

Pacotes

Programação em Banco de Dados

Computação@UFCG

Prof. Carlos Eduardo Santos Pires

Outline

- Partes de um Pacote
- Operações com Pacotes:
(re-)criação, implementação,
visualização, execução e
remoção
- Pacote sem corpo
- Escopo de um Pacote
- Sobrecarga de Função
Empacotada
- Variáveis Compartilhadas
- Ordem de Declarações

Pacotes

- São objetos do BD que agrupam logicamente elementos de programação PL/SQL em módulos.
 - Tipos definidos pelo usuário (object types), variáveis, exceções PL/SQL, cursores, procedures e funções.
- Um pacote funciona como uma biblioteca de funções e procedures.
- Variáveis e cursores são declarados da mesma forma que em blocos PL/SQL.

Pacotes

- As funções e procedures são criadas indiretamente, ou seja, cria-se um pacote que por sua vez contém funções e procedures.
- Podem possuir uma *while list* (cláusula ACCESSIBLE BY).

Partes de um Pacote

- **Especificação (PACKAGE)**

- Declara as implementações.
- Funciona como uma interface.

- **Corpo (PACKAGE BODY)**

- Contém as implementações das declarações declaradas na especificação.
- Possui o mesmo nome do PACKAGE.

Outline

- Partes de um Pacote
- Operações com Pacotes:
(re-)criação, implementação,
visualização, execução e
remoção
- Pacote sem corpo
- Escopo de um Pacote
- Sobrecarga de Função
Empacotada
- Variáveis Compartilhadas
- Ordem de Declarações

Sintaxe – PACKAGE

```
CREATE [OR REPLACE] [{EDITIONABLE | NONEDITIONABLE}] PACKAGE [schema.] package_name

[ACCESSIBLE BY

 ( [{FUNCTION | PROCEDURE | PACKAGE | TYPE}] [schema.]unit_name)
 [, [{FUNCTION | PROCEDURE | PACKAGE | TYPE}] [schema.]unit_name]]
[,... ]]]) {IS | AS}

[TYPE type_name IS

 {RECORD (column_list) | VARRAY(n) | TABLE [INDEX BY data_type]}]

[variable_name data_type {DEFAULT | :=} value; [ ...]]

[CURSOR cursor_name

 (parameter data_type [, parameter data_type [, ...]]) IS

 SELECT statement; [ ...]]

[TYPE reference_cursor IS REF CURSOR

 [ RETURN {catalog_row | cursor_row | record_structure} [ ...]]

[user_exception EXCEPTION; [ ...]]

[FUNCTION public_prototype;] [ ...]

[PROCEDURE public_prototype;] [ ...]

END [package_name];
```

Sintaxe – PACKAGE BODY

```
CREATE [OR REPLACE] PACKAGE BODY package_name {IS | AS}  
  [TYPE type_name IS  
    {RECORD (column_list) | VARRAY(n) | TABLE [INDEX BY data_type]}]  
  [variable_name data_type {DEFAULT | :=} value; [ ...]]  
  [CURSOR cursor_name  
    (parameter data_type [, parameter data_type [, ...]) IS  
    SELECT statement; [ ...]]  
  [TYPE reference_cursor IS REF CURSOR  
    [ RETURN {catalog_row | cursor_row | record_structure} [ ...]]  
  [FUNCTION local_implementation;] [ ...]  
  [PROCEDURE local_implementation;] [ ...]  
  [FUNCTION published_body;] [ ...]  
  [PROCEDURE published_body;] [ ...]  
END [package_name];
```


Criando a Especificação de um Pacote

```
CREATE OR REPLACE PACKAGE pck_emp
IS
    PROCEDURE query_emp
    (in_employee_id employees.employee_id%TYPE
    ,out_name OUT employees.first_name%TYPE
    ,out_salary OUT employees.salary%TYPE
    ,out_phone_number OUT employees.phone_number%TYPE);
    my_global_var NUMBER;
END pck_emp;
```

