Practices for Lesson 5: Creating Packages

Chapter 5

Practices for Lesson 5: Overview

Overview

In this practice, you create a package specification and body called `JOB_PKG`, containing a copy of your `ADD_JOB`, `UPD_JOB`, and `DEL_JOB` procedures as well as your `GET_JOB` function. You also create and invoke a package that contains private and public constructs by using sample data.

Note:

1. Before starting this practice, execute `/home/oracle/labs/plpu/code_ex/cleanup_scripts/cleanup_05.sql` script.
2. If you missed a step in a practice, please run the appropriate solution script for that practice step before proceeding to the next step or the next practice.

Practice 5-1: Creating and Using Packages

Overview

In this practice, you create package specifications and package bodies. You then invoke the constructs in the packages by using sample data.

Note: Execute `cleanup_05.sql script` from `/home/oracle/labs/plpu/code_ex/cleanup_scripts/` before performing the following tasks.

Task

1. Create a package specification and body called `JOB_PKG`, containing a copy of your `ADD_JOB`, `UPD_JOB`, and `DEL_JOB` procedures as well as your `GET_JOB` function.

   Note: Use the code from your previously saved procedures and functions when creating the package. You can copy the code in a procedure or function, and then paste the code into the appropriate section of the package.

   a. Create the package specification including the procedures and function headings as public constructs.

   b. Create the package body with the implementations for each of the subprograms.

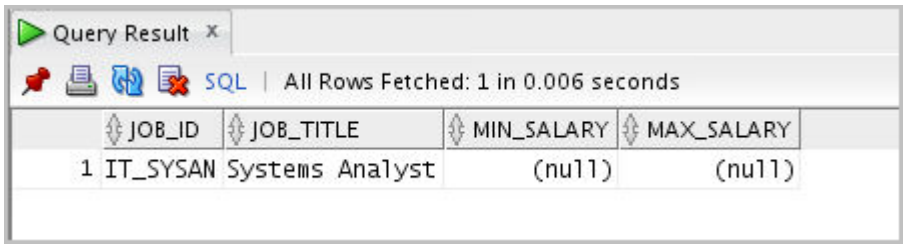
   c. Delete the following stand-alone procedures and function you just packaged using the Procedures and Functions nodes in the Object Navigation tree:

      1) The `ADD_JOB`, `UPD_JOB`, and `DEL_JOB` procedures

      2) The `GET_JOB` function

   d. Invoke your `ADD_JOB` package procedure by passing the values `IT_SYSAN` and `SYSTEMS ANALYST` as parameters.

   e. Query the `JOBS` table to see the result.



	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	IT_SYSAN	Systems Analyst	(null)	(null)

2. Create and invoke a package that contains private and public constructs.

   a. Create a package specification and a package body called `EMP_PKG` that contains the following procedures and function that you created earlier:

      1) `ADD_EMPLOYEE` procedure as a public construct

      2) `GET_EMPLOYEE` procedure as a public construct

      3) `VALID_DEPTID` function as a private construct

   b. Invoke the `EMP_PKG.ADD_EMPLOYEE` procedure, using department ID 15 for employee Jane Harris with the email ID `JAHARRIS`. Because department ID 15 does not exist, you should get an error message as specified in the exception handler of your procedure.

c. Invoke the `ADD_EMPLOYEE` package procedure by using department ID 80 for employee David Smith with the email ID `DASMITH`.

d. Query the `EMPLOYEES` table to verify that the new employee was added.

Solution 5-1: Creating and Using Packages

In this practice, you create package specifications and package bodies. You then invoke the constructs in the packages by using sample data.

