Using Explicit Cursors

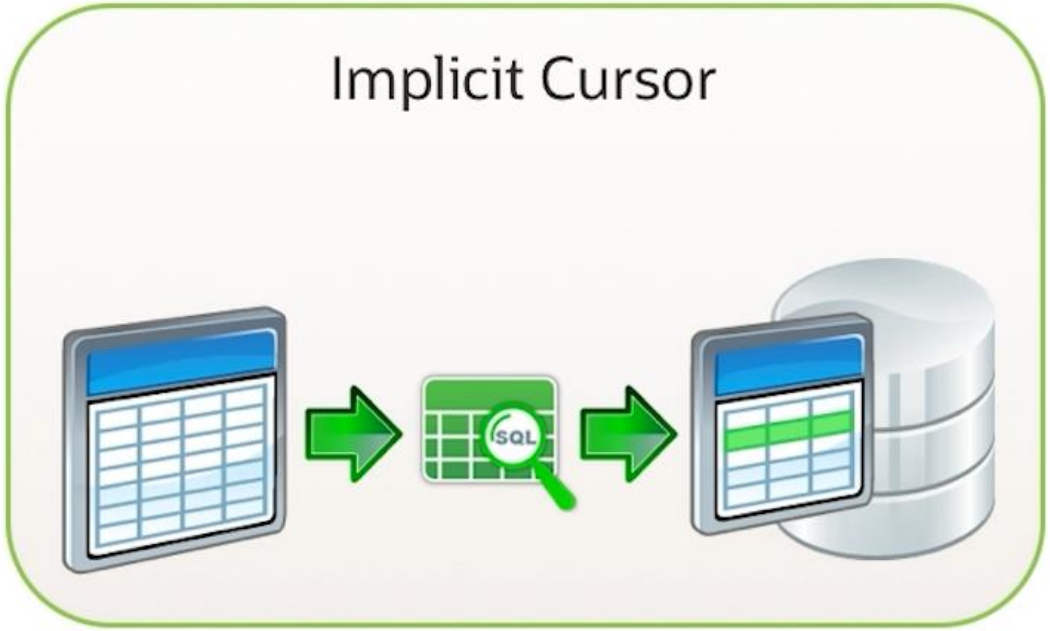
Objectives

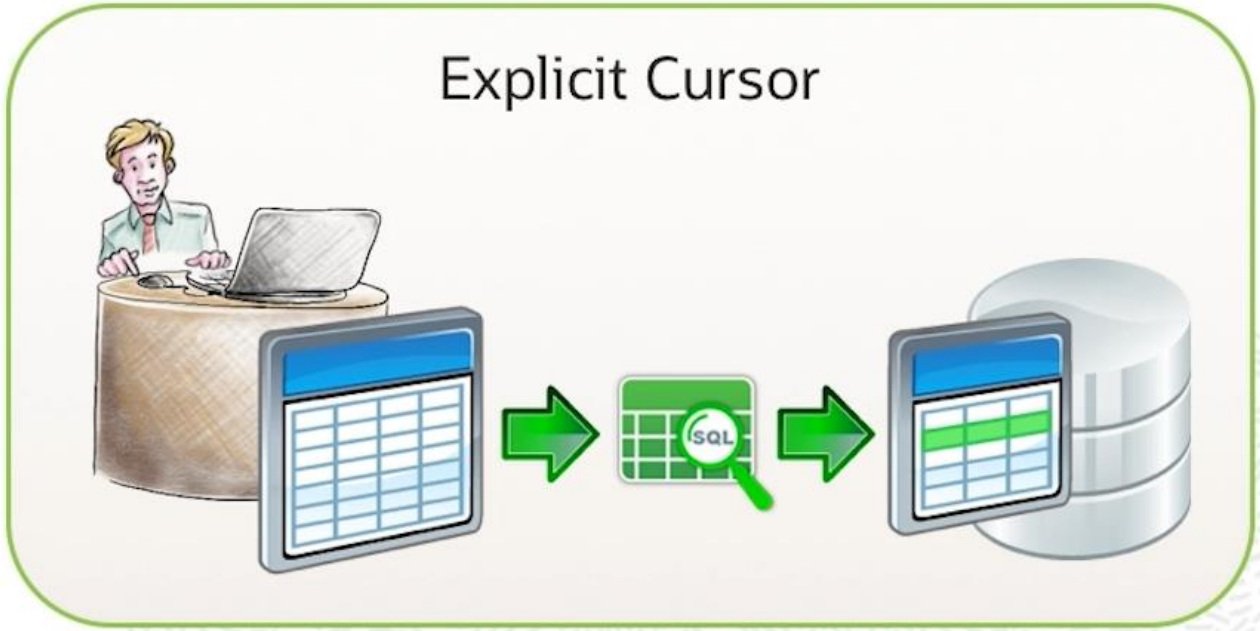
After completing this lesson, you should be able to do the following:

