Oracle (PL/SQL)

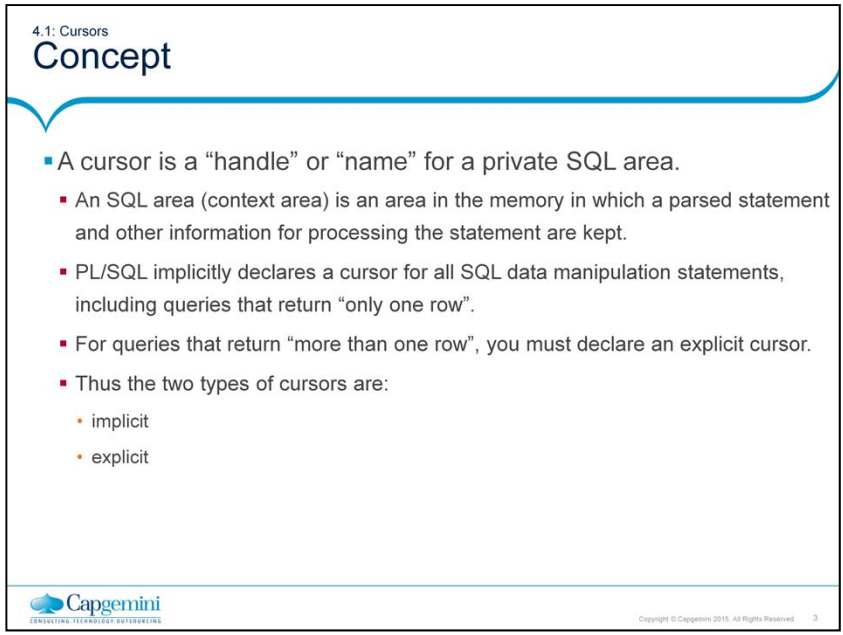
Lesson 04 : Cursors

Lesson Objectives

- To understand the following topics:
  - Introduction to Cursors
  - Implicit and Explicit Cursors
  - Cursor attributes
  - Processing Implicit Cursors and Explicit Cursors
  - Cursor with Parameters
  - Difference between Cursors and Cursor Variables
  - Use of Cursor Variables

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



Introduction to Cursors:

- ORACLE allocates memory on the Oracle server to process SQL statements. It is called as “**context area**”. Context area stores information like number of rows processed, the set of rows returned by a query, etc.
- A Cursor is a “handle” or “pointer” to the context area. Using this cursor the PL/SQL program can control the context area, and thereby access the information stored in it.
- There are two types of cursors - “explicit” and “implicit”.
  - ➢In an explicit cursor, a cursor name is explicitly assigned to a SELECT statement through CURSOR IS statement.
  - ➢An implicit cursor is used for all other SQL statements.
- Processing an explicit cursor involves four steps. In case of implicit cursors, the PL/SQL engine automatically takes care of these four steps.

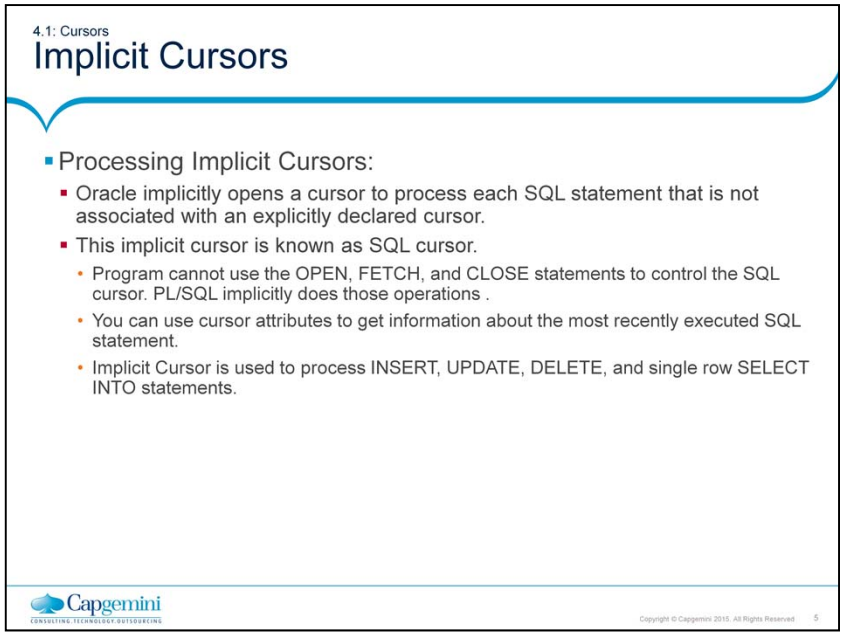
4.1: Cursors

Concept

- Implicit Cursor:
  - The PL/SQL engine takes care of automatic processing.
  - PL/SQL implicitly declares cursors for all DML statements.
  - They are simple SELECT statements and are written in the BEGIN block (executable section) of the PL/SQL.
  - They are easy to code, and they retrieve exactly one row

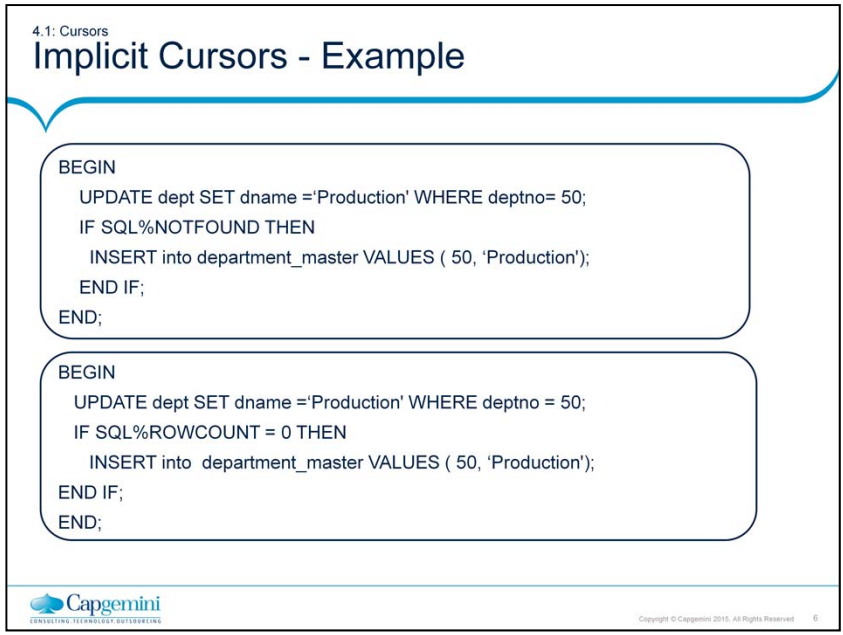
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4



Processing Implicit Cursors:

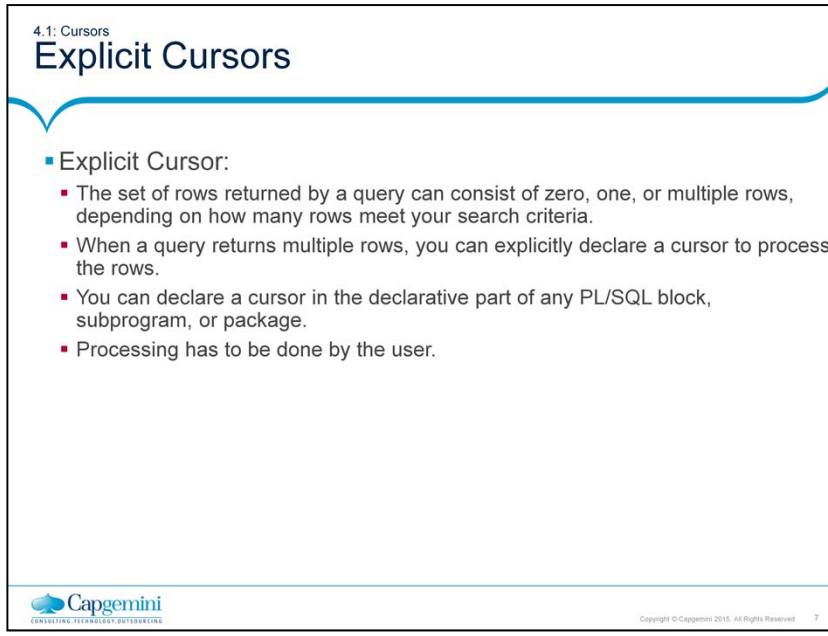
- All SQL statements are executed inside a context area and have a cursor, which points to that context area. This implicit cursor is known as SQL cursor.
- Implicit SQL cursor is not opened or closed by the program. PL/SQL implicitly opens the cursor, processes the SQL statement, and closes the cursor.
- Implicit cursor is used to process INSERT, UPDATE, DELETE, and single row SELECT INTO statements.
- The cursor attributes can be applied to the SQL cursor.



