

Banco de Dados – C07

Felipe Tagawa Reis

Procedures
Functions

Procedures

Procedures são blocos de código SQL que podem ser armazenados no banco de dados e executados quando necessário. De forma geral, executam comandos, modificam dados, mas não retornam valores diretamente. Prioriza consistência e segurança, na medida em que uma funcionalidade seja utilizada em diferentes lugares, a manutenção é facilitada e sua centralização evita a chance de erros.

Procedures

Procedures são blocos de código SQL que podem ser armazenados no banco de dados e executados quando necessário. De forma geral, executam comandos, modificam dados, mas não retornam valores diretamente. Prioriza consistência e segurança, na medida em que uma funcionalidade seja utilizada em diferentes lugares, a manutenção é facilitada e sua centralização evita a chance de erros.

Desvantagem: Requer maior processamento por parte do servidor.

Delimita início e fim de blocos de
Código ('\$ \$' no lugar de ';')

Procedures

```
1  DELIMITER $$
2
3  DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5  CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
percentual_aumento DECIMAL(5,2))$$
6
7  BEGIN
8      UPDATE Funcionarios
9      SET salario = salario * (1 + percentual_aumento / 100)
10     WHERE id = funcionario_id;
11
12     SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```

Delimita início e fim de blocos de
Código ('\$\$_ no lugar de ';')

Procedures

Criação da
Procedure

```
1 DELIMITER $$
2
3 DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5 CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
  percentual_aumento DECIMAL(5,2))$$
6
7 BEGIN
8     UPDATE Funcionarios
9     SET salario = salario * (1 + percentual_aumento / 100)
10    WHERE id = funcionario_id;
11
12    SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```

Delimita início e fim de blocos de Código ('\$\$' no lugar de ';')

Procedures

Criação da Procedure

```
1 DELIMITER $$
2
3 DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5 CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
  percentual_aumento DECIMAL(5,2))$$
6
7 BEGIN
8     UPDATE Funcionarios
9     SET salario = salario * (1 + percentual_aumento / 100)
10    WHERE id = funcionario_id;
11
12    SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```

Parâmetros

Delimita início e fim de blocos de Código ('\$\$' no lugar de ';')

Procedures

Criação da Procedure

Início do Corpo

Parâmetros

```
1 DELIMITER $$
2
3 DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5 CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
  percentual_aumento DECIMAL(5,2))$$
6
7 BEGIN
8     UPDATE Funcionarios
9     SET salario = salario * (1 + percentual_aumento / 100)
10    WHERE id = funcionario_id;
11
12    SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```

Delimita início e fim de blocos de Código ('\$\$' no lugar de ';')

Procedures

Criação da Procedure

Início do Corpo

Corpo

Parâmetros

```
1 DELIMITER $$
2
3 DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5 CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
  percentual_aumento DECIMAL(5,2))$$
6
7 BEGIN
8     UPDATE Funcionarios
9     SET salario = salario * (1 + percentual_aumento / 100)
10    WHERE id = funcionario_id;
11
12    SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```


Delimita início e fim de blocos de Código ('\$\$' no lugar de ';')

Procedures

Criação da Procedure

Início do Corpo

Corpo

Finaliza toda a Procedure

```
1 DELIMITER $$
2
3 DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5 CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
  percentual_aumento DECIMAL(5,2))$$
6
7 BEGIN
8     UPDATE Funcionarios
9     SET salario = salario * (1 + percentual_aumento / 100)
10    WHERE id = funcionario_id;
11
12    SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```

Parâmetros

Delimita início e fim de blocos de Código ('\$\$' no lugar de ';')

Procedures

Criação da Procedure

Início do Corpo

Corpo

Finaliza toda a Procedure

```
1 DELIMITER $$
2
3 DROP PROCEDURE IF EXISTS AtualizarSalario$$
4
5 CREATE PROCEDURE AtualizarSalario(IN funcionario_id INT, IN
  percentual_aumento DECIMAL(5,2))$$
6
7 BEGIN
8     UPDATE Funcionarios
9     SET salario = salario * (1 + percentual_aumento / 100)
10    WHERE id = funcionario_id;
11
12    SELECT 'Salário atualizado com sucesso!' AS Resultado;
13 END $$
14
15 DELIMITER ;
```

Parâmetros

Restaura o delimitador padrão para ';'

