



Laboratório de Bases de Dados

Prof. José Fernando Rodrigues Júnior

Aula 7 – Procedimentos e Funções

Material: Profa. Elaine Parros Machado de Sousa



Procedimentos e Funções

- Subprogramas PL/SQL
 - armazenados no SGBD
 - locais
 - em código PL/SQL anônimo
 - em subprogramas armazenados



Procedimentos e Funções Armazenados

- ***Stored Procedures/Functions***
 - armazenados no SGBD
 - código fonte
 - código compilado (bytecode-like ou nativo)
- Podem ser executados dentro de outros subprogramas ou blocos PL/SQL
- *Procedure*: chamado como instrução PL/SQL
- *Function*: chamada como parte de uma expressão

Procedimentos armazenados

```
CREATE OR REPLACE PROCEDURE insere_matricula (  
    p_disciplina Matricula.Sigla%TYPE,  
    p_turma Matricula.Numero%TYPE,  
    p_aluno Matricula.Aluno%TYPE ) AS /*pode ser IS*/
```

```
v_count NUMBER;  
e_lotada EXCEPTION;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO v_count FROM matricula M  
        WHERE M.sigla = p_disciplina and M.numero = p_turma and  
            M.ano = EXTRACT (YEAR FROM SYSDATE);
```

```
    IF v_count < 70 THEN
```

```
        insert into matricula values (p_disciplina, p_turma,  
                                       p_aluno, EXTRACT (YEAR FROM SYSDATE),  
                                       NULL);
```

```
    ELSE RAISE e_lotada;
```

```
    END IF;
```

```
EXCEPTION
```

```
    WHEN e_lotada
```

```
        THEN ...../*NUNCA imprimir msg de erro dentro do proc.*/
```

```
END insere_matricula; /*boa prática de programação*/
```

```
/*controle retorna para o bloco que chamou o proc.*/
```

Procedimentos armazenados

```
/*Programa Principal - PL/SQL anônimo*/
```

```
DECLARE
```

```
    v_disciplina Matricula.Sigla%TYPE;
```

```
    v_turma Matricula.Numero%TYPE;
```

```
    v_aluno Matricula.Aluno%TYPE;
```

```
BEGIN
```

```
    v_disciplina := 'SCC241';
```

```
    v_turma := 1;
```

```
    v_aluno := 222;
```

```
/*Parâmetros: notação posicional*/
```

```
    insere_matricula(v_disciplina, v_turma, v_aluno);
```

```
END;
```

Procedimentos armazenados

```
/*Programa Principal - PL/SQL anônimo*/
```

```
DECLARE
```

```
    v_disciplina Matricula.Sigla%TYPE;
```

```
    v_turma Matricula.Numero%TYPE;
```

```
    v_aluno Matricula.Aluno%TYPE;
```

```
BEGIN
```

```
    v_disciplina := 'SCC241';
```

```
    v_turma := 1;
```

```
    v_aluno := 222;
```

```
/*Parâmetros: notação identificada*/
```

```
insere_matricula(p_aluno => v_aluno,  
                p_disciplina => v_disciplina,  
                p_turma => v_turma);
```

```
END;
```

Funções armazenadas

```
CREATE OR REPLACE FUNCTION media (  
    p_aluno Matricula.Aluno%TYPE )  
    RETURN NUMBER IS /*pode ser AS*/  
  
    v_media NUMBER;  
  
BEGIN  
    SELECT AVG(nota) INTO v_media FROM MATRICULA  
        WHERE aluno = p_aluno;  
  
RETURN v_media; /* RETURN obrigatório para sair da função*/  
  
END media;  
  
/*E se não existir o aluno?*/
```

Funções armazenadas

```
/*Programa Principal - PL/SQL anônimo*/
```

```
DECLARE
```

```
    v_media NUMBER;
```

```
    v_aluno Matricula.Aluno%TYPE;
```

```
BEGIN
```

```
    v_aluno := 222;
```

```
    v_media := media(v_aluno);
```

```
    dbms_output.put_line('Média de ' || v_aluno || '  
= ' || v_media);
```

```
END;
```


