**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG THÀNH PHỐ HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------



**QUẢN LÝ PHÒNG GYM**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**SINH VIÊN THỰC HIỆN:**

1. 2001230550 - Nguyễn Ngà

2. 2001222443 - Nguyễn Thành Long

3. 2001230899 - Nguyễn Hoàng Thành

4. 2001230944 - Phạm Đặng Gia Thuận

**GVHD: Trần Quang Bình**

TP. HỒ CHÍ MINH, tháng 11 năm 2025

**MỤC LỤC**

[chương 1: XÂY DỰNG CƠ SỞ DỮ LIỆU 1](#_Toc215534399)

[1.1. Sơ đồ cấu trúc dữ liệu 1](#_Toc215534400)

[1.2. Mô tả thuộc tính từng bảng 1](#_Toc215534401)

[1.2.1. Nhóm quản lý sản phẩm & kho 1](#_Toc215534402)

[1.2.2. Nhóm quản lý khách hàng 2](#_Toc215534403)

[1.2.3. Nhóm quản lý nhân sự 3](#_Toc215534404)

[1.2.4. Nhóm dịch vụ (gói tập & lớp học) 4](#_Toc215534405)

[1.2.5. Nhóm đăng ký & lịch pt 5](#_Toc215534406)

[1.2.6. Nhóm bán hàng (giỏ hàng & hóa đơn) 6](#_Toc215534407)

[1.3. Thiết lập các ràng buộc 7](#_Toc215534408)

[1.3.1. Check 7](#_Toc215534409)

[1.3.2. Unique 8](#_Toc215534410)

[1.3.3. Default 9](#_Toc215534411)

[Chương 2. CÀI ĐẶT YÊU CẦU XỬ LÝ 10](#_Toc215534412)

[2.1. Giới thiệu chương 10](#_Toc215534413)

[2.2. Cài đặt Stored Procedure 10](#_Toc215534414)

[2.2.1. Nhóm Quản lý Giỏ hàng & Khách hàng 10](#_Toc215534415)

[2.2.2. Nhóm Quản lý Lớp học & Lịch học 16](#_Toc215534416)

[2.2.3. Nhóm Quản lý Nhân viên & Chuyên môn 24](#_Toc215534417)

[2.2.4. Nhóm Quản lý Sản phẩm & Loại sản phẩm 38](#_Toc215534418)

[2.2.5. Nhóm Quản lý Hình ảnh 48](#_Toc215534419)

[2.3. Cài đặt Trigger 51](#_Toc215534420)

[2.3.1. Kiểm tra dữ liệu & ràng buộc 51](#_Toc215534421)

[2.3.2. Quản lý & cập nhật dữ liệu 57](#_Toc215534422)

[2.4. Cài đặt Function 67](#_Toc215534423)

[Chương 3. QUẢN TRỊ HỆ THỐNG 71](#_Toc215534424)

[3.1. Chọn lựa cơ chế xác thực 71](#_Toc215534425)

[3.2. Phân quyền các actor trong cơ sở dữ liệu 71](#_Toc215534426)

[3.2.1. Actor quản lý nhân sự 71](#_Toc215534427)

[3.2.2. Actor khách hàng 72](#_Toc215534428)

[3.2.3. Actor quản lý sản phẩm 73](#_Toc215534429)

[3.2.4. Actor huấn luyện viên 74](#_Toc215534430)

[3.3. Sao lưu dữ liệu 75](#_Toc215534431)

[3.3.1. Sao lưu toàn bộ (Full Backup) 75](#_Toc215534432)

[3.3.2. Sao lưu khác biệt (Differential Backup) 75](#_Toc215534433)

[3.3.3. Sao lưu log giao dịch (Transaction Log Backup) 75](#_Toc215534434)

[Chương 4. XÂY DỰNG ỨNG DỤNG 77](#_Toc215534435)

[4.1. Chức năng các thành viên triển khai 77](#_Toc215534436)

[4.1.1. Thành viên “Nguyễn Thành Long” – Ứng dụng Quản lý sản phẩm 81](#_Toc215534437)

[4.1.2. Thành viên “Phạm Đặng Gia Thuận” – Ứng dụng Quản lí nhân viên 86](#_Toc215534438)

[4.1.3. Thành viên “Nguyễn Hoàng Thành” – Ứng dụng HLV 77](#_Toc215534439)

[4.1.4. Thành viên “Nguyễn Ngà” – Ứng dụng khách hàng 91](#_Toc215534440)

[4.2. KẾT LUẬN CHƯƠNG 103](#_Toc215534441)

[TÀI LIỆU THAM KHẢO 104](#_Toc215534442)

# chương 1: XÂY DỰNG CƠ SỞ DỮ LIỆU

## Sơ đồ cấu trúc dữ liệu

A computer screen shot of a computer

AI-generated content may be incorrect.

## Mô tả thuộc tính từng bảng

### Nhóm quản lý sản phẩm & kho

**Bảng LoaiSanPham (Danh mục sản phẩm)**

* MaLoaiSP: Khóa chính, mã định danh loại sản phẩm.
* TenLoaiSP: Tên loại (VD: Thực phẩm bổ sung, Dụng cụ tập...).

**Bảng SanPham (Hàng hóa)**

* MaSP: Khóa chính, mã định danh sản phẩm.
* TenSP: Tên sản phẩm.
* MaLoaiSP: Khóa ngoại, thuộc loại nào.
* DonGia: Giá bán gốc.
* SoLuongTon: Số lượng hiện còn trong kho.
* GiaKhuyenMai: Giá giảm (nếu có).
* Hang, XuatXu: Thương hiệu và Nơi sản xuất.
* BaoHanh: Thời gian bảo hành.
* MoTaChung, MoTaChiTiet: Thông tin giới thiệu sản phẩm.

**Bảng HINHANH (Thư viện ảnh sản phẩm)**

* MaHinh: Khóa chính.
* MaSP: Khóa ngoại, ảnh này của sản phẩm nào.
* Url: Đường dẫn/Tên file ảnh (VD: whey-gold.jpg).
* IsMain: Cờ đánh dấu (True/False) xem đây có phải ảnh đại diện chính không.

### Nhóm quản lý khách hàng

**Bảng LoaiKhachHang (Hạng thành viên)**

* MaLoaiKH: Khóa chính.
* TenLoai: Tên hạng (VD: HSSV, VIP, Member).
* MucGiam: Phần trăm giảm giá mặc định cho hạng này (VD: 10%).

**Bảng KhachHang (Thông tin thành viên)**

* MaKH: Khóa chính.
* TenKH, NgaySinh, SDT, Email, GioiTinh: Thông tin cá nhân.
* MaLoaiKH: Khóa ngoại, khách thuộc hạng nào (được cập nhật tự động dựa trên chi tiêu).
* TenDangNhap, MatKhau: Tài khoản để khách đăng nhập App/Web.

**Bảng DiaChi (Sổ địa chỉ giao hàng)**

* MaDC: Khóa chính.
* MaKH: Khóa ngoại, địa chỉ của ai.
* TinhThanhPho, QuanHuyen, PhuongXa, DiaChiCuThe: Chi tiết địa chỉ.
* LaDiaChiMacDinh: Cờ đánh dấu địa chỉ mặc định để giao hàng.

**Bảng CheckIn (Lịch sử ra vào)**

* MaCheckIn: Khóa chính.
* MaKH: Khóa ngoại, ai đang check-in.
* ThoiGianVao: Giờ khách đến phòng tập.
* ThoiGianRa: Giờ khách về (có thể null nếu chưa về).
* TrangThai: Kết quả check-in ("Hợp lệ" hoặc "Hết hạn").

### Nhóm quản lý nhân sự

**Bảng ChucVu (Vị trí công việc)**

* MaChucVu: Khóa chính.
* TenChucVu: Tên chức vụ (VD: PT, HLV Lớp, Quản lý, Lễ tân).

**Bảng ChuyenMon (Kỹ năng)**

* MaCM: Khóa chính.
* TenChuyenMon: Tên kỹ năng (VD: Yoga, Boxing, Dance...).
* MoTa: Mô tả về chuyên môn này.

**Bảng NhanVien (Hồ sơ nhân viên)**

* MaNV: Khóa chính.
* TenNV, SDT, NgaySinh, GioiTinh: Thông tin cá nhân.
* MaChucVu: Khóa ngoại, nhân viên giữ chức vụ gì.
* TenDangNhap, MatKhau: Tài khoản đăng nhập trang Admin Portal.

**Bảng NhanVienChuyenMon (Bảng nối - Kỹ năng nhân viên)**

* MaNV, MaCM: Khóa chính phức hợp.

*Ý nghĩa:* Lưu trữ việc một nhân viên có thể có nhiều chuyên môn (VD: Anh A biết cả Yoga và Boxing).

### Nhóm dịch vụ (gói tập & lớp học)

**Bảng GoiTap (Gói Membership)**

* MaGoiTap: Khóa chính.
* TenGoi: Tên gói (VD: 1 Tháng, 1 Năm, VIP).
* ThoiHan: Số tháng hiệu lực (VD: 1, 6, 12).
* Gia: Giá bán gói.
* MoTa: Quyền lợi của gói.

**Bảng LopHoc (Các lớp học nhóm)**

* MaLop: Khóa chính.
* TenLop: Tên lớp (VD: Yoga Sáng 3-5-7).
* MaCM: Khóa ngoại, lớp này dạy môn gì.
* HocPhi: Giá tiền để tham gia lớp này.
* NgayBatDau, NgayKetThuc: Thời gian khóa học diễn ra.
* SoBuoi: Tổng số buổi học.
* SiSoToiDa: Giới hạn số người.
* MaNV: Giáo viên chủ nhiệm (mặc định) của lớp.

**Bảng LichLop (Thời khóa biểu từng buổi)**

* MaLichLop: Khóa chính.
* MaLop: Khóa ngoại, buổi này thuộc lớp nào.
* MaNV: Khóa ngoại, ai dạy buổi này (thường là GV chủ nhiệm, nhưng có thể thay thế).
* NgayHoc: Ngày diễn ra.
* GioBatDau, GioKetThuc: Khung giờ học.

### Nhóm đăng ký & lịch pt

**Bảng DangKyGoiTap (Hồ sơ Membership của khách)**

* MaDKGT: Khóa chính.
* MaKH: Khách nào đăng ký.
* MaGoiTap: Đăng ký gói nào.
* NgayDangKy: Ngày mua.
* NgayBatDau, NgayKetThuc: Thời hạn sử dụng thẻ tập.
* TrangThai: "Còn hiệu lực", "Hết hạn".

**Bảng DangKyLop (Danh sách học viên lớp)**

* MaDKLop: Khóa chính.
* MaKH: Khách nào.
* MaLop: Học lớp nào.
* NgayDangKy: Ngày mua.
* TrangThai: Trạng thái học viên.

**Bảng DangKyPT (Hợp đồng thuê PT)**

* MaDKPT: Khóa chính.
* MaKH: Khách nào.
* MaNV: Thuê PT nào.
* SoBuoi: Tổng số buổi đã mua (VD: 10 buổi).
* GiaMoiBuoi: Giá tiền 1 buổi.
* TrangThai: "Chờ duyệt", "Còn hiệu lực", "Hết số buổi".

**Bảng LichTapPT (Lịch hẹn 1-kèm-1)**

* MaLichPT: Khóa chính.
* MaDKPT: Khóa ngoại, thuộc hợp đồng PT nào (để biết Khách và PT).
* NgayTap, GioBatDau, GioKetThuc: Thời gian hẹn tập.
* TrangThai: "Chưa tập", "Đã tập", "Hủy".

### Nhóm bán hàng (giỏ hàng & hóa đơn)

**Bảng ChiTietGioHang (Giỏ hàng tạm)**

* MaChiTietGH: Khóa chính.
* MaKH: Giỏ của ai.
* MaSP, MaGoiTap, MaLop: Khóa ngoại (chỉ 1 trong 3 cột này có giá trị, 2 cột kia NULL) -> Để biết khách đang chọn mua món gì.
* SoLuong: Số lượng sản phẩm trong giỏ hàng.
* NgayThem: Thời gian thêm vào giỏ.

**Bảng HoaDon (Đơn hàng tổng)**

* MaHD: Khóa chính.
* MaKH: Khách hàng mua.
* NgayLap: Thời gian thanh toán.
* TongTien: Tổng giá trị hàng.
* GiamGia: Tiền giảm (theo hạng thành viên).
* ThanhTien: Tiền thực trả (TongTien - GiamGia).
* TrangThai: "Chưa thanh toán", "Đã thanh toán".

**Bảng ChiTietHoaDon (Chi tiết các món đã mua)**

* MaCTHD: Khóa chính.
* MaHD: Khóa ngoại, thuộc hóa đơn nào.
* MaSP, MaDKGT, MaDKPT, MaDKLop: Khóa ngoại, xác định món hàng đã mua là gì.
* SoLuong, DonGia: Lưu lại giá tại thời điểm mua (để sau này giá gốc có đổi thì hóa đơn cũ không bị sai).

## Thiết lập các ràng buộc

### Check

* **Logic thời gian:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng dữ liệu** | **Tên ràng buộc** | **Điều kiện kiểm tra** | **Mục đích logic** |
| DangKyGoiTap | CK\_DKGT\_Ngay | NgayKetThuc ≥ NgayBatDau | Đảm bảo ngày kết thúc không trước ngày bắt đầu |
| LichTapPT | CK\_LichPT\_Gio | GioKetThuc ≥ GioBatDau | Đảm bảo giờ kết thúc không trước giờ bắt đầu |
| LichLop | CK\_LichLop\_Gio | GioKetThuc ≥ GioBatDau | Tương tự, kiểm tra logic thời gian trong lịch lớp |

* **Logic trạng thái:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng dữ liệu** | **Tên ràng buộc** | **Giá trị hợp lệ của cột TrangThai** | **Mục đích logic** |
| HoaDon | CK\_TRANGTHAIHD | 'Hủy', 'Đã thanh toán', 'Chưa thanh toán' | Đảm bảo hóa đơn chỉ có 3 trạng thái hợp lệ |
| DangKyGoiTap | CK\_TRANGTHAIGT | 'Còn hiệu lực', 'Hết hạn', 'Đã hủy' | Đảm bảo trạng thái đăng ký gói tập hợp lệ |

* **Giá trị và định dạng**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng dữ liệu** | **Tên ràng buộc / Mô tả** | **Điều kiện kiểm tra** | **Mục đích kiểm tra** |
| SanPham | CHECK DonGia ≥ 0 | [DonGia] >= 0 | Đảm bảo giá sản phẩm không âm |
| GoiTap | CHECK Gia ≥ 0 | [Gia] >= 0 | Đảm bảo giá gói tập không âm |
| LopHoc | CHECK HocPhi ≥ 0 | [HocPhi] >= 0 | Đảm bảo học phí không âm |
| NhanVien | CK\_NhanVien\_SDT | SoDT LIKE '0%' AND LEN(SoDT)=10 AND NOT SoDT LIKE '%[^0-9]%' | Kiểm tra số điện thoại đúng định dạng |
| KhachHang | CHECK GioiTinh | [GIOITINH] = 'Nam' OR [GIOITINH] = 'Nữ' | Giới hạn giới tính hợp lệ |
| LoaiKhachHang | CHECK MucGiam | [MucGiam] >= 0 AND [MucGiam] <= 100 |  |

### Unique

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng** | **Cột bị ràng buộc** | **Loại ràng buộc** | **Mục đích nghiệp vụ** |
| KhachHang | Email | UNIQUE NONCLUSTERED | Mỗi khách hàng có một email duy nhất |
| KhachHang | SDT | UNIQUE NONCLUSTERED | Mỗi khách hàng có một số điện thoại duy nhất |
| ChucVu | TenChucVu | UNIQUE NONCLUSTERED | Không trùng tên chức vụ (VD: chỉ có 1 'Quản lý') |
| NhanVien | TenDangNhap | UNIQUE NONCLUSTERED | Mỗi nhân viên có một tên đăng nhập duy nhất |

### Default

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng dữ liệu** | **Cột áp dụng** | **Giá trị mặc định** | **Mục đích thiết lập mặc định** |
| HoaDon | TrangThai | 'Chưa thanh toán' | Gán trạng thái mặc định khi tạo hóa đơn mới |
| LichLop | TrangThai | 'Chưa diễn ra' | Gán trạng thái mặc định cho lịch lớp chưa bắt đầu |
| LopHoc | SiSoToiDa | 30 | Sĩ số tối đa mặc định cho lớp học mới |
| ChiTietGioHang | SoLuong | 1 | Mặc định số lượng sản phẩm khi thêm vào giỏ hàng |

# CÀI ĐẶT YÊU CẦU XỬ LÝ

## Giới thiệu chương

Chương này trình bày quá trình cài đặt các yêu cầu xử lý trong hệ quản trị cơ sở dữ liệu cho hệ thống quản lý phòng gym. Trong đó, nhóm đã sử dụng các cấu trúc xử lý quan trọng của SQL Server như Stored Procedure, User-defined Function, Trigger, Cursor, Transaction và cơ chế phân quyền người dùng. Việc áp dụng các cấu trúc này giúp hệ thống đảm bảo tính chính xác và toàn vẹn dữ liệu, hạn chế tối đa lỗi nghiệp vụ và hỗ trợ tốt cho các ứng dụng WinForm/Web trong quá trình kết nối và thao tác với cơ sở dữ liệu.

## Cài đặt Stored Procedure

### Nhóm Quản lý Giỏ hàng & Khách hàng

1. sp\_ThemVaoGioHang

Tác dụng:

Procedure này dùng để thêm sản phẩm vào giỏ hàng của khách hàng. Nó đảm bảo:

* Kiểm tra sự tồn tại của sản phẩm.
* Kiểm tra số lượng tồn kho trước khi thêm.
* Nếu sản phẩm đã có trong giỏ hàng, cập nhật số lượng.
* Nếu sản phẩm chưa có, thêm mới vào bảng ChiTietGioHang.
* Xử lý giao dịch an toàn với TRANSACTION, rollback khi có lỗi.

