

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM



BÀI TẬP LỚN

TÊN HỌC PHẦN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

ĐỀ TÀI: Quản Lý Ngân Hàng

STT	Mã sinh viên	Họ và tên	Ngày sinh	Lớp
1	1871070011	Nguyễn Thế Phương Nam	22/12/2006	HTTT 18-01
2	1871070001	Lê Duy An	22/02/2006	HTTT 18-01

Hà Nội, năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM



BÀI TẬP LỚN

TÊN HỌC PHẦN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

ĐỀ TÀI: Quản Lý Ngân Hàng

STT	Mã sinh viên	Họ và tên	Ngày sinh	Điểm	
				Bằng số	Bằng chữ
1	1871070011	Nguyễn Thế Phương Nam	22/12/2006		
2	1871070001	Lê Duy An	22/02/2006		

CÁN BỘ CHẤM THI 1

CÁN BỘ CHẤM THI 2

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Trong bối cảnh nền kinh tế số phát triển mạnh mẽ, hoạt động ngân hàng ngày càng đòi hỏi mức độ chính xác, an toàn và hiệu quả cao trong việc quản lý dữ liệu. Khối lượng thông tin lớn như tài khoản khách hàng, giao dịch, khoản vay, lãi suất, và các dịch vụ tài chính cần được lưu trữ, xử lý và truy xuất một cách nhanh chóng, ổn định và bảo mật. Điều này đặt ra yêu cầu tất yếu cho việc xây dựng một hệ thống cơ sở dữ liệu hiện đại và khoa học.

Đề tài “**Xây dựng hệ thống quản lý ngân hàng**” trong môn *Hệ quản trị cơ sở dữ liệu* được thực hiện nhằm giúp sinh viên nắm vững các kiến thức và kỹ năng về thiết kế, xây dựng và vận hành cơ sở dữ liệu. Thông qua đề tài, nhóm tiến hành phân tích yêu cầu nghiệp vụ ngân hàng, thiết kế mô hình dữ liệu, chuẩn hoá bảng, xây dựng các truy vấn, thủ tục, trigger và đảm bảo các yếu tố toàn vẹn dữ liệu.

Việc triển khai đề tài không chỉ giúp hiểu sâu hơn về vai trò của cơ sở dữ liệu trong các hệ thống tài chính mà còn rèn luyện kỹ năng xây dựng một hệ thống quản lý thực tế, hỗ trợ quá trình học tập và ứng dụng sau này.

MỤC LỤC

LỜI NÓI ĐẦU	3
MỤC LỤC	4
CHƯƠNG 1. GIỚI THIỆU	6
1.1 Tóm tắt về cơ sở dữ liệu	6
1.2. Công cụ và công nghệ sử dụng	6
1.2.1 Microsoft SQL Server	6
1.2.2 SQL Server Management Studio (SSMS)	7
1.3 Bảng phân công	7
CHƯƠNG 2. Phân tích yêu cầu hệ thống	8
2.1 Phân tích và yêu cầu cơ sở dữ liệu (Nghiệp vụ)	8
2.1.1 Nghiệp vụ Quản lý Thông tin (Data Management):	8
2.1.2 Nghiệp vụ Tài khoản (Account Operations):	8
2.1.3 Nghiệp vụ Giao dịch (Transaction Operations):	8
2.1.4 Nghiệp vụ Tra cứu & Báo cáo (Query & Reporting):	9
2.1.5 Nghiệp vụ Bảo mật & Quản trị (Security & Admin):	9
2.2 Các chức năng chính của hệ thống	9
2.2.1. Quản lý Khách hàng & Tài khoản:	9
2.2.2. Quản lý Giao dịch:	9
2.2.3. Báo cáo và Tra cứu:	10
2.2.4. Quản lý Người dùng (Nội bộ):	10
2.2.5. Quản trị Hệ thống:	10
CHƯƠNG 3. THIẾT KẾ	11
3.1 Thiết kế mô hình thực thể quan hệ (ER)	11
3.2 Chuyển đổi mô hình ER thành lược đồ quan hệ	11
3.3 Cấu trúc các bảng dữ liệu	12

CHƯƠNG 4. Cài Đặt CSDL	15
4.1 Tạo bảng	15
4.2 Sơ đồ csdl	18
CHƯƠNG 5. Các Đối Tượng CSDL nâng cao	19
5.1 Cài đặt các chỉ mục (index)	19
5.2 views	19
5.3 Thủ tục (Stored Procedures)	21
5.4 Hàm người dùng tự định nghĩa (User Functions)	26
5.5 trigger	31
CHƯƠNG 6. Bảo mật và quản trị csdl	34
6.1 Quản lý người dùng và quyền quản lý	34
6.2 Bảo mật csdl	35
6.3 Quản lý sao lưu phục hồi	36
6.4 Quản lý hiệu suất csdl	37
CHƯƠNG 7. Kết luận và Khuyến Nghị	40
7.1 Tóm tắt kết quả	40
7.2 Khuyến nghị	41
7.2.1 Khuyến nghị về Áp dụng và Triển khai	41
7.2.2 Khuyến nghị Cải tiến Hệ thống (Ngắn hạn)	41
7.2.3 Khuyến nghị Phát triển và Nghiên cứu (Dài hạn)	41
KẾT LUẬN	43
DANH MỤC TÀI LIỆU THAM KHẢO	44

