Database Programming with PL/SQL

11-1: Persistent State of Package Variables

Practice Activities

Vocabulary

Identify the vocabulary word for the definition below:

Package State	The collection of package variables and their current values.

Try It / Solve It

1. Since Oracle Academy's Application Express (APEX) automatically commits changes, complete the following activity as if you were issuing the commands in an installed/local APEX environment with the ability to use COMMIT and ROLLBACK (without AUTOCOMMIT).

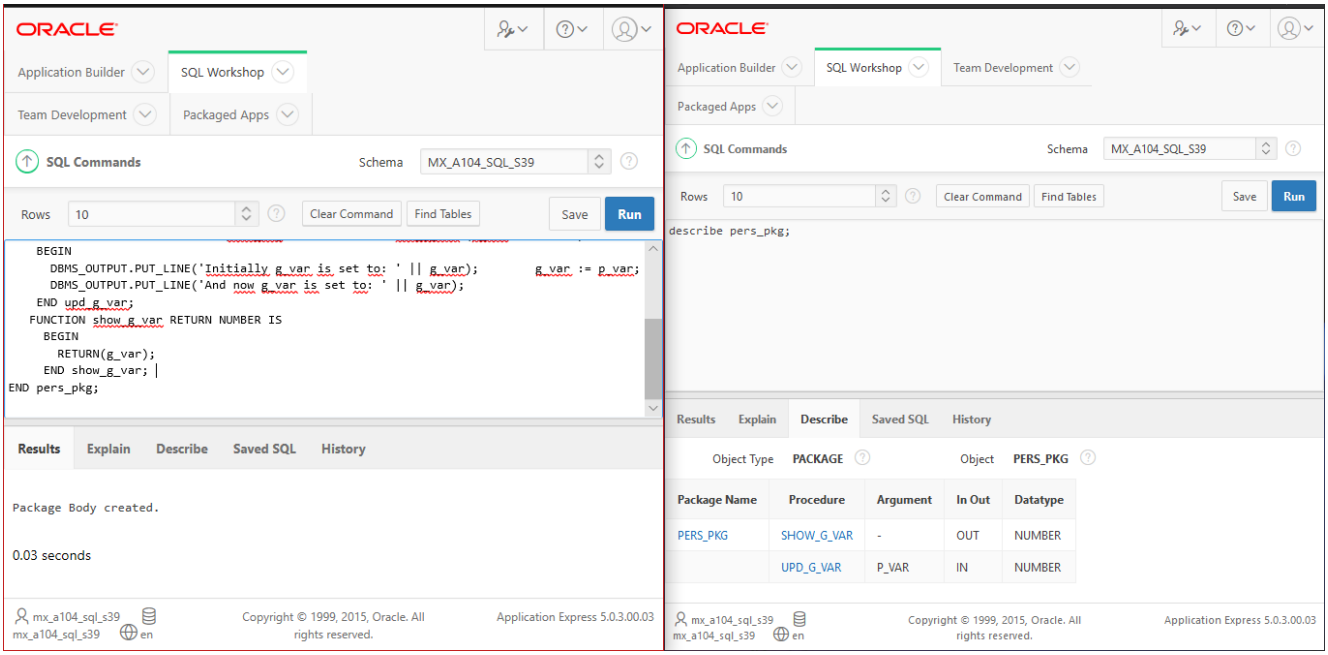
   In this question you will use a slightly modified version of the pers_pkg package which you studied in the lesson. You will need to have two APEX sessions running throughout this question, so start by having two browser sessions running (sometimes using different browsers for each session is helpful), each logged into APEX with your normal credentials. Also, do not leave the SQL Commands window during this question.

