 01

1. O que são procedimentos armazenados (*Stored Procedures*)?
2. Apresente duas vantagens em se utilizar esse tipo de recurso em um sistema de banco de dados.

Criando, executando e apagando procedimentos armazenados

A instrução para criar um procedimento armazenado é simples, basta utilizar o comando CREATE PROCEDURE em conjunto com a lista de parâmetros (caso necessário) a serem usados. A sintaxe de criação de um procedimento armazenado é descrita no destaque a seguir.

```
mysql> CREATE PROCEDURE nome_do_procedimento
  ([parâmetros de entrada e/ou saída])
  BEGIN
    comandos em SQL
  END;
```

Nessa expressão, no campo nome_do_procedimento, deve-se inserir o nome que se deseja atribuir para o procedimento, nome esse que deve seguir as mesmas regras usadas para os nomes das tabelas, conforme foi visto na **Aula 9**. Em seguida, caso haja, deve-se inserir a lista de parâmetros de entrada e/ou saída. Mesmo não

havendo parâmetros, devem ser inseridos os parênteses. Finalmente, entre as palavras BEGIN e END, devem ser descritos os comandos SQL que definem a operação a ser executada.

A lista de parâmetros segue a notação [IN | OUT] nome_do_parâmetro tipo_do_dado, ou seja, deve-se usar a palavra-chave que identifica se o parâmetro é de entrada (IN) ou saída (OUT). Se nada for informado, será entendido que o parâmetro é de entrada, o nome do parâmetro e o tipo que ele representa, conforme visto na **Aula 9**.

Vamos exercitar a criação de *Stored procedures* no banco de dados sistvendas para entendermos melhor o seu conceito? Na aula anterior sobre visões, um dos exemplos que foram trabalhados foi a consulta aos nomes de todos os produtos que foram vendidos. Visto que esse tipo de pesquisa tende a ser realizada diariamente num banco de dados de um sistema de vendas para gerenciamento de estoque, seria útil definir um procedimento armazenado para executar essa operação. O comando para criar essa *Stored procedure* é descrito no quadro a seguir.

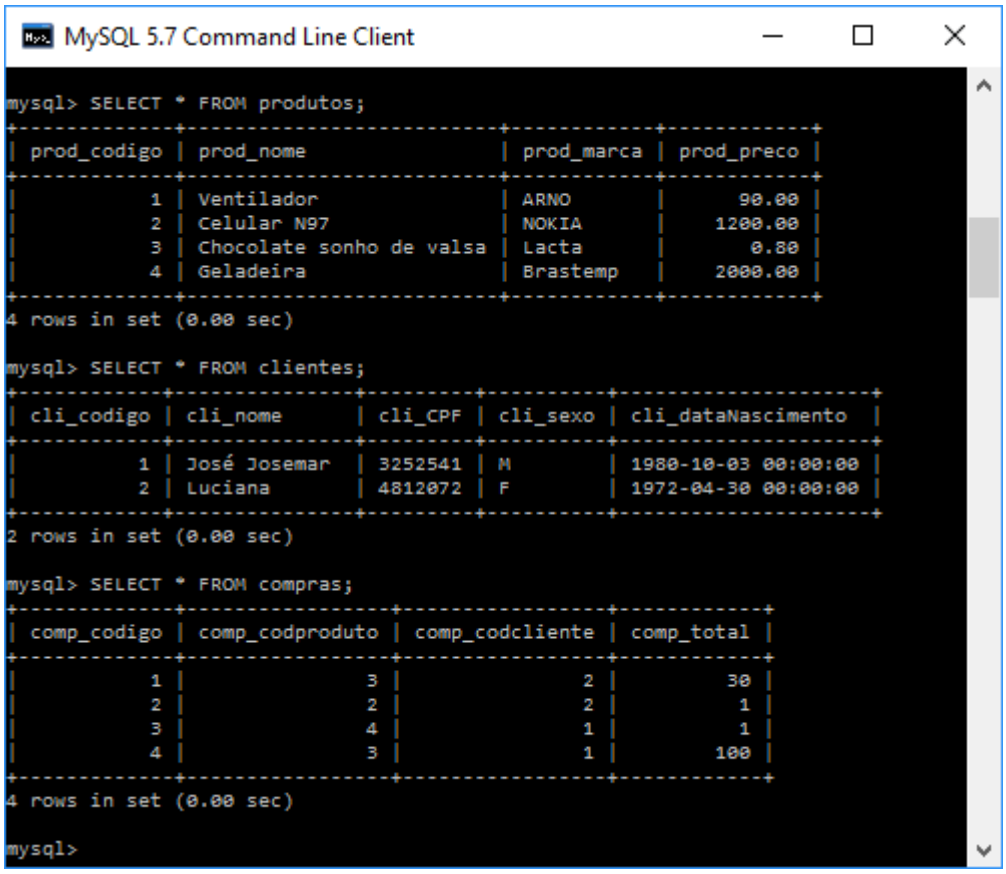
```
mysql> DELIMITER |
CREATE PROCEDURE produtos_vendidos()
   BEGIN
      SELECT  prod_nome
      FROM produtos, compras
      WHERE prod_codigo = comp_codproduto
      GROUP BY prod_nome;
   END
|
```

