Arquitetura e Administração de Bases de Dados
Oracle10g PL/SQL Programming

João Costa
jcosta@isec.pt

Agenda

- *Packages*

Packages

- Permitem encapsular objetos PL/SQL relacionados:
  - procedimentos,
  - funções,
  - constantes
  - exceções

Packages

- Divide-se em duas componente :
  - A especificação da package
  - O corpo ou definição da package

Packages - especificação

- É o interface da package
- Declara tipos, váriaveis, constantes, exceções, procedimentos e funções
- Podem ser referenciados de fora da package
- Contém informação sobre o conteúdo da package
- Não têm código PL/SQL
- Todos os objetos definidos são públicos
- Todo o código definido dentro do corpo é privado

Packages - especificação

```
CREATE OR REPLACE PACKAGE pack1 AS
  top exception;
  iva CONSTANT number := 0.23;

  FUNCTION  pvp(a number, b number) RETURN number;

  PROCEDURE test( a number);
END;
/
```

Nota: Não existe o CREATE function/procedure

Packages – corpo ou definição

- Implementação dos objetos definidos na especificação
- Também pode declarar tipos, váriaveis, constantes, exceções, procedimentos e funções
  - MAS SÃO PRIVATE
  - só podem ser chamados por objetos da package
- Contém o conteúdo da package
- Têm código PL/SQL

Packages – corpo ou definição

```
CREATE OR REPLACE PACKAGE BODY pack1 AS
  outro_erro exception;

  FUNCTION pvp (a number, b number) RETURN number IS
  BEGIN
    IF a = 1 THEN
       RAISE top;
    END IF;
    RETURN a + a*b;
  END;

  PROCEDURE test(a number) IS
   b number;
  BEGIN
     b := pvp(a, iva);
  EXCEPTION
    WHEN top THEN
      dbms_output.put_line('apanhei a top exception');
  END;
END;
```

Packages – Chamada

- Apenas os programas definidos na especificação podem ser executados (chamados) fora da package
- Utilizar como prefixo o nome da package
- Exemplos

```
EXEC   pack1.test(1);
SELECT pack1.pvp (preco_tabela) FROM livros;
```


Exercícios

Crie a package LIVRARIA com as seguintes definições:

- Constantes
  - `Iva_normal 23%`
  - `Iva_intermedia 13%`
  - `Iva_reduzido 6%`
- Excepções
  - `sem_stock`
  - `abaixo_preco_tabela`
- Funções
  - `Existe_em_stock(codlivro number) return number`
- Procedimento
  - `Compra (codlivro, codcliente, quant, preco)`
- Faça uma compra, chamando procedimento

Packages - especificação

```
CREATE OR REPLACE PACKAGE livraria AS
   livro_inexistente   exception;
   abaixo_preco_tabela exception;

  iva_normal     CONSTANT number := 0.23;
  iva_intermedio CONSTANT number := 0.13;
  iva_reduzido   CONSTANT number := 0.06;

  function  existe_em_stock (codlivro number) return number;
  procedure compra (codlivro number, codcliente number,
                    quant number, preco number);
END;
```

```
EXEC   livraria.compra(1,1,10,12);
```


Packages - especificação

```
CREATE OR REPLACE PACKAGE BODY livraria AS
  FUNCTION  existe_em_stock(codlivro number) RETURN number IS
    N number;
   BEGIN
    SELECT quant_em_stock INTO N FROM livros
    WHERE codigo_livro = cod_livro;
    RETURN N;
  EXCEPTION
    WHEN no_data_found THEN RAISE livro_inexistente;
   END;
  PROCEDURE compra (codlivro number, codcliente number, quant number,
   preco number) IS
  BEGIN
     if existe_em_stock (codilivro) >= quant then
    INSERT INTO vendas VALUES (seq_venda.nextval, SYSDATE,
            cod_cliente, cod_livro,quant,preco, preco * quant);
     end if;
 EXCEPTION
    WHEN livro_inexistente THEN RAISE null;
  END;
END;
```

Packages – especificação de cursores

Para um cursor **cb**

- é declarado na especificação da package, especificando
  - O nome
  - O tipo de dados de cada linha do resultado

```
CREATE PACKAGE pck_livraria AS
  CURSOR cb RETURN livros%ROWTYPE;  -- Declare cursor
END;
/
```

  - Não é especificada a pesquisa

Packages – especificação de cursores

Para um cursor **cb**

- é definido no corpo (BODY) da package, incluindo
  - o nome
  - O tipo de dados de cada linha do resultado
  - E a pesquisa do cursor

```
CREATE PACKAGE BODY pck_livraria AS
  CURSOR cb RETURN livros%ROWTYPE IS
    SELECT * FROM livros
    WHERE preco_tabela > 25;  -- Define cursor
END;
/
```

Packages – especificação de cursores

```
CREATE PACKAGE pck_livraria AS
  CURSOR cb RETURN livros%ROWTYPE;  -- Declare cursor
  PROCEDURE pa;
  PROCEDURE pb;
END;
/
```

```
EXECUTE pck_livraria.pa;      -- OK
EXECUTE pck_livraria.pb;      -- OK
```


