Database Programming with PL/SQL

6-5: Using Cursors FOR UPDATE

Practice Activities

Vocabulary

Identify the vocabulary word for each definition below:

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

Try It / Solve It

In this Practice you will INSERT and later UPDATE rows in a new table: PROPOSED_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 PROPOSED_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 PROPOSED_RAISES with date_proposed = today's date, date_appoved null, and proposed_new_salary 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(p_salario NUMBER) IS SELECT salary,EMPLOYEE_ID,DEPARTMENT_ID
FROM EMPLOYEES where salary < p_salario FOR UPDATE NOWAIT;

v_emp_record cur_emp%ROWTYPE;

BEGIN

open cur_emp(5000);

LOOP

fetch cur_emp INTO v_emp_record;

EXIT WHEN CUR_EMP%NOTFOUND;

INSERT INTO PROPOSED_RAISES(DATE_PROPOSED, DATE_APPROVED,EMPLOYEE_ID,
DEPARTMENT_ID, ORIGINAL_SALARY, PROPOSED_NEW_SALARY)
VALUES(SYSDATE,NULL,V_EMP_RECORD.EMPLOYEE_ID,V_EMP_RECORD.DEPARTMENT_ID
,V_EMP_RECORD.SALARY,v_emp_record.salary*.05 +v_emp_record.salary );

END LOOP;

END;
```

2. SELECT from the PROPOSED_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_RAISES table is empty before each test.

```
SELECT * FROM proposed_raises;

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

Before continuing, ensure there are 15 rows in PROPOSED_RAISES.

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 DATE_APPROVED FROM PROPOSED_RAISES FOR
UPDATE NOWAIT;
v_emp_record cur_emp%ROWTYPE;
BEGIN
open cur_emp;
LOOP
fetch cur_emp INTO v_emp_record;
EXIT WHEN cur_emp%NOTFOUND;
UPDATE PROPOSED_RAISES
SET date_approved = SYSDATE
WHERE CURRENT OF cur_emp;
END LOOP;
CLOSE cur_emp;
END;
```

   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 DATE_APPROVED FROM PROPOSED_RAISES
WHERE DEPARTMENT_ID = 50 FOR UPDATE NOWAIT;
v_emp_record cur_emp%ROWTYPE;
BEGIN
open cur_emp;
LOOP
fetch cur_emp INTO v_emp_record;
```

```
EXIT WHEN cur_emp%NOTFOUND;
DELETE PROPOSED_RAISES
WHERE CURRENT OF cur_emp;
END LOOP;
CLOSE cur_emp;
END;
```

4. Since Oracle Academy's Application Express automatically commits changes, complete the following activity as if you were issuing the commands in an installed/local environment with the ability to use COMMIT and ROLLBACK. The indicated errors and pauses will not actually happen in the Oracle Academy's online Application Express.

   We are going to set up two sessions into the same schema. From one of the sessions we will manually update an employee row *NOT COMMITING*. From the other session we will try to update everyone's salary, again *NOT COMMITING*. You should see the difference between NOWAIT and WAIT when using FOR UPDATE.

   In preparation, create a copy of the employees table by executing the following SQL statement. You should use the UPD_EMPS table for the rest of this exercise.

```
CREATE TABLE upd_emps AS SELECT * FROM employees;
```

   A. Open a second session in a new browser window and connect to your schema.

   B. In your first session, update employee_id 200 (Jennifer Whalen) so the stored first name is Jenny. *DO NOT COMMIT.* You now have a lock on row 200 that will last indefinitely.

   C. In your second session, write a PL/SQL block to give every employee in UPD_EMPS a $1 salary raise. Your cursor should be declared FOR UPDATE NOWAIT. Execute your code. What happens?

   D. Still in your second session, modify your block to remove the NOWAIT attribute from the cursor declaration. Re-execute the block. What happens this time?

   E. After waiting a minute or so, switch to your first session and COMMIT the update to

Jennifer Whalen’s row. Then switch back to your second session. What happened?