DATABASE MANAGEMENT SYSTEMS

Subject Teacher: Zartasha Baloch

CURSORS

Lecture # 34-35

Disclaimer: The material used in this presentation to deliver the lecture i.e., definitions/text and pictures/graphs etc. does not solely belong to the author/presenter. The presenter has gathered this lecture material from various sources on web/textbooks. Following sources are especially acknowledged:

1. Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
2. *https://www.tutorialspoint.com/plsql/index.htm*
3. *https://www.oracle.com/database/technologies/appdev/plsql.html*
4. Greenberg, Nancy, and Instructor Guide PriyaNathan. "Introduction to Oracle9i: SQL." ORACLE, USA (2001).

OBJECTIVES

- Implicit & Explicit cursors
- Cursor Attributes
- Controlling Explicit cursors

SQL CURSOR

- A cursor is a private SQL work area.
- A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor.
- A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.
- There are two types of cursors:
  - Implicit cursors
  - Explicit cursors
- The Oracle Server uses implicit cursors to parse and execute your SQL statements.
- Explicit cursors are explicitly declared by the programmer.

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

IMPLICIT CURSORS

- Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.
- Programmers cannot control the implicit cursors and the information in it.
- Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement.
  - For INSERT operations, the cursor holds the data that needs to be inserted.
  - For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

SQL CURSOR ATTRIBUTES

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

SQL%ROWCOUNT	**Number of rows affected by the most recent SQL statement (an integer value)**
SQL%FOUND	**Boolean attribute that evaluates to TRUE if the most recent SQL statement affects one or more rows**
SQL%NOTFOUND	**Boolean attribute that evaluates to TRUE if the most recent SQL statement does not affect any rows**
SQL%ISOPEN	**Always evaluates to FALSE because PL/SQL closes implicit cursors immediately after they are executed**

SQL CURSOR ATTRIBUTES

- Delete rows that have the specified order number from the ITEM table. Print the number of rows deleted.
- Example

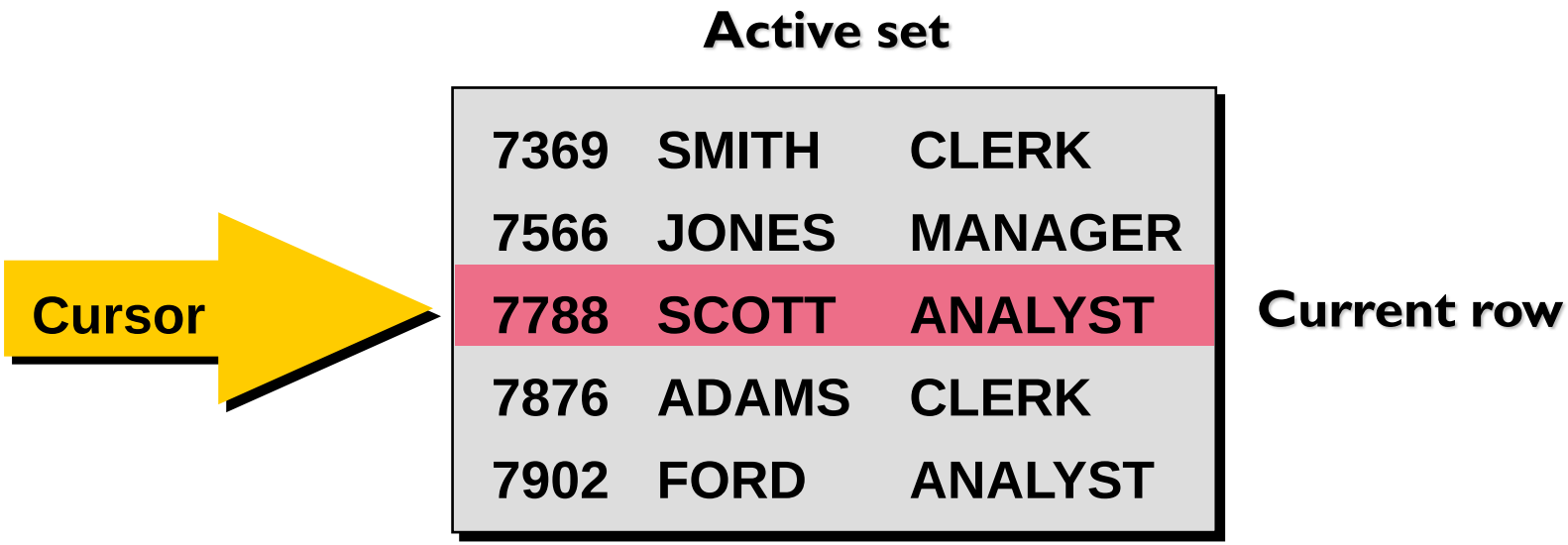
```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
  v_ordid  NUMBER := 605;
BEGIN
  DELETE FROM  item
  WHERE        ordid = v_ordid;
  :rows_deleted := (SQL%ROWCOUNT ||
                        ' rows deleted.');
END;
/
PRINT rows_deleted
```