Observe que foi criado um procedimento armazenado denominado de produtos_vendidos, não sendo definido qualquer parâmetro, e os comandos que definem sua operação encontram-se entre as palavras-chave BEGIN e END. A execução do procedimento armazenado é feita usando o comando apresentado no quadro a seguir.

```
mysql> CALL produtos_vendidos();
```

Para analisarmos a aplicação dos procedimentos armazenados no banco de dados sistvendas, apresentam-se, na **Figura 1**, os dados presentes nas tabelas do banco de dados.

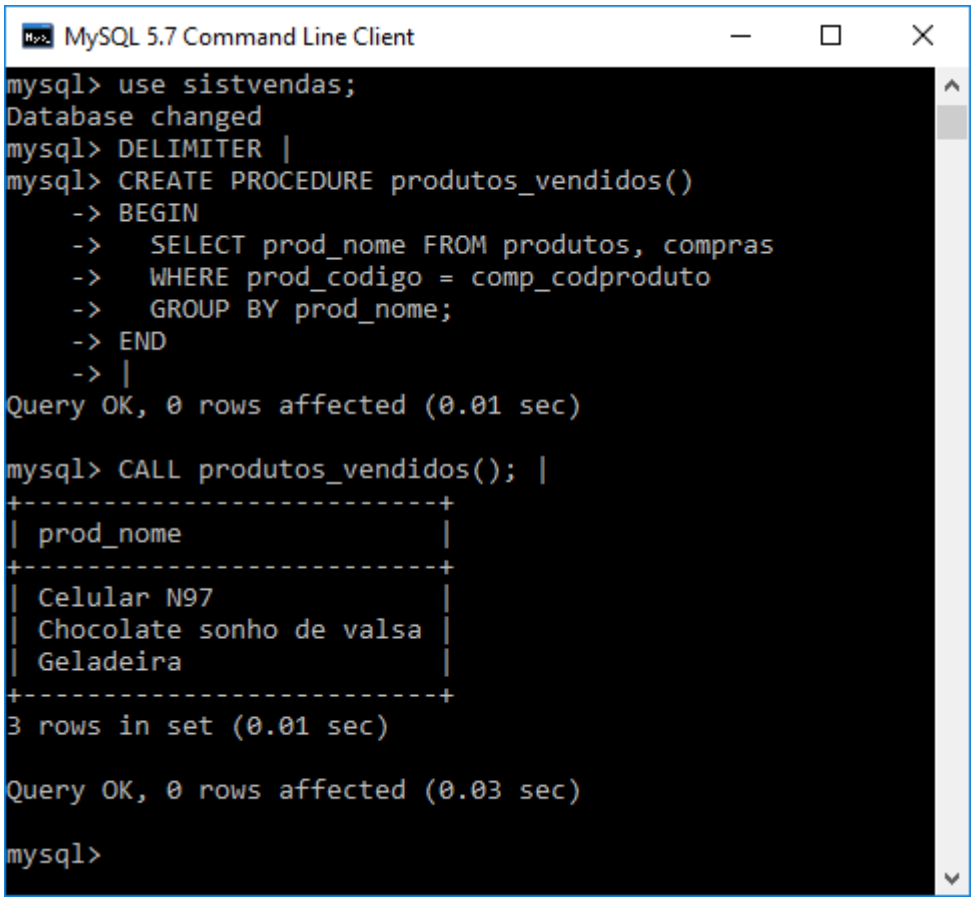
Figura 01 - Tela do MySQL após os comandos SELECT * FROM **produtos**, SELECT * FROM **clientes** e SELECT * FROM **compras**.



