Enamadi Sanath Yashwin
18CS01057

Assignment - 06

22.02.2021

Q1 OUTPUT

```
SQL> SELECT salary FROM employees WHERE employee_id = 130;

    SALARY
----------
       400

SQL>
SQL> DECLARE
  2  joe_cnt INT;
  3  new_salary employees.salary%TYPE;
  4  BEGIN
  5  -- check if any tuples exist with first_name being 'joe'
  6  SELECT COUNT(*) INTO joe_cnt FROM employees WHERE first_name = 'joe';
  7  IF joe_cnt > 0 THEN
  8  -- if yes, update the salary to min of all of them
  9  SELECT MIN(salary) INTO new_salary FROM employees WHERE first_name = 'joe';
 10  ELSE
 11  -- if no, update the salary to avg of all salaries
 12  SELECT AVG(salary) INTO new_salary FROM employees;
 13  END IF;
 14  UPDATE employees SET salary = new_salary WHERE employee_id = 130;
 15  END;
 16  /

PL/SQL procedure successfully completed.

SQL>
SQL> SELECT salary FROM employees WHERE employee_id = 130;

    SALARY
----------
       483
```

```
SQL> DECLARE
  2  efirst_name employees.first_name%TYPE;
  3  elast_name employees.last_name%TYPE;
  4  -- declare a cursor for the employees 5 and 10
  5  CURSOR c IS SELECT first_name, last_name FROM employees WHERE employee_id IN (540,980);
  6  BEGIN
  7  OPEN c;
  8  LOOP
  9  FETCH c INTO efirst_name, elast_name;
 10  EXIT WHEN c%NOTFOUND;
 11  -- display the names of employees in the cursor
 12  dbms_output.put_line(efirst_name||' '||elast_name);
 13  END LOOP;
 14  CLOSE c;
 15  END;
 16  /
Mike Tyson
Ben Gwalior
```

```
SQL> CREATE OR REPLACE FUNCTION departments_manager(departments_no IN INT)
  2  RETURN VARCHAR
  3  IS
  4  ename VARCHAR(64);
  5  efirst_name employees.first_name%TYPE;
  6  elast_name employees.last_name%TYPE;
  7  id employees.employee_id%TYPE;
  8  BEGIN
  9  -- select the manager_id attribute from the departments table for the given department
 10  SELECT manager_id INTO id FROM departments WHERE department_id = departments_no;
 11  -- get the details of the persom from the employees table using the previously selected manager_id attribute
 12  SELECT first_name, last_name INTO efirst_name, elast_name FROM employees WHERE employee_id = id;
 13  ename := efirst_name||' '||elast_name;
 14  RETURN ename;
 15  END;
 16  /

Function created.

SQL> DECLARE
  2  departments_no departments.department_id%TYPE;
  3  BEGIN
  4  departments_no := &departments_no;
  5  dbms_output.put_line(departments_manager(departments_no));
  6  END;
  7  /
Enter value for departments_no: 120
old   4: departments_no := &departments_no;
new   4: departments_no := 120;
James Musk

PL/SQL procedure successfully completed.
```

```
SQL> SELECT * FROM departments WHERE department_id = 120;

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID
------------- ------------------------------ ----------
          120 Project Management                    210

SQL> CREATE OR REPLACE PROCEDURE change_mngr(departments_no IN departments.department_id%TYPE)
  2  IS
  3  new_mngr employees.employee_id%TYPE;
  4  BEGIN
  5  -- assumption: only one person has the max salary
  6  -- select the employee_id with the max salary in the given department
  7  SELECT employee_id INTO new_mngr FROM employees WHERE
  8  salary = (SELECT MAX(salary) FROM employees WHERE department_id = departments_no);
  9  -- set the manager_id field of the departments table to the prviously selected employee_id
 10  UPDATE departments SET manager_id = new_mngr WHERE department_id = departments_no;
 11  END;
 12  /

Procedure created.

