Database Programming

Packages

Sagara Samarawickrama

CSD 4204 - CPCM GP 1,2 & 3

Packages

A package is a collection of PL/SQL objects grouped together under one package name. Packages may include procedures, functions, cursors, declarations, types, and variables. Collecting objects into a package has numerous benefits

Creating Packages

There are numerous benefits of using packages as a method to bundle your functions and procedures, the first being that a well-designed package is a logical grouping of objects such as functions, procedures, global variables, and cursors. All of the code is loaded into memory on the first call of the package. This means that the first call to the package is very expensive (it involves a lot of processing on the server), but all subsequent calls will result in improved Performance. For this reason, packages are often used in applications where procedures and functions are called repeatedly.

Packages

Packages allow you to make use of some of the concepts involved in object-oriented programming, even though PL/SQL is not a “true” object-oriented programming language. With the PL/SQL package, you can collect functions and procedures and provide them with a context.

Creating Package Specifications

An additional level of security applies when using packages. When a user executes a procedure in a package (or stored procedures and functions), the procedure operates with the same permissions as its owner. Packages allow the creation of private functions and procedures, which can be called only from other functions and procedures in the package. This enforces information hiding. The structure of the package thus encourages top-down design.

The package specification contains information about the contents of the package, but not the code for the procedures and functions. It also contains declarations of global/public variables. Anything placed in the declaration section of a PL/SQL block may be coded in a package specification.

Packages

All objects placed in the package specification are called public objects. Any function or procedure not in the package specification but coded in a package body is called a private function or procedure.

When public procedures and functions are being called from a package, the programmer writing the “calling” process needs only the information in the package specification, as it provides all the information needed to call one of the procedures or functions within the package

The syntax for the package specification is as follows:

```
PACKAGE package_name
IS
 [ declarations of variables and types ]
 [ specifications of cursors ]
 [ specifications of modules ]
END [ package_name ];
```

Packages

Package Body

The package body contains the actual executable code for the objects described in the package specification. It contains the code for all procedures and functions described in the specification and may additionally contain code for objects not declared in the specification; the latter type of packaged object is invisible outside the package and is referred to as “hidden.” When creating stored packages, the package specification and body can be compiled separately.

```
PACKAGE BODY package_name
IS
 [ declarations of variables and types ]
 [ specification and SELECT statement of cursors ]
 [ specification and body of modules ]
 [ BEGIN
executable statements ]
 [ EXCEPTION
exception handlers ]
END [ package_name ];
```

Packages

Rules for the Package Body

A number of rules must be followed in package body code. First, there must be an exact match between the cursor and module headers and their definitions in package specification. Second, declarations of variables, exceptions, type, or constants in the specification cannot be repeated in the body. Third, any element declared in the specification can be referenced in the body.

You use the following notation when calling packaged elements from outside the package:

package_name.element..

You do not need to qualify elements when they are declared and referenced inside the body of the package or when they are declared in a specification and referenced inside the body of the same package.

The following example shows the package specification for the package

manage_students. Later we will describe the creation of the body of the same package

Packages

Rules for the Package Body

A number of rules must be followed in package body code. First, there must be an exact match between the cursor and module headers and their definitions in package specification. Second, declarations of variables, exceptions, type, or constants in the specification cannot be repeated in the body. Third, any element declared in the specification can be referenced in the body.

You use the following notation when calling packaged elements from outside the package:

package_name.element..

You do not need to qualify elements when they are declared and referenced inside the body of the package or when they are declared in a specification and referenced inside the body of the same package.

The following example shows the package specification for the package

manage_students. Later we will describe the creation of the body of the same package

Packages

```
CREATE OR REPLACE PACKAGE manage_students
AS
  PROCEDURE find_sname
   (i_student_id IN student.student_id%TYPE,
    o_first_name OUT student.first_name%TYPE,
    o_last_name OUT student.last_name%TYPE
   );
  FUNCTION id_is_good
   (i_student_id IN student.student_id%TYPE)
   RETURN BOOLEAN;
END manage_students;
```

Upon running this script, the specification for the package manage_students will be compiled into the database. The specification for the package now indicates that there is one procedure and one function. The procedure find_sname requires one IN parameter, the student ID; it returns two OUT parameters, the student’s first name and the student’s last name. The function id_is_good takes in a single parameter, a student ID, and returns a Boolean value (true or false).