   A. In one of your sessions, create the package specification and body using the following code:

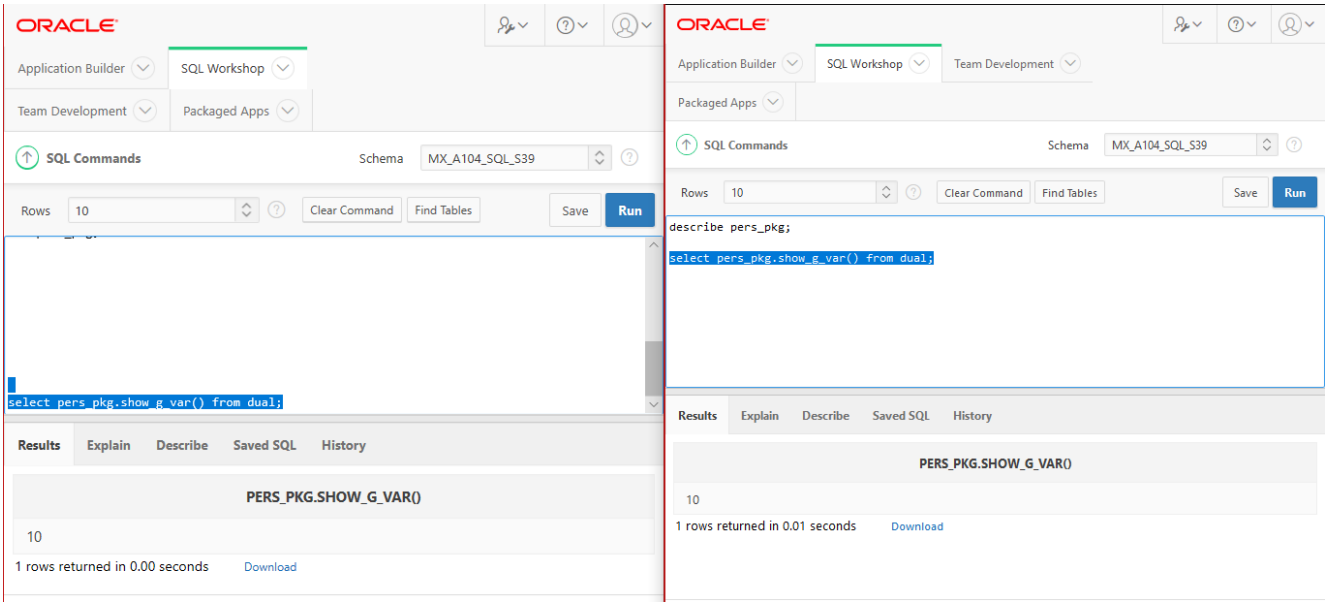
```
CREATE OR REPLACE PACKAGE pers_pkg IS
g_var    NUMBER := 10;
  PROCEDURE upd_g_var (p_var IN NUMBER);
FUNCTION show_g_var RETURN number;
END pers_pkg;
```

```
CREATE OR REPLACE PACKAGE BODY pers_pkg IS
PROCEDURE upd_g_var (p_var IN NUMBER) IS
   BEGIN
     DBMS_OUTPUT.PUT_LINE('Initially g_var is set to: ' || g_var);
g_var := p_var;
     DBMS_OUTPUT.PUT_LINE('And now g_var is set to: ' || g_var);
   END upd_g_var;
  FUNCTION show_g_var RETURN NUMBER IS
    BEGIN
     RETURN(g_var);
    END show_g_var;
END pers_pkg;
```

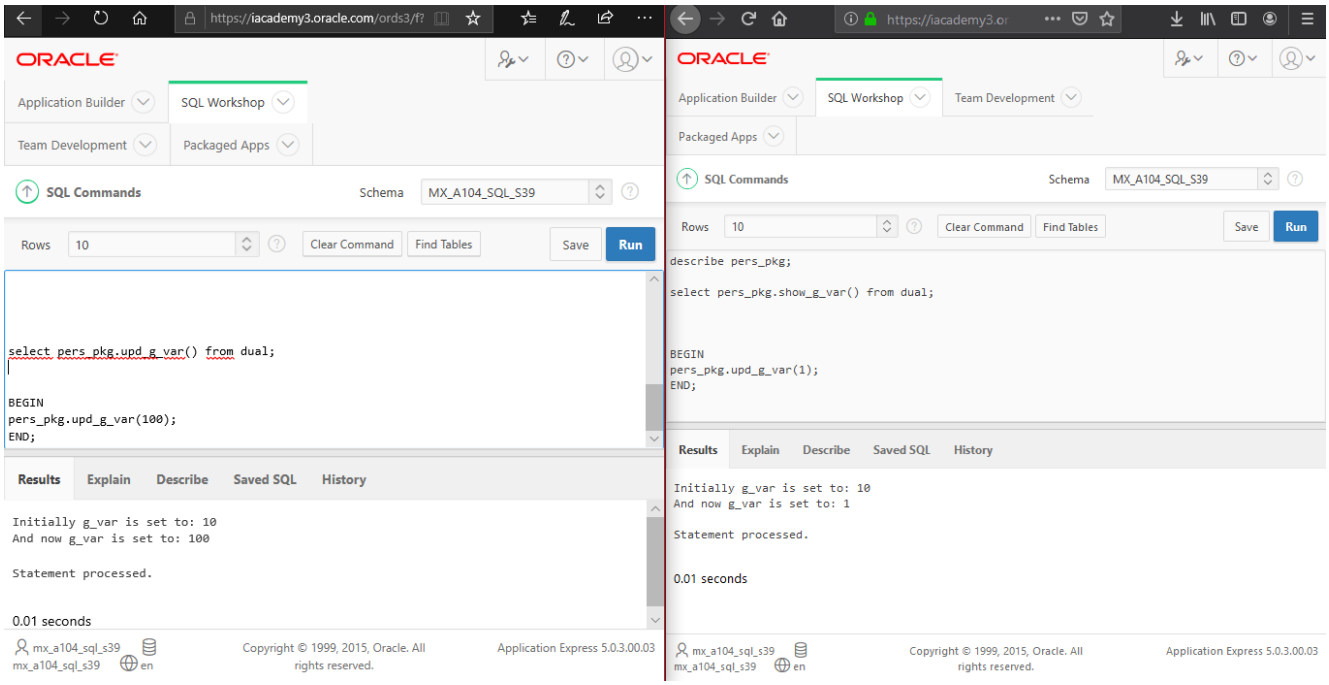
Then DESCRIBE your package in the other session to make sure that you can see it.