SQL> DECLARE
  2  departments_no departments.department_id%TYPE;
  3  BEGIN
  4  departments_no := &departments_no;
  5  change_mngr(departments_no);
  6  END;
  7  /
Enter value for departments_no: 120
old   4: departments_no := &departments_no;
new   4: departments_no := 120;

PL/SQL procedure successfully completed.

SQL> SELECT * FROM departments WHERE department_id = 120;

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID
------------- ------------------------------ ----------
          120 Project Management                    320
```

```
SQL> CREATE OR REPLACE TRIGGER decr_salary
  2  BEFORE UPDATE OF salary ON employees
  3  REFERENCING NEW AS n OLD AS o
  4  FOR EACH ROW
  5  BEGIN
  6  -- reject the update if the new salary is less than the old one
  7  IF :n.salary < :o.salary THEN
  8  RAISE_APPLICATION_ERROR('-20124', 'Salary is not allowed to be decremented');
  9  END IF;
 10  END;
 11  /

Trigger created.

SQL> SELECT * FROM employees WHERE employee_id = 130;

EMPLOYEE_ID FIRST_NAME                     LAST_NAME
----------- ------------------------------ ------------------------------
HIRE_DATE     JOB_ID     SALARY MANAGER_ID DEPARTMENT_ID
--------- ---------- ---------- ---------- -------------
        130 Andrew                         Pavlo
01-JAN-20        511        483        210           120


SQL> UPDATE employees SET salary = 400 WHERE employee_id = 130;
UPDATE employees SET salary = 400 WHERE employee_id = 130
       *
ERROR at line 1:
ORA-20124: Salary is not allowed to be decremented
ORA-06512: at "SYSTEM.DECR_SALARY", line 4
ORA-04088: error during execution of trigger 'SYSTEM.DECR_SALARY'
```

```
SQL>
SQL>
SQL> CREATE OR REPLACE TRIGGER validate_salary
  2  BEFORE UPDATE OF salary ON employees
  3  REFERENCING NEW AS n OLD AS o
  4  FOR EACH ROW
  5  DECLARE
  6  allowed_min jobs.min_salary%TYPE;
  7  allowed_max jobs.max_salary%TYPE;
  8  BEGIN
  9  -- get the limits for the coressponding job role
 10  SELECT min_salary, max_salary INTO allowed_min, allowed_max FROM jobs WHERE job_id = :n.job_id;
 11  -- check the if the updated salary lies in the range
 12  IF :n.salary < allowed_min OR :n.salary > allowed_max THEN
 13  RAISE_APPLICATION_ERROR('-20124', 'Salary not allowed');
 14  END IF;
 15  END;
 16  /

Trigger created.

SQL> SELECT * FROM employees WHERE employee_id = 130;

EMPLOYEE_ID FIRST_NAME                     LAST_NAME
----------- ------------------------------ ------------------------------
HIRE_DATE     JOB_ID     SALARY MANAGER_ID DEPARTMENT_ID
--------- ---------- ---------- ---------- -------------
       130 Andrew                         Pavlo
01-JAN-20        511        483        210           120


SQL> SELECT * FROM jobs WHERE job_id = 511;

    JOB_ID JOB_TITLE                      MIN_SALARY MAX_SALARY
---------- ------------------------------ ---------- ----------
       511 Software Engineer                     200        800

SQL> UPDATE employees SET salary = 900 WHERE employee_id = 130;
UPDATE employees SET salary = 900 WHERE employee_id = 130
       *
ERROR at line 1:
ORA-20124: Salary not allowed
ORA-06512: at "SYSTEM.VALIDATE_SALARY", line 9
ORA-04088: error during execution of trigger 'SYSTEM.VALIDATE_SALARY'
```

```
SQL> SELECT first_name, last_name, salary FROM employees WHERE employee_id IN