Processing Implicit Cursors:

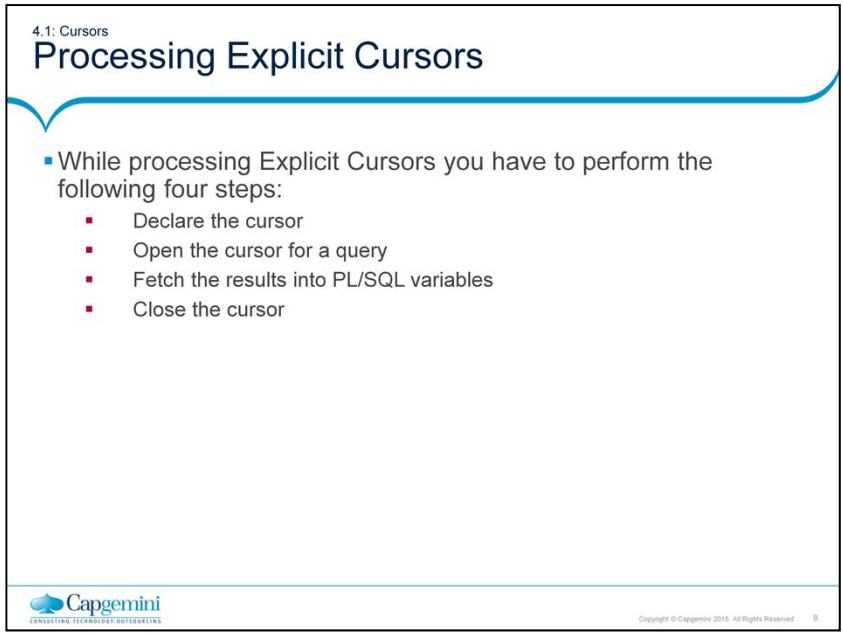
Note:

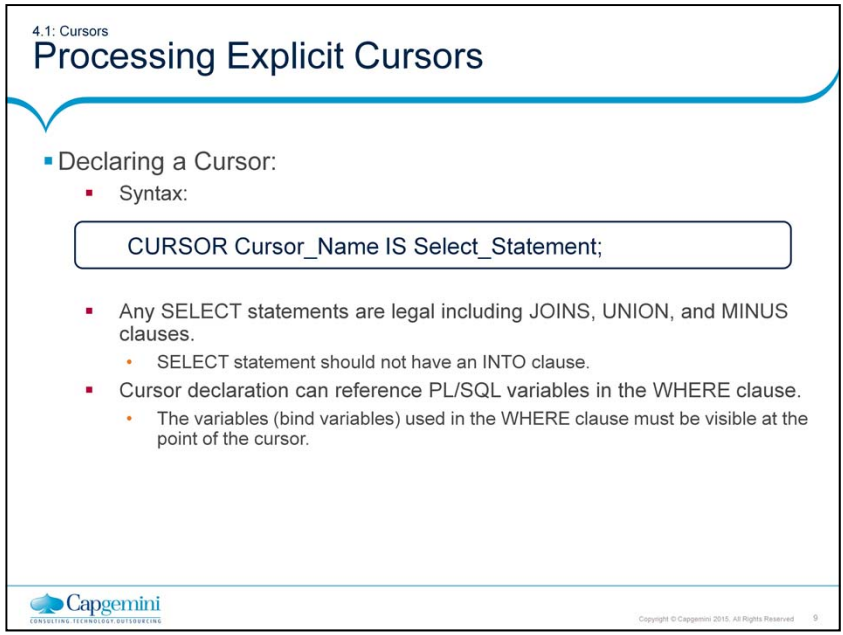
- SQL%NOTFOUND should not be used with SELECT INTO …. Statement.
- This is because SELECT INTO…. Statement will raise an ORACLE error if no rows are selected, and
  - control will pass to exception * section (discussed later), and
  - SQL%NOTFOUND statement will not be executed at all
- The slide shows two code snippets using Cursor attributes SQL%NOTFOUND and SQL%ROWCOUNT respectively.

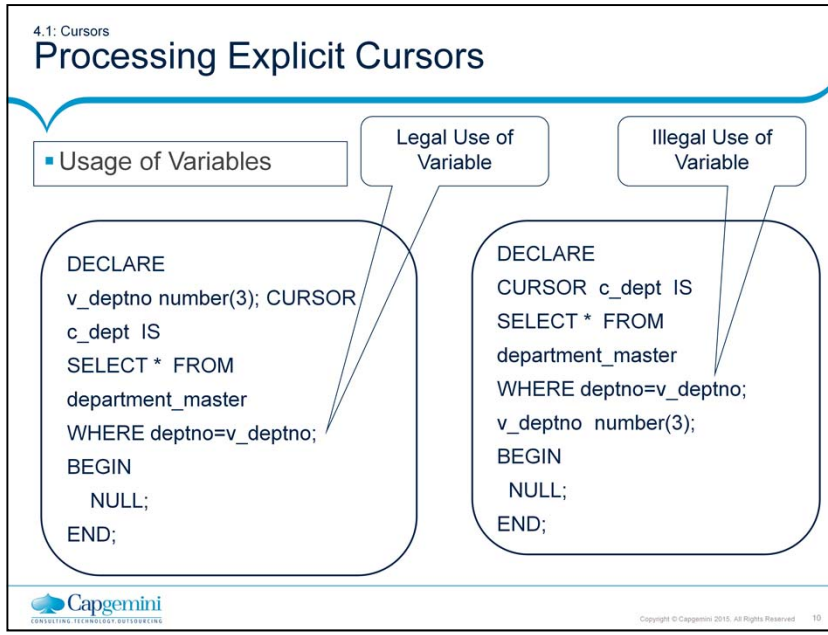


Explicit Cursor:

- When you need precise control over query processing, you can explicitly declare a cursor in the declarative part of any PL/SQL block, subprogram, or package.
- This technique requires more code than other techniques such as the implicit cursor FOR loop. But it is beneficial in terms of flexibility. You can:
  - ➢Process several queries in parallel by declaring and opening multiple cursors.
  - ➢Process multiple rows in a single loop iteration, skip rows, or split the processing into more than one loop.

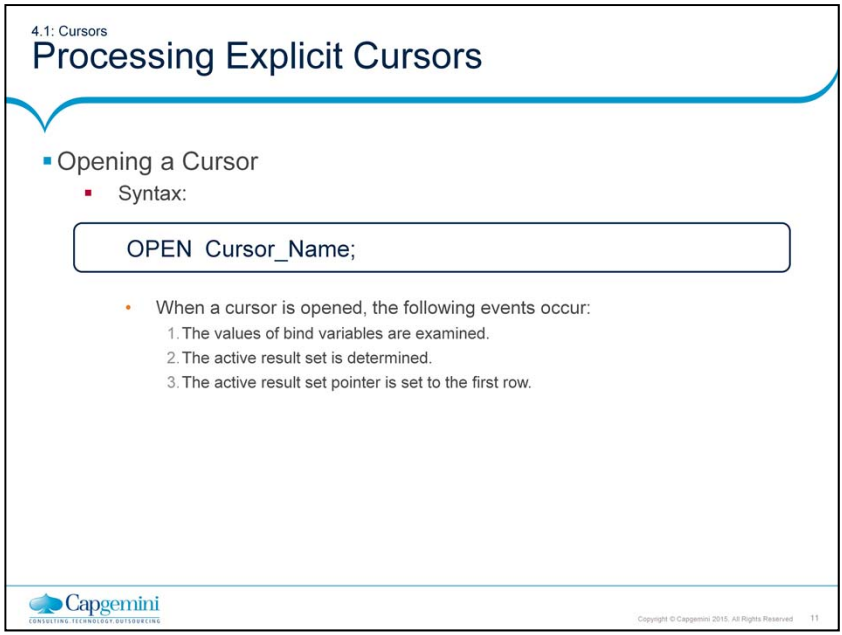

4.1: Cursors
Processing Explicit Cursors
While processing Explicit Cursors you have to perform the following four steps:
Declare the cursor
Open the cursor for a query
Fetch the results into PL/SQL variables
Close the cursor
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
8


4.1: Cursors
Processing Explicit Cursors
Declaring a Cursor:
Syntax:
CURSOR Cursor_Name IS Select_Statement;
Any SELECT statements are legal including JOINS, UNION, and MINUS clauses.
SELECT statement should not have an INTO clause.
Cursor declaration can reference PL/SQL variables in the WHERE clause.
The variables (bind variables) used in the WHERE clause must be visible at the point of the cursor.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
9



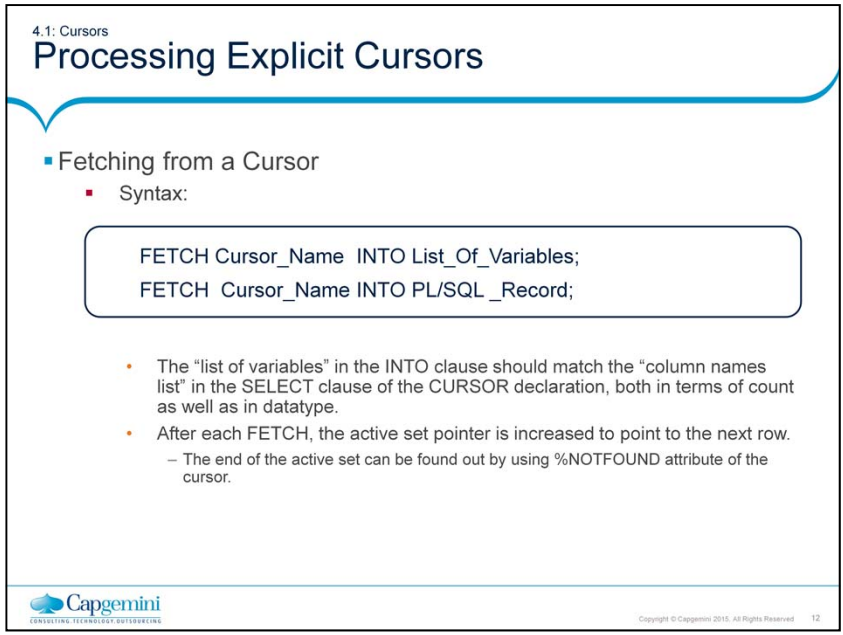
Processing Explicit Cursors: Declaring a Cursor:

- The code snippets on the slide show the usage of variables in cursor declaration. You cannot use a variable before it has been declared. It will be illegal.

