

Fundamentos de Banco de Dados

SQL
(Store Procedure e Gatilhos)

Cleyton Carvalho da Trindade

Stored Procedures

- É uma coleção de comandos SQL, que encapsula uma série de tarefas repetitivas, relativas ao acesso a banco, aceita parâmetros de entrada e retorna um valor de status ou conjunto de registros.
 - Por que usar um Stored Procedure?
 - Ajudam a reduzir o tráfego na rede, a melhorar o desempenho de consultas, a criar mecanismos de segurança e simplificar o código da aplicação, já que não haverá a necessidade de manter consultas SQL de várias linhas misturadas a toda lógica da sua aplicação.
-

Stored Procedures

- Os procedimentos armazenados, quando criados e compilados, são inseridos em uma tabela chamada ROUTINES no banco de dados INFORMATION_SCHEMA, que é o dicionário de dados do MySQL. Para listarmos todos os stored routines (Stored Procedure e Functions), basta emitirmos o seguinte comando no mysql Workbench:

```
SELECT * FROM INFORMATION_SCHEMA.ROUTINES;
```

Stored Procedures - Sintaxe

```
CREATE PROCEDURE proc_name([parameters,  
...])
```

```
[BEGIN]
```

```
    corpo_da_rotina;
```

```
[END]
```

Stored Procedures - Sintaxe

- **proc_name**: Nome do procedimento;
 - **parameters**: nessa parte do procedimento, informaremos os parâmetros da seguinte forma: [IN | OUT | INOUT] nome_parametro tipo_dado.
 - **corpo_da_rotina**: onde são definidos os comandos SQL que farão alguma manipulação e/ou defenderão alguma lógica, podendo retornar ou não algum resultado.
-

Stored Procedures - Exemplo

■ Exemplo

```
CREATE TABLE tbl_correntista (  
    correntista_id int auto_increment primary key,  
    correntista_nome varchar(60) not null unique,  
    correntista_cpf varchar(20) not null,  
    dt_cadastro timestamp default current_timestamp )
```

Stored Procedures - Exemplo

```
DELIMITER //
```

```
CREATE PROCEDURE mySp_correntistaInsert(v_nome  
    VARCHAR(60), v_cpf VARCHAR(20))
```

```
BEGIN
```

```
    IF ((v_nome != "") && (v_cpf != "")) THEN
```

```
        INSERT INTO tbl_correntista (correntista_nome,  
        correntista_cpf) VALUES (v_nome, v_cpf);
```

```
    ELSE
```

```
        SELECT 'NOME e CPF devem ser fornecidos para o  
        cadastro!' AS Msg;
```

```
    END IF;
```

```
END;
```

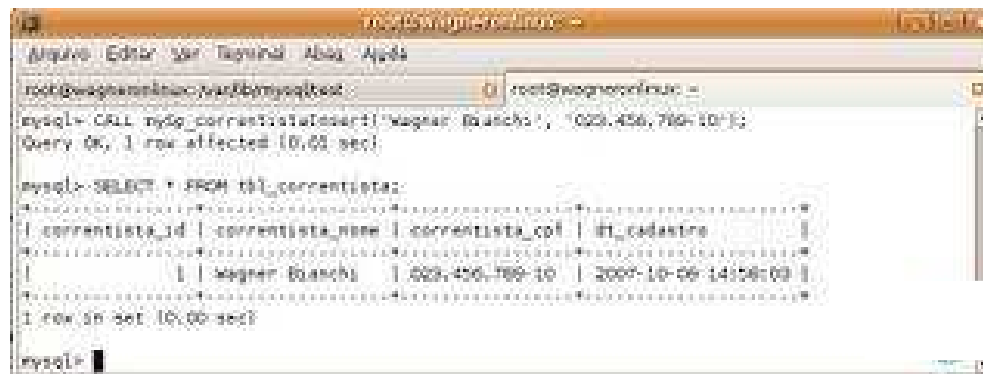
```
//
```

Stored Procedures - Exemplo

CALL mySp_correntistaInsert('Wagner Bianchi', '023.456.789-10');

```
+-----+
| MSG                                     |
+-----+
| NOME e CPF devem ser fornecidos para o cadastro! |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```



```
mysql> CALL mySp_correntistaInsert('Wagner Bianchi', '023.456.789-10');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM tbl_correntistas;
+-----+-----+-----+-----+
| correntista_id | correntista_nome | correntista_cpf | dt_cadastro |
+-----+-----+-----+-----+
| 1 | Wagner Bianchi | 023.456.789-10 | 2007-10-06 14:04:00 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Gatilhos (Triggers)

- São procedimentos especiais ativados quando ocorre uma inserção, atualização ou exclusão em uma tabela
 - Não são chamados diretamente pelo usuário.
 - Possibilitam:
 - Manter a integridade dos dados: atualizar tabelas associadas
 - Cria logs do sistema: a cada inclusão
 - Notificações de alterações no banco aos usuários
 - Validação de restrições de integridade mais complexas que as suportadas diretamente pelo SGBD
-

Triggers - Sintaxe

```
CREATE  
TRIGGER nome_trigger  
Tempo_trigger Evento_trigger  
ON nome_tabela FOR EACH ROW  
BEGIN  
    declaração_trigger  
END
```

Triggers - Sintaxe

- **Nome_trigger**: define o nome do procedimento, por exemplo, trg_test;
 - **Tempo_trigger**: define se o TRIGGER será ativado antes (BEFORE) ou depois (AFTER) do comando que o disparou;
 - **Evento_trigger**: aqui se define qual será o evento, INSERT, REPLACE, DELETE ou UPDATE;
 - **nome_tabela**: nome da tabela onde o TRIGGER ficará "pendurado" aguardando o trigger_event;
 - **declarações_trigger**: as definições do que o o TRIGGER deverá fazer quando for disparado
-

Triggers

■ Exemplo

- Trigger que atualiza o campo VALOR da tabela VENDAS cada vez que se alterar a tabela de itens

```
CREATE TRIGGER Insert_Item
```

```
AFTER INSERT ON Item
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE Venda set valor = valor + New.ValorTotal
```

```
    WHERE Pedido = New.Pedido;
```

```
END
```

O alias NEW indica o registro que está sendo inserido

Triggers

■ Exemplo

- Sistema para aluguel de carros - Atualização da quilometragem rodada por um carro após sua devolução

```
CREATE TRIGGER tr_kmrodados
```

```
AFTER UPDATE ON reservas
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE carros SET quilometragem = quilometragem +  
        NEW.quilometrosrodados WHERE NEW.carro_reserva =  
        carros.id;
```

```
END;
```

Triggers

■ Exemplo

- ❑ Exclusão de um item da tabela

```
CREATE TRIGGER DeleleItem
```

```
AFTER DELETE ON Item
```

```
BEGIN
```

```
    UPDATE Vendas SET valor = valor - OLD.ValorTotal
```

```
    WHERE Pedido = OLD.Pedido;
```

```
END
```

O alias OLD indica o registro que está sendo apagado