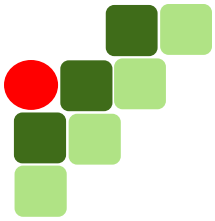
Banco de Dados II

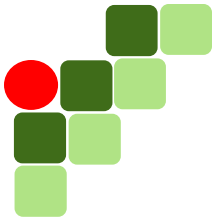
Cristiane Yaguinuma
cristiane.yaguinuma@ifsp.edu.br

- Packages PL/SQL



Roteiro da aula

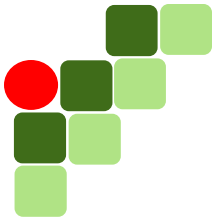
- O que é um package?
- Por que usar packages?
- Partes de um package
  - Especificação
  - Corpo
- Sobrecarga de subprogramas em packages
- Exercícios



O que é um package?

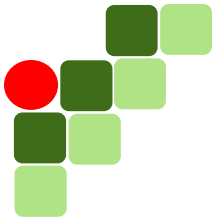
- É um objeto do banco de dados que agrupa tipos de dados, variáveis, subprogramas, cursores e exceções PL/SQL

- É compilado e armazenado no banco de dados e pode ser utilizado por aplicações externas



Por que usar packages?

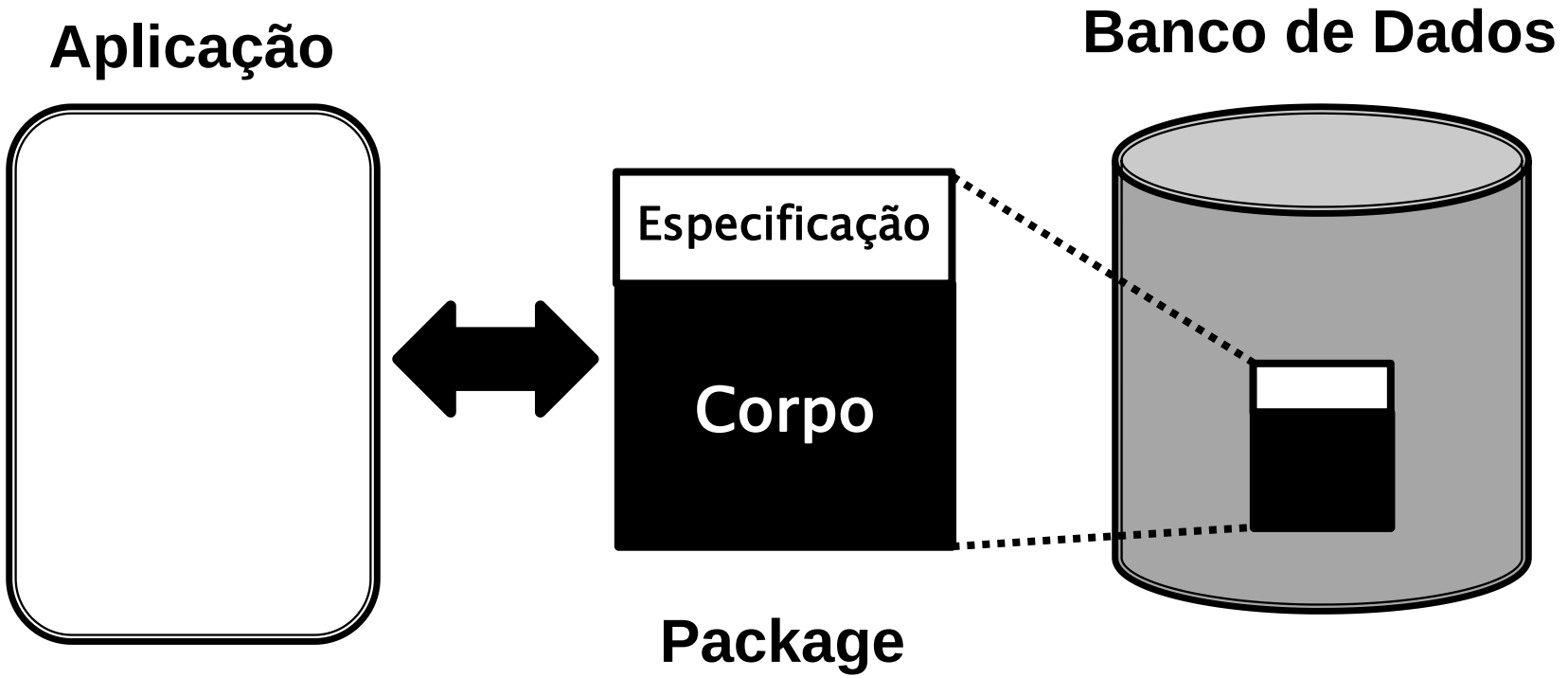
- Modularidade
  - Encapsular variáveis, subprogramas, cursores e exceções em módulos PL/SQL
  - Organizar funcionalidades de forma clara e bem definida para facilitar o desenvolvimento de aplicações
  - Modificar detalhes de implementação sem afetar a interface com aplicações