Fonte: MySQL 5.7 Command Line Client

A resposta do sistema SGBD para os comandos anteriores é ilustrada na **Figura 2**. É interessante notar o uso do operador DELIMITER antes da criação do procedimento. Ele é usado para redefinir o caractere delimitador de comandos como "|", isso é feito para que possamos usar o caractere ";" no meio do procedimento. Caso não efetuemos essa troca, o procedimento será enviado pela metade e uma mensagem de erro será enviada ao terminal, por problemas na sintaxe.

Figura 02 - Tela do MySQL após os comandos CREATE PROCEDURE e CALL.



Fonte: MySQL 5.7 Command Line Client

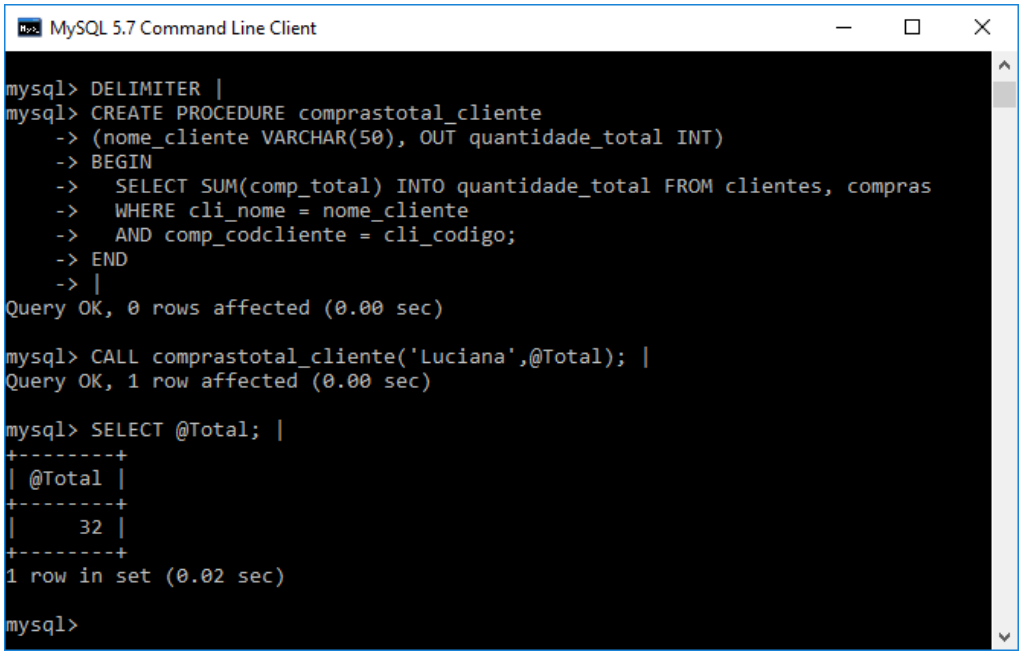
Vídeo 02 - Introdução a *Stored Procedures*

