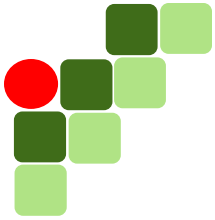
Banco de Dados II

Cristiane Yaguinuma
cristiane.yaguinuma@ifsp.edu.br

- Procedimentos e Funções PL/SQL

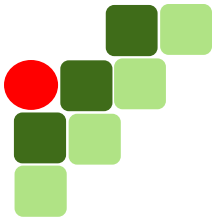


Roteiro da aula

- Comparação entre unidades de programa PL/SQL
- Procedimentos armazenados
- Funções
- Exercícios

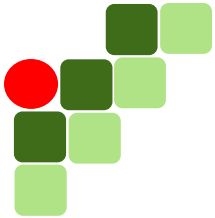
Tipos de unidades de programa em PL/SQL

Anônimo	Bloco executável de comandos PL/SQL que não possui um nome explícito.
Trigger	Rotina disparada automaticamente antes ou depois de comandos update, insert ou delete.
Procedure	Pode receber parâmetros de entrada ou de saída. Ativado como se fosse um comando da linguagem.
Function	Pode receber parâmetros apenas de entrada e, necessariamente, retorna um valor em seu nome. A ativação ocorre em expressões.



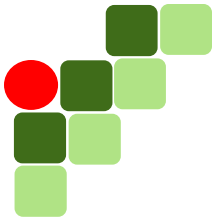
Procedures e functions

- Compilados e armazenados no Banco de Dados
- Possuem estrutura similar à do bloco anônimo
  - Declaração da variáveis opcional
    - Não inicia com a palavra-chave DECLARE
  - Seção executável obrigatória (entre BEGIN e END)
    - Contém a implementação da lógica do programa
  - Seção de exceção opcional
    - Pode incluir tratamento de exceções
- Podem ser ativados por outros blocos PL/SQL



Blocos Anônimos x Subprogramas

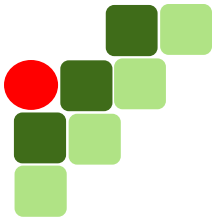
Blocos Anônimos	Subprogramas
Blocos PL/SQL não nomeados	Blocos PL/SQL nomeados
Compilados a cada execução	Compilados uma única vez
Não armazenado no BD	Armazenado no BD
Não pode ser invocado por outras aplicações	Pode ser invocado por outras aplicações
Não retorna valores	Pode retornar valores
Não recebe parâmetros	Pode receber parâmetros



Procedure

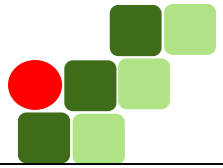
- Subprogramas que realizam tarefas específicas e são ativados como comandos

- Podem receber parâmetros de:
  - Entrada (IN)
  - Saída (OUT)
  - Entrada e Saída (IN OUT)



CREATE PROCEDURE

```
CREATE [OR REPLACE] PROCEDURE nome_procedure
 [(argumento1 [modo1] tipodedado1,
 argumento2 [modo2] tipodedado2,
 . . .)]
IS
    declaração de variáveis locais
BEGIN
    comandos PL/SQL
END;
/
```



CREATE PROCEDURE

```
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE add_dept
IS
  dept_id departments.department_id%TYPE;
  dept_name departments.department_name%TYPE;
BEGIN
  SELECT MAX(department_id) INTO dept_id
     FROM departments;
  dept_name:='Novo Departamento';
  INSERT INTO departments
     (department_id,department_name)
     VALUES((dept_id + 1),dept_name);
  DBMS_OUTPUT.PUT_LINE('Inserida '|| SQL%ROWCOUNT ||'
linha.');
END;
/
```

Para verificar quantas linhas foram afetadas pelo último comando DML executado

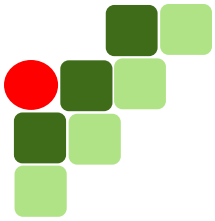
Executando procedure

- Dentro de algum bloco de programa (anônimo, procedure, function, trigger)

```
BEGIN
    add_dept;
END;
/
```

```
-- funciona no sql developer
EXEC add_dept;
```

```
-- para verificar depto adicionado
SELECT * FROM DEPARTMENTS;
```

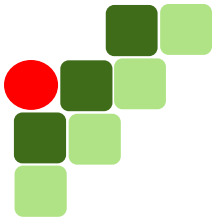


Procedure com parâmetros

```
CREATE OR REPLACE PROCEDURE add_dept_manager (
  dept_name departments.department_name%TYPE,
  emp_id employees.employee_id%TYPE)
IS
BEGIN
      UPDATE DEPARTMENTS
      SET manager_id = emp_id
      WHERE UPPER(department_name) = UPPER(dept_name);

      DBMS_OUTPUT.PUT_LINE('Atualizada(s) '||
SQL%ROWCOUNT ||' tupla(s)');

END;
/
```

