Task (3)
"University Exam Management System"

Description:

You are tasked with enhancing an exam management system for a university by implementing features to automate scheduling, manage student eligibility, record exam results, and generate analytical reports. The system will use PL/SQL procedures, functions, triggers, cursors, and transaction handling.

Tables & Relationships

Courses: (id, name, professor_id, credit_hours, prerequisite_course_id) Tracks course details, assigned professors, and the prerequisite course that must be completed before registering for this course.
Professors: (id, name, department) Stores information about university professors.
Students: (id, name, academic_status, total_credits) contains details of students and their academic status (e.g., active, suspended).
Register: (id, student_id, course_id) Tracks students who are registered for courses.
Exams: (id, course_id, exam_date, exam_type) Logs exam schedules for courses (e.g., midterm, final).
ExamResults: (id, registration_id, grade, status) Stores grade and pass/fail status for students after exams**.**
AuditTrail: (id, table_name, operation, old_data, new_data, timestamp) Logs updates or deletions to keep track of changes in any table.
Warnings: (id, student_id, warning_reason, warning_date) Stores warnings issued to students based on low performance or violations.

Features to be covered

1. User Management and Privileges

- Create a Manager User and grant them the role to create two users. Let User 1 create the Students and Courses tables. Let User 2 insert 5 rows of student data and their registered courses.
- Write a PL/SQL Procedure to automatically log the creation of new database users into the DBUserCreationLog table, capturing the username, the user who created them, and the timestamp.

2. Exam Eligibility Validation

- Ensures that a student meets the prerequisite requirements before registering for a course. Before inserting a record into the Register table, the trigger checks if the course has a prerequisite defined in the Courses table and verifies whether the student has completed it. If the prerequisite is not met, the trigger raises an error to block the registration.

3. Grade Calculation Function

- Create a PL/SQL function that calculates the grade for a student based on their exam performance. The function should take the ExamResults ID as input and compute the grade based on predefined ranges (e.g., 90-100 = A, 80-89 = B). The function updates the grade column in the ExamResults table and returns the grade as output.

4. Automated Warning Issuance

- Develop a PL/SQL procedure to issue warnings automatically for students who have received a failing grade (status = 'Fail') in two or more courses. The procedure should check the ExamResults table and log warnings in the Warnings table with details.

5. Audit Trail for Registration

- Create a BEFORE INSERT and BEFORE DELETE trigger for the Register table. The trigger logs every registration or deregistration event into the AuditTrail table. Each log entry should include the table name (Register), operation type (INSERT/DELETE), and relevant data changes, along with a timestamp.

6. Course Performance Report

- Write a PL/SQL cursor that generates a performance report for a specific course. The cursor should retrieve information from the ExamResults and Register tables, including student IDs, grades, and overall pass/fail statistics for the course. The report should summarize the number of students who passed and failed the course.

7. Exam Schedule Management

- Create a PL/SQL block to retrieve and display the schedule of exams for a specific course. The block should query the Exams table and display the course name, exam date, and type (e.g., midterm or final). If no exams are scheduled, the block should display an appropriate message.

8. Multi-Exam Grade Update with Transactions

- Develop a PL/SQL block that processes grade updates for multiple exams in a single transaction. The block should update the ExamResults table for a list of registration_ids with new grades. If any error occurs, the block should roll back all changes to ensure data consistency.

9. Student Suspension Based on Warnings

- Create a PL/SQL procedure to suspend students who have received three or more warnings. The procedure should query the Warnings table to identify affected students and update their academic_status in the Students table to "Suspended." Ensure the procedure logs these updates into the AuditTrail table.

10. Advanced Grade Management and Data Integrity:

- Write a PL/SQL Function that accepts a student's ID and calculates their current Grade Point Average (GPA) based on the grades in the ExamResults table and the credit hours from the Courses table. The function should return the calculated GPA.
- Create a Trigger on the ExamResults table that fires BEFORE UPDATE on the grade column. The trigger should check if the user attempting the update is an authorized user (e.g., a specific professor or admin role). If the user is unauthorized, the trigger should raise an application error to prevent the grade from being changed.

Bonus

11. **Blocker-Waiting Situation**

- Demonstrate generating a blocker-waiting situation using two transactions by User 1 and User 2. User 1 locks the students' table while updating the academic status of a student, and User 2 tries to simultaneously insert a new registration record for the same student in the Register table.

12. **Identifying Blocker and Waiting Sessions**

- Identify the sessions in the blocker-waiting situation created in Question 10 using SID and SERIAL# for both the blocker and waiting sessions. Display the details of these sessions and how they are resolved.

Instructions

- All team members should be aware of every task, as each member may be asked randomly about any task. Additionally, you must have a clear understanding of all the concepts studied in Oracle.
- All scripts should be placed in a single Oracle file with clear comments and documentation explaining the logic behind each point.

- **Prohibition of AI Tools: The use of AI tools such as ChatGPT or similar platforms is strictly prohibited. Submissions will be evaluated with AI detection tools. Teams found using AI tools will receive a zero for this task. Ensure the work reflects your own understanding and effort.**