Database Programming with PL/SQL

14-1: Introduction to Dependencies Practice

Activities

Vocabulary

No new vocabulary for this lesson.

Try It / Solve It

1. The code for the check_dept function and the new_emp procedure is below. These subprograms are referenced in the following questions. You may recognize these subprograms from previous exercises. If you do not have these subprograms in your schema, then create them by copying and executing the code.

```
CREATE OR REPLACE PROCEDURE new_emp
        (p_empid        IN employees.employee_id%TYPE,
         p_fname        IN employees.first_name%TYPE,
         p_lname        IN employees.last_name%TYPE,
         p_email        IN employees.email%TYPE,
         p_hdate        IN employees.hire_date%TYPE,
         p_job          IN employees.job_id%TYPE,
         p_dept_id      IN employees.department_id%TYPE) IS
BEGIN
  IF check_dept(p_dept_id) THEN
        INSERT INTO employees (employee_id, first_name, last_name,
                               email, hire_date, job_id, department_id)
         VALUES (p_empid, p_fname, p_lname, p_email,p_hdate, p_job, p_dept_id);
  ELSE
        DBMS_OUTPUT.PUT_LINE('Invalid department number given, please try again.');
END IF;
```

```
END;

CREATE OR REPLACE FUNCTION check_dept
        (p_dept_in   IN departments.department_id%TYPE)
RETURN BOOLEAN IS
  v_dept              departments.department_id%TYPE;
BEGIN
  SELECT department_id INTO v_dept
FROM departments
    WHERE department_id = p_dept_in;
  RETURN TRUE;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN FALSE;
END;
```

A. In your own words, explain why the issue of dependencies is important. Think of the EMPLOYEES table, the new_emp procedure, and the check_dept function. Use them as examples in your explanation.

Es importante analizar la dependencia que existe entre los objetos.

Porque al realizar un ligero cambio, por ejemplo cambiar el nombre de algún objeto, esto ocasionaría que se deshabilitaran los objetos que dependen de ese objeto.

B. What Data Dictionary view would you use to check to see if a dependent object had its status invalidated? DESCRIBE the view in Application Express and retrieve the status of the following objects in your schema: EMPLOYEES table, the new_emp procedure, and the check_dept function.

SELECT * FROM USER_DEPENDENCIES WHERE REFERENCED_NAME IN('EMPLOYEES','NEW_EMP','CHECK_DEPT');

2. Think about the structure of the EMPLOYEES table. Review the structure of the new_emp procedure and the check_dept function. Which are dependent objects and which are referenced objects? What object is also involved that is not mentioned here and how is it related? Which dependencies are direct and which are indirect?

Dependent	Referenced	Direct/Indirect
INSERT_EMP	MX_A104_SQL_S38	HARD
NEW_EMP	MX_A104_SQL_S39	HARD

INSERT_EMP	MX_A104_SQL_S39	HARD

3. Which data dictionary view shows information about dependencies? DESCRIBE it. Then query it to retrieve the name and type of all objects that are dependent on the employees table. Sort your results by object type.

DESCRIBE ALL_DEPENDENCIES;

SELECT * FROM ALL_DEPENDENCIES WHERE REFERENCED_NAME IN('EMPLOYEES') ORDER BY TYPE;

4. Answer the following questions about the utldtree script:

   A. List the three steps involved in using the utldtree script.

Object_type
Object_owner
Object_name

   B. The utldtree script has already been run for you. Check this using DESCRIBE to see the details regarding the four objects that it has created.

DESCRIBE deptree;

   C. Populate the table which utldtree created with dependency information for your departments table.

BEGIN

DEPTREE_FILL('TABLE','MX_A104_SQL_S39','DEPARTMENTS');

END;

SELECT * FROM ideptree;

DEPENDENCIES
VIEW MX_A104_SQL_S39.EMP_DETAILS_VIEW
PACKAGE BODY MX_A104_SQL_S39.CHECK_EMP_PKG
FUNCTION MX_A104_SQL_S39.CHECK_DEPT
PROCEDURE MX_A104_SQL_S39.NEW_EMP
PROCEDURE MX_A104_SQL_S39.INS_NEW_DEPT
PROCEDURE MX_A104_SQL_S39.INSERT_EMP
PROCEDURE MX_A104_SQL_S39.GET_DEPARTMENTS
TABLE MX_A104_SQL_S39.DEPARTMENTS

8 rows returned in 0.21 seconds Download

D. Using the deptree view, display the information you populated in step C. Explain the values which are displayed in the nested_level column.

Results | Explain | Describe | Saved SQL | History

NESTED_LEVEL	TYPE	NAME
0	TABLE	DEPARTMENTS
1	VIEW	EMP_DETAILS_VIEW
1	FUNCTION	CHECK_DEPT
2	PROCEDURE	INSERT_EMP
2	PROCEDURE	NEW_EMP
1	PROCEDURE	INS_NEW_DEPT
1	PACKAGE BODY	CHECK_EMP_PKG
1	PROCEDURE	GET_DEPARTMENTS

8 rows returned in 0.18 seconds Download

Objetos dependientes de la tabla departments.

E. Now use the ideptree view to display the same information.

SELECT * FROM ideptree;

DEPENDENCIES
VIEW MX_A104_SQL_S39.EMP_DETAILS_VIEW
PACKAGE BODY MX_A104_SQL_S39.CHECK_EMP_PKG
FUNCTION MX_A104_SQL_S39.CHECK_DEPT
PROCEDURE MX_A104_SQL_S39.NEW_EMP
PROCEDURE MX_A104_SQL_S39.INS_NEW_DEPT
PROCEDURE MX_A104_SQL_S39.INSERT_EMP
PROCEDURE MX_A104_SQL_S39.GET_DEPARTMENTS
TABLE MX_A104_SQL_S39.DEPARTMENTS

8 rows returned in 0.21 seconds Download

5. Answer the following dependency questions:

   A. List four ways in which dependency failures can be minimized.

   B. Look again at the code of your new_emp procedure and your check_dept function. List two ways in which this code helps to minimize dependency failures.

6. Answer the following questions on the new_emp procedure:

   A. Modify the check_dept function so that the IN parameter p_dept_in is declared as NUMBER instead of departments.department_id%TYPE (this is not a good thing to do, but we'll do it in this question anyway). Recreate the function. Look at the status of the new_emp procedure in user_objects. What happened and why?

   B. State two ways in which the new_emp procedure could be recompiled. Would the recompilation be successful in this case? Recompile the procedure to check your answer.