DATA 504 Project 2 – Using PL/SQL
to Implement Student Registration System
Due: To be announced

This project is to use Oracle's PL/SQL to create an application to support typical student registration tasks in a university. Students are required to form teams of two members for this project.

Due to the time constraint, only a subset of the database tables and a subset of the needed functionalities will be implemented in this project.

1. Preparation

The following tables from the Student Registration System will be used in this project:

Students(B#, first_name, last_name, status, gpa, email, bdate, deptname)
Courses(prog_code, course#, title)
Classes(classid, prog_code, course#, sect#, year, semester, limit, class_size, room)
Enrollments(B#, classid, lgrade)

In addition, the following table is also required for this project:

Logs(log#, op_name, op_time, table_name, operation, key_value)

Each tuple in the logs table describes who (op_name - the login name of a database user) has performed what operation (insert, delete, update) on which table (table_name) and which tuple (as indicated by the value of the primary key, key_value, of the tuple) at what time (op_time). Attribute log# is the primary key of the table. op_name can be produced by system function "user" and op_time can be produced by system function "sysdate".

The schemas and constraints of the first four tables are the same as those used in Project 1. Please use the following statements to create the Logs table (you can add them to the script file):

```
create table logs (log# number(4) primary key, op_name varchar2(10) not null, op_time
date not null, table_name varchar2(12) not null, operation varchar2(6) not null, key_value
varchar2(10));
```

2. PL/SQL Implementation

You need to create a PL/SQL package for this application. All procedures and functions should be included in this package. **Other Oracle objects such as sequences and triggers are to be created outside the package.** The following requirements and functionalities need to be implemented.

1. (4 points) Use a sequence to generate the values for log# automatically when new log records are inserted into the logs table. Start the sequence with 100 with an increment of 1.

2. (8 points) Write procedures in your package to display the tuples in each of the five tables for this project. As an example, you can write a procedure, say **show_students**, to display all students in the students table.
3. (30 points) Write a procedure in your package to enroll a student into a class (i.e., insert a tuple into the Enrollments table). The B# of the student and the classid of the class are provided as parameters (all new enrollments will have a null value for lgrade). In all the following cases, reject the enrollment request:
    a. If the student is not in the Students table, report "The B# is invalid."
    b. If the classid is not in the classes table, report "The classid is invalid."
    c. If the class is not offered in Spring 2020 (treating it as the current semester), report "Cannot enroll into a class from a previous semester."
    d. If the class is already full before the enrollment request, report "The class is already full."
    e. If the student is already in the class, report "The student is already in the class."
    f. If the student is already enrolled in four other classes in the same semester and the same year, report "Students cannot enroll into more than four courses in the same semester."

    For all the other cases, the requested enrollment should be carried out successfully. You need to make sure that all data are consistent after each enrollment. For example, after you successfully enrolled a student into a class, the class size of the class should be increased by 1. Use trigger(s) to implement the updates of values caused by successfully enrolling a student into a class. (Once again, it is recommended that all triggers for this project be implemented outside of the package.)
4. (20 points) Write a procedure in your package to drop a student from a class (i.e., delete a tuple from the Enrollments table). The B# of the student and the classid of the class are provided as parameters. In all the following cases, reject the drop request:
    a. If the student is not in the Students table, report "The B# is invalid."
    b. If the classid is not in the Classes table, report "The classid is invalid."
    c. If the student is not enrolled in the class, report "The student is not enrolled in the class."
    d. If the class is not offered in Spring 2020, report "Only enrollment in the current semester can be dropped."

    In all the other cases, the student will be dropped from the class. If the class is the last class for the student, report "This student is not enrolled in any classes." If the student is the last student in the class, report "The class now has no students." Again, you should make sure that all data are consistent after a successful enrollment drop and all updates caused by the drop need to be implemented using trigger(s).
5. (15 points) Write triggers to add tuples to the Logs table automatically whenever a student is successfully enrolled into or dropped from a class (i.e., when a tuple is inserted into or

deleted from the Enrollments table). For a logs record for enrollments, the key value is the concatenation of the B# value, a comma, and the classid value.

3. Documentation (15 points)

Documentation consists of the following aspects:

1. Each procedure and function and every other object you create for your project needs to be explained clearly regarding its objective and usage.

2. Your code needs to be reasonably documented with in-line comments.

3. *Team report*. This report should describe in reasonable detail how the team members collaborated for the project. More specifically, it needs to include the following information:
    a. Your meetings (describe when each meeting occurred and what was discussed for the project at the meeting?)
    b. What was your plan (schedule) for completing your project? How was the plan followed during the course of completing the project?
    c. Which team member is primarily responsible for which part of the project? Did the other member contribute to the task primarily assigned to another member?

4. What to submit

Each team needs to submit a report containing the following components (only one copy for each team is needed and one member can submit representing the team):

1. Names of team members.

2. The honesty statement: "We have done this assignment completely on our own. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism or cheating we will have to sign an official form that we have cheated and that this form will be stored in our official university records. We also understand that we will receive a grade of 0 for the involved assignment and our grades will be reduced by one level (e.g., from A to A- or from B+ to B) for our first offense, and that we will receive a grade of "F" for the course for any additional offense of any kind."

3. The *Team Report* (see the Documentation section).

4. Your entire PL/SQL code (including the package, triggers, and sequences).