CHƯƠNG 1. GIỚI THIỆU

1.1 Tóm tắt về cơ sở dữ liệu

- Trong thời đại công nghệ thông tin phát triển mạnh mẽ, cơ sở dữ liệu (CSDL) giữ vai trò then chốt trong hầu hết các hệ thống quản lý thông tin. Cơ sở dữ liệu được hiểu là tập hợp có tổ chức các dữ liệu có liên quan với nhau, được lưu trữ trên các phương tiện máy tính nhằm phục vụ cho việc truy xuất, cập nhật và quản lý thông tin một cách hiệu quả. Việc sử dụng CSDL giúp tổ chức dữ liệu khoa học, tránh trùng lặp, đảm bảo tính toàn vẹn và hỗ trợ khai thác thông tin nhanh chóng.
- Để quản lý dữ liệu một cách hiệu quả, các hệ thống thường sử dụng **Hệ quản trị cơ sở dữ liệu (Database Management System – DBMS)**. DBMS là phần mềm cho phép tạo lập, duy trì và thao tác trên CSDL. Hệ quản trị cơ sở dữ liệu cung cấp các chức năng quan trọng như: định nghĩa cấu trúc dữ liệu, kiểm soát truy cập, bảo mật, sao lưu – phục hồi, tối ưu hóa truy vấn và đảm bảo tính nhất quán của dữ liệu. Các DBMS phổ biến hiện nay gồm có SQL Server, MySQL, Oracle, PostgreSQL...
- Trong đề tài **Quản lý ngân hàng**, cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu đóng vai trò đặc biệt quan trọng. Ngân hàng là lĩnh vực có khối lượng dữ liệu lớn, đa dạng và yêu cầu độ chính xác, bảo mật cao, bao gồm thông tin khách hàng, tài khoản, giao dịch, khoản vay, lãi suất và các dịch vụ tài chính khác. Việc xây dựng một CSDL khoa học giúp hệ thống ngân hàng lưu trữ thông tin tập trung, hạn chế sai sót, hỗ trợ truy vấn nhanh và đảm bảo toàn vẹn dữ liệu.
- Bên cạnh đó, DBMS cho phép triển khai các chức năng nghiệp vụ phức tạp như kiểm tra số dư khi giao dịch, ghi nhận lịch sử giao dịch, quản lý tình trạng khoản vay, tính lãi tự động, phân quyền người dùng và đảm bảo an ninh dữ liệu. Nhờ đó, hệ thống quản lý ngân hàng hoạt động ổn định, an toàn và đáp ứng kịp thời nhu cầu tra cứu, thống kê và xử lý của nhân viên ngân hàng.
- Có thể khẳng định rằng việc ứng dụng cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu là giải pháp tối ưu trong việc xây dựng hệ thống quản lý ngân hàng, giúp nâng cao hiệu quả hoạt động, giảm thiểu rủi ro và hỗ trợ ra quyết định chính xác.

1.2. Công cụ và công nghệ sử dụng

1.2.1 Microsoft SQL Server

Microsoft SQL Server là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) do Microsoft phát triển, được sử dụng rộng rãi trong các hệ thống quản lý dữ liệu có quy mô từ nhỏ đến lớn. SQL Server cung cấp khả năng lưu trữ dữ liệu tập trung, xử lý truy vấn nhanh, hỗ trợ bảo mật mạnh mẽ và đảm bảo tính toàn vẹn của dữ liệu.

Trong đề tài quản lý ngân hàng, SQL Server được sử dụng để:

- Tạo và quản lý các bảng dữ liệu (Khách hàng, Tài khoản, Giao dịch, Khoản vay...).
- Xây dựng các mối quan hệ giữa dữ liệu.

- Viết truy vấn (SQL Query), thủ tục (Stored Procedure), trigger và các ràng buộc toàn vẹn.
- Đảm bảo hệ thống hoạt động ổn định, bảo mật và có khả năng mở rộng.

1.2.2 SQL Server Management Studio (SSMS)

SQL Server Management Studio là công cụ đồ họa hỗ trợ quản trị và phát triển trên SQL Server. SSMS cung cấp giao diện trực quan giúp người dùng dễ dàng thao tác với cơ sở dữ liệu mà không cần dùng hoàn toàn bằng dòng lệnh.