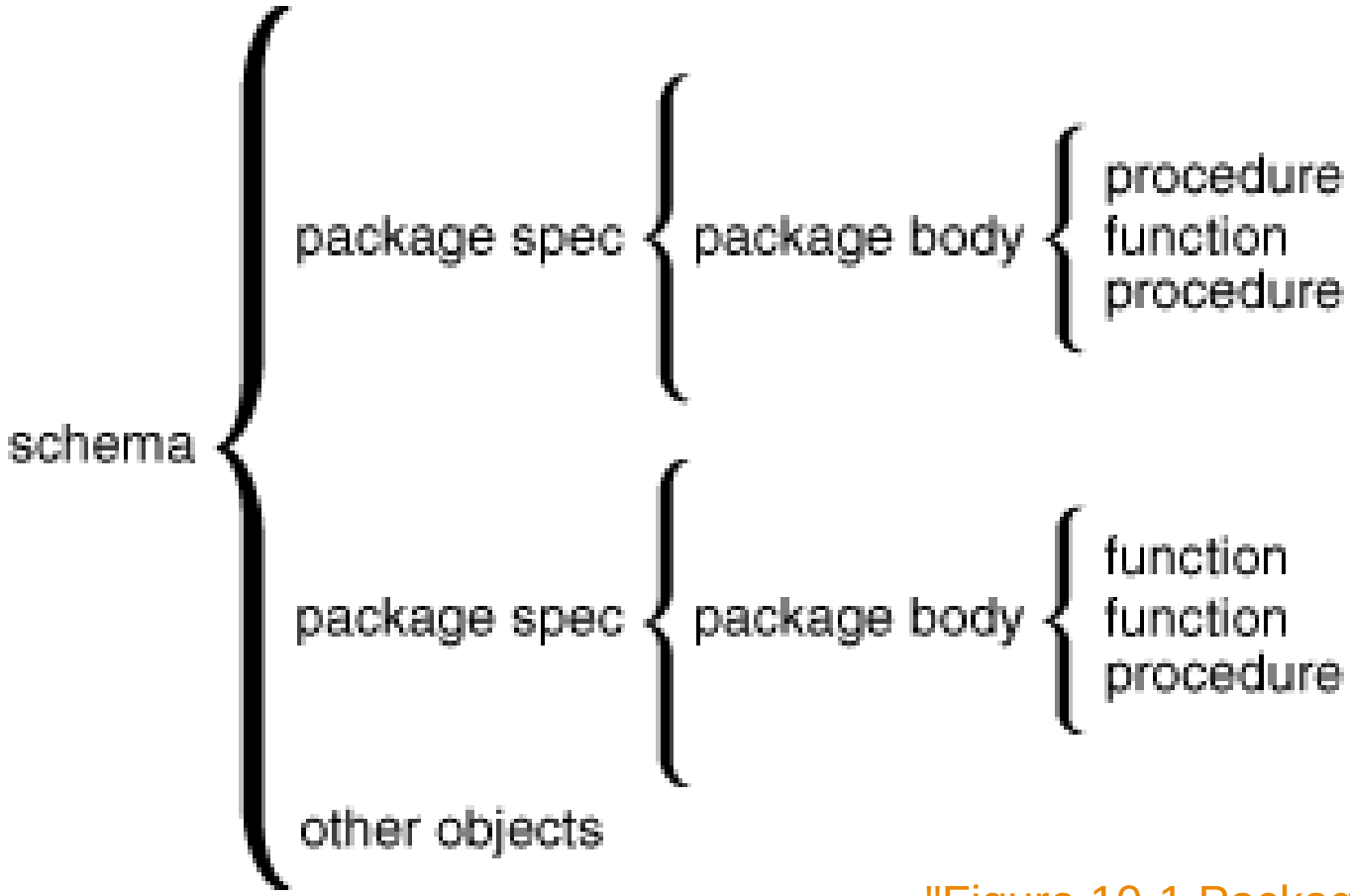
Por que usar packages?

