**Trigger\_V2**

**BANGDIEMDANH**

**Đảm bảo rằng chỉ có học sinh thuộc lớp học được phép điểm danh trong bảng này.**

CREATE TRIGGER CheckAttendance

ON BANDIEMDANH

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted AS i

LEFT JOIN XEPLOP AS x ON i.StudentID = x.UserID AND i.ClassID = x.ClassID

WHERE x.UserID IS NULL;

)

BEGIN

RAISERROR ('Invalid student for attendance', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**Đảm bảo rằng chỉ có thể thêm bản ghi điểm danh cho học sinh thuộc lớp học tương ứng.**

CREATE TRIGGER CheckAttendanceRecord

ON BANDIEMDANH

FOR INSERT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM BANDIEMDANH AS bd

JOIN inserted AS i ON i.StudentID = bd.StudentID

AND i.ClassID = bd.ClassID

AND i.AttendanceID <> bd.AttendanceID;

)

BEGIN

RAISERROR ('Attendance record already exists for the same student and class', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**BANGDIEM**

**Đảm bảo rằng điểm số nằm trong khoảng từ 0 đến 10.**

CREATE TRIGGER CheckGrade

ON BANGDIEM

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE Grade < 0 OR Grade > 10;

)

BEGIN

RAISERROR ('Invalid grade value', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**Đảm bảo rằng chỉ có thể thêm bản ghi điểm cho học sinh thuộc lớp học tương ứng.**

CREATE TRIGGER CheckGradeRecord

ON BANGDIEM

FOR INSERT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM BANGDIEM AS bd

JOIN inserted AS i ON i.StudentID = bd.StudentID

AND i.ClassID = bd.ClassID

AND i.GradeID <> bd.GradeID;

)

BEGIN

RAISERROR ('Grade record already exists for the same student and class', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**LOP**

**Trigger để tự động cập nhật số lượng học viên trong lớp khi có học viên mới được thêm vào:**

CREATE TRIGGER UpdateEnrollmentCount

ON LOP

AFTER INSERT

AS

BEGIN

UPDATE LOP

SET so\_luong\_hoc\_vien = (

SELECT COUNT(\*)

FROM HOCSINH

WHERE ClassID = inserted.ClassID

)

FROM LOP

INNER JOIN inserted ON LOP.ClassID = inserted.ClassID;

END;

**Trigger để tự động xóa thông tin lớp học khi lớp học bị xóa khỏi cơ sở dữ liệu:**

CREATE TRIGGER DeleteClassInfo

ON LOP

INSTEAD OF DELETE //UPDATE

AS

BEGIN

DELETE FROM LOP

WHERE ClassID IN (SELECT ClassID FROM deleted);

-- Xóa các bảng liên quan (ví dụ: BANGDIEM, BANDIEMDANH, ...)

DELETE FROM BANGDIEM

WHERE ClassID IN (SELECT ClassID FROM deleted);

DELETE FROM BANDIEMDANH

WHERE ClassID IN (SELECT ClassID FROM deleted);

-- Cập nhật thông tin khác nếu cần thiết

-- ...

-- Xóa các học sinh trong lớp

DELETE FROM HOCSINH

WHERE ClassID IN (SELECT ClassID FROM deleted);

END;

**Đảm bảo rằng số lượng học sinh trong một lớp không vượt quá giới hạn cho phép.**

CREATE TRIGGER CheckClassCapacity

ON LOP

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM LOP AS l

JOIN (

SELECT id\_lop\_hoc, COUNT(\*) AS student\_count

FROM HOCSINH

GROUP BY id\_lop\_hoc

) AS hs ON hs.id\_lop\_hoc, = lh.id\_lop\_hoc

JOIN inserted AS i ON i.id\_lop\_hoc= lh.id\_lop\_hoc

WHERE hs.student\_count > i.so\_luong\_gioi\_han;

)

BEGIN

RAISERROR ('Class capacity exceeded', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**HOCSINH**