- Distinguish between implicit and explicit cursors
- Discuss the reasons for using explicit cursors
- Declare and control explicit cursors
- Use simple loops and cursor `FOR` loops to fetch data
- Declare and use cursors with parameters
- Lock rows with the `FOR UPDATE` clause
- Reference the current row with the `WHERE CURRENT OF` clause

Cursors

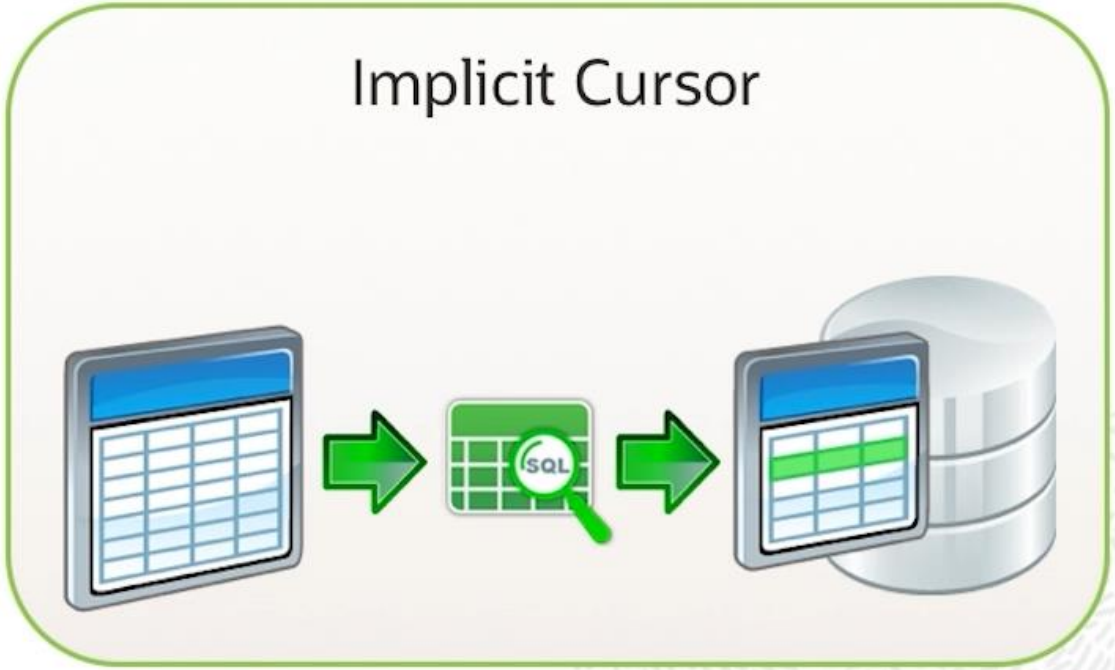
- A cursor is a pointer to a private memory area that is used to execute SQL statements in PL/SQL blocks.
- There are two type of cursors:
  - Implicit cursors
  - Explicit cursors





Implicit Cursors

- Are created and managed by PL/SQL
- Are created for `SELECT` or DML statements
- Closes after the associated statement is run



