



Using Explicit Cursors

Objectives

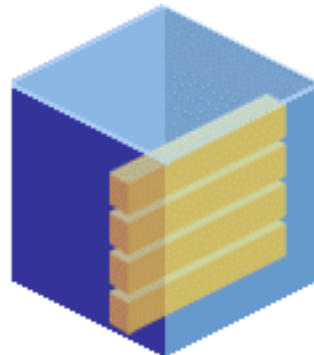
After completing this lesson, you should be able to do the following:

- 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

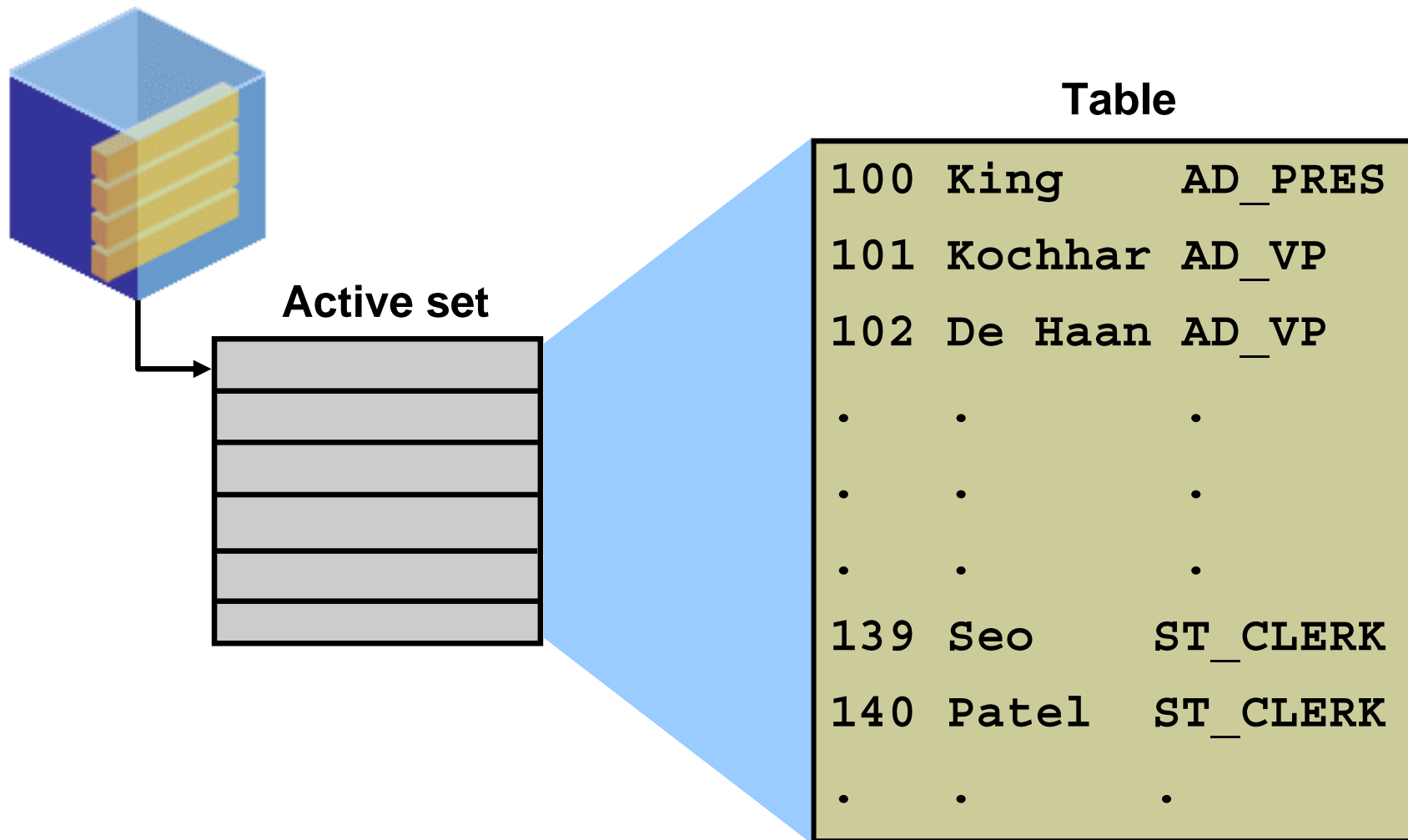
Cursors

Every SQL statement executed by the Oracle server has an associated individual cursor:

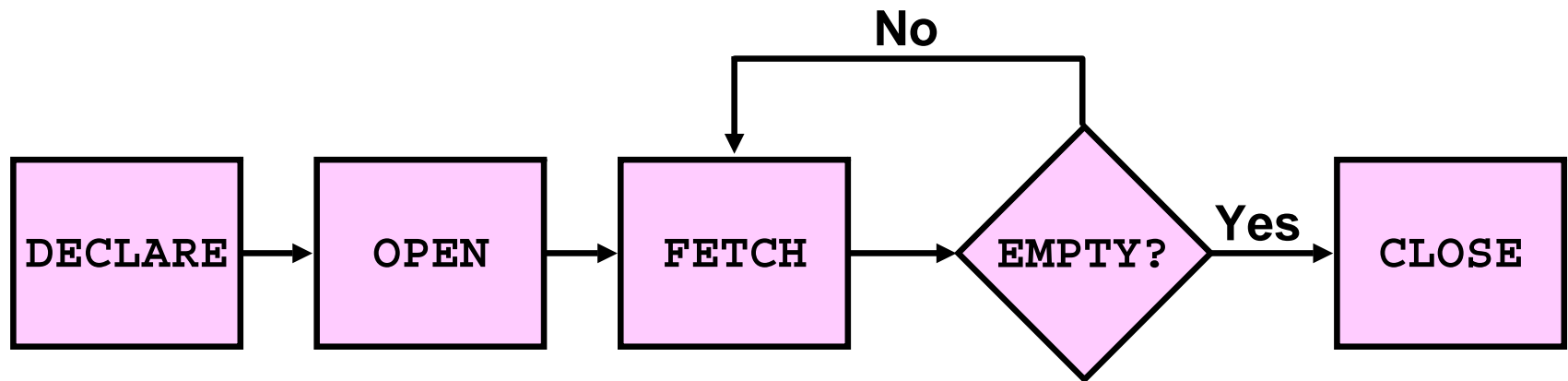
- Implicit cursors: Declared and managed by PL/SQL for all DML and PL/SQL SELECT statements
- Explicit cursors: Declared and managed by the programmer



Explicit Cursor Operations



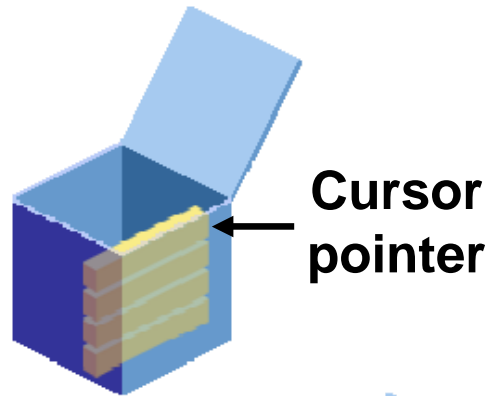
Controlling Explicit Cursors



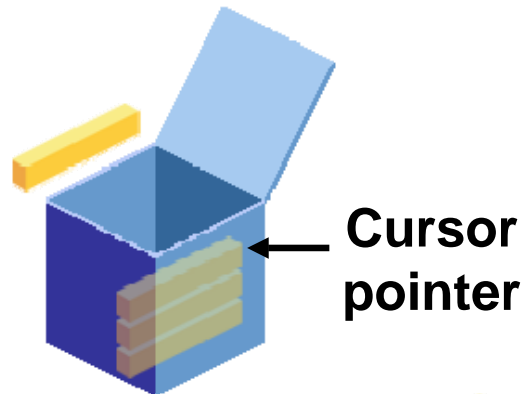
- Create a named SQL area.
- Identify the active set.
- Load the current row into variables.
- Test for existing rows.
 - Return to FETCH if rows are found.
- Release the active set.