IMPLICIT CURSOR

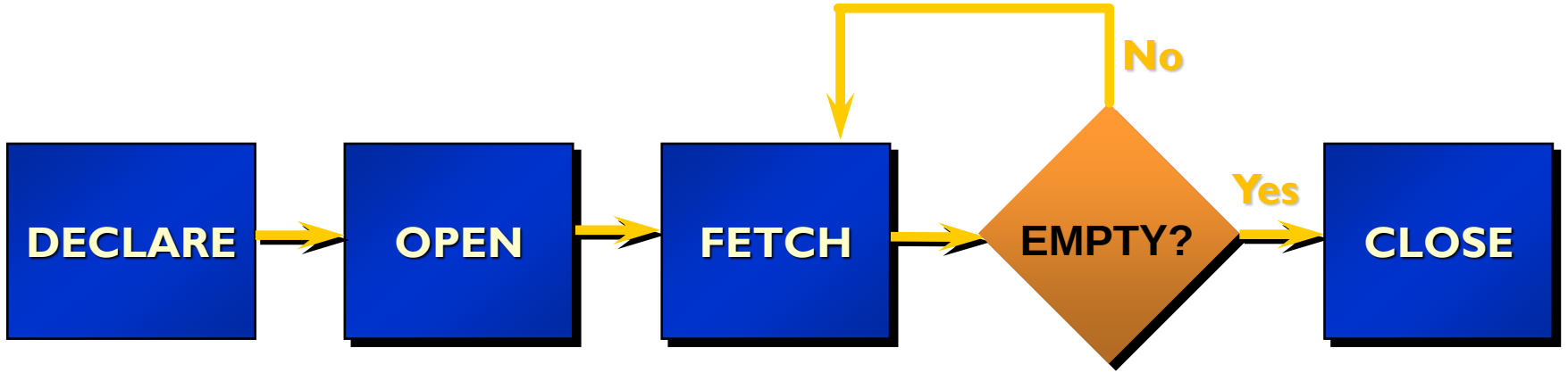
The following program will update the table and increase the salary of each customer by 500 and use the **SQL%ROWCOUNT** attribute to determine the number of rows affected.

```
DECLARE
   total_rows number(2);
BEGIN
   UPDATE customers
   SET salary = salary + 500;
   IF sql%notfound THEN
      dbms_output.put_line('no customers selected');
   ELSIF sql%found THEN
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers selected ');
   END IF;
END;
/
```