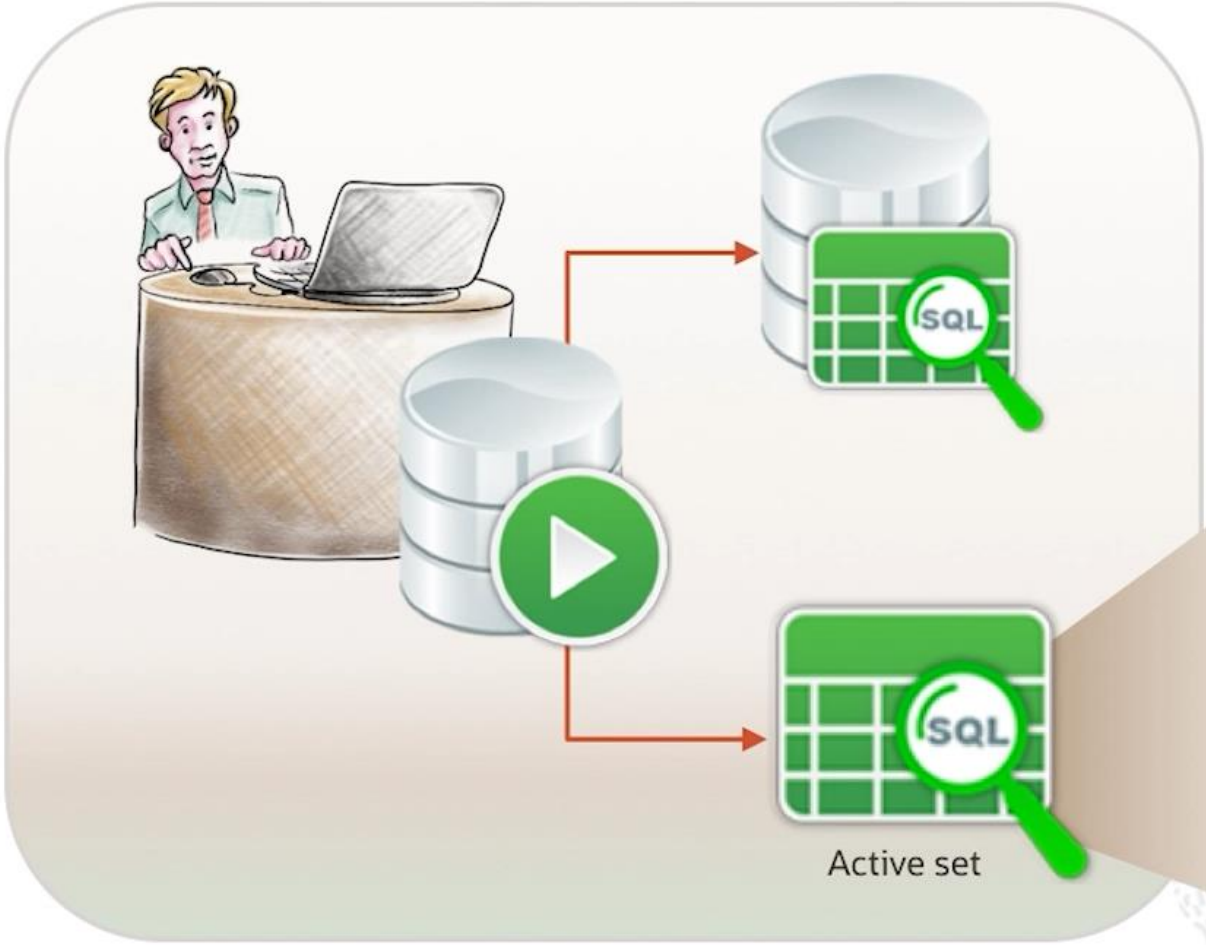
SQL Cursor Attributes for Implicit Cursors

Using SQL cursor attributes, you can test the outcome of your SQL statements.

SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement returned at least one row
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement did not return even one row
SQL%ROWCOUNT	An integer value that represents the number of rows affected by the most recent SQL statement

Explicit Cursor

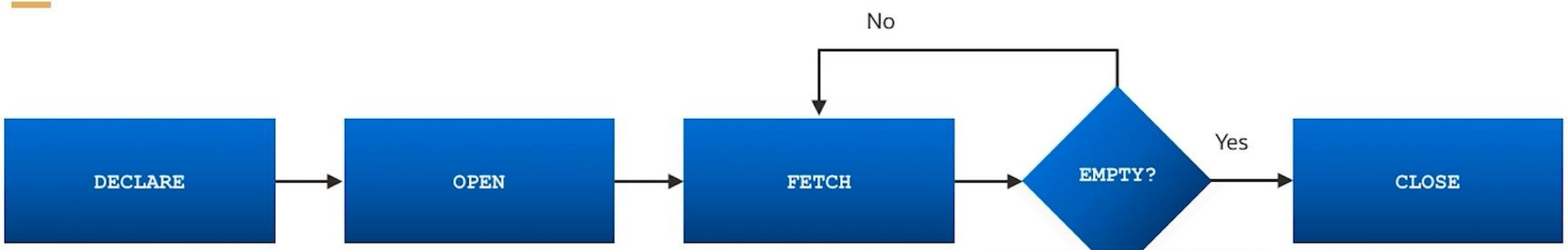
An explicit cursor is created and managed by a programmer in the PL/SQL block.