Trong dự án, SSMS được sử dụng để:

- Kết nối và quản trị SQL Server.
- Thực thi các câu lệnh SQL và kiểm tra kết quả.
- Thiết kế bảng, chỉnh sửa cột, tạo khóa chính/khóa ngoại.
- Theo dõi hoạt động hệ thống, sao lưu – phục hồi cơ sở dữ liệu.
- Quản lý phân quyền và bảo mật cho dữ liệu ngân hàng.

1.3 Bảng phân công

STT	Tên Công Việc	Người thực hiện	Ghi chú
1	Phân tích yêu cầu hệ thống	Lê Duy An	
2	Thiết kế CSDL	Lê Duy An	
3	Cài Đặt CSDL	Lê Duy An	
4	Các đối tượng CSDL nâng cao (index, views, thủ tục, function, trigger)	Nguyễn Thé Phương Nam	
5	Bảo mật và Quản trị CSDL (QL user- quyền truy cập, Bảo Mật, QSL)	Nguyễn Thé Phương Nam	

CHƯƠNG 2. PHÂN TÍCH YÊU CẦU HỆ THỐNG

2.1 Phân tích và yêu cầu cơ sở dữ liệu (Nghiệp vụ)

2.1.1 Nghiệp vụ Quản lý Thông tin (Data Management):

- Hệ thống phải lưu trữ thông tin chi tiết về **Khách hàng** (KHACHHANG).
- Hệ thống phải quản lý danh sách các **Chi nhánh** (CHINHANH) và **Nhân viên giao dịch** (NHANVIENGIAODICH) thuộc các chi nhánh đó.
- Hệ thống phải định nghĩa các loại hình dịch vụ, bao gồm **Loại tài khoản** (LOAITAIKHOAN - vd: Thanh toán, Tiết kiệm) và **Loại giao dịch** (LOAIGIAODICH - vd: Nạp, Rút, Chuyển khoản).

2.1.2 Nghiệp vụ Tài khoản (Account Operations):

- Mỗi **Tài khoản** (TAIKHOAN) phải được liên kết với một khách hàng, một chi nhánh và một loại tài khoản.
- Nghiệp vụ **Mở tài khoản** (sp_MoTaiKhoan trong proc.sql) phải cho phép tạo mới khách hàng (nếu chưa tồn tại) và một tài khoản liên kết với số dư ban đầu.

2.1.3 Nghiệp vụ Giao dịch (Transaction Operations):

- Đây là nghiệp vụ quan trọng nhất. Hệ thống phải đảm bảo **tính toàn vẹn dữ liệu** (ACID) khi thực hiện giao dịch, đặc biệt là **Chuyển tiền** (sp_ChuyenTien trong proc.sql).
 - Một nghiệp vụ chuyển tiền phải là một **Transaction** (sử dụng BEGIN TRAN, COMMIT, ROLLBACK):
 - o Phải kiểm tra số dư khả dụng của tài khoản nguồn.
 - o Trừ tiền tài khoản nguồn (UPDATE TAIKHOAN SET SoDu = SoDu - @SoTien).
 - o Cộng tiền tài khoản đích (UPDATE TAIKHOAN SET SoDu = SoDu + @SoTien).
 - o Ghi lại 2 bút toán (1 chuyển, 1 nhận) vào bảng GIAODICH.
 - o Nếu có bất kỳ lỗi nào (vd: sai tài khoản, không đủ tiền), toàn bộ thao tác phải được hủy bỏ (ROLLBACK).

2.1.4 Nghệp vụ Tra cứu & Báo cáo (Query & Reporting):

- Hệ thống phải cho phép người dùng (khách hàng hoặc nhân viên) **tra cứu thông tin** (từ functionuser.sql):
 - o Tra cứu lịch sử giao dịch theo mã khách hàng (fn_tracuugiaodichtheokh).
 - o Tra cứu giao dịch theo một khoảng thời gian (fn_TraCuuGiaoDichTheoNgay).
 - o Tra cứu nhanh tên khách hàng, số dư...

2.1.5 Nghệp vụ Bảo mật & Quản trị (Security & Admin):

- **Phân quyền (Authorization):** Hệ thống phải có cơ chế quản lý truy cập (từ QL_user_truycap.sql). Phải tạo được các LOGIN (đăng nhập) và USER (người dùng CSDL), gán họ vào các ROLE (vai trò) như 'nhanviennganhang' để cấp quyền (GRANT/DENY) cụ thể trên các bảng (vd: nhân viên chỉ được SELECT, INSERT vào GIAODICH nhưng không được DELETE).
- **Mã hóa (Encryption):** Dữ liệu nhạy cảm (như số dư) phải được bảo vệ. Hệ thống yêu cầu triển khai **Mã hóa dữ liệu trong suốt (TDE)** (từ bao_mat.sql) để mã hóa toàn bộ tệp dữ liệu (.mdf) và tệp log (.ldf).
- **Sao lưu (Backup):** Hệ thống phải có quy trình sao lưu (BACKUP DATABASE) và phục hồi (RESTORE) định kỳ để đảm bảo an toàn dữ liệu khi có sự cố (từ backup_restore.sql).

