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

# *Lab 5 –Procedures and Functions*

Date assigned: Tuesday, March 7, 2017

Date due: **Tuesday, March 7, 2017, 5:50pm**

**Objectives:**

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

* code and test PL/SQL procedures using IN and OUT parameters.
* use default parameters in PL/SQL procedures
* create and use PL/SQL functions

**To be handed in:**

1. The ***username*\_D20\_\_Procedures.docx** should be uploaded to **Moodle**.

**Reference:**

* S08, N08 found on Moodle

**To Start:**

1. Rename this document to ***username*\_D20\_\_Procedures*.*docx**.
2. Suggest you reset your database as you may have left modifications in the labs and class exercises. (Run the Shah Tables database script from Moodle)

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

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

**Format all blocks using the SQL Navigator Formatter**

**Provide output from the execution of the procedures and testing.**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Mark** | **Out of** |
| A – Intro to Procedures |  | 5 |
| B – Output Parameters |  | 10 |
| C – Default Parameters |  | 10 |
| D – Functions -1 |  | 8 |
| -2 |  | 8 |
| Organization - Blocks Formatted and naming standards used, assessment |  | 5 |
| **Total** |  | **46** |

**Naming Standards:**

|  |  |  |
| --- | --- | --- |
| **Identifier type** | **Prefix** | **Suffix** |
| type | **typ\_** |  |
| local variable | **lv\_** |  |
| local record | **lrec\_** |  |
| local cursor | **lcur\_** |  |
| exception | **e\_** |  |
| procedure name |  | **\_sp** |
| IN parameter | **pv\_** | **\_i** |
| OUT parameter | **pv\_** | **\_o** |
| IN OUT parameter | **pv\_** | **\_io** |
|  |  |  |

# Introduction to Procedures

***Purpose:*** Learn to code and execute a simple procedure using input parameters.

***To Do:***

## The following procedure moves a Naman-Navan employee from one department to another. Code the procedure in SQL Developer:

**CREATE OR REPLACE PROCEDURE d20\_L05\_transfer\_emp\_sp IS**

e\_no\_such\_dept **EXCEPTION**;

e\_null\_dept **EXCEPTION**;

**PRAGMA EXCEPTION\_INIT** (e\_no\_such\_dept, -2291);

**PRAGMA EXCEPTION\_INIT** (e\_null\_dept, -1407);

**BEGIN**

**UPDATE** nn\_employee

**SET** deptid = &deptid

**WHERE** employeeid = &employeeid;

**IF SQL**%**FOUND THEN**

**DBMS\_OUTPUT**.put\_line

('Update successful. Employee has been transferred');

**ELSE**

**DBMS\_OUTPUT**.put\_line

('Invalid employee number - does not exist.');

**END IF**;

**EXCEPTION**

**WHEN** e\_no\_such\_dept **THEN**

**DBMS\_OUTPUT**.put\_line ('Invalid dept id - does not exist');

**WHEN** e\_null\_dept **THEN**

**DBMS\_OUTPUT**.put\_line ('Deptid must be non-null.');

**END d20\_L05\_transfer\_emp\_sp**;

## Run the procedure. Use any values for department or employee. What happens? Do you get any output statements? What's the last line in the Script Output window?

dept: 123 emp: 111

Procedure D20\_L05\_TRANSFER\_EMP\_SP compiled

Nothing output really.

Expand the **Procedures** list in the **Connections** panel. Open **D20\_L05\_TRANSFER\_EMP\_SP**. What do you see? The same thing with &deptid and &empid replaced with what I entered

## Close **D20\_L05\_TRANSFER\_EMP\_SP**.

## Now we want to use input parameters instead of substitution variables for the **deptid** and **employeeid**. Add the following between the procedure name and the word **IS**:

**( pv\_deptid\_i nn\_dept.deptid%TYPE,**

**pv\_employeeid\_i nn\_employee.employeeid%TYPE )**

## Replace **&deptid** with **pv\_deptid\_i** and **&employeeid** with **pv\_employeeid\_i** in the **update** statement.

## Compile the procedure.

## You have still not run the procedure. To run the procedure, right-click on the procedure name in the Connections panel and select **Run**. An anonymous PL/SQL block will open with the following code:

DECLARE

PV\_DEPTID\_I NUMBER;

PV\_EMPLOYEEID\_I NUMBER;

BEGIN

PV\_DEPTID\_I := NULL;

PV\_EMPLOYEEID\_I := NULL;

**D20\_L05\_TRANSFER\_EMP\_SP(**

**PV\_DEPTID\_I => PV\_DEPTID\_I,**

**PV\_EMPLOYEEID\_I => PV\_EMPLOYEEID\_I**

**);**

END;

## Replace the NULL in the assignment statements after BEGIN with appropriate values for the deptid and employeeid. Click **OK**. What happens?