Exemplo: parâmetros com valor *default*

```
CREATE OR REPLACE PROCEDURE insere_matricula_2 (  
    p_disciplina Matricula.Sigla%TYPE,  
    p_turma Matricula.Numero%TYPE,  
    p_aluno Matricula.Aluno%TYPE,  
    p_nota Matricula.Nota%TYPE DEFAULT 0.0 ) IS  
    /*boa prática: parâmetros default sempre no final da lista*/  
    v_count NUMBER;  
    e_lotada EXCEPTION;  
  
BEGIN  
    SELECT COUNT(*) INTO v_count FROM matricula M  
        WHERE M.sigla = p_disciplina and M.numero = p_turma and  
            M.ano = EXTRACT (YEAR FROM SYSDATE);  
    IF v_count < 70 THEN  
        insert into matricula values (p_disciplina, p_turma,  
            p_aluno, EXTRACT (YEAR FROM SYSDATE),  
            p_nota);  
    ELSE RAISE e_lotada;  
    END IF;  
EXCEPTION  
    WHEN e_lotada  
        THEN .....;  
END insere_matricula 2;
```

Exemplo: parâmetros com valor *default*

```
/*Programa Principal - PL/SQL anônimo*/
```

```
DECLARE
```

```
    v_disciplina Matricula.Sigla%TYPE;
```

```
    v_turma Matricula.Numero%TYPE;
```

```
    v_aluno Matricula.Aluno%TYPE;
```

```
BEGIN
```

```
    v_disciplina := 'SCC241';
```

```
    v_turma := 1;
```

```
    v_aluno := 222;
```

```
/*Parâmetros: notação posicional - parâmetro default, no final da lista, é  
omitido*/
```

```
insere_matricula_2(v_disciplina, v_turma, v_aluno);
```

```
END;
```

Exemplo: parâmetros com valor *default*

```
/*Programa Principal - PL/SQL anônimo*/
```

```
DECLARE
```

```
    v_disciplina Matricula.Sigla%TYPE;
```

```
    v_turma Matricula.Numero%TYPE;
```

```
    v_aluno Matricula.Aluno%TYPE;
```

```
BEGIN
```

```
    v_disciplina := 'SCC241';
```

```
    v_turma := 1;
```

```
    v_aluno := 222;
```

```
/*Parâmetros: notação identificada - qualquer ordem*/
```

```
insere_matricula_2(p_aluno => v_aluno,
```

```
    p_disciplina => v_disciplina,
```

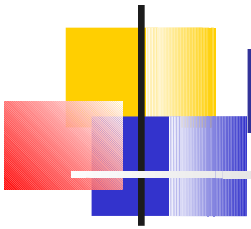
```
    p_turma => v_turma);
```

```
END;
```



Procedures e Functions

- Parâmetros
 - Formais - parâmetros na declaração
 - ex: `p_aluno`
 - Reais – valores passados como argumentos
 - ex: `v_aluno`
- Trata-se apenas de terminologia PL/SQL para conceitos já bem conhecidos



Procedures e Functions

■ Exemplo:

```
PROCEDURE raise_salary (emp_id INTEGER, amount REAL) IS
BEGIN
    UPDATE emp SET sal = sal + amount WHERE empno = emp_id;
END raise_salary;
```

```
raise_salary(emp_num, amount);
raise_salary(emp_num, merit + cola);
raise_salary(emp_num, '2500'); /*Cast*/
```



Procedures e Functions

■ Exemplo:

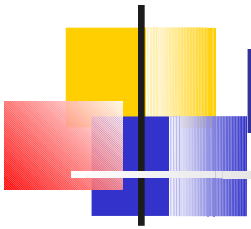
Parâmetros Formais

```
PROCEDURE raise_salary (emp_id INTEGER, amount REAL) IS  
BEGIN
```

```
    UPDATE emp SET sal = sal + amount WHERE empno = emp_id;  
END raise_salary;
```

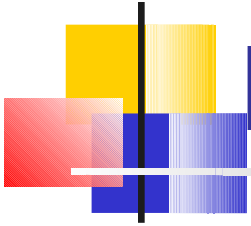
```
raise_salary(emp_num, amount);  
raise_salary(emp_num, merit + bonus);  
raise_salary(emp_num, '2500'); /*Cast*/
```