Agora que aprendemos a criar procedimentos armazenados, que tal aprendermos a criá-los usando parâmetros? Para exemplificar essa situação, suponha que desejamos saber a quantidade total de produtos comprados por um determinado cliente no nosso banco de dados sistvendas. Para isso, criamos o procedimento descrito no destaque a seguir.

```
mysql> DELIMITER |
  CREATE PROCEDURE comprastotal_cliente
  (nome_cliente varchar(50), OUT quantidade_total int)
  BEGIN
     SELECT  SUM(comp_total) INTO quantidade_total
     FROM clientes, compras
     WHERE cli_nome=nome_cliente
     AND comp_codcliente=cli_codigo;
  END
  |
```

Analisando cuidadosamente os comandos, podemos ver que o procedimento é definido com dois parâmetros, um de entrada denominado de nome_cliente e outro de saída denominado de quantidade_total. Perceba que não é necessário colocar a cláusula IN para definir o parâmetro como sendo de entrada, mas se deve usar a cláusula OUT na definição do parâmetro de saída. Observe também que após o nome dos parâmetros é definido seu tipo, como visto na **Aula 9**. Outra novidade mostrada na sequência de comandos é a estrutura SELECT ... INTO, cuja função é direcionar o resultado da consulta para uma variável definida após a cláusula INTO. No nosso caso, o resultado do SELECT é atribuído à variável quantidade_total, que é um parâmetro de saída. A **Figura 3** ilustra a resposta do SGBD após a criação do procedimento.

Figura 03 - Tela do MySQL após os comandos CREATE PROCEDURE, CALL e SELECT.



Fonte: MySQL 5.7 Command Line Client

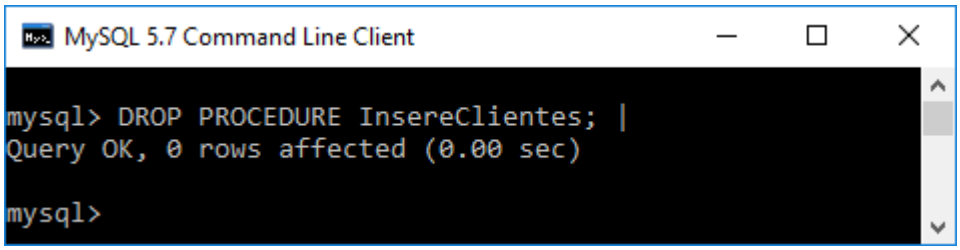
Também é ilustrada na **Figura 3** a execução do procedimento, consultando o total de produtos comprados pela cliente Luciana e armazenando o valor numa variável denominada Total. Ao se passar o nome de uma variável para um procedimento, é adicionado o caractere "@" antes do nome da variável. É interessante observar que durante o processamento interno do procedimento, como ocorre em qualquer linguagem de programação, a variável Total será associada ao parâmetro quantidade_total.

Observe a utilização do comando SELECT para verificar o conteúdo da variável Total.

Para excluir um procedimento armazenado do banco de dados, você deve utilizar o comando DROP, o qual tem sua sintaxe descrita no quadro a seguir. E a resposta do SGBD a esse comando é ilustrada na **Figura 4**.

```
1 mysql>DROP PROCEDURE nome_do_procedimento;
```

Figura 04 - Tela do MySQL após o comando DROP PROCEDURE.



Fonte: MySQL 5.7 Command Line Client

Vídeo 03 - Parâmetros

Atividade 02

Vamos praticar um pouco para que você se familiarize com os comandos apresentados.

Acesse o banco de dados **sispagamentos (Aula 14)** e elabore o que se pede.

1. Crie um procedimento armazenado para obter a lista dos empregados que não pagam imposto de renda (IR).
2. Crie um procedimento armazenado com passagem de parâmetros para cadastro de funcionários, inserindo seus dados na tabela empregados.
3. Crie um procedimento armazenado que retorne por uma variável de saída o número total de empregados que não tem desconto de imposto de renda.
4. Exclua do banco de dados os procedimentos anteriores.