Các bước xử lý:

1. Khởi tạo và chuẩn bị
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
   * Khai báo các biến cần thiết: giá hiện tại, mã chi tiết giỏ hàng, số lượng tồn, số lượng đang có trong giỏ.
2. Bắt đầu giao dịch (BEGIN TRANSACTION)
   * Lấy đơn giá và số lượng tồn kho từ bảng SanPham dựa trên @MaSP.
3. Kiểm tra điều kiện sản phẩm
   * Nếu sản phẩm không tồn tại (DonGia IS NULL) → báo lỗi *"Sản phẩm không tồn tại!"* và rollback.
   * Nếu số lượng tồn kho ≤ 0 → báo lỗi *"Sản phẩm đã hết hàng!"* và rollback.
4. Kiểm tra giỏ hàng hiện tại của khách
   * Truy vấn bảng ChiTietGioHang để xem sản phẩm đã có trong giỏ chưa.
   * Lấy MaChiTietGH và số lượng hiện có (SoLuongDangCo).
5. Xử lý thêm/cập nhật giỏ hàng
   * Tính tổng số lượng mới = số lượng nhập thêm + số lượng đang có.
   * Nếu sản phẩm đã tồn tại trong giỏ (@MaChiTietGH\_TonTai IS NOT NULL):
     + UPDATE số lượng và đơn giá trong ChiTietGioHang.
     + Commit giao dịch và kết thúc.
   * Nếu sản phẩm chưa có trong giỏ:
     + INSERT bản ghi mới vào ChiTietGioHang với thông tin khách hàng, sản phẩm, số lượng, đơn giá và ngày thêm.
     + Commit giao dịch.
6. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi: rollback giao dịch.
   * Lấy thông báo lỗi (ERROR\_MESSAGE()) và raise lại để báo cho người dùng.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemVaoGioHang]

@MaKH INT,

@MaSP INT,

@SoLuong INT = 1

AS

BEGIN

SET NOCOUNT ON;

DECLARE @DonGiaHienTai DECIMAL(12,2);

DECLARE @MaChiTietGH\_TonTai INT = NULL;

DECLARE @SoLuongTon INT;

DECLARE @SoLuongDangCo INT = 0;

BEGIN TRY

BEGIN TRANSACTION;

SELECT @DonGiaHienTai = DonGia,

@SoLuongTon = SoLuongTon

FROM SanPham

WHERE MaSP = @MaSP;

IF (@DonGiaHienTai IS NULL)

BEGIN

RAISERROR (N'Sản phẩm không tồn tại!', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

IF (@SoLuongTon <= 0)

BEGIN

RAISERROR (N'Sản phẩm đã hết hàng!', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

SELECT @MaChiTietGH\_TonTai = MaChiTietGH,

@SoLuongDangCo = SoLuong

FROM ChiTietGioHang

WHERE MaKH = @MaKH AND MaSP = @MaSP;

DECLARE @TongSoLuong INT = @SoLuong + @SoLuongDangCo;

IF @MaChiTietGH\_TonTai IS NOT NULL

BEGIN

UPDATE ChiTietGioHang

SET SoLuong = @TongSoLuong,

DonGia = @DonGiaHienTai

WHERE MaChiTietGH = @MaChiTietGH\_TonTai;

COMMIT TRANSACTION;

RETURN;

END

INSERT INTO ChiTietGioHang (MaKH, MaSP, SoLuong, DonGia, NgayThem)

VALUES (@MaKH, @MaSP, @SoLuong, @DonGiaHienTai, GETDATE());

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrMsg NVARCHAR(4000) = ERROR\_MESSAGE();

RAISERROR(@ErrMsg, 16, 1);

END CATCH

END

1. sp\_KhachHangDangKy

Tác dụng:

Procedure này dùng để đăng ký tài khoản khách hàng mới trong hệ thống. Nó đảm bảo:

* Kiểm tra trùng lặp tên đăng nhập, số điện thoại và email trước khi thêm.
* Nếu dữ liệu hợp lệ, thêm bản ghi mới vào bảng KhachHang.
* Đảm bảo tính toàn vẹn dữ liệu bằng giao dịch (TRANSACTION) và rollback khi có lỗi.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
   * Bắt đầu giao dịch (BEGIN TRANSACTION).
2. Kiểm tra tên đăng nhập
   * Nếu @TenDangNhap đã tồn tại trong bảng KhachHang → báo lỗi *"Tên đăng nhập này đã được sử dụng. Vui lòng chọn tên khác."* và rollback.
3. Kiểm tra số điện thoại (nếu có nhập)
   * Nếu @SDT không rỗng và đã tồn tại trong bảng KhachHang → báo lỗi *"Số điện thoại này đã được đăng ký cho một tài khoản khác."* và rollback.
4. Kiểm tra email (nếu có nhập)
   * Nếu @Email không rỗng và đã tồn tại trong bảng KhachHang → báo lỗi *"Email này đã được đăng ký cho một tài khoản khác."* và rollback.
5. Thêm khách hàng mới
   * Nếu tất cả kiểm tra đều hợp lệ → thực hiện INSERT vào bảng KhachHang với các thông tin:
     + Tên khách hàng (TenKH)
     + Ngày sinh (NgaySinh)
     + Số điện thoại (SDT)
     + Email (Email)
     + Giới tính (GioiTinh)
     + Tên đăng nhập (TenDangNhap)
     + Mật khẩu (MatKhau)
   * Sau khi thêm thành công → COMMIT TRANSACTION.
6. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi: rollback giao dịch.
   * Dùng THROW để ném lỗi ra ngoài cho hệ thống xử lý.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_KhachHangDangKy]

@TenKH NVARCHAR(100),

@GioiTinh NVARCHAR(60),

@NgaySinh DATE = NULL,

@SDT NVARCHAR(15) = NULL,

@Email NVARCHAR(100) = NULL,

@TenDangNhap VARCHAR(300),

@MatKhau VARCHAR(1000)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRANSACTION;

BEGIN TRY

IF EXISTS (SELECT 1 FROM KhachHang WHERE TenDangNhap = @TenDangNhap)

BEGIN

RAISERROR (N'Tên đăng nhập này đã được sử dụng. Vui lòng chọn tên khác.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

IF @SDT IS NOT NULL AND EXISTS (SELECT 1 FROM KhachHang WHERE SDT = @SDT)

BEGIN

RAISERROR (N'Số điện thoại này đã được đăng ký cho một tài khoản khác.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

IF @Email IS NOT NULL AND EXISTS (SELECT 1 FROM KhachHang WHERE Email = @Email)

BEGIN

RAISERROR (N'Email này đã được đăng ký cho một tài khoản khác.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

INSERT INTO KhachHang (TenKH, NgaySinh, SDT, Email, GioiTinh, TenDangNhap, MatKhau)

VALUES (@TenKH, @NgaySinh, @SDT, @Email, @GioiTinh, @TenDangNhap, @MatKhau);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

THROW;

END CATCH

END

1. sp\_KhachHangLogin

Tác dụng:

Procedure này dùng để xác thực thông tin đăng nhập của khách hàng. Nó kiểm tra xem tên đăng nhập và mật khẩu có khớp với dữ liệu trong bảng KhachHang hay không, và nếu đúng thì trả về thông tin cơ bản của khách hàng.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Truy vấn dữ liệu khách hàng
   * Thực hiện SELECT các trường:
     + MaKH (Mã khách hàng)
     + TenKH (Tên khách hàng)
     + SDT (Số điện thoại)
     + Email (Địa chỉ email)
   * Điều kiện lọc:
     + TenDangNhap = @TenDangNhap
     + MatKhau = @MatKhau
3. Kết quả trả về
   * Nếu thông tin đăng nhập hợp lệ → trả về thông tin khách hàng.
   * Nếu không hợp lệ → không trả về dữ liệu nào (result set rỗng).

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_KhachHangLogin]

@TenDangNhap VARCHAR(300),

@MatKhau VARCHAR(1000)

AS

BEGIN

SET NOCOUNT ON;

SELECT

MaKH,

TenKH,

SDT,

Email

FROM KhachHang

WHERE TenDangNhap = @TenDangNhap AND MatKhau = @MatKhau;

END

### Nhóm Quản lý Lớp học & Lịch học

1. sp\_ThemLichTapPT

Tác dụng:

Cho phép thêm một bản ghi mới vào bảng LichTapPT với các thông tin: mã đăng ký PT, ngày tập, giờ bắt đầu, giờ kết thúc, trạng thái.

Đảm bảo dữ liệu được thêm trong một transaction an toàn:

* Nếu thêm thành công → COMMIT TRANSACTION.
* Nếu có lỗi → ROLLBACK TRANSACTION để không lưu dữ liệu sai.

Có thêm lệnh WAITFOR DELAY '00:00:7' nhằm mô phỏng độ trễ (7 giây), thường dùng để kiểm thử tình huống đồng thời hoặc hiệu năng.

Các bước xử lý:

1. Thiết lập mức cô lập giao dịch
   * SET TRANSACTION ISOLATION LEVEL READ COMMITTED; → đảm bảo chỉ đọc dữ liệu đã được commit, tránh đọc dữ liệu chưa xác nhận.
2. Bắt đầu thủ tục
   * SET NOCOUNT ON; → tránh trả về số dòng ảnh hưởng, giúp kết quả gọn hơn.
3. Mở transaction
   * BEGIN TRANSACTION; → bắt đầu giao dịch.
4. Thực hiện khối TRY
   * INSERT INTO LichTapPT (...) VALUES (...) → thêm lịch tập mới.
   * WAITFOR DELAY '00:00:7'; → tạo độ trễ 7 giây.
   * COMMIT TRANSACTION; → xác nhận giao dịch nếu không có lỗi.
   * PRINT thông báo thành công.
5. Xử lý khối CATCH
   * Nếu có lỗi trong khối TRY:
     + ROLLBACK TRANSACTION; → hủy bỏ giao dịch.
     + THROW; → ném lỗi ra ngoài để hệ thống hoặc ứng dụng xử lý.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemLichTapPT]

@MaDKPT INT,

@NgayTap DATE,

@GioBatDau TIME,

@GioKetThuc TIME,

@TrangThai NVARCHAR(20)

AS

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

BEGIN

SET NOCOUNT ON;

BEGIN TRANSACTION;

BEGIN TRY

INSERT INTO LichTapPT (MaDKPT, NgayTap, GioBatDau, GioKetThuc, TrangThai)

VALUES (@MaDKPT, @NgayTap, @GioBatDau, @GioKetThuc, @TrangThai);

WAITFOR DELAY '00:00:7';

COMMIT TRANSACTION;

PRINT N'Thêm lịch tập PT thành công.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END

1. sp\_ThemLopHocVaLich

Tác dụng:

Procedure này dùng để thêm mới một lớp học và tự động sinh lịch học định kỳ hàng tuần trong khoảng thời gian từ ngày bắt đầu đến ngày kết thúc. Nó đảm bảo:

* Tạo bản ghi lớp học trong bảng LopHoc.
* Sinh ra các buổi học (lịch học) theo chu kỳ tuần trong bảng LichLop.
* Đảm bảo tính toàn vẹn dữ liệu bằng giao dịch (TRANSACTION) và rollback khi có lỗi.

Các bước xử lý:

1. Khởi tạo giao dịch
   * Bắt đầu TRANSACTION.
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
2. Thêm lớp học mới
   * Thực hiện INSERT vào bảng LopHoc với các thông tin:
     + Tên lớp (TenLop)
     + Mã nhân viên phụ trách (MaNV)
     + Mã chuyên môn (MaCM)
     + Học phí (HocPhi)
     + Ngày bắt đầu (NgayBatDau)
     + Ngày kết thúc (NgayKetThuc)
     + Sĩ số tối đa (SiSoToiDa)
   * Lấy mã lớp vừa thêm (@MaLop = SCOPE\_IDENTITY()).
3. Sinh lịch học tự động
   * Khởi tạo biến @Ngay = @NgayBatDau.
   * Dùng vòng lặp WHILE để duyệt từ ngày bắt đầu đến ngày kết thúc:
     + Mỗi vòng lặp, thực hiện INSERT vào bảng LichLop với thông tin:
       - Mã lớp (MaLop)
       - Mã nhân viên (MaNV)
       - Ngày học (@Ngay)
       - Giờ bắt đầu (@GioBatDau)
       - Giờ kết thúc (@GioKetThuc)
     + Sau đó tăng ngày học lên 1 tuần (DATEADD(WEEK, 1, @Ngay)).
4. Hoàn tất giao dịch
   * Nếu không có lỗi → COMMIT TRANSACTION.
   * In thông báo *"Thêm lớp học và sinh lịch tự động thành công."*.
5. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi: rollback giao dịch.
   * Lấy thông báo lỗi (ERROR\_MESSAGE()) và raise lại để báo cho người dùng.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemLopHocVaLich]

@TenLop NVARCHAR(50),

@MaCM INT,

@HocPhi DECIMAL(12,2),

@NgayBatDau DATE,

@NgayKetThuc DATE,

@SiSoToiDa INT,

@MaNV INT,

@GioBatDau TIME(7),

@GioKetThuc TIME(7)

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

SET NOCOUNT ON;

DECLARE @MaLop INT;

INSERT INTO LopHoc (TenLop, MaNV, MaCM, HocPhi, NgayBatDau, NgayKetThuc, SiSoToiDa)

VALUES (@TenLop, @MaNV, @MaCM, @HocPhi, @NgayBatDau, @NgayKetThuc, @SiSoToiDa);

SET @MaLop = SCOPE\_IDENTITY();

DECLARE @Ngay DATE = @NgayBatDau;

WHILE @Ngay <= @NgayKetThuc

BEGIN

INSERT INTO LichLop(MaLop, MaNV, NgayHoc, GioBatDau, GioKetThuc)

VALUES (@MaLop, @MaNV, @Ngay, @GioBatDau, @GioKetThuc);

SET @Ngay = DATEADD(WEEK, 1, @Ngay);

END

COMMIT TRANSACTION;

PRINT N'Thêm lớp học và sinh lịch tự động thành công.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

RAISERROR (@ErrorMessage, 16, 1);

END CATCH

END

1. sp\_ThemLichLop

Tác dụng:

Procedure này dùng để thêm một buổi học (lịch lớp) cho lớp học đã có. Nó đảm bảo rằng chỉ những nhân viên có chức vụ phù hợp (HLV Lớp hoặc PT) mới được phép thêm lịch, nhằm duy trì tính hợp lệ và đúng vai trò trong hệ thống.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra chức vụ nhân viên
   * Gọi hàm dbo.f\_KiemTraNhanVienCoChucVu để kiểm tra xem nhân viên có chức vụ HLV Lớp hoặc PT hay không.
   * Nếu nhân viên không có một trong hai chức vụ này → báo lỗi *"Nhân viên này không phải là HLV Lớp hoặc PT, không thể thêm vào lịch lớp."* và kết thúc procedure.
3. Thêm lịch lớp
   * Nếu nhân viên hợp lệ → thực hiện INSERT vào bảng LichLop với các thông tin:
     + Mã lớp (MaLop)
     + Ngày học (NgayHoc)
     + Giờ bắt đầu (GioBatDau)
     + Giờ kết thúc (GioKetThuc)
4. Thông báo kết quả
   * In ra thông báo *"Thêm lịch lớp thành công."* để xác nhận thao tác đã hoàn tất.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemLichLop]

@MaLop INT,

@MaNV INT,

@NgayHoc DATE,

@GioBatDau TIME,

@GioKetThuc TIME

AS

BEGIN

SET NOCOUNT ON;

IF (dbo.f\_KiemTraNhanVienCoChucVu(@MaNV, N'HLV Lớp') = 0 AND dbo.f\_KiemTraNhanVienCoChucVu(@MaNV, N'PT') = 0)

BEGIN

RAISERROR (N'Nhân viên này không phải là HLV Lớp hoặc PT, không thể thêm vào lịch lớp.', 16, 1);

RETURN;

END

INSERT INTO LichLop (MaLop, NgayHoc, GioBatDau, GioKetThuc)

VALUES (@MaLop, @NgayHoc, @GioBatDau, @GioKetThuc);

PRINT N'Thêm lịch lớp thành công.';

END

1. XoaLichVaLop

Tác dụng:

Procedure này dùng để **xóa toàn bộ lịch học và lớp học** dựa trên mã lớp (@MaLop). Nó đảm bảo rằng khi một lớp bị xóa thì tất cả các buổi học (lịch lớp) liên quan cũng được xóa theo, tránh dữ liệu rác hoặc tham chiếu sai.

Các bước xử lý:

1. Khởi tạo giao dịch
   * Bắt đầu TRANSACTION để đảm bảo tính toàn vẹn dữ liệu.
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
2. Xóa lịch học của lớp
   * Thực hiện DELETE từ bảng LichLop với điều kiện MaLop = @MaLop.
   * Điều này đảm bảo tất cả các buổi học liên quan đến lớp sẽ bị xóa trước.
3. Xóa lớp học
   * Thực hiện DELETE từ bảng LopHoc với điều kiện MaLop = @MaLop.
   * Sau khi lịch học đã bị xóa, lớp học chính cũng được xóa khỏi hệ thống.
4. Hoàn tất giao dịch
   * Thực hiện COMMIT TRANSACTION để xác nhận thay đổi.

Code trong SQL:

ALTER procedure [dbo].[XoaLichVaLop]

@MaLop int

As

Begin

Begin transaction;

Set nocount on;

delete from LichLop where MaLop = @MaLop

delete from LopHoc where MaLop = @MaLop

Commit transaction;

End

1. sp\_SuaLopHoc

Tác dụng:

Procedure này dùng để sửa thông tin lớp học và đồng bộ lại lịch học tương lai nếu có thay đổi về giáo viên hoặc thời gian. Nó đảm bảo:

* Kiểm tra sự tồn tại của lớp học trước khi sửa.
* Ngăn chặn việc trùng tên lớp với lớp khác.
* Cập nhật thông tin lớp học trong bảng LopHoc.
* Đồng bộ lịch học trong bảng LichLop cho các buổi chưa diễn ra.

Các bước xử lý:

1. Kiểm tra tồn tại lớp học
   * Nếu @MaLop không tồn tại trong bảng LopHoc → báo lỗi *"Lớp học không tồn tại."* và kết thúc procedure.
2. Kiểm tra trùng tên lớp
   * Nếu @TenLop đã tồn tại ở lớp khác (khác @MaLop) → báo lỗi *"Tên lớp học đã được sử dụng bởi một lớp khác."* và kết thúc procedure.
3. Cập nhật thông tin lớp học
   * Thực hiện UPDATE bảng LopHoc với các thông tin mới:
     + Tên lớp (TenLop)
     + Mã chuyên môn (MaCM)
     + Học phí (HocPhi)
     + Ngày bắt đầu (NgayBatDau)
     + Ngày kết thúc (NgayKetThuc)
     + Số buổi (SoBuoi)
     + Sĩ số tối đa (SiSoToiDa)
     + Mã nhân viên (MaNV)
4. Xử lý giờ học mới (nếu có)
   * Nếu @GioBatDau và @GioKetThuc được truyền vào:
     + Kiểm tra @GioKetThuc phải lớn hơn @GioBatDau. Nếu không → báo lỗi *"Giờ kết thúc phải lớn hơn giờ bắt đầu."*.
     + Gán giá trị giờ mới cho biến cập nhật (@GioBD\_Update, @GioKT\_Update).
   * Nếu không truyền giờ mới → giữ nguyên giờ cũ trong lịch học.
5. Cập nhật lịch học (chỉ lịch tương lai)
   * Thực hiện UPDATE bảng LichLop với các thay đổi:
     + Giáo viên (MaNV) nếu có giá trị mới.
     + Giờ bắt đầu (GioBatDau) nếu có giá trị mới.
     + Giờ kết thúc (GioKetThuc) nếu có giá trị mới.
   * Điều kiện: chỉ cập nhật các buổi học có NgayHoc >= ngày hiện tại.
6. Thông báo kết quả
   * In ra thông báo *"Cập nhật lớp học và đồng bộ lịch thành công."*.