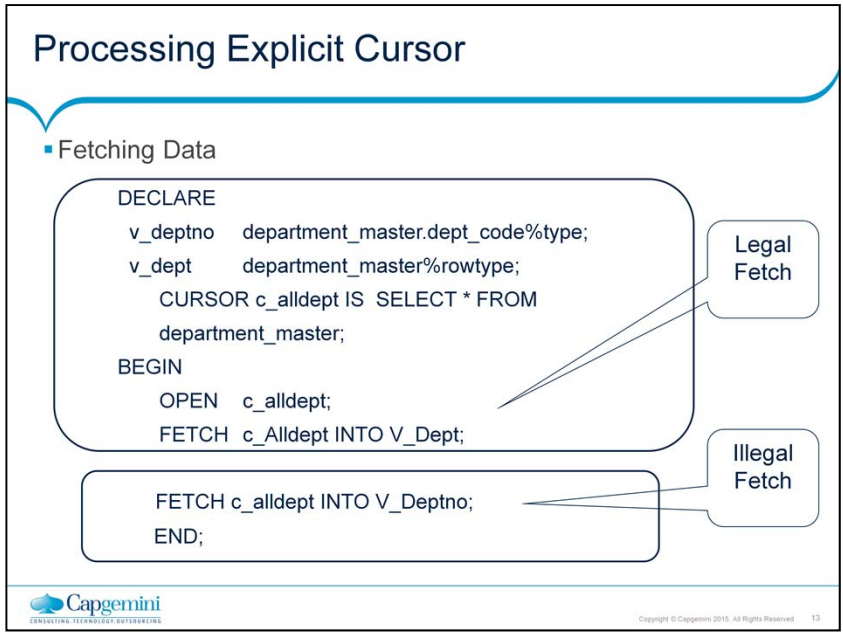


Processing Explicit Cursors: Opening a Cursor:

- When a Cursor is opened, the following events occur:
  1. The values of "bind variables" are examined.
  2. Based on the values of bind variables , the "active result set" is determined.
  3. The active result set pointer is set to the first row.
- "Bind variables" are evaluated only once at Cursor open time.
  - Changing the value of Bind variables after the Cursor is opened will not make any changes to the active result set.
  - The query will see changes made to the database that have been committed prior to the OPEN statement.
- You can open more than one Cursor at a time.

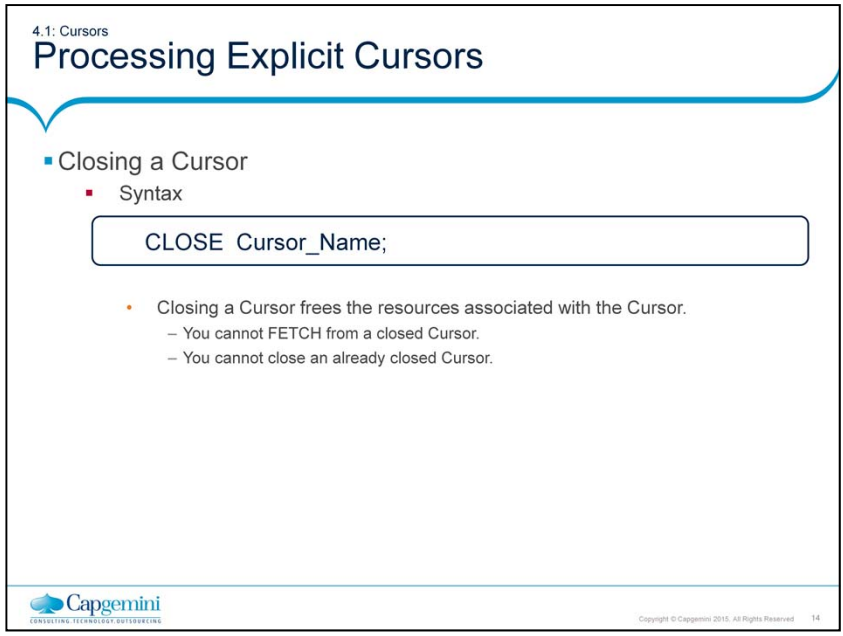


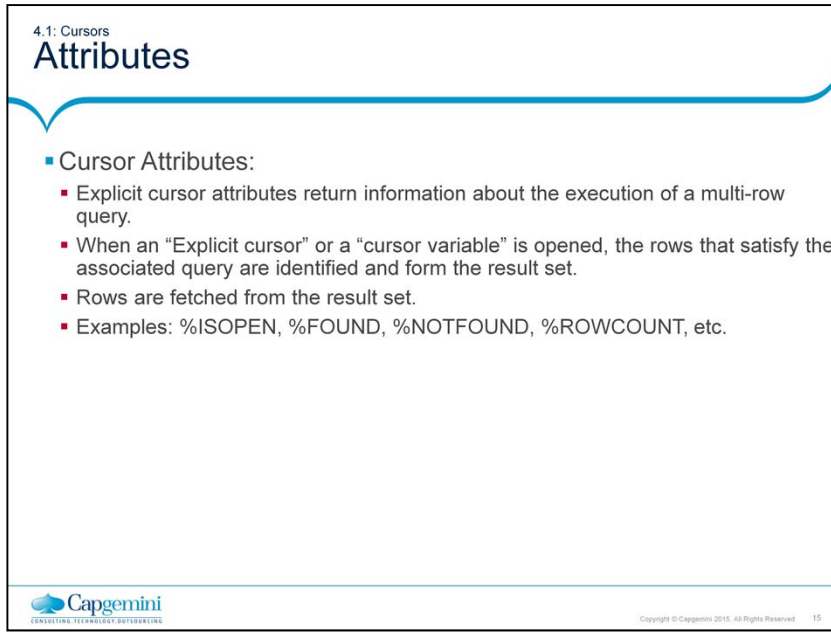
Processing Explicit Cursors: Fetching from Cursor:

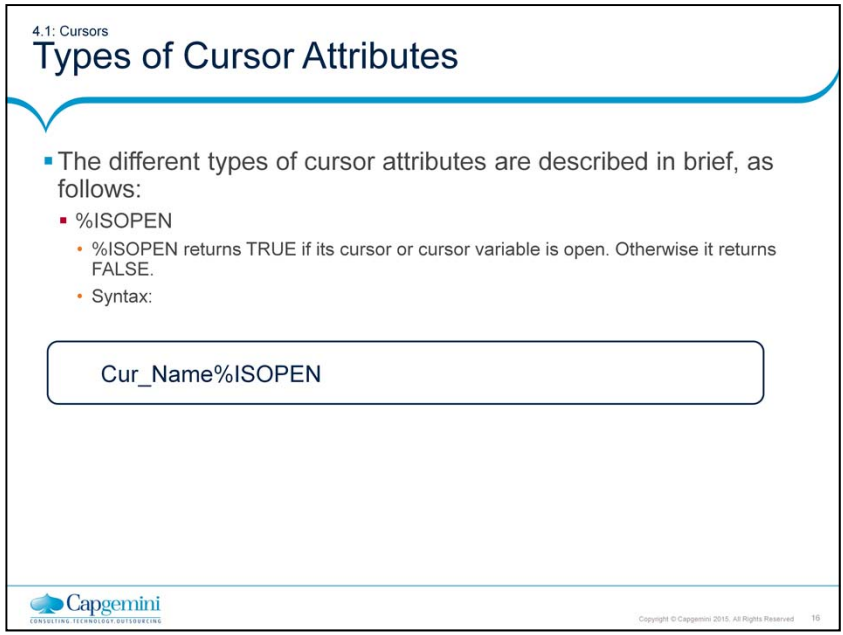


Processing Explicit Cursors: Fetching from Cursor:

- The code snippets on the slide shows example of fetching data from cursor. The second snippet FETCH is illegal since SELECT * selects all columns of the table, and there is only one variable in INTO list.
- For each column value returned by the query associated with the cursor, there must be a corresponding, type-compatible variable in the INTO list.
- To change the result set or the values of variables in the query, you must close and reopen the cursor with the input variables set to their new values.

