Universidade Presbiteriana Mackenzie

Banco de Dados – Aula 20

Stored Procedure

Profa. Elisângela Botelho Gracias

Faculdade de Computação e Informática

Roteiro da Apresentação

- Procedure
- CREATE PROCEDURE
- Excluindo uma Procedure

Roteiro da Apresentação

- Procedure
- CREATE PROCEDURE
- Excluindo uma Procedure

Procedure

- Conceito de *Procedure*
  - Uma *stored procedure* é um grupo de comandos SQL e PL/SQL que executam uma determinada tarefa

Procedure

- Ao contrário de um trigger, que é executado automaticamente quando um evento de disparo ocorre, uma procedure precisa ser chamada a partir de um programa ou ser executada manualmente pelo usuário

Roteiro da Apresentação

- Procedure
- CREATE PROCEDURE
- Excluindo uma Procedure

CREATE PROCEDURE

```
CREATE [OR REPLACE] PROCEDURE nome_da_procedure
(Argumento1 modo Tipo_de_Dados,
 Argumento2 modo Tipo_de_Dados,
 ArgumentoN modo Tipo_de_Dados)
IS
   variáveis locais, constantes, ...
BEGIN
   Bloco PL/SQL
END nome_da_procedure;
```

CREATE PROCEDURE

- **OR REPLACE**: recria uma procedure já existente, sem ter que eliminá-la

- **Argumento**: é o nome da variável que será enviada ou retornada do ambiente chamador para a procedure e pode ser passada em um dos 3 modos (IN, OUT, IN OUT). Nunca utilize para um argumento um nome de um atributo de uma tabela que esteja utilizando na stored procedure do banco de dados

CREATE PROCEDURE

- **IN**: especifica que se deve determinar um valor para o argumento quando o procedimento for chamado. Se não for especificado nenhum modo, o IN é o padrão
- **OUT**: especifica que o procedimento devolve um valor para esse argumento quando o procedimento for chamado
- **IN OUT**: especifica que se deve determinar um valor para o argumento quando o procedimento for chamado, e que o procedimento devolve um valor ao seu ambiente de chamada após sua execução

CREATE PROCEDURE

- **Tipo_de_dados**: é o tipo de dado do argumento, que são especificados sem comprimento, precisão ou escala (por exemplo: VARCHAR). O Oracle deriva o comprimento, precisão ou escala de um argumento do ambiente a partir do qual o procedimento é chamado.
  - Uma outra forma de atribuir o tipo de dado a um argumento é **herdar o tipo de dados de um atributo** de uma tabela da seguinte maneira: **nome_tabela.nome_atributo%type**

CREATE PROCEDURE

- **BLOCO PL/SQL**: é o bloco PL/SQL que o Oracle executa

CREATE PROCEDURE

Considere o seguinte Banco de Dados para os exemplos desta aula:

