DATABASE PROGRAMMING

EMPACOTANDO OS ***ELEMENTOS DO BANCO***

9

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Sintaxe da criação do PACKAGE SPECIFICATION.....5
Código-fonte 2 – Exemplo de criação da especificação de um pacote apenas com constantes.....6
Código-fonte 3 – Exemplo de uso de constante do tipo texto definido na especificação do pacote.....6
Código-fonte 4 – Exemplo de uso de constante.....7
Código-fonte 5 – Exemplo de criação da especificação de um pacote 7
Código-fonte 6 – Exemplo do uso do comando DESC e seu resultado 8
Código-fonte 7 – Sintaxe da criação do PACKAGE BODY 9
Código-fonte 8 – Exemplo de criação do corpo de um pacote.....10
Código-fonte 9 – Exemplo de uso de uma função em um pacote usando bloco anônimo 11
Código-fonte 10 – Exemplo de uso de uma função em um pacote usando SQL 11
Código-fonte 11 – Exemplo de uso de um procedimento em um pacote usando bloco anônimo 11
Código-fonte 12 – Exemplo de uso de um procedimento executado com o comando EXEC 12
Código-fonte 13 – Exemplo de novas funcionalidades na especificação do pacote RH.....13
Código-fonte 14 – Exemplo de novas funcionalidades no corpo do pacote RH.....15
Código-fonte 15 – Exemplo de teste da função CONTRATA_FUNC.....15
Código-fonte 16 – Exemplo de teste do procedimento DEMITE_FUNC 15

SUMÁRIO

1 EMPACOTANDO OS ELEMENTOS DO BANCO 4
1.1 Definição 4
1.2 Sintaxe do Package Specification 4
1.2.1 Package Specification 5
1.3 Sintaxe do Body Specification 8
1.3.1 Package Body 9
CONCLUSÃO 16
REFERÊNCIAS 17

1 EMPACOTANDO OS ELEMENTOS DO BANCO

Depois de criarmos todos os procedimentos (*procedures*) e funções (*functions*), chegou a hora de agruparmos todos em pacotes. Uma das vantagens de trabalhar com pacotes é que permitem a organização das aplicações com mais eficiência. Imagine que criou várias funções e procedimentos para a área de contabilidade, com os pacotes, pode agrupá-los em um único objeto.

1.1 Definição

Para a Oracle (2016), pacotes são áreas de armazenamentos dos procedimentos ou PROCEDURES, funções ou FUNCTIONS, constantes, variáveis e cursores em PL/SQL que, dependendo do modo que construir, compartilharão as informações desse PACKAGE com outros aplicativos. Como regra geral, as chamadas aos pacotes darão referência a procedimentos ou funções.

Os pacotes também facilitam a tarefa de conceder privilégios para usuários e grupo de usuários executarem suas tarefas, permitem que os objetos do pacote sejam modificados sem que os objetos de esquema dependentes precisem ser recompilados, habilitam o Oracle a ler múltiplos objetos de pacote na memória de uma única vez e podem conter variáveis globais e cursores que estão disponíveis para todos os procedimentos e funções em um pacote.

1.2 Sintaxe do Package Specification

Para a Oracle (2016), um pacote ou *PACKAGE* possui duas partes. A primeira parte é chamada de especificação de pacote ou *PACKAGE SPECIFICATION* e a segunda parte é denominada de corpo do pacote ou *PACKAGE BODY*. São estas duas partes que possibilitam a criação do pacote no banco de dados. A especificação declara tudo que fará parte do pacote, e o corpo, por sua vez, apresentará o conteúdo do pacote propriamente dito. A sintaxe da especificação do pacote é a seguinte:

```
CREATE [ OR REPLACE ] PACKAGE nome_pacote
{IS ou AS}

[ variáveis ]

[ especificação dos cursores ]

[ especificação dos módulos ]

END [nome_pacote ];
```

Código-fonte 1 – Sintaxe da criação do PACKAGE SPECIFICATION
Fonte: ORACLE (2016)

CREATE OR REPLACE é a instrução para a criação ou a substituição do pacote.

nome_pacote é o nome que será dado ao pacote.

[variáveis] é a especificação do nome das variáveis, objetos públicos, tipos públicos, exceções e PRAGMAS públicas.

[cursores] é a especificação dos cursores.

[módulos] é o nome dos módulos do pacote.