Usando estruturas de programação em procedimentos armazenados

Os procedimentos armazenados são estruturas extremamente poderosas que facilitam a interação dos usuários com o servidor de banco de dados, além de ajudar no controle de acesso aos dados, aumentando a segurança do sistema, como mencionado anteriormente. Um dos requisitos que contribuem para essas características dos procedimentos está na possibilidade de uso de estruturas de controle de fluxo de dados, como estruturas de decisão e de repetição.

Você está familiarizado com algumas dessas estruturas, pois já foram apresentadas nas disciplinas de Lógica e Linguagem de Programação. As estruturas de controle de fluxo definidas no MySQL são: IF, CASE, LOOP, LEAVE, ITERATE, REPEAT e WHILE. Vamos discutir as duas mais usadas, o IF, que define estruturas de decisão, e o WHILE, que define estruturas de repetição. O destaque a seguir apresenta a estrutura do IF em MySQL.

```
IF condição 1 THEN comandos em SQL
   [ELSEIF condição 2 THEN comandos em SQL] ...
   [ELSE comandos em SQL]
END IF
```

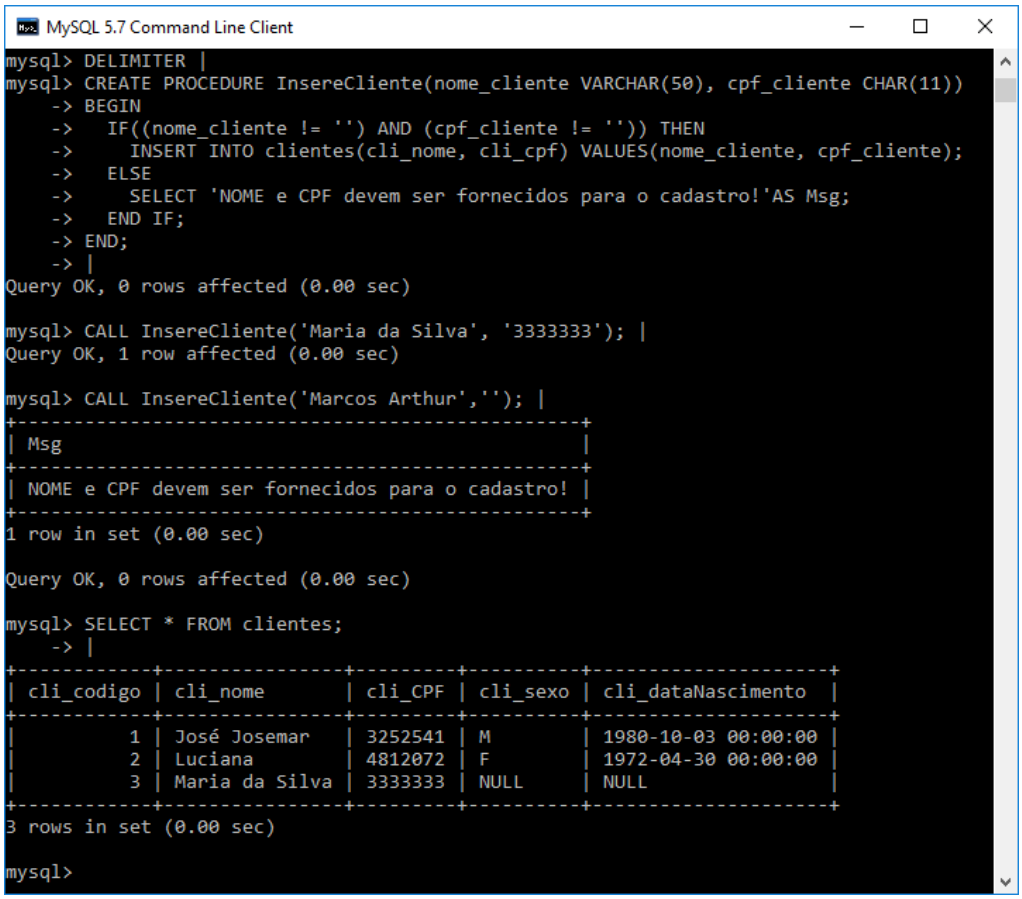
A sintaxe pode ser explicada do seguinte modo: SE (IF) a condição 1 é satisfeita, um grupo de comandos em SQL é executado, SENÃO SE (ELSEIF) a condição 2 é satisfeita, outro grupo de comandos em SQL é executado e, finalmente, se nenhuma (SENÃO – ELSE) das condições anteriores for satisfeita, o último grupo de comando é executado. Lembrando que os termos entre chaves (“[“ e “]”) são opcionais.

