CS532 Project 2
Using PL/SQL and JDBC to Implement Student Registration System

Turn in your source code by 11:59AM April 13 through blackboard. Also, turn in printed hard copy documents during your demo on April 13.

This project is to use Oracle's PL/SQL and JDBC to create an application to support typical student registration tasks at a university. 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

In this project, the following tables from the Student Registration System will be used:

Students(sid, firstname, lastname, status, gpa, email)
Courses(dept_code, course_no, title)
Prerequisites(dept_code, course_no, pre_dept_code, pre_course_no)
Classes(classid, dept_code, course_no, sect_no, year, semester, limit, class_size)
Enrollments(sid, classid, lgrade)

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

Logs(logid, who, time, table_name, operation, key_value)

Each tuple in the logs table describes who (the login name of a database user) has performed what operation (insert, delete, update) on which table (give the table name) and which tuple (as indicated by the value of the primary key of the tuple) at what time. Attribute logid is the primary key of the table.

Carefully read proj2data2018.sql provided under the Project 2 folder on Blackboard to understand the schema, and execute it in your Oracle account to create the tables and insert the data to the tables (populate the database). To run it in your Oracle account, do:

SQL> start proj2data2018

2. PL/SQL Implementation (50 points)

You need to create a PL/SQL package as well as other Oracle objects (sequences, triggers, etc.) outside the package to support the application. The following requirements and functionalities need to be implemented.

1. (2 points) Use a sequence to generate the values for logid automatically when new log records are inserted into the logs table. All logid values should have 3 digits.
2. (3 points) Write procedures in your package to display the tuples in each of the six tables: students, courses, prerequisites, classes, enrollments, logs. As an example, you can write a procedure, say **show_students**, to display all students in the students table.
3. (2 points) Write a procedure in your package to add students into the students table. The information of the student to be inserted is provided as parameters to your procedure.
4. (3 points) Write a procedure in your package that, for a given student (with sid provided as a parameter), can list the sid, lastname, and status of the student as well as all classes the student has

taken or is taking. For each class, show classid, dept_code, course_no, title, year, and semester. (dept_code and course_no should be displayed together, e.g., CS532.) If the student is not in the students table, report "The sid is invalid." If the student has not taken any course, report "The student has not taken any course."

5. (3 points) Write a procedure in your package that, for a given course (with dept_code and course_no as parameters), returns all its prerequisite courses (show dept_code and course_no 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. In addition, if C2 has course C3 has a prerequisite, then C3 is an indirect prerequisite for C1. Please also note that indirect prerequisites can be more than two levels away.
6. (3 points) Write a procedure in your package that, for a given class (with classid provided as a parameter), lists the classid, course title, semester and year of the class as well as all the students (show sid and lastname) who have taken or are taking the class. If the class is not in the classes table, report "The cid is invalid." If no student has taken or is taking the class, report "No student is enrolled in the class."
7. (12 points) Write a procedure in your package to enroll a student into a class. The sid of the student and the classid of the class are provided as parameters. If the student is not in the students table, report "The sid is invalid." If the class is not in the classes table, report "The classid is invalid." If the enrollment of the student into a class would cause "class_size > limit", reject the enrollment and report "The class is closed." If the student is already in the class, report "The student is already in the class." If the student is already enrolled in two other classes in the same semester and the same year, report "You are overloaded." and allow the student to be enrolled. If the student is already enrolled in three other classes in the same semester and the same year, report "Students cannot be enrolled in more than three classes in the same semester." and reject the enrollment. If the student has not completed the required prerequisite courses with minimum grade "C", reject the enrollment and report "Prerequisite courses have not been completed." For all the other cases, the requested enrollment should be performed. You should make sure that all data are consistent after each enrollment. For example, after you successfully enrolled a student into a class, the size of the corresponding class should be updated accordingly. 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.
8. (9 points) Write a procedure in your package to drop a student from a class (i.e., delete a tuple from the enrollments table). The sid of the student and the classid of the class are provided as parameters. If the student is not in the students table, report "The sid is invalid." 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 dropping the student from the class would cause a violation of the prerequisite requirement for another class, then reject the drop attempt and report "The drop is not permitted because another class uses it as a prerequisite." In all the other cases, the student should 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 the drop and all updates caused by the drop should be implemented using trigger(s).
9. (5 points) Write a procedure in your package to delete a student from the students table based on a given sid (as a parameter). If the student is not in the students table, report "The sid is invalid." When a student is deleted, all tuples in the enrollments table involving the student should also be deleted (use a trigger to implement this) and this will trigger a number of actions as described in #7.
10. (8 points) Write triggers to add tuples to the logs table automatically whenever a student is added to or 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 enrollments table).

3. Interface (35 points)

Implement an interactive and menu-driven interface in the bingsuns environment using Java and JDBC (see sample programs in the project 2 folder on blackboard). 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 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. As an example, for enrolling a student into a class, the parameter values include sid and classid. Then an operation corresponding to the selected option will be performed with appropriate message displayed. **A nice GUI or Web interface will receive 5 bonus points.**

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, 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 that are printed 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 may need to regenerate these messages in your Java program.

4. Documentation (15 points)

Documentation consists of the following aspects:

1. Design document. Each procedure and function and every other object you create for your project needs to be explained clearly regarding its objective and usage. Relationships among different objects (including procedures and functions) need to be made clear (a diagram can be used). You can also discuss your methodology (your approach, design, etc).

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

5. Hand-ins and Demo and Grading

1. You need to hand in a hardcopy of your entire code together with your documentation for the project. Only one copy for each team is needed.

2. All teams are required to demonstrate your project to the instructor/TA using tuples created by the instructor.

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