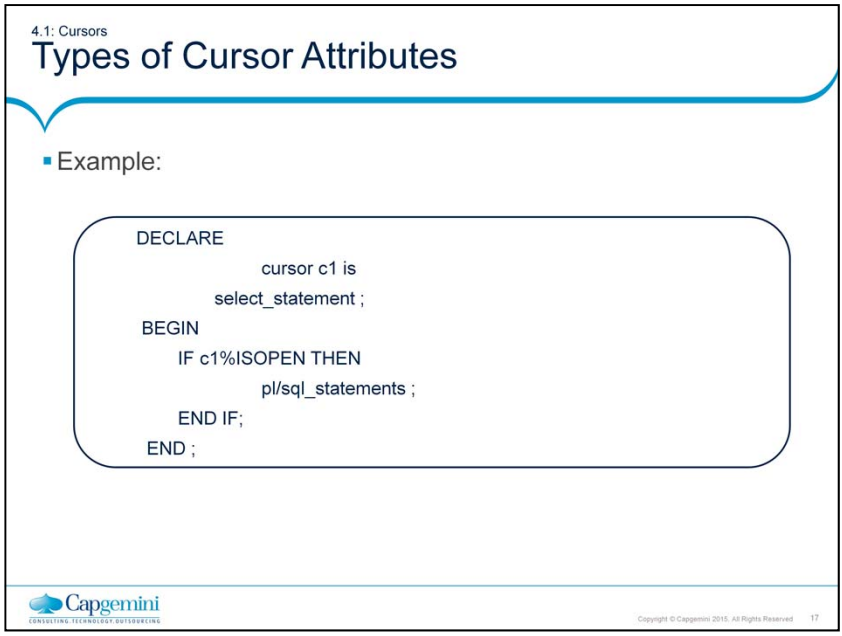

4.1: Cursors
Processing Explicit Cursors
Closing a Cursor
Syntax
CLOSE Cursor_Name;
Closing a Cursor frees the resources associated with the Cursor.
– You cannot FETCH from a closed Cursor.
– You cannot close an already closed Cursor.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
14


4.1: Cursors
Attributes
Cursor Attributes:
Explicit cursor attributes return information about the execution of a multi-row query.
When an “Explicit cursor” or a “cursor variable” is opened, the rows that satisfy the associated query are identified and form the result set.
Rows are fetched from the result set.
Examples: %ISOPEN, %FOUND, %NOTFOUND, %ROWCOUNT, etc.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
15



Cursor Attributes: %ISOPEN

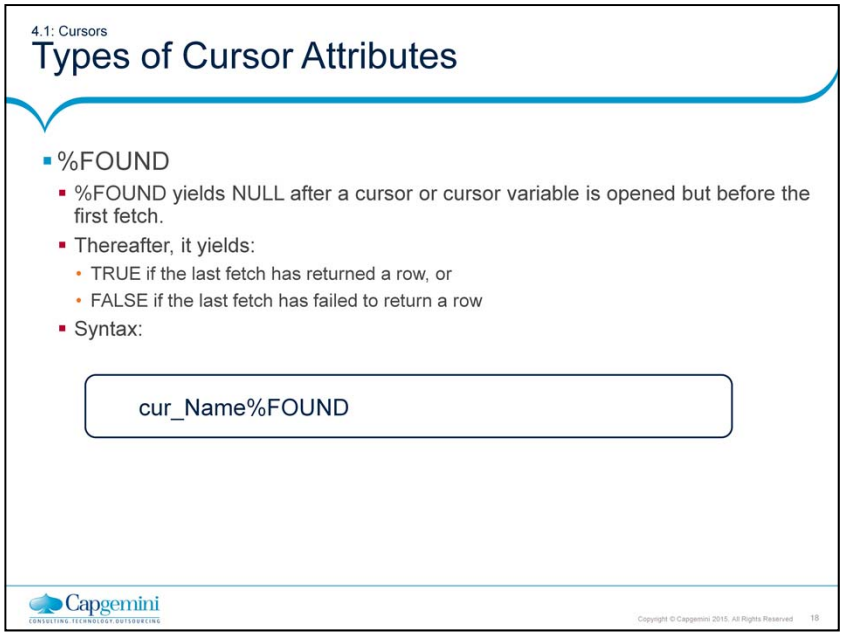
- This attribute is used to check the open/close status of a Cursor.
- If the Cursor is already open, the attribute returns TRUE.
- Oracle closes the SQL cursor automatically after executing its associated SQL statement. As a result, %ISOPEN always yields FALSE for Implicit cursor.



Cursor Attributes: %ISOPEN (contd.)

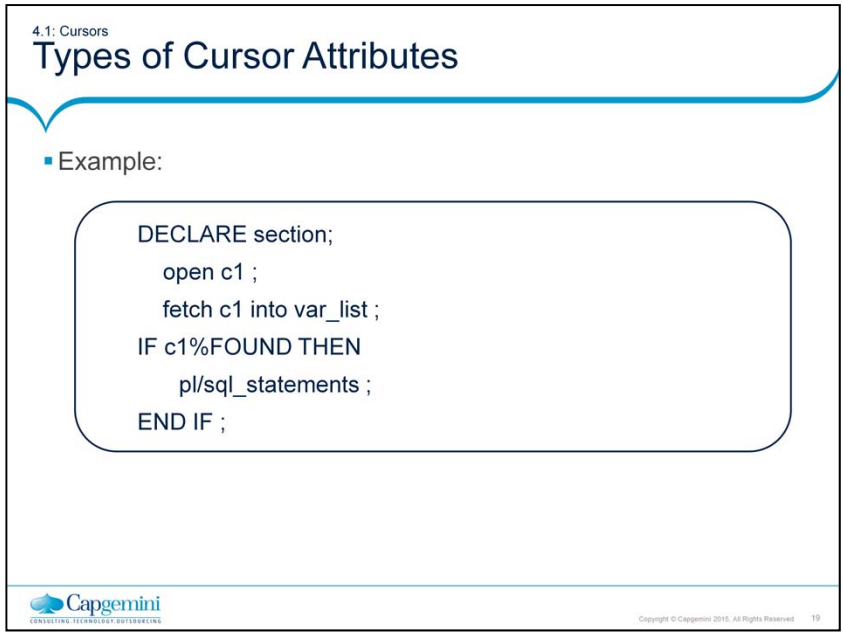
Note:

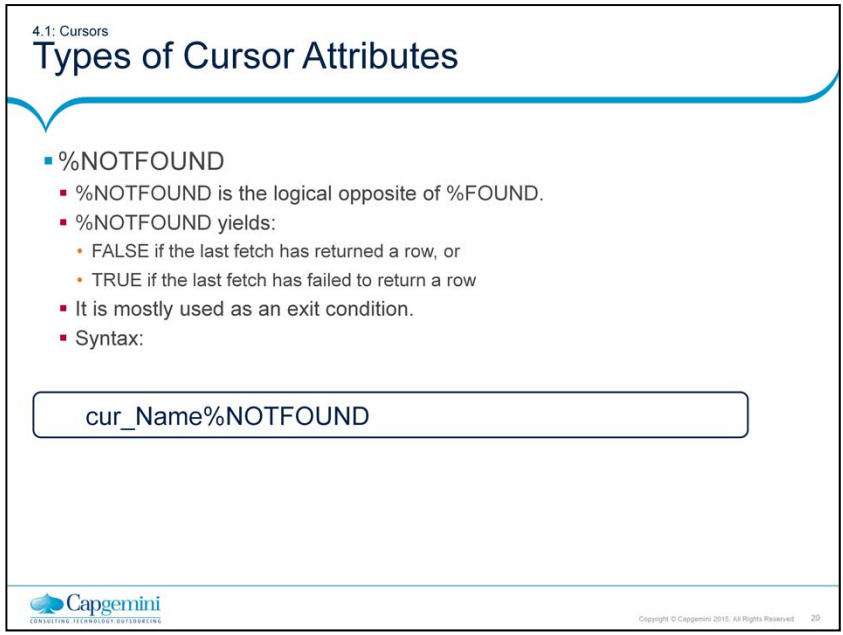
- In the example shown in the slide, C1%ISOPEN returns FALSE as the cursor is yet to be opened.
- Hence, the PL/SQL statements within the IF...END IF are not executed.



Cursor Attributes: %FOUND:

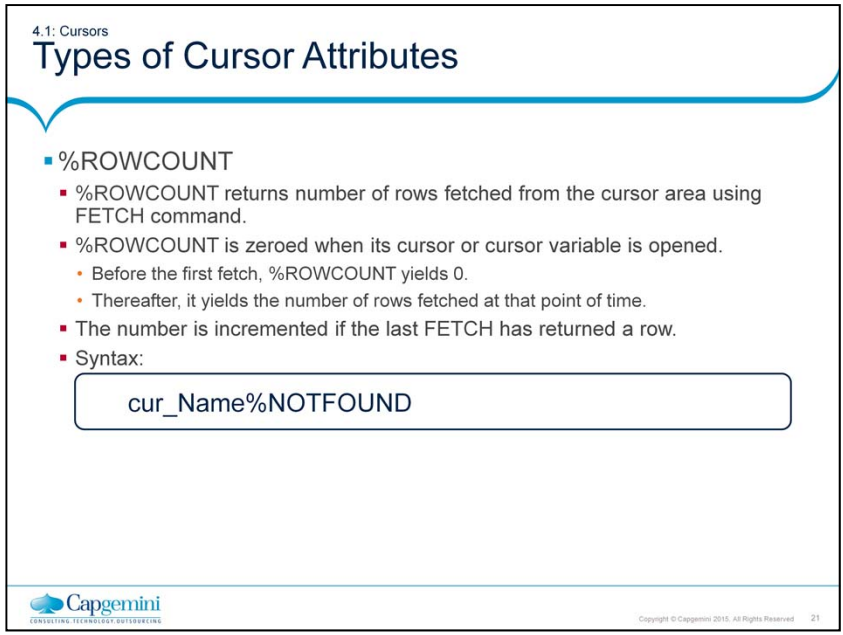
- Until a SQL data manipulation statement is executed, %FOUND yields NULL. Thereafter, %FOUND yields TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows, or a SELECT INTO statement returned one or more rows. Otherwise, %FOUND yields FALSE


4.1: Cursors
Types of Cursor Attributes
Example:
DECLARE section;
open c1 ;
fetch c1 into var_list ;
IF c1%FOUND THEN
pl/sql_statements ;
END IF ;
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
19



Cursor Attributes: %NOTFOUND:

- %NOTFOUND is the logical opposite of %FOUND. %NOTFOUND yields TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, %NOTFOUND yields FALSE.



