

BASE DE DADOS



Oracle
PL/SQL –Procedimentos,
Funções e
Triggers

Teóricas
Ano Letivo 2018/2019
Rosa Reis

PLSQL

Introdução

- * Procedimentos e Funções são blocos PL/SQL armazenados de forma compilada, e executados no servidor
- * A estrutura de blocos dos subprogramas é semelhante à estrutura de blocos anônimos.
- * Posteriormente o subprogramas tornam-se os blocos de construção de pacotes e triggers.
- * Procedimentos e funções têm bastantes benefícios devido à modularização do código:
 - Fácil Manutenção: as modificações só precisam ser feitas uma vez para melhorar os vários aplicativos e minimizar os testes.
 - Reutilização de código: os subprogramas estão localizados num só lugar.
 - > Quando compilados e validados, eles podem ser usados e reutilizados em qualquer número de aplicações.

PLSQL

Introdução

- * Características de um subprograma
 - Tem um nome e pode ter parâmetros
 - Modo dos parâmetros
 - ✧ **IN** (*de entrada*; passagem por valor)
 - ✓ Passa um valor que não pode ser alterado pelo subprograma
 - ✧ **OUT** (*de saída*; passagem por referência)
 - ✓ Valor de retorno. Deve ser inicializado no subprograma
 - ✧ **IN OUT** (*de entrada/saída*; passagem por referência)
 - ✓ Passa um valor e retorna o valor atualizado pelo subprograma

PLSQL

Procedures

- * São subprogramas que têm por objetivo executar uma determinada ação
 - Não permitem retornar valores para o seu *caller*, exceto se especificarmos parâmetros de output
- * Estrutura básica de criação de uma procedure.

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [ (parameter [,parameter]) ]

[IS | AS]
  [declaration_section]

BEGIN
  executable_section

[EXCEPTION
  exception_section]

END [procedure_name];
```

PLSQL

Procedures

```
create table cliente(cliente_id int primary key, nome varchar(50), datahora_registro date);
```

```
create or replace procedure proc_novocliente(p_nome varchar) is
    v_cliente_id int;
    v_datahora_registro date;
begin
    --Obter o próximo identificador.
    select max(cliente_id) into v_cliente_id from cliente;
    if v_cliente_id is null then
        v_cliente_id := 1;
    else
        v_cliente_id := v_cliente_id + 1;
    end if;
    --Instante atual, com a precisao do segundo.
    v_datahora_registro := sysdate;
    --Registrar o novo cliente.
    insert into cliente(cliente_id, nome, datahora_registro) values (v_cliente_id, p_nome, v_datahora_registro);
end;
```

PLSQL

Procedures

```
create table cliente(cliente_id int primary key, nome varchar(50), datahora_registro date);
```

```
create or replace procedure proc_procuranome(p_cliente_id int, p_nome out varchar)
IS
BEGIN
    --Obter o nome do cliente p_cliente_id
    select nome into p_nome
    from cliente
    where cliente_id = p_cliente_id;
END;
```

PLSQL

Invocando Procedures

- * Podemos invocar (executar) um procedimento a partir de:

- um bloco anônimo

```
begin
  proc_novocliente('Pedro');
  ....
end;
```

```
declare
  s_nome varchar(20);
begin
  proc_procuranome( 2, s_nome);
  ....
end;
```

- outro procedimento

```
CREATE OR REPLACE PROCEDURE subproc
  ...
END subproc;
```

```
CREATE OR REPLACE PROCEDURE mainproc
  ...
IS BEGIN
  ...
    subproc(...);
  ...
END mainproc;
```

- chamada "diretamente"

```
exec proc_novocliente('Pedro');
```

Não podemos chamar um procedimento de dentro de uma instrução SQL, como SELECT

PLSQL

Sumário

IN	OUT	IN OUT
Modo Default	Deve ser especificado	Deve ser especificado
O valor é passado para o subprograma	Retornou ao ambiente de chamada	Passado para o subprograma; retorna ao ambiente de chamada
Atua como parâmetro formal	Variável não inicializada	Variável inicializada
Parâmetro atual pode ser um literal, constante, expressão ou variável inicializada	Deve ser uma variável	Deve ser uma variável
Pode ser associado a um valor default	Não pode ser associado a um valor default	Não pode ser associado a um valor default

PLSQL

Functions

- * São subprogramas que têm por **objetivo retornar um resultado**;
- * Funções devem ser chamadas como parte de uma expressão SQL ou PL/SQL.
 - Podem ser usadas como **argumentos de um comando SELECT**, apenas **se não incluírem instruções de SQL DML insert/update/delete**;
 - **Certos tipos de retorno (booleano, por exemplo)** impedem a função de ser chamada como parte de um SELECT;
 - Em expressões PL/SQL, o identificador de função atua como uma variável cujo valor depende dos parâmetros passados para ele;
- * Uma função deve ter uma cláusula **RETURN** no cabeçalho e pelo menos uma instrução **RETURN na seção executável**.

PLSQL

Functions

- * O cabeçalho de uma função é como um cabeçalho de um PROCEDURE com duas diferenças:
 - O modo de parâmetro deve ser apenas IN.
 - A cláusula RETURN é usada no lugar do modo OUT.