Parâmetros Reais



Procedures e Functions

- **Parâmetros - Modos**
 - **IN** (padrão)
 - Passagem de parâmetros apenas para dentro do procedimento/função
 - **OUT**
 - Passagem de parâmetros apenas para fora do procedimento/função
 - **IN OUT**
 - Passagem de parâmetros para dentro e para fora do procedimento/função



Procedures e Functions

```
Procedure CalculaNotaFinal(nota1 IN NUMBER, nota2 IN  
    NUMBER, notafinal OUT NUMBER)
```

```
BEGIN
```

```
    notafinal := (nota1 + nota2) / 2;
```

```
END;
```

```
//-----
```

```
DECLARE
```

```
    nota NUMBER;
```

```
BEGIN
```

```
    CalculaNotaFinal(3.4, 7.8, nota);
```

```
    dbms_output.put_line(nota);
```

```
END;
```




Procedures e Functions

- Parâmetros - Modos

- **IN** (padrão)

- parâmetro formal: atua como uma constante
 - parâmetro real: constante, variável inicializada, literal ou expressão

- **OUT**

- parâmetro formal: atua como uma variável não inicializada (**tem valor NULL dentro do proc/func., mesmo que o parâmetro real tenha sido inicializado**)
 - parâmetro real: variável

- **IN OUT**

- parâmetro formal: atua como uma variável inicializada
 - parâmetro real: variável



Procedures e Functions

```
Procedure CalculaNotaFinal(nota1 IN NUMBER, nota2 IN NUMBER,  
    notafinal OUT NUMBER)
```

```
BEGIN
```

```
    nota1 := 5.0;    /*Erro*/
```

```
    notafinal := (nota1 + nota2) / 2;
```

```
END;
```

```
//-----
```

```
DECLARE
```

```
    nota NUMBER;
```

```
BEGIN
```

```
    nota := 10.00;    /*Não tem efeito*/
```

```
    CalculaNotaFinal(3.4, 7.8, nota);
```

```
    dbms_output.put_line(nota);
```

```
END;
```

Exemplo

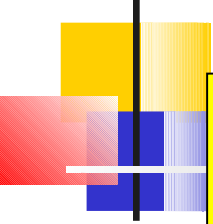
```
CREATE OR REPLACE PROCEDURE media (  
    p_aluno IN Matricula.Aluno%TYPE, /*parâmetros formais*/  
    p_media OUT NUMBER) IS  
BEGIN  
    SELECT AVG(nota) INTO p_media FROM MATRICULA  
        WHERE aluno = p_aluno;  
  
END media;
```

```
/*Programa Principal - PL/SQL anônimo*/  
DECLARE  
    v_media NUMBER(2,1);  
    v_aluno Matricula.Aluno%TYPE;  
BEGIN  
    v_aluno := 222;  
    media(v_aluno, v_media); /*parâmetros reais*/  
    dbms_output.put_line('Média de ' || v_aluno || ' = ' ||  
        v_media);  
END;
```



Procedures e Functions

- Passagem de parâmetros
 - por valor
 - padrão para **OUT** e **IN OUT**
 - parâmetros reais não são perdidos se uma exceção não tratada for levantada no procedimento ou na função
 - por referência
 - padrão para **IN**
 - **NOCOPY**
 - tenta passar parâmetro **OUT** e **IN OUT** por referência
 - **dica** de compilador, ou seja, nem sempre é executado
 - problema com exceções não tratadas



Procedures and Functions

O PL/SQL contraria a intuição de que parâmetros IN seriam passados por valor, e parâmetros OUT seriam passados por referência.

IN são sempre por referência.

OUT são por valor, copiados para o parâmetro formal na chamada, e copiados de volta para o parâmetro real ao final do procedimento.

OUT NOCOPY, usam-se valores por referência sem operações de cópia.