2.2 Các chức năng chính của hệ thống

2.2.1. Quản lý Khách hàng & Tài khoản:

- Chức năng Mở tài khoản (hỗ trợ bởi sp_MoTaiKhoan).
- Chức năng Cập nhật thông tin khách hàng.
- Chức năng Tra cứu thông tin tài khoản / số dư.

2.2.2. Quản lý Giao dịch:

- Chức năng **Chuyển tiền** (hỗ trợ bởi sp_ChuyenTien).

- Chức năng Nạp tiền (yêu cầu một thủ tục tương tự sp_ChuyenTien nhưng chỉ có 1 vé cộng tiền và ghi 1 log).
- Chức năng Rút tiền (tương tự Nạp tiền nhưng là trừ tiền).

2.2.3. Báo cáo và Tra cứu:

- Chức năng **Tra cứu lịch sử giao dịch** (hỗ trợ bởi fn_tracuugiaodichtheokh).
- Chức năng Lập báo cáo giao dịch theo ngày/tháng (hỗ trợ bởi fn_TraCuuGiaoDichTheoNgay).

2.2.4. Quản lý Người dùng (Nội bộ):

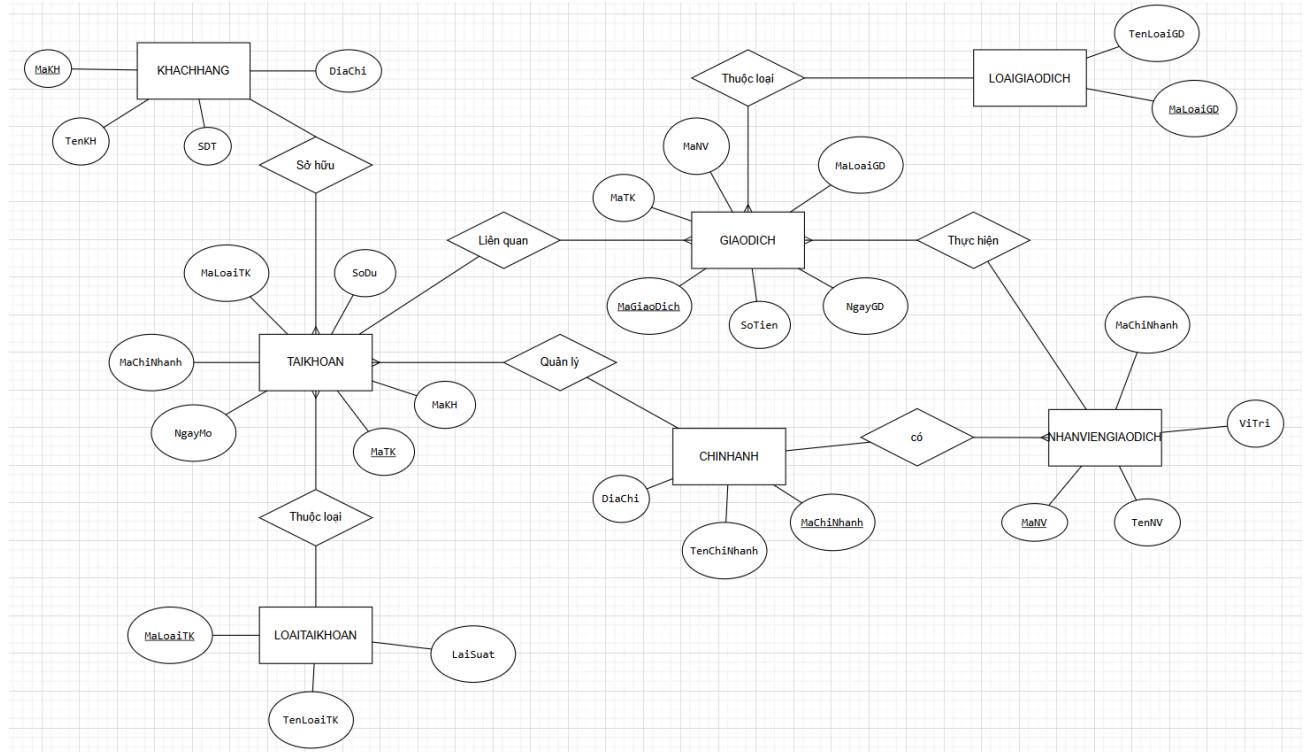
- Chức năng Tạo/Khóa tài khoản nhân viên (hỗ trợ bởi CREATE/DROP LOGIN).
- Chức năng Phân quyền cho nhân viên (hỗ trợ bởi GRANT/DENY và ROLE).
- Chức năng Quản lý danh sách nhân viên, chi nhánh.