Packages

Although the body has not yet been entered into the database, the package is still available for other applications. For example, if you included a call to one of these procedures in another stored procedure, that procedure would compile (but would not execute). This is illustrated by the following example.

```
SET SERVEROUTPUT ON
DECLARE
  v_first_name student.first_name%TYPE;
  v_last_name student.last_name%TYPE;
BEGIN
  manage_students.find_sname
    (125, v_first_name, v_last_name);
  DBMS_OUTPUT.PUT_LINE(v_first_name||' '||v_last_name);
END;
```

This procedure cannot run because only the specification for the procedure exists in the database, not the body

Packages

Creating Package Bodies

Now we will create the body of the manage_students packages,which were specified in the previous section.

```
CREATE OR REPLACE PACKAGE BODY manage_students
AS
  PROCEDURE find_sname
    (i_student_id IN student.student_id%TYPE,
     o_first_name OUT student.first_name%TYPE,
     o_last_name OUT student.last_name%TYPE
     )
  IS
   v_student_id  student.student_id%TYPE;
   BEGIN
     SELECT first_name, last_name
       INTO o_first_name, o_last_name
       FROM student
      WHERE student_id = i_student_id;
    EXCEPTION
     WHEN OTHERS
     THEN
       DBMS_OUTPUT.PUT_LINE
    ('Error in finding student_id: '||v_student_id);
    END find_sname;
```

Packages

```
   FUNCTION id_is_good
    (i_student_id IN student.student_id%TYPE)
    RETURN BOOLEAN
   IS
    v_id_cnt number;
   BEGIN
    SELECT COUNT(*)
      INTO v_id_cnt
      FROM student
     WHERE student_id = i_student_id;
     RETURN 1 = v_id_cnt;
    EXCEPTION
    WHEN OTHERS
    THEN
     RETURN FALSE;
    END id_is_good;
  END manage_students;
```

This script compiles the package manage_students into the database. The specification for the package indicates that there is one procedure and one function. The procedure find_sname requires one IN parameter, the student ID; it returns two OUT parameters, the student’s first name and the student’s last name

Packages

The function id_is_good takes in a single parameter of a student ID and returns a Boolean value (true or false). Although the body has not yet been entered into the database, the package is still available for other applications. For example, if you included a call to one of these procedures in another stored procedure, that procedure would compile (but would not execute).