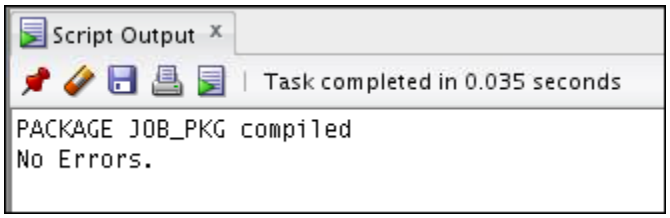
1. Create a package specification and body called `JOB_PKG`, containing a copy of your `ADD_JOB`, `UPD_JOB`, and `DEL_JOB` procedures as well as your `GET_JOB` function.

   Note: Use the code from your previously saved procedures and functions when creating the package. You can copy the code in a procedure or function, and then paste the code into the appropriate section of the package.

   a. Create the package specification including the procedures and function headings as public constructs.

      Open the `/home/oracle/labs/plpu/solns/sol_05.sql` script. Uncomment and select the code under Task 1_a. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the package specification. The code and the result are displayed as follows:

```
CREATE OR REPLACE PACKAGE job_pkg IS
  PROCEDURE add_job (p_jobid jobs.job_id%TYPE, p_jobtitle
jobs.job_title%TYPE);
  PROCEDURE del_job (p_jobid jobs.job_id%TYPE);
  FUNCTION get_job (p_jobid IN jobs.job_id%type) RETURN
jobs.job_title%type;
  PROCEDURE upd_job(p_jobid IN jobs.job_id%TYPE, p_jobtitle IN
jobs.job_title%TYPE);
END job_pkg;
/
SHOW ERRORS
```



   b. Create the package body with the implementations for each of the subprograms.

      Uncomment and select the code under Task 1_b. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the package body. The code and the result are displayed as follows:

```
CREATE OR REPLACE PACKAGE BODY job_pkg IS
  PROCEDURE add_job (
    p_jobid jobs.job_id%TYPE,
    p_jobtitle jobs.job_title%TYPE) IS
  BEGIN
    INSERT INTO jobs (job_id, job_title)
    VALUES (p_jobid, p_jobtitle);
    COMMIT;
```

```
  END add_job;

  PROCEDURE del_job (p_jobid jobs.job_id%TYPE) IS
    BEGIN
      DELETE FROM jobs
      WHERE job_id = p_jobid;
      IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20203, 'No jobs deleted.');
      END IF;
    END DEL_JOB;

  FUNCTION get_job (p_jobid IN jobs.job_id%type)
    RETURN jobs.job_title%type IS
    v_title jobs.job_title%type;
    BEGIN
      SELECT job_title
      INTO v_title
      FROM jobs
      WHERE job_id = p_jobid;
      RETURN v_title;
    END get_job;

  PROCEDURE upd_job(
    p_jobid IN jobs.job_id%TYPE,
    p_jobtitle IN jobs.job_title%TYPE) IS
    BEGIN
      UPDATE jobs
      SET job_title = p_jobtitle
      WHERE job_id = p_jobid;
      IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20202, 'No job updated.');
      END IF;
    END upd_job;

END job_pkg;
/

SHOW ERRORS
```

```
Script Output
Task completed in 0.03 seconds
Package body JOB_PKG compiled

No errors.
```

c. Delete the following stand-alone procedures and functions you just packaged by using the Procedures and Functions nodes in the Object Navigation tree:

   1) The `ADD_JOB`, `UPD_JOB`, and `DEL_JOB` procedures

   2) The `GET_JOB` function

   To delete a procedure or a function, right-click the procedure's name or function's name in the Object Navigation tree, and then select Drop from the pop-up menu. The Drop window is displayed. Click Apply to drop the procedure or function. A confirmation window is displayed. Click OK.

d. Invoke your `ADD_JOB` package procedure by passing the values `IT_SYSAN` and `SYSTEMS ANALYST` as parameters.

   Uncomment and select the code under Task 1_d. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to invoke the package's procedure. The code and the result are displayed as follows:

```
EXECUTE job_pkg.add_job('IT_SYSAN', 'Systems Analyst')
```

```
Script Output
Task completed in 0.057 seconds
PL/SQL procedure successfully completed.
```

e. Query the `JOBS` table to see the result.