Implementando um Corpo de Pacote

```
CREATE OR REPLACE PACKAGE BODY pck_emp IS
  PROCEDURE query_emp
    (in_employee_id employees.employee_id%TYPE
    ,out_name OUT employees.first_name%TYPE
    ,out_salary OUT employees.salary%TYPE
    ,out_phone_number OUT employees.phone_number%TYPE) IS
  BEGIN
    SELECT first_name, salary , phone_number
    INTO out_name, out_salary, out_phone_number
    FROM employees
    WHERE employee_id = in_employee_id;
  END query_emp;
END pck_emp;
```

Executando uma Subrotina Empacotada

```
DECLARE
```

```
  aEmployee_id employees.employee_id%TYPE:=206;
```

```
  aFirst_name employees.first_name%TYPE;
```

```
  aSalary employees.salary%TYPE;
```

```
  aPhone_number employees.phone_number%TYPE;
```

```
BEGIN
```

```
  pck_emp.query_emp(aEmployee_id,aFirst_name,aSalary,aPhone_number);
```

```
  DBMS_OUTPUT.PUT_LINE('aFirst_name --> ' || aFirst_name);
```

```
  DBMS_OUTPUT.PUT_LINE('aSalary --> ' || aSalary);
```

```
  DBMS_OUTPUT.PUT_LINE('aPhone_number --> ' || aPhone_number);
```

```
END;
```

Removendo um Pacote

- **DROP PACKAGE BODY** pck_emp; -- remove apenas o corpo
- **DROP PACKAGE** pck_emp; -- remove especificação e corpo

Executando uma Subrotina Empacotada

- Toda função ou procedure declarada na especificação de um pacote deve ser implementada no corpo.
- A chamada a uma função ou procedure declarada, mas NÃO implementada, irá produzir um erro.

Executando uma Subrotina Empacotada

```
DROP PACKAGE BODY pck_emp;
```

```
DECLARE
    aEmployee_id employees.employee_id%TYPE:=206;
    aFirst_name employees.first_name%TYPE;
    aSalary employees.salary%TYPE;
    aPhone_number employees.phone_number%TYPE;
BEGIN
    pck_emp.query_emp(aEmployee_id,aFirst_name,aSalary,aPhone_number);
    DBMS_OUTPUT.PUT_LINE('aFirst_name --> ' || aFirst_name);
    DBMS_OUTPUT.PUT_LINE('aSalary --> ' || aSalary);
    DBMS_OUTPUT.PUT_LINE('aPhone_number --> ' || aPhone_number);
END;
```

DECLARE

ERRO na linha 1:

ORA-04067: não executado; package body "HR.PCK_EMP" não existe

ORA-06508: PL/SQL: não foi localizada a unidade de programa que está sendo chamada: "HR.PCK_EMP"

ORA-06512: em line 7

Visualizando a Especificação de um Pacote

- No SQL*Plus:

DESCRIBE **pck_emp**

PROCEDURE QUERY_EMP

Nome do Argumento	Tipo	In/Out Padrão?
-----	-----	-----
IN_EMPLOYEE_ID	NUMBER(6)	IN
OUT_NAME	VARCHAR2(20)	OUT
OUT_SALARY	NUMBER(8,2)	OUT
OUT_PHONE_NUMBER	VARCHAR2(20)	OUT

Atenção:

variáveis e tipos de dados do pacote não são exibidos no DESCRIBE (SQL*Plus).

Exemplo: variável “my_global_var”

Outline

- Partes de um Pacote
- Operações com Pacotes: (re-)criação, implementação, visualização, execução e remoção.
- **Pacote sem corpo**
- Escopo de um Pacote
- Sobrecarga de Função Empacotada
- Variáveis Compartilhadas
- Ordem de Declarações

Pacote sem Corpo (*Bodiless package*)

- Em alguns casos, pode não ser necessário criar um corpo de pacote.
 - Por exemplo, quando a especificação não tiver funções e procedures, não há nada a ser implementado no corpo.

