PL/SQL 6-5

Agustín Alejandro Mota Hinojosa

November 26, 2023

Contents

1 Vocabulary

FOR UPDATE Declares that each row is locked as it is being fetched so other users cannot modify the rows while the cursor is open NOWAIT A keyword used to tell the Oracle server not to wait if the requested rows have already been locked by another user

2 Try It / Solve It

In this Practice you will INSERT and later UPDATE rows in a new table: $\text{PROPOSED}_{\text{RAISES}}$, which will store details of salary increases proposed for suitable employees. Create this table by executing the following SQL statement:

```
CREATE TABLE proposed_raises
(date_proposed DATE,
date_approved DATE,
employee_id NUMBER(6),
department_id NUMBER(4),
original_salary NUMBER(8,2),
proposed_new_salary NUMBER(8,2));
```

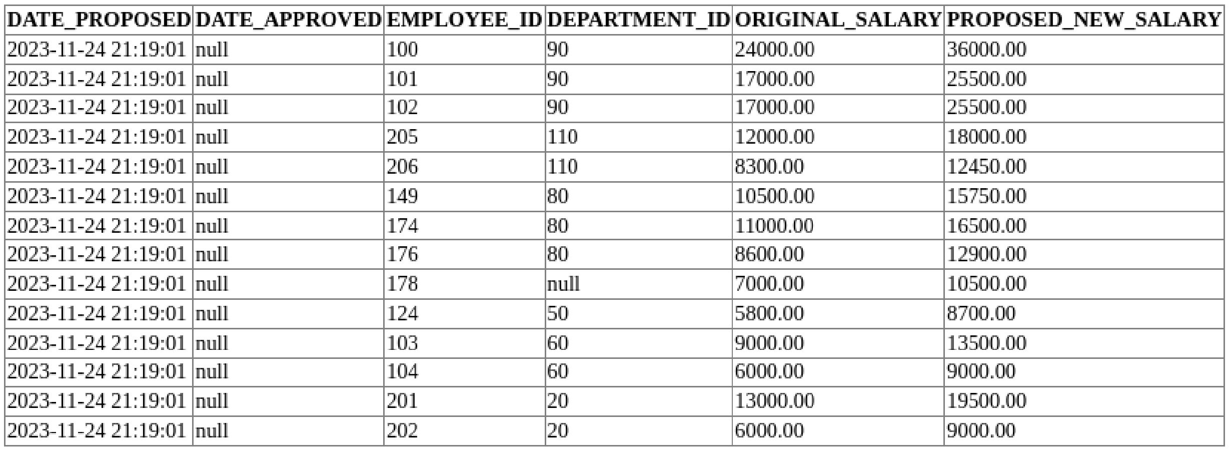
1. Write a PL/SQL block that inserts a row into $\text{PROPOSED}_{\text{RAISES}}$ for each eligible employee. The eligible employees are those whose salary is below a chosen value. The salary value is passed as a parameter to the cursor. For each eligible employee, insert a row into $\text{PROPOSED}_{\text{RAISES}}$ with $\text{date}_{\text{proposed}}$ = today's date, $\text{date}_{\text{appoved}}$ null, and $\text{proposed}_{\text{newsalary}}$ 5% greater than the current salary. The cursor should LOCK the employees rows so that no one can modify the employee data while the cursor is open. Test your code using a chosen salary value of 5000.

```
DECLARE
    CURSOR cur_emp (chosen_salary NUMBER) IS (
        SELECT employee_id,salary,department_id FROM employees WHERE salary < chosen_salary
    );
    v_value NUMBER;
BEGIN
    v_value := 5000;
    FOR rec IN cur_emp(v_value) LOOP
        INSERT INTO proposed_raises
```

```
            (date_proposed, date_approved, employee_id, department_id,
                            original_salary, proposed_new_salary)
        VALUES
            (SYSDATE,null,rec.employee_id,rec.department_id,rec.salary,rec.salary * 1.5);
    END LOOP;
END;
```

2. SELECT from the PROPOSED$_{\text{RAISES}}$ table to see the results of your INSERT statements. There should be 15 rows. If you run your block in question 1 more than once, make sure the PROPOSED$_{\text{RAISES}}$ table is empty before each test.

```
SELECT * FROM proposed_raises;
DELETE FROM proposed_raises; -- to clear all rows from the table
```