2.2.5. Quản trị Hệ thống:

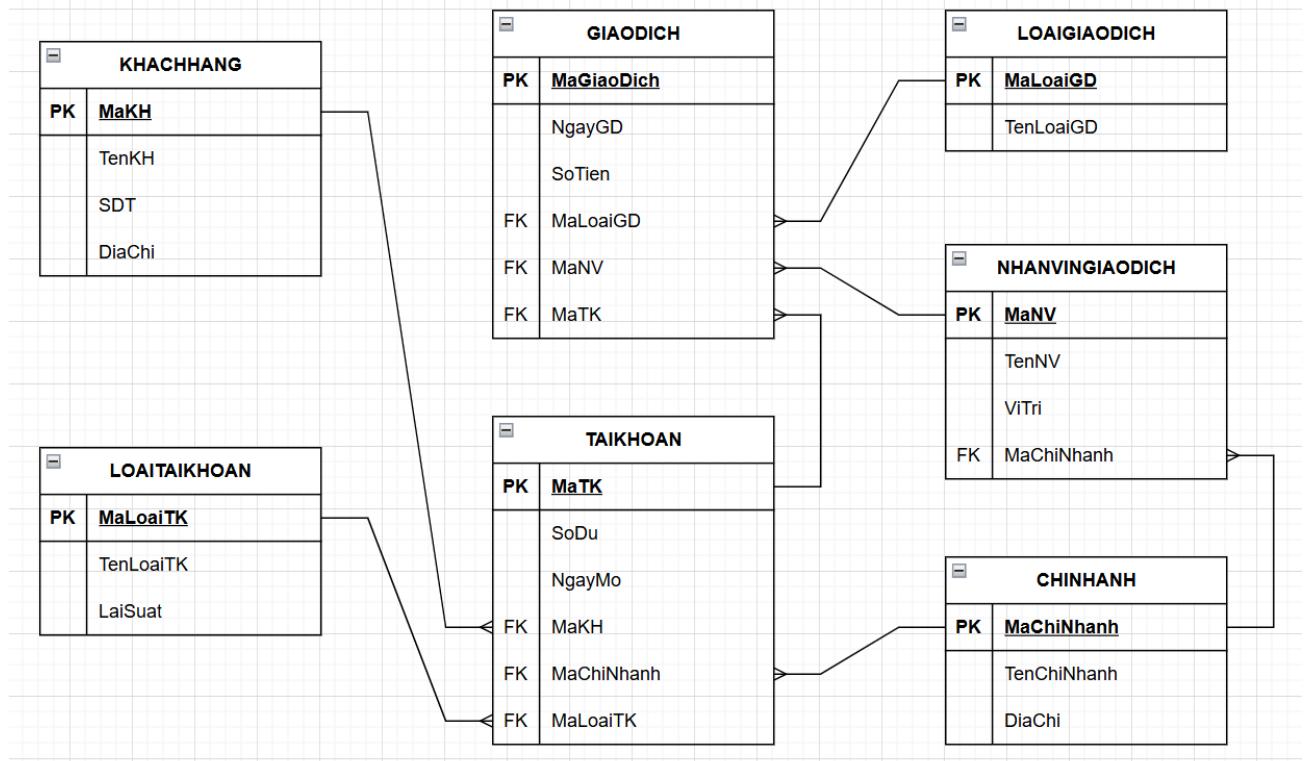
- Chức năng Sao lưu CSDL (hỗ trợ bởi BACKUP DATABASE).
- Chức năng Phục hồi CSDL (hỗ trợ bởi RESTORE DATABASE).
- Chức năng Kích hoạt / Giám sát mã hóa TDE (hỗ trợ bởi các lệnh trong bao_mat.sql).

CHƯƠNG 3. THIẾT KẾ

3.1 Thiết kế mô hình thực thể quan hệ (ER)



3.2 Chuyển đổi mô hình ER thành lược đồ quan hệ



3.3 Cấu trúc các bảng dữ liệu

Bảng CHINHANH

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaChiNhanh	VARCHAR(10)	PK, NOT NULL	Mã định danh duy nhất của chi nhánh.
2	TenChiNhanh	NVARCHAR(100)	NOT NULL	Tên chi nhánh.
3	DiaChi	NVARCHAR(255)	NONE	Địa chỉ của chi nhánh.

Bảng KHACHHANG

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaLoaiGD	VARCHAR(10)	PK, NOT NULL	Mã định danh loại giao dịch.
2	TenLoaiGD	NVARCHAR(100)	NOT NULL	Tên loại giao dịch.

Bảng LOAITAIKHOAN

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaNV	VARCHAR(10)	PK, NOT NULL	Mã định danh duy nhất của nhân viên.
2	TenNV	NVARCHAR(100)	NOT NULL	Tên của nhân viên.
3	ViTri	NVARCHAR(100)	NONE	Vị trí công việc (vd: Giao dịch viên).

Bảng LOAIGIAODICH

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaLoaiGD	VARCHAR(10)	PK, NOT NULL	Mã định danh loại giao dịch.
2	TenLoaiGD	NVARCHAR(100)	NOT NULL	Tên loại giao dịch.