Code trong SQL:4

ALTER PROCEDURE [dbo].[sp\_SuaLopHoc]

@MaLop INT,

@TenLop NVARCHAR(50),

@MaCM INT,

@HocPhi DECIMAL(12,2),

@NgayBatDau DATE,

@NgayKetThuc DATE,

@SoBuoi INT,

@SiSoToiDa INT,

@MaNV INT = NULL,

@GioBatDau TIME(7) = NULL,

@GioKetThuc TIME(7) = NULL

AS

BEGIN

SET NOCOUNT ON;

-- 1. Kiểm tra tồn tại

IF NOT EXISTS (SELECT 1 FROM LopHoc WHERE MaLop = @MaLop)

BEGIN

RAISERROR (N'Lớp học không tồn tại.', 16, 1); RETURN;

END

-- 2. Kiểm tra trùng tên

IF EXISTS (SELECT 1 FROM LopHoc WHERE TenLop = @TenLop AND MaLop <> @MaLop)

BEGIN

RAISERROR (N'Tên lớp học "%s" đã được sử dụng bởi một lớp khác.', 16, 1, @TenLop); RETURN;

END

-- 3. Cập nhật thông tin Lớp Học

UPDATE LopHoc

SET TenLop = @TenLop, MaCM = @MaCM, HocPhi = @HocPhi,

NgayBatDau = @NgayBatDau, NgayKetThuc = @NgayKetThuc,

SoBuoi = @SoBuoi, SiSoToiDa = @SiSoToiDa, MaNV = @MaNV

WHERE MaLop = @MaLop;

-- 4. CẬP NHẬT LỊCH LỚP (QUAN TRỌNG)

-- Nếu có thay đổi Giờ hoặc Giáo viên, ta cập nhật lại LichLop

-- (Chỉ cập nhật những lịch chưa diễn ra để giữ lịch sử)

DECLARE @GioBD\_Update TIME(7);

DECLARE @GioKT\_Update TIME(7);

-- Nếu người dùng không nhập giờ mới (NULL hoặc 00:00), giữ nguyên giờ cũ trong LichLop

-- (Logic này sẽ được xử lý kỹ hơn ở C# để truyền NULL xuống)

IF @GioBatDau IS NOT NULL AND @GioKetThuc IS NOT NULL

BEGIN

IF @GioKetThuc <= @GioBatDau

BEGIN

RAISERROR (N'Giờ kết thúc phải lớn hơn giờ bắt đầu.', 16, 1); RETURN;

END

SET @GioBD\_Update = @GioBatDau;

SET @GioKT\_Update = @GioKetThuc;

END

-- Cập nhật Lịch: Đổi Giáo viên và Đổi Giờ (nếu có)

UPDATE LichLop

SET

MaNV = ISNULL(@MaNV, MaNV), -- Nếu MaNV mới khác NULL thì cập nhật

GioBatDau = ISNULL(@GioBD\_Update, GioBatDau), -- Nếu có giờ mới thì cập nhật

GioKetThuc = ISNULL(@GioKT\_Update, GioKetThuc)

WHERE MaLop = @MaLop

AND NgayHoc >= CAST(GETDATE() AS DATE); -- Chỉ cập nhật lịch tương lai

PRINT N'Cập nhật lớp học và đồng bộ lịch thành công.';

END

### Nhóm Quản lý Nhân viên & Chuyên môn

1. sp\_ThemNhanVien

Tác dụng:

Procedure này dùng để thêm mới một nhân viên vào hệ thống. Nó đảm bảo:

* Kiểm tra trùng lặp tên đăng nhập trước khi thêm.
* Nếu hợp lệ, thêm thông tin nhân viên vào bảng NhanVien.
* Thông báo kết quả sau khi thêm thành công.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra tên đăng nhập
   * Thực hiện truy vấn SELECT để kiểm tra xem @TenDangNhap đã tồn tại trong bảng NhanVien hay chưa.
   * Nếu tồn tại → báo lỗi *"Tên đăng nhập đã tồn tại."* và kết thúc procedure.
3. Thêm nhân viên mới
   * Nếu tên đăng nhập chưa tồn tại → thực hiện INSERT vào bảng NhanVien với các thông tin:
     + Mã chức vụ (MaChucVu)
     + Tên đăng nhập (TenDangNhap)
     + Mật khẩu (MatKhau)
     + Giới tính (GioiTinh)
     + Tên nhân viên (TenNV)
     + Số điện thoại (SDT) nếu có
     + Ngày sinh (NgaySinh) nếu có
4. Thông báo kết quả
   * In ra thông báo *"Thêm nhân viên thành công."* để xác nhận thao tác đã hoàn tất.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemNhanVien]

@MaChucVu INT,

@TenDangNhap VARCHAR(300),

@MatKhau VARCHAR(1000),

@GioiTinh NVARCHAR(60),

@TenNV NVARCHAR(100),

@SDT NVARCHAR(15) = NULL,

@NgaySinh DATE = NULL

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM NhanVien WHERE TenDangNhap = @TenDangNhap)

BEGIN

RAISERROR (N'Tên đăng nhập đã tồn tại.', 16, 1);

RETURN;

END

INSERT INTO NhanVien (MaChucVu, TenDangNhap, MatKhau, GioiTinh, TenNV, SDT, NgaySinh)

VALUES (@MaChucVu, @TenDangNhap, @MatKhau, @GioiTinh, @TenNV, @SDT, @NgaySinh);

PRINT N'Thêm nhân viên thành công.';

END

1. sp\_ThemChucVu

Tác dụng:

Procedure này dùng để thêm mới một chức vụ vào hệ thống. Nó đảm bảo:

* Kiểm tra trùng lặp tên chức vụ trước khi thêm.
* Nếu hợp lệ, thêm chức vụ mới vào bảng ChucVu.
* Thông báo kết quả sau khi thêm thành công.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra trùng tên chức vụ
   * Thực hiện truy vấn SELECT để kiểm tra xem @TenChucVu đã tồn tại trong bảng ChucVu hay chưa.
   * Nếu tồn tại → báo lỗi *"Tên chức vụ đã tồn tại."* và kết thúc procedure.
3. Thêm chức vụ mới
   * Nếu tên chức vụ chưa tồn tại → thực hiện INSERT vào bảng ChucVu với giá trị @TenChucVu.
4. Thông báo kết quả
   * In ra thông báo *"Thêm chức vụ thành công."* để xác nhận thao tác đã hoàn tất.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemChucVu]

@TenChucVu NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM ChucVu WHERE TenChucVu = @TenChucVu)

BEGIN

RAISERROR (N'Tên chức vụ đã tồn tại.', 16, 1);

RETURN;

END

INSERT INTO ChucVu (TenChucVu) VALUES (@TenChucVu);

PRINT N'Thêm chức vụ thành công.';

END

1. sp\_ThemChuyenMon

Tác dụng:

Procedure này dùng để thêm mới một chuyên môn vào hệ thống. Nó đảm bảo:

* Kiểm tra trùng lặp tên chuyên môn trước khi thêm.
* Nếu hợp lệ, thêm chuyên môn mới cùng mô tả (nếu có) vào bảng ChuyenMon.
* Thông báo kết quả sau khi thêm thành công.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra trùng tên chuyên môn
   * Thực hiện truy vấn SELECT để kiểm tra xem @TenChuyenMon đã tồn tại trong bảng ChuyenMon hay chưa.
   * Nếu tồn tại → báo lỗi *"Tên chuyên môn này đã tồn tại. Vui lòng chọn tên khác."* và kết thúc procedure.
3. Thêm chuyên môn mới
   * Nếu tên chuyên môn chưa tồn tại → thực hiện INSERT vào bảng ChuyenMon với các giá trị:
     + Tên chuyên môn (TenChuyenMon)
     + Mô tả (MoTa) nếu có
4. Thông báo kết quả
   * In ra thông báo *"Thêm chuyên môn thành công."* để xác nhận thao tác đã hoàn tất.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemChuyenMon]

@TenChuyenMon NVARCHAR(50),

@MoTa NVARCHAR(500) = NULL

AS

BEGIN

SET NOCOUNT ON;

-- Kiểm tra trùng tên

IF EXISTS (SELECT 1 FROM ChuyenMon WHERE TenChuyenMon = @TenChuyenMon)

BEGIN

RAISERROR (N'Tên chuyên môn này đã tồn tại. Vui lòng chọn tên khác.', 16, 1);

RETURN;

END

INSERT INTO ChuyenMon (TenChuyenMon, MoTa)

VALUES (@TenChuyenMon, @MoTa);

PRINT N'Thêm chuyên môn thành công.';

END

1. sp\_XoaNhanVien

Tác dụng:

Procedure này dùng để xóa nhân viên khỏi hệ thống một cách an toàn, đồng thời xử lý và cập nhật các dữ liệu liên quan để tránh lỗi ràng buộc hoặc mất cân bằng dữ liệu. Nó đặc biệt chú trọng đến các ràng buộc kế toán, đảm bảo không xóa nhân viên nếu đã có giao dịch tài chính.

Các bước xử lý:

1. Kiểm tra ràng buộc kế toán
   * Truy vấn bảng ChiTietHoaDon kết hợp với DangKyPT để kiểm tra xem nhân viên có liên quan đến giao dịch tài chính (ví dụ: bán gói PT trong hóa đơn).
   * Nếu có → báo lỗi *"Nhân viên này đã có giao dịch tài chính... Không thể xóa vì lý do kế toán."* và kết thúc procedure.
2. Bắt đầu giao dịch (BEGIN TRANSACTION)
   * Đảm bảo mọi thao tác xóa/cập nhật được thực hiện đồng bộ.
3. Xóa chuyên môn của nhân viên
   * Xóa dữ liệu trong bảng NhanVienChuyenMon với MaNV tương ứng.
4. Xóa lịch lớp của nhân viên
   * Xóa các buổi dạy lẻ tẻ trong bảng LichLop mà nhân viên này phụ trách.
5. Xử lý gói đăng ký PT
   * Xóa các lịch tập PT (LichTapPT) liên quan đến nhân viên.
   * Xóa các gói đăng ký PT (DangKyPT) mà nhân viên này phụ trách.
6. Gỡ nhân viên khỏi lớp học
   * Cập nhật bảng LopHoc, đặt MaNV = NULL cho các lớp mà nhân viên này đang làm chủ nhiệm.
   * Điều này giúp giữ lớp học nhưng loại bỏ nhân viên khỏi vai trò quản lý.
7. Xóa nhân viên
   * Sau khi xử lý toàn bộ dữ liệu liên quan, xóa nhân viên khỏi bảng NhanVien.
8. Hoàn tất giao dịch
   * Nếu tất cả thành công → COMMIT TRANSACTION.
   * In thông báo *"Đã xóa nhân viên và cập nhật dữ liệu liên quan thành công."*.
9. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi → rollback giao dịch.
   * Lấy thông báo lỗi (ERROR\_MESSAGE()) và raise lại để báo cho người dùng.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_XoaNhanVien]

@MaNV INT

AS

BEGIN

SET NOCOUNT ON;

-- 1. KIỂM TRA AN TOÀN (Ràng buộc Kế toán - Giữ nguyên)

IF EXISTS (

SELECT 1

FROM ChiTietHoaDon CT

JOIN DangKyPT DK ON CT.MaDKPT = DK.MaDKPT

WHERE DK.MaNV = @MaNV

)

BEGIN

RAISERROR (N'Nhân viên này đã có giao dịch tài chính (Bán gói PT trong Hóa đơn). Không thể xóa vì lý do kế toán. Hãy đổi mật khẩu hoặc hủy kích hoạt tài khoản.', 16, 1);

RETURN;

END

BEGIN TRANSACTION;

BEGIN TRY

-- 2. Xóa Chuyên Môn (Bảng NhanVienChuyenMon)

DELETE FROM NhanVienChuyenMon WHERE MaNV = @MaNV;

-- 3. Xóa Lịch Lớp (Bảng LichLop) - Xóa các buổi dạy lẻ tẻ

DELETE FROM LichLop WHERE MaNV = @MaNV;

-- 4. Xử lý Gói Đăng Ký PT

DELETE FROM LichTapPT

WHERE MaDKPT IN (SELECT MaDKPT FROM DangKyPT WHERE MaNV = @MaNV);

DELETE FROM DangKyPT WHERE MaNV = @MaNV;

-- ============================================================

-- == 5. MỚI THÊM: GỠ NHÂN VIÊN KHỎI LỚP HỌC (QUAN TRỌNG) ==

-- ============================================================

-- Cập nhật các lớp mà nhân viên này đang làm chủ nhiệm thành NULL

UPDATE LopHoc

SET MaNV = NULL

WHERE MaNV = @MaNV;

-- 6. Cuối cùng: Xóa Nhân Viên

DELETE FROM NhanVien WHERE MaNV = @MaNV;

COMMIT TRANSACTION;

PRINT N'Đã xóa nhân viên và cập nhật dữ liệu liên quan thành công.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

RAISERROR (@ErrorMessage, 16, 1);

END CATCH

END

1. sp\_XoaChucVu

Tác dụng:

Procedure này dùng để xóa một chức vụ khỏi hệ thống. Nó đảm bảo:

* Không cho phép xóa các chức vụ quan trọng (Quản lý, PT, HLV Lớp) vì hệ thống đang dựa vào chúng để phân quyền.
* Nếu chức vụ hợp lệ để xóa, sẽ gỡ chức vụ khỏi tất cả nhân viên đang giữ, sau đó xóa chức vụ khỏi bảng ChucVu.
* Đảm bảo dữ liệu nhất quán bằng cơ chế giao dịch (TRANSACTION) và rollback khi có lỗi.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
   * Lấy tên chức vụ từ bảng ChucVu dựa trên @MaChucVu.
2. Kiểm tra chức vụ quan trọng
   * Nếu tên chức vụ nằm trong danh sách: *Quản lý, PT, HLV Lớp* → báo lỗi *"Đây là chức vụ hệ thống quan trọng... Bạn không được phép xóa."* và kết thúc procedure.
3. Bắt đầu giao dịch (BEGIN TRANSACTION)
   * Đảm bảo mọi thao tác xóa/cập nhật được thực hiện đồng bộ.
4. Gỡ chức vụ khỏi nhân viên
   * Cập nhật bảng NhanVien, đặt MaChucVu = NULL cho tất cả nhân viên đang giữ chức vụ này.
   * Điều này giúp giữ nhân viên nhưng loại bỏ chức vụ của họ.
5. Xóa chức vụ
   * Thực hiện DELETE từ bảng ChucVu với MaChucVu tương ứng.
6. Hoàn tất giao dịch
   * Nếu thành công → COMMIT TRANSACTION.
   * In thông báo *"Đã xóa chức vụ thành công. Các nhân viên giữ chức vụ này đã được cập nhật thành (Chưa có chức vụ)."*.
7. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi → rollback giao dịch.
   * Lấy thông báo lỗi (ERROR\_MESSAGE()) và raise lại để báo cho người dùng.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_XoaChucVu]

@MaChucVu INT

AS

BEGIN

SET NOCOUNT ON;

-- 1. KIỂM TRA: Không cho xóa các chức vụ quan trọng của hệ thống

-- (Ví dụ: Quản lý, PT, HLV Lớp... vì logic code C# đang dựa vào tên của chúng)

DECLARE @TenChucVu NVARCHAR(50);

SELECT @TenChucVu = TenChucVu FROM ChucVu WHERE MaChucVu = @MaChucVu;

IF @TenChucVu IN (N'Quản lý', N'PT', N'HLV Lớp')

BEGIN

RAISERROR (N'Đây là chức vụ hệ thống quan trọng (đang được dùng để phân quyền Logic). Bạn không được phép xóa.', 16, 1);

RETURN;

END

BEGIN TRANSACTION;

BEGIN TRY

-- 2. GỠ CHỨC VỤ KHỎI NHÂN VIÊN (Thay vì xóa nhân viên)

-- Cập nhật tất cả nhân viên đang giữ chức vụ này thành NULL (Không có chức vụ)

UPDATE NhanVien

SET MaChucVu = NULL

WHERE MaChucVu = @MaChucVu;

-- 3. XÓA CHỨC VỤ

DELETE FROM ChucVu WHERE MaChucVu = @MaChucVu;

COMMIT TRANSACTION;

PRINT N'Đã xóa chức vụ thành công. Các nhân viên giữ chức vụ này đã được cập nhật thành (Chưa có chức vụ).';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

RAISERROR (@ErrorMessage, 16, 1);

END CATCH

END

1. sp\_XoaChuyenMon

Tác dụng:

Procedure này dùng để xóa một chuyên môn khỏi hệ thống. Nó đảm bảo:

* Không cho phép xóa chuyên môn nếu đã có lớp học liên quan (tránh mất dữ liệu lịch sử và ràng buộc tài chính).
* Nếu hợp lệ, gỡ chuyên môn khỏi tất cả nhân viên, sau đó xóa chuyên môn khỏi bảng ChuyenMon.
* Đảm bảo dữ liệu nhất quán bằng cơ chế giao dịch (TRANSACTION) và rollback khi có lỗi.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
2. Kiểm tra ràng buộc lớp học
   * Truy vấn bảng LopHoc để kiểm tra xem chuyên môn (@MaCM) có đang được sử dụng trong bất kỳ lớp học nào không.
   * Nếu có → báo lỗi *"Chuyên môn này đang được sử dụng trong các Lớp Học. Không thể xóa..."* và kết thúc procedure.
3. Bắt đầu giao dịch (BEGIN TRANSACTION)
   * Đảm bảo mọi thao tác xóa/cập nhật được thực hiện đồng bộ.
4. Gỡ chuyên môn khỏi nhân viên
   * Xóa dữ liệu trong bảng NhanVienChuyenMon với MaCM tương ứng.
   * Điều này giúp loại bỏ chuyên môn khỏi tất cả nhân viên đang sở hữu nó.
5. Xóa chuyên môn
   * Thực hiện DELETE từ bảng ChuyenMon với MaCM tương ứng.
6. Hoàn tất giao dịch
   * Nếu thành công → COMMIT TRANSACTION.
   * In thông báo *"Đã xóa chuyên môn thành công."*.
7. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi → rollback giao dịch.
   * Lấy thông báo lỗi (ERROR\_MESSAGE()) và raise lại để báo cho người dùng.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_XoaChuyenMon]

@MaCM INT

AS

BEGIN

SET NOCOUNT ON;

-- 1. Kiểm tra ràng buộc: Nếu đã có Lớp Học thuộc chuyên môn này -> CẤM XÓA

-- (Vì Lớp học dính đến tiền nong, lịch sử...)