Departamento = {Cod_Depto, Nome_Depto}

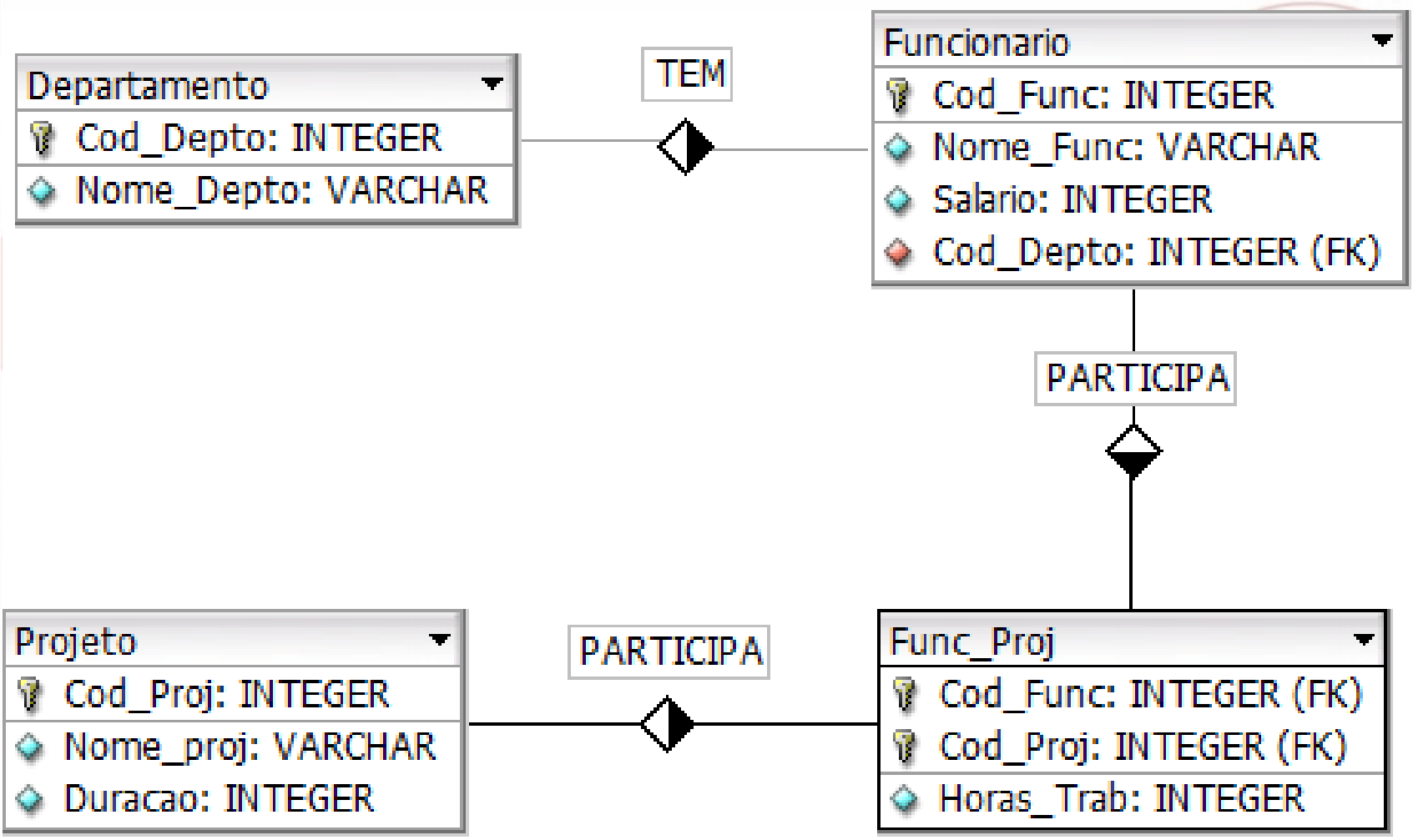
Funcionario = {Cod_Func, Nome_Func, Salario, Cod_Depto}

- *Cod_depto é chave estrangeira que referencia o atributo Cod_depto da tabela Departamento*

Projeto = {Cod_Proj, Nome_Proj, Duracao}

Func_Proj = {Cod_Func, Cod_Proj, Horas_Trab}

- *Cod_Func é chave estrangeira que referencia o atributo Cod_Func da tabela Funcionario*
- *Cod_Proj é chave estrangeira que referencia o atributo Cod_Proj da tabela Projeto*


Departamento
Cod_Depto: INTEGER
Nome_Depto: VARCHAR
TEM
Funcionario
Cod_Func: INTEGER
Nome_Func: VARCHAR
Salario: INTEGER
Cod_Depto: INTEGER (FK)
PARTICIPA
Projeto
Cod_Proj: INTEGER
Nome_proj: VARCHAR
Duracao: INTEGER
PARTICIPA
Func_Proj
Cod_Func: INTEGER (FK)
Cod_Proj: INTEGER (FK)
Horas_Trab: INTEGER

