INFO-532 Database Systems

Project 2 – Report

Using PL/SQL and JDBC to Implement Student Registration System

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.

Group Members:
Atharv Mhatre
Preet Chaudhari
Rishank Karkera
Tarun Parmar
Axshish Karade

Table of Contents

1. **Introduction**

The Student Enrollment System is a robust and comprehensive project designed to manage various aspects of graduate student enrollment in a university. It consists of numerous procedures, functions, and triggers that cater to the different requirements of student enrollment, class management, and maintaining prerequisite relationships. This in-depth report provides a detailed explanation of the various components, their objectives, and their usage within the system.

2. **Procedures**

2.1 ENROLL_GRAD
Objective: To enroll a graduate student into a class.
Usage:

2.1.1 Check if the student exists
The procedure first checks if the student with the given B# exists in the STUDENTS table. If the student does not exist, an error is raised, indicating that the student is invalid.

2.1.2 Check if the student is a graduate student
The procedure then checks if the student is a graduate student by examining the ST_LEVEL column in the STUDENTS table. If the student is not a graduate student, an error is raised, informing that the student is not a graduate student.

2.1.3 Check if the class exists
The procedure checks if the class with the given CLASSID exists in the CLASSES table. If the class does not exist, an error is raised, indicating that the class is invalid.

2.4 Check if the class is from the current semester
The procedure checks if the class is from the current semester by comparing the class's YEAR and SEMESTER with the CUR_SEM table. If the class is not from the current semester, an error is raised, stating that the student cannot enroll in a class from a previous semester.

2.5 Check if the class is full
The procedure checks if the class is full by comparing the CLASS_SIZE with the LIMIT of the class. If the class is full, an error is raised, informing that the class is already full.

2.6 Check if the student is already in the class
The procedure checks if the student is already enrolled in the class by examining the G_ENROLLMENTS table. If the student is already enrolled in the class, an error is raised, stating that the student is already in the class.

2.7 Check if the student is enrolled in more than five classes in the same semester

The procedure checks if the student is enrolled in more than five classes in the same semester by counting the number of classes the student is enrolled in for the current semester. If the student is enrolled in five classes, an error is raised, informing that the student cannot be enrolled in more than five classes in the same semester.

2.8 Check if the prerequisites are satisfied
The procedure checks if the student has satisfied the prerequisites for the class by examining the PREREQUISITES and SCORE_GRADE tables. If the student has not satisfied the prerequisites, an error is raised, stating that the student cannot be enrolled in the class because the prerequisites are not satisfied.

2.9 Enroll the student in the class
If all the conditions above are met, the student is enrolled in the class by inserting a new record in the G_ENROLLMENTS table.

2.2 DROP_GRAD
Objective: To drop a graduate student from a class.
Usage:

2.2.1 Check if the student exists
The procedure first checks if the student with the given B# exists in the STUDENTS table. If the student does not exist, an error is raised, indicating that the student is invalid.

2.2.2 Check if the student is a graduate student
The procedure then checks if the student is a graduate student by examining the ST_LEVEL column in the STUDENTS table. If the student is not a graduate student, an error is raised, informing that the student is not a graduate student.

2.2.3 Check if the class exists
The procedure checks if the class with the given CLASSID exists in the CLASSES table. If the class does not exist, an error is raised, indicating that the class is invalid.

2.2.4 Check if the student is enrolled in the class
The procedure checks if the student is enrolled in the class by examining the G_ENROLLMENTS table. If the student is not enrolled in the class, an error is raised, stating that the student is not in the class.

2.2.5 Check if the enrollment is in the current semester
The procedure checks if the enrollment is in the current semester by comparing the class's YEAR and SEMESTER with the CUR_SEM table. If the enrollment is not in the current semester, an error is raised, informing that only enrollment in the current semester can be dropped.

2.2.6 Check if the student is enrolled in only one class in the current semester
The procedure checks if the student is enrolled in only one class in the current semester by counting the number of classes the student is enrolled in for the current semester. If the student is enrolled in only one class, an error is raised, stating that the student cannot be dropped from the class as it is their only class in the current semester.

2.2.7 Drop the student from the class
If all the conditions above are met, the student is dropped from the class by deleting the corresponding record in the G_ENROLLMENTS table.

2.3 DEL_STUDENT
Objective: To delete a student from the STUDENTS table.
Usage: Removes a student from the STUDENTS table, ensuring that the student is not enrolled in any classes. Raises an error message if the student is enrolled in one or more classes.

2.4 GET_PREQ
Objective: To retrieve all direct and indirect prerequisite courses for a given course.
Usage: Accepts a course ID and department code as input parameters and returns a list of all prerequisite courses (direct and indirect) for the specified course. Raises an error message if the provided course ID and department code are invalid.

2.5 LIST_CLASS
Objective: To list the B#, first name, and last name of every student in a given class.
Usage: Accepts a class ID as an input parameter and returns a list of students in the class, along with their B#, first name, and last name. Raises an error message if the provided class ID is invalid.

