# *Database Management II (420-D20-HR)*

# *Lab 3 – PL/SQL Basic Controls*

Date assigned: Tuesday, Feb 8, 2017

Date due: **Tuesday, Feb 8, 2017, 17:50**

**An easy lab as you still need to finish Lab 2 if not already done.**

**Estimated time to completion: < 1 hour**

**Objectives:**

At the end of this lab you will be able to:

* Use iteration control structures in PL/SQL
* Use PL/SQL Control structures, such as IF, CASE
* Use implicit cursor attribute to test the results of SQL commands

**References:**

Class notes and examples (S04, S05)

Oracle PL/SQL documentation

**Marking and Time management:**

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Question | Mark | Effort (minutes) |
| 1. Basic | 1 SCD | 8 | 0 |
|  | 2 CASE | 5 | 0 |
|  | 3 Fibonacci | 8 | 0 |
|  | 4 reverse loop | 5 | 0 |
| 1. Implicit Cursors | 1 Code | 6 | 0 |
|  | 1 Output summary | 6 | 0 |
| 1. Test data, then code | 1 Test data | 4 | 0 |
|  | 1 Code & Output | 8 | 0 |
| English and handed in properly |  | 5 | 0 |
| Totals |  | 55 | 0 |

**To Start:**

1. Rename this document to ***username*\_D20\_L03\_Basic\_Controls.docx**
2. Start **SQL Developer** and connect to your Oracle account.

**To be handed in:**

1. ***username\_*D20\_L03\_Basic\_Controls.docx** containing your responses created for this lab should uploaded to **Moodle**. Please remember to fill in the Effort section of the Marking Rubric.

**For each of the blocks created for the following problems, include a comment block at the top of the block with the question number, your name and a brief description of what the block does. Insert comments throughout to explain the steps.**

**Use ISO/ANSI standards joins for all joins.**

**Name all identifiers according to the naming standards shown below.**

**Format all blocks using the SQL Developer Formatter.**

**Provide sample output.**

**Naming Standards:**

|  |  |
| --- | --- |
| **Identifier type** | **Prefix** |
| local variables | lv\_ |
| local constant | lc\_ |
| local record | lrec\_ |
| local cursor | lcur\_ |
| local type | ltyp\_ |

# Basic Controls

## Write an anonymous block that:

* + Prompts the user for a number
  + Prompts the user for 'S', 'C' or 'D'.
  + Use a nested if statement to calculate and display:
    - the square of the number if the user enters 'S'
    - the cube of the number if the user enters 'C'
    - the double of the number if the user enters 'D'
    - a message stating that you don't understand the option . if the user enters anything else,.
  + The block should work for both upper and lower case values for the calculation type.

**SQL:**

**DECLARE**

**lv\_num NUMBER;**

**lv\_char VARCHAR(25);**

**BEGIN**

**lv\_num := &EnterNumber;**

**lv\_char := '&EnterCharacter';**

**IF (upper(lv\_char) = 'S')**

**THEN DBMS\_OUTPUT.PUT\_LINE(lv\_num \* lv\_num);**

**ELSIF (upper(lv\_char) = 'C')**

**THEN DBMS\_OUTPUT.PUT\_LINE(lv\_num \* lv\_num \* lv\_num);**

**ELSIF (upper(lv\_char) = 'D')**

**THEN DBMS\_OUTPUT.PUT\_LINE(lv\_num \* 2);**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('You obviously didnt get it...');**

**END IF;**

**END;**

**Sample output:**

4

## Repeat the previous question using a CASE statement to determine what the user entered.

**SQL:**

**DECLARE**

**lv\_num NUMBER;**

**lv\_char VARCHAR(25);**

**BEGIN**

**lv\_num := &EnterNumber;**

**lv\_char := '&EnterCharacter';**

**CASE upper(lv\_char)**

**WHEN 'S'**

**THEN DBMS\_OUTPUT.PUT\_LINE(lv\_num \* lv\_num);**

**WHEN 'C'**

**THEN DBMS\_OUTPUT.PUT\_LINE(lv\_num \* lv\_num \* lv\_num);**

**WHEN 'D'**

**THEN DBMS\_OUTPUT.PUT\_LINE(lv\_num \* 2);**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('You obviously didnt get it...');**

**END CASE;**

**END;**

**Sample output:**

4

## Output the first 20 numbers from the Fibonacci Sequence (look it up if you don’t know what this is). Output all the values on a single line. The output should look like:

**0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181**

**end of Fibonacci Sequence**

## Hint: To print without a carriage return at the end of the line, use **dbms\_output.put()** instead of **dbms\_output.put\_line()**. When you want to go to the next line, use **dbms\_output.new\_line()**.

