CodeMavercs

Database Procedures, Triggers, Views, and Transactions with Examples

This document provides examples of how to use the views, triggers, stored procedures, and transactions in a university database system. Detailed explanations are included for each example.

# 1. Using Views

Views can be queried like regular tables to fetch data. Here's how you can use the views we've created:

## Example 1: Querying Active Student Enrollments

```sql  
SELECT \* FROM ActiveStudentEnrollments;  
```

Explanation: This query retrieves all active students along with the courses they are enrolled in.

## Example 2: Viewing Course Offerings with Professors and Departments

```sql  
SELECT \* FROM CourseOfferings;  
```

Explanation: This query fetches all courses offered, along with the name of the professor teaching the course, and the department.

## Example 3: Getting Student Grades for Each Course

```sql  
SELECT \* FROM StudentGradesView;  
```

Explanation: This query retrieves all students and their corresponding grades in different courses.

# 2. Using Stored Procedures with Transactions

Stored procedures handle database operations like INSERT, UPDATE, DELETE, and SELECT. Below are examples of how to use them.

## Inserting Data into the Users Table

```sql  
EXEC InsertUser @Username = 'JohnDoe', @Password = 'JohnPass123', @Role = 'Admin';  
```

Explanation: This executes the InsertUser procedure to insert a new user.

## Updating a User's Information

```sql  
EXEC UpdateUser @UserID = 1, @Username = 'JohnUpdated', @Password = 'NewPass123', @Role = 'Admin';  
```

Explanation: This updates the user's details.

## Example: Insert a Professor with a Transaction

```sql  
BEGIN TRANSACTION;  
BEGIN TRY  
 EXEC InsertProfessor @ProfessorName = 'Dr. James Brown', @HireDate = '2019-08-15', @DepartmentID = 1, @EntityID = 10;  
 COMMIT TRANSACTION;  
 PRINT 'Insert succeeded';  
END TRY  
BEGIN CATCH  
 ROLLBACK TRANSACTION;  
 PRINT 'Insert failed. Transaction rolled back';  
END CATCH;  
```

Explanation: This transaction inserts a professor and rolls back if an error occurs.

# 3. Using Triggers to Manage Audits Automatically

## Example 1: Inserting Data and Observing Triggers in Action

```sql  
EXEC InsertStudent @Name = 'Emma Watson', @DateOfBirth = '1995-04-15', @DepartmentID = 2, @EnrollmentDate = '2024-01-12', @EntityID = 12;  
```

Explanation: This will insert a new student and the trigger will log this change into the ChangeLog table.

## Example 2: Updating Data and Observing Trigger Logging

```sql  
EXEC UpdateStudent @StudentID = 1, @Name = 'Emma Thompson', @DateOfBirth = '1995-04-15', @DepartmentID = 2, @EnrollmentDate = '2024-01-12', @EntityID = 12;  
```

Explanation: When the student's name is updated, the UPDATE trigger will log the change.

# 4. Using Triggers for Auditing Data Changes

When any change (insert, update, delete) occurs, the corresponding trigger logs the data into the ChangeLog table.

## Example: Auditing Student Updates

```sql  
SELECT \* FROM ChangeLog WHERE TableName = 'Students';  
```

Explanation: This query will show logs of all changes made to the Students table.

# 5. Querying Nested Data with Views

## Example 1: List of Students with Low Grades in a Course

```sql  
SELECT \* FROM LowGrades WHERE Grade < 'C';  
```

Explanation: This query returns all students with a grade below 'C'.

## Example 2: Getting Enrollment Summary

```sql  
SELECT \* FROM CourseEnrollmentSummary;  
```

Explanation: This shows how many students are enrolled in each course.

# 6. Dealing with Real-life Data Scenarios

## Example 1: Managing Enrollments

```sql  
BEGIN TRANSACTION;  
BEGIN TRY  
 EXEC InsertStudent @Name = 'John Doe', @DateOfBirth = '2000-05-10', @DepartmentID = 3, @EnrollmentDate = '2024-09-01', @EntityID = 15;  
 EXEC InsertEnrollment @StudentID = SCOPE\_IDENTITY(), @CourseID = 1, @EnrollmentDate = '2024-09-01', @EntityID = 16;  
 COMMIT TRANSACTION;  
 PRINT 'Enrollment succeeded';  
END TRY  
BEGIN CATCH  
 ROLLBACK TRANSACTION;  
 PRINT 'Enrollment failed. Transaction rolled back';  
END CATCH;  
```

Explanation: This transaction enrolls a new student in a course and rolls back if there's any issue.