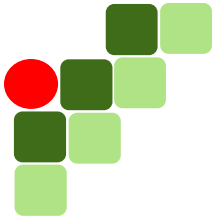


Executando procedure

- Dentro de algum bloco de programa (anônimo, procedure, function, trigger)

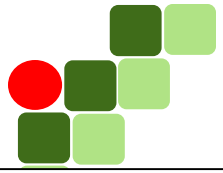
```
BEGIN
  add_dept_manager ('IT Support', 100);
END;
/

-- para verificar
SELECT * FROM DEPARTMENTS;
```



Modos dos parâmetros

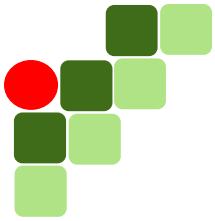
IN	OUT	IN OUT
Modo padrão	Deve ser especificado	Deve ser especificado
Passa valores de entrada para o subprograma	Retorna valor de saída para quem invocou o subprograma	Passa um valor inicial para o subprograma e retorna um valor atualizado para quem invocou o subprograma
Atua como um valor constante – o subprograma não pode modificá-lo	O valor inicial padrão é NULL. O subprograma tem que atribuir um valor de retorno.	O parâmetro pode ser lido e modificado pelo subprograma
Valor do parâmetro é passado por referência	Por padrão, parâmetro passado por valor (cópia do valor)	Por padrão, parâmetro passado por valor (cópia do valor)



Procedure com parâmetros

```
CREATE OR REPLACE PROCEDURE p (
  a       INTEGER,
  b    IN INTEGER,
  c   OUT INTEGER,
  d IN OUT FLOAT
) IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Dentro do procedimento p:');
  DBMS_OUTPUT.PUT_LINE('IN a = ' || a);
  DBMS_OUTPUT.PUT_LINE('IN b = ' || b);
  DBMS_OUTPUT.PUT_LINE('OUT c = ' || c);
  DBMS_OUTPUT.PUT_LINE('IN OUT d = ' || d);
  c := a+b;
  d := a-b;
END;
/
```

```
DECLARE
  x INTEGER := 10;
  y INTEGER := 2;
  z INTEGER;
  w FLOAT := 5;
BEGIN
  DBMS_OUTPUT.PUT_LINE('ANTES do procedimento p:');
  DBMS_OUTPUT.PUT_LINE('x = ' || x);
  DBMS_OUTPUT.PUT_LINE('y = ' || y);
  DBMS_OUTPUT.PUT_LINE('z = ' || z);
  DBMS_OUTPUT.PUT_LINE('w = ' || w);
  p(x,y,z,w);
  DBMS_OUTPUT.PUT_LINE('DEPOIS do procedimento p:');
  DBMS_OUTPUT.PUT_LINE('x = ' || x);
  DBMS_OUTPUT.PUT_LINE('y = ' || y);
  DBMS_OUTPUT.PUT_LINE('z = ' || z);
  DBMS_OUTPUT.PUT_LINE('w = ' || w);
END;
/
```

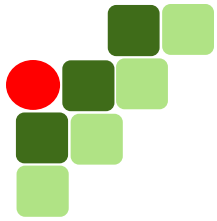


Invocando Procedure em aplicações externas

Um procedimento armazenado pode ser invocado de qualquer aplicação, tal como uma aplicação Java

Pode-se invocar um procedimento armazenado em aplicações externas com o comando:

```
CALL <nome_procedure>
```



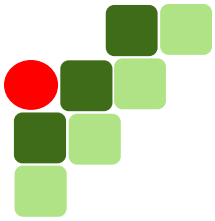
Invocando Procedure em aplicações externas

PL/SQL:

```
CREATE PROCEDURE RAISE_PRICE(
      IN coffeeName varchar(32),
      IN maximumPercentage float,
      INOUT newPrice float) ...
```

JAVA:

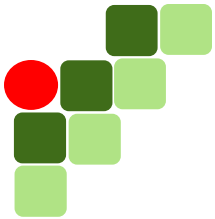
```
CallableStatement cs  =
      con.prepareCall("{call RAISE_PRICE(?,?,?)}");
cs.setString(1, coffeeNameArg);
cs.setFloat(2, maximumPercentageArg);
cs.registerOutParameter(3, Types.NUMERIC);
cs.setFloat(3, newPriceArg);
cs.execute();
float newPriceArg = cs.getFloat(3);
```



Exercício

- Defina um procedimento que informe na tela quais os empregados que podem receber aumento, dependendo do id do gerente. Imprima na tela nome e salário dos empregados que podem receber aumento. Podem receber aumento os empregados cujo salário for menor do que 5000 e que tenham manager_id igual a 101 ou 124.