Para exemplificar a utilização da estrutura IF ... END IF num procedimento armazenado, imagine que precisamos definir um procedimento para inserção de clientes no nosso banco de dados sistvendas, mas para isso, devemos garantir que sejam informados o nome e o CPF do cliente. O destaque a seguir apresenta uma solução para esse problema. Analise com atenção a sequência de comandos apresentada.

```
DELIMITER |
CREATE PROCEDURE InsereCliente (nome_cliente varchar(50), cpf_cliente char(11))
BEGIN
   IF ((nome_cliente != '') AND (cpf_cliente != '')) THEN
      INSERT INTO clientes (cli_nome, cli_cpf) VALUES (nome_cliente, cpf_cliente);
   ELSE
      SELECT 'NOME e CPF devem ser fornecidos para o cadastro!' AS Msg;
   END IF;
END;
|
```

Observe que no quadro anterior é definido um procedimento com dois parâmetros de entrada (nome_cliente, cpf_cliente) que serão inseridos na tabela **clientes** SE os respectivos parâmetros não forem nulos, SENÃO é apresentada na tela uma mensagem de erro. A condição do IF é definida usando o operador "!=" que significa DIFERENTE DE e o operador lógico AND, ou seja, SE nome_cliente DIFERENTE DE vazio E cpf_cliente DIFERENTE DE vazio, a operação de inserção é executada. Observe também o uso do comando SELECT para impressão de mensagens na tela. A **Figura 5** apresenta a construção do procedimento descrito anteriormente no MySQL.

Figura 05 - Tela do MySQL após o comando CREATE PROCEDURE com estrutura IF/ELSE, CALL e SELECT.



Fonte: MySQL 5.7 Command Line Client

Além da criação do procedimento discutido no quadro anterior, a **Figura 5** traz um exemplo de uso do procedimento informando corretamente os dados de inserção, outro em que os dados não são informados de forma correta e os dados contidos na tabela clientes após a execução do procedimento. Lembre que, pela definição da tabela clientes, a coluna cli_codigo é chave primária, de modo que, para não haver problemas na utilização do procedimento, é necessário que esta coluna seja do tipo **AUTO_INCREMENT**, além disso a tabela clientes deverá permitir a inserção de novos registros com os atributos cli_sexo e cli_dataNascimento com os valores NULL, pois a nossa procedure que criamos utiliza apenas o nome e CPF para inserir um novo cliente.

Vídeo 04 - Estruturas de Controle

Agora que vimos a utilização das estruturas de decisão, vamos discutir um dos exemplos de estrutura de repetição. O quadro a seguir apresenta a estrutura do WHILE em MySQL. A sintaxe pode ser explicada do seguinte modo: ENQUANTO (WHILE) a condição é satisfeita EXECUTE (DO) um grupo de comandos em SQL.

