INFO-532 Database Systems
Project 2 – Using PL/SQL and JDBC
to Implement Student Registration System
(Due May 02, 2024, in Brightspace)

Please remember to include the following statement with your submitted Project 2 report and SIGN it by all team members. Your project won't be graded without the signed statement.
"We have done this assignment completely on our own except for the tools/software acknowledged in the project report. 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."

This project is to use Oracle's PL/SQL and JDBC to create an application to support typical student registration tasks in a university. This is a team project. Each team can have up to **four students**. Teams are formed by students themselves and they're responsible for the teams they create/belong to. Teams/groups are made in Brightspace. **Each team must be formed April 05, 2023**.

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, st_level, gpa, email, bdate)
Courses(dept_code, course#, title)
Course_credit(course#, credits)
Classes(classid, dept_code, course#, sect#, year, semester, limit, class_size, room)
G_Enrollments(G_B#, classid, score)
Score_Grade(score, lgrade)
Prerequisites(dept_code, course#, pre_dept_code, pre_course#)

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

Logs(log#, user_name, op_time, table_name, operation, tuple_keyvalue)

Each tuple in the logs table describes who (user_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 of the tuple tuple_keyvalue) at what time (op_time). Attribute log# is the primary key of the table.

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

```
create table prerequisites (dept_code varchar2(4) not null,
course# number(3) not null, pre_dept_code varchar2(4) not null,
pre_course# number(3) not null,
primary key (dept_code, course#, pre_dept_code, pre_course#),
```

```
foreign key (dept_code, course#) references courses on delete cascade,
foreign key (pre_dept_code, pre_course#) references courses on delete cascade);

create table logs (log# number(4) primary key,
user_name varchar2(10) not null,
op_time date not null,
table_name varchar2(13) not null,
operation varchar2(6) not null,
tuple_keyvalue varchar2(20));
```

You should populate these tables with appropriate tuples to test your programs.

2. PL/SQL Implementation (50 points)

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. (2 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 1000 with an increment of 1.
2. (4 points) Write procedures in your package to display the tuples in each of the eight tables for this project. As an example, you can write a procedure, say **show_students**, to display all students in the students table.
3. (3 points) Write a procedure in your package that, for a given class (with classid provided as an in parameter), will list the B#, the first name and last name of every student in the class. If the provided classid is invalid (i.e., not in the Classes table), report "The classid is invalid."
4. (4 points) Write a procedure in your package that, for a given course (with dept_code and course# as parameters), can return all its prerequisite courses (show dept_code and course# together as in CS532), including both direct and indirect prerequisite courses. If course C1 has course C2 as a prerequisite, C2 is a direct prerequisite of C1. If C2 has course C3 as a direct prerequisite and C3 is not a direct prerequisite of CS, then C3 is an indirect prerequisite for C1. Please note that indirect prerequisites can be more than two levels away. If the provided (dept_code, course#) is invalid, report "dept_code || course# does not exist." – show dept_code and course# together as in CS532.
5. (14 points) Write a procedure in your package to enroll a graduate student into a class (i.e., insert a tuple into the G_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 score). If the B# is not in the Students table, report "The B# is invalid." If the B# does not correspond to a graduate student, report "This is not a graduate student." If the classid is not in the classes table, report "The classid is invalid." If the class is not offered in the current semester (suppose Spring 2021 is the current semester), reject the enrollment and report "Cannot enroll into a class from a previous semester." If the class is already full before the enrollment request, reject the enrollment request and report "The class is already full." If the student is already in the class, report "The student is already in the class." If the student is already enrolled in five other classes in the same semester and the same year, report "Students cannot be enrolled in more than five classes in the same semester." and reject the enrollment. If the student has not completed the required prerequisite courses with at least a grade C, reject the enrollment and report "Prerequisite not satisfied." 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. (It is recommended that all triggers for this project be implemented outside of the package.)

