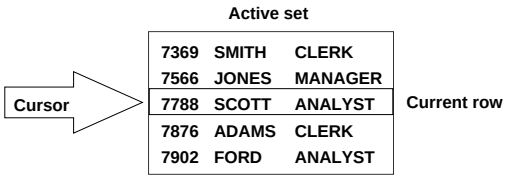
# Writing Explicit Cursors

## About Cursors

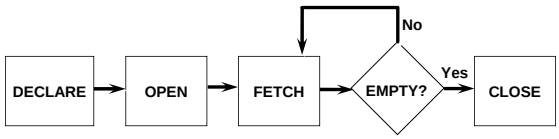
Every SQL statement executed by the Oracle Server has an individual cursor associated with it:

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

## Explicit Cursor Functions

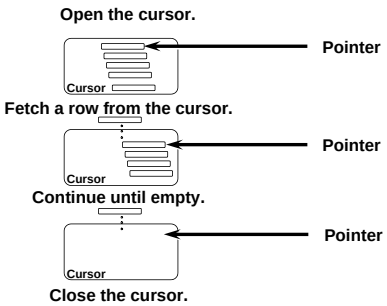



## Controlling Explicit Cursors




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

## Controlling Explicit Cursors




## Declaring the Cursor

**Syntax**

```
CURSOR cursor_name IS
     select_statement;
```

- Do not include the INTO clause in the cursor declaration.
- If processing rows in a specific sequence is required, use the ORDER BY clause in the query.

## Declaring the Cursor

**Example**

```
DECLARE
  CURSOR emp_cursor IS
    SELECT empno, ename
    FROM   emp;

  CURSOR dept_cursor IS
    SELECT *
    FROM   dept
    WHERE  deptno = 10;
BEGIN
  ...
```

## Opening the Cursor

**Syntax**

```
OPEN  cursor_name;
```

- Open the cursor to execute the query and identify the active set.
- If the query returns no rows, no exception is raised.
- Use cursor attributes to test the outcome after a fetch.

## Fetching Data from the Cursor

**Syntax**

```
FETCH cursor_name INTO [variable1, variable2, ...]
                       | record_name];
```

- Retrieve the current row values into variables.
- Include the same number of variables.
- Match each variable to correspond to the columns positionally.
- Test to see if the cursor contains rows.

## Fetching Data from the Cursor

**Examples**

```
FETCH emp_cursor INTO v_empno, v_ename;
```

```
...
OPEN defined_cursor;
LOOP
  FETCH defined_cursor INTO defined_variables
  EXIT WHEN ...;
  ...
    -- Process the retrieved data
  ...
END;
```

## Closing the Cursor

**Syntax**

```
CLOSE    cursor_name;
```

- Close the cursor after completing the processing of the rows.
- Reopen the cursor, if required.
- Do not attempt to fetch data from a cursor once it has been closed.

## Explicit Cursor Attributes

**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 total number of rows returned so far |

## The %NOTFOUND and %ROWCOUNT Attributes

- Use the %ROWCOUNT cursor attribute to retrieve an exact number of rows.
- Use the %NOTFOUND cursor attribute to determine when to exit the loop.

## Cursors and Records

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

Example

```
DECLARE
  CURSOR emp_cursor IS
    SELECT  empno, ename
    FROM    emp;
  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, and close occur.
- The record is implicitly declared.

## Cursor FOR Loops

Retrieve employees one by one until no more are left.

Example

```
DECLARE
  CURSOR emp_cursor IS
    SELECT ename, deptno
    FROM   emp;
BEGIN
  FOR emp_record IN emp_cursor LOOP
         -- implicit open and implicit fetch occur
    IF emp_record.deptno = 30 THEN
       ...
  END LOOP; -- implicit close occurs
END;
```

## Cursor FOR Loops Using Subqueries

No need to declare the cursor.

Example

```
BEGIN
  FOR emp_record IN (SELECT ename, deptno
                     FROM   emp) LOOP
         -- implicit open and implicit fetch occur
    IF emp_record.deptno = 30 THEN
       ...
  END LOOP; -- implicit close occurs
END;
```

## What Are Cursor Variables?

Memory

## Step 1: Defining a REF CURSOR Type

- Define a REF CURSOR type:

```
TYPE  ref_type_name IS REF CURSOR
  [RETURN return_type];
```

  - *ref_type_name* is a type specifier in subsequent declarations.
  - *return_type* represents a record type.
  - *return_type* indicates a strong cursor

```
DECLARE
  TYPE emp_cur_typ IS REF CURSOR
    RETURN emp%ROWTYPE;
  ...
```

## Step 2: Declaring a Cursor Variable

- Declare a cursor variable of a cursor type:

```
CURSOR_VARIABLE_NAME    REF_TYPE_NAME
```

  - *cursor_variable_name* is the name of the cursor variable.
  - *ref_type_name* is the name of a REF CURSOR type.

```
DECLARE
  TYPE emp_cur_typ IS REF CURSOR
    RETURN  emp%ROWTYPE;
  emp_cur_val emp_cur_typ;
```




## Summary

- Cursor types:
  - Implicit cursors: Used for all DML statements and single-row queries.
  - Explicit cursors: Used for queries of zero, one, or more rows.
- You can manipulate explicit cursors.
- You can evaluate the cursor status by using cursor attributes.
- You can use cursor FOR loops.