```
CREATE [OR REPLACE] FUNCTION function_name
  [(parameter1 [mode1] datatype1, ...)]
RETURN datatype IS|AS
  [local_variable_declarations; ...]
BEGIN
  -- actions;
  RETURN expression;
END [function_name];
```

PLSQL

Create Functions

```
CREATE OR REPLACE FUNCTION get_sal
(p_id IN employees.employee_id%TYPE)
RETURN NUMBER IS
v_sal employees.salary%TYPE := 0;
BEGIN
SELECT salary
INTO v_sal
FROM employees
WHERE employee_id = p_id;
RETURN v_sal;
END get_sal;
```

```
CREATE OR REPLACE FUNCTION get_sal
(p_id IN employees.employee_id%TYPE) RETURN NUMBER IS
v_sal employees.salary%TYPE := 0;
BEGIN
SELECT salary INTO v_sal
FROM employees WHERE employee_id = p_id;
RETURN v_sal;
EXCEPTION
WHEN NO_DATA_FOUND THEN RETURN NULL;
END get_sal;
```

PLSQL

Create Functions

recursividade

```
CREATE OR REPLACE FUNCTION Factorial ( X number)
RETURN number
IS
F number;
BEGIN
IF X = 0 THEN
F:= 1;
ELSE
F:= x * factorial ( x - 1));
END IF;
RETURN F;
END;
```

PLSQL

Invocando Functions

- * Podemos invocar (executar) uma função a partir de:
 - uma expressão PL/SQL

```
DECLARE
    fact number;
BEGIN
    fact:= factorial(6);
    DBMS_OUTPUT.PUT_LINE( ' O factorial de' || ' é' || fact);
END;
```

- um parâmetro de outro subprograma
 - ❖ Neste exemplo, a função get_sal com todos os seus argumentos está aninhada no parâmetro requerido pelo procedimento DBMS_OUTPUT.PUT_LINE.

```
... DBMS_OUTPUT.PUT_LINE(get_sal(100)) ;
```

- uma expressão em uma instrução SQL.

```
SELECT job_id, get_sal(employee_id) FROM employees;
```

Restrições sobre o uso de funções em instruções SQL

- * Para usar uma função definida pelo utilizador numa instrução SQL, a função deve estar em conformidade com as regras e restrições da linguagem SQL;
- * A função pode aceitar apenas tipos de dados SQL válidos como parâmetros IN e deve RETORNAR um tipo de dados SQL válido;
- * Tipos específicos de PL /SQL, como BOOLEAN e % ROWTYPE, não são permitidos;
- * Funções chamadas numa instrução SELECT não podem conter instruções DML.

Restrições sobre o uso de funções em instruções SQL

- * Funções chamadas a partir de uma instrução UPDATE ou DELETE numa tabela não pode conter DML da mesma tabela;
- * Funções chamadas a partir de qualquer instrução SQL não podem terminar transações (isto é, não pode executar COMMIT ou ROLLBACK operações);
- * Funções chamadas a partir de qualquer instrução SQL não podem emitir DDL (por exemplo, CREATE TABLE) ou DCL (por exemplo, ALTER SESSION) porque também fazem um COMMIT implícito;
- * Chamadas para subprogramas que quebram estas restrições também não são permitidas em uma função.

Restrições sobre o uso de funções em instruções SQL

Exemplo 1

```
CREATE OR REPLACE FUNCTION dml_call_sql(p_sal NUMBER)
  RETURN NUMBER IS
BEGIN
  INSERT INTO employees(employee_id, last_name, email,
                        hire_date, job_id, salary)
  VALUES(1, 'Frost', 'jfrost@company.com',
         SYSDATE, 'SA_MAN', p_sal);
  RETURN (p_sal + 100);
END dml_call_sql;
```

```
UPDATE employees
  SET salary = dml_call_sql(2000)
 WHERE employee_id = 174;
```

ORA-04091: table US_1217_S90_PLSQL.EMPLOYEES is mutating, trigger/function may not see it

Funções chamadas a partir de uma instrução UPDATE ou DELETE numa tabela não pode conter DML da mesma tabela

Restrições sobre o uso de funções em instruções SQL

Exemplo 2

```
CREATE OR REPLACE FUNCTION query_max_sal (p_dept_id NUMBER)
  RETURN NUMBER IS
  v_num NUMBER;
BEGIN
  SELECT MAX(salary) INTO v_num FROM employees
    WHERE department_id = p_dept_id;
  RETURN (v_num);
END;
```

Quando usado dentro da seguinte instrução DML, ele retorna a mensagem de erro “mutating table” semelhante à mensagem de erro mostrada no slide anterior.

```
UPDATE employees SET salary = query_max_sal(department_id)
  WHERE employee_id = 174;
```

Functions - Retornando um Cursor

Exemplo

- * Um **REF CURSOR** é um tipo de dados PL/SQL cujo valor é o endereço de memória de uma área de trabalho de consulta na base de dados.
- * Em essência, um REF CURSOR é um ponteiro ou um identificador para um conjunto de resultados na base de dados.

```
create or replace function get_dept_emps(p_deptno in number) return
sys_refcursor is
    v_rc sys_refcursor;
begin
    OPEN v_rc FOR
        SELECT empno, ename, mgr, sal FROM emp
        WHERE deptno = p_deptno;
    return v_rc;
end;
```

```
declare
    v_rc sys_refcursor;
begin
    v_rc := get_dept_emps(10); -- This returns an open cursor
    dbms_output.put_line('Rows: ' || v_rc%ROWCOUNT);
    close v_rc;
end;
```