Na especificação do pacote podemos definir novos tipos, declarar variáveis globais, tipos, objetos, exceções, cursores, procedimentos e funções. O que é definido na especificação do pacote poderá ser compartilhado com outros scripts ou programas em SQL e PL/SQL.

1.2.1 Package Specification

Para Puga, França e Goya (2015), Package Specification tem como função criar a interface das aplicações. São os tipos de variáveis: **cursores**, **exceções**, **nomear rotinas** e **funções**.

A especificação de um pacote será produzida antes da criação do corpo do pacote e pode existir sem que haja um corpo de pacote associado a ele.

Vejamos um exemplo simples:

```
CREATE OR REPLACE PACKAGE faculdade AS
   cnome CONSTANT VARCHAR2(4) := 'FIAP';
   cfone CONSTANT VARCHAR2(13) := '(11)3385-8010';
   cnota CONSTANT NUMBER(2) := 10;
END faculdade;
/
```

Código-fonte 2 – Exemplo de criação da especificação de um pacote apenas com constantes
Fonte: Elaborado pelo autor (2017)

Nesse exemplo, estamos criando um pacote denominado FACULDADE e definindo três constantes, CNOME, CFONE e CNOTA. Esse exemplo é um caso especial de criação de pacote. Como não temos subprogramas associados a ele, não é necessário criar um corpo de pacote.

Para referenciar as funções, procedimentos, itens e tipos definidos na especificação do pacote, usamos o nome e aquilo que queremos referenciar separados por um ponto.

Veja no exemplo:

```
SET SERVEROUTPUT ON

DECLARE
 melhor VARCHAR2(30);
BEGIN
  melhor := faculdade.cnome || ', a melhor faculdade';
  dbms_output.put_line(melhor);
END;
/
```

Código-fonte 3 – Exemplo de uso de constante do tipo texto definido na especificação do pacote
Fonte: Elaborado pelo autor (2017)

No exemplo, estamos referenciando a constante CNOME do pacote FACULDADE, usando o nome do pacote e da constante separados por um ponto, no caso, FACULDADE.CNOME.

Vejamos outro exemplo simples:

```
SET SERVEROUTPUT ON

DECLARE
 conta NUMBER(6);
BEGIN
  conta := faculdade.cnota ** 2;
  dbms_output.put_line(conta);
END;
/
```

Código-fonte 4 – Exemplo de uso de constante
do tipo numérico definido na especificação do pacote
Fonte: Elaborado pelo autor (2017)

No exemplo, estamos referenciando a constante CNOTA do pacote FACULDADE, usando o nome do pacote e da constante separados por um ponto, no caso, FACULDADE.CNOTA. Perceba que estamos elevando o valor da constante à potência de dois e atribuindo o resultado da conta a uma variável antes de exibi-lo.

Vejamos outro exemplo, desta vez, com a especificação de subprogramas:

```
CREATE OR REPLACE PACKAGE rh as
   FUNCTION descobrir_salario
     (p_id IN emp.empno%TYPE)
     RETURN NUMBER;
   PROCEDURE reajuste
     (v_codigo_emp IN emp.empno%type,
      v_porcentagem IN number DEFAULT 25);
END rh;
/
```

Código-fonte 5 – Exemplo de criação da especificação de um pacote
Fonte: Elaborado pelo autor (2017)

O exemplo acima cria a especificação de um pacote denominado RH. Nele, estamos declarando que existem dois subprogramas, a função DESCOBRIR_SALARIO e o procedimento REAJUSTE. A especificação da função informa que tem um parâmetro de entrada e o retorno de um valor numérico. A especificação do procedimento informa que tem dois parâmetros de entrada, ambos numéricos. A descrição dos parâmetros de entrada pode ser obtida por meio do comando DESC ou DESCRIBE.

Veja o exemplo abaixo:

```
DESC rh

FUNCTION DESCOBRIR_SALARIO RETURNS NUMBER
 Argument Name      Type       In/Out Default?
 ------------------ ---------- ------ --------
 P_ID               NUMBER(4)  IN
PROCEDURE REAJUSTE
 Argument Name      Type       In/Out Default?
 ------------------ ---------- ------ --------
 V_CODIGO_EMP       NUMBER(4)  IN
 V_PORCENTAGEM      NUMBER     IN     DEFAULT
```

Código-fonte 6 – Exemplo do uso do comando DESC e seu resultado
Fonte: Elaborado pelo autor (2017)

Observe que ao executar o comando DESC, indicando o pacote RH, descobriu-se que o pacote possui dois subprogramas: a função DESCOBRIR_SALÁRIO, que retorna um valor numérico (FUNCTION DESCOBRIR_SALARIO RETURNS NUMBER) com uma entrada numérica de quatro posições de nome P_ID; e um procedimento de nome REAJUSTE (PROCEDURE REAJUSTE) com dois parâmetros de entrada (V_CODIGO_EMP e V_PORCENTAGEM), ambos numéricos.