Packages – especificação de cursores

```
CREATE PACKAGE BODY pck_livraria AS
   CURSOR cb RETURN livros%ROWTYPE IS
      SELECT * FROM livros
      WHERE preco_tabela > 25;  -- Define cursor
   CURSOR ca IS
      SELECT * FROM livros
      WHERE preco_tabela > 10;  -- Define cursor
   PROCEDURE pa IS
   BEGIN
      FOR r IN ca
      LOOP
       dbms_output.put_line('Livro ' || r.titulo);
      END LOOP;
    END;
   PROCEDURE pb IS
   BEGIN
      FOR r IN cb
      LOOP
       dbms_output.put_line('Livro ' || r.titulo);
      END LOOP;
    END;
END;
```

Packages – especificação de cursores

```
CREATE PACKAGE pck_livraria AS
  CURSOR cb RETURN livros%ROWTYPE;  -- Declare cursor
  PROCEDURE pa;
  PROCEDURE pb;
END;
/
```

```
EXECUTE pck_livraria.pa;      -- OK
EXECUTE pck_livraria.pb;      -- OK
```

```
CREATE PROCEDURE pc is
BEGIN
FOR R IN pck_livraria.cb
LOOP
  dbms_output.put_line('Livro ' || r.titulo);
END LOOP;
END;
```

```
EXECUTE pc;                   -- OK
```


Packages – especificação de cursores

```
CREATE PACKAGE pck_livraria AS
  CURSOR cb RETURN livros%ROWTYPE;  -- Declare cursor
  PROCEDURE pa;
  PROCEDURE pb;
END;
/
```

```
CREATE PROCEDURE pd is
BEGIN
FOR R IN pck_livraria.ca
LOOP
  dbms_output.put_line('Livro ' || r.titulo);
END LOOP;
```

```
LINE/COL ERROR
-------- -----------------------------------------------------
3/5      PL/SQL: Statement ignored
3/27     PLS-00302: o componente 'CA' deve ser declarado
Errors: check compiler log
```

Packages – Overloading

Permite overloading

- Funcões e procedimentos com multiplas especificações

```
CREATE PACKAGE pck_livraria1 AS
 FUNCTION getPreco(codLivro NUMBER) RETURN number;
 FUNCTION getPreco(codLivro NUMBER, iva NUMBER) RETURN number;

 PROCEDURE compra(codLivro NUMBER,codCliente NUMBER);
 PROCEDURE compra(titulo VARCHAR ,codCliente NUMBER);
 PROCEDURE compra(codLivro NUMBER,codCliente NUMBER, quant NUMBER);
END;
/
```


Packages – Overloading

```
CREATE or replace PACKAGE BODY pck_livraria1 AS
 FUNCTION getPreco(codLivro NUMBER) RETURN number is
  begin
       return codLivro;
  end;
 FUNCTION getPreco(codLivro NUMBER, iva NUMBER) RETURN number is
  begin
       return codLivro * (1 + iva);
  end;
 PROCEDURE compra(codLivro NUMBER,codCliente NUMBER)
  begin
       dbms_output.put_line('compra cod ' || codLivro );
  end;
 PROCEDURE compra(titulo VARCHAR ,codCliente NUMBER)
  begin
       dbms_output.put_line('compra tit ' || titulo);
  end;
 PROCEDURE compra(codLivro NUMBER,codCliente NUMBER, quant NUMBER) is
  begin
       dbms_output.put_line('compra codq ' || codLivro );
  end;
END;
```

Packages – Overloading

```
exec pck_livraria1.compra(1,1);
compra cod 1

exec pck_livraria1.compra(1,1,20);
compra codq 1

exec pck_livraria1.compra('Ola',1);
compra tit Ola

SELECT pck_livraria1.getPreco(20), pck_livraria1.getPreco(20,0.23)
FROM DUAL;
PCK_LIVRARIA1.GETPRECO(20) PCK_LIVRARIA1.GETPRECO(20,0.23)
-------------------------- ------------------------------
                        20                           24,6
```


Packages - Vantagens

- **Modularidade** - permite encapsular tipos, itens e sub-programas relacionados. Cada package fica fácil de entender, e as interfaces entre eles tornam-se mais simples, claras e bem definidas.
- **Aplicação simples de design** –ao especificar o interface na especificação da package, permite que a package possa ser compilada sem o seu corpo. Subprogramas que referenciam o pacote podem ser compilados também.
- **Ocultação de informação** – permite especificar quais tipos, itens e sub-programas serão públicos (visíveis e acessíveis) ou privados (escondido e inacessível). O package esconde a implementação de subprogramas privados e assim permite proteger a integridade dos packages.
- **Performance** – Ao executar um subprograma do package pela primeira vez, todo o package é carregado em memória. Chamadas subsequentes, desse ou de outro subprograma do package, serão mais rápidas e eficientes por estar em memória.

Exercícios

Criar a package ***promocoes,*** *que contém*

A. o cursor ***livros_da_semana*** que terá os livros editados esta semana
B. a ***taxa_de_iva*** dos livros (6%)
C. o procedimento ***desconto_da_semana*** que aplique uma percentagem de desconto (a receber como argumento) aos livros da semana
D. a função ***pvp*** que calcule o preço final após aplicar a taxa_de_iva ao preço a receber como argumento