## Connecting to the database pdumaresq.

## ORA-01438: value larger than specified precision allowed for this column

## ORA-06512: at "PDUMARESQ.D20\_L05\_TRANSFER\_EMP\_SP", line 11

## ORA-06512: at line 8

## Process exited.

## Disconnecting from the database pdumaresq.

## Rollback any changes you made with the procedure.

## You can also run the procedure from SQL Developer using the CALL statement. Code the following and run it:

**call d20\_L05\_transfer\_emp\_sp(20, 111)**

## Rollback your changes and copy your completed procedure to ***username\_*D20\_L05\_Procedures.docx**.

CREATE OR REPLACE PROCEDURE d20\_L05\_transfer\_emp\_sp (

pv\_deptid\_i nn\_dept.deptid%TYPE,

pv\_employeeid\_i nn\_employee.employeeid%TYPE

)

IS

e\_no\_such\_dept EXCEPTION;

e\_null\_dept EXCEPTION;

PRAGMA EXCEPTION\_INIT (e\_no\_such\_dept, -2291);

PRAGMA EXCEPTION\_INIT (e\_null\_dept, -1407);

BEGIN

UPDATE nn\_employee

SET deptid = pv\_deptid\_i

WHERE employeeid = pv\_employeeid\_i;

IF SQL%FOUND THEN

DBMS\_OUTPUT.put\_line

('Update successful. Employee has been transferred');

ELSE

DBMS\_OUTPUT.put\_line

('Invalid employee number - does not exist.');

END IF;

EXCEPTION

WHEN e\_no\_such\_dept THEN

DBMS\_OUTPUT.put\_line ('Invalid dept id - does not exist');

WHEN e\_null\_dept THEN

DBMS\_OUTPUT.put\_line ('Deptid must be non-null.');

END d20\_L05\_transfer\_emp\_sp;

call d20\_L05\_transfer\_emp\_sp(20, 111)

# Output Parameters

***Purpose:*** Learn to use OUT parameters in an Oracle procedure

***To Do:***

## Enter following procedure:

**CREATE OR REPLACE PROCEDURE** D20\_L05\_my\_first\_procedure\_sp(

pv\_courseid\_i iu\_course.courseid%**TYPE**,

pv\_course\_title\_o iu\_course.title%**TYPE**

)

**IS**

**BEGIN**

**SELECT** title

**INTO** pv\_course\_title\_o

**FROM** iu\_course

**WHERE** courseid = pv\_courseid\_i;

**END** D20\_L05\_my\_first\_procedure\_sp;

What happens when you compile it? \_\_Errors\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Change the second parameter to:

pv\_course\_title\_o **OUT** iu\_course.title%**TYPE**

Compile it again.

***Executing a Procedure:***

To execute your procedure:

1. Include the call in an anonymous block you write:

**DECLARE**

lv\_title iu\_course.title%**TYPE**;

**BEGIN**

**D20\_L05\_**my\_first\_procedure\_sp ('CIS253', lv\_title);

**DBMS\_OUTPUT**.put\_line ('The title for course CIS253 is ' || lv\_title);

**END**;

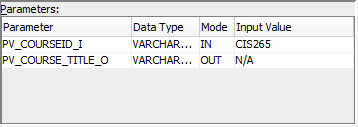
**OR**

2. Run an anonymous block created for you by SQL Developer:

a) Expand the **Procedures** folder in the **Connections** panel.

b) Right-click on **D20\_L05\_my\_first\_procedure\_sp** and select **Run**.