Table

100	King	AD_PRES
101	Kochhar	AD_VP
102	De Haan	AD_VP
.	.	.
.	.	.
.	.	.
139	Seo	ST_CLERK
140	Patel	ST_CLERK
.	.	.

Controlling Explicit Cursors



- Create a named SQL area.

- Identify the active set.

- Load the current row into variables.

- Test for existing rows.
- Return to `FETCH` if rows are found.

- Release the active set.

Declaring the Cursor

Syntax:

```
CURSOR cursor_name IS
     select_statement;
```

Examples:

```
DECLARE
  CURSOR c_emp_cursor IS
  SELECT employee_id, last_name FROM employees
  WHERE department_id =30;
```

```
DECLARE
  v_locid NUMBER:= 1700;
  CURSOR c_dept_cursor IS
  SELECT * FROM departments
  WHERE location_id = v_locid;
...
```

Opening the Cursor

```
DECLARE
  CURSOR c_emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
...
BEGIN
  OPEN c_emp_cursor;
```

Fetching Data from the Cursor

```
DECLARE
  CURSOR c_emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
  v_empno employees.employee_id%TYPE;
  v_lname employees.last_name%TYPE;
BEGIN
  OPEN c_emp_cursor;
  FETCH c_emp_cursor INTO v_empno, v_lname;
  DBMS_OUTPUT.PUT_LINE( v_empno ||'  '||v_lname);
END;
/
```

```
PL/SQL procedure successfully completed.

114   Raphaely
```

Fetching Data from the Cursor

```
DECLARE
  CURSOR c_emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
  v_empno employees.employee_id%TYPE;
  v_lname employees.last_name%TYPE;
BEGIN
  OPEN c_emp_cursor;
  LOOP
    FETCH c_emp_cursor INTO v_empno, v_lname;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_empno ||'  '||v_lname);
  END LOOP;
END;
/
```

Closing the Cursor

```
...
  LOOP
    FETCH c_emp_cursor INTO empno, lname;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_lname);
  END LOOP;
 CLOSE c_emp_cursor;
END;
/
```

Cursors and Records

Process the rows of the active set by fetching values into a PL/SQL record.

```
DECLARE
  CURSOR c_emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE  department_id =30;
   v_emp_record  c_emp_cursor%ROWTYPE;
BEGIN
  OPEN c_emp_cursor;
  LOOP
    FETCH c_emp_cursor INTO v_emp_record;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_emp_record.employee_id
                          ||' '||v_emp_record.last_name);
  END LOOP;
  CLOSE c_emp_cursor;
END;
```

```
1  DECLARE
2  CURSOR empcursor is select * from employees;
3  emprecord employees%ROWTYPE;
4  begin
5  OPEN empcursor;
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line(emprecord.last_name );
10 END LOOP;
11 end;
```

Script Output

Task completed in 0.057 seconds

```
Walsh
Feeney
OConnell
Grant
Whalen
Hartstein
Fay
Mavris
Baer
Higgins
Gietz


PL/SQL procedure successfully completed.
```

It is not mandatory to close the cursor, but if you run the cursor with different parameter, you will get an error that the cursor is already opened.

```
1  DECLARE
2  CURSOR empcursor is select * from employees;
3  emprecord employees%ROWTYPE;
4  begin
5  OPEN empcursor;
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line(emprecord.last_name );
10 END LOOP;
11 close empcursor;
12 end;
```

Script Output

Task completed in 0.057 seconds

```
Walsh
Feeney
OConnell
Grant
Whalen
Hartstein
Fay
Mavris
Baer
Higgins
Gietz


PL/SQL procedure successfully completed.
```

```
3  emprecord employees%ROWTYPE;
4  begin
5  OPEN empcursor;
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line(emprecord.last_name );
10 END LOOP;
11 OPEN empcursor;
12 LOOP
13 FETCH empcursor into emprecord;
14 exit when empcursor%notfound;
15 dbms_output.put_line(emprecord.last_name );
16 END LOOP;
17 end;
```

Script Output

Task completed in 0.053 seconds

```
END LOOP;
OPEN empcursor;
LOOP
FETCH empcursor into emprecord;
exit when empcursor%notfound;
dbms_output.put_line(emprecord.last_name );
END LOOP;
end;
Error report -
ORA-06511: PL/SQL: cursor already open
ORA-06512: at line 2
ORA-06512: at line 11
06511. 00000 -  "PL/SQL: cursor already open"
*Cause:    An attempt was made to open a cursor that was already open.
*Action:   Close cursor first before reopening.
```