IF EXISTS (SELECT 1 FROM LopHoc WHERE MaCM = @MaCM)

BEGIN

RAISERROR (N'Chuyên môn này đang được sử dụng trong các Lớp Học. Không thể xóa. Hãy sửa tên hoặc ngừng sử dụng thay vì xóa.', 16, 1);

RETURN;

END

BEGIN TRANSACTION;

BEGIN TRY

-- 2. Dọn dẹp bảng NhanVienChuyenMon (Gỡ môn này khỏi tất cả nhân viên)

DELETE FROM NhanVienChuyenMon WHERE MaCM = @MaCM;

-- 3. Xóa Chuyên Môn

DELETE FROM ChuyenMon WHERE MaCM = @MaCM;

COMMIT TRANSACTION;

PRINT N'Đã xóa chuyên môn thành công.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

RAISERROR (@ErrorMessage, 16, 1);

END CATCH

END

1. sp\_XoaNhanVienChuyenMon

Tác dụng:

Procedure này dùng để gỡ bỏ một chuyên môn khỏi nhân viên trong hệ thống. Nó đảm bảo rằng việc gỡ chuyên môn chỉ được thực hiện khi nhân viên không còn lịch dạy hoặc vai trò chủ nhiệm liên quan đến chuyên môn đó, nhằm giữ dữ liệu lịch sử và tránh sai lệch trong quản lý lớp học.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
2. Kiểm tra lịch lớp
   * Truy vấn bảng LichLop kết hợp với LopHoc để kiểm tra:
     + Nhân viên (@MaNV) có lịch dạy thuộc chuyên môn (@MaCM) hay không.
     + Nếu lịch học có ngày học lớn hơn hoặc bằng ngày hiện tại (tức là buổi học hôm nay hoặc tương lai) → báo lỗi *"Nhân viên này đang có lịch dạy lớp thuộc chuyên môn X trong tương lai. Không thể gỡ bỏ chuyên môn này."* và kết thúc procedure.
3. Kiểm tra vai trò chủ nhiệm lớp
   * Truy vấn bảng LopHoc để kiểm tra:
     + Nhân viên (@MaNV) có đang là giáo viên chủ nhiệm của lớp thuộc chuyên môn (@MaCM) hay không.
     + Nếu lớp đó chưa kết thúc (NgayKetThuc >= ngày hiện tại) → báo lỗi *"Nhân viên này đang là Giáo viên chủ nhiệm của lớp thuộc chuyên môn X. Vui lòng đổi giáo viên chủ nhiệm trước khi gỡ chuyên môn."* và kết thúc procedure.
4. Xóa chuyên môn khỏi nhân viên
   * Nếu vượt qua cả hai kiểm tra trên → thực hiện DELETE từ bảng NhanVienChuyenMon với điều kiện MaNV = @MaNV và MaCM = @MaCM.
5. Thông báo kết quả
   * In ra thông báo *"Đã gỡ bỏ chuyên môn khỏi nhân viên thành công."*.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_XoaNhanVienChuyenMon]

@MaNV INT,

@MaCM INT

AS

BEGIN

SET NOCOUNT ON;

-- 1. KIỂM TRA LỊCH LỚP (Đang dạy hoặc sắp dạy)

-- Nếu nhân viên này có lịch dạy (LichLop) thuộc lớp có chuyên môn này

-- VÀ lịch đó chưa kết thúc (Ngày học >= Hôm nay) -> CẤM GỠ

IF EXISTS (

SELECT 1

FROM LichLop LL

JOIN LopHoc LH ON LL.MaLop = LH.MaLop

WHERE LL.MaNV = @MaNV

AND LH.MaCM = @MaCM

AND LL.NgayHoc >= CAST(GETDATE() AS DATE) -- Chỉ chặn lịch tương lai/hôm nay

)

BEGIN

DECLARE @TenMon NVARCHAR(50);

SELECT @TenMon = TenChuyenMon FROM ChuyenMon WHERE MaCM = @MaCM;

RAISERROR (N'Nhân viên này đang có lịch dạy lớp thuộc chuyên môn "%s" trong tương lai. Không thể gỡ bỏ chuyên môn này.', 16, 1, @TenMon);

RETURN;

END

-- 2. KIỂM TRA CHỦ NHIỆM LỚP (Mới thêm)

-- Nếu nhân viên này đang là GV Chủ Nhiệm (MaNV trong bảng LopHoc)

-- của một lớp thuộc chuyên môn này và lớp đó chưa kết thúc -> CẤM GỠ

IF EXISTS (

SELECT 1

FROM LopHoc

WHERE MaNV = @MaNV

AND MaCM = @MaCM

AND NgayKetThuc >= CAST(GETDATE() AS DATE)

)

BEGIN

DECLARE @TenMon2 NVARCHAR(50);

SELECT @TenMon2 = TenChuyenMon FROM ChuyenMon WHERE MaCM = @MaCM;

RAISERROR (N'Nhân viên này đang là Giáo viên chủ nhiệm của lớp thuộc chuyên môn "%s". Vui lòng đổi giáo viên chủ nhiệm trước khi gỡ chuyên môn.', 16, 1, @TenMon2);

RETURN;

END

-- 3. Nếu thỏa mãn hết -> Xóa

DELETE FROM NhanVienChuyenMon

WHERE MaNV = @MaNV AND MaCM = @MaCM;

PRINT N'Đã gỡ bỏ chuyên môn khỏi nhân viên thành công.';

END

1. sp\_SuaChuyenMon

Tác dụng:

Procedure này dùng để sửa thông tin chuyên môn trong hệ thống. Nó đảm bảo:

* Kiểm tra sự tồn tại của chuyên môn trước khi sửa.
* Ngăn chặn việc trùng tên chuyên môn với bản ghi khác.
* Cập nhật tên và mô tả chuyên môn trong bảng ChuyenMon.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
2. Kiểm tra sự tồn tại của chuyên môn
   * Truy vấn bảng ChuyenMon để kiểm tra xem @MaCM có tồn tại hay không.
   * Nếu không tồn tại → báo lỗi *"Chuyên môn không tồn tại."* và kết thúc procedure.
3. Kiểm tra trùng tên chuyên môn
   * Truy vấn bảng ChuyenMon để kiểm tra xem @TenChuyenMon đã được sử dụng bởi chuyên môn khác (khác @MaCM) hay chưa.
   * Nếu trùng → báo lỗi *"Tên chuyên môn này đã được sử dụng bởi mục khác."* và kết thúc procedure.
4. Cập nhật chuyên môn
   * Nếu vượt qua các kiểm tra trên → thực hiện UPDATE bảng ChuyenMon với các giá trị mới:
     + TenChuyenMon = @TenChuyenMon
     + MoTa = @MoTa
5. Thông báo kết quả
   * In ra thông báo *"Cập nhật chuyên môn thành công."*.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_SuaChuyenMon]

@MaCM INT,

@TenChuyenMon NVARCHAR(50),

@MoTa NVARCHAR(500) = NULL

AS

BEGIN

SET NOCOUNT ON;

-- Kiểm tra tồn tại

IF NOT EXISTS (SELECT 1 FROM ChuyenMon WHERE MaCM = @MaCM)

BEGIN

RAISERROR (N'Chuyên môn không tồn tại.', 16, 1);

RETURN;

END

-- Kiểm tra trùng tên (nhưng bỏ qua chính bản ghi đang sửa)

IF EXISTS (SELECT 1 FROM ChuyenMon WHERE TenChuyenMon = @TenChuyenMon AND MaCM <> @MaCM)

BEGIN

RAISERROR (N'Tên chuyên môn này đã được sử dụng bởi mục khác.', 16, 1);

RETURN;

END

UPDATE ChuyenMon

SET TenChuyenMon = @TenChuyenMon,

MoTa = @MoTa

WHERE MaCM = @MaCM;

PRINT N'Cập nhật chuyên môn thành công.';

END

1. sp\_GanChuyenMonChoNhanVien

Tác dụng:

Procedure này dùng để gán một chuyên môn cho nhân viên trong hệ thống. Nó đảm bảo rằng chỉ những nhân viên có chức vụ phù hợp (PT hoặc HLV Lớp) mới được gán chuyên môn, đồng thời tránh việc gán trùng lặp chuyên môn đã có.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra chức vụ nhân viên
   * Gọi hàm dbo.f\_KiemTraNhanVienCoChucVu để kiểm tra xem nhân viên (@MaNV) có chức vụ PT hoặc HLV Lớp hay không.
   * Nếu không thuộc hai chức vụ này → báo lỗi *"Nhân viên này không phải là PT hoặc HLV Lớp, không thể gán chuyên môn."* và kết thúc procedure.
3. Kiểm tra trùng lặp chuyên môn
   * Truy vấn bảng NhanVienChuyenMon để kiểm tra xem nhân viên đã có chuyên môn (@MaCM) này chưa.
   * Nếu đã tồn tại → báo lỗi *"Nhân viên này đã có chuyên môn này rồi."* và kết thúc procedure.
4. Gán chuyên môn cho nhân viên
   * Nếu vượt qua các kiểm tra trên → thực hiện INSERT vào bảng NhanVienChuyenMon với các giá trị:
     + Mã nhân viên (MaNV)
     + Mã chuyên môn (MaCM)
5. Thông báo kết quả
   * In ra thông báo *"Gán chuyên môn cho nhân viên thành công."*.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_GanChuyenMonChoNhanVien]

@MaNV INT,

@MaCM INT

AS

BEGIN

SET NOCOUNT ON;

-- (BỔ SUNG) Kiểm tra logic: Chỉ gán chuyên môn cho PT hoặc HLV Lớp

IF (dbo.f\_KiemTraNhanVienCoChucVu(@MaNV, N'PT') = 0 AND dbo.f\_KiemTraNhanVienCoChucVu(@MaNV, N'HLV Lớp') = 0)

BEGIN

RAISERROR (N'Nhân viên này không phải là PT hoặc HLV Lớp, không thể gán chuyên môn.', 16, 1);

RETURN;

END

IF EXISTS (SELECT 1 FROM NhanVienChuyenMon WHERE MaNV = @MaNV AND MaCM = @MaCM)

BEGIN

RAISERROR (N'Nhân viên này đã có chuyên môn này rồi.', 16, 1);

RETURN;

END

INSERT INTO NhanVienChuyenMon (MaNV, MaCM) VALUES (@MaNV, @MaCM);

PRINT N'Gán chuyên môn cho nhân viên thành công.';

END

### Nhóm Quản lý Sản phẩm & Loại sản phẩm

1. sp\_ThemSanPham

Tác dụng:

Procedure này dùng để thêm mới một sản phẩm vào hệ thống. Nó cho phép nhập đầy đủ thông tin sản phẩm, bao gồm giá gốc, giá khuyến mãi, hãng sản xuất, xuất xứ, bảo hành, mô tả chung và mô tả chi tiết. Đồng thời, nó có kiểm tra logic để đảm bảo dữ liệu hợp lệ (ví dụ: giá khuyến mãi không được lớn hơn giá gốc).

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra logic giá khuyến mãi
   * Nếu @GiaKhuyenMai khác NULL và lớn hơn @DonGia → báo lỗi *"Giá khuyến mãi không được lớn hơn giá gốc."* và kết thúc procedure.
3. Thêm sản phẩm mới
   * Nếu vượt qua kiểm tra trên → thực hiện INSERT vào bảng SanPham với các thông tin:
     + Tên sản phẩm (TenSP)
     + Mã loại sản phẩm (MaLoaiSP)
     + Đơn giá (DonGia)
     + Số lượng tồn (SoLuongTon)
     + Giá khuyến mãi (GiaKhuyenMai) nếu có
     + Hãng (Hang) nếu có
     + Xuất xứ (XuatXu) nếu có
     + Bảo hành (BaoHanh) nếu có
     + Mô tả chung (MoTaChung) nếu có
     + Mô tả chi tiết (MoTaChiTiet) nếu có
4. Thông báo kết quả
   * In ra thông báo *"Thêm sản phẩm mới thành công."*.
5. Trả về mã sản phẩm mới
   * Sử dụng SCOPE\_IDENTITY() để trả về MaSPMoi (mã sản phẩm vừa được tạo).
   * Điều này giúp ứng dụng có thể sử dụng ngay mã sản phẩm mới cho các thao tác tiếp theo (ví dụ: thêm vào giỏ hàng, quản lý tồn kho).

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemSanPham]

    @TenSP NVARCHAR(100),

    @MaLoaiSP INT,

    @DonGia DECIMAL(12,2),

    @SoLuongTon INT,

    -- Các tham số mới (để giá trị mặc định là NULL)

    @GiaKhuyenMai DECIMAL(12,2) = NULL,

    @Hang NVARCHAR(100) = NULL,

    @XuatXu NVARCHAR(100) = NULL,

    @BaoHanh NVARCHAR(50) = NULL,

    @MoTaChung NVARCHAR(1000) = NULL,

    @MoTaChiTiet NVARCHAR(MAX) = NULL

AS

BEGIN

    SET NOCOUNT ON;

    -- (Tùy chọn) Kiểm tra logic: Giá khuyến mãi không được lớn hơn giá gốc

    IF @GiaKhuyenMai IS NOT NULL AND @GiaKhuyenMai > @DonGia

    BEGIN

        RAISERROR(N'Giá khuyến mãi không được lớn hơn giá gốc.', 16, 1);

        RETURN;

    END

    INSERT INTO SanPham (

        TenSP, MaLoaiSP, DonGia, SoLuongTon,

        GiaKhuyenMai, Hang, XuatXu, BaoHanh, MoTaChung, MoTaChiTiet

    )

    VALUES (

        @TenSP, @MaLoaiSP, @DonGia, @SoLuongTon,

        @GiaKhuyenMai, @Hang, @XuatXu, @BaoHanh, @MoTaChung, @MoTaChiTiet

    );

    PRINT N'Thêm sản phẩm mới thành công.';

    -- Trả về MaSP vừa được tạo để ứng dụng có thể dùng

    SELECT SCOPE\_IDENTITY() AS MaSPMoi;

END

1. sp\_ThemLoaiSanPham

Tác dụng:

Procedure này dùng để thêm mới một loại sản phẩm vào hệ thống. Nó đảm bảo:

* Kiểm tra trùng lặp tên loại sản phẩm trước khi thêm.
* Nếu hợp lệ, thêm loại sản phẩm mới vào bảng LoaiSanPham.
* Trả về mã loại sản phẩm vừa được tạo để ứng dụng có thể sử dụng ngay.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng.
2. Kiểm tra trùng tên loại sản phẩm
   * Truy vấn bảng LoaiSanPham để kiểm tra xem @TenLoaiSP đã tồn tại hay chưa.
   * Nếu tồn tại → báo lỗi *"Tên loại sản phẩm đã tồn tại!"* và kết thúc procedure.
3. Thêm loại sản phẩm mới
   * Nếu tên loại sản phẩm chưa tồn tại → thực hiện INSERT vào bảng LoaiSanPham với giá trị @TenLoaiSP.
4. Thông báo kết quả
   * In ra thông báo *"Thêm loại sản phẩm thành công."*.
5. Trả về mã loại sản phẩm mới
   * Sử dụng SCOPE\_IDENTITY() để trả về MaLoaiMoi (mã loại sản phẩm vừa được tạo).
   * Điều này giúp ứng dụng có thể sử dụng ngay mã loại sản phẩm mới cho các thao tác tiếp theo (ví dụ: thêm sản phẩm thuộc loại này).

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_ThemLoaiSanPham]

@TenLoaiSP NVARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM LoaiSanPham WHERE TenLoaiSP = @TenLoaiSP)

BEGIN

RAISERROR(N'Tên loại sản phẩm đã tồn tại!', 16, 1);

RETURN;

END

INSERT INTO LoaiSanPham (TenLoaiSP)

VALUES (@TenLoaiSP);

PRINT N'Thêm loại sản phẩm thành công.';

SELECT SCOPE\_IDENTITY() AS MaLoaiMoi; -- Trả về mã mới

END

1. sp\_XoaSanPham

Tác dụng:

Procedure này dùng để xóa một sản phẩm khỏi hệ thống dựa trên mã sản phẩm (@MaSP). Nó đồng thời trả về số dòng bị ảnh hưởng để ứng dụng (Form) có thể biết thao tác xóa có thành công hay không.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng ngoài ý muốn.
2. Xóa sản phẩm
   * Thực hiện DELETE từ bảng SanPham với điều kiện MaSP = @MaSP.
   * Nếu sản phẩm tồn tại → bản ghi sẽ bị xóa.
   * Nếu sản phẩm không tồn tại → không có bản ghi nào bị xóa.
3. Trả về kết quả
   * Sử dụng RETURN @@ROWCOUNT để trả về số dòng bị ảnh hưởng:
     + 1 → xóa thành công một sản phẩm.
     + 0 → không có sản phẩm nào bị xóa (có thể do mã sản phẩm không tồn tại).

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_XoaSanPham]

@MaSP INT

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM SanPham

WHERE MaSP = @MaSP;

-- Trả về số dòng bị ảnh hưởng để Form biết có xóa thành công hay không

RETURN @@ROWCOUNT;

END

1. sp\_XoaLoaiSanPham

Tác dụng:

Procedure này dùng để xóa một loại sản phẩm khỏi hệ thống dựa trên mã loại sản phẩm (@MaLoaiSP). Nó đồng thời trả về mã kết quả để ứng dụng biết thao tác có thành công hay không, hoặc lý do thất bại.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
   * Khai báo biến @KetQua để lưu kết quả xử lý.
2. Kiểm tra sự tồn tại của loại sản phẩm
   * Nếu @MaLoaiSP không tồn tại trong bảng LoaiSanPham → gán @KetQua = -1 và trả về.
   * Ý nghĩa: *Loại sản phẩm không tồn tại*.
3. Kiểm tra ràng buộc sản phẩm con
   * Nếu có sản phẩm (SanPham) thuộc loại này → gán @KetQua = -2 và trả về.
   * Ý nghĩa: *Không thể xóa vì loại sản phẩm này vẫn còn sản phẩm con*.
4. Xóa loại sản phẩm
   * Nếu vượt qua các kiểm tra trên → thực hiện DELETE từ bảng LoaiSanPham với MaLoaiSP tương ứng.
   * Gán @KetQua = @@ROWCOUNT để biết số dòng bị ảnh hưởng (thường là 1 nếu xóa thành công).
5. Trả về kết quả
   * Trả về giá trị @KetQua để ứng dụng có thể xử lý:
     + 1 → xóa thành công.
     + 0 → không có bản ghi nào bị xóa.
     + -1 → loại sản phẩm không tồn tại.
     + -2 → loại sản phẩm còn sản phẩm con, không thể xóa.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_XoaLoaiSanPham]

@MaLoaiSP INT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @KetQua INT = 0;

-- Kiểm tra loại có tồn tại không

IF NOT EXISTS (SELECT 1 FROM LoaiSanPham WHERE MaLoaiSP = @MaLoaiSP)

BEGIN

SET @KetQua = -1; -- không tồn tại

SELECT @KetQua AS KetQua;

RETURN;

END

-- Kiểm tra có sản phẩm thuộc loại này không

IF EXISTS (SELECT 1 FROM SanPham WHERE MaLoaiSP = @MaLoaiSP)

BEGIN

SET @KetQua = -2; -- còn sản phẩm con

SELECT @KetQua AS KetQua;

RETURN;

END

-- Thực hiện xóa

DELETE FROM LoaiSanPham WHERE MaLoaiSP = @MaLoaiSP;

SET @KetQua = @@ROWCOUNT; -- 1 nếu xóa thành công

SELECT @KetQua AS KetQua;

END

1. sp\_SuaSanPham

Tác dụng:

Procedure này dùng để sửa thông tin sản phẩm trong hệ thống. Nó đảm bảo:

* Kiểm tra sự tồn tại của sản phẩm trước khi sửa.
* Cập nhật đầy đủ các thông tin liên quan đến sản phẩm (tên, loại, giá, số lượng, khuyến mãi, hãng, xuất xứ, bảo hành, mô tả).
* Trả về số dòng được cập nhật để ứng dụng biết thao tác có thành công hay không.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
2. Kiểm tra sự tồn tại của sản phẩm
   * Truy vấn bảng SanPham để kiểm tra xem @MaSP có tồn tại hay không.
   * Nếu không tồn tại → báo lỗi *"Sản phẩm không tồn tại."* và kết thúc procedure.
3. Cập nhật thông tin sản phẩm
   * Nếu sản phẩm tồn tại → thực hiện UPDATE bảng SanPham với các giá trị mới:
     + Tên sản phẩm (TenSP)
     + Mã loại sản phẩm (MaLoaiSP)
     + Đơn giá (DonGia)
     + Số lượng tồn (SoLuongTon)
     + Giá khuyến mãi (GiaKhuyenMai) nếu có
     + Hãng (Hang) nếu có
     + Xuất xứ (XuatXu) nếu có
     + Bảo hành (BaoHanh) nếu có
     + Mô tả chung (MoTaChung) nếu có
     + Mô tả chi tiết (MoTaChiTiet) nếu có
4. Trả về kết quả
   * Sử dụng SELECT @@ROWCOUNT AS SoDongCapNhat để trả về số dòng được cập nhật:
     + 1 → cập nhật thành công một sản phẩm.
     + 0 → không có bản ghi nào được cập nhật (có thể do dữ liệu không thay đổi).