**SQL:**

**DECLARE**

**lv\_fib1 number := 0;**

**lv\_fib2 number := 1;**

**lv\_tmp number;**

**lv\_count number := 0;**

**BEGIN**

**WHILE lv\_count < 20 LOOP**

**DBMS\_OUTPUT.PUT(lv\_fib1 || ' ');**

**lv\_tmp := lv\_fib2;**

**lv\_fib2 := lv\_fib1 + lv\_fib2;**

**lv\_fib1 := lv\_tmp;**

**lv\_count := lv\_count +1;**

**END LOOP;**

**DBMS\_OUTPUT.NEW\_LINE();**

**END;**

**Sample output:**

**0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181**

## Print the numbers from 10 to 1 using a FOR loop.

## Your output should look like:

**The numbers from 10 to 1 are:**

**10 9 8 7 6 5 4 3 2 1**

**SQL:**

**BEGIN**

**FOR lv\_count IN REVERSE 0..10 LOOP**

**DBMS\_OUTPUT.PUT(lv\_count || ' ');**

**END LOOP;**

**DBMS\_OUTPUT.NEW\_LINE();**

**END;**

**Sample output:**

**10 9 8 7 6 5 4 3 2 1 0**

# Implicit Cursors and SQL Statements

***Purpose:*** Learn to use implicit cursor attributes to check the results of an SQL INSERT, UPDATE or DELETE statement

|  |  |
| --- | --- |
| Every SQL statement has associated with it ***implicit*** cursor attributes that contain information about the statement just completed. The implicit cursor attributes are: | |
| **Cursor Attribute** | **Description** |
| **SQL%ROWCOUNT** | contains the number of rows inserted, updated or deleted |
| **SQL%FOUND** | a Boolean attribute that contains true if any rows were updated, deleted or inserted |
| **SQL%NOTFOUND** | a Boolean attribute that contains true if any rows were updated, deleted or inserted |

## It has been decided that one faculty member will do the advising for all students in a particular major. You have written the following PL/SQL block to do the appropriate updates:

**BEGIN**

**UPDATE iu\_**student

**SET** facultyid = &facultyid

**WHERE** majorid = &majorid;

**END**;

Run the block for facultyid 345 and majorid 100. What happens? Do you know if any rows were updated? Do you know how many rows were updated?

Add code to test the SQL attributes (%ROWCOUNT, %FOUND, %NOTFOUND) and fill in the answer table below.

Remember to roll back the transaction after each test run.

**SQL:**

**BEGIN**

**UPDATE iu\_student**

**SET facultyid = &facultyid**

**WHERE majorid = &majorid;**

**DBMS\_OUTPUT.PUT\_LINE('Num Rows updated? ' || SQL%ROWCOUNT);**

**IF SQL%FOUND**

**THEN DBMS\_OUTPUT.PUT\_LINE('Any Rows updated? TRUE');**

**ELSE DBMS\_OUTPUT.PUT\_LINE('Any Rows updated? FALSE');**

**END IF;**

**IF SQL%NOTFOUND**

**THEN DBMS\_OUTPUT.PUT\_LINE('No Rows updated? TRUE');**

**ELSE DBMS\_OUTPUT.PUT\_LINE('No Rows updated? FALSE');**

**END IF;**

**END;**

**Output summary:**

**345, 100**

**Num Rows updated? 1**

**Any Rows updated? TRUE**

**No Rows updated? FALSE**

**111, 200**

**Num Rows updated? 6**

**Any Rows updated? TRUE**

**No Rows updated? FALSE**

**222, 300**

**Num Rows updated? 0**

**Any Rows updated? FALSE**

**No Rows updated? TRUE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FacultyId | Major | # rows updated | Any rows updated (T/F) | No rows updated (T/F) |
| 345 | 100 | 1 | True | False |
| 111 | 200 | 6 | True | False |
| 222 | 300 | 0 | False | True |

# Test data, code

***Purpose:*** refresh on how to determine test data for testing your code. More practice with code controls.

You will write tests and code to determine if a given year is a leap year or not.

A leap year is divisible by 4 but not by 100, or it is divisible by 400.

## Complete the following Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Conditions** | | | **Input data** | **Expected Results** |
| **Year divisible by 4** | **Year divisible by 100** | **Year divisible by 400** | **Year** | **Is leap year?** |
| 1. | N | - | - | 1999 | No |
| 2. | Y | N | - | 2004 | Yes |
| 3. | Y | Y | N | 2100 | No |
| 4. | Y | Y | Y | 2400 | yes |

## Code and test a PL/SQL block. It will prompt for a year, and output whether it is a leap year or not.

Hint: the function MOD(n,d) divided n by d and returns the integer remainder.

**SQL:**

**DECLARE**

**type lt\_years IS VARRAY(4) of number;**

**tests lt\_years;**

**BEGIN**

**tests := lt\_years(1999, 2004, 1900, 2400);**

**FOR x IN tests.first..tests.last LOOP**

**IF MOD(tests(x), 4) = 0 THEN**

**IF MOD(tests(x), 100) = 0 THEN**

**IF MOD(tests(x), 400) = 0 THEN**

**DBMS\_OUTPUT.PUT\_LINE(tests(x) || ' is a leap year');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE(tests(x) || ' is not a leap year');**

**END IF;**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE(tests(x) || ' is a leap year');**

**END IF;**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE(tests(x) || ' is not a leap year');**

**END IF;**

**END LOOP;**

**END;**

**Output (run all 4 of your tests and provide output):**

**1999 is not a leap year**

**2004 is a leap year**

**1900 is not a leap year**

**2400 is a leap year**

# Assessment

1. What did you learn in completing this lab?

That recursion is difficult and not worth it now in SQL

1. What did you have difficulty with?

The testing part

1. What did you do well?

Fibonacci

1. How many hours did you spend in completing this lab?

1 1/2

1. What took you the most time?

# Part B