- Melhor desempenho
  - Quando um subprograma de um package é invocado, o Oracle carrega o pacote inteiro em memória
  - Execuções subsequentes de outros subprogramas do mesmo pacote são mais rápidas
    - Não é necessário ler os subprogramas do disco (já estão em memória)

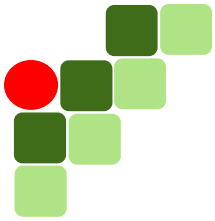
Partes de um package



Partes de um package



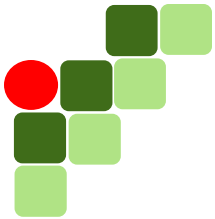
"Figure 10-1 Package Scope"



Partes de um package

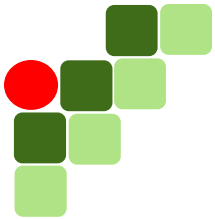
- Especificação – obrigatória
  - Declara os itens públicos que podem ser referenciados externamente ao pacote
  - Similar ao conceito de Application Programming Interface (API)

- Corpo (BODY) – opcional
  - Define consultas associadas a cursores e o código dos subprogramas públicos
  - Declara e define itens privados que não podem ser referenciados fora do pacote



Partes de um package

- Somente as declarações na especificação são visíveis e acessíveis por aplicações
- Os detalhes de implementação do corpo são ocultos e inacessíveis por aplicações
- É possível modificar o corpo sem alterar a especificação
  - O corpo do pacote é considerado uma “caixa-preta” para aplicações

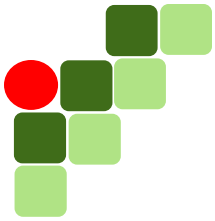


Especificação de um package

- Declaração de itens públicos
  - Tipos de dados, variáveis, constantes, subprogramas, cursores, exceções que são usadas por diversos subprogramas
  - Subprogramas que invocam uns aos outros
    - Não é necessário verificar a ordem de compilação dentro do pacote
  - Subprogramas sobrecarregados
    - Contêm o mesmo nome mas diferentes parâmetros

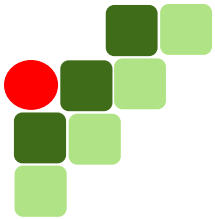
CREATE PACKAGE

```
[CREATE [OR REPLACE] PACKAGE package_name {IS | AS}
   [PRAGMA SERIALLY_REUSABLE;]
   [collection_type_definition ...]
   [record_type_definition ...]
   [subtype_definition ...]
   [collection_declaration ...]
   [constant_declaration ...]
   [exception_declaration ...]
   [object_declaration ...]
   [record_declaration ...]
   [variable_declaration ...]
   [cursor_body ...]
   [function_spec ...]
   [procedure_spec ...]
   [call_spec ...]
END [package_name];
```



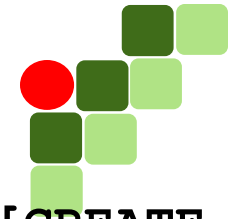
CREATE PACKAGE

```
CREATE OR REPLACE PACKAGE emp_admin AS
    TYPE EmpRecType IS RECORD (
       emp_id NUMBER,
       salary NUMBER);
    CURSOR desc_salary RETURN EmpRecType;
    invalid_id EXCEPTION;
    FUNCTION hire_employee(fname VARCHAR2, lname VARCHAR2,
       eemail VARCHAR2, jobid VARCHAR2, esalary NUMBER,
       mgr NUMBER, deptid NUMBER) RETURN NUMBER;
    PROCEDURE fire_employee (empid NUMBER);
    FUNCTION nth_highest_salary (n NUMBER) RETURN EmpRecType;
END emp_admin;
/
```



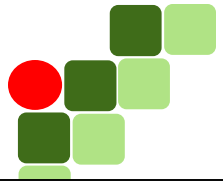
PACKAGE BODY

- O corpo do package implementa a especificação do package
- Se a especificação de um package contém a declaração de cursores ou subprogramas, é obrigatório definir o corpo do package
- Os mesmos nomes usados nas declarações da especificação devem ser usados no corpo



CREATE PACKAGE BODY

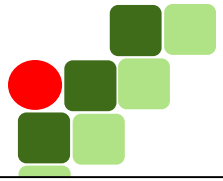
```
[CREATE [OR REPLACE] PACKAGE BODY package_name {IS | AS}
   [PRAGMA SERIALLY_REUSABLE;]
   [collection_type_definition ...]
   [record_type_definition ...]
   [subtype_definition ...]
   [collection_declaration ...]
   [constant_declaration ...]
   [exception_declaration ...]
   [object_declaration ...]
   [record_declaration ...]
   [variable_declaration ...]
   [cursor_body ...]
   [function_spec ...]
   [procedure_spec ...]
   [call_spec ...]
[BEGIN
    sequence_of_statements]
END [package_name];]
```