Outros comandos

- Por alguma razão o SQLDeveloper não exibe os erros de compilação de funções e procedimentos
- Usar:

/

```
select * from user_errors where  
type = 'PROCEDURE' AND      -- ou FUNCTION  
name = 'NOME_PROC_OU_FUNCAO_EM_MAIUSCULAS';
```



Outros comandos

- Pesquisa:
 - `alter/drop procedure`
 - `alter/drop function`
- Ex:

```
ALTER PROCEDURE insere_matricula  
    COMPILE;
```

```
/*Por que recompilar um procedimento/função armazenado?*/
```



PL/SQL

- Manual de consulta:

PL/SQL

User's Guide and Reference



PL/SQL – Exemplo de aula 1

```
SET SERVEROUTPUT ON;
```

```
/
```

```
CREATE OR REPLACE PROCEDURE consulta_universal(p_table VARCHAR2) IS
```

```
    sql_text VARCHAR2(300);
```

```
    total NUMBER;
```

```
BEGIN
```

```
    sql_text := 'SELECT COUNT(*) FROM '||p_table;
```

```
    EXECUTE IMMEDIATE sql_text INTO total;
```

```
    dbms_output.put_line('Total: '||total);
```

```
END;
```

```
/
```

```
select * from user_errors where type = 'PROCEDURE' AND name = 'CONSULTA_UNIVERSAL';
```

```
/
```

```
DECLARE
```

```
BEGIN
```

```
    consulta_universal('LBD01_VINCULO_USP');
```

```
END;
```



PL/SQL – Exemplo de aula 2

```
CREATE OR REPLACE PROCEDURE consulta_universal(p_table VARCHAR2, p_nome VARCHAR2 DEFAULT NULL,
                                              p_valor VARCHAR2 DEFAULT NULL) IS
    sql_text VARCHAR2(300);
    total NUMBER;
BEGIN
    sql_text := 'SELECT COUNT(*) FROM '||p_table;

    IF(p_nome IS NOT NULL) AND (p_valor IS NOT NULL) THEN
        sql_text := sql_text || ' WHERE ' || p_nome || ' = ' || p_valor;
    END IF;

    EXECUTE IMMEDIATE sql_text INTO total;

    dbms_output.put_line('Total: '||total);

/

select * from user_errors where type = 'PROCEDURE' AND name = 'CONSULTA_UNIVERSAL';

/

DECLARE
BEGIN
    consulta_universal('LBD01_VINCULO_USP');
    consulta_universal('LBD03_ALUNO', 'IDADE','18');
    consulta_universal('LBD11_GRUPO', 'CODCURSO','1');
END;
```



PL/SQL – Exemplo de aula 3

```
CREATE OR REPLACE FUNCTION consulta_universal_f(p_table VARCHAR2, p_nome VARCHAR2 DEFAULT NULL,
                                                p_valor VARCHAR2 DEFAULT NULL)

RETURN NUMBER IS
    sql_text VARCHAR2(300);
    total NUMBER;
BEGIN
    sql_text := 'SELECT COUNT(*) FROM '||p_table;

    IF(p_nome IS NOT NULL) AND (p_valor IS NOT NULL) THEN
        sql_text := sql_text || ' WHERE ' || p_nome || ' = ' || p_valor;
    END IF;

    EXECUTE IMMEDIATE sql_text INTO total;

    RETURN total;
END;

/

select * from user_errors where type = 'PROCEDURE' AND name = 'CONSULTA_UNIVERSAL';

/

DECLARE
BEGIN
    dbms_output.put_line('Total: '||consulta_universal_f('LBD01_VINCULO_USP'));

    dbms_output.put_line('Total: '||consulta_universal_f('LBD03_ALUNO', 'IDADE', '18'));

    dbms_output.put_line('Total: '||consulta_universal_f('LBD11_GRUPO', 'CODCURSO', '1'));

END;
```



PL/SQL – Exemplo de aula 4

```
CREATE OR REPLACE PROCEDURE insere_lectona(p_prof LBD06_LECIONA.NROUSPPROF%TYPE, p_cod LBD06_LECIONA.CODDISC%TYPE) IS
BEGIN
    INSERT INTO lbd06_lectona VALUES(p_prof, p_cod);
EXCEPTION
    WHEN OTHERS THEN
        CASE
            WHEN (SQLERRM LIKE '%PK_LECIONA%') THEN
                raise_application_error(-20001,'O relacionamento já existe.');
```