1. Trigger để tự động cập nhật trạng thái điểm danh của học viên khi có bản ghi mới trong bảng "DiemDanh" (Attendance):

Trạng thái: 1 có mặt,2 vào trễ,3 vắng có phép,4 vắng không thông

CREATE TRIGGER UpdateAttendanceStatus

ON DiemDanh

AFTER INSERT

AS

BEGIN

UPDATE HocVien

SET AttendanceStatus = (

SELECT MAX(DiemDanh)

FROM DiemDanh

WHERE HocVienID = inserted.HocVienID

)

WHERE HocVienID IN (

SELECT HocVienID

FROM inserted

);

END;

1. Đảm bảo rằng ngày sinh (DateOfBirth) của học sinh không lớn hơn ngày hiện tại

CREATE TRIGGER CheckStudentBirthdate

ON HOCSINH

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE DateOfBirth > GETDATE();

)

BEGIN

RAISERROR ('Invalid student birthdate', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**TAIKHOAN**

1. Trigger kiểm tra trùng lặp UserID:

-- Create trigger to check for duplicate UserID in TAIKHOAN table

CREATE TRIGGER CheckDuplicateUserID

BEFORE INSERT ON TAIKHOAN

FOR EACH ROW

BEGIN

DECLARE duplicate\_count INT;

-- Check if the new UserID already exists in TAIKHOAN table

SELECT COUNT(\*) INTO duplicate\_count

FROM TAIKHOAN

WHERE UserID = NEW.UserID;

-- If duplicate UserID found, raise an error

IF duplicate\_count > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'UserID already exists in TAIKHOAN table';

END IF;

END;

1. Trigger kiểm tra trùng lặp Username:

-- Create trigger to check for duplicate Username in TAIKHOAN table

CREATE TRIGGER CheckDuplicateUsername

BEFORE INSERT ON TAIKHOAN

FOR EACH ROW

BEGIN

DECLARE duplicate\_count INT;

-- Check if the new Username already exists in TAIKHOAN table

SELECT COUNT(\*) INTO duplicate\_count

FROM TAIKHOAN

WHERE Username = NEW.Username;

-- If duplicate Username found, raise an error

IF duplicate\_count > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Username already exists in TAIKHOAN table';

END IF;

END;

1. Đảm bảo rằng mỗi tài khoản có một vai trò hợp lệ.

CREATE TRIGGER CheckRole

ON TAIKHOAN

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted AS i

LEFT JOIN UserRole AS r ON i.RoleID = r.RoleID

WHERE r.RoleID IS NULL;

)

BEGIN

RAISERROR ('Invalid role for account', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**XEPLOP**

**Đảm bảo rằng mỗi học sinh chỉ được ghi danh vào một lớp học duy nhất.**

CREATE TRIGGER CheckUniqueEnrollment

ON XEPLOP

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT UserID

FROM inserted

GROUP BY UserID

HAVING COUNT(\*) > 1;

)

BEGIN

RAISERROR ('Duplicate enrollment for a student', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**PHANHOI**

**Đảm bảo rằng nội dung phản hồi không được để trống.**

CREATE TRIGGER CheckFeedbackContent

ON PHANHOI

FOR INSERT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE Content IS NULL OR LTRIM(RTRIM(Content)) = ‘’;

)

BEGIN

RAISERROR ('Feedback content cannot be empty', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**COURSE**

**Đảm bảo rằng ngày tạo khóa học (ngay\_tao) không được lớn hơn ngày hiện tại.**

CREATE TRIGGER CheckCourseCreationDate

ON COURSE

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE ngay\_tao > GETDATE();

)

BEGIN

RAISERROR ('Invalid course creation date', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**Đảm bảo rằng thời gian bắt đầu (ngay\_bd) khóa học không sau thời gian kết thúc (ngay\_kt).**

CREATE TRIGGER CheckCourseDuration

ON KHOAHOC

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE ngay\_bd > ngay\_kt;

)

BEGIN

RAISERROR ('Invalid course duration', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

**Đảm bảo tên khoá học không bị trùng**

CREATE TRIGGER CheckDuplicateCourse

ON COURSE

FOR INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM COURSE AS c

JOIN inserted AS i ON i.Coursename = c.Coursename

WHERE i.CourseID <> c.CourseID;

)

BEGIN

RAISERROR ('Duplicate subject name', 16, 1);

ROLLBACK TRANSACTION;

END

END