Oracle Basics (PL/SQL)

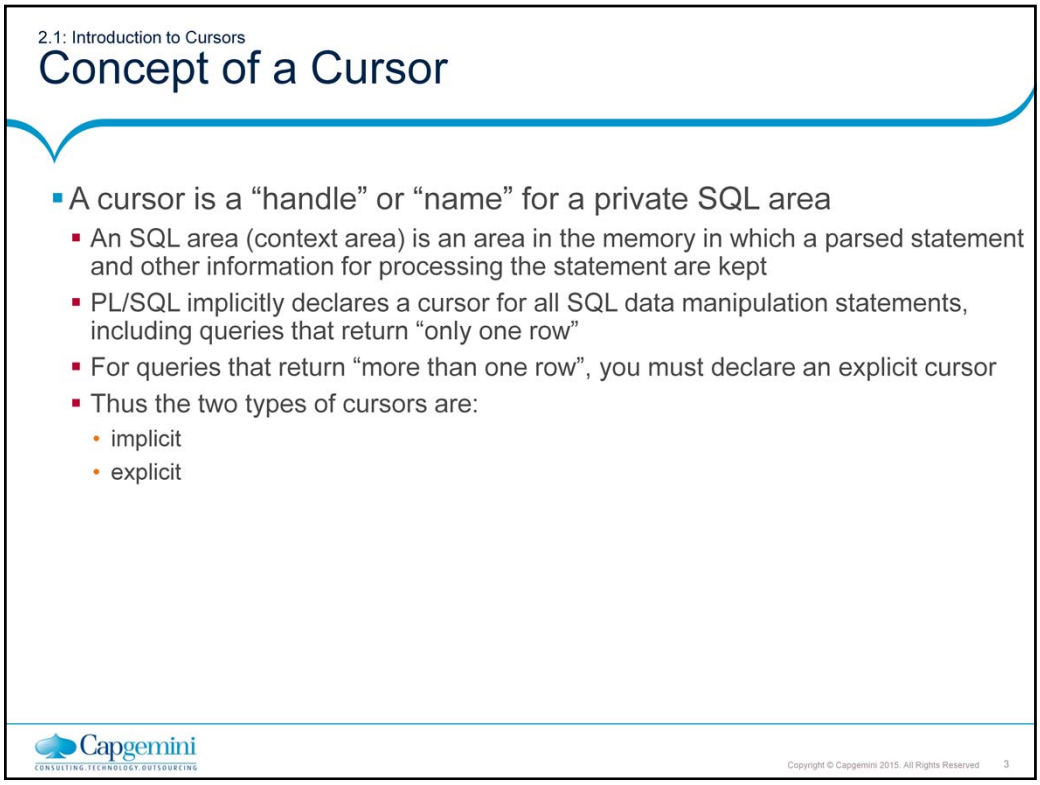
Lesson 02 Introduction to Cursors

Lesson Objectives

- To understand the following topics:
  - Introduction to Cursors
  - Implicit Cursors
  - Explicit Cursors
  - Cursor attributes
  - Processing Implicit Cursors and Explicit Cursors

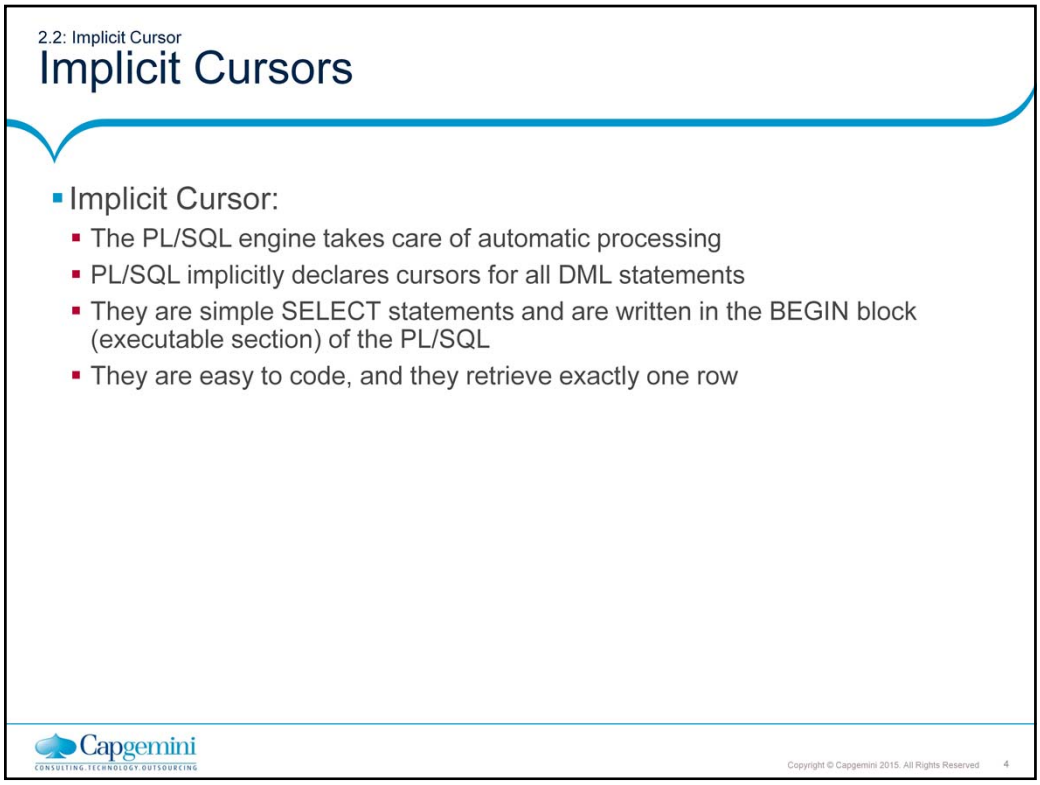
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

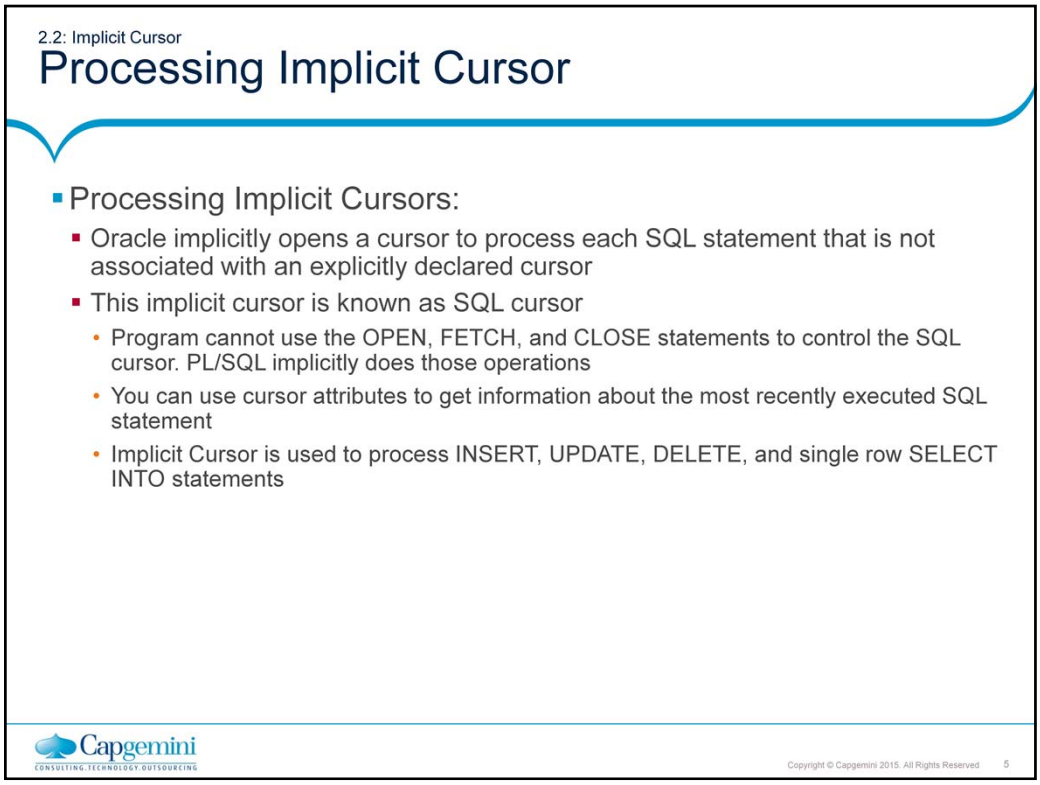
 2



Introduction to Cursors:

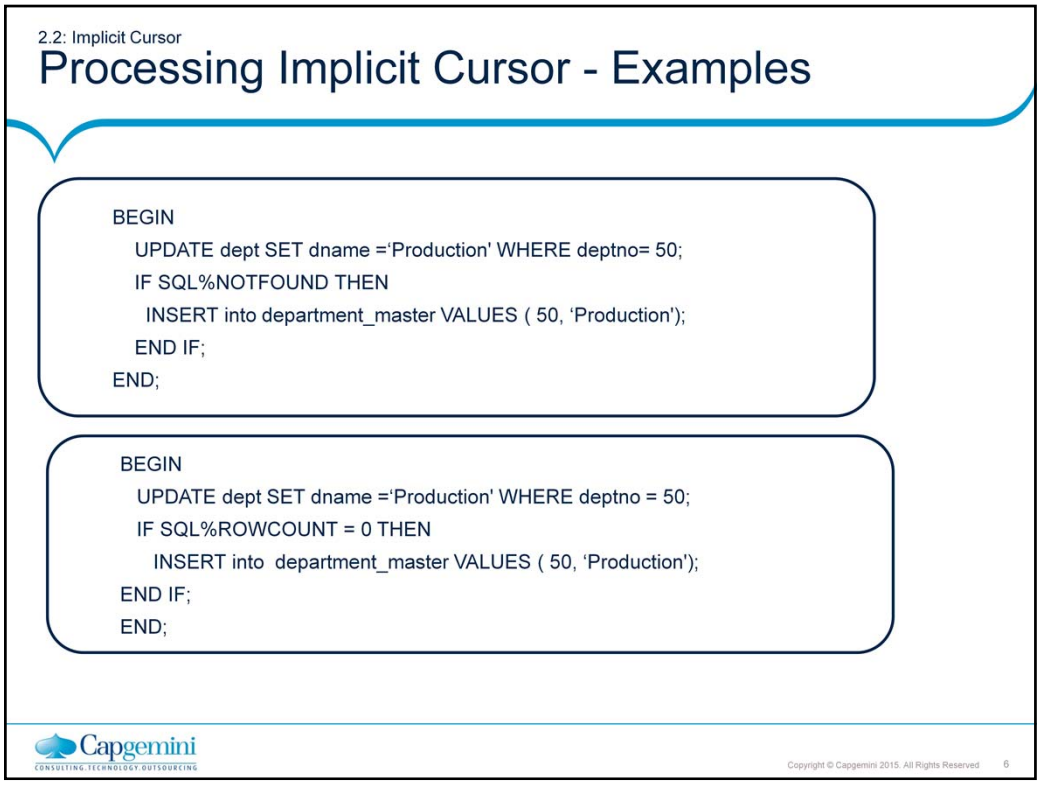
- ORACLE allocates memory on the Oracle server to process SQL statements. It is called as “**context area**”. Context area stores information like number of rows processed, the set of rows returned by a query, etc.
- A Cursor is a “handle” or “pointer” to the context area. Using this cursor the PL/SQL 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.


2.2: Implicit Cursor
Implicit Cursors
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
Copyright © Capgemini 2015. All Rights Reserved 4



Processing Implicit Cursors:

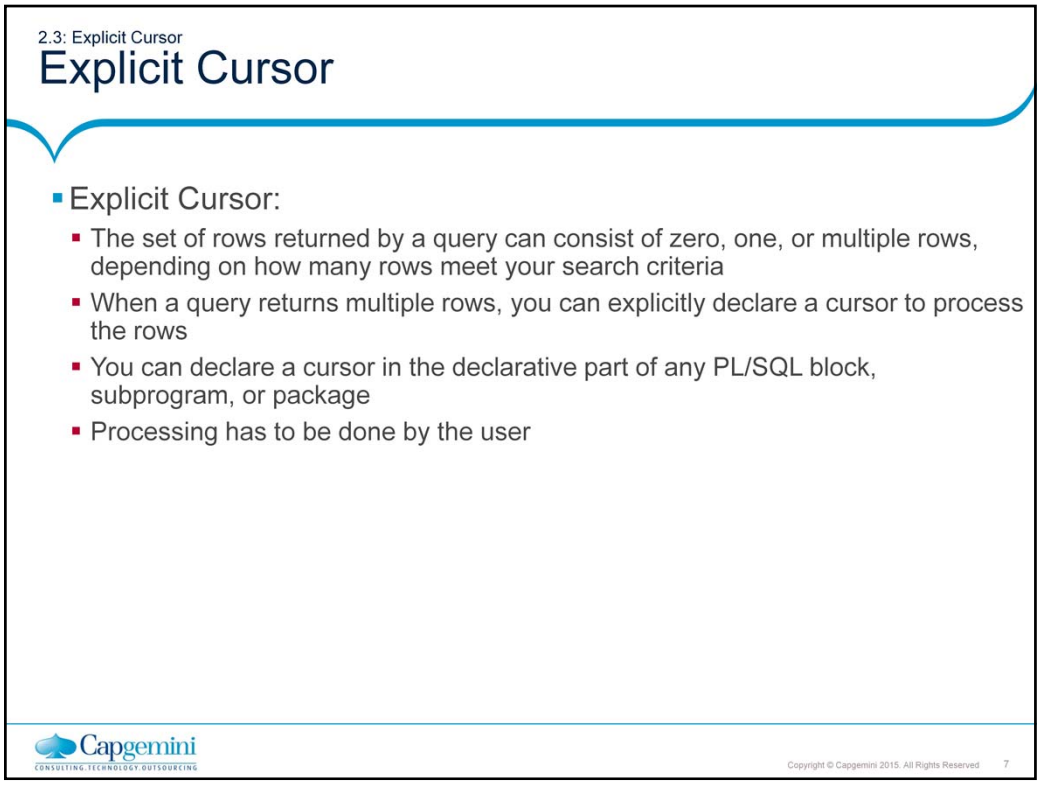
- All SQL statements are executed inside a context area and have a cursor, which points to that context area. This implicit cursor is known as SQL cursor.
- Implicit SQL cursor is not opened or closed by the program. PL/SQL implicitly opens the cursor, processes the SQL statement, and closes the cursor.
- 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.

Processing Explicit Cursors:

Overview of Explicit Cursors

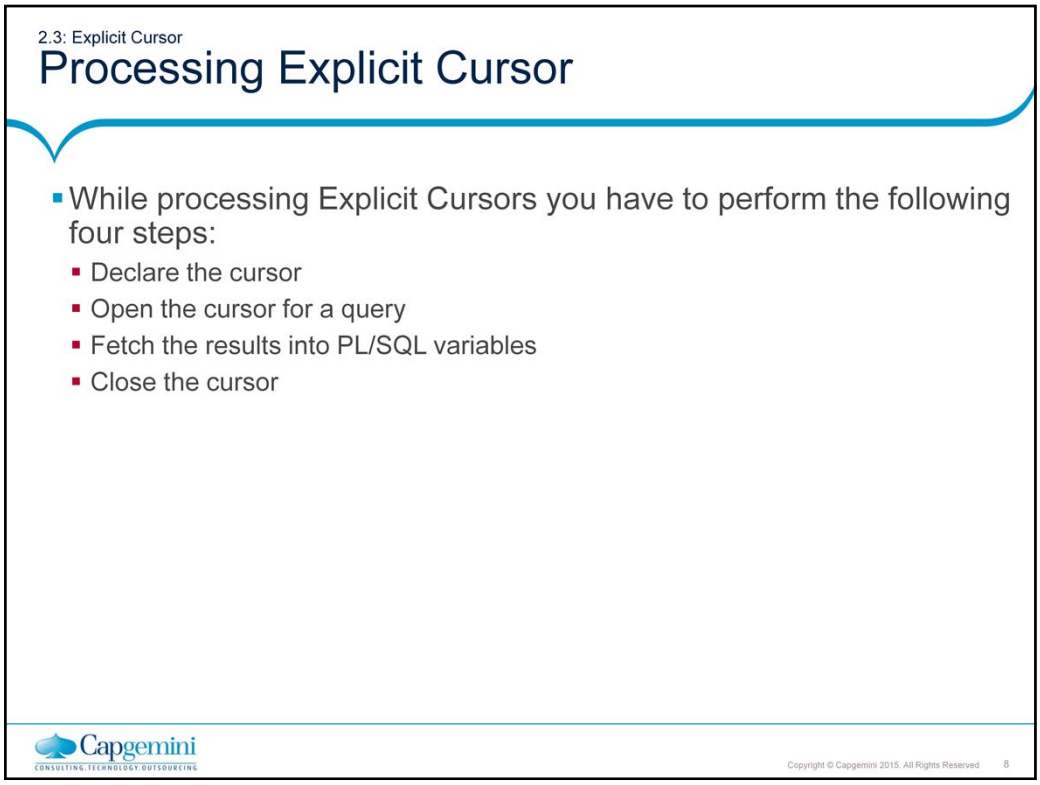
•You use three commands to control a cursor: OPEN, FETCH, and CLOSE.

•First, you initialize the cursor with the OPEN statement, which identifies the result set. Then, you can execute FETCH repeatedly until all rows have been retrieved, or you can use the BULK COLLECT clause to fetch all rows at once. When the last row has been processed, you release the cursor with the CLOSE statement.

Declaring a Cursor

You must declare a cursor before referencing it in other statements. You give the cursor a name and associate it with a specific query. You can optionally declare a return type for the cursor (such as table_name%ROWTYPE). You can optionally specify parameters that you use in the WHERE clause instead of referring to local variables. These parameters can have default values.

contd.



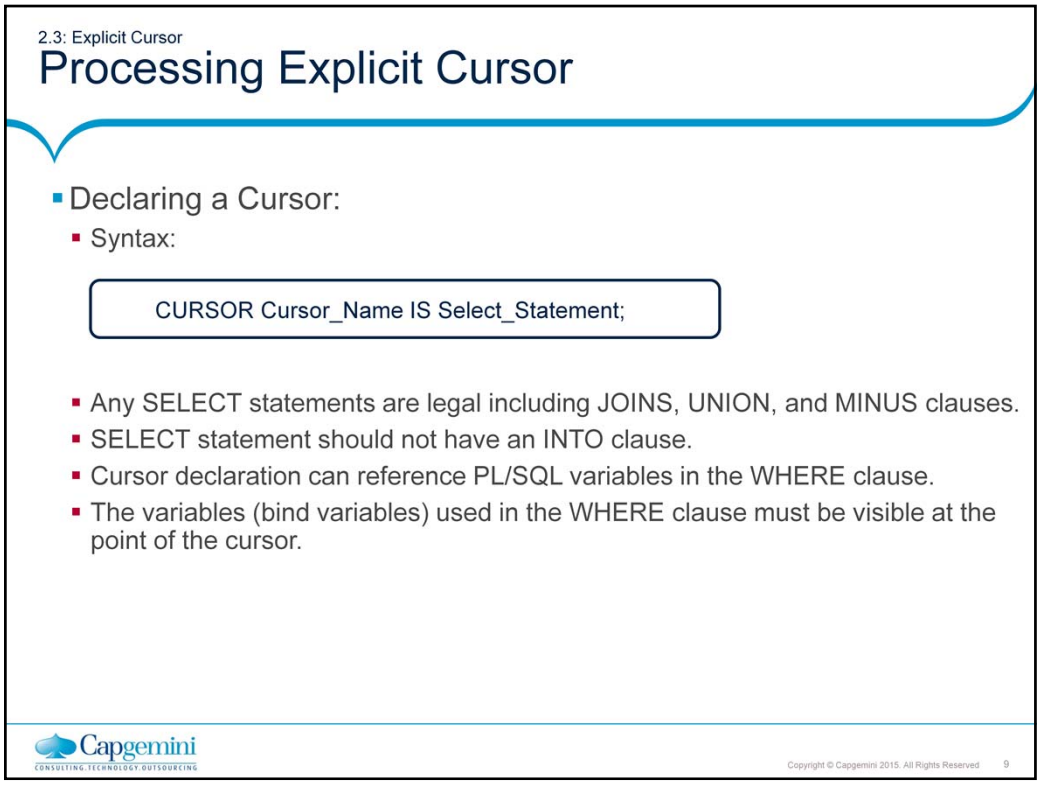
Processing Explicit Cursors:

For example: You might declare cursors like the one given below:

```
DECLARE
  CURSOR c1 IS SELECT
empno, ename, job, sal
FROM emp
    WHERE sal > 2000;
  CURSOR c2 RETURN
dept%ROWTYPE IS
    SELECT * FROM dept
WHERE deptno = 10;
```

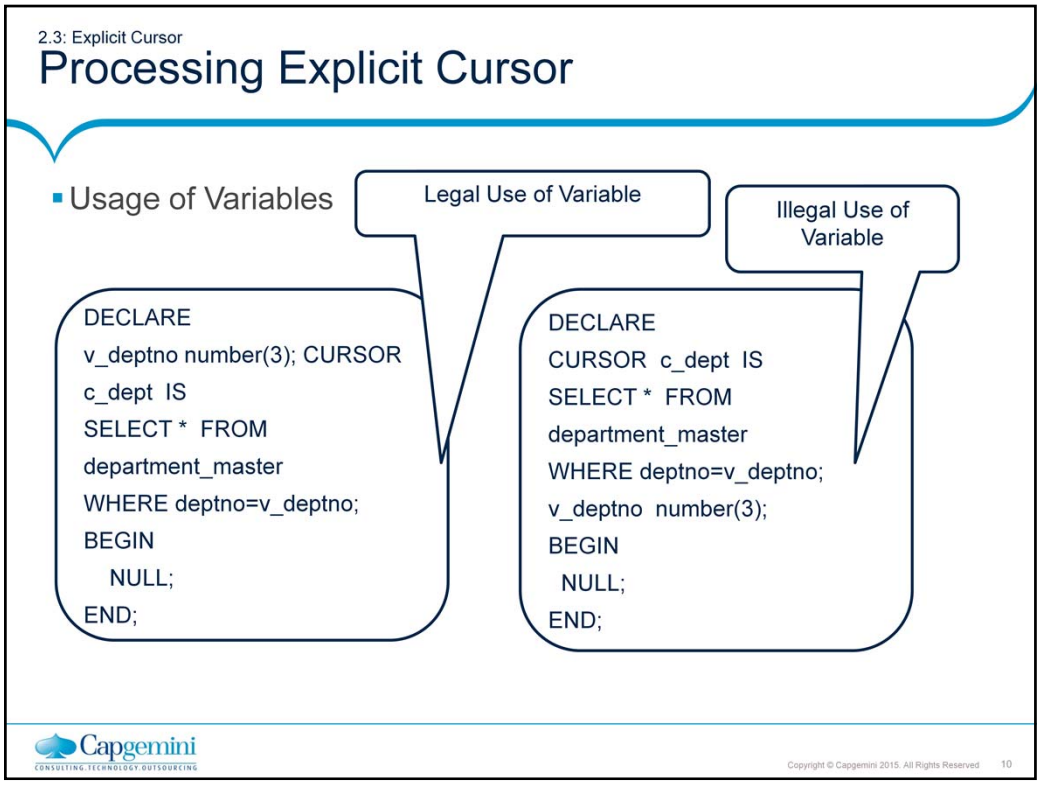
The cursor is not a PL/SQL variable; you cannot assign values to a cursor or use it in an expression. Cursors and variables follow the same scoping rules. Naming cursors after database tables is possible, however, it is not recommended.

contd.



Processing Explicit Cursors:

Declaring a Cursor (contd.)
contd.