   Uncomment and select the code under Task 1_e. Click the Run Script icon (or press F5) or the Execute Statement (or press F9) on the SQL Worksheet toolbar to query the `JOBS` table. The code and the result are displayed as follows:

```
SELECT *
FROM jobs
WHERE job_id = 'IT_SYSAN';
```

Query Result

SQL | All Rows Fetched: 1 in 0.001 seconds

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	IT_SYSAN	Systems Analyst	(null)	(null)

2. Create and invoke a package that contains private and public constructs.

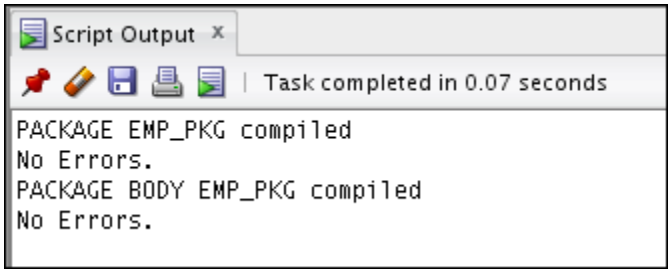
   a. Create a package specification and a package body called `EMP_PKG` that contains the following procedures and function that you created earlier:

1) `ADD_EMPLOYEE` procedure as a public construct
2) `GET_EMPLOYEE` procedure as a public construct
3) `VALID_DEPTID` function as a private construct

Uncomment and select the code under Task 2_a. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to invoke the package's procedure. The code and the result are displayed as follows:

```
CREATE OR REPLACE PACKAGE emp_pkg IS
  PROCEDURE add_employee(
    p_first_name employees.first_name%TYPE,
    p_last_name employees.last_name%TYPE,
    p_email employees.email%TYPE,
    p_job employees.job_id%TYPE DEFAULT 'SA_REP',
    p_mgr employees.manager_id%TYPE DEFAULT 145,
    p_sal employees.salary%TYPE DEFAULT 1000,
    p_comm employees.commission_pct%TYPE DEFAULT 0,
    p_deptid employees.department_id%TYPE DEFAULT 30);
PROCEDURE get_employee(
    p_empid IN employees.employee_id%TYPE,
    p_sal OUT employees.salary%TYPE,
    p_job OUT employees.job_id%TYPE);
END emp_pkg;
/
SHOW ERRORS

CREATE OR REPLACE PACKAGE BODY emp_pkg IS
  FUNCTION valid_deptid(p_deptid IN
departments.department_id%TYPE) RETURN BOOLEAN IS
    v_dummy PLS_INTEGER;
  BEGIN
    SELECT 1
    INTO v_dummy
    FROM departments
    WHERE department_id = p_deptid;
    RETURN TRUE;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
    RETURN FALSE;

END valid_deptid;

  PROCEDURE add_employee(
    p_first_name employees.first_name%TYPE,
    p_last_name employees.last_name%TYPE,
```

```
    p_email employees.email%TYPE,
    p_job employees.job_id%TYPE DEFAULT 'SA_REP',
    p_mgr employees.manager_id%TYPE DEFAULT 145,
    p_sal employees.salary%TYPE DEFAULT 1000,
    p_comm employees.commission_pct%TYPE DEFAULT 0,
    p_deptid employees.department_id%TYPE DEFAULT 30) IS
  BEGIN
    IF valid_deptid(p_deptid) THEN
      INSERT INTO employees(employee_id, first_name, last_name,
email,
        job_id, manager_id, hire_date, salary, commission_pct,
department_id)
      VALUES (employees_seq.NEXTVAL, p_first_name, p_last_name,
p_email,
        p_job, p_mgr, TRUNC(SYSDATE), p_sal, p_comm, p_deptid);
    ELSE
      RAISE_APPLICATION_ERROR (-20204, 'Invalid department ID.
Try again.');
    END IF;
  END add_employee;

  PROCEDURE get_employee(
    p_empid IN employees.employee_id%TYPE,
    p_sal OUT employees.salary%TYPE,
    p_job OUT employees.job_id%TYPE) IS
  BEGIN
    SELECT salary, job_id
    INTO p_sal, p_job
    FROM employees
    WHERE employee_id = p_empid;
  END get_employee;
END emp_pkg;
/
SHOW ERRORS
```