Bảng NHANVIENGIAODICH

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaNV	VARCHAR(10)	PK, NOT NULL	Mã định danh duy nhất của nhân viên.
2	TenNV	NVARCHAR(100)	NOT NULL	Tên của nhân viên.
3	ViTri	NVARCHAR(100)	NULL	Vị trí công việc (vd: Giao dịch viên).
4	MaChiNhanh	VARCHAR(10)	FK (CHINHANH)	Mã chi nhánh nơi nhân viên làm việc.

Bảng TAIKHOAN

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaTK	VARCHAR(10)	PK, NOT NULL	Mã tài khoản (Số tài khoản) duy nhất.
2	SoDu	DECIMAL(18, 2)	NOT NULL, DEFAULT 0	Số dư hiện tại của tài khoản.
3	NgayMo	DATETIME	DEFAULT GETDATE()	Ngày mở tài khoản.
4	MaKH	VARCHAR(10)	FK (KHACHHANG)	Mã khách hàng sở hữu tài khoản này.
5	MaChiNhanh	VARCHAR(10)	FK (CHINHANH)	Mã chi nhánh quản lý tài khoản này.
6	MaLoaiTK	VARCHAR(10)	FK (LOAITAIKHOAN)	Mã loại tài khoản (Thanh toán hay Tiết kiệm).

Bảng GIAODICH

STT	Tên Trường	Kiểu Dữ liệu	Ràng buộc	Giải Thích
1	MaGiaoDich	VARCHAR(10)	PK, NOT NULL	Mã định danh duy nhất của giao dịch.
2	SoTien	DECIMAL(18, 2)	NULL	Số tiền của giao dịch.

3	NgayGD	DATETIME	DEFAULT GETDATE()	Thời gian thực hiện giao dịch.
4	MaTK	VARCHAR(10)	FK (TAIKHOAN)	Mã tài khoản liên quan đến giao dịch.
5	MaNV	VARCHAR(10)	FK (NHANVIENGIAODICH)	Mã nhân viên đã thực hiện giao dịch.
6	MaLoaiGD	VARCHAR(10)	FK (LOAIGIAODICH)	Mã loại giao dịch (Nạp, Rút, Chuyển...).

CHƯƠNG 4. CÀI ĐẶT CSDL

4.1 Tạo bảng

```
USE master;
GO
```

```
IF EXISTS (SELECT * FROM sysdatabases WHERE name = 'QLTHUVIEN')
    DROP DATABASE QLTHUVIEN;
GO
```

```
CREATE DATABASE QLTHUVIEN;
```

```
GO
```

```
USE QLTHUVIEN;
```

```
GO
```

-- 1. BẢNG THỂ LOẠI

```
CREATE TABLE THELOAI (
    MaTheLoai VARCHAR(20) PRIMARY KEY,
    TenTheLoai NVARCHAR(100) NOT NULL
);
```

-- 2. BẢNG TÁC GIẢ

```
CREATE TABLE TACGIA (
    MaTacGia VARCHAR(20) PRIMARY KEY,
    TenTacGia NVARCHAR(100) NOT NULL
);
```

-- 3. BẢNG SÁCH

```
CREATE TABLE SACH (
    MaSach VARCHAR(20) PRIMARY KEY,
    TenSach NVARCHAR(200) NOT NULL,
    MaTacGia VARCHAR(20) NOT NULL,
    NamXuatBan INT,
    SoLuong INT CHECK (SoLuong >= 0),
    MaTheLoai VARCHAR(20) NOT NULL,
    FOREIGN KEY (MaTacGia) REFERENCES TACGIA(MaTacGia),
    FOREIGN KEY (MaTheLoai) REFERENCES THELOAI(MaTheLoai)
);
```

-- 4. BẢNG ĐỘC GIẢ

```
CREATE TABLE DOCGIA (
    MaDocGia VARCHAR(20) PRIMARY KEY,
    TenDocGia NVARCHAR(100) NOT NULL,
    DiaChi NVARCHAR(200),
    SoDienThoai VARCHAR(15)
);
```

-- 5. BẢNG THỦ THƯ

```
CREATE TABLE THUTHU (
    MaThuThu VARCHAR(20) PRIMARY KEY,
    TenThuThu NVARCHAR(100) NOT NULL,
    SoDienThoai VARCHAR(15),
    Email VARCHAR(100) UNIQUE,
    TaiKhoan VARCHAR(50) NOT NULL UNIQUE,
    MatKhau NVARCHAR(255) NOT NULL
);
```

-- 6. BẢNG MUỢN SÁCH

```
CREATE TABLE MUONSACH (
    MaMuonSach VARCHAR(20) PRIMARY KEY,
    MaDocGia VARCHAR(20) NOT NULL,
    MaThuThu VARCHAR(20) NOT NULL,
    NgayMuon DATE NOT NULL DEFAULT GETDATE(),
    NgayTra DATE NULL,
    FOREIGN KEY (MaDocGia) REFERENCES DOCGIA(MaDocGia),
    FOREIGN KEY (MaThuThu) REFERENCES THUTHU(MaThuThu),
    CONSTRAINT CK_NgayTra CHECK (NgayTra IS NULL OR NgayTra >=
    NgayMuon)
);
```