Cursor Attributes: %ROWCOUNT

For example: To give a 10% raise to all employees earning less than Rs. 2500.

```
DECLARE
        V_Salary   emp.sal%TYPE;
        V_Empno  emp.empno%TYPE;
        CURSOR C_Empsal  IS
        SELECT empno, sal FROM emp
        WHERE sal <2500;
BEGIN
        IF  NOT C_Empsal%ISOPEN THEN
        OPEN  C_Empsal  ;
        END  IF ;
        LOOP
        FETCH  C_Empsal  INTO
        V_Empno,V_Salary;
        EXIT WHEN  C_Empsal %NOTFOUND ;
                --Exit out of block when no
        rows
        UPDATE emp SET sal = 1.1 * V_Salary
        WHERE empno = V_Empno;
        END LOOP ;
        CLOSE  C_Empsal ;
        COMMIT ;
END ;
```

4.1: Cursors

Cursor FETCH loops

- They are examples of simple loop statements.
- The FETCH statement should be followed by the EXIT condition to avoid infinite looping.
- Condition to be checked is cursor%NOTFOUND.
- Examples: LOOP .. END LOOP, WHILE LOOP, etc

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 22

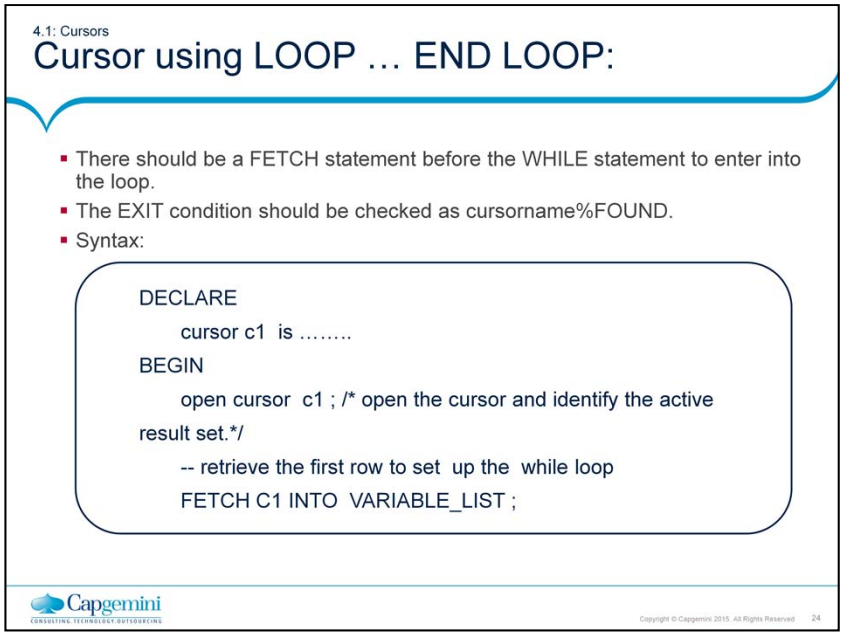
4.1: Cursors

Cursor using LOOP … END LOOP:

```
DECLARE
     cursor c1 is ……..
BEGIN
    open cursor c1;  /* open the cursor and identify the active result set.*/
LOOP
    fetch c1 into  variable_list ;
    -- exit out of the loop when there are no more rows.
    /* exit is done before processing to prevent handling of null rows.*/
    EXIT WHEN C1%NOTFOUND ;
    /* Process the fetched rows using variables and  PL/SQLstatements */
END LOOP;
    -- Free resources used by the cursor
    close  c1;
    -- commit
    commit;
END;
```

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 23



Processing Explicit Cursors: Cursor using WHILE loops:

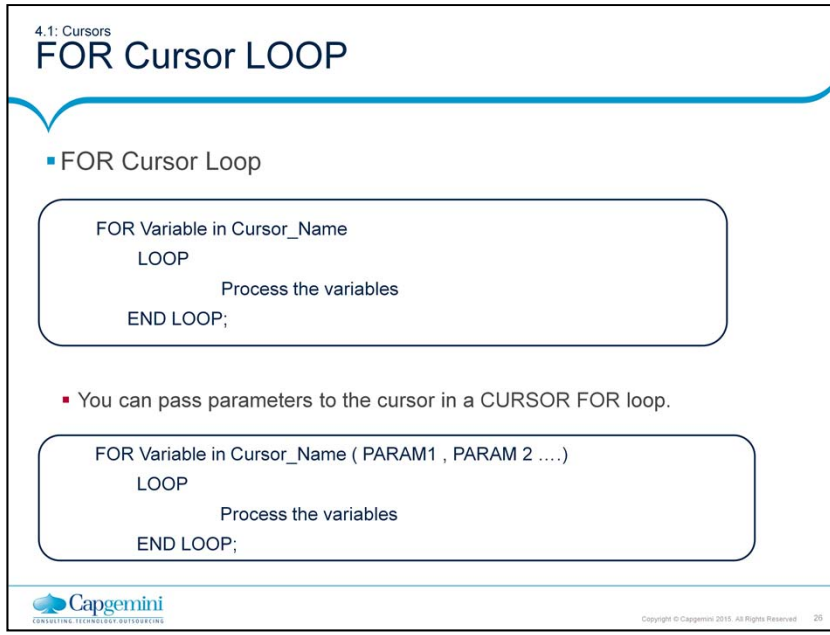
Note:

- FETCH statement appears twice, once before the loop and once after the loop processing.
- This is necessary so that the loop condition will be evaluated for each iteration of the loop.

Cursor using WHILE loops…contd

```
/*Continue  looping ,  processing & fetching till last row is
retrieved.*/
WHILE C1%FOUND
LOOP
      fetch c1 into variable_list;
END LOOP;
   CLOSE C1; -- Free resources used by the cursor.
   commit;   -- commit
END;
```

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING



Processing Explicit Cursors: FOR CURSOR Loop:

- For all other loops we had to go through all the steps of OPEN, FETCH, AND CLOSE statements for a cursor.
- PL/SQL provides a shortcut via a CURSOR FOR Loop. The CURSOR FOR Loop implicitly handles the cursor processing.

```
DECLARE
CURSOR  C1  IS ……..
BEGIN
-- An implict Open  of the  cursor C1 is done here.
-- Record variable should not be declared in
declaration section
FOR  Record_Variable   IN  C1 LOOP
-- An implicit Fetch is done here
-- Process the fetched rows using variables  and
PL/SQL statements
-- Before the loop is continued an implicit
C1%NOTFOUND test is done by PL/SQL
END LOOP;
-- An automatic close  C1 is issued after termination of
the loop
-- Commit
COMMIT ;
END;
/
```

- In this snippet, the record variable is implicitly declared by PL/SQL and is of the type C1%ROWTYPE and the scope of Record_Variable is only for the cursor FOR LOOP.