Para o nosso exemplo ficar completo, precisamos criar a especificação do pacote. Vejamos como isso pode ser feito.

1.3 Sintaxe do Body Specification

A sintaxe do corpo do pacote é a seguinte:

```
CREATE [ OR REPLACE ] PACKAGE BODY nome_pacote
{IS ou AS}

[ variáveis ]

[ especificação dos cursores ]

[ especificação dos módulos ]

  [BEGIN
    sequência_de_comandos
```

```
  [EXCEPTION
     exceções ] ]

END [nome_pacote ];
```

Código-fonte 7 – Sintaxe da criação do PACKAGE BODY
Fonte: ORACLE (2016)

CREATE OR REPLACE é a instrução para a criação ou a substituição do corpo do pacote.

nome_pacote é o nome que será dado ao pacote. É o mesmo usado na especificação do pacote.

[variáveis] é a especificação do nome das variáveis, objetos públicos, tipos públicos, exceções e PRAGMAS privadas.

[cursores] é a definição completa dos cursores.

[módulos] é a definição completa dos procedimentos e funções.

A sintaxe é similar à da criação da especificação do pacote, exceto pela palavra-chave BODY e o código implementado das especificações do pacote.

1.3.1 Package Body

Para a Oracle (2016), o corpo do pacote implementa as suas especificações. Em outras palavras, o corpo do pacote contém a implementação de cada cursor e subprograma declarados na sua especificação. É importante lembrar que os subprogramas definidos em um corpo do pacote são acessíveis fora se as suas especificações também aparecem nas especificações do pacote.

É no corpo do pacote que são definidas as variáveis privadas e onde estão os detalhes da implementação. Essas informações ficam ocultas da aplicação.

Pode depurar, melhorar ou substituir o corpo do pacote sem precisar alterar sua especificação. Vejamos um exemplo simples de criação do corpo de um pacote:

Empacotando os elementos do banco

```
CREATE OR REPLACE PACKAGE BODY rh
 AS

  FUNCTION descobrir_salario
    (p_id IN emp.empno%TYPE)
  RETURN NUMBER
  IS
    v_salario emp.sal%TYPE := 0;
  BEGIN
    SELECT sal INTO v_salario
      FROM emp
     WHERE empno = p_id;
    RETURN v_salario;
  END descobrir_salario;

  PROCEDURE reajuste
  (v_codigo_emp IN emp.empno%type,
   v_porcentagem IN number DEFAULT 25)
  IS
  BEGIN
  UPDATE emp
    SET sal = sal + (sal *( v_porcentagem / 100 ) )
   where empno = v_codigo_emp;
  COMMIT;
  END reajuste;

END rh;
/
```

Código-fonte 8 – Exemplo de criação do corpo de um pacote
Fonte: Elaborado pelo autor (2017)

O exemplo acima cria o corpo para o pacote RH. Aqui desenvolvemos os detalhes da implementação da função DESCOBRIR_SALARIO e do procedimento REAJUSTE. Como dito acima, esses detalhes de implementação ficam ocultos da aplicação. Vejamos um exemplo de uso desse pacote.

```
SET SERVEROUTPUT ON

DECLARE
   v_sal NUMBER(8,2);
BEGIN
   v_sal := rh.descobrir_salario(7900);
   DBMS_OUTPUT.PUT_LINE(v_sal);
END;
```

```
/
```

Código-fonte 9 – Exemplo de uso de uma função em um pacote usando bloco anônimo
Fonte: Elaborado pelo autor (2017)

O exemplo acima executa a função DESCOBRIR_SALARIO do pacote RH. Note que a chamada é feita por meio da RH.DESCOBRIR_SALARIO, ou seja, primeiro indicamos qual é o pacote e, em seguida, executamos a função, o nome do pacote e o nome da função são separados por um ponto. Na chamada da função, usamos o parâmetro 7900, o resultado é atribuído à variável V_SAL, exibida logo em seguida. Vejamos outra forma de executar o pacote.

```
SELECT rh.descobrir_salario(7900)
  FROM dual;
/
```