Calling Stored Packages

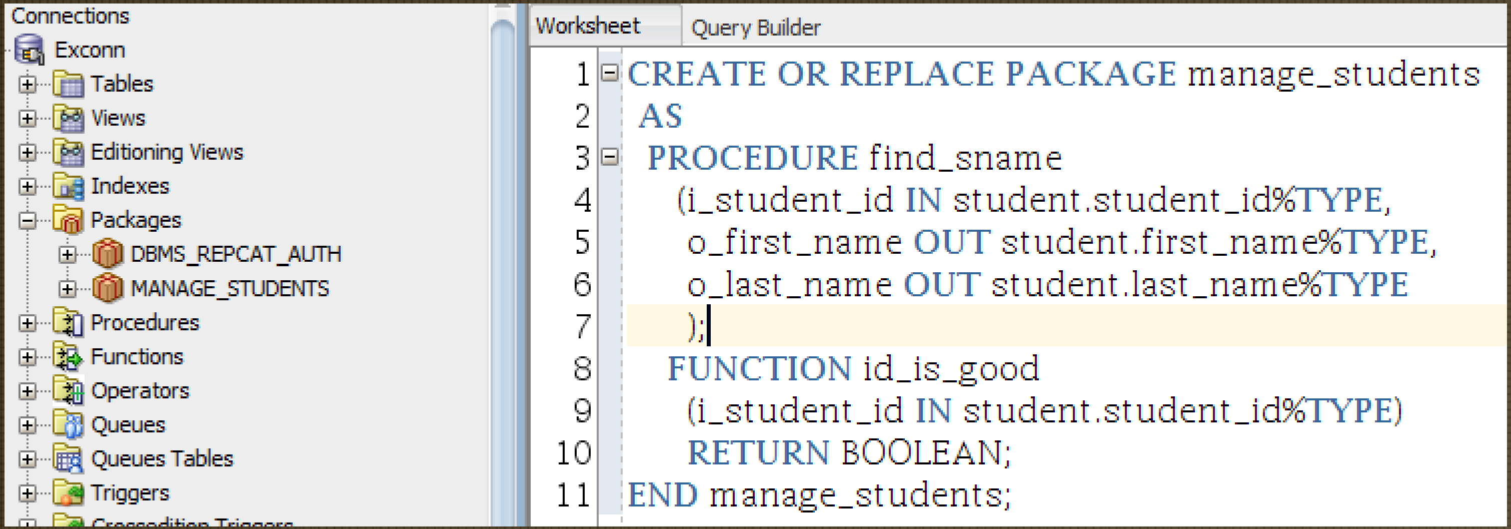
Now we will use elements of the manage_students package in another code block.

Packages

```
SET SERVEROUTPUT ON
DECLARE
  v_first_name student.first_name%TYPE;
  v_last_name student.last_name%TYPE;
BEGIN
  IF manage_students.id_is_good(&&v_id)
  THEN
    manage_students.find_sname(&&v_id, v_first_name,
       v_last_name);
  DBMS_OUTPUT.PUT_LINE('Student No. '||&&v_id||' is '
      ||v_last_name||', '||v_first_name);
ELSE
   DBMS_OUTPUT.PUT_LINE
   ('Student ID: '||&&v_id||' is not in the database.');
END IF;
END;
```

This is a correct PL/SQL block for running the function and the procedure in the package manage_students. If an existing student_id is entered, then the name of the student is displayed. If the student ID is not valid, then an error message is displayed.