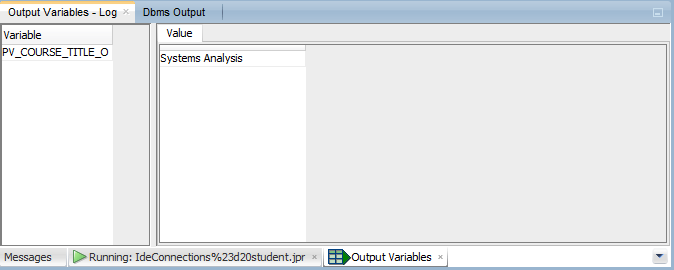
c) In the Run PL/SQL window, go to the PL/SQL block and change the parameter Input Value:



d) Uncomment the DBMS\_OUTPUT statement that prints the output parameter.

e) Click the **OK** button.

f) The results are displayed in **Output Variables** tab:



## Write a procedure called **D20\_L05\_get\_student\_mark\_sp** to accept a student number and a csid and return the student name, course title, final mark and a status for that student in that course. If the student did not complete the course (i.e. is exempt, withdrew, is currently taking or has never taken), the final mark will be null. If either the input studentid or the input csid is invalid, an application error should be raised with an appropriate message.

The status will contain one of the following:

| **Status** | **Meaning** |
| --- | --- |
| *C* | The student has a final mark in the course |
| *T* | The student is registered for the course but does not have a final mark |
| *X* | The student has an exemption for the course. (regstatus is 'X') |
| *W* | The student has withdrawn from the course. (regstatus is 'W') |
| *U* | The student is not taking the course |

Test your procedure for the test cases on the following page.

CREATE OR REPLACE PROCEDURE D20\_L05\_get\_student\_mark\_sp (

pv\_studentid iu\_student.student\_id%TYPE,

pv\_csid iu\_crssection.csid%TYPE,

pv\_studentname OUT iu\_student.first%TYPE, --this will blow up on long names

pv\_courseTitle OUT iu\_course.title%TYPE,

pv\_finalmark OUT iu\_registration.final%TYPE,

pv\_status OUT iu\_registration.regstatus%TYPE

)

AS

DECLARE

lv\_status char;

BEGIN

--should be raising the exceptions, not throwing them in an exception block

SELECT s.first || ' ' || s.last AS name, c.title as course, r.final, r.regstatus

INTO pv\_studentname, pv\_courseTitle, pv\_finalmark, pv\_status, pv\_status

FROM iu\_student s, iu\_course c, iu\_crssection crs, iu\_registration r

WHERE s.studentid = r.studentid

AND r.csid = crs.csid

AND c.courseid = crs.courseid

AND s.studentid = pv\_studentid

AND r.csid = pv\_csid;

if pv\_finalmark is not null

then lv\_status := 'C';

elsif pv\_finalmark is null

then if pv\_status = 'R'

then lv\_status := 'T';

elsif pv\_status = 'X'

then lv\_status := 'X';

elsif pv\_status = 'W'

then lv\_status := 'W';

else

lv\_status := 'U';

end if;

else

lv\_status := 'U';

end if;

pv\_status := lv\_status;

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN DBMS\_OUTPUT.PUT\_LINE();

END D20\_L05\_get\_student\_mark\_sp;

**D20\_L05\_GET\_STUDENT\_MARK\_SP Test Cases**

| **Test Case** | **Input Data** | | **Table Data** | | | **Expected Results** | | | | **Test Results**  **Pass/Fail** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **studentid** | **csid** | **courseid** | **final** | **Reg**  **status** | **Student** | **Title** | **final** | **Status** |  |
| 1. Final is A | 00100 | 1104 | AC101 | A | R | Jose Diaz | Accounting | A | C |  |
| 1. Final is B | 00100 | 1102 | CIS253 | B | R | Jose Diaz | Database Systems | B | C |  |
| 1. Final is D | 00101 | 1102 | CIS253 | D | R | Mickey Tyler | Database Systems | D | C |  |
| 1. Final is F | 00100 | 1101 | CIS265 | F | R | Jose Diaz | Systems Analysis | F | C |  |
| 1. Status is W | 00103 | 1206 | CIS265 | null | W | Deborah Rickles | Systems Analysis | null | W |  |
| 1. Status is X | 00104 | 1206 | CIS265 | null | X | Brian Lee | Systems Analysis | null | X |  |
| 1. Final is null | 00104 | 1207 | LA123 | null | R | Brian Lee | English Literature | null | T |  |
| 1. The student was not registered in the section | 00100 | 1103 | MA150 | - | - | Jose Diaz | College Algebra | null | U |  |
| 1. studentid is invalid | 00604 | 1103 | MA150 | - | - | ORA-20000: 00604 is not a valid studentid. | | | |  |
| 1. csid is invalid | 00104 | 2207 | - | - | - | ORA-20000: 2207 is not a valid csid. | | | |  |