Código-fonte 10 – Exemplo de uso de uma função em um pacote usando SQL
Fonte: Elaborado pelo autor (2017)

Nesse caso, usamos uma consulta SQL para executar a função DESCOBRIR_SALARIO do pacote RH. Vamos a mais um teste de execução:

```
SET SERVEROUTPUT ON

DECLARE
   v_sal NUMBER(8,2);
BEGIN
   v_sal := rh.descobrir_salario(7900);
   DBMS_OUTPUT.PUT_LINE ('Salario atual - ' || v_sal);

   rh.reajuste (7900, faculdade.cnota);

   v_sal := rh.descobrir_salario(7900);
   DBMS_OUTPUT.PUT_LINE ('Salario atualizado - ' || v_sal);
END;
/
```

Código-fonte 11 – Exemplo de uso de um procedimento em um pacote usando bloco anônimo
Fonte: Elaborado pelo autor (2017)

O exemplo acima mostra vários usos dos recursos dos pacotes. O programa está usando a função RH.DESCOBRIR_SALARIO para obter o salário atual do

funcionário de código 7900 e, em seguida, exibir seu salário. O procedimento RH.REAJUSTE aumenta o salário de um funcionário específico, calculado por meio de um percentual informado.

Em nosso exemplo, o funcionário 7900 receberá o aumento de 10%, valor da constante FACULDADE.CNOTA. Em seguida, obtemos e exibimos o novo salário do funcionário. Perceba que, desta vez, nosso programa usou dados fornecidos por outro pacote. O procedimento poderia ter sido executado por meio do comando EXEC.

Veja no exemplo:

```
exec rh.reajuste (7900, faculdade.cnota);
```

Código-fonte 12 – Exemplo de uso de um procedimento executado com o comando EXEC
Fonte: Elaborado pelo autor (2017)

No exemplo acima, o procedimento RH.REAJUSTE atualiza o salário do funcionário 7900 em 10%. O valor do aumento foi obtido da constante CNOTA do pacote FACULDADE. Vamos acrescentar mais algumas funcionalidades no pacote RH:

```
CREATE OR REPLACE PACKAGE rh AS

  TYPE RegEmp IS RECORD
    (v_empno emp.empno%TYPE,
     v_sal emp.sal%TYPE);

  TYPE RegDept IS RECORD
    (v_deptno dept.deptno%TYPE,
     v_loc dept.deptno%TYPE);

  CURSOR c_sal RETURN RegEmp;

  salario_invalido EXCEPTION;

  FUNCTION contrata_func
    (v_ename  emp.ename%TYPE,
     v_job    emp.job%TYPE,
     v_mgr    emp.mgr%TYPE,
     v_sal    emp.sal%TYPE,
     v_comm   emp.comm%TYPE,
     v_deptno emp.deptno%TYPE)
```

```
      RETURN INT;

  PROCEDURE demite_func
     (v_empno emp.empno%TYPE);

  PROCEDURE reajuste
     (v_codigo_emp IN emp.empno%type,
      v_porcentagem IN number DEFAULT 25);

  FUNCTION maiores_salarios
     (n INT)
     RETURN RegEmp;

END rh;
/
```

Código-fonte 13 – Exemplo de novas funcionalidades na especificação do pacote RH
Fonte: Oracle (2016), adaptado pelo autor (2017)

Para deixar um pouco mais clara a potencialidade dos pacotes, fizemos algumas alterações na especificação do pacote RH. Agora, temos dois registros, REGEMP e REGDEPT, o cursor C_SAL, a exceção SALARIO_INVALIDO, as funções CONTRATA_FUNC e MAIORES_SALARIOS e os procedimentos DEMITE_FUNC e REAJUSTE. Iremos falar sobre registros em um módulo posterior. Vamos criar o corpo para o nosso pacote:

```
CREATE OR REPLACE PACKAGE BODY rh AS

  CURSOR c_sal RETURN RegEmp IS
    SELECT empno, sal FROM emp ORDER BY sal DESC;

  FUNCTION contrata_func (
      v_ename  emp.ename%TYPE,
      v_job    emp.job%TYPE,
      v_mgr    emp.mgr%TYPE,
      v_sal    emp.sal%TYPE,
      v_comm   emp.comm%TYPE,
      v_deptno emp.deptno%TYPE) RETURN INT IS
    cod_novo_emp INT;
  BEGIN
    SELECT max(empno) + 1 INTO cod_novo_emp FROM emp;
    INSERT INTO emp (empno, ename, job, mgr,
                     hiredate, sal, comm, deptno)
            VALUES (cod_novo_emp, v_ename, v_job,
                     v_mgr, SYSDATE, v_sal,
                     v_comm, v_deptno);
```