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:

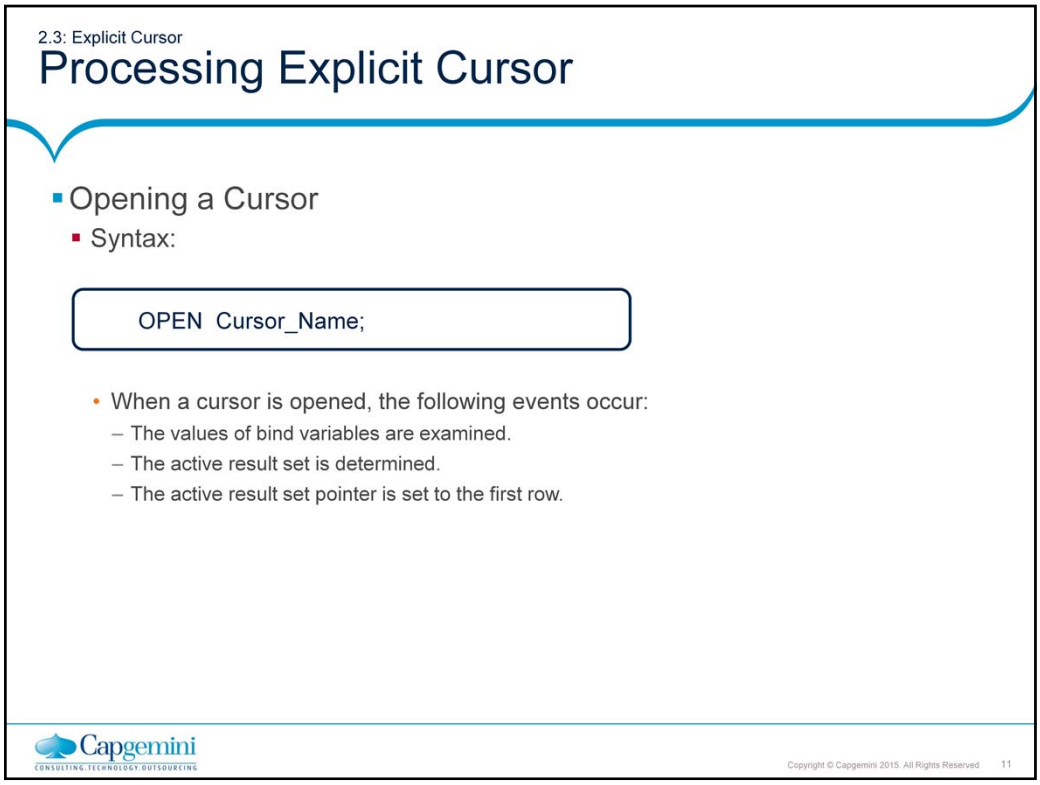
Opening a Cursor

•Opening the cursor executes the query and identifies the result set, which consists of all rows that meet the query search criteria.

•For cursors declared using the FOR UPDATE clause, the OPEN statement also locks those rows. An example of the OPEN statement follows:

```
DECLARE
  CURSOR c1 IS SELECT
ename, job FROM emp
WHERE sal < 3000;
  ...
BEGIN
  OPEN c1;
  ...
END;
```

Rows in the result set are retrieved by the FETCH statement, not when the OPEN statement is executed.



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.

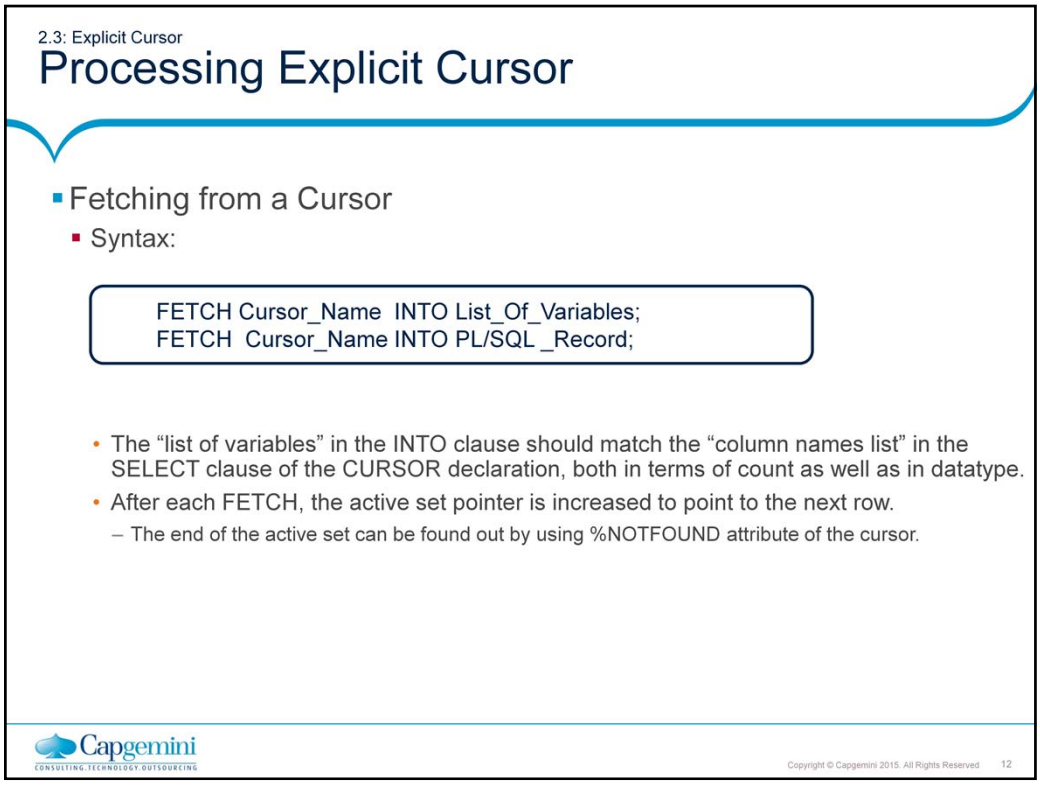
Processing Explicit Cursors:

Fetching with a Cursor
•Unless you use the BULK COLLECT clause the FETCH statement retrieves the rows in the result set one at a time. Each fetch retrieves the current row and advances the cursor to the next row in the result set.

•You can store each column in a separate variable, or store the entire row in a record that has the appropriate fields (usually declared using %ROWTYPE):

-- This cursor queries 3 columns.
-- Each column is fetched into a separate variable.
FETCH c1 INTO my_empno, my_ename, my_deptno;
-- This cursor was declared as SELECT * FROM employees.
-- An entire row is fetched into the my_employees record, which
-- is declared with the type employees%ROWTYPE.
FETCH c2 INTO my_employees;

contd.



Processing Explicit Cursors: Fetching from Cursor:

Processing Explicit Cursors: Fetching with a Cursor (contd.)