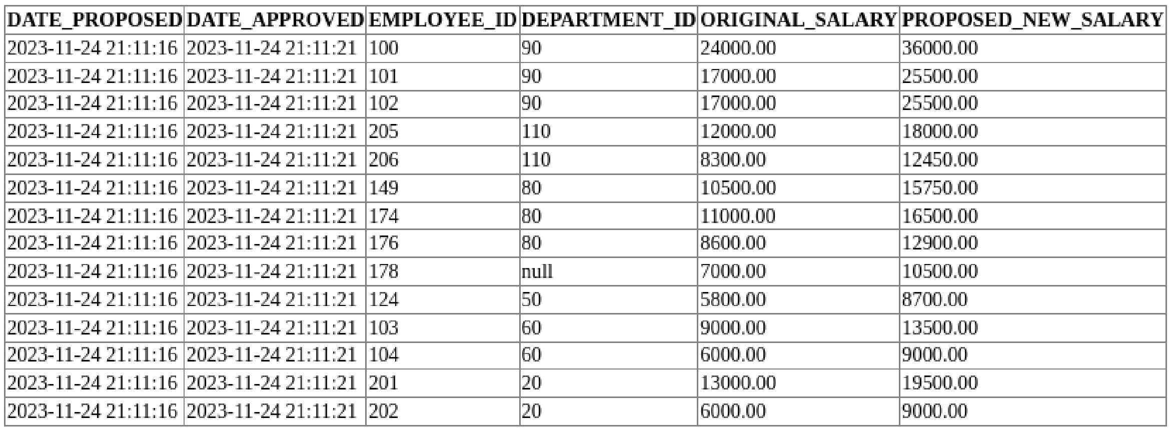


DATE_PROPOSED	DATE_APPROVED	EMPLOYEE_ID	DEPARTMENT_ID	ORIGINAL_SALARY	PROPOSED_NEW_SALARY
2023-11-24 21:19:01	null	100	90	24000.00	36000.00
2023-11-24 21:19:01	null	101	90	17000.00	25500.00
2023-11-24 21:19:01	null	102	90	17000.00	25500.00
2023-11-24 21:19:01	null	205	110	12000.00	18000.00
2023-11-24 21:19:01	null	206	110	8300.00	12450.00
2023-11-24 21:19:01	null	149	80	10500.00	15750.00
2023-11-24 21:19:01	null	174	80	11000.00	16500.00
2023-11-24 21:19:01	null	176	80	8600.00	12900.00
2023-11-24 21:19:01	null	178	null	7000.00	10500.00
2023-11-24 21:19:01	null	124	50	5800.00	8700.00
2023-11-24 21:19:01	null	103	60	9000.00	13500.00
2023-11-24 21:19:01	null	104	60	6000.00	9000.00
2023-11-24 21:19:01	null	201	20	13000.00	19500.00
2023-11-24 21:19:01	null	202	20	6000.00	9000.00

3. Imagine these proposed salary increases have been approved by company management.

   (a) Write and execute a PL/SQL block to read each row from the `PROPOSED_RAISES` table. For each row, UPDATE the `date_approved` column with today's date. Use the WHERE CURRENT OF... syntax to UPDATE each row. After running your code, SELECT from the `PROPOSED_RAISES` table to view the updated data.

```
DECLARE
    CURSOR cur_emp IS
        SELECT * FROM proposed_raises FOR UPDATE NOWAIT;
    v_emp_rec cur_emp%ROWTYPE;
BEGIN
    OPEN cur_emp;
    LOOP
        FETCH cur_emp INTO v_emp_rec;
        EXIT WHEN cur_emp%NOTFOUND;
        UPDATE proposed_raises
            SET date_approved = SYSDATE
        WHERE CURRENT OF cur_emp;
    END LOOP;
    CLOSE cur_emp;
END;
```



DATE_PROPOSED	DATE_APPROVED	EMPLOYEE_ID	DEPARTMENT_ID	ORIGINAL_SALARY	PROPOSED_NEW_SALARY
2023-11-24 21:11:16	2023-11-24 21:11:21	100	90	24000.00	36000.00
2023-11-24 21:11:16	2023-11-24 21:11:21	101	90	17000.00	25500.00
2023-11-24 21:11:16	2023-11-24 21:11:21	102	90	17000.00	25500.00
2023-11-24 21:11:16	2023-11-24 21:11:21	205	110	12000.00	18000.00
2023-11-24 21:11:16	2023-11-24 21:11:21	206	110	8300.00	12450.00
2023-11-24 21:11:16	2023-11-24 21:11:21	149	80	10500.00	15750.00
2023-11-24 21:11:16	2023-11-24 21:11:21	174	80	11000.00	16500.00
2023-11-24 21:11:16	2023-11-24 21:11:21	176	80	8600.00	12900.00
2023-11-24 21:11:16	2023-11-24 21:11:21	178	null	7000.00	10500.00
2023-11-24 21:11:16	2023-11-24 21:11:21	124	50	5800.00	8700.00
2023-11-24 21:11:16	2023-11-24 21:11:21	103	60	9000.00	13500.00
2023-11-24 21:11:16	2023-11-24 21:11:21	104	60	6000.00	9000.00
2023-11-24 21:11:16	2023-11-24 21:11:21	201	20	13000.00	19500.00
2023-11-24 21:11:16	2023-11-24 21:11:21	202	20	6000.00	9000.00

(b) Management has now decided that employees in department 50 cannot have a salary increase after all. Modify your code from question 3 to DELETE employees in department 50 from `PROPOSED_RAISES`. This could be done by a simple DML statement (DELETE FROM `proposed_raises` WHERE `department_id` = 50;), but we want to do it using a FOR UPDATE cursor. Test your code, and view the `PROPOSED_RAISES` table again to check that the rows have been deleted.

```
DECLARE
    CURSOR cur_emp IS
        SELECT *
            FROM proposed_raises
        WHERE department_id = 50 FOR UPDATE NOWAIT;
    v_emp_rec cur_emp%ROWTYPE;
BEGIN
    OPEN cur_emp;
    LOOP
        FETCH cur_emp INTO v_emp_rec;
        EXIT WHEN cur_emp%NOTFOUND;
        DELETE FROm proposed_raises WHERE CURRENT OF cur_emp;
    END LOOP;
    CLOSE cur_emp;
END;
```