Empacotando os elementos do banco

```
    RETURN cod_novo_emp;
  END contrata_func;

  PROCEDURE demite_func (v_empno emp.empno%TYPE) IS
  BEGIN
    DELETE FROM emp WHERE empno = v_empno;
  END demite_func;

  FUNCTION sal_ok
    (v_sal emp.sal%TYPE)
     RETURN BOOLEAN IS
    min_sal emp.sal%TYPE;
    max_sal emp.sal%TYPE;
  BEGIN
    SELECT min(sal), max(sal) INTO
          min_sal, max_sal
      FROM emp;
    RETURN (v_sal >= min_sal) AND (v_sal <= max_sal);
  END sal_ok;

 PROCEDURE reajuste
 (v_codigo_emp IN emp.empno%type,
  v_porcentagem IN number DEFAULT 25)
 IS
  v_sal emp.sal%TYPE;
 BEGIN
    SELECT sal INTO v_sal
      FROM emp
     WHERE empno = v_codigo_emp;
    IF sal_ok(v_sal + (v_sal*(v_porcentagem/100))) THEN
      UPDATE emp
        SET sal =
            v_sal + (v_sal*(v_porcentagem/100))
       WHERE empno = v_codigo_emp;
    ELSE
      RAISE salario_invalido;
    END IF;
 END reajuste;

  FUNCTION maiores_salarios (n INT) RETURN RegEmp IS
    emp_rec RegEmp;
  BEGIN
    OPEN c_sal;
    FOR i IN 1..n LOOP
      FETCH c_sal INTO emp_rec;
    END LOOP;
    CLOSE c_sal;
    RETURN emp_rec;
  END maiores_salarios;
```

```
END rh;
```

Código-fonte 14 – Exemplo de novas funcionalidades no corpo do pacote RH
Fonte: Oracle (2016), adaptado pelo autor (2017)

O novo corpo do pacote RH contém a especificação do cursor C_SAL, uma função local denominada SAL_OK, a especificação de duas funções globais CONTRATA_FUNC e MAIORES_SALARIO e os procedimentos DEMITE_FUNC e REAJUSTE. Usaremos alguns desses módulos em módulos posteriores. Vamos testar dois dos novos subprogramas:

```
SET SERVEROUTPUT ON

DECLARE
  novo_cod emp.empno%TYPE;
BEGIN
  novo_cod :=
     rh.contrata_func('Rita','Gerente',7839,9000,NULL,10);
  DBMS_OUTPUT.PUT_LINE ('Funcionario ' || novo_cod || '
cadastrado');   END;   /
```

Código-fonte 15 – Exemplo de teste da função CONTRATA_FUNC
Fonte: Elaborado pelo autor (2017)

Nosso teste usa um bloco anônimo para executar a função RH.CONTRATA_FUNC, que cadastra um novo funcionário na tabela de empregados e retorna o código do novo funcionário cadastrado:

```
BEGIN
  rh.demite_func (7935);
END;
/
```

Código-fonte 16 – Exemplo de teste do procedimento DEMITE_FUNC
Fonte: Elaborado pelo autor (2017)

Esse novo procedimento usa o código do funcionário informado no parâmetro e o remove da tabela dos empregados.

Como você pode ver, o uso de pacotes ajuda na administração das funções e procedimentos do seu sistema, oferecendo facilidades de reuso e manutenção de código.

Empacotando os elementos do banco

CONCLUSÃO

Embora não seja obrigatória, a criação de pacotes é muito indicada para organizar os vários elementos/objetos de um banco de dados, facilitando muito o uso dos elementos e a aplicação e a revogação de permissões de segurança.

REFERÊNCIAS

DILLON, S.; BECK, C.; KYTE, T.; KALLMAN, J.; ROGERS, H. **Beginning Oracle Programming**. São Paulo: Apress, 2013.

FEUERSTEIN, S.; PRIBYL, B. **Oracle Pl/SqlProgramming**. 6. ed. California, USA: O'Reilly Media, 2014.

ORACLE, **Oracle Database:** PL/SQL Language Reference 12c Release 2 (12.2) B28370-05. USA: Oracle Press, 2016.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados**. São Paulo: Pearson, 2015.