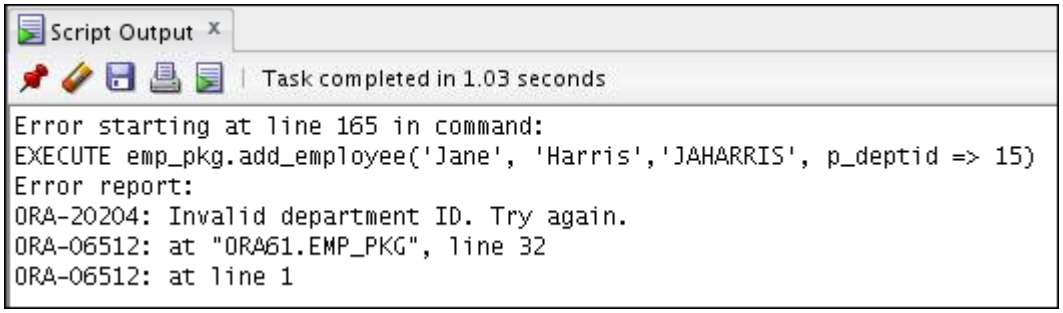
b. Invoke the `EMP_PKG.ADD_EMPLOYEE` procedure, using department ID 15 for employee Jane Harris with the email ID `JAHARRIS`. Because department ID 15 does

not exist, you should get an error message as specified in the exception handler of your procedure.

Uncomment and select the code under Task 2_b. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to invoke the package's procedure. The code and the result are displayed as follows:

Note: You must complete step 5-2-a before performing this step. If you didn't complete step 5-2-a, run the code under Task 2_a first.

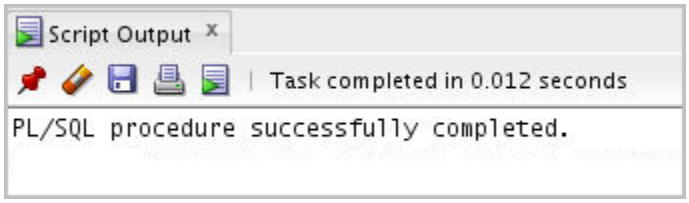
```
EXECUTE emp_pkg.add_employee('Jane', 'Harris','JAHARRIS',
p_deptid => 15)
```



c. Invoke the `ADD_EMPLOYEE` package procedure by using department ID 80 for employee David Smith with the email ID `DASMITH`.

Uncomment and select the code under Task 2_c. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to invoke the package's procedure. The code and the result are displayed as follows:

```
EXECUTE emp_pkg.add_employee('David', 'Smith','DASMITH',
p_deptid => 80)
```



d. Query the `EMPLOYEES` table to verify that the new employee was added.

Uncomment and select the code under Task 2_d. Click the Run Script icon (or press F5) or the Execute Statement icon (or press F9), while making sure the cursor is on any of the `SELECT` statement code, on the SQL Worksheet toolbar to query the `EMPLOYEES` table. The code and the result (Execute Statement icon) are displayed as follows:

```
SELECT *
FROM employees
WHERE last_name = 'Smith';
```

The following output is displayed in the Results tab because we executed the code using the F9 icon.

Query Result

SQL | All Rows Fetched: 3 in 0.006 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	C
1	210	David	Smith	DASMITH	(null)	17-OCT-16	SA_REP	1000	
2	159	Lindsey	Smith	LSMITH	011.44.1345.729268	10-MAR-13	SA_REP	8000	
3	171	William	Smith	WSMITH	011.44.1343.629268	23-FEB-15	SA_REP	7400	