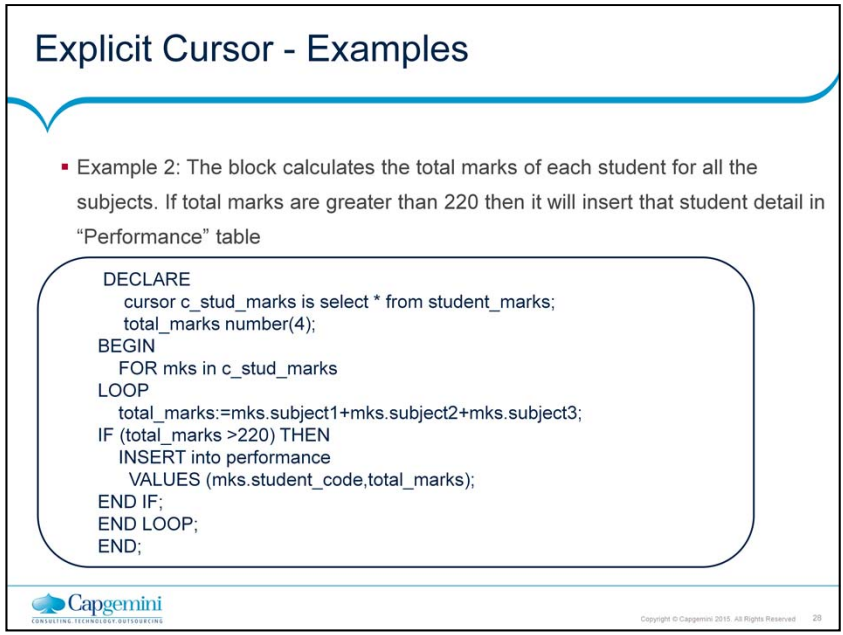
Explicit Cursor - Examples

- Example 1: To add 10 marks in subject3 for all students

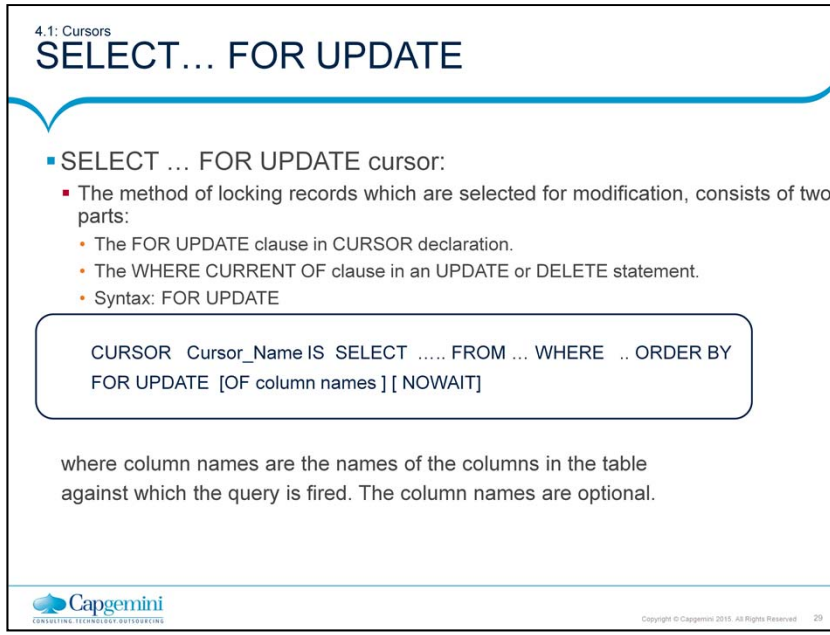
```
DECLARE
      v_sno      student_marks.student_code%type;
      cursor c_stud_marks is
     select student_code from student_marks;
BEGIN
      OPEN c_stud_marks;
      FETCH c_stud_marks into v_sno;
      WHILE c_stud_marks%found
        LOOP
          UPDATE student_marks SET subject3 =subject3+10
          WHERE student_code = v_sno ;
   FETCH c_stud_marks into v_empno;
   END LOOP ;
   CLOSE c_stud_marks;
END;
```

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 27

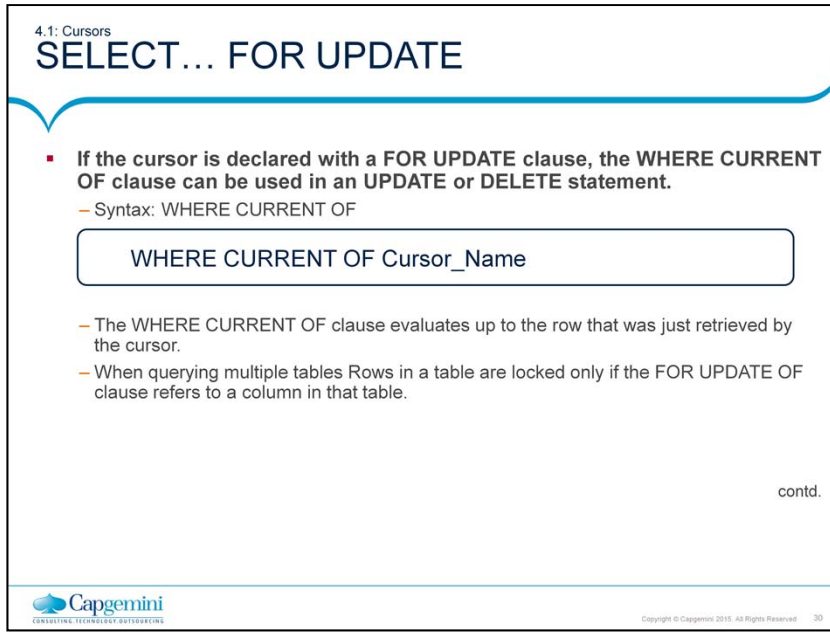


In the above example "Performance" is a user defined table.



Processing Explicit Cursors: Using FOR UPDATE

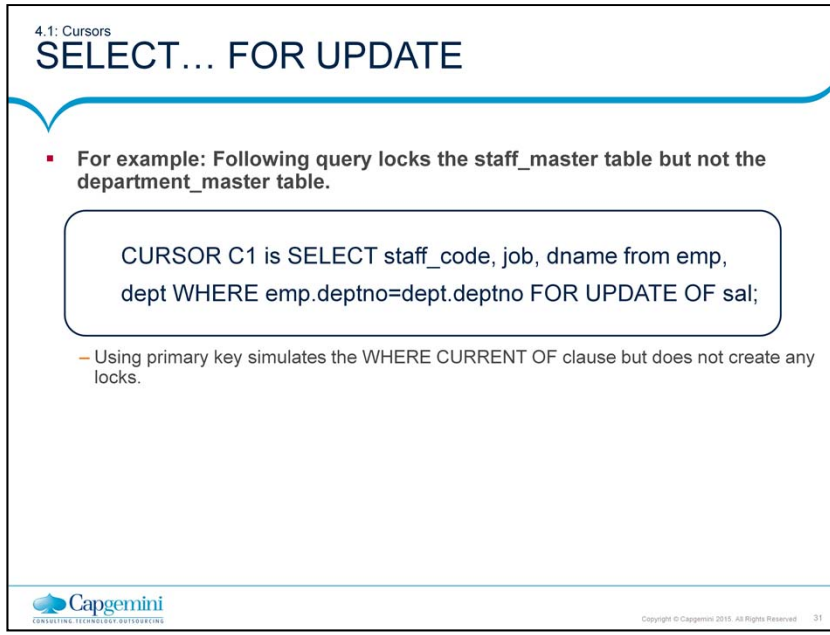
- The FOR UPDATE clause is part of the SELECT statement of the cursor. It is the last clause of the SELECT statement after the ORDER BY clause (if present).
- Normally, a SELECT operation does not lock the rows being accessed. This allows others to change the data selected by the SELECT statement. Besides, at OPEN time the active set consists of changes which were committed. Any changes made after OPEN even if they are committed, are not reflected in the active result set unless the cursor is reopened. This results in Data Inconsistency.
  - ➢If the FOR UPDATE clause is present, exclusive “row locks” are taken on the rows in the active set.
  - ➢These locks prevent other sessions / users from changing these rows unless the changes are committed.
- If another session / user already has locks on the rows in the active set, then the SELECT FOR UPDATE will wait indefinitely for these locks to be released. This statement will hang the system till the locks are released.
  - ➢To avoid this NOWAIT clause is used.
  - ➢With the NOWAIT clause SELECT FOR UPDATE will not wait for the locks acquired by previous sessions to be released and will return immediately with an ORACLE error.



Processing Cursors: Using WHERE CURRENT OF:

Note:

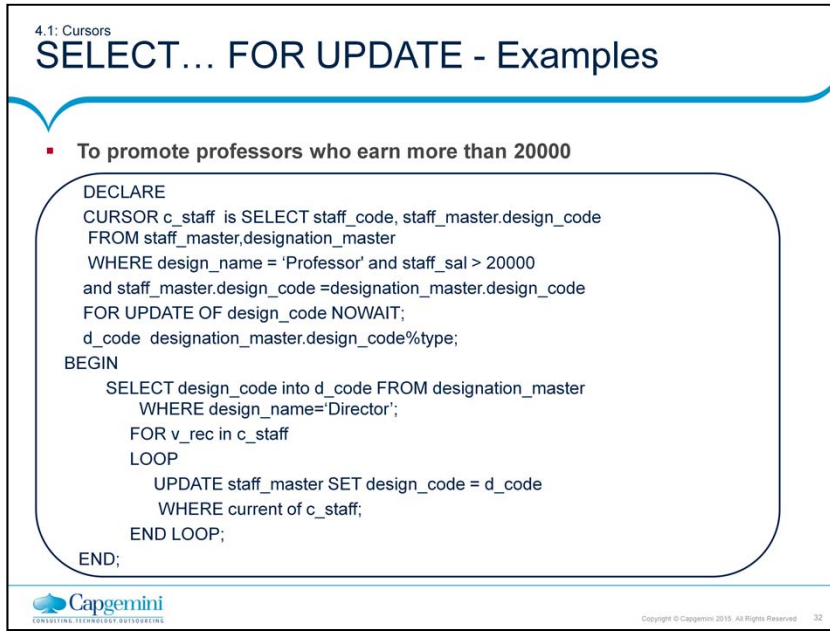
- When querying multiple tables you can use the FOR UPDATE OF column to confine row locking for a particular table.

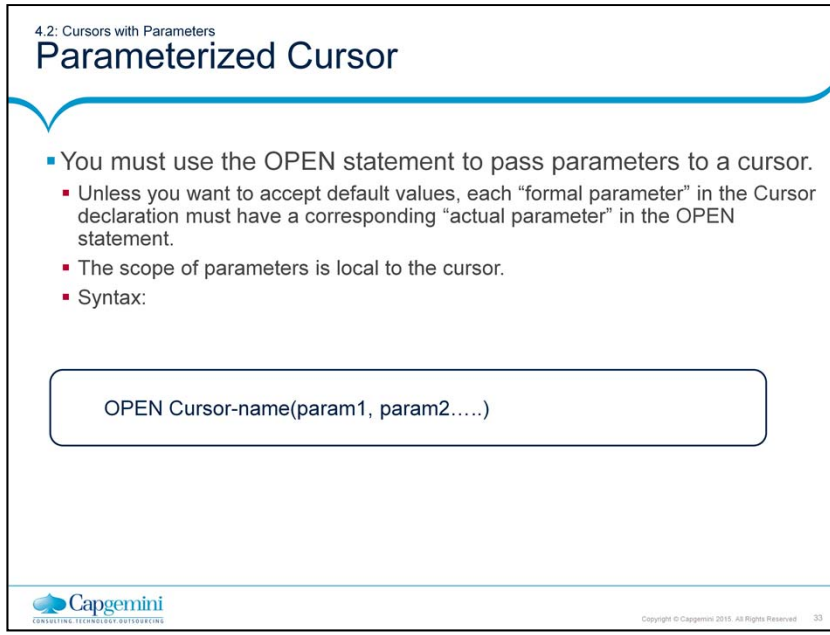


Processing Cursors: Using WHERE CURRENT OF (contd.):

Note:

- If you are using NOWAIT clause, then OF Column_List is essential for syntax purpose.
- Any COMMIT should be done after the processing is over in a CURSOR LOOP which has FOR UPDATE clause. This is because after commit locks will be released, the cursor will become invalid, and further FETCH will result in an error.
- This problem can be solved by using primary key. Using primary key simulates the WHERE CURRENT of clause, however it does not create any locks.