Pacote sem Corpo (*Bodiless package*)

```
CREATE OR REPLACE PACKAGE bodiless IS  
    CURSOR c1 IS SELECT last_name  
                  FROM employees;  
END;
```

```
BEGIN  
    FOR emp_rec IN bodiless.c1 LOOP  
        dbms_output.put_line('Last name: '  
        || emp_rec.last_name);  
    END LOOP;  
END;
```

```
Last name: Abel  
Last name: Ande  
Last name: Atkinson  
Last name: Austin  
...
```

Outline

- Partes de um Pacote
- Operações com Pacotes: (re-)criação, implementação, visualização, execução e remoção.
- Pacote sem corpo
- Escopo de um Pacote
- Sobrecarga de Função Empacotada
- Variáveis Compartilhadas
- Ordem de Declarações

Escopo de Pacote

- Os componentes declarados na **especificação** de um pacote são **públicos**.
- Os elementos declarados apenas no **corpo** de um pacote são **protegidos**.

Escopo de Pacote

- Tipos de dados definidos pelo usuário públicos podem ser usados em outros programas.
- Os elementos definidos na seção de declaração de subrotinas públicas ou privadas são **privadas**.

Escopo de Pacote


```
CREATE OR REPLACE PACKAGE pack01 IS
```

```
  PROCEDURE proc01;
```

```
  FUNCTION func01 (p_data DATE) RETURN INTEGER;
```

```
  v_global NUMBER;
```

```
END;
```



Componentes Públicos:
visíveis fora do pacote

Escopo de Pacote

```
CREATE OR REPLACE PACKAGE BODY pack01 IS
  v_privada NUMBER; -- variável protegida: visível por proc02, proc01 e func01
  PROCEDURE proc02 IS -- procedure protegida: visível por proc01 e func01
  BEGIN
    null;
  END;
  PROCEDURE proc01 IS
  BEGIN
    proc02;
  END;
  FUNCTION func01 (p_data DATE) RETURN INTEGER IS
    v_var01 NUMBER; -- variável privada: visível apenas por func01
  BEGIN
    RETURN 1;
  END;
END pack01;
```

Outline

- Partes de um Pacote
- Operações com Pacotes: (re-)criação, implementação, visualização, execução e remoção.
- Pacote sem corpo
- Escopo de um Pacote
- Sobrecarga de Função Empacotada
- Variáveis Compartilhadas
- Ordem de Declarações

Sobrecarga de Função Empacotada

- A sobrecarga permite criar mais de uma função ou procedure com o mesmo nome mas com **assinaturas diferentes**.
- As assinaturas são definidas pela lista de parâmetros formais.
- Uma função ou procedure sobrecarregada diverge na quantidade de parâmetros ou na família dos tipos de dados dos parâmetros.

Sobrecarga de Função Empacotada – Exemplo 1

CREATE OR REPLACE PACKAGE **overloading** IS

FUNCTION adding (a **NUMBER**, b **NUMBER**) RETURN NUMBER;

FUNCTION adding (a **VARCHAR2**, b **NUMBER**) RETURN NUMBER;

FUNCTION adding (a **NUMBER**, b **VARCHAR2**) RETURN NUMBER;

FUNCTION adding (a **VARCHAR2**, b **VARCHAR2**) RETURN BINARY_INTEGER;

FUNCTION adding RETURN NUMBER;

FUNCTION adding (a **VARCHAR2**) RETURN NUMBER;

FUNCTION adding (a **NUMBER**) RETURN NUMBER;

END overloading;

Visualizando os Parâmetros das Subrotinas

```
SQL> DESCRIBE user_arguments
```

Nome	Nulo?	Tipo
-----	-----	-----
OBJECT_NAME		VARCHAR2(128)
PACKAGE_NAME		VARCHAR2(128)
OBJECT_ID	NOT NULL	NUMBER
OVERLOAD		VARCHAR2(40)
SUBPROGRAM_ID		NUMBER
ARGUMENT_NAME		VARCHAR2(128)
POSITION	NOT NULL	NUMBER
SEQUENCE	NOT NULL	NUMBER
DATA_TYPE		VARCHAR2(30)
...		