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_SuaSanPham]

@MaSP INT,

@TenSP NVARCHAR(100),

@MaLoaiSP INT,

@DonGia DECIMAL(12, 2),

@SoLuongTon INT,

@GiaKhuyenMai DECIMAL(12, 2) = NULL,

@Hang NVARCHAR(100) = NULL,

@XuatXu NVARCHAR(100) = NULL,

@BaoHanh NVARCHAR(50) = NULL,

@MoTaChung NVARCHAR(1000) = NULL,

@MoTaChiTiet NVARCHAR(MAX) = NULL

AS

BEGIN

SET NOCOUNT ON;

IF NOT EXISTS (SELECT 1 FROM SanPham WHERE MaSP = @MaSP)

BEGIN

RAISERROR(N'Sản phẩm không tồn tại.', 16, 1);

RETURN;

END

UPDATE SanPham

SET

TenSP = @TenSP,

MaLoaiSP = @MaLoaiSP,

DonGia = @DonGia,

SoLuongTon = @SoLuongTon,

GiaKhuyenMai = @GiaKhuyenMai,

Hang = @Hang,

XuatXu = @XuatXu,

BaoHanh = @BaoHanh,

MoTaChung = @MoTaChung,

MoTaChiTiet = @MoTaChiTiet

WHERE MaSP = @MaSP;

--Trả về số dòng được cập nhật

SELECT @@ROWCOUNT AS SoDongCapNhat;

END;

1. sp\_SuaLoaiSanPham

Tác dụng:

Procedure này dùng để sửa tên loại sản phẩm trong hệ thống. Nó đảm bảo:

* Kiểm tra sự tồn tại của loại sản phẩm trước khi sửa.
* Ngăn chặn việc sửa nếu tên mới trùng với tên cũ (không thay đổi).
* Cập nhật tên loại sản phẩm khi hợp lệ.
* Trả về mã kết quả để ứng dụng (C#) có thể xử lý và hiển thị thông báo phù hợp.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON).
   * Khai báo biến @KetQua để lưu kết quả xử lý.
2. Bắt đầu giao dịch (BEGIN TRANSACTION)
   * Đảm bảo mọi thao tác cập nhật được thực hiện đồng bộ.
3. Kiểm tra sự tồn tại của loại sản phẩm
   * Nếu @MaLoaiSP không tồn tại trong bảng LoaiSanPham → gán @KetQua = -1.
   * Ý nghĩa: *Loại sản phẩm không tồn tại*.
4. Kiểm tra trùng tên loại sản phẩm
   * Nếu tên mới (@TenLoaiSP) trùng với tên hiện tại của loại sản phẩm → gán @KetQua = 0.
   * Ý nghĩa: *Không có thay đổi*.
5. Cập nhật loại sản phẩm
   * Nếu vượt qua các kiểm tra trên → thực hiện UPDATE bảng LoaiSanPham với tên mới.
   * Gán @KetQua = @@ROWCOUNT để biết số dòng được cập nhật (thường là 1 nếu thành công).
6. Hoàn tất giao dịch
   * Nếu không có lỗi → COMMIT TRANSACTION.
7. Xử lý lỗi (BEGIN CATCH)
   * Nếu có lỗi trong quá trình thực thi → rollback giao dịch.
   * Gán @KetQua = -99 để báo lỗi ra ứng dụng.
8. Trả về kết quả
   * Trả về giá trị @KetQua để ứng dụng có thể xử lý:
     + 1 → cập nhật thành công.
     + 0 → tên mới trùng tên cũ, không thay đổi.
     + -1 → loại sản phẩm không tồn tại.
     + -99 → lỗi hệ thống trong quá trình thực thi.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_SuaLoaiSanPham]

@MaLoaiSP INT,

@TenLoaiSP NVARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @KetQua INT = 0;

BEGIN TRY

BEGIN TRANSACTION;

-- Không tồn tại

IF NOT EXISTS (SELECT 1 FROM LoaiSanPham WHERE MaLoaiSP = @MaLoaiSP)

BEGIN

SET @KetQua = -1;

END

-- Tên mới trùng với tên cũ → không thay đổi

ELSE IF EXISTS (

SELECT 1 FROM LoaiSanPham

WHERE MaLoaiSP = @MaLoaiSP AND TenLoaiSP = @TenLoaiSP

)

BEGIN

SET @KetQua = 0;

END

ELSE

BEGIN

UPDATE LoaiSanPham

SET TenLoaiSP = @TenLoaiSP

WHERE MaLoaiSP = @MaLoaiSP;

SET @KetQua = @@ROWCOUNT;

END

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

-- Trả lỗi ra C#

SET @KetQua = -99;

END CATCH;

SELECT @KetQua AS KetQua;

END

### Nhóm Quản lý Hình ảnh

1. sp\_HinhAnh\_Them

Tác dụng:

Procedure này dùng để thêm hình ảnh cho sản phẩm trong hệ thống. Nó cho phép lưu đường dẫn hình ảnh (Url) và đánh dấu hình chính (IsMain) nếu cần. Sau khi thêm, procedure trả về mã hình ảnh mới để ứng dụng có thể sử dụng ngay.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng ngoài ý muốn.
2. Thêm hình ảnh
   * Thực hiện INSERT vào bảng HINHANH với các thông tin:
     + Mã sản phẩm (MaSP)
     + Đường dẫn hình ảnh (Url)
     + Cờ đánh dấu hình chính (IsMain) nếu có
3. Trả về mã hình ảnh mới
   * Sử dụng SCOPE\_IDENTITY() để trả về MaHinhMoi (mã hình ảnh vừa được tạo).
   * Điều này giúp ứng dụng có thể sử dụng ngay mã hình ảnh mới cho các thao tác tiếp theo (ví dụ: hiển thị hình chính, quản lý gallery sản phẩm).

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_HinhAnh\_Them]

@MaSP INT,

@Url NVARCHAR(500),

@IsMain BIT = NULL

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO HINHANH (MaSP, Url, IsMain)

VALUES (@MaSP, @Url, @IsMain);

-- Trả về mã hình mới tạo (nếu app cần)

SELECT SCOPE\_IDENTITY() AS MaHinhMoi;

END;

1. sp\_HinhAnh\_Xoa

Tác dụng:

Procedure này dùng để xóa một hình ảnh sản phẩm trong hệ thống dựa trên mã hình (@MaHinh). Sau khi xóa, nó trả về ID của hình ảnh đã xóa để ứng dụng có thể xác nhận thao tác.

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng ngoài ý muốn.
2. Xóa hình ảnh
   * Thực hiện DELETE từ bảng HINHANH với điều kiện MaHinh = @MaHinh.
   * Nếu hình ảnh tồn tại → bản ghi sẽ bị xóa.
   * Nếu không tồn tại → không có bản ghi nào bị xóa.
3. Trả về kết quả
   * Trả về giá trị @MaHinh dưới tên MaHinhDaXoa.
   * Điều này giúp ứng dụng biết ID nào đã được yêu cầu xóa, đồng thời xác nhận thao tác.

Code trong SQL:

ALTER PROC [dbo].[sp\_HinhAnh\_Xoa]

@MaHinh INT

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM HINHANH

WHERE MaHinh = @MaHinh;

-- Trả về ID đã xóa (nếu xóa thành công)

SELECT @MaHinh AS MaHinhDaXoa;

END;

1. sp\_HinhAnh\_Sua

Tác dụng:

Procedure này dùng để cập nhật thông tin hình ảnh sản phẩm trong hệ thống. Nó cho phép thay đổi:

* Sản phẩm liên kết (MaSP)
* Đường dẫn hình ảnh (Url)
* Cờ đánh dấu hình chính (IsMain)

Các bước xử lý:

1. Khởi tạo
   * Tắt chế độ đếm số dòng (SET NOCOUNT ON) để tránh trả về số lượng bản ghi ảnh hưởng ngoài ý muốn.
2. Cập nhật hình ảnh
   * Thực hiện UPDATE bảng HINHANH với các giá trị mới:
     + MaSP = @MaSP
     + Url = @Url
     + IsMain = @IsMain
   * Điều kiện: MaHinh = @MaHinh (chỉ cập nhật đúng hình ảnh cần sửa).
3. Trả về kết quả
   * Trả về giá trị @MaHinh dưới tên UpdatedID.
   * Điều này giúp ứng dụng biết ID nào đã được sửa thành công.

Code trong SQL:

ALTER PROCEDURE [dbo].[sp\_HinhAnh\_Sua]

@MaHinh INT,

@MaSP INT,

@Url NVARCHAR(500),

@IsMain BIT

AS

BEGIN

SET NOCOUNT ON;

UPDATE HINHANH

SET MaSP = @MaSP,

Url = @Url,

IsMain = @IsMain

WHERE MaHinh = @MaHinh;

SELECT @MaHinh AS UpdatedID; -- ⭐ TRẢ VỀ GIÁ TRỊ

END;

## Cài đặt Trigger

### Kiểm tra dữ liệu & ràng buộc

1. trg\_CheckNgayBDKT

Tác dụng:

Trigger trg\_CheckNgayBDKT đảm bảo rằng trong bảng LopHoc:

* Ngày bắt đầu (NgayBatDau) không được lớn hơn ngày kết thúc (NgayKetThuc).
* Nếu vi phạm, trigger sẽ báo lỗi và rollback giao dịch, ngăn không cho dữ liệu sai được lưu.

Các bước xử lý:

1. Trigger chạy AFTER INSERT, UPDATE trên bảng LopHoc.
2. Sử dụng bảng ảo inserted để lấy dữ liệu mới được thêm/cập nhật.
3. Kiểm tra điều kiện: NgayBatDau > NgayKetThuc
   * Nếu có bản ghi vi phạm → thực hiện RAISERROR và ROLLBACK TRANSACTION.
4. Nếu không vi phạm → dữ liệu được lưu bình thường.

Code trong SQL:

create or alter trigger trg\_CheckNgayBDKT

on LopHoc

AFTER insert, update

AS

BEGIN

IF EXISTS(

select \* from inserted

where NgayBatDau > NgayKetThuc)

BEGIN

raiserror (N'Thời gian bắt đầu không được phép nhỏ hơn kêt thúc', 16, 1);

rollback transaction;

END

END;

1. trg\_KiemTraNhanVienChuyenMon

Tác dụng:

Trigger trg\_KiemTraNhanVienChuyenMon đảm bảo rằng:

* Khi thêm hoặc cập nhật lớp học (LopHoc), nhân viên (MaNV) được gán cho lớp phải có chuyên môn (MaCM) phù hợp.
* Nếu nhân viên không có chuyên môn đó trong bảng NhanVienChuyenMon, trigger sẽ báo lỗi và rollback giao dịch.

Các bước xử lý:

1. Trigger chạy AFTER INSERT, UPDATE trên bảng LopHoc.
2. Sử dụng bảng ảo inserted để lấy dữ liệu mới được thêm/cập nhật.
3. Kiểm tra điều kiện:
   * Nhân viên có chuyên môn đó trong bảng NhanVienChuyenMon hay không
   * Nếu có bản ghi vi phạm (nhân viên không có chuyên môn lớp này) → báo lỗi và rollback.
4. Nếu không vi phạm → dữ liệu được lưu bình thường.

Code trong SQL:

create or alter trigger trg\_KiemTraNhanVienChuyenMon

on LopHoc

after insert, update

as

begin

if exists(

select \* from inserted i

where i.MaCM not in (

select MaCM from NhanVienChuyenMon cm

where i.MaNV = cm.MaNV)

)

BEGIN

raiserror (N'Nhân viên không có chuyên môn lớp này', 16, 1);

rollback transaction;

END

end;

1. trg\_KiemTraTrungLichNhanVien

Tác dụng:

Trigger trg\_KiemTraTrungLichNhanVien đảm bảo rằng:

* Khi thêm hoặc cập nhật lịch lớp (LichLop), nhân viên (MaNV) không bị trùng lịch với:
  1. Một lớp khác trong cùng ngày (LichLop).
  2. Lịch tập cá nhân (LichTapPT) đã đăng ký.

Nếu phát hiện trùng lịch, trigger sẽ báo lỗi và rollback giao dịch.

Các bước xử lý:

1. Trigger chạy AFTER INSERT, UPDATE trên bảng LichLop.
2. Sử dụng bảng ảo inserted để lấy dữ liệu mới được thêm/cập nhật.
3. Kiểm tra điều kiện trùng lịch:
   * Trong bảng LichLop:
     + Cùng nhân viên (MaNV).
     + Cùng ngày (NgayHoc).
     + Khác mã lịch lớp (MaLichLop).
     + Thời gian bị giao nhau.
   * Trong bảng LichTapPT (qua DangKyPT):
     + Cùng nhân viên (MaNV).
     + Ngày tập trùng với ngày học (NgayTap = NgayHoc).
     + Thời gian bị giao nhau.
4. Nếu có bản ghi vi phạm → RAISERROR và ROLLBACK TRANSACTION.

Code trong SQL:

CREATE or alter TRIGGER trg\_KiemTraTrungLichNhanVien

ON LichLop

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (

SELECT 1

FROM inserted i

WHERE EXISTS (

SELECT 1

FROM LichLop l

WHERE l.MaNV = i.MaNV

AND l.NgayHoc = i.NgayHoc

AND l.MaLichLop <> i.MaLichLop

AND l.GioBatDau < i.GioKetThuc

AND l.GioKetThuc > i.GioBatDau

)

OR EXISTS (

SELECT 1

FROM LichTapPT L

JOIN DangKyPT D ON L.MaDKPT = D.MaDKPT

WHERE D.MaNV = i.MaNV

AND L.NgayTap = i.NgayHoc

AND L.GioBatDau < i.GioKetThuc

AND L.GioKetThuc > i.GioBatDau

)

)

BEGIN

RAISERROR(N'Nhân viên bị trùng lịch!', 16, 1);

ROLLBACK TRANSACTION;

END

END;

1. trg\_NhanVien\_KiemTraTuoi

Tác dụng:

Ngăn chặn việc thêm hoặc cập nhật nhân viên có tuổi dưới 18 vào hệ thống.

Nếu phát hiện vi phạm, trigger sẽ:

* Báo lỗi bằng RAISERROR để hiển thị thông báo cho người dùng.
* Hủy bỏ giao dịch (ROLLBACK TRANSACTION) để không lưu dữ liệu sai.

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy sau khi có thao tác INSERT hoặc UPDATE trên bảng NhanVien.
2. Kiểm tra cột NgaySinh
   * Dùng IF UPDATE(NgaySinh) để chỉ thực hiện kiểm tra khi cột NgaySinh có thay đổi hoặc khi thêm mới nhân viên.
3. Xác định vi phạm
   * Từ bảng ảo inserted, lấy các bản ghi vừa thêm/cập nhật.
   * Dùng DATEDIFF(YEAR, NgaySinh, GETDATE()) để tính tuổi.
   * Nếu tuổi < 18 → coi là vi phạm.
4. Xử lý vi phạm
   * Nếu có vi phạm:
     + Gửi thông báo lỗi (RAISERROR) với mức độ nghiêm trọng 16.
     + Hủy bỏ giao dịch (ROLLBACK TRANSACTION) để ngăn dữ liệu sai được lưu.
5. Kết thúc trigger
   * Nếu không có vi phạm, dữ liệu được thêm/cập nhật sẽ được lưu bình thường.

Code trong SQL:

create TRIGGER [dbo].[trg\_NhanVien\_KiemTraTuoi]

ON [dbo].[NhanVien]

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

-- Chỉ kiểm tra nếu cột NgaySinh có sự thay đổi (hoặc thêm mới)

IF UPDATE(NgaySinh)

BEGIN

-- Kiểm tra xem có dòng nào vi phạm không

-- Logic: Năm hiện tại - Năm sinh < 18

IF EXISTS (

SELECT 1

FROM inserted

WHERE NgaySinh IS NOT NULL

-- DATEDIFF trả về số năm chênh lệch

AND DATEDIFF(YEAR, NgaySinh, GETDATE()) < 18

)

BEGIN

-- Báo lỗi đỏ rực

RAISERROR (N'Lỗi nghiệp vụ: Nhân viên chưa đủ 18 tuổi. Không thể lưu vào hệ thống.', 16, 1);

-- Hoàn tác, hủy bỏ lệnh vừa thực hiện

ROLLBACK TRANSACTION;

END

END

END;

1. trg\_NhanVienChuyenMon\_KiemTraChucVu

Tác dụng:

Đảm bảo chỉ những nhân viên có chức vụ phù hợp (PT hoặc HLV Lớp) mới được phép gán chuyên môn trong bảng NhanVienChuyenMon.

Nếu người dùng cố tình thêm hoặc cập nhật chuyên môn cho nhân viên có chức vụ khác (ví dụ: Lễ tân, Quản lý), trigger sẽ:

* Phát sinh lỗi (RAISERROR) để báo cho người dùng.
* Hủy bỏ giao dịch (ROLLBACK TRANSACTION) để ngăn dữ liệu sai nghiệp vụ được lưu vào hệ thống.

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy sau khi có thao tác INSERT hoặc UPDATE trên bảng NhanVienChuyenMon.
2. Kiểm tra dữ liệu vừa thêm/cập nhật
   * Dùng bảng ảo inserted để lấy các bản ghi mới.
   * Kết nối (JOIN) với bảng NhanVien để biết chức vụ của nhân viên (MaChucVu).
   * Kết nối tiếp với bảng ChucVu để lấy tên chức vụ (TenChucVu).
3. Xác định vi phạm
   * Nếu TenChucVu không nằm trong danh sách cho phép (PT, HLV Lớp), thì coi là vi phạm nghiệp vụ.
4. Xử lý vi phạm
   * Nếu phát hiện vi phạm:
     + Gửi thông báo lỗi đỏ (RAISERROR) với mức độ nghiêm trọng (severity 16).
     + Hủy bỏ toàn bộ giao dịch (ROLLBACK TRANSACTION) để ngăn dữ liệu sai được ghi vào.
5. Kết thúc trigger
   * Nếu không có vi phạm, dữ liệu được thêm/cập nhật sẽ được lưu bình thường.

Code trong SQL:

CREATE TRIGGER [dbo].[trg\_NhanVienChuyenMon\_KiemTraChucVu]

ON [dbo].[NhanVienChuyenMon]

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

-- Kiểm tra xem có dòng nào vừa thêm vào mà vi phạm luật không

IF EXISTS (

SELECT 1

FROM inserted i

JOIN NhanVien nv ON i.MaNV = nv.MaNV

JOIN ChucVu cv ON nv.MaChucVu = cv.MaChucVu

WHERE cv.TenChucVu NOT IN (N'PT', N'HLV Lớp') -- Những chức vụ ĐƯỢC PHÉP

)

BEGIN

-- Nếu phát hiện vi phạm -> Báo lỗi đỏ

RAISERROR (N'Lỗi nghiệp vụ: Chỉ có nhân viên chức vụ "PT" hoặc "HLV Lớp" mới được phép gán chuyên môn. Các chức vụ khác (Lễ tân, Quản lý...) không cần chuyên môn.', 16, 1);

-- Hủy bỏ lệnh vừa thực hiện

ROLLBACK TRANSACTION;

END

END

### Quản lý & cập nhật dữ liệu

1. trg\_TuDienSoBuoi

Tác dụng:

Trigger trg\_TuDienSoBuoi đảm bảo rằng:

* Mỗi khi có thay đổi trong bảng LichLop (thêm, sửa, xóa), hệ thống sẽ tự động cập nhật lại số buổi (SoBuoi) của lớp học (LopHoc).
* Nhờ đó, dữ liệu luôn nhất quán, không cần tính toán thủ công.