4.1: Cursors
SELECT… FOR UPDATE - Examples
To promote professors who earn more than 20000
DECLARE
CURSOR c_staff is SELECT staff_code, staff_master.design_code
FROM staff_master,designation_master
WHERE design_name = 'Professor' and staff_sal > 20000
and staff_master.design_code =designation_master.design_code
FOR UPDATE OF design_code NOWAIT;
d_code designation_master.design_code%type;
BEGIN
SELECT design_code into d_code FROM designation_master
WHERE design_name='Director';
FOR v_rec in c_staff
LOOP
UPDATE staff_master SET design_code = d_code
WHERE current of c_staff;
END LOOP;
END;
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
32

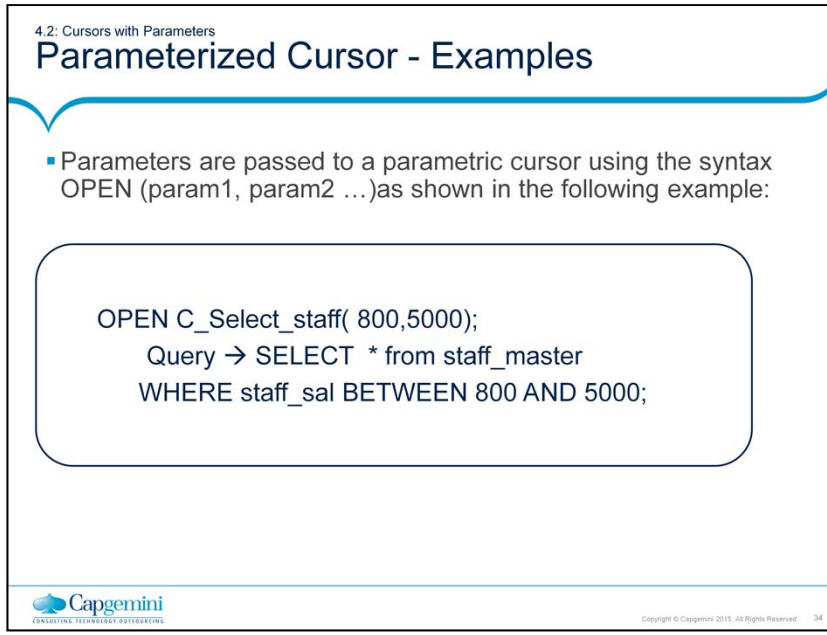


Cursor with Parameters:

- A cursor can take parameters, which can appear in the associated query wherever constants can appear. The formal parameters of a cursor must be IN parameters; they supply values in the query, but do not return any values from the query. You cannot impose the constraint NOT NULL on a cursor parameter.
- Cursor parameters can be referenced only within the query specified in the cursor declaration. The parameter values are used by the associated query when the cursor is opened.
- Example:

```
CURSOR C_Select_staff (Low_Sal NUMBER
DEFAULT  0 , High_Sal DEFAULT 10000) IS
SELECT  * from staff_master WHERE  staff_sal
BETWEEN  Low_Sal  AND  High_Sal);
```

- The scope of parameters is local to the cursor.
- The values of cursor parameters are used by associate queries when the cursor is OPEN.



Cursor with Parameters: Passing Parameters to Cursors

- In a FOR CURSOR LOOP parameters are passed using the following syntax:

```
FOR  Variable in Cursor_Name (PARAM1 ,
PARAM 2 ….);
FOR V_Get_Det in C_Select_staff( 800,5000)
LOOP
Process the variables
END LOOP;
```

- Unless you want to accept “default values”, each “formal parameter” in the cursor declaration must have a corresponding “actual parameter” in the OPEN statement.
  - ➢Formal parameters having default values need not have corresponding actual values.
  - ➢Each parameter must belong to a datatype which is compatible with the data type of its corresponding formal parameter.

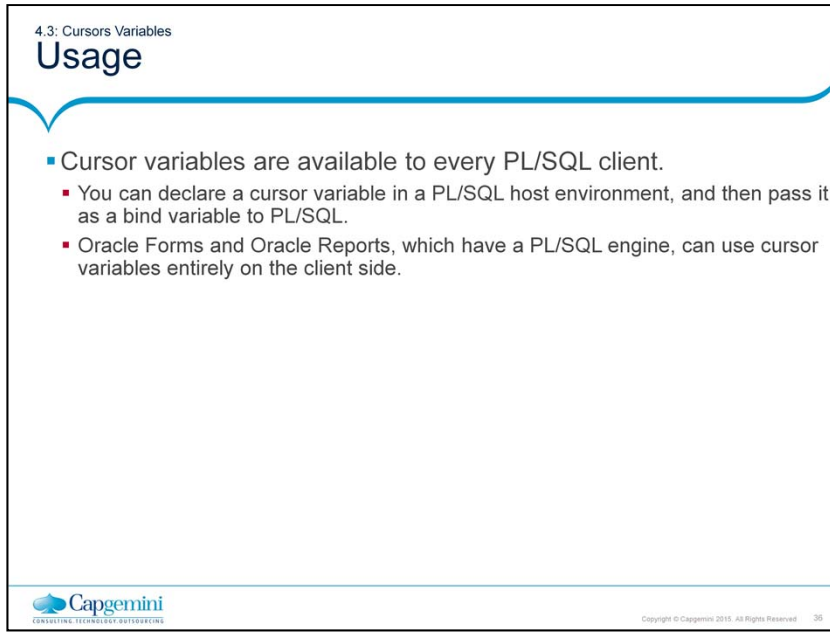
4.3: Cursors Variables

Usage

- Like a Cursor, a Cursor Variable points to the current row in the result set of a multi-row query.
  - A Cursor is static whereas a Cursor Variable is dynamic because it is not tied to a specific query.
  - You can open a Cursor Variable for any type-compatible query.
    - This offers more flexibility
  - You can assign new values to a Cursor Variable and pass it as a parameter to subprograms, including those in database.
    - This offers an easy way to centralize data retrieval.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

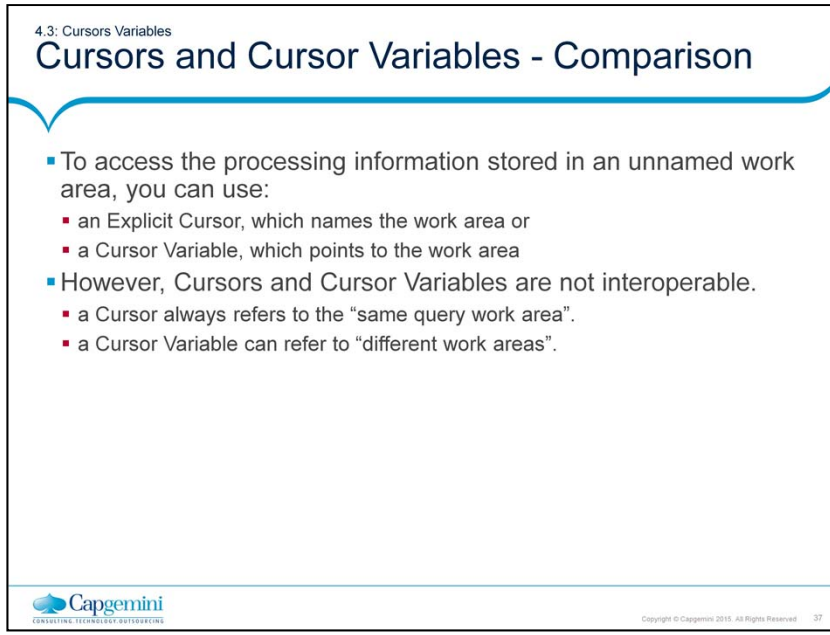
 35



Usage of Cursor Variables:

Note:

- Cursor variables are like ‘C’ pointers, which hold the memory location (address) of some item instead of the item itself.
- So when you are declaring a Cursor Variable you are creating a “pointer”, and not an “item”.



Cursors and Cursor Variables:

- In PL/SQL, a pointer has datatype ***REF*** X, where ***REF*** is a short name for *REFERENCE* and X stands for a class of an object. Therefore, a Cursor Variable has datatype REF CURSOR.
- To execute a multi-row query, Oracle opens an unnamed work area that stores processing information.
  - To access the information,
    - you can use an explicit cursor, which names the work area. Or,
    - you can use a cursor variable, which points to the work area.
  - A Cursor always refers to the "same query work area", whereas a Cursor Variable can refer to "different work areas". So, cursors and cursor variables are not interoperable that is, you cannot use one where the other is expected.
- Mainly, Cursor Variables are used "to pass query result sets" between PL/SQL stored subprograms and various clients.
  - Neither PL/SQL nor any of its clients owns a result set.
  - They simply "share a pointer" to the query work area in which the result set is stored.

  For example: Oracle Forms application, and Oracle Server can all refer to the same work area.

Cursors and Cursor Variables (contd.):

- A query work area remains accessible as long as any Cursor Variable points to it. Therefore, you can pass the value of a Cursor Variable freely from one scope to another.

