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

# *Lab 7 – Triggers*

Date assigned: Tuesday, March 21, 2017

Date due: **Tuesday, March 21, 2017, 17:50**

**Objectives:**

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

* Create and test a trigger.

**To be handed in:**

1. The ***username\_*D20\_L07\_Triggers.docx** file should be uploaded to Moodle.

**References:**

Class notes S11. S12, N12 and examples from Moodle

**To Start:**

1. Rename this document to ***username\_*D20\_L07\_Triggers.docx** and add your answers and output as requested.

**Important:**

* Name all identifiers according to the naming standards shown below.
* Format all blocks using the SQL Navigator Formatter.
* Use ISO/ANSI standards for all joins.
* Show test output

**Marking**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Mark** | **Out of** |
| A – Triggers |  |  |
| * + 1 (nn\_employee\_bui\_stmt\_trg) |  | 8 |
| * + 2 (iu\_student\_ai\_trg) |  | 8 |
| * + 3 (nn\_employee\_bi\_trg) |  | 8 |
| * + 4 |  | 4 |
| * + 5 |  | 20 |
| B – SYS\_REFCURSOR |  | 8 |
| Organization - Blocks Formatted and naming standards used, assessment |  | 5 |
| **Total** |  | **61** |

**Use the following naming conventions:**

| **Name** | **Prefix** | **Suffix** |
| --- | --- | --- |
| exception | **e\_** |  |
| function name |  | **\_sf** |
| global variable | **gv\_** |  |
| IN OUT parameter variable | **pv\_** | **\_io** |
| IN parameter variable | **pv\_** | **\_i** |
| local cursor | **lcur\_** |  |
| local record | **lrec\_** |  |
| local variable | **lv\_** |  |
| OUT parameter variable | **pv\_** | **\_o** |
| package |  | **\_pkg** |
| package function |  | **\_pf** |
| package procedure |  | **\_pp** |
| parameter record | **prec\_** | **\_i, \_o or \_io** |
| procedure name |  | **\_sp** |
| local type | **ltyp\_** |  |
| local record | **lrec\_** |  |
| local table | **ltbl\_** |  |
| trigger |  | **\_trg** |

**Trigger Naming Conventions:**

***table name\_*[b|a]*op*\_[*col/*stmt*]\_*trg**

where:

- **b** is used for a before trigger

- **a** is used for an after trigger

- *op* is the list of DML operations that fire the trigger (u- update, i-insert, d-delete)

- *col* is the column name for a trigger on a single column

- **stmt** is the for a statement level trigger

# Triggers

***Objectives*:** Learnto create and test a trigger.

To Do:

## Write a trigger that is fired before the DML statement’s execution on the IU\_EMPLOYEE table for inserts and updates. The trigger ensures that the hiredate does not fall on a Sunday (an exception is raised if this is the case). Call the trigger **nn\_employee\_bui\_stmt\_trg**.

**Trigger Code:**

**CREATE OR REPLACE TRIGGER nn\_employee\_bui\_stmt\_trg**

**BEFORE INSERT OR UPDATE ON NN\_EMPLOYEE FOR EACH ROW**

**BEGIN**

**IF to\_char(:new.hiredate, 'DY') LIKE 'SUN' THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Can''t hire someone on a Sunday!!!');**

**END IF;**

**END;**

**Test code:**

**UPDATE nn\_employee e**

**SET e.hiredate = to\_date('19-Mar-17', 'dd-Mon-yy')**

**WHERE e.employeeid = 111;**

**--Proper testing needs both a pass and a fail case**

**Sample output from Test Code:**

**Error report -**

**SQL Error: ORA-20001: Can't hire someone on a Sunday!!!**

**ORA-06512: at "PDUMARESQ.NN\_EMPLOYEE\_BUI\_STMT\_TRG", line 3**

**ORA-04088: error during execution of trigger 'PDUMARESQ.NN\_EMPLOYEE\_BUI\_STMT\_TRG'**

## Create a new table called **iu\_tracking**. It should have the following columns:

**studentid varchar2(5)**

**username varchar2(20)**

**insertdate date**

## Write a trigger that is fired after an INSERT statement is executed for the IU\_STUDENT table. The trigger writes the new student’s ID, users name, and system date in the IU\_TRACKING table. Call the trigger **iu\_student\_ai\_trg**.