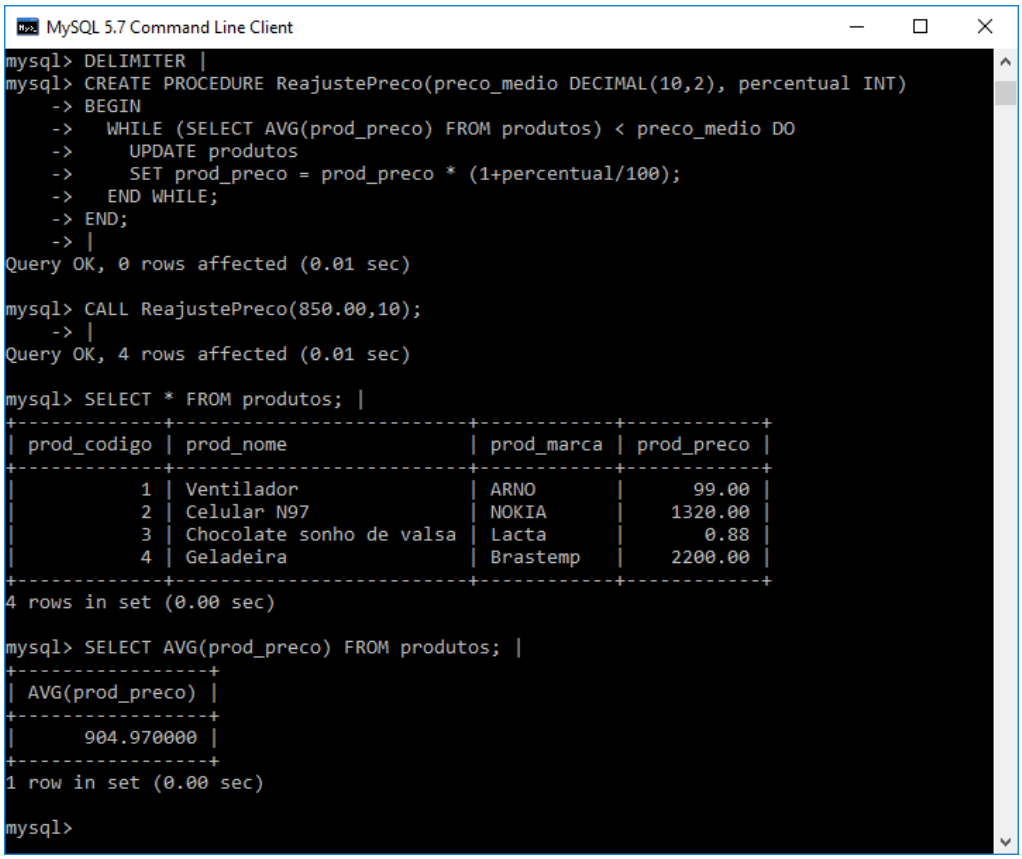
```
WHILE condição DO
   comandos em SQL
END WHILE
```

Para exemplificar a utilização da estrutura WHILE ... END WHILE num procedimento armazenado, imagine que precisamos definir um procedimento para reajustar os preços da tabela produtos do nosso banco de dados sistvendas, com a condição de que os preços de todos os produtos sejam reajustados de um percentual fixo, enquanto um determinado preço médio não seja atingido. O quadro a seguir apresenta uma solução para esse problema. Analise com atenção a sequência de comandos apresentada.

```
mysql> DELIMITER |
CREATE PROCEDURE ReajustePreco (preco_medio DECIMAL(10,2), percentual INT)
  BEGIN
    WHILE (SELECT AVG(prod_preco) FROM produtos) < preco_medio DO
        UPDATE produtos
        SET prod_preco = prod_preco*(1+percentual/100);
    END WHILE;
  END;
|
```

Observe que no quadro anterior é definido um procedimento com dois parâmetros de entrada (preco_medio, percentual), que são usados dentro de uma estrutura de repetição (WHILE) de modo que, ENQUANTO a média dos preços dos produtos for menor que preco_medio, todos os produtos têm seu valor atualizado de um determinado percentual. A **Figura 6** apresenta a construção do procedimento descrito anteriormente no MySQL.

Figura 06 - Tela do MySQL após o comando CREATE PROCEDURE com estrutura WHILE, CALL e SELECT.



Fonte: MySQL 5.7 Command Line Client

Além da criação do procedimento discutido no destaque anterior, a **Figura 6** traz um exemplo de uso do procedimento informando como parâmetros de entrada, preco_medio igual a 850.00 e percentual igual a 10; a tabela de produtos em que se pode constatar o aumento dos preços quando comparada com tabela similar mostrada na **Figura 1** e o uso do SELECT para calcular o novo preço médio depois da aplicação do aumento.

