
ORACLE
Academy

Database Programming with PL/SQL

5-5

Using Cursors for Update

ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Create PL/SQL code to lock rows before an update using the appropriate clause
  - Explain the effect of using NOWAIT in an update cursor declaration
  - Create PL/SQL code to use the current row of the cursor in an UPDATE or DELETE statement

If using the online/hosted Application Express at iAcademy, statements are automatically committed upon completion (unlike a typical production database where a deliberate COMMIT statement must be issued to actually modify data stored in the database). Therefore, all your DML statements will be committed upon completion (thereby releasing all locks held), and you will not be able to demonstrate locking and lock waits.

Purpose

- If multiple users are connected to the database at the same time, the possibility exists that another user updated the rows of a particular table after you opened your cursor and fetched the rows
- We can lock rows as we open the cursor in order to prevent other users from updating them – this is important especially if we are going to do updates ourselves
- An open cursor provides a read-consistent view of the data fetched by the cursor
- This means that any updates made by other users since the cursor was opened will not be seen when we fetch the rows, even if the updates were committed
- Our session would have to close and re-open the cursor in order to see the committed updates

Declaring a Cursor with the FOR UPDATE Syntax

- When we declare a cursor FOR UPDATE, each row is locked as we open the cursor
- This prevents other users from modifying the rows while our cursor is open
- It also allows us to modify the rows ourselves using a … WHERE CURRENT OF … clause

```
CURSOR cursor_name IS
  SELECT      ...   FROM ...
    FOR UPDATE [OF column_reference][NOWAIT | WAIT n];
```

- This does not prevent other users from reading the rows

Declaring a Cursor with the FOR UPDATE Clause

- column_reference is a column in the table whose rows we need to lock

```
CURSOR cursor_name IS
  SELECT      ... FROM ...
    FOR UPDATE [OF column_reference][NOWAIT | WAIT n];
```

- If the rows have already been locked by another session:
  - NOWAIT returns an Oracle server error immediately
  - WAIT n waits for n seconds, and returns an Oracle server error if the other session is still locking the rows at the end of that time

NOWAIT Keyword in the FOR UPDATE Clause Example

- The optional NOWAIT keyword tells the Oracle server not to wait if any of the requested rows have already been locked by another user
- Control is immediately returned to your program so that it can do other work before trying again to acquire the lock
- If you omit the NOWAIT keyword, then the Oracle server waits indefinitely until the rows are available

```
DECLARE
  CURSOR cur_emps IS
    SELECT employee_id, last_name FROM employees
      WHERE department_id = 80 FOR UPDATE NOWAIT;
  ...
```

NOWAIT Keyword in the FOR UPDATE Clause

- If the rows are already locked by another session and you have specified NOWAIT, then opening the cursor will result in an error
- You can try to open the cursor later
- You can use WAIT n instead of NOWAIT and specify the number of seconds to wait and check whether the rows are unlocked
- If the rows are still locked after n seconds, then an error is returned

It may help to think of a lock as being like a red traffic light. If another session has already locked the rows, the traffic light is red. Using the default, our session is an infinitely patient driver who will wait indefinitely until the traffic light turns green (i.e. the locks are released). If we use NOWAIT, we have no patience at all; we see the red light and immediately back off.

FOR UPDATE OF column-name Example

- If the cursor is based on a join of two tables, we may want to lock the rows of one table but not the other
- To do this, we specify any column of the table we want to lock

```
DECLARE
  CURSOR emp_cursor IS
    SELECT e.employee_id, d.department_name
      FROM  employees e, departments d
      WHERE e.department_id = d.department_id
      AND department_id = 80 FOR UPDATE OF salary;
  ...
```


If we don't specify a column-name, then rows of both tables are locked. This causes unnecessary extra locking when (in this example) we want to lock only the EMPLOYEES rows, not the DEPARTMENTS rows.

It doesn't matter which column-name we use. Individual columns are never locked, only whole rows.

WHERE CURRENT OF Clause Syntax

- The WHERE CURRENT OF clause is used in conjunction with the FOR UPDATE clause to refer to the current row (the most recently FETCHed row) in an explicit cursor
- The WHERE CURRENT OF clause is used in the UPDATE or DELETE statement, whereas the FOR UPDATE clause is specified in the cursor declaration

```
WHERE CURRENT OF cursor-name;
```

- cursor_name Is the name of a declared cursor (The cursor must have been declared with the FOR UPDATE clause)

WHERE CURRENT OF Clause

- You can use WHERE CURRENT OF for updating or deleting the current row from the corresponding database table
- This enables you to apply updates and deletes to the row currently being addressed, without the need to use a WHERE clause
- You must include the FOR UPDATE clause in the cursor query so that the rows are locked on OPEN

```
WHERE CURRENT OF cursor-name;
```

WHERE CURRENT OF Clause Example

- Use cursors to update or delete the current row
- Include the FOR UPDATE clause in the cursor query to lock the rows first
- Use the WHERE CURRENT OF clause to reference the current row from an explicit cursor

```
UPDATE employees
   SET salary = ...
   WHERE CURRENT OF cur_emps;
```

NOWAIT, FOR UPDATE, and WHERE CURRENT OF Clause

- In this example, we don't need a column-reference in the FOR UPDATE clause because the cursor is not based on a join

```
DECLARE
  CURSOR cur_emps IS
    SELECT employee_id, salary FROM my_employees
      WHERE salary <= 20000 FOR UPDATE NOWAIT;
  v_emp_rec cur_emps%ROWTYPE;
BEGIN
 OPEN cur_emps;
 LOOP
   FETCH cur_emps INTO v_emp_rec;
   EXIT WHEN cur_emps%NOTFOUND;
   UPDATE my_employees
     SET salary = v_emp_rec.salary*1.1
     WHERE CURRENT OF cur_emps;
 END LOOP;
 CLOSE cur_emps;
END;
```

FOR UPDATE Second Example

- FOR UPDATE OF salary locks only the MY_EMPLOYEES rows, not the MY_DEPARTMENTS rows
- Note that we update the table-name, not the cursor-name!

```
DECLARE
  CURSOR cur_eds IS
    SELECT employee_id, salary, department_name
      FROM my_employees e, my_departments d
      WHERE e.department_id = d.department_id
      FOR UPDATE OF salary NOWAIT;
BEGIN
  FOR v_eds_rec IN cur_eds LOOP
    UPDATE my_employees
      SET salary = v_eds_rec.salary * 1.1
      WHERE CURRENT OF cur_eds;
  END LOOP;
END;
```

Terminology

- Key terms used in this lesson included:
  - FOR UPDATE
  - NOWAIT

ORACLE Academy

- FOR UPDATE – Declares that each row is locked as it is being fetched so other users can not 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.

Summary

- In this lesson, you should have learned how to:
  - Create PL/SQL code to lock rows before an update using the appropriate clause
  - Explain the effect of using NOWAIT in an update cursor declaration
  - Create PL/SQL code to use the current row of the cursor in an UPDATE or DELETE statement


ORACLE
Academy