Các bước xử lý:

1. Trigger chạy AFTER INSERT, UPDATE, DELETE trên bảng LichLop.
2. Xác định các lớp bị ảnh hưởng thông qua bảng ảo inserted và deleted.
   * inserted chứa bản ghi mới (sau khi insert/update).
   * deleted chứa bản ghi cũ (trước khi update/delete).
   * Dùng UNION để lấy tất cả MaLop liên quan.
3. Thực hiện UPDATE bảng LopHoc:
   * Với mỗi lớp bị ảnh hưởng, tính lại số buổi bằng COUNT(\*) từ bảng LichLop.
   * Gán giá trị này vào cột SoBuoi của lớp.

Code trong SQL:

create or alter trigger trg\_TuDienSoBuoi

on LichLop

after insert, update, delete

as

begin

update LopHoc

set SoBuoi = (select Count(\*)

from LichLop

where LichLop.MaLop = LopHoc.MaLop)

where MaLop in (select MaLop from inserted

union

select MaLop from deleted);

end;

1. trg\_TuDienSiSo

Tác dụng:

Trigger trg\_TuDienSiSo đảm bảo rằng:

* Mỗi khi có thay đổi trong bảng DangKyLop (thêm, sửa, xóa đăng ký), hệ thống sẽ tự động cập nhật lại sĩ số hiện tại (SiSoHienTai) của lớp học (LopHoc).
* Nhờ đó, dữ liệu luôn nhất quán, không cần tính toán thủ công.

Các bước xử lý:

1. Trigger chạy AFTER INSERT, UPDATE, DELETE trên bảng DangKyLop.
2. Xác định các lớp bị ảnh hưởng thông qua bảng ảo inserted và deleted.
   * inserted chứa bản ghi mới (sau khi insert/update).
   * deleted chứa bản ghi cũ (trước khi update/delete).
   * Dùng UNION để lấy tất cả MaLop liên quan.
3. Thực hiện UPDATE bảng LopHoc:
   * Với mỗi lớp bị ảnh hưởng, tính lại sĩ số bằng COUNT(\*) từ bảng DangKyLop.
   * Gán giá trị này vào cột SiSoHienTai của lớp.

Code trong SQL:

create or alter trigger trg\_TuDienSiSo

on DangKyLop

after insert, update, delete

as begin

update LopHoc

set SiSoHienTai = (Select Count(\*)

from DangKyLop dk

where dk.MaLop = LopHoc.MaLop)

where MaLop in (select MaLop from inserted

union

select MaLop from deleted)

end;

1. trg\_ThemMaNV

Tác dụng:

Trigger trg\_ThemMaNV đảm bảo rằng:

* Khi thêm hoặc cập nhật bản ghi trong bảng LichLop, nếu chưa có MaNV thì hệ thống sẽ tự động gán MaNV từ lớp học (LopHoc) tương ứng.
* Nhờ đó, dữ liệu giữa LichLop và LopHoc luôn đồng bộ.

Các bước xử lý:

1. Trigger chạy AFTER INSERT, UPDATE trên bảng LichLop.
2. Sử dụng bảng ảo inserted để lấy dữ liệu mới được thêm/cập nhật.
3. Thực hiện UPDATE trên bảng LichLop (ll):
   * Gán MaNV = lh.MaNV từ bảng LopHoc.
   * Điều kiện: ll.MaLichLop = i.MaLichLop (chỉ cập nhật bản ghi vừa được thêm/cập nhật).
   * Chỉ cập nhật khi ll.MaNV IS NULL.

Code trong SQL:

create or alter trigger trg\_ThemMaNV

on LichLop

after insert, update

as

begin

UPDATE ll

SET MaNV = lh.MaNV

FROM LichLop ll

JOIN LopHoc lh ON ll.MaLop = lh.MaLop

JOIN inserted i ON ll.MaLichLop = i.MaLichLop

WHERE ll.MaNV IS NULL;

end;

1. trg\_CapNhatSoLuongTon

Tác dụng:

Khi một bản ghi mới được thêm vào bảng ChiTietHoaDon (ví dụ khách hàng mua sản phẩm), trigger sẽ chạy ngay sau đó (AFTER INSERT).

Trigger sẽ giảm số lượng tồn (SoLuongTon) của sản phẩm trong bảng SanPham theo số lượng đã bán (SoLuong trong ChiTietHoaDon).

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy sau khi có thao tác INSERT trên bảng ChiTietHoaDon.
2. Lấy dữ liệu từ bảng ảo inserted
   * Bảng inserted chứa các bản ghi vừa được thêm vào ChiTietHoaDon.
   * Mỗi bản ghi có MaSP (mã sản phẩm) và SoLuong (số lượng bán).
3. Cập nhật số lượng tồn kho
   * Dùng UPDATE trên bảng SanPham:
     + Tìm sản phẩm có MaSP trùng với bản ghi trong inserted.
     + Giảm SoLuongTon bằng cách trừ đi số lượng bán (SoLuong).
4. Điều kiện kiểm tra
   * WHERE i.MaSP IS NOT NULL đảm bảo chỉ xử lý khi có mã sản phẩm hợp lệ.
5. Kết thúc trigger
   * Sau khi chạy, số lượng tồn kho của sản phẩm được cập nhật chính xác.

Code trong SQL:

CREATE TRIGGER [dbo].[trg\_CapNhatSoLuongTon]

ON [dbo].[ChiTietHoaDon]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

UPDATE S

SET S.SoLuongTon = S.SoLuongTon - i.SoLuong

FROM SanPham S

JOIN inserted i ON S.MaSP = i.MaSP

WHERE i.MaSP IS NOT NULL;

END

1. trg\_CapNhatLoaiKhachHangHoaDon

Tác dụng:

Khi một hóa đơn mới được thêm vào bảng HoaDon, trigger sẽ chạy ngay sau đó (AFTER INSERT).

Trigger tính tổng chi tiêu của khách hàng dựa trên tất cả hóa đơn đã thanh toán.

Dựa vào tổng chi tiêu, hệ thống sẽ tự động phân loại khách hàng:

* Gym VIP: tổng chi tiêu ≥ 5,000,000.
* Gym Member: tổng chi tiêu ≥ 1,000,000.
* Giữ nguyên loại hiện tại nếu chưa đạt ngưỡng.

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy sau khi có thao tác INSERT trên bảng HoaDon.
2. Lấy mã loại khách hàng chuẩn
   * Truy vấn bảng LoaiKhachHang để lấy MaLoaiKH của hai loại:
     + Gym Member
     + Gym VIP.
3. Tính tổng chi tiêu của khách hàng
   * Từ bảng HoaDon, tính SUM(ThanhTien) cho từng khách hàng (MaKH) với điều kiện TrangThai = 'Đã thanh toán'.
4. Cập nhật loại khách hàng
   * Dùng UPDATE trên bảng KhachHang:
     + Nếu tổng chi tiêu ≥ 5,000,000 → gán MaLoaiKH = Gym VIP.
     + Nếu tổng chi tiêu ≥ 1,000,000 → gán MaLoaiKH = Gym Member.
     + Ngược lại giữ nguyên loại hiện tại.
   * Chỉ cập nhật khi loại khách hàng hiện tại khác với loại mới tính toán hoặc chưa có loại (IS NULL).
5. Kết thúc trigger
   * Sau khi chạy, loại khách hàng được cập nhật đúng theo tổng chi tiêu mới nhất.

Code trong SQL:

Create TRIGGER [dbo].[trg\_CapNhatLoaiKhachHangHoaDon]

ON [dbo].[HoaDon]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @MaLoaiMember INT, @MaLoaiVip INT;

SELECT @MaLoaiMember = MaLoaiKH FROM LoaiKhachHang WHERE TenLoai = N'Gym Member';

SELECT @MaLoaiVip = MaLoaiKH FROM LoaiKhachHang WHERE TenLoai = N'Gym VIP';

UPDATE kh

SET MaLoaiKH = CASE

WHEN TongChiTieu >= 5000000 THEN @MaLoaiVip

WHEN TongChiTieu >= 1000000 THEN @MaLoaiMember

ELSE MaLoaiKH

END

FROM KhachHang kh

INNER JOIN (

SELECT MaKH, SUM(ThanhTien) AS TongChiTieu

FROM HoaDon

WHERE TrangThai = N'Đã thanh toán'

GROUP BY MaKH

) hd ON kh.MaKH = hd.MaKH

WHERE kh.MaLoaiKH IS NULL OR kh.MaLoaiKH <>

CASE

WHEN hd.TongChiTieu >= 5000000 THEN @MaLoaiVip

WHEN hd.TongChiTieu >= 1000000 THEN @MaLoaiMember

ELSE kh.MaLoaiKH

END;

END

1. trg\_CapNhatLoaiKhachHang

Tác dụng:

Sau khi một hóa đơn mới được thêm (AFTER INSERT), trigger sẽ:

* Tính tổng chi tiêu của khách hàng dựa trên tất cả hóa đơn đã thanh toán.
* So sánh tổng chi tiêu với ngưỡng định sẵn:
  + ≥ 5,000,000 → cập nhật thành **Gym VIP**.
  + ≥ 1,000,000 → cập nhật thành **Gym Member**.
  + < 1,000,000 → giữ nguyên loại hiện tại.

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy sau khi có thao tác INSERT trên bảng HoaDon.
2. Lấy mã loại khách hàng chuẩn
   * Truy vấn bảng LoaiKhachHang để lấy MaLoaiKH của hai loại:
     + Gym Member
     + Gym VIP.
3. Tính tổng chi tiêu của khách hàng
   * Từ bảng HoaDon, tính SUM(ThanhTien) cho từng khách hàng (MaKH) với điều kiện TrangThai = 'Đã thanh toán'.
4. Cập nhật loại khách hàng
   * Dùng UPDATE trên bảng KhachHang:
     + Nếu tổng chi tiêu ≥ 5,000,000 → gán MaLoaiKH = Gym VIP.
     + Nếu tổng chi tiêu ≥ 1,000,000 → gán MaLoaiKH = Gym Member.
     + Ngược lại giữ nguyên loại hiện tại.
   * Chỉ cập nhật khi loại khách hàng hiện tại khác với loại mới tính toán hoặc chưa có loại (IS NULL).
5. Kết thúc trigger
   * Sau khi chạy, loại khách hàng được cập nhật đúng theo tổng chi tiêu mới nhất.

Code trong SQL:

Create TRIGGER [dbo].[trg\_CapNhatLoaiKhachHangHoaDon]

ON [dbo].[HoaDon]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @MaLoaiMember INT, @MaLoaiVip INT;

SELECT @MaLoaiMember = MaLoaiKH FROM LoaiKhachHang WHERE TenLoai = N'Gym Member';

SELECT @MaLoaiVip = MaLoaiKH FROM LoaiKhachHang WHERE TenLoai = N'Gym VIP';

UPDATE kh

SET MaLoaiKH = CASE

WHEN TongChiTieu >= 5000000 THEN @MaLoaiVip

WHEN TongChiTieu >= 1000000 THEN @MaLoaiMember

ELSE MaLoaiKH

END

FROM KhachHang kh

INNER JOIN (

SELECT MaKH, SUM(ThanhTien) AS TongChiTieu

FROM HoaDon

WHERE TrangThai = N'Đã thanh toán'

GROUP BY MaKH

) hd ON kh.MaKH = hd.MaKH

WHERE kh.MaLoaiKH IS NULL OR kh.MaLoaiKH <>

CASE

WHEN hd.TongChiTieu >= 5000000 THEN @MaLoaiVip

WHEN hd.TongChiTieu >= 1000000 THEN @MaLoaiMember

ELSE kh.MaLoaiKH

END;

END

1. trg\_XoaTuanTuTheoLoai

Tác dụng:

Khi người dùng thực hiện lệnh DELETE trên bảng LoaiSanPham, trigger này sẽ không xóa trực tiếp (vì dùng INSTEAD OF DELETE), mà thay vào đó:

* Tìm tất cả sản phẩm thuộc loại đó.
* Xóa toàn bộ hình ảnh liên quan đến từng sản phẩm.
* Xóa chính sản phẩm đó.
* Cuối cùng mới xóa loại sản phẩm.

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy khi có thao tác DELETE trên bảng LoaiSanPham.
2. Lấy danh sách loại sản phẩm cần xóa
   * Dùng bảng ảo deleted để lấy MaLoaiSP của các loại sản phẩm bị xóa.
   * Mở con trỏ curLoai để duyệt từng loại sản phẩm.
3. Xử lý từng loại sản phẩm
   * In thông báo: *"Đang xử lý loại sản phẩm: X"*.
4. Lấy danh sách sản phẩm thuộc loại đó
   * Mở con trỏ curSP để duyệt tất cả MaSP trong bảng SanPham có MaLoaiSP tương ứng.
5. Xóa dữ liệu liên quan đến từng sản phẩm
   * Xóa tất cả hình ảnh trong bảng HinhAnh có MaSP đó.
   * In thông báo: *"Đã xóa ảnh sản phẩm X"*.
   * Xóa sản phẩm trong bảng SanPham.
   * In thông báo: *"Đã xóa sản phẩm X"*.
6. Đóng con trỏ sản phẩm
   * Sau khi duyệt hết sản phẩm, đóng và giải phóng con trỏ curSP.
7. Xóa loại sản phẩm
   * Sau khi xóa hết sản phẩm con, thực hiện DELETE loại sản phẩm trong bảng LoaiSanPham.
   * In thông báo: *"Đã xóa loại sản phẩm X"*.
8. Tiếp tục với loại sản phẩm khác
   * Lặp lại cho đến khi duyệt hết các loại trong con trỏ curLoai.
9. Kết thúc trigger
   * Đóng và giải phóng con trỏ curLoai.
   * In thông báo: *"Hoàn tất xóa loại, sản phẩm và hình ảnh."*.

Code trong SQL:

CREATE TRIGGER [dbo].[trg\_XoaTuanTuTheoLoai]

ON [dbo].[LoaiSanPham]

INSTEAD OF DELETE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @MaLoaiSP INT;

DECLARE @MaSP INT;

DECLARE curLoai CURSOR FOR SELECT MaLoaiSP FROM deleted;

OPEN curLoai;

FETCH NEXT FROM curLoai INTO @MaLoaiSP;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT CONCAT(N'Đang xử lý loại sản phẩm: ', @MaLoaiSP);

DECLARE curSP CURSOR FOR SELECT MaSP FROM SanPham WHERE MaLoaiSP = @MaLoaiSP;

OPEN curSP;

FETCH NEXT FROM curSP INTO @MaSP;

WHILE @@FETCH\_STATUS = 0

BEGIN

DELETE FROM HinhAnh WHERE MaSP = @MaSP;

PRINT CONCAT(N'Đã xóa ảnh sản phẩm ', @MaSP);

DELETE FROM SanPham WHERE MaSP = @MaSP;

PRINT CONCAT(N'Đã xóa sản phẩm ', @MaSP);

FETCH NEXT FROM curSP INTO @MaSP;

END

CLOSE curSP; DEALLOCATE curSP;

-- Sau khi xóa hết sản phẩm con, xóa loại sản phẩm

DELETE FROM LoaiSanPham WHERE MaLoaiSP = @MaLoaiSP;

PRINT CONCAT(N'Đã xóa loại sản phẩm ', @MaLoaiSP);

FETCH NEXT FROM curLoai INTO @MaLoaiSP;

END

CLOSE curLoai; DEALLOCATE curLoai;

PRINT N'Hoàn tất xóa loại, sản phẩm và hình ảnh.';

END

1. trg\_OneMainImage

Tác dụng:

Đảm bảo mỗi sản phẩm (MaSP) chỉ có một hình ảnh chính (IsMain = 1) trong bảng HINHANH.

Khi thêm hoặc cập nhật hình ảnh:

* Nếu có hình ảnh mới được đánh dấu là chính (IsMain = 1), thì tất cả các hình ảnh khác của cùng sản phẩm sẽ tự động bỏ cờ chính (IsMain = 0).

Các bước xử lý:

1. Kích hoạt trigger
   * Trigger chạy sau khi có thao tác INSERT hoặc UPDATE trên bảng HINHANH.
2. Lấy thông tin sản phẩm
   * Từ bảng ảo inserted, trigger kiểm tra xem có bản ghi nào có IsMain = 1.
   * Nếu có, lấy MaSP của sản phẩm đó.
3. Kiểm tra điều kiện
   * Nếu @MaSP khác NULL (tức là có ảnh chính mới được thêm/cập nhật).
4. Cập nhật dữ liệu
   * Thực hiện UPDATE trên bảng HINHANH:
     + Tìm tất cả hình ảnh thuộc cùng sản phẩm (MaSP).
     + Loại trừ hình ảnh vừa được thêm/cập nhật (NOT IN (SELECT MaHinh FROM inserted)).
     + Đặt IsMain = 0 cho các hình ảnh đó.
5. Kết thúc trigger
   * Sau khi cập nhật, sản phẩm sẽ chỉ còn duy nhất một ảnh chính.

Code trong SQL:

CREATE TRIGGER [dbo].[trg\_OneMainImage]

ON [dbo].[HINHANH]

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @MaSP INT;

SELECT @MaSP = MaSP FROM inserted WHERE IsMain = 1;

IF (@MaSP IS NOT NULL)

BEGIN

UPDATE HINHANH

SET IsMain = 0

WHERE MaSP = @MaSP

AND MaHinh NOT IN (SELECT MaHinh FROM inserted);

END

END

## Cài đặt Function

1. f\_DemSoLuongNhanVienTheoChuyenMon

Tác dụng:

Hàm này dùng để đếm số lượng nhân viên theo chuyên môn trong hệ thống. Nó trả về một giá trị kiểu INT thể hiện tổng số nhân viên có chuyên môn được chỉ định.

Các bước xử lý:

1. Tham số đầu vào
   * @MaCM INT: Mã chuyên môn cần kiểm tra.
2. Khai báo biến kết quả
   * @SoLuong INT: Biến lưu số lượng nhân viên có chuyên môn tương ứng.
3. Truy vấn dữ liệu
   * Thực hiện SELECT COUNT(\*) từ bảng NhanVienChuyenMon với điều kiện MaCM = @MaCM.
   * Gán kết quả vào biến @SoLuong.
4. Trả về kết quả
   * Hàm trả về giá trị @SoLuong.

Code trong SQL:

ALTER FUNCTION [dbo].[f\_DemSoLuongNhanVienTheoChuyenMon]

(

@MaCM INT

)

RETURNS INT

AS

BEGIN

DECLARE @SoLuong INT;

SELECT @SoLuong = COUNT(\*)

FROM NhanVienChuyenMon

WHERE MaCM = @MaCM;

RETURN @SoLuong;

END

1. f\_KiemTraNhanVienCoChucVu

Tác dụng:

Hàm này dùng để kiểm tra xem một nhân viên có giữ chức vụ cụ thể hay không. Nó trả về giá trị kiểu BIT:

* 1 → Nhân viên có chức vụ được chỉ định.
* 0 → Nhân viên không có chức vụ đó.

Các bước xử lý:

1. Tham số đầu vào
   * @MaNV INT: Mã nhân viên cần kiểm tra.
   * @TenChucVuKiemTra NVARCHAR(50): Tên chức vụ cần kiểm tra.
2. Truy vấn dữ liệu
   * Thực hiện JOIN giữa bảng NhanVien và ChucVu dựa trên MaChucVu.
   * Kiểm tra điều kiện:
     + NV.MaNV = @MaNV
     + CV.TenChucVu = @TenChucVuKiemTra
3. Trả về kết quả
   * Nếu tồn tại bản ghi thỏa mãn → RETURN 1.
   * Nếu không tồn tại → RETURN 0.