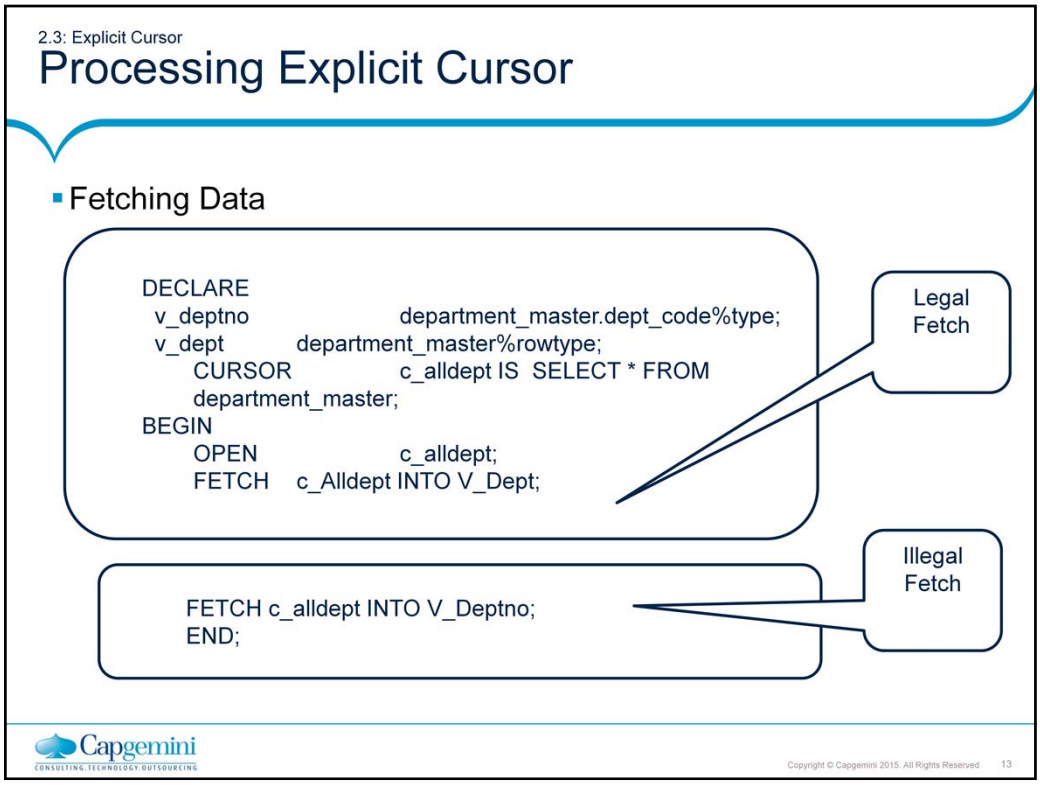
•Typically, you use the FETCH statement in the following way:

```
LOOP
  FETCH c1 INTO
my_record;
  EXIT WHEN
c1%NOTFOUND;
  -- process data record
END LOOP;
```

•The query can reference PL/SQL variables within its scope. Any variables in the query are evaluated only when the cursor is opened. In the following example, each retrieved salary is multiplied by 2, even though factor is incremented after every fetch:

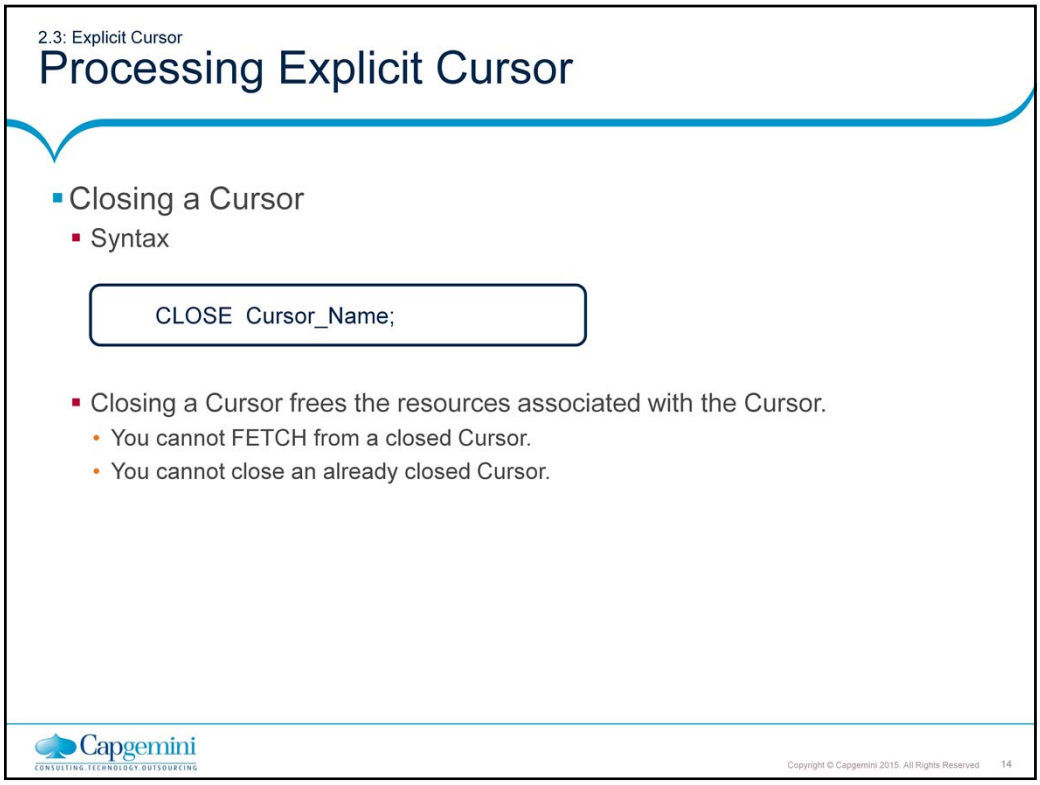
```
DECLARE
 my_sal
employees.salary%TYPE;
 my_job
employees.job_id%TYPE;
 factor INTEGER := 2;
 CURSOR c1 IS
  SELECT factor*salary
FROM employees WHERE
job_id = my_job;
BEGIN
  OPEN c1;  -- here factor
equals 2
  LOOP
    FETCH c1 INTO my_sal;
    EXIT WHEN
c1%NOTFOUND;
    factor := factor + 1;  --
does not affect FETCH
  END LOOP;
END;
/
```

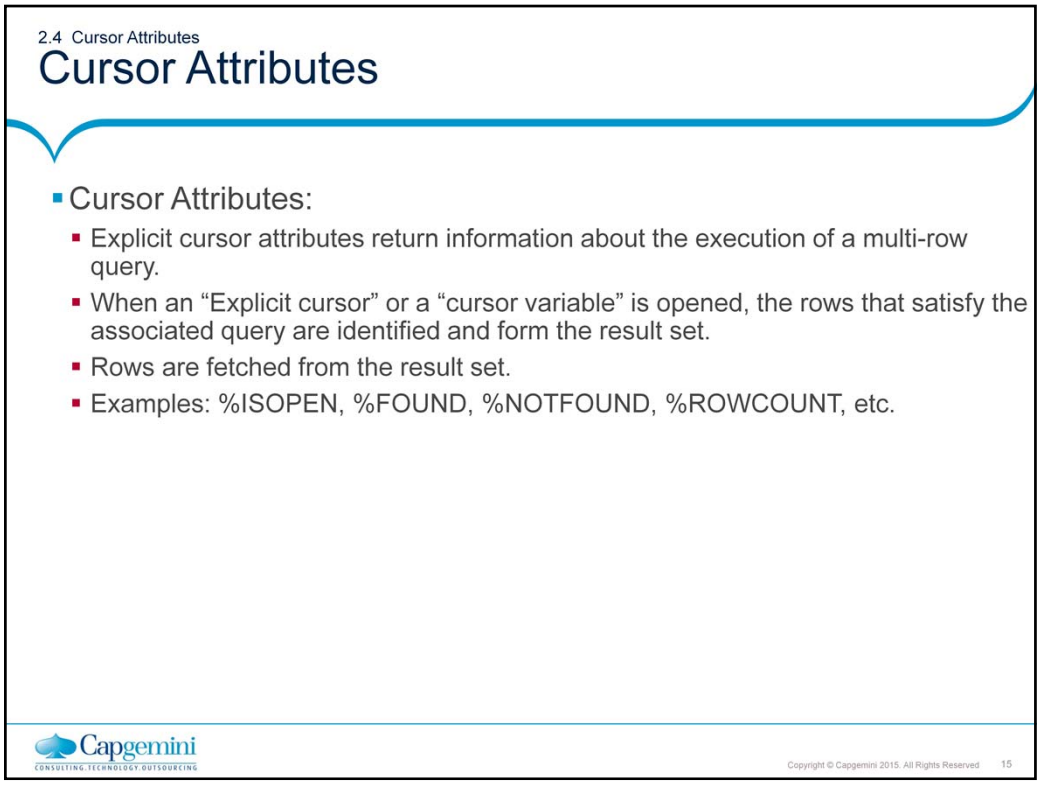
•you can use a different INTO list on separate fetches with the same cursor. Each fetch retrieves another row and assigns values to the target variables