-- 7. BẢNG CHI TIẾT MUỢN

```
CREATE TABLE CHITIETMUON (
    MaMuonSach VARCHAR(20) NOT NULL,
    MaSach VARCHAR(20) NOT NULL,
    SoLuongMuon INT CHECK (SoLuongMuon > 0),
    PRIMARY KEY (MaMuonSach, MaSach),
    FOREIGN KEY (MaMuonSach) REFERENCES MUONSACH(MaMuonSach),
    FOREIGN KEY (MaSach) REFERENCES SACH(MaSach)
);
GO
```

```
USE QLTHUVIEN;
GO
```

-- 1. THELOAI

```
INSERT INTO THELOAI (MaTheLoai, TenTheLoai)
VALUES
('TL01', N'Truyện'),
('TL02', N'Khoa học'),
('TL03', N'Lịch sử'),
('TL04', N'Thiếu nhi'),
('TL05', N'Văn học');
```

-- 2. TACGIA

```
INSERT INTO TACGIA (MaTacGia, TenTacGia)
VALUES
```

('TG01', N'Nguyễn Nhật Ánh'),
(‘TG02’, N’J.K. Rowling’),
(‘TG03’, N’Tô Hoài’),
(‘TG04’, N’George Orwell’),
(‘TG05’, N’Stephen Hawking’);

-- 3. SACH

INSERT INTO SACH (MaSach, TenSach, MaTacGia, NamXuatBan, SoLuong, MaTheLoai)

VALUES

(‘S01’, N’Mắt biếc’, ‘TG01’, 1990, 10, ‘TL05’),
(‘S02’, N’Harry Potter’, ‘TG02’, 2001, 15, ‘TL01’),
(‘S03’, N’Đé Mèn Phiêu Lưu Ký’, ‘TG03’, 1980, 8, ‘TL04’),
(‘S04’, N’1984’, ‘TG04’, 1949, 5, ‘TL05’),
(‘S05’, N’Lược sử thời gian’, ‘TG05’, 1988, 7, ‘TL02’);

-- 4. DOCGIA

INSERT INTO DOCGIA (MaDocGia, TenDocGia, DiaChi, SoDienThoai)

VALUES

(‘DG01’, N’Nguyễn An’, N’Hà Nội’, ‘0911223344’),
(‘DG02’, N’Lê Bình’, N’Ninh Bình’, ‘0988777666’),
(‘DG03’, N’Trần Cường’, N’Hải Phòng’, ‘0901234567’),
(‘DG04’, N’Phạm Dũng’, N’Hồ Chí Minh’, ‘0977555333’),
(‘DG05’, N’Hoàng Giang’, N’Huế’, ‘0933444555’);

-- 5. THUTHU

INSERT INTO THUTHU (MaThuThu, TenThuThu, SoDienThoai, Email, TaiKhoan, MatKhau)

VALUES

(‘TT01’, N’Nguyễn Xuân Kiên’, ‘0912345678’, ‘quan.nv@thuvien.com’, ‘admin’, ‘admin123’),
(‘TT02’, N’Nguyễn Thị Thảo Vân’, ‘0987654321’, ‘lanh.tt@thuvien.com’, ‘nhanvien1’, ‘nhanvien123’);

-- 6. MUONSACH

INSERT INTO MUONSACH (MaMuonSach, MaDocGia, MaThuThu, NgayMuon, NgayTra)

VALUES

(‘MS001’, ‘DG01’, ‘TT01’, ‘2025-11-01’, ‘2025-11-10’),
(‘MS002’, ‘DG02’, ‘TT02’, ‘2025-11-02’, NULL),
(‘MS003’, ‘DG03’, ‘TT01’, ‘2025-11-03’, ‘2025-11-07’),
(‘MS004’, ‘DG04’, ‘TT02’, ‘2025-11-04’, NULL),
(‘MS005’, ‘DG05’, ‘TT01’, ‘2025-11-05’, NULL);

-- 7. CHITIETMUON

INSERT INTO CHITIETMUON (MaMuonSach, MaSach, SoLuongMuon)

VALUES

(‘MS001’, ‘S01’, 1),

```

('MS001', 'S03', 2),
('MS002', 'S02', 1),
('MS003', 'S05', 1),
('MS004', 'S04', 1),
('MS005', 'S01', 1);
GO