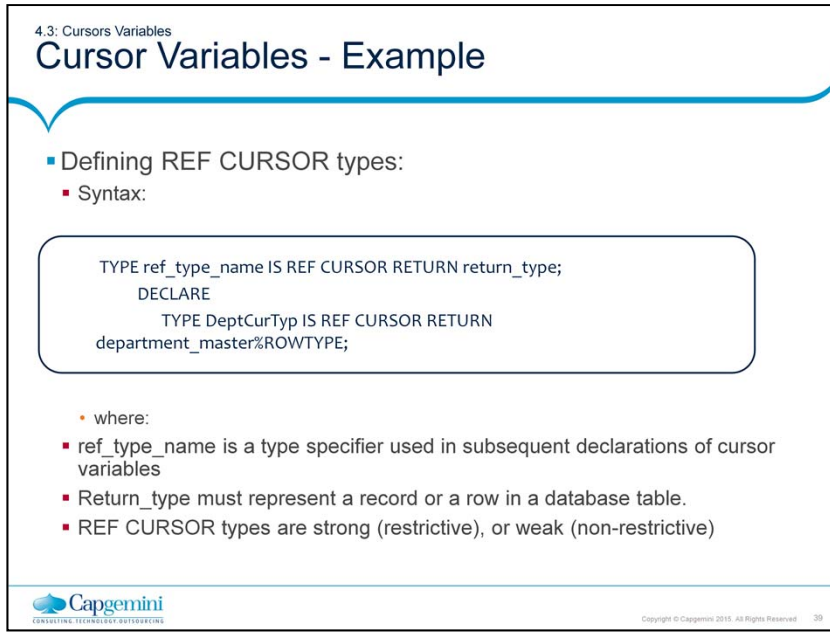
  For example: If you pass a “host cursor variable” to a PL/SQL block, the work area to which the Cursor Variable points remains accessible after the block completes.

- If you have a PL/SQL engine on the client side, then calls from client to server impose no restrictions.

  For example: You can declare a Cursor Variable on the client side, open and fetch from it on the server side, and then continue to fetch from it back on the client side. Also, you can reduce network traffic by having a *PL/SQL* block open (or close) several “host cursor variables” in a single round trip.

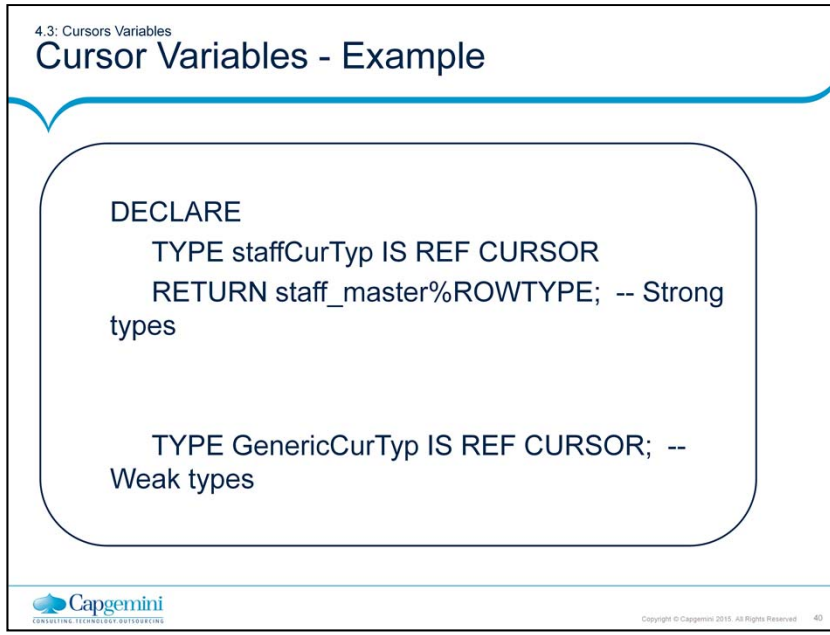
Creating Cursor Variables:

- To create cursor variables, two steps are required in the same sequence:
  1. Define a *REF CURSOR* type
  2. Declare cursor variables of that type
- You can define REF CURSOR types in any PL/SQL block, subprogram, or package.



Cursors and Cursor Variables: Defining REF CURSOR types:

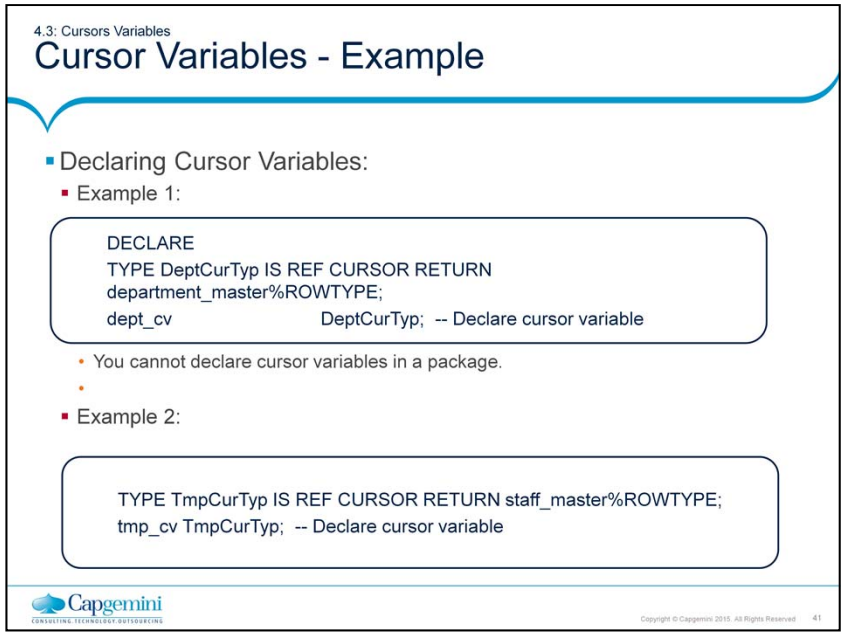
- **Example:** In the example shown on the slide , you specify a Return Type that represents a row in the database table dept:



Cursors and Cursor Variables: Defining REF CURSOR types (contd.):

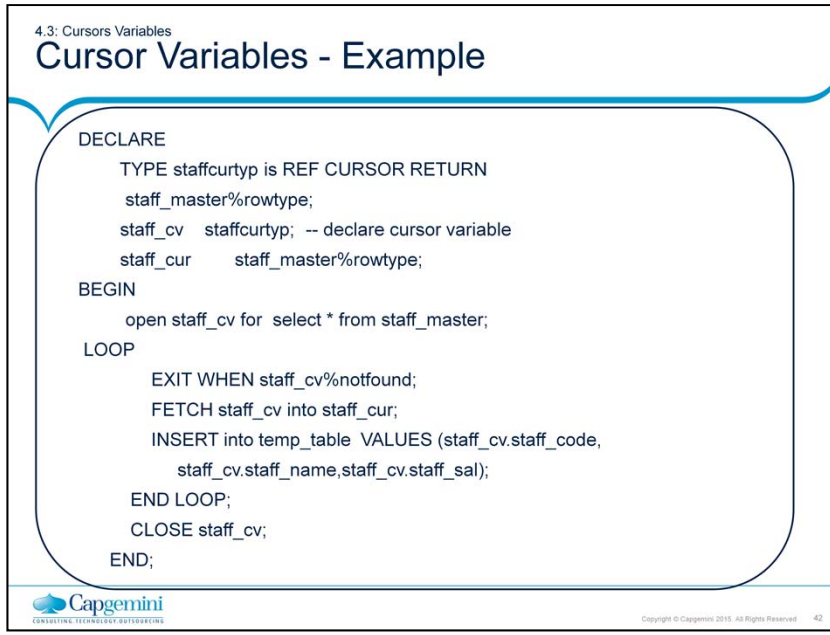
Note:

- A strong REF CURSOR type definition specifies a Return Type.
- However, a weak definition does not specify a Return Type.
- Strong REF CURSOR types are less error prone because the PL/SQL compiler lets you associate a strongly typed Cursor Variable only with type-compatible queries.
- However, weak REF CURSOR types are more flexible because the compiler lets you associate a weakly typed cursor variable with any query.



Cursors and Cursor Variables: Declaring Cursor Variable:

- As shown in examples shown on the slide, in the RETURN clause of a REF CURSOR type definition, you can use %ROWTYPE to specify a record type that represents a row returned by a strongly (not weakly) typed cursor variable.



- In the example shown on the slide, using a cursor variable we are fetching data from a table and inserting it in a temp_table.

Summary

- In this lesson, you have learnt:
  - Cursor is a “handle” or “name” for a private SQL area.
    - Implicit cursors are declared for queries that return only one row.
    - Explicit cursors are declared for queries that return more than one row.
  - Like a Cursor, a Cursor Variable points to the current row in the result set of a multi-row query.
    - However, Cursors and Cursor Variables are not interoperable.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

Review Question

- Question 1: A “Cursor” is static whereas a “Cursor Variable” is dynamic because it is not tied to a specific query.
  - True / False
- Question 2: %COUNT returns number of rows fetched from the cursor area by using FETCH command.
  - True / False

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 44

Review Question

- Question 3: Implicit SQL cursor is opened or closed by the program.
  - True / False
- Question 4: A ___ specifies a Return Type.
- Question 5: PL/SQL provides a shortcut via a ___ Loop, which implicitly handles the cursor processing.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 45