Packages

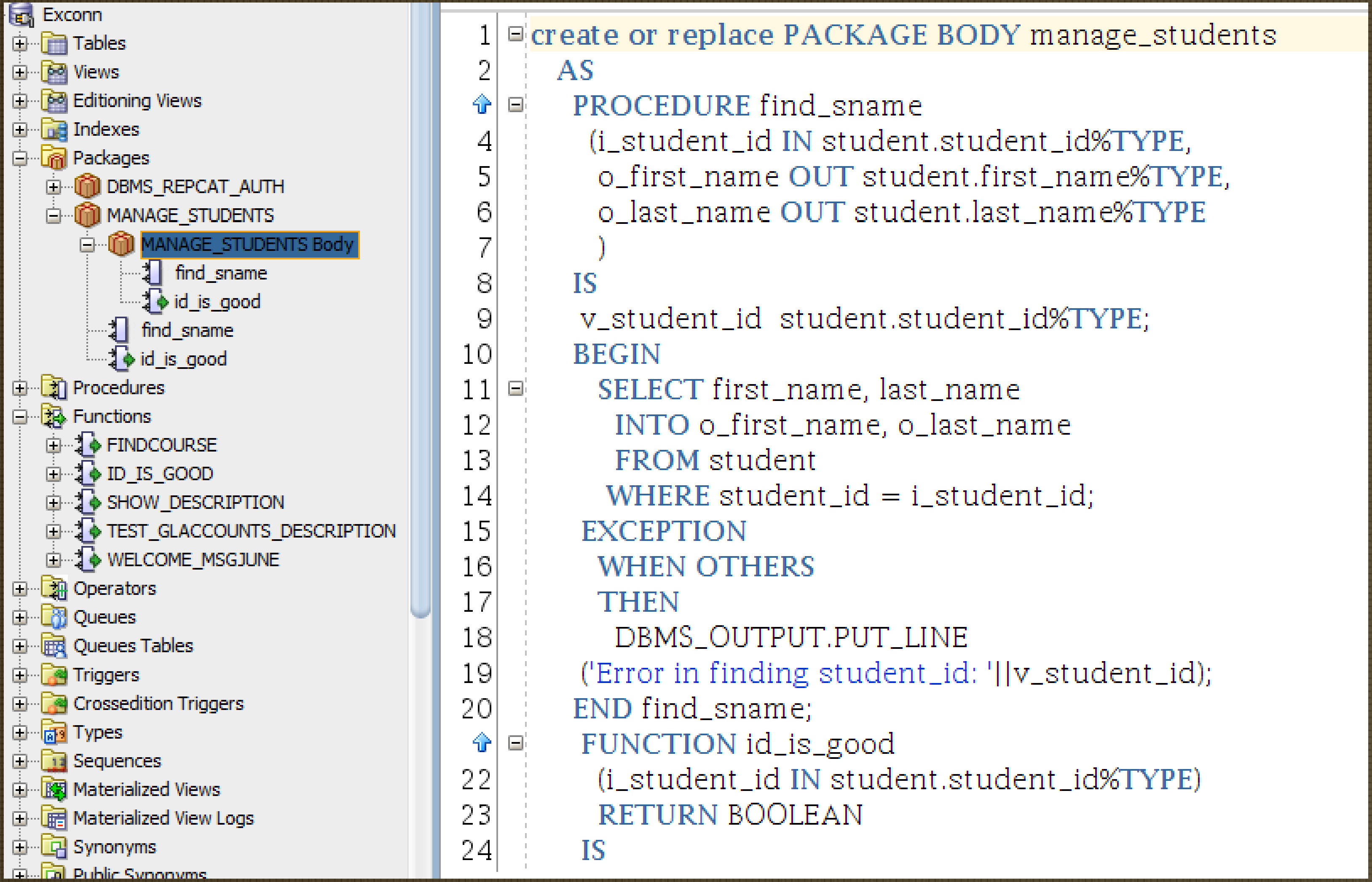


```
CREATE OR REPLACE PACKAGE manage_students
 AS
  PROCEDURE find_sname
     (i_student_id IN student.student_id%TYPE,
      o_first_name OUT student.first_name%TYPE,
      o_last_name OUT student.last_name%TYPE
      );
    FUNCTION id_is_good
      (i_student_id IN student.student_id%TYPE)
      RETURN BOOLEAN;
END manage_students;
```

Now managestudents package was created. That package contains two blocks one procedure named find_sname and one_function named id_is_good