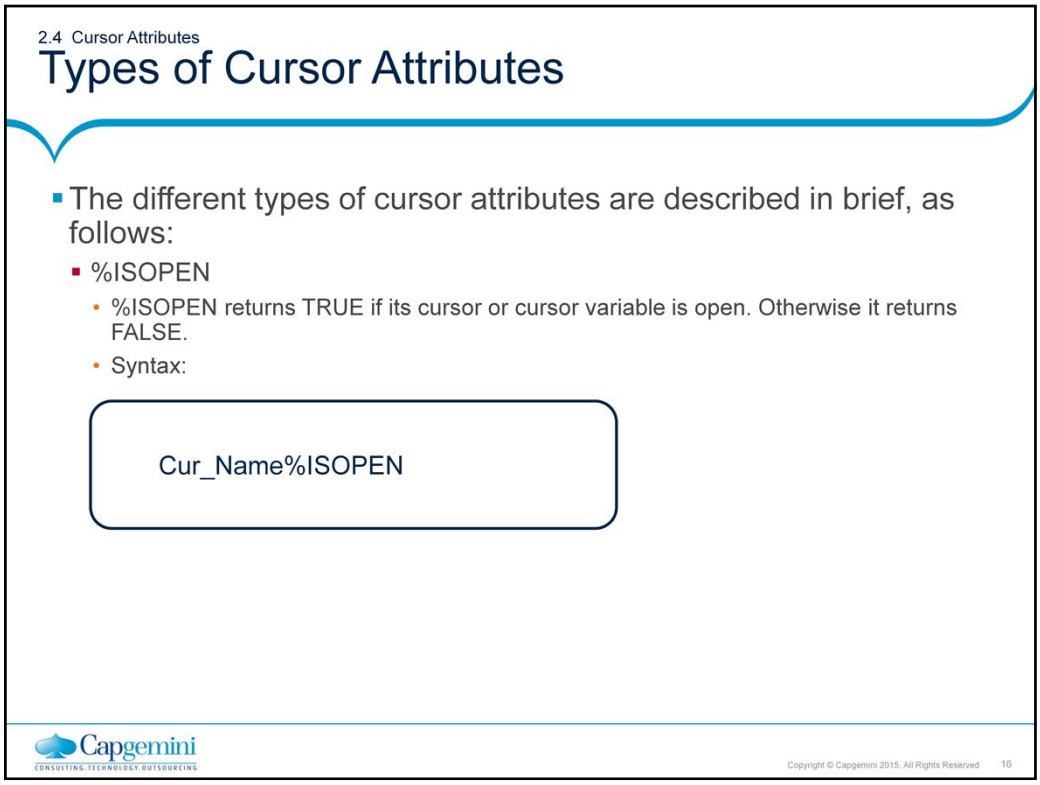


Processing Explicit Cursors: Fetching from Cursor:

- 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.

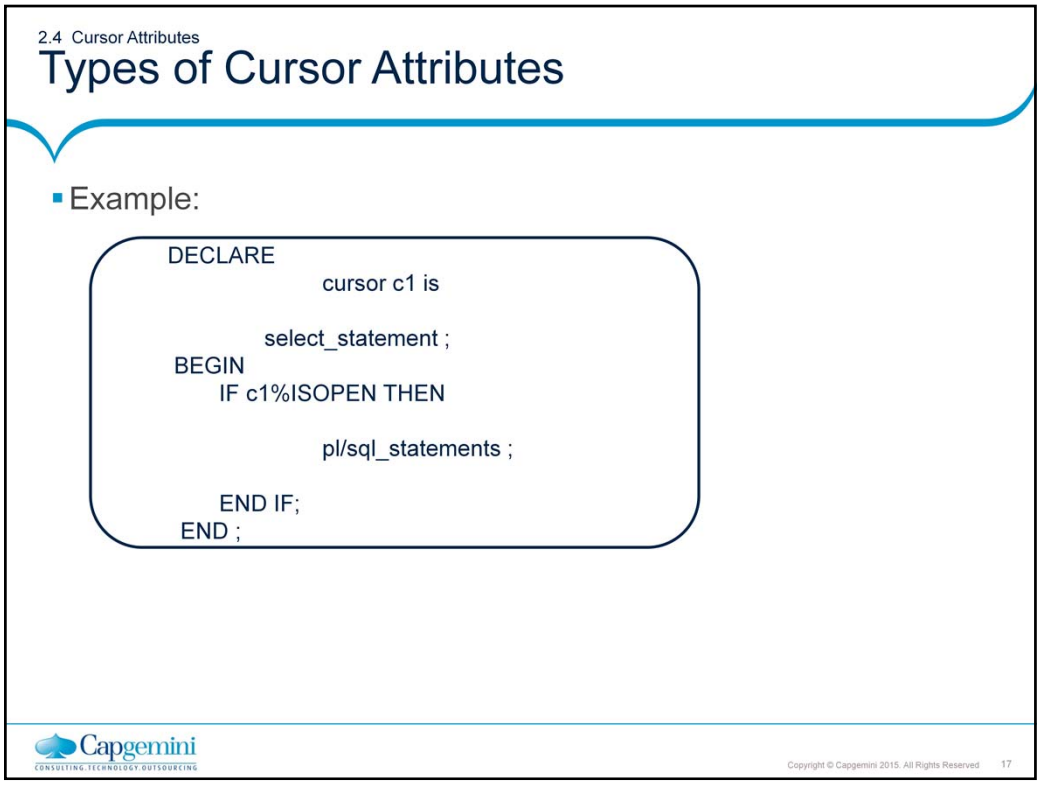

2.3: Explicit Cursor
Processing Explicit Cursor
Closing a Cursor
Syntax
CLOSE Cursor_Name;
Closing a Cursor frees the resources associated with the Cursor.
You cannot FETCH from a closed Cursor.
You cannot close an already closed Cursor.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini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


2.4 Cursor Attributes
Cursor 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

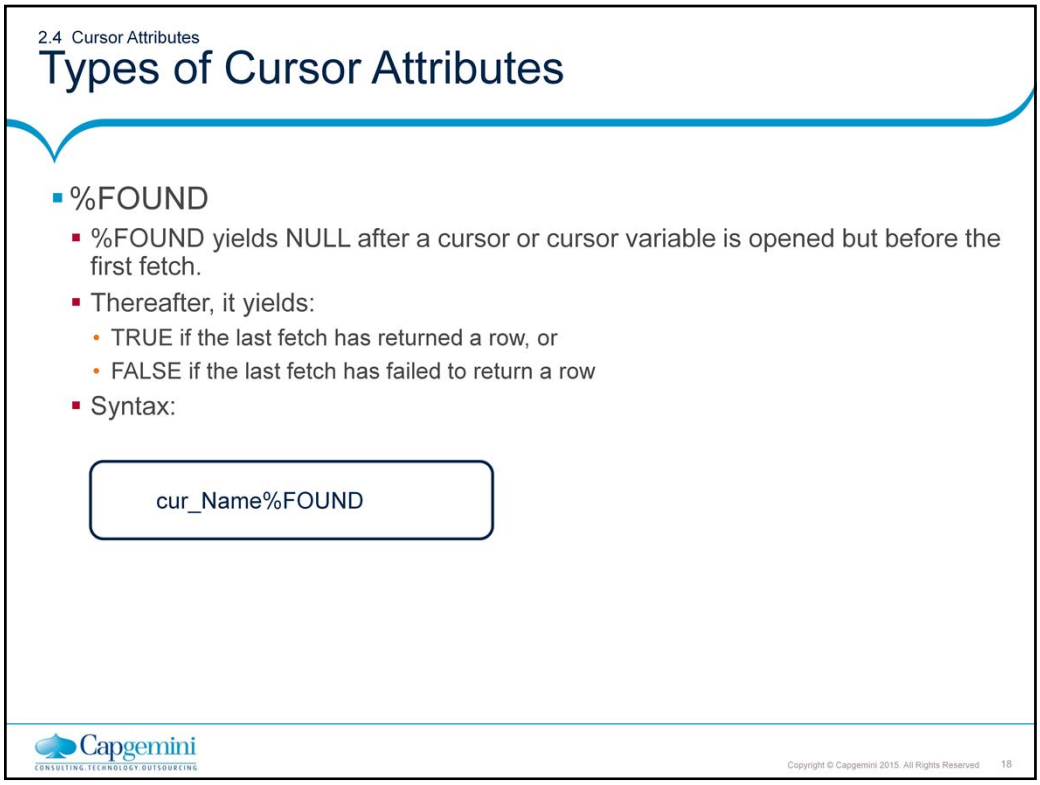
- This attribute is used to check the open/close status of a Cursor.
- If the Cursor is already open, the attribute returns TRUE.
- Oracle closes the SQL cursor automatically after executing its associated SQL statement. As a result, %ISOPEN always yields FALSE for Implicit cursor.