Departamento

Cod_Depto	Nome_Depto
1	Marketing
2	Vendas
3	Dados
4	Pesquisa

Funcionário

Cod_Func	Nome_Func	Salario	Cod_Depto
101	Joao da Silva	2000	2
102	Mario Souza	1500	1
103	Sergio Santos	2400	2
104	Maria Castro	1200	1
105	Marcio Santana	1400	4

Projeto

Cod_Proj	Nome_Proj	Duracao
1001	Sistema A	2
1002	Sistema B	6
1003	Sistema X	4

Func_Proj

Cod_Func	Cod_Proj	Horas_Trab
101	1001	24
101	1002	160
102	1001	56
102	1003	45
103	1001	86
103	1003	64
104	1001	46

```
-- Script de Criação do BD Projeto

DROP TABLE Func_Proj CASCADE CONSTRAINT;
DROP TABLE Projeto CASCADE CONSTRAINT;
DROP TABLE Funcionario CASCADE CONSTRAINT;
DROP TABLE Departamento CASCADE CONSTRAINT;

CREATE TABLE Departamento
(Cod_Depto INTEGER,
 Nome_Depto VARCHAR(20) NOT NULL,
 PRIMARY KEY(Cod_Depto));

CREATE TABLE Funcionario
(Cod_Func INTEGER,
 Nome_Func VARCHAR(20) NOT NULL,
 Salario INTEGER NOT NULL,
 Cod_Depto INTEGER NOT NULL,
 PRIMARY KEY(Cod_Func),
 FOREIGN KEY (Cod_Depto) REFERENCES Departamento (Cod_Depto));

CREATE TABLE Projeto
(Cod_Proj INTEGER,
 Nome_Proj VARCHAR(20) NOT NULL,
 Duracao INTEGER NOT NULL,
 PRIMARY KEY(Cod_Proj));

CREATE TABLE Func_Proj
(Cod_Func INTEGER,
 Cod_Proj INTEGER,
 Horas_Trab INTEGER,
 PRIMARY KEY(Cod_Func, Cod_Proj),
 FOREIGN KEY (Cod_Func) REFERENCES Funcionario(Cod_Func),
 FOREIGN KEY (Cod_Proj) REFERENCES Projeto(Cod_Proj));
```

```
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (1, 'Marketing');
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (2, 'Vendas');
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (3, 'Dados');
INSERT INTO Departamento (Cod_Depto, Nome_Depto) VALUES (4, 'Pesquisa');

INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (101, 'Joao da Silva ', 2000, 2);
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (102, 'Mario Souza', 1500, 1);
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (103, 'Sergio Santos', 2400, 2);
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (104, 'Maria Castro', 1200, 1);
INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto) VALUES (105, 'Marcio Santana', 1400, 4);

INSERT INTO Projeto (Cod_Proj, Nome_Proj, Duracao) VALUES (1001, 'SistemaA', 2);
INSERT INTO Projeto (Cod_Proj, Nome_Proj, Duracao) VALUES (1002, 'SistemaB', 6);
INSERT INTO Projeto (Cod_Proj, Nome_Proj, Duracao) VALUES (1003, 'SistemaX', 4);

INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (101, 1001, 24);
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (101, 1002, 160);
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (102, 1001, 56);
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (102, 1003, 45);
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (103, 1001, 86);
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (103, 1003, 64);
INSERT INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab) VALUES (104, 1001, 46);
COMMIT;
```

CREATE PROCEDURE

- Exemplo1:
  - deseja-se criar uma *procedure* que **aumente o salário de todos os funcionários em 10%**
  - observe que esta ***procedure* não tem nenhum argumento**

CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE Aumento_Salario
IS
BEGIN
        UPDATE Funcionario
        SET Salario = Salario * 1.1;
END Aumento_Salario;
```

CREATE PROCEDURE

- A execução de uma *procedure* é feita por meio de uma chamada ao seu nome
- Pode ser feita dentro de um *trigger* ou de outra *procedure*, bastando especificar o nome da *procedure* com seus argumentos (se tiver)
- A partir do SQL Developer ou do SQL*Plus, uma *procedure* pode ser chamada por meio do comando EXECUTE
- Exemplo: **EXECUTE Aumento_Salario;**

CREATE PROCEDURE

- Observe, agora, se os salários de todos os funcionários foram aumentados em 10%