2.6 SHOW_CLASSES, SHOW_COURSES, SHOW_COURSE_CREDIT, SHOW_G_ENROLLMENTS, SHOW_LOGS, SHOW_PREREQUISITES, SHOW_SCORE_GRADE, and SHOW_STUDENTS
Objective: To display tuples in each of the respective tables in the project.
Usage: These procedures retrieve and display records from the respective tables in the system, providing an easy way to view the data.

3. Triggers

3.1 student_delete_trigger
Objective: To prevent deletion from the STUDENTS table without proper logging.
Usage:

3.1.1 Declaration of variables The trigger declares two variables:
- CURRENT_COUNT: To store the count of classes for the student in the current semester.
- LAST_COUNT: To store the total count of classes the student is enrolled in.

3.1.2 Loop through the student's enrollments The trigger iterates through each enrollment record for the student in the G_ENROLLMENTS table using a FOR loop.

3.1.3 Check if the class is in the current semester The trigger performs an INNER JOIN operation between the CLASSES and CUR_SEM tables to check if the class is in the current semester. It then stores the count in the CURRENT_COUNT variable. If the count is zero, the trigger raises an error with the message "Only enrollment in the current semester can be dropped."

3.1.4 Check if the student has only one class in the current semester The trigger performs another INNER JOIN operation between the G_ENROLLMENTS, CLASSES, and CUR_SEM tables to get the total count of classes the student is enrolled in during the current semester. It then stores the count in the LAST_COUNT variable. If the count is one, the trigger raises an error with the message "Student with B#:<B#> cannot be dropped from class:<ClassID>. This is the only class for this student in the current semester."

3.1.5 End of the loop The trigger ends the loop after checking all the student's enrollments in the G_ENROLLMENTS table.

3.2 student_log_trigger
Objective: To log the deletion of a student from the STUDENTS table.
Usage:

3.2.1 Activated after a DELETE operation on the STUDENTS table.

3.2.2 Inserts a record into the LOGS table, containing the log sequence number, the user who performed the operation, the current timestamp, the table name (STUDENTS), the operation type (DELETE), and the old B# value.

3.3 enrollment_log_trigger
Objective: To log the insertion or deletion of a tuple in the G_ENROLLMENTS table.
Usage:

3.3.1 Activated after an INSERT or DELETE operation on the G_ENROLLMENTS table.

3.3.2 If the operation is an INSERT, the trigger inserts a record into the LOGS table with the log sequence number, user who performed the operation, current timestamp, table name (G_ENROLLMENTS), operation type (INSERT), and the concatenated new G_B# and CLASSID values.

3.3.3 If the operation is a DELETE, the trigger inserts a record into the LOGS table with the log sequence number, user who performed the operation, current timestamp, table name (G_ENROLLMENTS), operation type (DELETE), and the concatenated old G_B# and CLASSID values

3.4 enrollment_update_trigger
Objective: To update the class size in the CLASSES table after a tuple is inserted or deleted from the G_ENROLLMENTS table.
Usage:

3.4.1 Activated after an INSERT or DELETE operation on the G_ENROLLMENTS table.
3.4.2 If the operation is an INSERT, the trigger updates the CLASS_SIZE in the CLASSES table by incrementing it by 1 for the corresponding CLASSID.
3.4.3 If the operation is a DELETE, the trigger updates the CLASS_SIZE in the CLASSES table by decrementing it by 1 for the corresponding CLASSID.

4. Conclusion
The Student Enrollment System is designed to efficiently manage the enrollment process and prerequisites for graduate students in a university setting. This in-depth report presents a thorough analysis of the various components, their objectives, and their usage, allowing users to gain a comprehensive understanding of the system's

Meetings and Plans:

	Preet	Atharv	Ashish	Rishank	Tarun
Meeting 1 04/07/2023	Initiated the Stored Procedure and Triggers for the project			Created outline for the Spring Boot for the JDBC	Developed a mockup user interface for a student registration system
Meeting 2 04/21/2023	Conducted testing and debugging to identify and address any issues or bugs related to the stored procedures and triggers.			Created the database schema and table structures, and how they will be mapped to Java objects using JDBC	Created the overall architecture and design of the application, including how the UI will interact with the back-end services or data storage
Meeting 3 04/28/2023	Completed and ensured that it meets all of the project requirements related to the stored procedures and triggers			Completed Spring Boot application and ensure that it meets all of the project requirements.	Completed UI and ensure that it meets all of the project requirements.

Responsibilities:

Preet, Atharv, and **Ashish** worked together on Task, which involved Stored Procedures and Triggers(PL/SQL Implementation). We collaborated to conduct testing and debugging to identify and fix any issues related to these procedures and triggers. Together, we successfully completed all project requirements related to this task and ensured that everything was functioning properly.

Rishank was primarily responsible for creating the outline for the Spring Boot for the JDBC, as well as creating the database schema and table structures, and how they will be mapped to Java objects using JDBC. He also completed the Spring Boot application and ensured that it met all of the project requirements.

Tarun was primarily responsible for developing the mockup user interface for a student registration system, creating the overall architecture and design of the application, and ensuring that the UI meets all of the project requirements