Code trong SQL:

ALTER FUNCTION [dbo].[f\_KiemTraNhanVienCoChucVu] (

@MaNV INT,

@TenChucVuKiemTra NVARCHAR(50)

)

RETURNS BIT

AS

BEGIN

-- Kiểm tra xem nhân viên có chức vụ được chỉ định hay không

IF EXISTS (

SELECT 1

FROM NhanVien NV

JOIN ChucVu CV ON NV.MaChucVu = CV.MaChucVu

WHERE NV.MaNV = @MaNV

AND CV.TenChucVu = @TenChucVuKiemTra

)

BEGIN

RETURN 1; -- Có chức vụ này

END

RETURN 0; -- Không có

END

1. fn\_TimKiemSanPham

Tác dụng:

Hàm này dùng để tìm kiếm sản phẩm trong hệ thống theo nhiều tiêu chí. Nó trả về một bảng kết quả (table-valued function) chứa thông tin chi tiết của sản phẩm phù hợp với điều kiện tìm kiếm.

Các bước xử lý:

1. Tham số đầu vào
   * @MaSP INT = NULL: Mã sản phẩm cần tìm (có thể bỏ qua).
   * @TenSP NVARCHAR(100) = NULL: Tên sản phẩm cần tìm (có thể bỏ qua, hỗ trợ tìm kiếm gần đúng bằng LIKE).
2. Truy vấn dữ liệu
   * Lấy dữ liệu từ bảng SanPham.
   * Điều kiện lọc:
     + Nếu @MaSP khác NULL → lọc theo mã sản phẩm.
     + Nếu @TenSP khác NULL → lọc theo tên sản phẩm có chứa chuỗi nhập vào (LIKE '%TênSP%').
   * Nếu cả hai tham số đều NULL → trả về toàn bộ sản phẩm.
3. Kết quả trả về
   * Một bảng gồm các cột:
     + MaSP
     + TenSP
     + MaLoaiSP
     + DonGia
     + SoLuongTon
     + GiaKhuyenMai
     + Hang
     + XuatXu
     + BaoHanh
     + MoTaChung
     + MoTaChiTiet

Code trong SQL:

ALTER FUNCTION [dbo].[fn\_TimKiemSanPham]

(

@MaSP INT = NULL,

@TenSP NVARCHAR(100) = NULL

)

RETURNS TABLE

AS

RETURN

(

SELECT

MaSP,

TenSP,

MaLoaiSP,

DonGia,

SoLuongTon,

GiaKhuyenMai,

Hang,

XuatXu,

BaoHanh,

MoTaChung,

MoTaChiTiet

FROM SanPham

WHERE

(@MaSP IS NULL OR MaSP = @MaSP)

AND (@TenSP IS NULL OR TenSP LIKE N'%' + @TenSP + N'%')

);

# QUẢN TRỊ HỆ THỐNG

Chương này trình bày các giải pháp quản trị cơ sở dữ liệu và quản lý người dùng mà nhóm đã triển khai cho hệ thống quản lý phòng gym. Các nội dung bao gồm: giải pháp phân quyền – xác thực, tạo tài khoản đăng nhập, gán nhóm quyền, cấp và thu hồi quyền truy cập, cùng với cơ chế sao lưu – phục hồi dữ liệu và thiết lập lịch sao lưu định kỳ nhằm đảm bảo an toàn dữ liệu.

## Chọn lựa cơ chế xác thực

Hệ thống sử dụng cơ chế xác thực tiêu chuẩn của SQL Server:

1. SQL Authentication

* Mỗi người dùng có Login (username – password) riêng do quản trị viên tạo.
* Dễ quản lý, độc lập với tài khoản Windows.
* Phù hợp với ứng dụng WinForm/Web kết nối từ nhiều thiết bị.

1. Cơ chế Mapping Login → User

* Login xác thực vào SQL Server.
* User gắn với database để thực thi quyền.
* Roles quản lý quyền theo nhóm → dễ mở rộng và bảo trì.

## Phân quyền các actor trong cơ sở dữ liệu

### Actor quản lý nhân sự

Nhóm quyền: **ManagerRole**

Cấp phát quyền:

* + Trên bảng dữ liệu

|  |  |
| --- | --- |
| **Bảng dữ liệu** | **Quyền được cấp** |
| NhanVien | SELECT, INSERT, UPDATE |
| ChucVu | SELECT, INSERT, UPDATE |
| ChuyenMon | SELECT, INSERT, UPDATE |
| NhanVienChuyenMon | SELECT, INSERT, UPDATE |
| GoiTap | SELECT, INSERT, UPDATE |
| HoSo | SELECT, INSERT, UPDATE |
| LichHop | SELECT, INSERT, UPDATE |
| HoaDon | SELECT, INSERT, UPDATE |
| ChiTietHoaDon | SELECT, INSERT, UPDATE |
| DiaChi | SELECT, INSERT, UPDATE |

* + Trên procedure:

|  |  |
| --- | --- |
| **Procedure** | **Quyền được cấp** |
| sp\_SuaChuyenMon | EXECUTE |
| sp\_SuaLopHoc | EXECUTE |
| sp\_ThemChuyenMon | EXECUTE |
| sp\_ThemChucVu | EXECUTE |
| sp\_GanChuyenMonChoNhanVien | EXECUTE |
| sp\_ThemNhanVien | EXECUTE |
| sp\_XoaChucVu | EXECUTE |
| sp\_XoaNhanVien | EXECUTE |
| sp\_XoaNhanVienChuyenMon | EXECUTE |

### Actor khách hàng

Nhóm quyền: **CustomerRole**

Cấp phát quyền:

* + Trên bảng dữ liệu:

|  |  |
| --- | --- |
| **Bảng dữ liệu** | **Quyền được cấp** |
| KhachHang | SELECT, UPDATE |
| DiaChi | SELECT, INSERT, UPDATE |
| HoaDon | SELECT, INSERT, UPDATE |
| ChiTietHoaDon | SELECT, INSERT |
| DangKyGoiTap | SELECT, INSERT, UPDATE |
| DangKyLop | SELECT, INSERT |
| DangKyPT | SELECT, INSERT, UPDATE |
| LichLop | SELECT, UPDATE |
| LichTapPT | SELECT, UPDATE |
| ChiTietGioHang | SELECT, UPDATE, DELETE |
| GoiTap | SELECT |
| ChuyenMon | SELECT |
| SanPham | SELECT |
| LoaiSanPham | SELECT |
| HINHANH | SELECT |
| LoaiKhachHang | SELECT |
| NhanVien | SELECT |

* + Trên procedure:

|  |  |
| --- | --- |
| **Procedure** | **Quyền được cấp** |
| sp\_KhachHangLogin | EXECUTE |
| sp\_DangKyLop\_KiemTra | EXECUTE |
| sp\_KhachHangDangKy | EXECUTE |
| sp\_ThemVaoGioHang | EXECUTE |
| sp\_DangKyTapThu | EXECUTE |

### Actor quản lý sản phẩm

Nhóm quyền: **StaffRole**

Cấp phát quyền:

* Trên bảng dữ liệu:

|  |  |
| --- | --- |
| **Bảng** | **Quyền được cấp** |
| SanPham | SELECT |
| LoaiSanPham | SELECT |
| NhaCungCap | SELECT |

* Trên procedure:

|  |  |
| --- | --- |
| **Procedure** | **Quyền được cấp** |
| sp\_ThemSanPham | EXECUTE |
| sp\_SuaSanPham | EXECUTE |
| sp\_XoaSanPham | EXECUTE |
| sp\_ThemLoaiSanPham | EXECUTE |
| sp\_SuaLoaiSanPham | EXECUTE |
| sp\_XoaLoaiSanPham | EXECUTE |
| sp\_ThemNhaCungCap | EXECUTE |
| sp\_SuaNhaCungCap | EXECUTE |
| sp\_XoaNhaCungCap | EXECUTE |
| sp\_ThemAnh\_Xoa | EXECUTE |

### Actor huấn luyện viên

Nhóm quyền: **QL\_Lop**

Cấp phát quyền:

* Trên bảng dữ liệu:

|  |  |
| --- | --- |
| **Bảng** | **Quyền được cấp** |
| LopHoc | SELECT, INSERT, UPDATE, DELETE |
| DangKyPT | SELECT, INSERT, UPDATE, DELETE |
| LichTapPT | SELECT, INSERT, UPDATE, DELETE |
| HoaDon | SELECT, INSERT |
| ChiTietHoaDon | SELECT, INSERT |
| KhachHang | SELECT |
| NhanVien | SELECT |

* Trên các procedure:

|  |  |
| --- | --- |
| **Procedure** | **Quyền** |
| sp\_ThemLichLop | EXECUTE |
| sp\_ThemLichTapPT | EXECUTE |
| sp\_ThemLopHocValich | EXECUTE |
| XoaLichValop | EXECUTE |

## Sao lưu dữ liệu

Sao lưu được thiết kế theo 3 cấp độ: Full Backup, Differential Backup, và Transaction Log Backup, nhằm đảm bảo dữ liệu có thể phục hồi trong trường hợp lỗi hệ thống, mất điện hoặc người dùng xóa nhầm dữ liệu.

### Sao lưu toàn bộ (Full Backup)

BACKUP DATABASE QL\_PHONGGYM

TO DISK = 'D:\Database\Backup-QL\_PHONGGYM\QL\_PHONGGYM\_Full.bak'

WITH INIT, NAME = 'Full Backup';

### Sao lưu khác biệt (Differential Backup)

BACKUP DATABASE QL\_PHONGGYM

TO DISK = 'D:\Database\Backup-QL\_PHONGGYM\QL\_PHONGGYM\_Diff.bak'

WITH DIFFERENTIAL, NAME = 'Differential Backup';

### Sao lưu log giao dịch (Transaction Log Backup)

BACKUP LOG QL\_PHONGGYM

TO DISK = 'D:\Database\Backup-QL\_PHONGGYM\QL\_PHONGGYM\_LOG.trn'

# XÂY DỰNG ỨNG DỤNG

**4.1. Cài đặt kết nối từ ứng dụng đến SQL Server**

Các thành viên trong nhóm đều sử dụng chung một **chuỗi kết nối cơ bản**, thay đổi tùy vào tài khoản được cấp quyền:

A computer screen shot of a computer code

AI-generated content may be incorrect.

Mỗi bạn dùng **Login/User riêng** (đã được tạo ở Chương 3), nhưng database **GymDB** là tập trung và dùng chung cho cả nhóm.

## Chức năng các thành viên triển khai

### Thành viên “Nguyễn Hoàng Thành” – Ứng dụng HLV quản lý lớp và khách PT

**a) Kiến trúc hệ thống**

Mô hình MVC:

* + Model: Các lớp thực thể như NhanVien, LopHoc, LichLop, DangKyPT, HoaDon, ChiTietHoaDon, KhachHang.
  + View: Các giao diện Razor (.cshtml) hiển thị lịch làm việc, danh sách lớp, danh sách khách PT, form đăng nhập.
  + Controller: QL\_Gym\_HLVController xử lý logic, gọi dữ liệu từ Model, truyền sang View.

*Khi thêm lịch đều đã kiểm tra lịch của nhân viên, đảm bảo không có trường hợp bị trùng lịch*

**b) Các chức năng chính trong Controller**

Đăng nhập/Đăng xuất HLV

* HLV\_DangNhap(): Đăng nhập huấn luyện viên, lưu thông tin vào Session["hlv"].

A screenshot of a fitness app

AI-generated content may be incorrect.

* HLV\_DangXuat(): Đăng xuất, xóa session.

A close-up of a sign

AI-generated content may be incorrect.

Quản lý lịch làm việc

* Phương thức loadSchedule(DateTime selectDate):
  + Tính toán tuần (thứ 2 → CN).
  + Lấy lịch lớp (LichLop) và lịch PT (LichTapPT) của huấn luyện viên đăng nhập.
  + Phân loại theo buổi sáng, chiều, tối.
  + Truyền dữ liệu sang View qua ViewBag.
* LichLamViec(): Hiển thị lịch làm việc tuần hiện tại.

A screenshot of a computer

AI-generated content may be incorrect.

* ChinhSuaLich(): Cho phép chỉnh sửa lịch theo ngày.
* XoaLich(): Xóa lịch lớp hoặc lịch PT.

A screenshot of a computer

AI-generated content may be incorrect.

* ThemLich(): Thêm lịch lớp hoặc lịch PT bằng Stored Procedure.

A screenshot of a computer

AI-generated content may be incorrect.

Quản lý lớp học

* DanhSachLop(): Hiển thị danh sách lớp mà huấn luyện viên phụ trách.

A screenshot of a computer

AI-generated content may be incorrect.

* ChinhSuaLop(): Chỉnh sửa thông tin lớp học (tên lớp, học phí, ngày bắt đầu/kết thúc, sĩ số).

A screenshot of a computer

AI-generated content may be incorrect.

* ThemLop(): Thêm lớp mới, có thể gọi Stored Procedure sp\_ThemLopHocVaLich để tự động tạo lịch.

A screenshot of a chat

AI-generated content may be incorrect. A screenshot of a chat

AI-generated content may be incorrect.

* XoaLichVaLop(): Xóa lớp và toàn bộ lịch liên quan.

A screenshot of a computer

AI-generated content may be incorrect.

Quản lý khách hàng PT

* DanhSachKhachPT(): Hiển thị danh sách khách PT theo trạng thái (Chờ duyệt, Còn hiệu lực, Kết thúc).
* TimKiem(): Tìm kiếm lớp học hoặc khách PT theo từ khóa.

A screenshot of a computer

AI-generated content may be incorrect.

* NhanPT(): Nhận khách PT, tự động tạo hóa đơn và chi tiết hóa đơn, áp dụng giảm giá theo loại khách hàng.

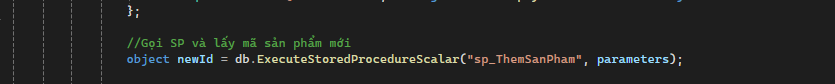
A screenshot of a computer

AI-generated content may be incorrect.

### Thành viên “Nguyễn Thành Long” – Ứng dụng Quản lý sản phẩm

**a) Sử dụng Stored Procedure**

Ứng dụng gọi trực tiếp SP sp\_ThemSanPham để thêm mới sản phẩm vào hệ thống.



**b) Sử dụng Transaction**

* Trong thủ tục sửa loại sản phẩm khi người dùng chỉnh sửa thông tin loại sản phẩm, thủ tục:

A screenshot of a computer code

AI-generated content may be incorrect.

* Gọi thủ tục trong C#



**c) Sử dụng Function**

* Ứng dụng có chức năng tìm kiếm sản phẩm theo tên hoặc mã, gọi chức năng:

Hàm tìm kiếm

A screenshot of a computer

AI-generated content may be incorrect.

Gọi trong C#

A black screen with white text

AI-generated content may be incorrect.

**d) Sử dụng Cursor**

Trigger trg\_XoaTuanTuTheoLoai mà ứng dụng kích hoạt có sử dụng nhiều Cursor lồng nhau:

* Cursor duyệt danh sách loại
* Cursor duyệt danh sách sản phẩm
* Cursor duyệt danh sách hình ảnh của từng sản phẩm
* Nhờ đó việc xóa dữ liệu diễn ra tuần tự và không vi phạm khóa ngoại.

A screenshot of a computer

AI-generated content may be incorrect.

**e) Sử dụng Transaction**

* Khi người dùng chỉnh sửa thông tin sản phẩm hoặc loại sản phẩm, thủ tục được kích hoạt. Thủ tục này có BEGIN TRAN – COMMIT – ROLLBACK, bảo vệ dữ liệu trong trường hợp xảy ra lỗi.
* Ứng dụng nhận lại mã kết quả để hiển thị thông báo phù hợp.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Gọi trong C#

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

### Thành viên “Phạm Đặng Gia Thuận” – Ứng dụng Quản lí nhân viên

1. Giới thiệu mô đun và công nghệ sử dụng

Phân hệ Nhân sự là cổng giao tiếp chính giữa nhân viên và nhân viên chuyên môn. Mô đun này được xây dựng trên nền tảng **ASP.NET MVC 5**, tập trung vào trải nghiệm người dùng (UI/UX) và tính toàn vẹn dữ liệu khi có nhiều người truy cập đồng thời.

* **Ngôn ngữ:** C# (Backend), Razor (Frontend).
* **Mô hình kết nối:** Kết hợp Entity Framework (cho truy vấn xem) và ADO.NET (cho các nghiệp vụ ghi dữ liệu phức tạp).
* **Điểm nhấn kỹ thuật:** Áp dụng đầy đủ các kỹ thuật SQL nâng cao đã học như **Stored Procedure, Transaction, Trigger** để xử lý logic nghiệp vụ ngay tại tầng cơ sở dữ liệu.

1. Cài đặt kết nối từ ứng dụng đến SQL Server

A. Chuỗi kết nối (Connection String)

A black screen with white text

AI-generated content may be incorrect.

B. Lớp ngữ cảnh dữ liệu (DbContext)

A screen shot of a computer

AI-generated content may be incorrect.

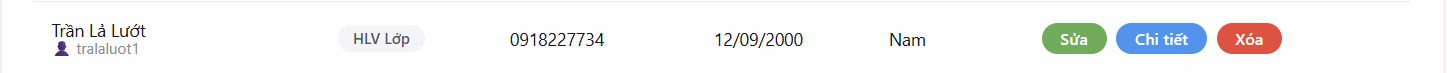
1. Gọi thực thi Procedure/Function/Trigger trong code

Ví dụ 1: Gọi Stored Procedure (Xóa nhân viên) -NhanVienController.cs

A computer screen shot of a program code

AI-generated content may be incorrect.

**Tạo nhân viên mới:**



**Gán chuyên môn cho nhân viên:**

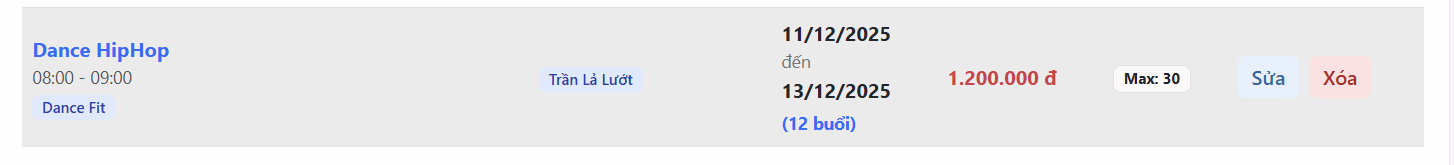
A screenshot of a computer

AI-generated content may be incorrect.

**Thêm nhân viên chuyên môn vào lớp học mới:**

A screenshot of a computer

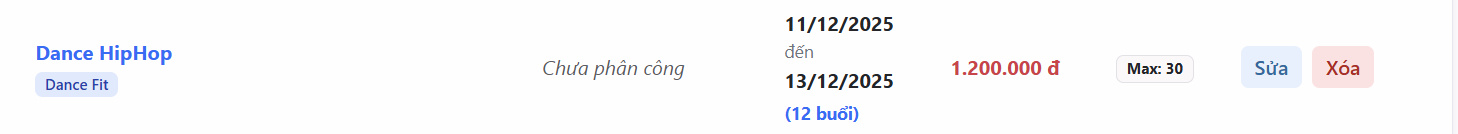
AI-generated content may be incorrect.



**Sau khi tạo lớp học nó sẽ tự tạo lịch học mới bằng sp\_ThemLopHocVaLich**



**Sau khi xóa nó sẽ hiện ra chưa phân công**



**Mục đích:** Xóa hồ sơ nhân viên (Thôi việc).

**Xử lý chính:**

Kiểm tra an toàn: Cấm xóa nếu nhân viên đã có giao dịch tài chính (Hóa đơn bán hàng/PT) để bảo vệ dữ liệu kế toán.