CREATE PACKAGE BODY

```
CREATE OR REPLACE PACKAGE BODY emp_admin AS
  number_hired NUMBER; -- visible only in this package
  CURSOR desc_salary RETURN EmpRecType IS
       SELECT employee_id, salary
       FROM EMPLOYEES
       ORDER BY salary DESC;

 FUNCTION hire_employee (fname VARCHAR2,
      lname VARCHAR2, eemail VARCHAR2, jobid VARCHAR2,
      esalary NUMBER, mgr NUMBER, deptid NUMBER)
   RETURN NUMBER IS
      new_emp_id NUMBER;
   BEGIN
      -- CONTINUA NO PROXIMO SLIDE
```

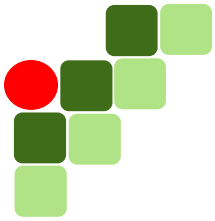
```
        new_emp_id := EMPLOYEES_SEQ.NEXTVAL;

        INSERT INTO EMPLOYEES(employee_id, first_name,
        last_name, email, hire_date, job_id, salary,
        manager_id, department_id)
        VALUES (new_emp_id, fname, lname, eemail, SYSDATE,
        jobid, esalary, mgr, deptid);

        number_hired := number_hired + 1;

        DBMS_OUTPUT.PUT_LINE('The number of employees hired
is ' || TO_CHAR(number_hired) );

        RETURN new_emp_id;

    END hire_employee;
    -- PACKAGE CONTINUA NO PROXIMO SLIDE
```



CREATE PACKAGE BODY

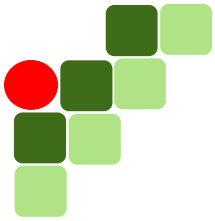
```
 PROCEDURE fire_employee (emp_id NUMBER) IS
 BEGIN
      DELETE FROM EMPLOYEES
      WHERE employee_id = emp_id;

      IF SQL%NOTFOUND THEN
            RAISE invalid_id;
      END IF;

      EXCEPTION
            WHEN invalid_id THEN
            DBMS_OUTPUT.PUT_LINE('Employee id not found.
Please try again with another id.');
 END fire_employee;
-- PACKAGE CONTINUA NO PROXIMO SLIDE
```

```
 FUNCTION nth_highest_salary ( n NUMBER )
 RETURN EmpRecType IS
    emp_rec EmpRecType;
 BEGIN
    OPEN desc_salary;
    FOR i IN 1..n LOOP
        FETCH desc_salary INTO emp_rec;
    END LOOP;
    CLOSE desc_salary;
    RETURN emp_rec;
 END nth_highest_salary;

BEGIN -- package initialization part [optional]
      number_hired := 0;
END emp_admin; -- FIM DO PACKAGE
/
```



Invocando subprogramas de packages

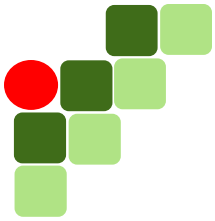
```
DECLARE
   new_emp_id NUMBER(6);
BEGIN
   new_emp_id  := emp_admin.hire_employee('João',
       'Carvalho','jcarvalho','IT_PROG',5000,null,90);
   emp_admin.fire_employee(new_emp_id);
END;
/

CALL DBMS_OUTPUT.PUT_LINE('The 10th highest salary
belongs to employee id: ' ||
TO_CHAR(emp_admin.nth_highest_salary(10).emp_id));
```



Invocando subprogramas de packages

- Em outros subprogramas PL/SQL

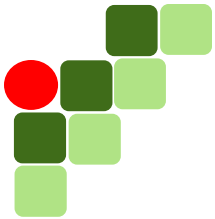
```
CREATE OR REPLACE PROCEDURE teste IS
    emp_rec emp_admin.EmpRecType;
BEGIN
...
    emp_admin.hire_employee('João', ...);
...
END;
/
```



Invocando subprogramas de packages

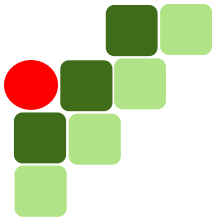
- Em programas externos (ex. java)

```
...
Connection con = ds.getConnection();
CallableStatement cs = con.prepareCall(
 "{call emp_admin.hire_employee(?,?,?,?,?,?,?)}");
cs.setString(1, "João");
...
cs.executeUpdate();
```