  2  (SELECT project_lead FROM projects WHERE start_date < DATE '1990-12-31' AND department_id = 120);

FIRST_NAME                     LAST_NAME                          SALARY
------------------------------ ------------------------------ ----------
Richard                        Jackson                               750
```

```
SQL> DECLARE
  2  user_date employees.hire_date%TYPE;
  3  user_dept employees.department_id%TYPE;
  4  efirst_name employees.first_name%TYPE;
  5  elast_name employees.last_name%TYPE;
  6  ejob_id jobs.job_id%TYPE;
  7  ejob_title jobs.job_title%TYPE;
  8  emngr_id employees.manager_id%TYPE;
  9  ehire_date employees.hire_date%TYPE;
 10  emngr_dept employees.department_id%TYPE;
 11  CURSOR c IS SELECT first_name, last_name, job_id, hire_date, manager_id FROM employees ;
 12  BEGIN
 13  user_date := date '1880-12-12';
 14  user_dept := &user_dept;
 15  OPEN c;
 16  LOOP
 17  -- fetch all details from the cursor
 18  FETCH c INTO efirst_name, elast_name, ejob_id, ehire_date, emngr_id;
 19  EXIT WHEN c%NOTFOUND;
 20  -- if the current emp has been hired after a given date
 21  IF ehire_date > user_date THEN
 22  -- get the department in which his manager works
 23  SELECT department_id INTO emngr_dept FROM employees WHERE employee_id = emngr_id;
 24  -- if that dept is same as the given dept
 25  IF emngr_dept = user_dept THEN
 26  SELECT job_title INTO ejob_title FROM jobs where job_id = ejob_id;
 27  dbms_output.put_line('Name: '||efirst_name||' '||elast_name||', '||'Job_title: '||ejob_title);
 28  END IF;
 29  END IF;
 30  END LOOP;
 31  CLOSE c;
 32  END;
 33  /
Enter value for user_dept: 120
old  14: user_dept := &user_dept;
new  14: user_dept := 120;
Name: Andrew Pavlo, Job_title: Software Engineer
Name: Jack Lite, Job_title: Application Engineer
Name: Joe Biden, Job_title: Software Engineer
Name: Ben Gwalior, Job_title: Software Engineer

PL/SQL procedure successfully completed.
```

```
SQL> CREATE OR REPLACE TRIGGER emp_change
  2  BEFORE UPDATE OF job_id, department_id ON employees
  3  REFERENCING NEW AS n OLD AS o
  4  FOR EACH ROW
  5  DECLARE
  6  cur_user VARCHAR(32);
  7  change_type CHAR(1);
  8  BEGIN
  9  -- get the current user of the database
 10  cur_user := USER;
 11  -- determine the type of change, department_change: D, job_change: J, both: B
 12  IF :n.job_id <> :o.job_id THEN change_type := 'J';
 13  ELSIF :n.department_id <> :o.department_id THEN change_type := 'D';
 14  ELSIF :n.department_id <> :o.department_id AND :n.job_id <> :o.job_id THEN change_type := 'B';
 15  ELSE change_type := 'N';
 16  END IF;
 17  -- accordingly modify the employment_change table
 18  INSERT INTO employment_change VALUES (:n.employee_id, :o.job_id, :n.job_id, :o.department_id,
 19  :n.department_id, SYSDATE, change_type, cur_user);
 20  END;
 21  /

Trigger created.

SQL> SELECT * FROM employees WHERE employee_id = 210;

EMPLOYEE_ID FIRST_NAME                     LAST_NAME
----------- ------------------------------ ------------------------------
HIRE_DATE     JOB_ID     SALARY MANAGER_ID DEPARTMENT_ID
--------- ---------- ---------- ---------- -------------
        210 James                          Musk
05-DEC-20        512        300        980           120


SQL> UPDATE employees SET job_id = 512 WHERE employee_id = 210;

1 row updated.