6. (10 points) Write a procedure in your package to drop a graduate student from a class (i.e., delete a tuple from the G_Enrollments table). The B# of the student and the classid of the class are provided as parameters. If the student is not in the Students table, report "The B# is invalid." If the B# does not correspond to a graduate student, report "This is not a graduate student." If the classid is not in the Classes table, report "The classid is invalid." If the student is not enrolled in the class, report "The student is not enrolled in the class." If the class is not offered in Spring 2021, reject the drop attempt and report "Only enrollment in the current semester can be dropped." . If the class is the last class for the student in Spring 2021, reject the drop request and report "This is the only class for this student in Spring 2021 and cannot be dropped." In all the other cases, the student will be dropped from the class. 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).
7. (5 points) Write a procedure in your package to delete a student from the Students table based on a given B# (as a parameter). If the student is not in the Students table, report "The B# is invalid." When a student is deleted, all tuples in the G_Enrollments table involving the student should also be deleted (use a trigger to implement this). Note that such a deletion may trigger a number of actions as described in the above item (item 6).
8. (8 points) Write triggers to add tuples to the Logs table automatically whenever a student is deleted from the Students table, or when a student is successfully enrolled into or dropped from a class (i.e., when a tuple is inserted into or deleted from the G_Enrollments table). For a logs record for G_Enrollments, the key value is the concatenation of the B# value, a comma, and the classid value.

3. Interface (35 points)

Implement an interactive and menu-driven interface in the harveyv environment using Java and JDBC (see sample demo programs attached with the Project). More details are given below:

1. The basic requirement for the interface is a text-based menu-driven interface. You first display menu options for a user to select (e.g., 1 for displaying a table, 2 for enrolling a graduate student into a class, …). An option may have sub-options depending on your need. Once a final option is selected, the interface may prompt the user to enter parameter values from the terminal. As an example, for enrolling a graduate student into a class, the parameter values include B# and classid. Then an operation corresponding to the selected option will be performed with appropriate message displayed.

2. Your interface program should utilize as many of your PL/SQL code as possible. Some of the procedures may need to be rewritten in order for them to be used in your Java/JDBC program. In particular, in order to pass retrieved results from a PL/SQL block (e.g., show_students) to the Java/JDBC program, the block needs to be implemented as a ref cursor function (see the third sample program for an example).

3. Note that messages generated by the dbms_output package in your PL/SQL package or triggers will not be displayed on your monitor screen when you run your Java/JDBC application. You can either regenerate these messages in your Java program or use dbms_output.get_line( ) to catch the messages generated by dbms_output.put_line( ).

4. **You could also receive a bonus of up to 50 points, if you develop a web interface that provides the following features:**

a. To view/add/delete students from the database.
b. To view/add/delete courses from the database.
c. To view/add/delete classes from the database.
d. To enroll/drop courses(enrollment) of a student from the database.
e. To view logs and enrollment status.

Please note: You must use existing procedures to support above functionalities, however if any procedure doesn't support it then your code can execute queries depending on the parameters of your functionality respecting all constraints.

4. 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 well 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. Procedures/code components implemented – must provide an overview of the functionalities implemented with screenshots of execution.
    b. Your meetings – describe when each meeting occurred and what was discussed for the project at the meeting.
    c. Your plans – describe your team's plan (schedule) for completing your project. Also report how the plan was followed during the course of completing the project.
    d. Your responsibilities – describe how the tasks are divided among the team members. Specifically, describe which member is primarily responsible for which part of the project and how the other members contribute to the task primarily assigned to a particular member.
    e. Your self-assessment of the team work – which of the following phrases best describe how well your team has worked together: (1) worked really well together; (2) generally worked well with some minor issues; (3) struggled as a team but made it work eventually; (4) struggled throughout mostly. Your self-assessment will NOT affect the grade of your project.

4. If you do not agree with your team report, you may submit a separate personal report to detail your disagreement with the team report. Personal reports are emailed to the instructor separately from other project documents and will be kept confidential by the instructor.

5. Hand-ins, Demo and Grading

1. Each team needs to submit a report containing the following components (only one copy for each team is needed and one member can submit it on behalf of the team) to the Project 2 submission folder:

- Names of team members.

- 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.”

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

- Readme file must contain steps on how to run your project including any necessary dependencies or config files.

- The entire PL/SQL code (including the package, triggers, and sequences).

- The entire Java codes.

- You can also add a video to support your demonstration.

  Please note: It’s your responsibility to ensure that your submission especially your code must be in a complete (i.e. workable state), so be careful with your submission. Re-submissions maybe subject to penalty. If at all your project is incomplete/non-workable state, please discuss this with course instructor/TA before the submission due date.

2. Each team is required to demonstrate the completed project to the instructor using tuples created by the instructor. More instructions on demo will be provided before the demo.

3. The grading will be based on the quality of your code, the documentation and on how successful of your demo is. Demo is mandatory for all projects.