```
SELECT *

FROM Funcionario;
```

CREATE PROCEDURE

- Se a *procedure* não foi compilada com sucesso (corretamente) ou contém erros, pode-se utilizar o seguinte comando, para que o Oracle aponte o erro: **SHOW ERRORS;**

CREATE PROCEDURE

- Exemplo2 (*procedure* com argumentos):
  - deseja-se criar uma *procedure* que **aumente os salários dos funcionários**
  - só que desta vez os salários serão aumentados somente para os funcionários de um **determinado código de departamento** e com um **determinado percentual**

CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE Aumento_Salario_Depto
(vdepto IN Funcionario.Cod_Depto%type,
 percentual IN INTEGER)
 IS
 BEGIN
        UPDATE Funcionario
        SET Salario = Salario * (1 + percentual/100)
        WHERE (Cod_Depto = vdepto);
 END Aumento_Salario_Depto;
```

CREATE PROCEDURE

- Observe que a *procedure* do Exemplo2 contém dois argumentos:
  - **vdepto** que herdou o tipo de dado do atributo Cod_Depto da tabela Funcionario: **vdepto IN Funcionario.Cod_Depto%type**
  - **percentual** é um argumento cujo tipo do dado é **INTEGER**
  - os dois argumentos possuem o **modo IN**, ou seja, quando essa **procedure for chamada**, **deve passar um valor para cada um dos argumentos**

CREATE PROCEDURE

- Executando a *procedure* criada no exemplo2:
  - **EXECUTE Aumento_Salario_Depto(1,20);**
  - A execução desta *procedure* com os valores de argumentos acima fará o seguinte:
    - aumentará em 20% os salários dos funcionários que são do departamento de código = 1
  - Observe que deve-se obedecer a **ordem em que os argumentos foram criados na *procedure*** na hora de passar os valores

CREATE PROCEDURE

- Observe, novamente, como os salários dos funcionários, cujo cod_depto = 1, foram atualizados:

```
SELECT *

FROM Funcionario

WHERE (Cod_depto = 1);
```

CREATE PROCEDURE

- Exemplo3 (*procedure* com argumentos):
  - deseja-se criar uma *procedure* que **aumente os salários dos funcionários**, com um determinado **percentual**, que são de um determinado departamento (será utilizado o **nome do departamento** e não o código, como no exemplo2)

CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE Aumento_Sal_NomeDep
(vnome IN VARCHAR,
 perc IN INTEGER)
 IS
 BEGIN
    UPDATE Funcionario
    SET Salario = Salario * (1 + perc/100)
    WHERE Cod_Depto IN (SELECT Cod_Depto
                        FROM Departamento
                        WHERE (Nome_Depto = vnome));
END Aumento_Sal_NomeDep;
```

CREATE PROCEDURE

- Executando a *procedure* criada no exemplo3:
  - **EXECUTE** Aumento_Sal_NomeDep **('Vendas', 50);**
  - A execução desta procedure com os valores de argumentos acima fará o seguinte: aumentará em 50% os salários dos funcionários que são do departamento de Vendas
  - Observe que deve-se obedecer a **ordem em que os argumentos foram criados na procedure** na hora de passar os valores

CREATE PROCEDURE

- Observe, novamente, como os salários dos funcionários do departamento de 'Vendas' receberam um aumento de 50%:

```
SELECT F.Nome_Func, F.Salario, D.Nome_Depto
FROM Funcionario F INNER JOIN Departamento D
ON (F.Cod_Depto = D.Cod_Depto)
WHERE (D.Nome_Depto = 'Vendas');
```

CREATE PROCEDURE

- Uma **outra forma de fazer o exemplo3** seria:
  - Criando uma variável vdepto para receber o código do departamento com um determinado nome;
  - Com o valor desta variável vdepto, não seria necessário fazer um UPDATE com sub-select, pois já teria o código do departamento, de acordo com o nome de departamento passado

```
CREATE OR REPLACE PROCEDURE Aumento_Sal_NomeDep
(vnome IN VARCHAR,
 perc IN INTEGER)
 IS
        vdepto INTEGER;
 BEGIN
        SELECT Cod_Depto INTO vdepto
        FROM Departamento
        WHERE (Nome_Depto = vnome);

        UPDATE Funcionario
        SET Salario = Salario * (1 + perc/100)
        WHERE (Cod_Depto  = vdepto);

END Aumento_Sal_NomeDep;
```

```
CREATE OR REPLACE PROCEDURE Aumento_Sal_NomeDep
(vnome IN VARCHAR,
 perc IN INTEGER)
 IS
        vdepto INTEGER;
 BEGIN
        SELECT Cod_Depto INTO vdepto
        FROM Departamento
        WHERE (Nome_Depto = vnome);

        UPDATE Funcionario
        SET Salario = Salario * (1 + perc/100)
        WHERE (Cod_Depto  = vdepto);

END Aumento_Sal_NomeDep;
```

Argumentos (ou parâmetros) da *procedure*

Variável local

```
CREATE OR REPLACE PROCEDURE Aumento_Sal_NomeDep
(vnome IN VARCHAR,
 perc IN INTEGER)
 IS
 vdepto INTEGER;
 BEGIN
    SELECT Cod_Depto INTO vdepto
    FROM Departamento
    WHERE (Nome_Depto = vnome);

    UPDATE Funcionario
    SET Salario = Salario * (1 + perc/100)
    WHERE (Cod_Depto = vdepto);

END Aumento_Sal_NomeDep;
```

Esse SELECT retorna o código do departamento, de acordo com o parâmetro vnome, e armazena-o em vdepto

```
CREATE OR REPLACE PROCEDURE Aumento_Sal_NomeDep
(vnome IN VARCHAR,
 perc IN INTEGER)
 IS
 vdepto INTEGER;
 BEGIN
    SELECT Cod_Depto
    FROM Departa
    WHERE (N  e_De

    UPDATE Funcionario
    SET Salario = Salario * (1 + perc/100)
    WHERE (Cod_Depto  = vdepto);

END Aumento_Sal_NomeDep;
```

O UPDATE é realizado utilizando o vdepto, que recebeu o código do departamento no SELECT.

Roteiro da Apresentação

- Procedure
- CREATE PROCEDURE
- Excluindo uma Procedure

Excluindo uma Procedure

- Para excluir uma *procedure* deve ser usado o comando:

```
DROP PROCEDURE nome_da_procedure;
```

- Exemplo:

```
DROP PROCEDURE Aumenta_Salario;
```

Exercício Resolvido

EXERCÍCIO RESOLVIDO

- Crie uma *procedure* que insira um funcionário em um projeto já existente, considerando que serão passados os seguintes argumentos:
  - **nome do funcionário** e do **projeto** (já existentes no banco de dados), e, também, o **número de horas** que ele trabalhou neste projeto

```
CREATE OR REPLACE PROCEDURE Insere_Func_Proj
(nfunc IN VARCHAR, nproj IN VARCHAR,  horas IN INTEGER)
IS
     vcodf INTEGER;
     vcodp INTEGER;
BEGIN
          SELECT Cod_Func INTO vcodf
          FROM Funcionario
          WHERE (Nome_Func = nfunc);

          SELECT Cod_Proj INTO vcodp
          FROM Projeto
          WHERE (Nome_Proj = nproj );

          INSERT
          INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab)
          VALUES (vcodf, vcodp, horas) ;
END Insere_Func_Proj;
```