SQL> SELECT * FROM employment_change;

EMPLOYEE_ID OLD_JOB_ID NEW_JOB_ID OLD_DEPARTMENT_ID NEW_DEPARTMENT_ID CHANGE_DA
----------- ---------- ---------- ----------------- ----------------- ---------
T TUSER
- --------------------------------
        210        512        512               120               120 25-FEB-21
N SYSTEM
```

```
SQL> CREATE OR REPLACE TRIGGER emp_change
  2  BEFORE DELETE ON employees
  3  REFERENCING NEW AS n OLD AS o
  4  FOR EACH ROW
  5  BEGIN
  6  INSERT INTO exemployee VALUES (:o.employee_id, :o.first_name, :o.last_name, :o.department_id, SYSDATE);
  7  END;
  8  /

Trigger created.

SQL> DELETE FROM employees WHERE employee_id = 130;

1 row deleted.

SQL> SELECT * FROM exemployee;

EMPLOYEE_ID FIRST_NAME                     LAST_NAME
----------- ------------------------------ ------------------------------
DEPARTMENT_ID RELEASE_D
------------- ---------
        130 Andrew                         Pavlo
          120 25-FEB-21
```

```
SQL> SELECT employee_id, salary FROM employees WHERE department_id = 230;

EMPLOYEE_ID     SALARY
----------- ----------
        650        300
        760        450
        870        300
        980        750

SQL> CREATE OR REPLACE PROCEDURE incr_salary(departments_no IN departments.department_id%TYPE)
  2  IS
  3  employee_no employees.employee_id%TYPE;
  4  -- select all the employee_ids from the given department
  5  CURSOR c IS SELECT employee_id FROM employees WHERE department_id = departments_no;
  6  BEGIN
  7  OPEN c;
  8  LOOP
  9  FETCH c INTO employee_no;
 10  EXIT WHEN c%NOTFOUND;
 11  -- update salary for each member of the department
 12  UPDATE employees SET salary = salary+100 WHERE employee_id = employee_no;
 13  END LOOP;
 14  CLOSE c;
 15  END;
 16  /

Procedure created.

SQL> DECLARE
  2  departments_no departments.department_id%TYPE;
  3  BEGIN
  4  departments_no := &departments_no;
  5  incr_salary(departments_no);
  6  END;
  7  /
Enter value for departments_no: 230
old   4: departments_no := &departments_no;
new   4: departments_no := 230;
DECLARE
*
ERROR at line 1:
ORA-20124: Salary not allowed
ORA-06512: at "SYSTEM.VALIDATE_SALARY", line 9
ORA-04088: error during execution of trigger 'SYSTEM.VALIDATE_SALARY'
ORA-06512: at "SYSTEM.INCR_SALARY", line 12
ORA-06512: at line 5
```

```
SQL> SELECT employee_id, salary FROM employees WHERE department_id = 230;

EMPLOYEE_ID     SALARY
----------- ----------
        650        300
        760        450
        870        300
        980        750

SQL> CREATE OR REPLACE PROCEDURE incr_salary(departments_no IN departments.department_id%TYPE)
  2  IS
  3  employee_no employees.employee_id%TYPE;
  4  -- select all the employee_ids from the given department
  5  CURSOR c IS SELECT employee_id FROM employees WHERE department_id = departments_no;
  6  BEGIN
  7  OPEN c;
  8  LOOP
  9  FETCH c INTO employee_no;
 10  EXIT WHEN c%NOTFOUND;
 11  -- update salary for each member of the department
 12  UPDATE employees SET salary = salary+20 WHERE employee_id = employee_no;
 13  END LOOP;
 14  CLOSE c;
 15  END;
 16  /

Procedure created.

SQL>
SQL> DECLARE
  2  departments_no departments.department_id%TYPE;
  3  BEGIN
  4  departments_no := &departments_no;
  5  incr_salary(departments_no);
  6  END;
  7  /
Enter value for departments_no: 230
old   4: departments_no := &departments_no;
new   4: departments_no := 230;

PL/SQL procedure successfully completed.

SQL> SELECT employee_id, salary FROM employees WHERE department_id = 230;

EMPLOYEE_ID     SALARY
----------- ----------
        650        320
        760        470
        870        320
        980        770
```