Procedures - Delimiter

Forma Incorreta

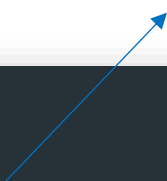
```
-- X PROBLEMA: Sem delimiter personalizado  
CREATE PROCEDURE Teste()  
BEGIN  
    SELECT 'Oi'; -- MySQL pensa que a procedure termina aqui!  
    SELECT 'Tchau';  
END; -- Isso nunca seria executado
```

Forma Correta

```
DELIMITER $$  
CREATE PROCEDURE Teste()  
BEGIN  
    SELECT 'Oi'; -- MySQL ignora este ;  
    SELECT 'Tchau'; -- E este também  
END $$ -- Só termina aqui!  
DELIMITER ;
```

Procedures - Utilização

Chamando a Procedure



```
16  
17 CALL AtualizarSalario(123, 15.5); -- Aumenta 15.5% no  
salário do funcionário ID 123  
18  
19 CALL AtualizarSalario(456, 10.0); -- Aumenta 10% no salário  
do funcionário ID 456
```

Procedures - Utilização

Chamando a Procedure

```
16  
17 CALL AtualizarSalario(123, 15.5); -- Aumenta 15.5% no  
salário do funcionário ID 123  
18  
19 CALL AtualizarSalario(456, 10.0); -- Aumenta 10% no salário  
do funcionário ID 456
```

```
DROP PROCEDURE AtualizarSalario;
```

Apagando a Procedure

Procedures – While + IF

```
1  DELIMITER $$
2
3  DROP PROCEDURE IF EXISTS ContarNumeros$$
4
5  CREATE PROCEDURE ContarNumeros(
6      IN limite INT
7  )$$
8
9  BEGIN
10     -- Declaração de variáveis
11     DECLARE contador INT DEFAULT 1;
12     DECLARE pares INT DEFAULT 0;
13     DECLARE ímpares INT DEFAULT 0;
14
15     -- Loop WHILE para contar de 1 até o limite
16     WHILE contador <= limite DO
17
18         -- Verifica se o número é par ou ímpar
19         IF contador MOD 2 = 0 THEN
20             -- É par
21             SET pares = pares + 1;
22         ELSE
23             -- É ímpar
24             SET ímpares = ímpares + 1;
25         END IF;
26
27         -- Incrementa contador
28         SET contador = contador + 1;
29
30     END WHILE;
31
32     -- Retorna resultado
33     SELECT
34         limite AS 'Limite',
35         pares AS 'Números Pares',
36         ímpares AS 'Números Ímpares';
37
38 END $$
39
40 DELIMITER ;
```

Functions

Functions são blocos de código SQL que podem ser armazenados no banco de dados e executados quando necessário. De forma geral, modificam dados, retornando valores diretamente.

Prioriza consistência e segurança, na medida em que uma funcionalidade seja utilizada em diferentes lugares, a manutenção é facilitada e sua centralização evita a chance de erros;

Não podem executar comandos como INSERT, UPDATE e DELETE diretamente (mais simples que Procedures)

Functions

Aleatoriedade?

```
DELIMITER $$
```

```
DROP FUNCTION IF EXISTS soma $$
```

```
CREATE FUNCTION soma(a FLOAT, b FLOAT) RETURNS FLOAT  
DETERMINISTIC
```

```
BEGIN
```

```
    RETURN a + b;
```

```
END $$
```

```
DELIMITER ;
```

```
SELECT soma(10.5, 4.25) AS Resultado;
```

```
DROP FUNCTION soma;
```

Tipo do Retorno

Chamando a Function

Exercício

```
DROP DATABASE IF EXISTS empresa_funcionarios;  
CREATE DATABASE empresa_funcionarios;  
USE empresa_funcionarios;
```

```
CREATE TABLE Funcionario (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50),  
    idade INT,  
    departamento VARCHAR(20),  
    email VARCHAR(100)  
);
```

id	nome	idade	departamento
1	Carlos	30	rh
2	Lucia	27	ti
3	Bruno	35	adm

- ☐ Crie uma Function criaEmailFuncionario para criar um email para cada funcionário no estilo: 'nome@departamento.empresa.com';
- ☐ Crie uma Procedure insereFuncionario para fazer as inserções dos funcionários;
- ☐ Use a função CONCAT() para gerar o email, colocando os atributos entre vírgulas.

AULA
CONCLUÍDA!