```
CREATE OR REPLACE PROCEDURE Insere_Func_Proj
(nfunc IN VARCHAR, nproj IN VARCHAR,  horas IN INTEGER)
IS
    vcodf INTEGER;
    vcodp INTEGER;
BEGIN
        SELECT Cod_Func INTO vcodf
        FROM Funcionario
        WHERE (Nome_Func = nfunc);

        SELECT Cod_Proj INTO vcodp
        FROM Projeto
        WHERE (Nome_Proj = nproj );

        INSERT
        INTO Func_Proj (Cod_Func, Cod_Proj, Horas_Trab)
        VALUES (vcodf, vcodp, horas) ;
END Insere_Func_Proj;
```

EXERCÍCIO RESOLVIDO

Execute a *procedure* para ver se está funcionando:

EXECUTE Insere_Func_Proj **(**'Marcio Santana', 'SistemaA', 56**);**

- Depois da execução, verifique se uma nova linha foi inserida na tabela Func_Proj, com os valores dos argumentos acima.

Exercícios

EXERCÍCIO 1

1) Crie uma *stored procedure* que **elimine um determinado departamento da empresa.** Mas os **funcionários daquele departamento deverão ser transferidos para outro departamento**.

Serão passados, como **argumentos**:

- o nome do departamento a ser eliminado
- o nome do departamento que os funcionários deverão ser transferidos.

Depois da *procedure* criada, utilize-a para eliminar o departamento de 'Marketing' e transferir os funcionários deste departamento para 'Vendas'.

EXERCÍCIO 2

2) Crie uma *stored procedure* que **insira um novo funcionário na tabela funcionário**, considerando que serão passados os seguintes **argumentos**:

- nome do funcionário;
- salário do funcionário;
- nome do departamento que o funcionário irá pertencer (considere que esse departamento já exista no banco de dados).

Obs: o código do funcionário NÃO será passado como argumento.

Depois da *procedure* criada, utilize-a para inserir você no departamento de 'Vendas'.

EXERCÍCIO1 - RESOLUÇÃO

```
CREATE OR REPLACE PROCEDURE Elimina_Depto
(excluir IN VARCHAR,  transferir IN VARCHAR)
IS
    vdepto_excluir INTEGER;
    vdepto_transferir INTEGER;
BEGIN
    SELECT Cod_Depto INTO vdepto_excluir
    FROM Departamento
    WHERE (Nome_Depto = excluir);

    SELECT Cod_Depto INTO vdepto_transferir
    FROM Departamento
    WHERE (Nome_Depto = transferir);

    UPDATE Funcionario
    SET Cod_Depto = vdepto_transferir
    WHERE (Cod_Depto = vdepto_excluir);

    DELETE
    FROM Departamento
    WHERE (Cod_Depto = vdepto_excluir);

END Elimina_Depto;
```

EXERCÍCIO2 - RESOLUÇÃO

Mackenzie

```
CREATE OR REPLACE PROCEDURE Insere_Funcionario
(vnome IN VARCHAR,
 vsalario IN INTEGER,
 vdepto IN VARCHAR)
IS
    vcodfunc INTEGER;
    vcoddepto INTEGER;

BEGIN
    SELECT MAX(Cod_Func) + 1 INTO vcodfunc
    FROM Funcionario;

    SELECT Cod_Depto INTO vcoddepto
    FROM Departamento
    WHERE (Nome_Depto = vdepto);

    INSERT INTO Funcionario (Cod_Func, Nome_Func, Salario, Cod_Depto)
    VALUES (vcodfunc, vnome, vsalario, vcoddepto);

END Insere_Funcionario;
```

Bibliografia

- FANDERUFF, D. *Dominando o Oracle 9i: Modelagem e Desenvolvimento*. São Paulo: Pearson Education do Brasil, 2003.
- RAMALHO, J. A. *Oracle 9i*. São Paulo: Berkeley Brasil, 2002

Obrigado

Profa. Elisângela Botelho Gracias
elisangela.botelho@mackenzie.br