- Torne o procedimento genérico, passando o id do gerente e o limite superior de salário como parâmetros. Use cursor com parâmetros.

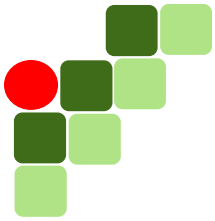


Function

- Pode receber apenas parâmetros de entrada e retornam um valor

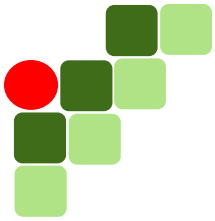
```
CREATE [OR REPLACE] FUNCTION nome_function
[(lista de parametros)]
RETURN tipo_de_retorno
IS
    declaração de variáveis locais
BEGIN
    comandos PL/SQL
END;
```

```
CREATE OR REPLACE FUNCTION check_sal(
   empno employees.employee_id%TYPE)
RETURN BOOLEAN IS
   dept_id employees.department_id%TYPE;
   sal employees.salary%TYPE;
   avg_sal employees.salary%TYPE;
BEGIN
  SELECT salary,department_id INTO sal, dept_id
  FROM employees WHERE employee_id = empno;
  SELECT AVG(salary) INTO avg_sal
  FROM employees WHERE department_id = dept_id;
  IF sal > avg_sal THEN
     RETURN TRUE;
  END IF;
  RETURN FALSE;
END;
/
```



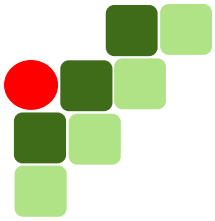
Executando function

```
SET SERVEROUTPUT ON
BEGIN
IF (check_sal(100)) THEN
      DBMS_OUTPUT.PUT_LINE('Salário > média');
ELSE
      DBMS_OUTPUT.PUT_LINE('Salário < média');
END IF;
END;
/
```



CREATE FUNCTION

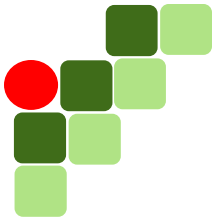
```
CREATE OR REPLACE FUNCTION qtd_salario_maior
(empno employees.employee_id%TYPE)
RETURN NUMBER
IS
      qtd NUMBER;
BEGIN
      SELECT COUNT(*) INTO qtd FROM employees
         WHERE salary >
                  (SELECT salary FROM employees
                   WHERE employee_id = empno);
      RETURN qtd;
END;
/
```



Executando function

```
DECLARE
  empno Employees.employee_id%TYPE := &empno;
BEGIN
DBMS_OUTPUT.PUT_LINE(qtd_salario_maior(empno) ||
' empregados ganham mais que o empregado ' ||
empno);
END;
/
```

```
SELECT employee_id, qtd_salario_maior(employee_id) qtd
FROM Employees
ORDER BY qtd DESC;
```



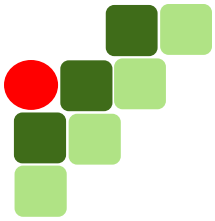
Invocando Function em aplicações externas

PL/SQL:

```
FUNCTION balance (acct_id NUMBER)
RETURN NUMBER
IS ...
```

JAVA:

```
CallableStatement cstmt =
   conn.prepareCall("{? = CALL balance(?)}");
cstmt.registerOutParameter(1, Types.FLOAT);
cstmt.setInt(2, acctNo);
cstmt.execute();
float acctBal = cstmt.getFloat(1);
```



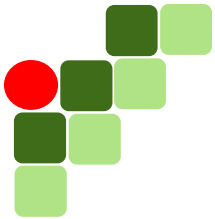
Procedure e function

- Verificando procedimentos criados:

```
SELECT object_name FROM user_objects
WHERE object_type='PROCEDURE';
```

- Verificando funções criadas:

```
SELECT object_name FROM user_objects
WHERE object_type='FUNCTION';
```

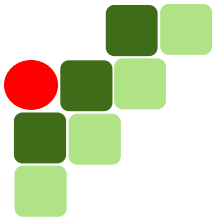


Excluindo Procedure e function

Para excluir uma procedure ou function de um banco de dados, use os comandos:

```
DROP PROCEDURE nome_procedure;

DROP FUNCTION nome_funcao;
```



Exercícios

- Faça uma função que, dado o id do empregado, retorne quantos empregados são mais antigos que ele na empresa.

- Faça uma função que, dado o id do empregado, retorne a quantidade de anos completos que ele trabalha na empresa.
  - Sugestão: usar a função

```
MONTHS_BETWEEN(SYSDATE, hire_date)/12;
```