Package sem corpo

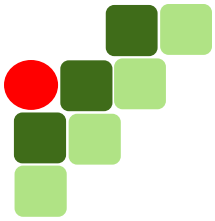
```
CREATE PACKAGE trans_data AS -- package sem corpo
   TYPE TimeRec IS RECORD (
      minutes SMALLINT,
      hours SMALLINT);
   TYPE TransRec IS RECORD (
      category VARCHAR2,
      account INT,
      amount REAL,
      time_of TimeRec);
   minimum_balance CONSTANT REAL := 10.00;
   number_processed INT;
   insufficient_funds EXCEPTION;
END trans_data;
/
```



Package sem corpo

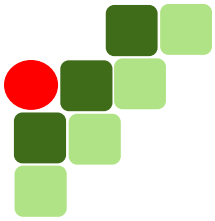
- No exemplo, o package trans_data não precisa ter definição de corpo
  - Tipos, constantes, variáveis e exceções não possuem uma implementação correspondente

- Packages sem corpo são usados para definir variáveis globais
  - Usadas por diversos subprogramas e triggers
  - Valores são mantidos durante uma sessão de acesso ao banco de dados



Itens públicos e privados em packages

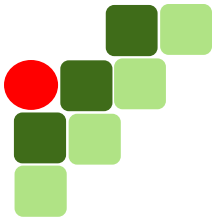
- Itens públicos
  - Itens declarados na especificação e implementados no corpo do package
- Itens privados
  - Itens definidos somente no corpo são restritos para uso dentro do package
  - Não são acessíveis por subprogramas PL/SQL fora do package



Sobrecarga de subprogramas em packages

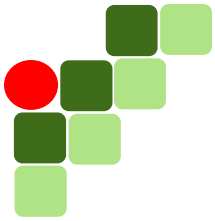
- É possível sobrecarregar dois ou mais subprogramas em um mesmo package
  - Mesmo nome com diferentes conjuntos de parâmetros

```
CREATE OR REPLACE PACKAGE overload_example AS
      PROCEDURE test_program (var1 VARCHAR2);

      PROCEDURE test_program (var1 DATE);

END overload_example;
/
```



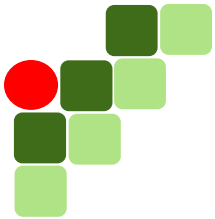
Sobrecarga de subprogramas em packages

```
CREATE PACKAGE BODY overload_example AS
      PROCEDURE test_program (var1 VARCHAR2) IS
      BEGIN
         DBMS_OUTPUT.PUT_LINE('Versão 1: ' || var1);
      END test_program;

      PROCEDURE test_program (var1 DATE) IS
      BEGIN
         DBMS_OUTPUT.PUT_LINE('Versão 2: ' || var1);
      END test_program;
END overload_example;
/
```



Packages específicos da Oracle

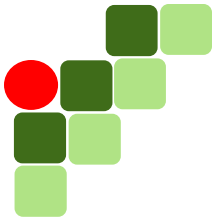
- DBMS_OUTPUT: exibir mensagens
  - Procedimentos put, put_line
- UTL_FILE: leitura e escrita de arquivos
  - Procedimentos open, put, get, close, get_line, put_line
- Mais packages da Oracle:
  - Database PL/SQL Packages and Types Reference



Recomendações

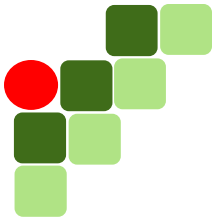
- Escreva packages com funcionalidades gerais
  - Promover a reutilização em várias aplicações
  - Evitar escrever packages que duplicam funcionalidades já providas pelo SGBD

- Defina a especificação do package antes do corpo
  - Coloque na especificação somente os elementos que devem estar visíveis aos programas que invocam o package



Recomendações

- Coloque o mínimo possível de itens na especificação
  - Para reduzir a necessidade de recompilar quando o código do package é modificado
- Mudanças no corpo do package não requerem a recompilação dos programas que o usam
- Mudanças na especificação requerem que cada programa que usa o package seja recompilado também



Referências

- Material e figuras extraídos de:
  - Oracle Database PL/SQL Language Reference
    - PL/SQL Packages
  - PL/SQL User's Guide and Reference Release 2 (9.2)
    - PL/SQL Packages
  - Using a PL/SQL Procedure With JDBC
  - Calling PL/SQL from Java