Transaction xóa dây chuyền (Cascade Delete thủ công):

* Xóa Lịch dạy (LichLop).
* Xóa Chuyên môn (NhanVienChuyenMon).
* Cập nhật lớp chủ nhiệm (LopHoc) về NULL.
* Cuối cùng xóa NhanVien.

Ví dụ 2: Gọi Trigger (Kiểm tra tuổi nhân viên) - NhanVienController.cs

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.



Trigger trg\_NhanVien\_KiemTraTuoi được kích hoạt tự động khi thêm mới nhân viên. Nếu nhân viên dưới 18 tuổi, Trigger sẽ ROLLBACK và trả về lỗi. Ứng dụng sẽ bắt lỗi này và hiển thị cho người dùng.

Ví dụ 3: Gọi function (Đếm số lượng nhân viên theo chuyên môn)

A blue and white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Trong hệ thống của bạn, bảng ChuyenMon (Ví dụ: Yoga, Boxing) và bảng NhanVien (Các HLV) có mối quan hệ Nhiều-Nhiều thông qua bảng trung gian NhanVienChuyenMon.

* Một môn có thể do nhiều HLV dạy.
* Một HLV có thể dạy nhiều môn.

### Thành viên “Nguyễn Ngà” – Ứng dụng khách hàng

**a) Giới thiệu mô đun & Công nghệ sử dụng**

Phân hệ Khách hàng là cổng giao tiếp chính giữa hội viên và phòng tập. Mô đun này được xây dựng trên nền tảng **ASP.NET MVC 5**, tập trung vào trải nghiệm người dùng (UI/UX) và tính toàn vẹn dữ liệu khi có nhiều người truy cập đồng thời.

* **Ngôn ngữ:** C# (Backend), Razor (Frontend).
* **Mô hình kết nối:** Kết hợp Entity Framework (cho truy vấn xem) và ADO.NET (cho các nghiệp vụ ghi dữ liệu phức tạp).
* A screenshot of a website

  AI-generated content may be incorrect.**Điểm nhấn kỹ thuật:** Áp dụng đầy đủ các kỹ thuật SQL nâng cao đã học như **Stored Procedure, Transaction, Trigger** để xử lý logic nghiệp vụ ngay tại tầng cơ sở dữ liệu.

**b) Cấu hình kết nối SQL Server**

Kết nối CSDL được thiết lập trong file Web.config, trỏ trực tiếp đến máy chủ Database tập trung qua địa chỉ IP LAN.

**Code cấu hình thực tế (Web.config):**

<connectionStrings>  
 <add name="QL\_PHONGGYMEntities" connectionString="metadata=res://\*/Models.Model1.csdl|res://\*/Models.Model1.ssdl|res://\*/Models.Model1.msl;provider=System.Data.SqlClient;provider connection string="data source=100.70.158.110;initial catalog=QL\_PHONGGYM;persist security info=True;user id=GymCustomerLogin;password=GymCustomerLogin!;trustservercertificate=True;MultipleActiveResultSets=True;App=EntityFramework""   
 providerName="System.Data.EntityClient" />  
</connectionStrings>

A screen shot of a computer program

AI-generated content may be incorrect.

**c) Hiện thực các chức năng nghiệp vụ chi tiết**

Dưới đây là các chức năng chính do sinh viên thực hiện, minh họa việc gọi thực thi các đối tượng CSDL nâng cao:

A. Chức năng Đăng ký tài khoản (Transaction & Kiểm tra trùng lặp)

**Mô tả:** Sử dụng Stored Procedure sp\_KhachHangDangKy để đảm bảo tính duy nhất của Tên đăng nhập, Email và SĐT trong một giao dịch an toàn (Atomic).

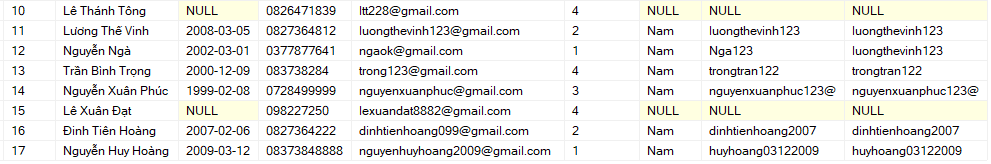
**1. Mã nguồn C# (AccountRepository.cs):**

public bool CusRegister(KhachHangRegisterViewModel model)  
{  
 try  
 {  
 // Gọi SP thay vì dùng EF.Add() để đảm bảo hiệu năng và kiểm tra ràng buộc chặt chẽ  
 \_context.Database.ExecuteSqlCommand(  
 "EXEC sp\_KhachHangDangKy @TenKH, @GioiTinh, @NgaySinh, @SDT, @Email, @TenDangNhap, @MatKhau",  
 new SqlParameter("@TenKH", model.TenKH),  
 new SqlParameter("@GioiTinh", model.GioiTinh),  
 new SqlParameter("@NgaySinh", (object)model.NgaySinh ?? DBNull.Value),  
 new SqlParameter("@SDT", model.SDT),  
 new SqlParameter("@Email", model.Email),  
 new SqlParameter("@TenDangNhap", model.TenDangNhap),  
 new SqlParameter("@MatKhau", model.MatKhau)  
 );  
 return true;  
 }  
 catch { return false; }  
}

**2. Mã nguồn SQL (Stored Procedure):**

ALTER PROCEDURE [dbo].[sp\_KhachHangDangKy]  
 @TenKH NVARCHAR(100), @GioiTinh NVARCHAR(60), @NgaySinh DATE,  
 @SDT NVARCHAR(15), @Email NVARCHAR(100),  
 @TenDangNhap VARCHAR(300), @MatKhau VARCHAR(1000)  
AS  
BEGIN  
 SET NOCOUNT ON;  
 BEGIN TRANSACTION; -- Bắt đầu giao dịch  
  
 BEGIN TRY  
 -- Kiểm tra trùng lặp (Business Logic tại DB)  
 IF EXISTS (SELECT 1 FROM KhachHang WHERE TenDangNhap = @TenDangNhap)  
 BEGIN  
 RAISERROR (N'Tên đăng nhập này đã được sử dụng.', 16, 1);  
 ROLLBACK TRANSACTION;  
 RETURN;

END  
  
 -- Insert dữ liệu  
 INSERT INTO KhachHang (TenKH, NgaySinh, SDT, Email, GioiTinh, TenDangNhap, MatKhau)  
 VALUES (@TenKH, @NgaySinh, @SDT, @Email, @GioiTinh, @TenDangNhap, @MatKhau);  
  
 COMMIT TRANSACTION;  
 END TRY  
 BEGIN CATCH  
 IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;  
 THROW;  
 END CATCH  
END



A screenshot of a computer

AI-generated content may be incorrect.

B. Chức năng Thêm vào giỏ hàng (Logic Upsert & Kiểm tra tồn kho)

**Mô tả:** Xử lý logic phức tạp: Kiểm tra tồn kho -> Nếu có trong giỏ thì Update số lượng, chưa có thì Insert.

**1. Mã nguồn C# (CartRepository.cs):**

public bool AddToCart(int maKH, int maSP, int soLuong)  
{  
 try  
 {  
 \_context.Database.ExecuteSqlCommand(  
 "EXEC sp\_ThemVaoGioHang @MaKH, @MaSP, @SoLuong",  
 new SqlParameter("@MaKH", maKH),  
 new SqlParameter("@MaSP", maSP),  
 new SqlParameter("@SoLuong", soLuong)  
 );  
 return true;   
 }  
 catch (SqlException ex)  
 {  
 // Bắt lỗi RAISERROR từ SQL để hiển thị thông báo  
 throw ex;   
 }  
}

**2. Mã nguồn SQL (Stored Procedure):**

ALTER PROCEDURE [dbo].[sp\_ThemVaoGioHang]  
 @MaKH INT, @MaSP INT, @SoLuong INT  
AS  
BEGIN  
 SET NOCOUNT ON;  
 BEGIN TRANSACTION;  
  
 DECLARE @SoLuongTon INT;  
 SELECT @SoLuongTon = SoLuongTon FROM SanPham WHERE MaSP = @MaSP;  
  
 -- Kiểm tra tồn kho  
 IF (@SoLuongTon < @SoLuong)  
 BEGIN  
 ROLLBACK TRANSACTION;  
 RAISERROR (N'Sản phẩm không đủ số lượng tồn kho!', 16, 1);  
 RETURN;  
 END  
  
 -- Logic Upsert (Update hoặc Insert)  
 IF EXISTS (SELECT 1 FROM ChiTietGioHang WHERE MaKH = @MaKH AND MaSP = @MaSP)  
 BEGIN  
 UPDATE ChiTietGioHang   
 SET SoLuong = SoLuong + @SoLuong   
 WHERE MaKH = @MaKH AND MaSP = @MaSP;  
 END  
 ELSE  
 BEGIN  
 DECLARE @DonGia DECIMAL(18,0);  
 SELECT @DonGia = DonGia FROM SanPham WHERE MaSP = @MaSP;  
 INSERT INTO ChiTietGioHang (MaKH, MaSP, SoLuong, DonGia, NgayThem)  
 VALUES (@MaKH, @MaSP, @SoLuong, @DonGia, GETDATE());  
 END  
  
 COMMIT TRANSACTION;  
A screenshot of a product

AI-generated content may be incorrect.END

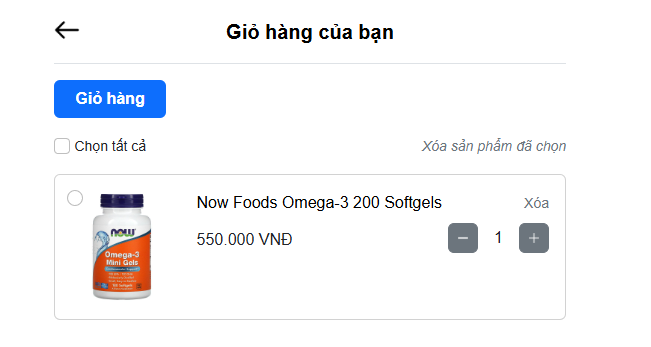
**A screenshot of a phone

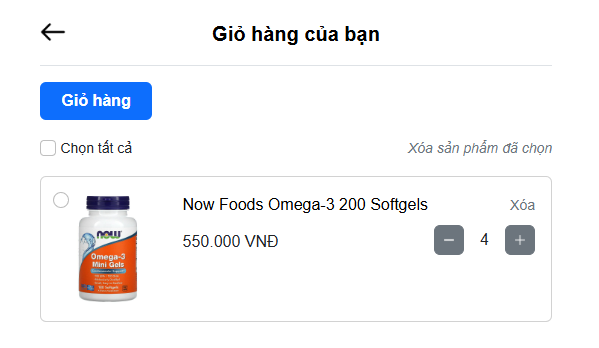
AI-generated content may be incorrect.**

C. Chức năng Xóa & Chỉnh sửa số lượng sản phẩm trong giỏ hàng

**1. Chỉnh sửa số lượng sản phẩm**

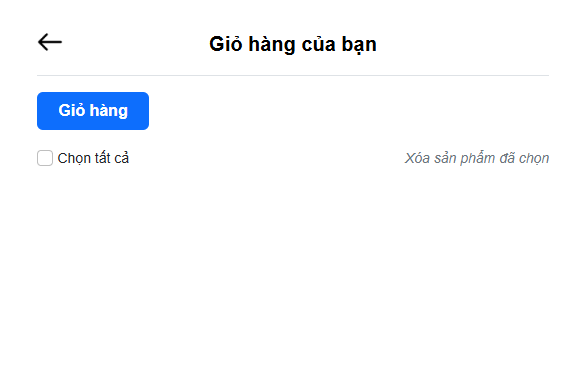
**Mô tả:** Khách hàng có thể thay đổi số lượng của từng sản phẩm đã chọn trong giỏ hàng. Hệ thống hỗ trợ tăng/giảm số lượng thông qua các nút thao tác (+/-) hoặc nhập trực tiếp số lượng mong muốn vào ô nhập liệu.





**2. Xóa sản phẩm khỏi giỏ hàng**

**Mô tả:** Cho phép khách hàng loại bỏ các sản phẩm không còn nhu cầu mua sắm ra khỏi danh sách giỏ hàng



D. Chức năng Thanh toán & Đăng ký Lớp (Transaction Scope & Locking)

**Mô tả:** Đây là chức năng quan trọng nhất, sử dụng DbContext.Database.BeginTransaction() để đảm bảo tính toàn vẹn dữ liệu qua nhiều bảng (Hóa đơn, Chi tiết, Đăng ký lớp). Đặc biệt áp dụng kỹ thuật **Locking** để ngăn chặn Overbooking (vượt quá sĩ số lớp).

**1. Mã nguồn C# (CartRepository.cs - Hàm TaoHoaDon):**

public void TaoHoaDon(FormCollection form, int makh, List<GioHangViewModel> cart, int? maLop)  
{  
 // Bắt đầu Transaction C# để bao gói toàn bộ quy trình  
 using (var trans = \_context.Database.BeginTransaction())  
 {  
 try  
 {  
 // 1. Tạo Header Hóa Đơn  
 HoaDon hoaDon = new HoaDon {  
 MaKH = makh,  
 TongTien = Convert.ToDecimal(form["tongTien"]),  
 /\* ... gán các thuộc tính khác ... \*/  
 NgayLap = DateTime.Now  
 };  
 \_context.HoaDons.Add(hoaDon);  
 \_context.SaveChanges();  
  
 // 2. Xử lý Đăng ký lớp (Gọi SP có Locking)  
 if (maLop.HasValue)  
 {  
 var idParam = new SqlParameter("@MaDKLop", System.Data.SqlDbType.Int) { Direction = System.Data.ParameterDirection.Output };  
   
 // Gọi SP kiểm tra sĩ số và khóa dữ liệu  
 \_context.Database.ExecuteSqlCommand(  
 "EXEC sp\_DangKyLop\_KiemTra @MaKH, @MaLop, @MaDKLop OUT",  
 new SqlParameter("@MaKH", makh),  
 new SqlParameter("@MaLop", maLop),  
 idParam  
 );  
   
 // Tạo chi tiết hóa đơn cho lớp học  
 int newMaDKLop = (int)idParam.Value;  
 /\* ... code tạo ChiTietHoaDon ... \*/  
 }  
  
 // 3. Xử lý Sản phẩm & Gói tập (như code hiện tại)  
 /\* ... Code vòng lặp foreach insert ChiTietHoaDon ... \*/  
  
 \_context.SaveChanges();  
   
 // Nếu không có lỗi -> Commit Transaction  
 trans.Commit();  
 }  
 catch (Exception)  
 {  
 // Nếu có lỗi (Lớp đầy, Lỗi mạng) -> Rollback toàn bộ  
 trans.Rollback();  
 throw;  
 }  
 }  
}

**2. Mã nguồn SQL (SP Kiểm tra Lớp học & Locking):**

CREATE PROCEDURE [dbo].[sp\_DangKyLop\_KiemTra]  
 @MaKH INT, @MaLop INT, @MaDKLop INT OUTPUT  
AS  
BEGIN  
 SET NOCOUNT ON;  
   
 -- Kiểm tra sĩ số với UPDLOCK (Khóa cập nhật - Chặn người khác đọc/ghi)  
 DECLARE @SiSoHienTai INT, @SiSoToiDa INT;  
   
 SELECT @SiSoHienTai = COUNT(\*)   
 FROM DangKyLop WITH (UPDLOCK) -- <== Kỹ thuật Locking quan trọng  
 WHERE MaLop = @MaLop AND TrangThai = N'Còn hiệu lực';  
  
 SELECT @SiSoToiDa = SiSoToiDa FROM LopHoc WHERE MaLop = @MaLop;  
  
 IF (@SiSoHienTai >= @SiSoToiDa)  
 BEGIN  
 RAISERROR (N'Lớp học đã đủ sĩ số!', 16, 1);  
 RETURN;  
 END  
  
 -- Nếu còn chỗ -> Insert  
 INSERT INTO DangKyLop (MaKH, MaLop, NgayDangKy, TrangThai)  
 VALUES (@MaKH, @MaLop, GETDATE(), N'Còn hiệu lực');  
  
 SET @MaDKLop = SCOPE\_IDENTITY();

END;

A screenshot of a computer

AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

E. Tự động hóa nghiệp vụ bằng Trigger

**Mô tả:** Hệ thống sử dụng Trigger để tự động trừ kho và nâng hạng khách hàng, đảm bảo dữ liệu nhất quán mà không phụ thuộc vào tầng ứng dụng.

**1. Trigger Trừ Kho (trg\_CapNhatSoLuongTon):**

ALTER TRIGGER [dbo].[trg\_CapNhatSoLuongTon]  
ON [dbo].[ChiTietHoaDon]  
AFTER INSERT  
AS  
BEGIN  
 SET NOCOUNT ON;  
 -- Tự động trừ số lượng tồn trong bảng SanPham  
 UPDATE S  
 SET S.SoLuongTon = S.SoLuongTon - i.SoLuong  
 FROM SanPham S  
 JOIN inserted i ON S.MaSP = i.MaSP  
 WHERE i.MaSP IS NOT NULL;  
END

**2. Trigger Nâng hạng Khách hàng (trg\_CapNhatLoaiKhachHangHoaDon):**

ALTER TRIGGER [dbo].[trg\_CapNhatLoaiKhachHangHoaDon]  
ON [dbo].[HoaDon]  
AFTER INSERT  
AS  
BEGIN  
 -- Tự động tính tổng tiền và nâng hạng  
 UPDATE kh  
 SET MaLoaiKH = CASE   
 WHEN TongChiTieu >= 5000000 THEN @MaLoaiVip  
 WHEN TongChiTieu >= 1000000 THEN @MaLoaiMember  
 ELSE MaLoaiKH   
 END  
 FROM KhachHang kh  
 INNER JOIN (  
 SELECT MaKH, SUM(ThanhTien) AS TongChiTieu  
 FROM HoaDon WHERE TrangThai = N'Đã thanh toán' GROUP BY MaKH  
 ) hd ON kh.MaKH = hd.MaKH;  
END

**Minh chứng hoạt động:**

* **Trước:** Sản phẩm A (Tồn: 100). Khách hàng B (Hạng: Thường).
* **Hành động:** Khách B mua Sản phẩm A trị giá 6.000.000đ.
* **Sau:** Sản phẩm A (Tồn: 99). Khách hàng B (Hạng: VIP).

A close up of a bottle

AI-generated content may be incorrect.



## KẾT LUẬN CHƯƠNG

Trong chương này, nhóm đã xây dựng các ứng dụng WinForm/Web tương ứng cho từng thành viên, tất cả cùng kết nối và sử dụng chung cơ sở dữ liệu SQL Server. Mỗi ứng dụng đều thực hiện đúng yêu cầu khi tích hợp các cấu trúc xử lý đã triển khai ở các chương trước như Stored Procedure, Function, Trigger, Cursor và Transaction. Các chức năng của ứng dụng đã được kiểm thử và hoạt động ổn định, thể hiện khả năng tương tác tốt với cơ sở dữ liệu tập trung. Việc mỗi thành viên xây dựng một module riêng nhưng dùng chung một CSDL giúp hệ thống thống nhất, dễ mở rộng và đảm bảo dữ liệu được xử lý nhất quán, an toàn.

# TÀI LIỆU THAM KHẢO

* SÁCH
* NGUỒN TỪ INTERNET VÀ WEBSITE