**Trigger Code:**

**DROP TABLE iu\_tracking;**

**CREATE TABLE iu\_tracking (**

**studentid varchar2(5),**

**username varchar2(30),**

**insertdate date**

**);**

**CREATE OR REPLACE TRIGGER iu\_student\_ai\_trg**

**AFTER INSERT ON iu\_student FOR EACH ROW**

**BEGIN**

**INSERT INTO iu\_tracking (**

**studentid, username, insertdate**

**) VALUES (**

**:new.studentid, :new.first || ' ' || :new.last, sysdate**

**);**

**END;**

**Test code (insert into IU\_STUDENT, then select from IU\_TRACKING):**

**INSERT INTO iu\_student (**

**studentid, first, last**

**) VALUES (**

**999, 'philip', 'dumaresq'**

**);**

**Sample output from Test Code:**



## Create a nn\_employee\_seq sequence that starts at 600 and increments by one. Create a before insert trigger for the **NN\_EMPLOYEE** table called **nn\_employee\_bi\_trg** that sets the new employeeId based on the sequence and the hiredate on sysdate. This way, the insert guarantees a unique employeeId (abstracting away the sequenc e) and the hire date is set when an employee entry is created.

**Trigger Code:**

**CREATE SEQUENCE nn\_employee\_seq**

**START WITH 600**

**INCREMENT BY 1;**

**SET SERVEROUTPUT ON;**

**CREATE OR REPLACE TRIGGER nn\_employee\_bi\_trg**

**BEFORE INSERT ON nn\_employee FOR EACH ROW**

**DECLARE**

**lv\_empid nn\_employee.employeeid%TYPE;**

**BEGIN**

**SELECT nn\_employee\_seq.NEXTVAL INTO lv\_empid FROM dual;**

**:new.employeeid := lv\_empid;**

**:new.hiredate := sysdate;**

**END nn\_employee\_bi\_trg;**

**Test code:**

**Insert into NN\_EMPLOYEE (**

**EMPLOYEEID, LNAME, FNAME, POSITIONID, SUPERVISOR,**

**HIREDATE, SALARY, COMMISSION, DEPTID, QUALID**

**) values (**

**1, 'Philip', 'Dumaresq', 1, null,**

**to\_date('60-04-15','RR-MM-DD'), 265000, 35000, 20, 1**

**);**

**Sample output from Test Code:**

****

## Run the following update statement to put the correct values into the **creditsearned** column of the student:

**UPDATE iu\_student s**

**SET creditsearned =**

**(SELECT SUM (CASE**

**WHEN FINAL IN ('A', 'B', 'C', 'D')**

**THEN credits**

**ELSE 0**

**END)**

**FROM iu\_course**

**JOIN iu\_crssection USING (courseid)**

**JOIN iu\_registration r USING (csid)**

**WHERE r.studentid = s.studentid);**

## Cool update huh? Do you understand how it works? Explain what this update does and how it works:

This update will modify the credits earned for each row in the students table

## Create an after trigger for the **iu\_registration** table called **iu\_registration\_aidu\_trg** that updates the **creditsEarned** column of the **student** table if a value was entered in the **final** column of the **iu\_registration** table. If the value was changed from a pass to a fail, a withdrawal or a null value, subtract the number of credits for the course from the **creditsEarned** column. If the value was changed from a fail, a withdrawal or a null to a pass, add the number of credits for the course to the **creditsEarned** column. If a row was deleted from the **iu\_registration** table and the **final** was a pass, subtract the number of credits for the course from the **creditsEarned** column. If a row is added with a passing **final**, add the number of credits to the **creditsEarned** column.

Note: If you can avoid it, do NOT calculate credits earned from scratch, figure out how much to add/remove from the existing value (this will be more run-time efficient).