Explicit cursor attribute :

CursorName%ISOPEN

Others explicit cursors attribute:
CursorName%FOUND;
CursorName%NotFound;

```
1  DECLARE
2  CURSOR empcursor is select * from employees;
3  emprecord employees%ROWTYPE;
4  begin
5  OPEN empcursor;
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line(emprecord.last_name );
10 END LOOP;
11 IF EMPCURSOR%ISOPEN then close empcursor; end if;
12 OPEN empcursor;
13 LOOP
14 FETCH empcursor into emprecord;
15 exit when empcursor%notfound;
16 dbms_output.put_line(emprecord.last_name );
17 END LOOP;
18 end;
```

```
1  DECLARE
2  CURSOR empcursor (pdept number) is select * from employees where department_id=pdept;
3  emprecord employees%ROWTYPE;
4  begin
5  OPEN empcursor(10);
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line(emprecord.last_name );
10 END LOOP;
11 IF EMPCURSOR%ISOPEN then close empcursor; end if;
12 OPEN empcursor(20);
13 LOOP
14 FETCH empcursor into emprecord;
15 exit when empcursor%notfound;
16 dbms_output.put_line(emprecord.last_name );
17 END LOOP;
18 end;
```

Cursors with parameters :

```
1  DECLARE
2  CURSOR empcursor (pdept number) is select * from employees where department_id=pdept;
3  emprecord employees%ROWTYPE;
4  begin
5  OPEN empcursor(10);
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line('10 '||emprecord.last_name );
10 END LOOP;
11 IF EMPCURSOR%ISOPEN then close empcursor; end if;
12 OPEN empcursor(20);
13 LOOP
14 FETCH empcursor into emprecord;
15 exit when empcursor%notfound;
16 dbms_output.put_line('20 '||emprecord.last_name );
17 END LOOP;
18 close empcursor;
19 end;
```

Script Output

Task completed in 0.046 seconds

```
10 Whalen
20 Hartstein
20 Fay


PL/SQL procedure successfully completed.
```

```
1  DECLARE
2  CURSOR empcursor (pdept number) is select last_name, salary from employees where department_id=pdept;
3  emprecord empcursor%ROWTYPE;
4  begin
5  OPEN empcursor(10);
6  LOOP
7  FETCH empcursor into emprecord;
8  exit when empcursor%notfound;
9  dbms_output.put_line('10 '||emprecord.last_name );
10 END LOOP;
11 IF EMPCURSOR%ISOPEN then close empcursor; end if;
12 OPEN empcursor(20);
13 LOOP
14 FETCH empcursor into emprecord;
15 exit when empcursor%notfound;
16 dbms_output.put_line('20 '||emprecord.last_name );
17 END LOOP;
18 close empcursor;
19 end;
```

Script Output

Task completed in 0.041 seconds

```
10 Whalen
20 Hartstein
20 Fay

PL/SQL procedure successfully completed.
```

Cursor FOR Loops

Syntax:

```
FOR record_name IN cursor_name LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- The cursor FOR loop is a shortcut to process explicit cursors.
- Implicit open, fetch, exit, and close occur.
- The record is implicitly declared.

Cursor FOR Loops

```
DECLARE
  CURSOR c_emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
BEGIN
   FOR emp_record IN c_emp_cursor
    LOOP
     DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
     ||' ' ||emp_record.last_name);
    END LOOP;
END;
/
```

```
PL/SQL procedure successfully completed.

114   Raphaely
115   Khoo
116   Baida
117   Tobias
118   Himuro
119   Colmenares
```

Explicit Cursor Attributes

Use explicit cursor attributes to obtain status information about a cursor.

Attribute	Type	Description
`%ISOPEN`	Boolean	Evaluates to TRUE if the cursor is open
`%NOTFOUND`	Boolean	Evaluates to TRUE if the most recent fetch does not return a row
`%FOUND`	Boolean	Evaluates to TRUE if the most recent fetch returns a row; complement of `%NOTFOUND`
`%ROWCOUNT`	Number	Evaluates to the number of rows that has been fetched

%ISOPEN Attribute

- You can fetch rows only when the cursor is open.
- Use the %ISOPEN cursor attribute before performing a fetch to test whether the cursor is open.

Example:

```
IF NOT c_emp_cursor%ISOPEN THEN
    OPEN c_emp_cursor;
END IF;
LOOP
  FETCH c_emp_cursor...
```

%ROWCOUNT and %NOTFOUND: Example