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

2.4 Cursor Attributes

Types of Cursor Attributes

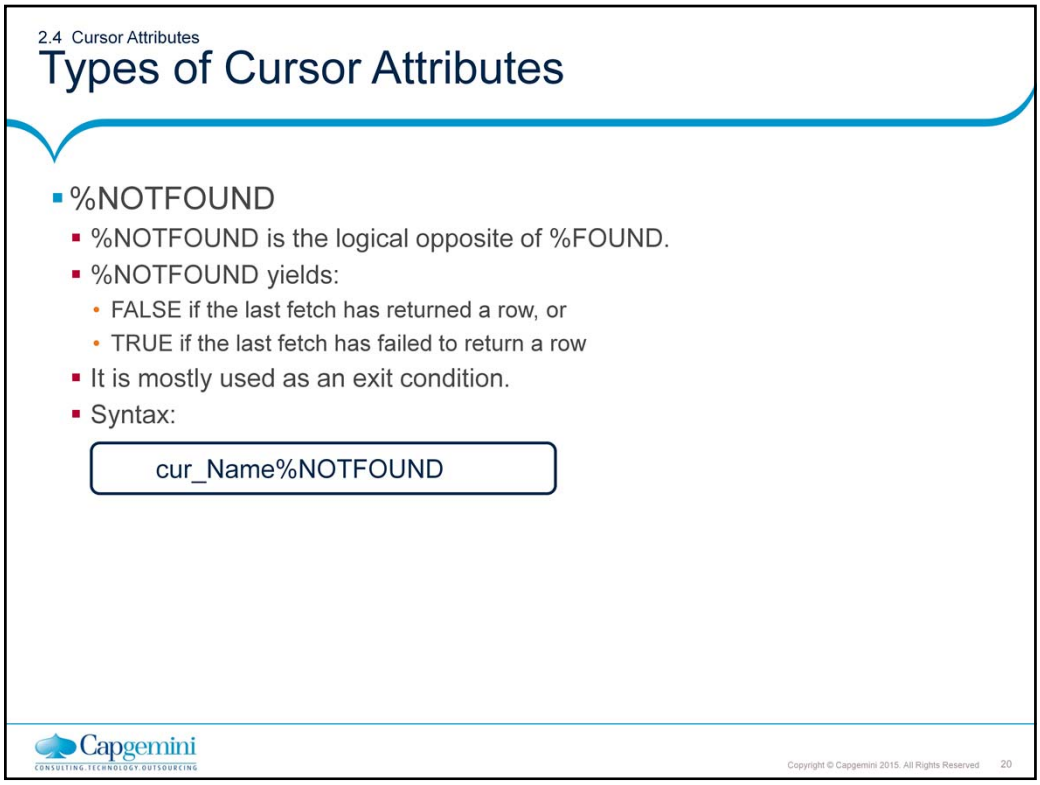
- Example:

```
DECLARE section;
   open c1 ;
   fetch c1 into var_list ;
IF c1%FOUND THEN

      pl/sql_statements ;
END IF ;
```