EXPLICIT CURSOR FUNCTIONS



CONTROLLING EXPLICIT CURSORS



Create a named SQL area

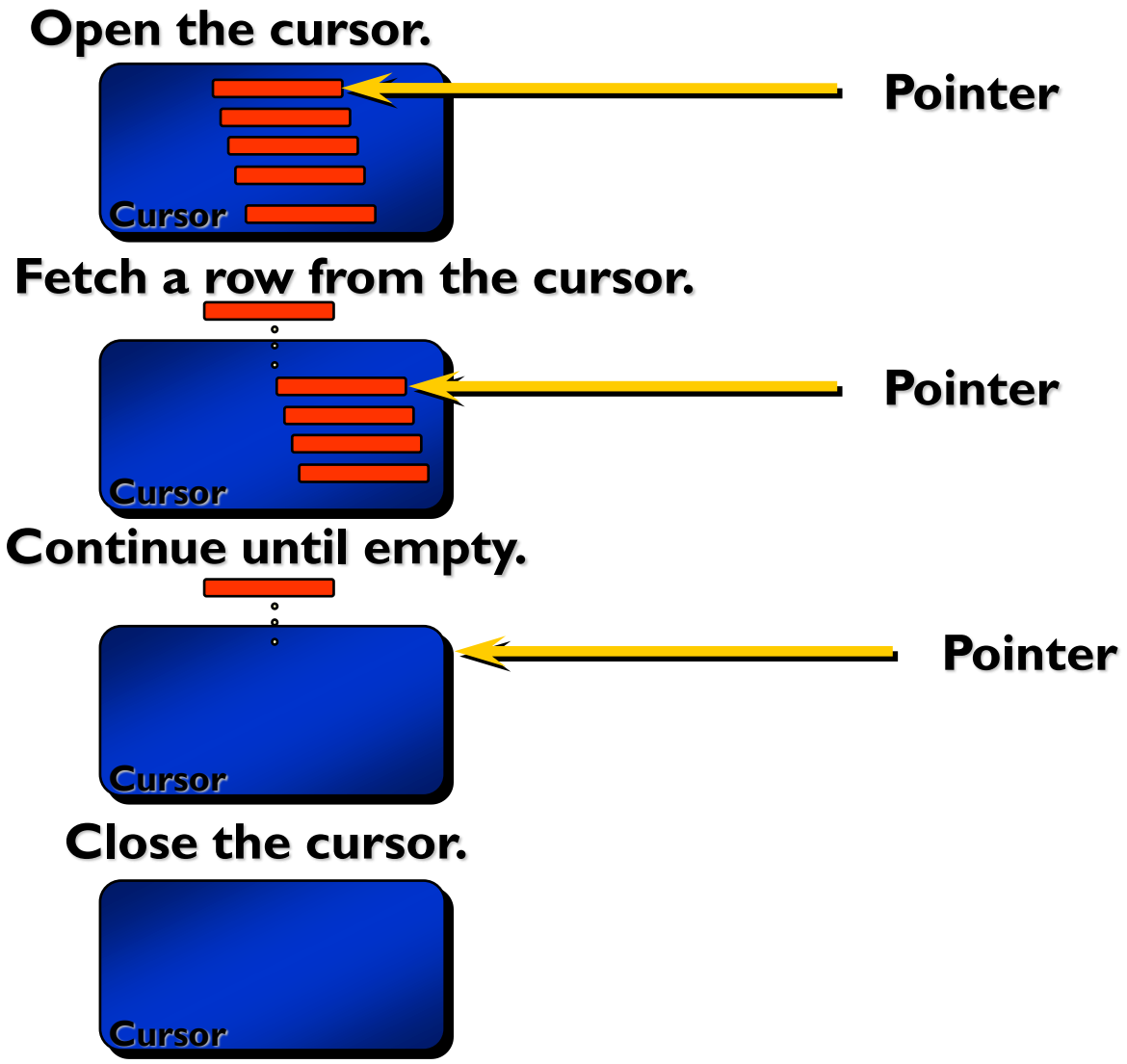
Identify the active set

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.