Visão USER_ARGUMENTS

SET linesize 120

COL object_name for a10

COL package_name for a15

COL argument_name for a5

COL data_type for a10

SELECT object_name, package_name, argument_name, position, data_type,
sequence, subprogram_id

FROM **user_arguments**

WHERE object_name = 'ADDING' AND package_name = 'OVERLOADING' AND
position <> 0

ORDER BY subprogram_id, sequence;

Visão USER_ARGUMENTS

OBJECT_NAME	PACKAGE_NAME	ARGUMENT_NAME	POSITION	DATA_TYPE	SEQUENCE	SUBPROGRAM_ID
ADDING	OVERLOADING	A	1	NUMBER	2	1
ADDING	OVERLOADING	B	2	NUMBER	3	1
ADDING	OVERLOADING	A	1	VARCHAR2	2	2
ADDING	OVERLOADING	B	2	NUMBER	3	2
ADDING	OVERLOADING	A	1	NUMBER	2	3
ADDING	OVERLOADING	B	2	VARCHAR2	3	3
ADDING	OVERLOADING	A	1	VARCHAR2	2	4
ADDING	OVERLOADING	B	2	VARCHAR2	3	4
ADDING	OVERLOADING	A	1	VARCHAR2	2	6
ADDING	OVERLOADING	A	1	NUMBER	2	7

Sobrecarga de Função Empacotada – Exemplo 2

```
CREATE OR REPLACE PACKAGE not_overloading IS  
    FUNCTION adding (a NUMBER, b NUMBER) RETURN NUMBER;  
    FUNCTION adding (one NUMBER, two NUMBER) RETURN  
BINARY_INTEGER;  
END not_overloading;
```

Sobrecarga de Função Empacotada – Exemplo 2

```
CREATE PACKAGE BODY not_overloading IS  
  FUNCTION adding (a NUMBER, b NUMBER) RETURN NUMBER IS  
  BEGIN  
    null;  
  END;  
  FUNCTION adding (one NUMBER, two NUMBER) RETURN BINARY_INTEGER IS  
  BEGIN  
    null;  
  END;  
END;
```

Sobrecarga de Função Empacotada – Exemplo 2

```
DECLARE
```

```
    v_num NUMBER;
```

```
BEGIN
```

```
    v_num := not_overloading.adding(1,2);
```

```
END;
```

ERRO na linha 4:

ORA-06550: linha 4, coluna 12:

PLS-00307: muitas declarações de
'ADDING' são compatíveis com
esta chamada

ORA-06550: linha 4, coluna 3:

PL/SQL: Statement ignored

Cláusula ACCESSIBLE BY

```
CREATE OR REPLACE PACKAGE small_one
```

```
ACCESSIBLE BY
```

```
(FUNCTION hr.gateway,
```

```
PROCEDURE bd2.backdoor,
```

```
PACKAGE hr.api,
```

```
TYPE xp.hobbit ) IS
```

```
FUNCTION add (lv_a NUMBER, lv_b NUMBER ) RETURN NUMBER;
```

```
END small_one;
```

Atenção: a cláusula ACCESSIBLE BY só pode ser usada na especificação do pacote.

Outline

- Partes de um Pacote
- Operações com Pacotes: (re-)criação, implementação, visualização, execução e remoção.
- Pacote sem corpo
- Escopo de um Pacote
- Sobrecarga de Função Empacotada
- **Variáveis Compartilhadas**
- Ordem de Declarações

Variáveis Compartilhadas

```
CONN hr/hr@pdborcl
```

```
CREATE OR REPLACE PACKAGE hr.pack10 IS  
  g_num NUMBER := 10;  
END;
```

```
BEGIN  
  dbms_output.put_line(pack10.g_num);  
END;
```

10

```
BEGIN
```

```
  pack10.g_num := 11;  
  dbms_output.put_line(pack10.g_num);  
END;
```