## When you are finished, copy the code to create the procedure below, update the Pass/Fail column of the test table.

**PL/SQL:**

**CREATE OR REPLACE PROCEDURE D20\_L05\_get\_student\_mark\_sp (**

**pv\_studentid iu\_student.student\_id%TYPE,**

**pv\_csid iu\_crssection.csid%TYPE,**

**pv\_studentname OUT iu\_student.first%TYPE,**

**pv\_courseTitle OUT iu\_course.title%TYPE,**

**pv\_finalmark OUT iu\_registration.final%TYPE,**

**pv\_status OUT iu\_registration.regstatus%TYPE**

**)**

**AS**

**DECLARE**

**lv\_status char;**

**BEGIN**

**SELECT s.first || ' ' || s.last AS name, c.title as course, r.final, r.regstatus**

**INTO pv\_studentname, pv\_courseTitle, pv\_finalmark, pv\_status, pv\_status**

**FROM iu\_student s, iu\_course c, iu\_crssection crs, iu\_registration r**

**WHERE s.studentid = r.studentid**

**AND r.csid = crs.csid**

**AND c.courseid = crs.courseid**

**AND s.studentid = pv\_studentid**

**AND r.csid = pv\_csid;**

**if pv\_finalmark is not null**

**then lv\_status := 'C';**

**elsif pv\_finalmark is null**

**then if pv\_status = 'R'**

**then lv\_status := 'T';**

**elsif pv\_status = 'X'**

**then lv\_status := 'X';**

**elsif pv\_status = 'W'**

**then lv\_status := 'W';**

**else**

**lv\_status := 'U';**

**end if;**

**else**

**lv\_status := 'U';**

**end if;**

**pv\_status := lv\_status;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND**

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

**END D20\_L05\_get\_student\_mark\_sp;**

**Sample output:**

# Default Parameters

***Objectives*:** Learn to use default parameters in a PL/SQL procedure.

***To Do:***

* 1. Write a procedure called **D20\_L05\_give\_raise\_sp** that gives a raise to employees at Naman-Navan. Your procedure should contain only **one** update statement.

**Parameters**:

pv\_percentage\_raise\_i: the raise percentage (e.g. 25 for 25%)

pv\_deptid\_i: the department (default null). If coded, all employees in this department are given a raise.

pv\_positionid\_i: the positionid (default null). If coded, all employees in this position are given a raise.

pv\_num\_employees\_o: the number of employees that were given a raise.

**PL/SQL:**

**CREATE OR REPLACE PROCEDURE D20\_L05\_give\_raise\_sp (**

**pv\_percentage\_raise\_i in number,**

**pv\_deptid\_i in number DEFAULT null,**

**pv\_positionid\_i in number DEFAULT null,**

**pv\_num\_employees\_o OUT number**

**)**

**AS**

**BEGIN**

**UPDATE nn\_employee**

**SET salary = salary \* ((pv\_percentage\_raise\_i/100)+1)**

**WHERE deptid = pv\_deptid\_i**

**AND positionid = pv\_positionid\_i;**

**pv\_num\_employees\_o := SQL%ROWCOUNT;**

**END;**

**Sample output:**

## The anonymous block **D20\_L05\_test\_give\_raise\_sp** has been created to test the procedure using the test cases on the next page. It is in the **Moodle** folder for this lab. Open it and test your procedure.

**Sample output:**

**PL/SQL procedure successfully completed.**

# Functions

***Objectives*:** Learnto create and use PL/SQL functions.

***To Do:***

## Write a function, and pass a department number to it. Call your function **D20\_L05\_is\_valid\_department\_sf**. If the DEPT table does not contain that department number, return a FALSE value; otherwise, return a TRUE value. Test with an anonymous block that print the appropriate message in the calling program based on the result. The anonymous block should test for both the TRUE and FALSE cases.