B. In both sessions, execute a SELECT statement that calls the show_g_var function to see the starting value of g_var. Verify that value 10 is returned in both sessions.



C. Now in the first session call the upd_g_var procedure with a value of 100 and in the second session call the upd_g_var procedure with a value of 1. Verify the results are as you expect: 100 in the first session and 1 in the second.



D. Since Oracle Academy's APEX automatically commits changes, unlike an installed/local APEX environment, we cannot actually see different values for the variable g_var in the different sessions outside of a PL/SQL block.

If we did not have AUTOCOMMIT, we could:

- in the first session, execute the upd_g_var procedure with a value of 50
- call the show_g_var function in the first session and it would display that g_var in that session has a value of 50
- call the show_g_var function in the second session and it would display that g_var in that session still has a value of 1

This is because each session has a separate copy of g_var and changes made in one session do not affect the other session.

2. Write a package called cursor_state that declares a global cursor as a join of EMPLOYEES and DEPARTMENTS. The cursor should select every employee's first and last name, department name, and the employee's salary. The package should also contain three public procedures: the first one opens the cursor; the second one has an IN parameter of type NUMBER and fetches and displays a number of rows as well as the current value of the loop counter. The third procedure closes the cursor. Remember to test the state of the cursor before you try to open or close it within each procedure.

```
CREATE OR REPLACE PACKAGE cursor_state IS

CURSOR c_global IS SELECT E.first_name, E.last_name, D.department_name, E.salary FROM
EMPLOYEES E INNER JOIN DEPARTMENTS D ON (E.department_id=D.department_id);

 PROCEDURE abre_cursor();

PROCEDURE proc2(parametro1 IN NUMBER);

PROCEDURE cierra_cursor();

END;


CREATE OR REPLACE PACKAGE BODY cursor_state IS

CURSOR c_global IS SELECT E.first_name, E.last_name, D.department_name, E.salary FROM
EMPLOYEES E INNER JOIN DEPARTMENTS D ON (E.department_id=D.department_id);

 PROCEDURE abre_cursor() as

BEGIN

Open c_global;

END;

PROCEDURE proc2(parametro1 IN NUMBER) AS

Record1 c_global;

BEGIN

Fetch c_global INTO record1;

Loop

Dbms_output.put_line(record1.first_name || record1.last_name);

Exit when SQL%NOT_FOUND;

End loop;

END;

PROCEDURE cierra_cursor() AS
```

```
BEGIN
Close c_global;
END;
END;
```

A. Test your code by executing an anonymous block that makes four calls to the package. The first call opens the cursor, the second fetches 3 rows, the third fetches 7 rows, and the fourth closes the cursor.

```
BEGIN
cursor_state.abre_cursor;
cursor_state.proc2(3);
cursor_state.proc2(7);
cursor_state.cierra_cursor;
END;
```

B. In the output, what is the source of the numbers 1, 2, 3, 1, 2, 3, 4, 5, 6, and 7? Explain why the first and fourth rows both have the number 1, yet have different employee names.