```
DECLARE
  CURSOR c_emp_cursor IS SELECT employee_id,
    last_name FROM employees;
  v_emp_record c_emp_cursor%ROWTYPE;
BEGIN
  OPEN c_emp_cursor;
  LOOP
   FETCH c_emp_cursor INTO v_emp_record;
   EXIT WHEN c_emp_cursor%ROWCOUNT > 10 OR
                    c_emp_cursor%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE( v_emp_record.employee_id
              ||' '||v_emp_record.last_name);
  END LOOP;
  CLOSE c_emp_cursor;
END ; /
```

```
PL/SQL procedure successfully completed.

174  Abel
166  Ande
130  Atkinson
105  Austin
204  Baer
116  Baida
167  Banda
172  Bates
192  Bell
151  Bernstein
```

Cursor FOR Loops Using Subqueries

There is no need to declare the cursor.

```
BEGIN
  FOR emp_record IN (SELECT employee_id,
last_name
   FROM employees WHERE department_id =30)
  LOOP
   DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
   ||' '||emp_record.last_name);
  END LOOP;
END;
/
```

```
PL/SQL procedure successfully completed.

114   Raphaely
115   Khoo
116   Baida
117   Tobias
118   Himuro
119   Colmenares
```

Cursors with Parameters

- Pass parameter values to a cursor when the cursor is opened and the query is executed.
- Open an explicit cursor several times with a different active set each time.

Syntax:

```
CURSOR cursor_name
  [(parameter_name datatype, ...)]
IS
  select_statement;
```

```
OPEN   cursor_name(parameter_value,.....) ;
```

Cursors with Parameters

```
DECLARE
  CURSOR c_emp_cursor (deptno NUMBER)
  IS
     SELECT employee_id, last_name FROM employees WHERE department_id = deptno;
  v_empno employees.employee_id%TYPE;
  v_lname employees.last_name%TYPE;
BEGIN
  OPEN c_emp_cursor (30);
  LOOP
    FETCH c_emp_cursor INTO v_empno, v_lname;
    EXIT  WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_lname);
  END LOOP;
  CLOSE c_emp_cursor;
  DBMS_OUTPUT.PUT_LINE('Opening the cursor the second time');
  OPEN c_emp_cursor (10);
  LOOP
    FETCH c_emp_cursor INTO v_empno, v_lname;
    EXIT  WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_lname);
  END LOOP;
  CLOSE c_emp_cursor;
END;
```

```
PL/SQL procedure successfully completed.

114   Raphaely
115   Khoo
116   Baida
117   Tobias
118   Himuro
119   Colmenares
Opening the cursor second time
200   Whalen
```

FOR UPDATE Clause

Syntax:

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

- Use explicit locking to deny access to other sessions for the duration of a transaction.
- Lock the rows *before* an update or a delete.

WHERE CURRENT OF Clause

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

Syntax:

```
WHERE CURRENT OF cursor ;
```

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

WHERE CURRENT OF Clause: Example

```
DECLARE
  CURSOR c_emp_cursor IS
   SELECT employee_id, salary FROM employees
   WHERE department_id =30 FOR UPDATE;
BEGIN
   FOR emp_record IN c_emp_cursor
    LOOP
     DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
     ||' ' ||emp_record.salary);
      UPDATE employees
      SET salary = 5000
      WHERE CURRENT OF c_emp_cursor;
    END LOOP;
END;
/
SELECT employee_id, salary FROM employees
WHERE department_id =30;
```

Quiz

Explicit cursor functions enable a programmer to manually control explicit cursors in the PL/SQL block.

a. True
b. False

Summary

In this lesson, you should have learned how to:

- Distinguish cursor types:
  - Implicit cursors are used for all DML statements and single-row queries.
  - Explicit cursors are used for queries of zero, one, or more rows.
- Create and handle explicit cursors
- Use simple loops and cursor `FOR` loops to handle multiple rows in the cursors
- Evaluate cursor status by using cursor attributes
- Use the `FOR UPDATE` and `WHERE CURRENT OF` clauses to update or delete the current fetched row

Practice 8: Overview

This practice covers the following topics:

- Declaring and using explicit cursors to query the rows of a table
- Using a cursor `FOR` loop
- Applying cursor attributes to test the cursor status
- Declaring and using cursors with parameters
- Using the `FOR UPDATE` and `WHERE CURRENT OF` clauses