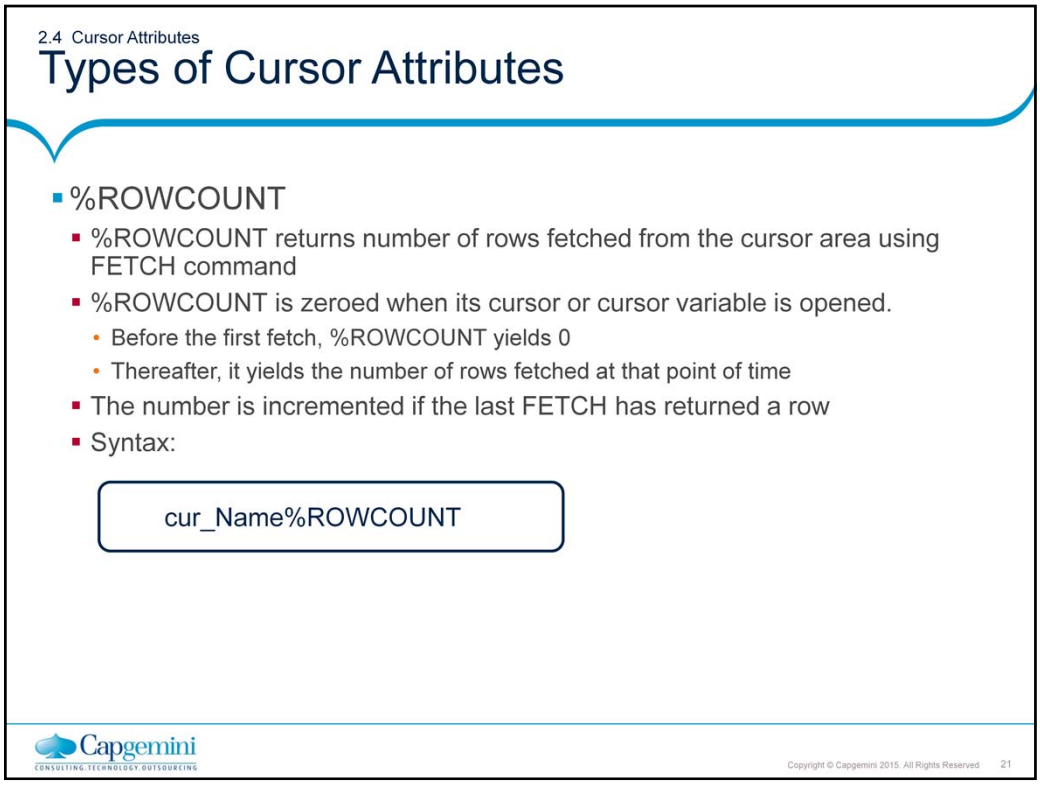
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 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 ;
```

2.4 Cursor Attributes

Summary of Cursor Attributes

Cursor Attribute	Return Type	Description	Syntax
%ISOPEN	Boolean	returns TRUE if its cursor or cursor variable is open. Otherwise it returns FALSE	cur_Name%ISOPEN
%FOUND	Boolean	returns TRUE if the last fetch has returned a row, or FALSE if the last fetch has failed to return a row	cur_Name%FOUND
%NOTFOUND	Boolean	returns FALSE if the last fetch has returned a row, or TRUE if the last fetch has failed to return a row	cur_Name%NOTFOUND
%ROWCOUNT	Nunber	Before the first fetch, %ROWCOUNT yields 0 Thereafter, it yields the number of rows fetched at that point of time	cur_Name%ROWCOUNT

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

2.5 Processing 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

 23

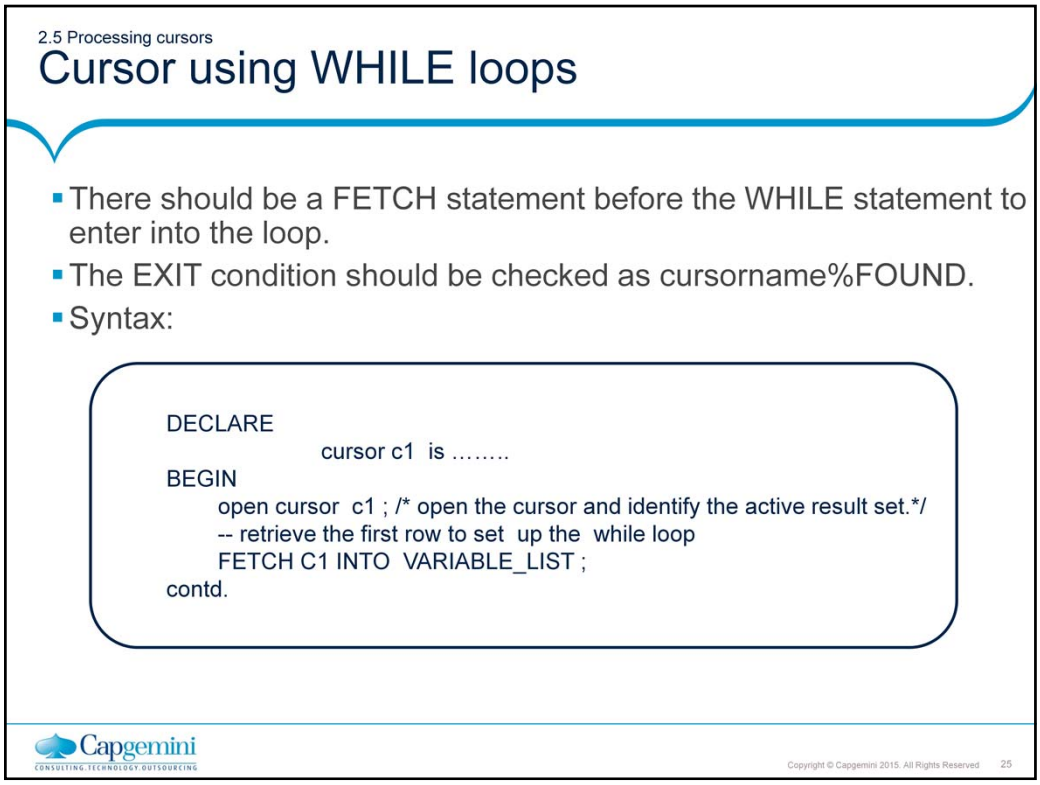
2.5 Processing cursors

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

 24



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.

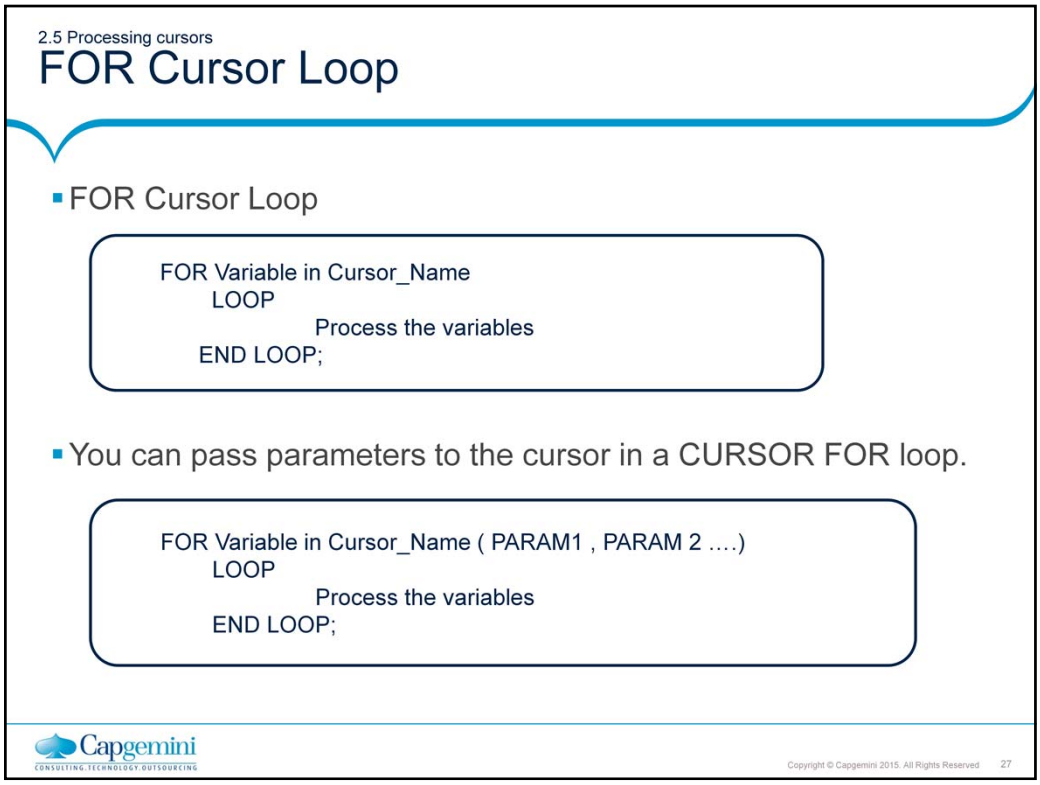
2.5 Processing cursors

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

 26



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.

2.5 Processing cursors

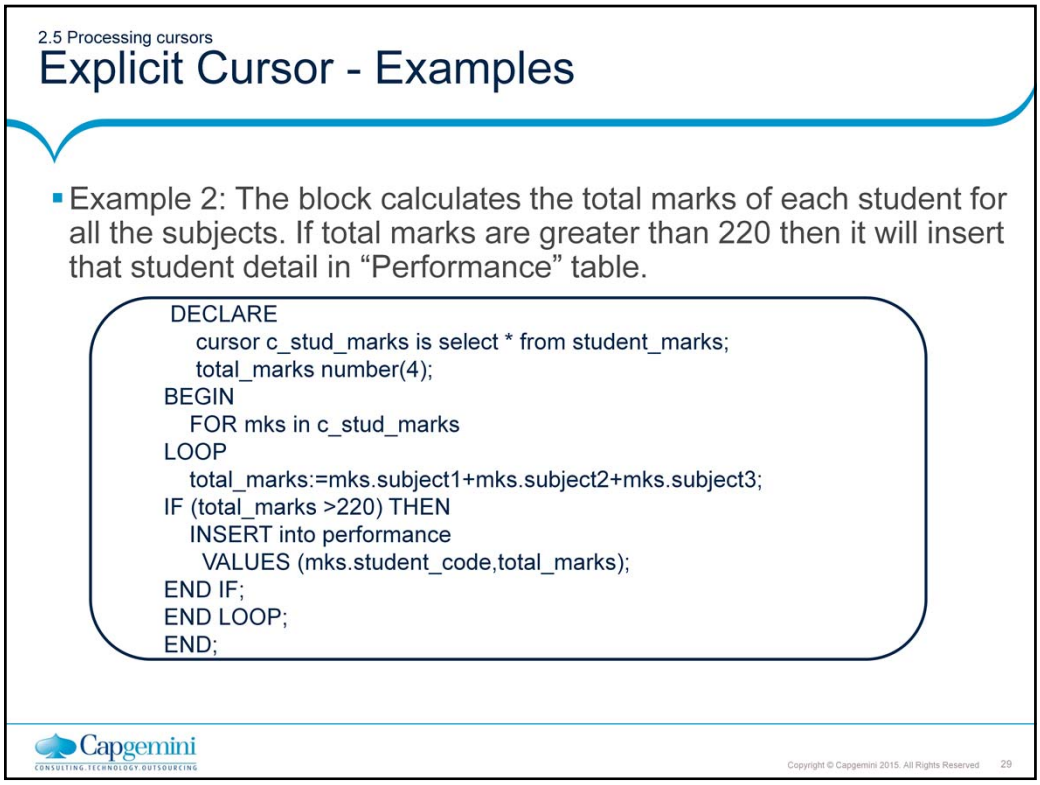
Explicit Cursor - Examples

- 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

 28



In the above example “Performance” is a user defined 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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

Answers for Review Questions:

Question 1:
Answer: False

Question 2:
Answer: False

Question 3:
Answer: CURSOR FOR

Review – Questions

- Question 1 : %COUNT returns number of rows fetched from the cursor area by using FETCH command.
  - True / False
- Question 2: Implicit SQL cursor is opened or closed by the program.
  - True / False
- Question 3: PL/SQL provides a shortcut via a ___ Loop, which implicitly handles the cursor processing.