FIT3171 Databases

Week 11 Tutorial Activities

PLSQL (Triggers & Procedures)

FIT Database Teaching Team

Complete the week 11 tutorial activities listed below:

FIT3171 2022 S1

FIT3171 Databases

Author: FIT Database Teaching Team

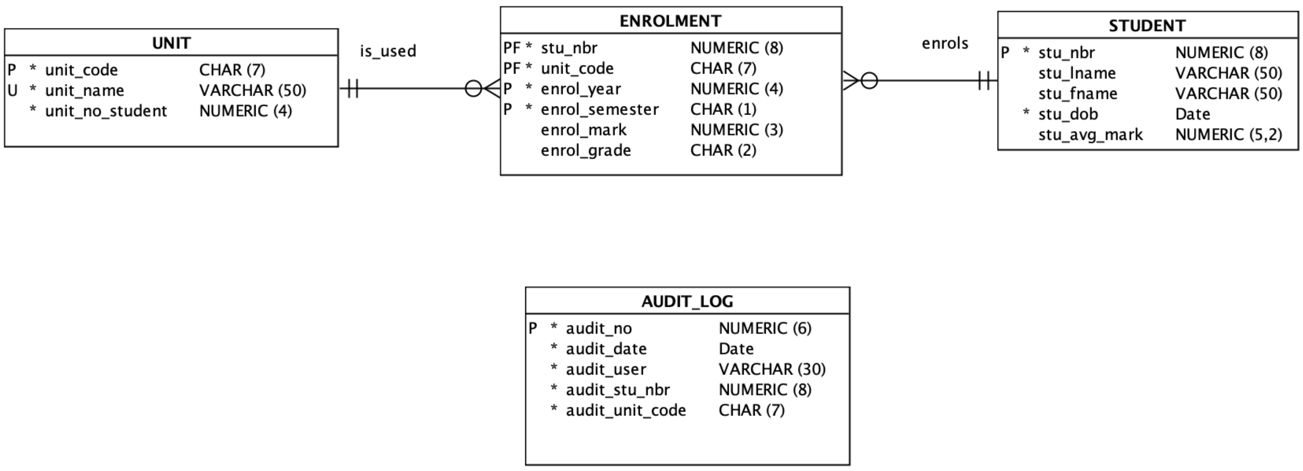
Learning Objectives:

At the completion of these tutorial activities, you should be able to:

- code and execute basic ORACLE procedures, including parameters passed in and out
- code and compile ORACLE triggers
- test the correctness of the logic in a procedure or a trigger

11.1 Class Discussion

Using the student_enrolment_schema_insert.sql supplied on Moodle, create and load the tables.



A 'BEFORE' trigger is used for verifying the value or if wish to change values before the system executes the triggering statement. For example, the trigger that checks whether a user inserts at least one name (either first name or last name) is a before trigger. An 'AFTER' trigger is used when the block of code in the trigger body is executed after the triggering statement.

Create a trigger to enforce that a student must have at least one name (either first name or last name) when a new student is added to the STUDENT table. If the insert would result in neither a stu_lname or a stu_fname being added, the trigger must raise an error and prevent the insert. Write a test harness for this trigger.

11.2 Practical Exercises

Create a new sql file called **week11_plsql.sql**, place this file in your working directory in the App11 folder. Be sure you have run the student_enrolment_schema_insert.sql and set up the tables in your account before proceeding. Write your answers for questions 1-6. Make sure to include a line break (enter) then a slash ('/') then another line break (enter) to separate the PL/SQL block and the SQL block.

Create and test your trigger/procedure one by one (ie. write a test harness for each trigger/procedure).

Part A. Trigger Exercises

1. Create a trigger called **unit_upd_cascade** that updates matching rows in the ENROLMENT table whenever a unit code is updated in the UNIT table. The changes in the enrolment table should be announced via dbms_output.put_line

Check whether the trigger works using SQL. Hint - update a unit code in the unit table and observe what happens in the enrolment table.

2. Create a trigger called **change_enrolment** to:

- Maintain the integrity of the total number of students who have attempted a given unit in the UNIT table, and
- Record a delete operation in the audit_log when an enrolment is deleted.

3. Create a trigger called **maintain_student_average** to maintain the student's average mark whenever a mark is updated.

4. Create a trigger called **calculate_grade** that calculates the student's grade (enrolment.enrol_grade) whenever a mark is updated for an enrolment or a new enrolment is added with a mark. Hint: you can avoid the "mutating table" error here by directly manipulating the new value of enrol_grade in your trigger based on the new enrol_mark value.

- enrol_mark >= 80 is a HD grade
- enrol_mark >= 70 and enrol_mark < 80 is a D grade
- enrol_mark >= 60 and enrol_mark < 70 is a C grade
- enrol_mark >= 50 and enrol_mark < 60 is a P grade
- enrol_mark < 50 is a N grade

Check whether the trigger works using SQL.

Part B. Procedure Exercises

5. Create a stored procedure named **prc_new_enrolment** to insert a new row into the ENROLMENT table. The procedure should satisfy the following requirements:

a. It has four input parameters: in_stu_nbr and in_unit_code, in_enrol_year, in_enrol_sem, and one output parameter: out_message.
b. Verify that the student number and unit code must exist in the STUDENT and UNIT tables, respectively, and ensure that this enrolment does not currently exist.
c. If the supplied student number and unit code exist, proceed with the insert statement and return an OUT successful message, otherwise, return an OUT message that indicates the relevant insertion issue (e.g., incorrect student number and/or incorrect unit code).

6. Create a table called UPDATE_LOG. This table requires 9 attributes: update_id (PK), update_date, update_user, stu_nbr, unit_code, enrol_year, enrol_semester, update_prev_mark, update_new_mark. Then, create a stored procedure named **prc_update_enrolment** to update an enrolment mark for a student in the ENROLMENT table. The procedure should satisfy the following requirements:

a. It has five parameters: in_stu_nbr, in_unit_code, in_enrol_year, in_enrol_sem, in_enrol_mark, and one output parameter: out_message.
b. Verify that the enrolment must exist in the ENROLMENT table (i.e. it is a valid enrolment).
c. If the supplied enrolment details exist, proceed with the update statement, log the update into the UPDATE_LOG table and return an OUT successful message, otherwise, return an OUT message that indicates the relevant update issue (e.g. invalid enrolment).

Important

You need to get into the habit of establishing this as a standard FIT3171 workflow - pull at the start of your working session, work on the activities you wish to/are able to complete during this session, save the files, add all (stage), commit and then push the changes back to the FIT GitLab server.