Vamos treinar um pouco o uso das estruturas de decisão e repetição?

Atividade 03

Vamos praticar um pouco para que você se familiarize com os comandos apresentados.

Acesse o banco de dados sispagamentos (**Aula 14**) e elabore o que se pede:

1. Crie um procedimento armazenado com passagem de parâmetros para cadastro de funcionários, inserindo seus dados na tabela empregados.
2. Crie um procedimento armazenado com passagem de parâmetros para cadastro de funcionários, atualizando seus dados na tabela empregados.
3. Crie um procedimento armazenado com passagem de parâmetros para cadastro de funcionários, apagando seus dados na tabela empregados. Exclua do banco de dados os procedimentos anteriores.
4. Nos três casos, certifique-se de que os valores estão sendo informados de forma correta, caso contrário, imprima uma mensagem de erro.

Conclusão

Encerramos por aqui nossa aula sobre procedimentos armazenados na linguagem SQL. Na próxima aula, aprenderemos a utilizar outra importante estrutura para operação com banco de dados conhecidas como funções.

Faça a autoavaliação com atenção e veja se precisa parar e refletir mais sobre a criação e o uso dos procedimentos armazenados, principalmente como as estruturas de controle de fluxo dados, que podem ser úteis em diferentes aplicações.

Escreva no seu caderno todos os comandos SQL (e respectivas funções) aprendidos nesta aula para não esquecer.

Bons estudos e boa sorte!

Resumo

Nesta aula, você estudou a criação e utilização dos procedimentos armazenados. Aprendeu a usar os comandos CREATE PROCEDURE, CALL e DROP PROCEDURE para criar, executar e apagar um procedimento. Viu também como utilizar o recurso de lista de parâmetros para definir parâmetros de entrada e saída para os procedimentos. Por fim, você observou o uso dos procedimentos em conjunto com as estruturas de controle de fluxo de dados, IF ... END IF e WHILE ... END WHILE.

Autoavaliação

1. Execute as operações a seguir, considerando o banco de dados sispagamentos.
    a. Crie uma *Stored Procedure* que mostre para cada empregado a cidade em que ele mora.
    b. Crie uma *Stored Procedure* para exibir quantos empregados moram em uma determinada cidade. Passe como parâmetro de entrada o nome da cidade.
    c. Crie uma *Stored Procedure* que, passando como parâmetro o código do empregado, mostre o nome dele e seu salário bruto e líquido (após os descontos).
    d. Crie uma *Stored Procedure* que tenha como entrada o código do empregado e mostre como parâmetros de saída o seu nome e cidade onde reside.
2. Execute as operações a seguir, considerando o banco de dados **sisvendas**:
    a. Crie uma *Stored Procedure* que tenha como parâmetro de entrada o código da compra e mostre como parâmetro de saída a quantidade de itens total nessa compra.

b. Crie uma *Stored Procedure* para informar a classificação de cada cliente. Caso a soma do preço dos produtos comprados pelo cliente seja maior que R$ 1000, sua categoria é cliente VIP; caso seja entre R$ 500 e 1000, sua categoria é cliente normal; caso seja menor que R$ 500, cliente popular. O código do cliente deve ser passado como parâmetro.

3. Consulte o manual de referência do MySQL (disponível em https://dev.mysql.com/doc/refman/5.7/en/flow-control-statements.html) e analise as estruturas de controle de fluxo de dados CASE e REPEAT.

Referências

BEIGHLEY, L. **Use a cabeça SQL**. Rio de Janeiro: Editora AltaBooks, 2008.

MySQL 5.7 Reference Manual. Disponível em: <http://dev.mysql.com/doc/refman/5.7/en/>. Acesso em: 28 jan. 2017.

WIKIPÉDIA. **SQL**. Disponível em: <http://pt.wikipedia.org/wiki/SQL>. Acesso em: 26 set. 2012.