Prof. Taciano Balardin

www.taciano.pro.br

taciano@ulbra.edu.br

2015-2

Banco de Dados II

E-MAIL DE CONTATO:
taciano@ulbra.edu.br

SITE DA DISCIPLINA:
http://www.taciano.pro.br/

TRIGGERS

EXERCÍCIO

MySQL IF Sintaxe

```
IF if_expression THEN commands
   [ELSEIF elseif_expression THEN commands]
   [ELSE commands]
END IF;
```

```
BEGIN
UPDATE produtos SET estoque = estoque + (OLD.qtd - NEW.qtd)
WHERE id = NEW.produto_id;
END
```

Utilizando as tabelas de *produtos* e *itensvenda*:
Criar uma trigger que vai **atualizar o valor do estoque** no momento em que um registro de venda for **ALTERADO** no banco de dados.

DESAFIO

Stored Procedures e Functions

BANCO DE DADOS II
AULA 07

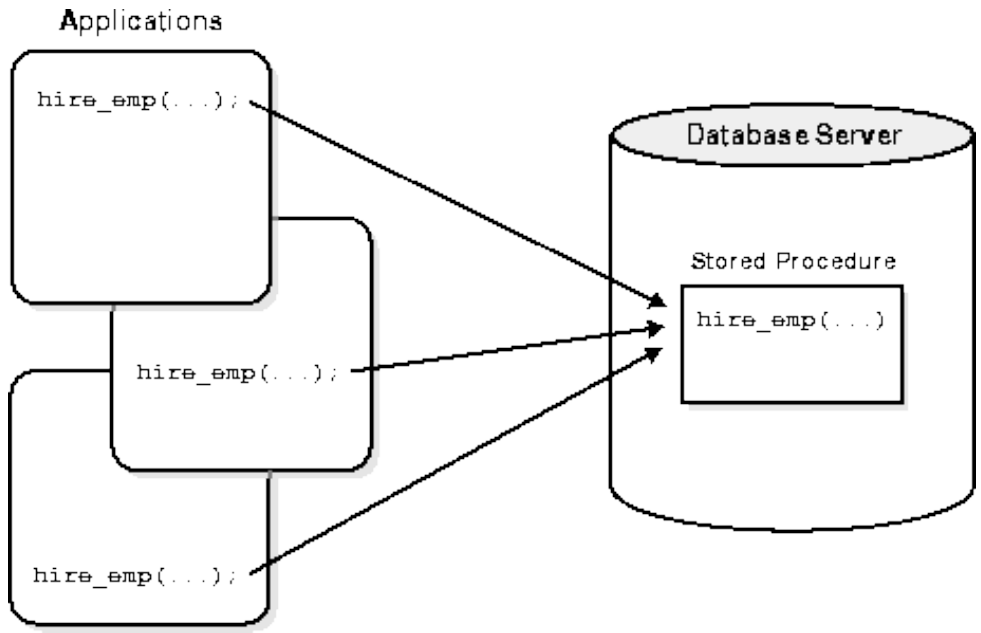
Prof. Taciano Balardin

Stored Procedures

- Stored procedures são rotinas definidas no banco de dados, identificadas por um nome pelo qual podem ser invocadas.
- Um procedimento desses pode **executar uma série de instruções**, **receber parâmetros** e **retornar valores**.

Vantagens

- 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.



Desvantagens

- Alguém que tenha **acesso não autorizado** ao banco de dados **poderá visualizar e alterar** os procedimentos.
- **Requer maior conhecimento** de manipulação do banco de dados (SQL) para realizar as operações internamente.
- Dificuldade para Debug.

Stored Procedures - Sintaxe

CREATE PROCEDURE *nome_procedimento***(** [parâmetros, ...] **)**

[características]

[**BEGIN**]

corpo_da_rotina;

[**END**]

Onde consta *nome_procedimento*, deve-se informar o nome que identificará o procedimento armazenado. Este nome segue as mesmas regras para definição de variáveis, não podendo iniciar com número ou caracteres especiais.

Stored Procedures - Sintaxe

CREATE PROCEDURE *nome_procedimento***(** [parâmetros, ...] **)**

[características]

[**BEGIN**]

corpo_da_rotina;

[**END**]

Os “parâmetros” são opcionais e, caso não sejam necessários, devem permanecer apenas os parênteses vazios na declaração do procedure. Para que um procedimento receba parâmetros, é necessário seguir certa sintaxe (dentro dos parênteses).

Stored Procedures - Sintaxe Parâmetros

(MODO nome TIPO, MODO nome TIPO, MODO nome TIPO, ...**)**

- O **nome** dos parâmetros também segue as mesmas regras de definição de variáveis.
- O **TIPO** nada mais é que do tipo de dado do parâmetro (int, varchar, decimal, etc).
- O **MODO** indica a forma como o parâmetro será tratado no procedimento, pode ser:
  - **IN**: indica que o parâmetro é apenas para entrada/recebimento de dados, não podendo ser usado para retorno.
  - **OUT**: usado para parâmetros de saída. Para esse tipo não pode ser informado um valor direto (como 'teste', 1 ou 2.3), deve ser passada uma variável "por referência".
  - **INOUT**: como é possível imaginar, este tipo de parâmetro pode ser usado para os dois fins (entrada e saída de dados). Nesse caso também deve ser informada uma variável e não um valor direto.

Function - Prática

```
CREATE FUNCTION hello(valor VARCHAR(255))
        RETURNS VARCHAR(255)
BEGIN
        RETURN CONCAT('Hello, ',valor,'!');
END
```

```
mysql> SELECT hello('world');
```

Stored Procedures - Prática IN

```
CREATE PROCEDURE sortear_func(IN quantidade INT)
BEGIN
  SELECT * FROM funcionario ORDER BY RAND() LIMIT quantidade;
END
```

```
mysql> CALL sortear_func(5);
```

Stored Procedures - Prática OUT

```
CREATE PROCEDURE conta_func(OUT quantidade INT)
BEGIN
  SELECT COUNT(id) INTO quantidade FROM funcionario;
END
```

```
mysql> CALL conta_func(@qtd);
mysql> SELECT @qtd;
```

Stored Procedures - Prática INOUT

```
CREATE PROCEDURE elevar_ao_quadrado(INTOUT numero INT)
BEGIN
  SET numero = numero * numero;
END
```

```
mysql> SET @valor = 5;
mysql> CALL elevar_ao_quadrado(@valor);
mysql> SELECT @valor;
```