**Trigger Code:**

**Test code (provided, see d20\_iu\_registration\_aidu\_trg\_test file from moodle). Test cases implemented are:**

| **Test Case** | **Input Data** | | | **Table Data** | | **Results** |
| --- | --- | --- | --- | --- | --- | --- |
| **studentId** | **csid** | **New****final** | **Old****final** | **creditsEarned** | **creditsEarned** |
| Change a null final to a pass | 00100 | 1207 | B | Null | 6 | 9 |
| Change a failing final to a pass. | 00100 | 1101 | A | F | 6 | 9 |
| Change a withdrawal to a pass. | 00103 | 1101 | C | W | 3 | 6 |
| Change a null final to a fail | 00100 | 1205 | F | Null | 6 | 6 |
| Change a pass to a fail | 00100 | 1102 | F | B | 6 | 3 |
| Change a withdrawal to a fail | 00103 | 1101 | F | W | 3 | 3 |
| Change a pass to a different pass | 00100 | 1104 | B | A | 6 | 6 |
| Change a pass to null | 00100 | 1104 | null | A | 6 | 3 |
| Delete a row that was a fail | 00100 | 1101 | - | F | 6 | 6 |
| Delete a row that was a pass | 00100 | 1104 | - | A | 6 | 3 |
| Delete a row that was a withdrawal | 00103 | 1101 | - | W | 3 | 3 |
| Delete a row with a null final | 00100 | 1205 | - | Null | 6 | 6 |
| Add a row with a null final | 00100 | 1206 | Null | - | 6 | 6 |
| Add a row with a passing final | 00100 | 1103 | B | - | 6 | 9 |
| Add a row with a failing final | 00100 | 1103 | F | - | 6 | 6 |
| Add a row with a withdrawal in final | 00100 | 1103 | W | - | 6 | 6 |

**Sample output from Test Code:**

**Test case 1 successful**

**Test case 2 successful**

**Test case 3 successful**

**Test case 4 successful**

**Test case 5 successful**

**Test case 6 successful**

**Test case 7 successful**

**Test case 8 successful**

**Test case 9 successful**

**Test case 10 successful**

**Test case 11 successful**

**Test case 12 successful**

**Test case 13 successful**

**Test case 14 successful**

**Test case 15 successful**

**Test case 16 successful**

# SYS\_REFCURSOR

1. Create get\_student\_final\_sp() that takes as input a csid and outputs a SYS\_REFCURSOR that returns a studentId and FINAL mark. Hint: See S05 example04 as a starting point

**PL/SQL Code:**

**CREATE OR REPLACE PROCEDURE get\_student\_final\_sp(**

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

**pv\_cursor\_o OUT SYS\_REFCURSOR )**

**AS**

**BEGIN**

**OPEN pv\_cursor\_o FOR**

**SELECT studentid, final**

**FROM iu\_registration**

**WHERE csid = pv\_csid\_i;**

**END;**

1. Create a calling block to loop through and output the studentIds and Final marks

**Test code:**

**DECLARE**

**lv\_csid iu\_crssection.csid%TYPE;**

**lv\_cursor SYS\_REFCURSOR;**

**lv\_studentid iu\_registration.studentid%TYPE;**

**lv\_final iu\_registration.final%TYPE;**

**BEGIN**

**lv\_csid := &CSID;**

**get\_student\_final\_sp(lv\_csid, lv\_cursor);**

**LOOP**

**FETCH lv\_cursor INTO lv\_studentid, lv\_final;**

**EXIT WHEN lv\_cursor%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE('Student ID: '||lv\_studentid||' Final: '||lv\_final);**

**END LOOP;**

**END;**

**Sample output from Test Code:**

**Student ID: 00100 Final: F**

**Student ID: 00103 Final: W**

# Assessment

1. What did you learn in completing this lab?

SYS\_REFCURSORS

1. What did you have difficulty with?

Took a while to figure out weird Oracle issues

1. What did you do well?

Got through test cases pretty well once I got that nvl thing working. Rest of the lab went slow cause I was really tired.

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

4

1. What took you the most time?

Figuring out weird Oracle issues