**PL/SQL:**

**CREATE OR REPLACE FUNCTION D20\_L05\_is\_valid\_department\_sf (**

**pv\_deptid in number**

**)**

**return boolean**

**AS**

**lv\_rows number;**

**BEGIN**

**SELECT count(\*) into lv\_rows**

**FROM nn\_Dept**

**WHERE DEPTID = pv\_deptid;**

**if lv\_rows > 0**

**then return true;**

**else**

**return false;**

**end if;**

**END;**

**Sample output:**

**BEGIN**

**--shitty way, I know**

**case D20\_L05\_is\_valid\_department\_sf(99)**

**WHEN true**

**THEN DBMS\_OUTPUT.PUT\_LINE('True');**

**WHEN false**

**THEN DBMS\_OUTPUT.PUT\_LINE('False');**

**END case;**

**case D20\_L05\_is\_valid\_department\_sf(10)**

**WHEN true**

**THEN DBMS\_OUTPUT.PUT\_LINE('True');**

**WHEN false**

**THEN DBMS\_OUTPUT.PUT\_LINE('False');**

**END case;**

**END;**

**PL/SQL procedure successfully completed.**

**False**

**True**

## Write a function called D20\_L05\_CSID\_capacity that takes a CSID and returns a number 0-100 indicating the percentage of capacity (i.e 75 indicates that we’re at 75% of the MAXCOUNT for a CSID). Write an anonymous block that takes a cursor and loops through all CSIDs for courses starting with 420. Output the course name, the CSID, the term and the room building and room number (if a room is assigned), and the %capacity (by calling your new function).

**PL/SQL:**

**CREATE OR REPLACE FUNCTION D20\_L05\_CSID\_capacity (**

**pv\_csid in iu\_crssection.csid%TYPE**

**)**

**return number**

**AS**

**lv\_per number;**

**BEGIN**

**select count(r.csid) / crs.maxcount \* 100 into lv\_per**

**from iu\_crssection crs, iu\_registration r**

**where crs.csid = pv\_csid**

**AND R.CSID = crs.CSID**

**group by crs.maxcount;**

**return lv\_per;**

**END;**

**declare**

**cursor lcur\_csid**

**is select csid from iu\_crssection;**

**--cursor should only select those with 420 csid**

**lv\_per number;**

**begin**

**for x in lcur\_csid loop**

**lv\_per := D20\_L05\_CSID\_capacity(x.csid);**

**DBMS\_OUTPUT.PUT\_LINE('CSID ' || x.csid || ': ' || lv\_per);**

**end loop;**

**end;**

**Sample output:**

**CSID 1101: 6.66666666666666666666666666666666666667**

**CSID 1102: 7.5**

**CSID 1103: 4**

**CSID 1104: 100**

**CSID 1105: 4**

**CSID 1205: 2.85714285714285714285714285714285714286**

**CSID 1206: 10**

**CSID 1207: 10**

**CSID 1208: 100**

**CSID 1209: 2.5**

**CSID 1210: 100**

**CSID 1211: 4**

# Assessment

1. What did you learn in completing this lab?

How to do functions

1. What did you have difficulty with?

I was speed running this, I kinda just did it and if it went wrong I kept going…

1. What did you do well?

Learnt functions

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

1 1/2

1. What took you the most time?

Learning syntax

**D20\_L05\_give\_raise\_sp Test Cases (For C2)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Procedure Call [[1]](#footnote-1)** | **Table Data** | | | | **Output Parameter** | **Updated Table Data** |
|  |  | **employeeid** | **deptid** | **positionid** | **salary** | **pv\_num\_employees\_o** | **salary** |
| 1. All employees | D20\_L05\_give\_raise\_sp( pv\_percentage\_raise\_i => 10, pv\_num\_employees\_o => lv\_num\_employees); | 111  123  246  433  543  200  135  222 | 10  10  20  20  40  30  30  10 | 1  2  2  3  2  5  2  4 | 265000  150000  75000  66500  80000  24500  45000  35000 | 8 | 291500  165000  82500  73150  88000  26950  49500  38500 |
| 2. All employees in one department | D20\_L05\_give\_raise\_sp( pv\_percentage\_raise\_i => 15, pv\_deptid\_i => 10, pv\_num\_employees\_o => lv\_num\_employees); | 123  222 | 10  10 | 2  4 | 75000  35000 | 2 | 304750  86250  40250 |
| 3. All employees in one position | D20\_L05\_give\_raise\_sp( pv\_percentage\_raise\_i => 20, pv\_positionid\_i => 2, pv\_num\_employees\_o => lv\_num\_employees); | 246  123  543  135 | 20  10  20  30 | 2  2  2  2 | 150000  75000  80000  45000 | 4 | 180000  90000  96000  54000 |
| 4. All employees in one position in one department | D20\_L05\_give\_raise\_sp( pv\_percentage\_raise\_i => 25, pv\_deptid\_i => 20,  pv\_positionid\_i => 2, pv\_num\_employees\_o => lv\_num\_employees); | 543  246 | 20  20 | 2  2 | 80000  150000 | 2 | 100000  187500 |

1. Assume that the procedure call is in an anonymous block in which a variable called lv\_num\_employees has been declared. [↑](#footnote-ref-1)