OPENING THE CURSOR

```
OPEN  cursor_name;
```

- Syntax

  - 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

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

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

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

CURSORS AND RECORDS

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

```
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;
  ...
```

CURSORS AND RECORDS

```
DECLARE
   c_id customers.id%type;
   c_name customer.name%type;
   c_addr customers.address%type;
   CURSOR c_customers is
      SELECT id, name, address FROM customers;
BEGIN
   OPEN c_customers;
   LOOP
   FETCH c_customers into c_id, c_name, c_addr;
      EXIT WHEN c_customers%notfound;
      dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
   END LOOP;
   CLOSE c_customers;
END;
/
```

CURSOR FOR LOOPS

```
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

```
DECLARE
        CURSOR c_product IS
          SELECT product_name, list_price FROM products ORDER BY list_price DESC;
BEGIN
        FOR r_product IN c_product
        LOOP
          dbms_output.put_line( r_product.product_name || ': $' || r_product.list_price );
        END LOOP;
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;
```

CURSOR FOR LOOPS USING SUBQUERIES

```
BEGIN
FOR r_product IN
        ( SELECT product_name, list_price
        FROM products
        ORDER BY list_price DESC )
LOOP
        dbms_output.put_line( r_product.product_name || ': $' || r_product.list_price );
END LOOP;
END;
```

CURSORS WITH PARAMETERS

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

- 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.

CURSORS WITH PARAMETERS

- Pass the department number and job title to the WHERE clause.
- Example

```
DECLARE
  CURSOR emp_cursor
  (v_deptno NUMBER, v_job VARCHAR2) IS
    SELECT    empno, ename
    FROM      emp
    WHERE     deptno = v_deptno
     AND      job = v_job;
BEGIN
  OPEN emp_cursor(10, 'CLERK');
...
```

THE FOR UPDATE CLAUSE

The SELECT FOR UPDATE statement allows you to lock the records in the cursor result set. You are not required to make changes to the records in order to use this statement. The record locks are released when the next commit or rollback statement is issued.

```
SELECT ...
FROM           ...
FOR UPDATE [OF column_reference][NOWAIT]
```

- Explicit locking lets you deny access for the duration of a transaction.
- Lock the rows *before* the update or delete.
- Using **FOR UPDATE NOWAIT** will cause the rows to be busy and acquires a lock until a commit or rollback is executed. Any other session that tries to acquire a lock will get an Oracle error message like ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired instead of waiting the lock to release.

THE FOR UPDATE CLAUSE

```
CURSOR cursor_name
IS
   select_statement
   FOR UPDATE [OF column_list] [NOWAIT];
```

cursor_name:	The name of the cursor.
select_statement:	A SELECT statement that will populate your cursor result set.
column_list:	The columns in the cursor result set that you wish to update.
NOWAIT:	Optional. The cursor does not wait for resources.

THE FOR UPDATE CLAUSE

- Retrieve the employees who work in department 30.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT empno, ename, sal
    FROM   emp
    WHERE  deptno = 30
    FOR UPDATE NOWAIT;
```

THE WHERE CURRENT OF CLAUSE

- If you plan on updating or deleting records that have been referenced by a SELECT FOR UPDATE statement, you can use the WHERE CURRENT OF statement.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.

THE WHERE CURRENT OF CLAUSE

The WHERE CURRENT OF statement allows you to update or delete the record that was last fetched by the cursor.

```
 UPDATE table_name
  SET set_clause
  WHERE CURRENT OF cursor_name;
```

```
 DELETE FROM table_name
WHERE CURRENT OF cursor_name;
```

THE WHERE CURRENT OF CLAUSE

```
DECLARE
  CURSOR sal_cursor IS
    SELECT  sal
    FROM emp
    WHERE   deptno = 30
    FOR UPDATE NOWAIT;
BEGIN
  FOR emp_record IN sal_cursor LOOP
    UPDATE  emp
    SET  sal = emp_record.sal * 1.10
    WHERE CURRENT OF sal_cursor;
  END LOOP;
  COMMIT;
END;
```