Controlling Explicit Cursors

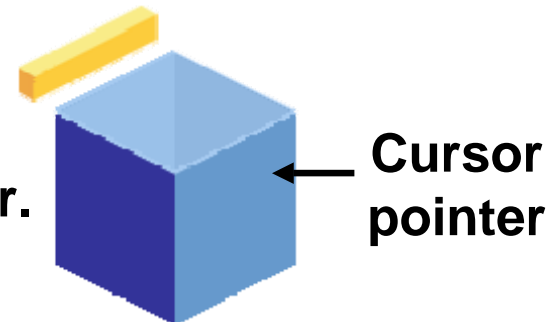
1 Open the cursor.



2 Fetch a row.



3 Close the cursor.



Declaring the Cursor

Syntax:

```
CURSOR cursor_name IS  
    select_statement;
```

Examples:

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

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

Opening the Cursor

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


Fetching Data from the Cursor

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name FROM employees
    WHERE department_id =30;
  empno employees.employee_id%TYPE;
  lname employees.last_name%TYPE;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO empno, lname;
  DBMS_OUTPUT.PUT_LINE( empno || ' ' || lname);
  ...
END;
/
```

Fetching Data from the Cursor

```
SET SERVEROUTPUT ON
DECLARE
    CURSOR emp_cursor IS
        SELECT employee_id, last_name FROM employees
        WHERE department_id =30;
    empno employees.employee_id%TYPE;
    lname employees.last_name%TYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO empno, lname;
        EXIT WHEN emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( empno || ' ' || lname);
    END LOOP;
    ...
END;
/
```

Closing the Cursor

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

Cursors and Records

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

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name FROM employees
    WHERE department_id =30;
  emp_record emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    ...
```

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

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

Explicit Cursor Attributes

Obtain status information about a cursor.

Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to <code>TRUE</code> if the cursor is open
%NOTFOUND	Boolean	Evaluates to <code>TRUE</code> if the most recent fetch does not return a row
%FOUND	Boolean	Evaluates to <code>TRUE</code> if the most recent fetch returns a row; complement of <code>%NOTFOUND</code>
%ROWCOUNT	Boolean	Evaluates to the total number of rows returned so far

%ISOPEN Attribute

- 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 emp_cursor%ISOPEN THEN
    OPEN emp_cursor;
END IF;
LOOP
    FETCH emp_cursor...
```


%ROWCOUNT and %NOTFOUND: Example

```
SET SERVEROUTPUT ON
DECLARE
    empno    employees.employee_id%TYPE;
    ename    employees.last_name%TYPE;
    CURSOR emp_cursor IS SELECT employee_id,
                                last_name FROM employees;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO empno, ename;
        EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                  emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (TO_CHAR (empno)
                               || ' ' || ename);
    END LOOP;
    CLOSE emp_cursor;
END ;
/
```

Cursor FOR Loops Using Subqueries

There is no need to declare the cursor.

Example:

```
SET SERVEROUTPUT ON
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;
/
```

Cursors with Parameters

Syntax:

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

- Pass the 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.

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

Cursors with Parameters

```
SET SERVEROUTPUT ON
DECLARE
    CURSOR    emp_cursor (deptno NUMBER) IS
        SELECT employee_id, last_name
        FROM    employees
        WHERE   department_id = deptno;
    dept_id NUMBER;
    lname     VARCHAR2(15);
BEGIN
    OPEN emp_cursor (10);
    ...
    CLOSE emp_cursor;
    OPEN emp_cursor (20);
    ...
```

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* the update or delete.

WHERE CURRENT OF Clause

Syntax:

```
WHERE CURRENT OF cursor ;
```

- 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 emp_cursor;
```

Cursors with Subqueries

Example:

```
DECLARE
  CURSOR my_cursor IS
    SELECT t1.department_id, t1.department_name,
           t2.staff
    FROM   departments t1, (SELECT department_id,
                                   COUNT(*) AS staff
                            FROM employees
                            GROUP BY department_id) t2
    WHERE  t1.department_id = t2.department_id
    AND    t2.staff >= 3;
...

```

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 the cursor status by using the cursor attributes
- Use the `FOR UPDATE` and `WHERE CURRENT OF` clauses to update or delete the current fetched row

Practice 7: Overview

This practice covers the following topics:

- Declaring and using explicit cursors to query 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