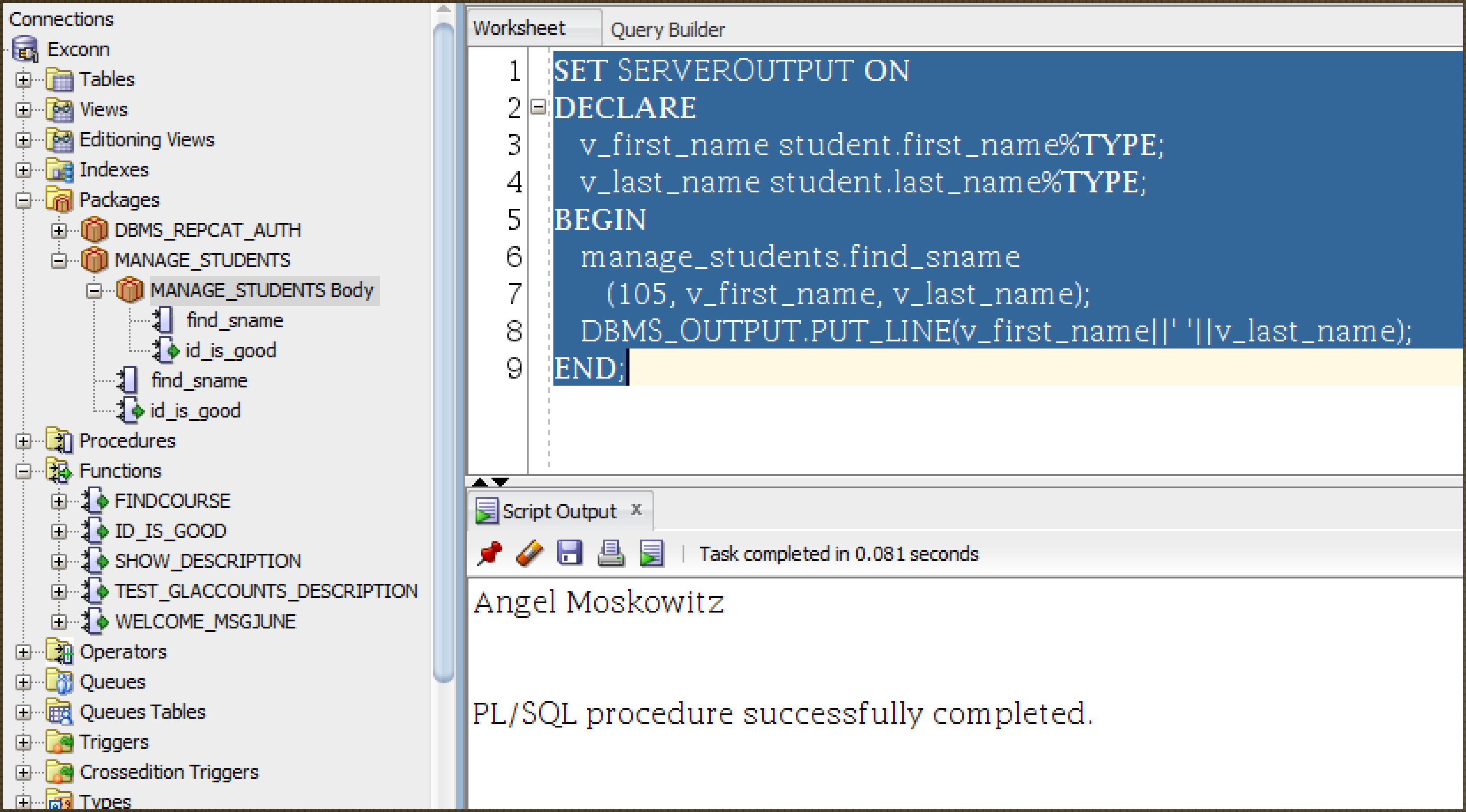
Then we create the package body

Packages



```
create or replace PACKAGE BODY manage_students
  AS
   PROCEDURE find_sname
     (i_student_id IN student.student_id%TYPE,
      o_first_name OUT student.first_name%TYPE,
      o_last_name OUT student.last_name%TYPE
      )
   IS
    v_student_id  student.student_id%TYPE;
   BEGIN
      SELECT first_name, last_name
        INTO o_first_name, o_last_name
        FROM student
       WHERE student_id = i_student_id;
    EXCEPTION
      WHEN OTHERS
      THEN
        DBMS_OUTPUT.PUT_LINE
    ('Error in finding student_id: '||v_student_id);
   END find_sname;
    FUNCTION id_is_good
      (i_student_id IN student.student_id%TYPE)
      RETURN BOOLEAN
    IS
```

Packages



```
SET SERVEROUTPUT ON
DECLARE
  v_first_name student.first_name%TYPE;
  v_last_name student.last_name%TYPE;
BEGIN
  manage_students.find_sname
    (105, v_first_name, v_last_name);
  DBMS_OUTPUT.PUT_LINE(v_first_name||' '||v_last_name);
END;
```

```
Angel Moskowitz


PL/SQL procedure successfully completed.
```

Q
&
A