11

Variáveis Compartilhadas

```
CONN hr/hr@pdborcl
```

```
BEGIN
```

```
  dbms_output.put_line(pack10.g_num);
```

```
END;
```

```
10
```

- É possível forçar uma mudança e resetar as variáveis executando o comando:
- **ALTER PACKAGE pack10
COMPILE SPECIFICATION;**

Cursores Compartilhados

```
CREATE OR REPLACE PACKAGE pack02
IS
  CURSOR c1 IS
    SELECT last_name
    FROM employees
    WHERE salary > 12000;
  PROCEDURE proc01;
  PROCEDURE proc02;
END;
```

Cursoros Compartilhados

```
CREATE OR REPLACE PACKAGE pack02
```

```
IS
```

```
    CURSOR c1 IS
```

```
        SELECT last_name
```

```
        FROM employees
```

```
        WHERE salary > 12000;
```

```
    PROCEDURE proc01;
```

```
    PROCEDURE proc02;
```

```
END;
```

Cursores Compartilhados

```
CREATE OR REPLACE PACKAGE BODY pack02 IS
  v_last_name employees.last_name%TYPE;
  PROCEDURE proc01 IS
  BEGIN
    OPEN c1;
    LOOP
      FETCH c1 INTO v_last_name;
      EXIT WHEN (c1%ROWCOUNT > 4) OR
(c1%NOTFOUND);
      DBMS_OUTPUT.PUT_LINE(v_last_name);
    END LOOP;
  END;
```

```
PROCEDURE proc02 IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE(v_last_name);
    LOOP
      FETCH c1 INTO v_last_name;
      EXIT WHEN c1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_last_name);
    END LOOP;
    CLOSE c1;
  END;
END;
```

Cursoros Compartilhados

> SET SERVEROUTPUT ON

> EXEC pack02.proc01

King

Kochhar

De Haan

> EXEC pack02.proc02

Russell

Partners

Hartstein

Higgins

Outline

- Partes de um Pacote
- Operações com Pacotes: (re-)criação, implementação, visualização, execução e remoção.
- Pacote sem corpo
- Escopo de um Pacote
- Sobrecarga de Função Empacotada
- Variáveis Compartilhadas
- Ordem de Declarações

Ordem de Declarações

- Funções e procedures protegidas podem acessar todos os componentes contidos na especificação do pacote.
- Além disso, podem acessar componentes declarados somente antes deles no corpo do pacote (PL/SQL usa um parser de passo único).

Ordem de Declarações

- O parser falha se identificadores forem referenciados antes de serem declarados.
- Tipicamente os identificadores devem ser declarados na seguinte ordem:
 - tipos de dados
 - variáveis
 - exceções
 - funções
 - procedures

Ordem de Declarações

```
CREATE OR REPLACE PACKAGE ordem IS  
  PROCEDURE a;  
END;
```

```
CREATE OR REPLACE PACKAGE BODY ordem IS  
  PROCEDURE a IS  
  BEGIN  
    b;  
  END;  
  PROCEDURE b IS  
  BEGIN  
    null;  
  END;  
END;
```



Erros para PACKAGE BODY ORDEM:
LINE/COL ERROR

4/5 PLS-00313: 'B' não declarado nesta abrangência

Ordem de Declarações

```
CREATE OR REPLACE PACKAGE ordem IS  
    PROCEDURE a;  
END;
```

```
CREATE OR REPLACE PACKAGE BODY ordem IS  
    PROCEDURE b;  
    PROCEDURE a IS  
        BEGIN  
            b;  
        END;  
    PROCEDURE b IS  
        BEGIN  
            null;  
        END;  
END;
```



Corpo de Pacote criado.

Considerações Finais

- É possível alterar (recriar) o corpo de um pacote sem necessariamente ter que recriar a especificação do pacote.
- Não existe um comando para mover uma procedure ou função stand-alone para um pacote.
 - Deve ser feito manualmente.