```

```

SELECT * FROM THELOAI;
SELECT * FROM TACGIA;
SELECT * FROM SACH;
SELECT * FROM DOCGIA;
SELECT * FROM THUTHU;
SELECT * FROM MUONSACH;
SELECT * FROM CHITIETMUON;
GO

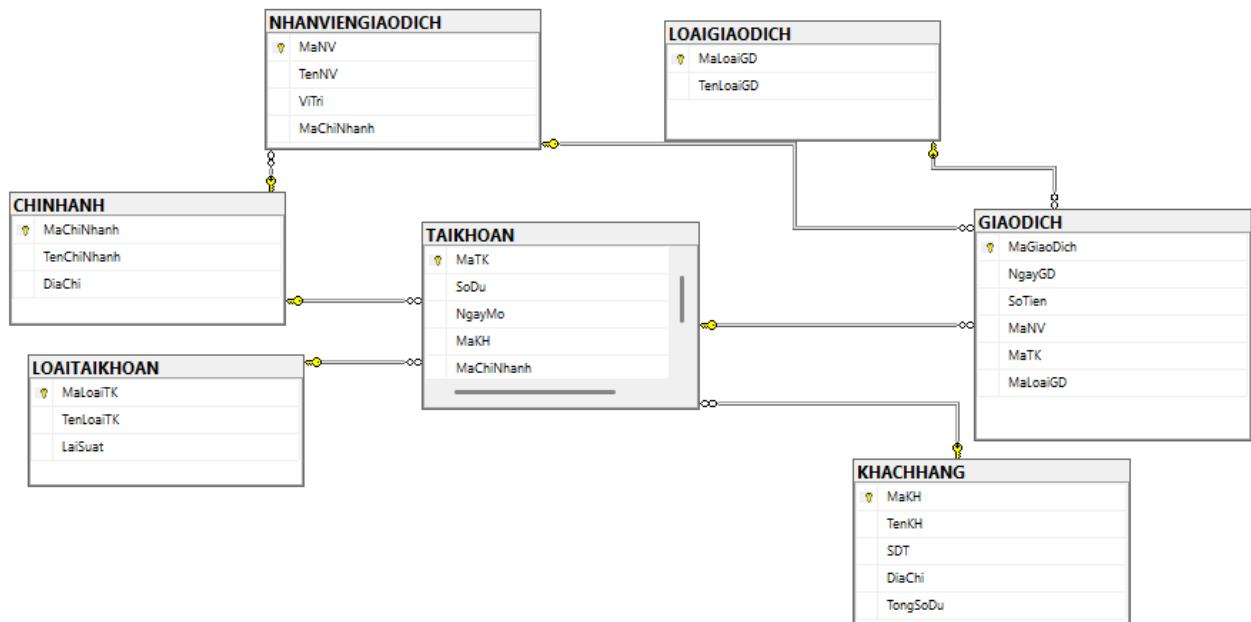
```

```

SELECT DG.TenDocGia, TT.TenThuThu, TS.TenSach, CTM.SoLuongMuon,
MS.NgayMuon, MS.NgayTra
FROM CHITIETMUON AS CTM
JOIN MUONSACH AS MS ON CTM.MaMuonSach = MS.MaMuonSach
JOIN DOCGIA AS DG ON MS.MaDocGia = DG.MaDocGia
JOIN SACH AS TS ON CTM.MaSach = TS.MaSach
JOIN THUTHU AS TT ON MS.MaThuThu = TT.MaThuThu;
GO

```

4.2 Sơ đồ csdl



CHƯƠNG 5. CÁC ĐỐI TƯỢNG CSDL NÂNG CAO

5.1 Cài đặt các chỉ mục (index)

-- Mục đích: Tăng tốc tìm kiếm hoặc join bảng TAIKHOAN dựa trên MaKhachHang (MaKH).

```
CREATE NONCLUSTERED INDEX IX_TAIKHOAN_MaKH  
ON TAIKHOAN (MaKH);
```

-- Mục đích: Tăng tốc tìm kiếm hoặc lọc các tài khoản theo MaChiNhanh.

```
CREATE NONCLUSTERED INDEX IX_TAIKHOAN_MaChiNhanh  
ON TAIKHOAN (MaChiNhanh);
```

-- Mục đích: Tăng tốc tìm kiếm lịch sử giao dịch của một MaTaiKhoan (MaTK) cụ thể.

```
CREATE NONCLUSTERED INDEX IX_GIAODICH_MaTK  
ON GIAODICH (MaTK);
```

-- Mục đích: Tăng tốc tìm kiếm các giao dịch được thực hiện bởi một MaNhanVien (MaNV).

```
CREATE NONCLUSTERED INDEX IX_GIAODICH_MaNV  
ON GIAODICH (MaNV);
```

-- Mục đích: Tăng tốc lọc, sắp xếp, hoặc làm báo cáo giao dịch theo NgayGiaoDich (NgayGD).

-- (Lưu ý: Tên index là 'MaLoaiGD' nhưng lại được tạo trên cột 'NgayGD').

```
CREATE NONCLUSTERED INDEX IX_GIAODICH_MaLoaiGD  
ON GIAODICH (NgayGD);
```

-- Mục đích: Tăng tốc đáng kể việc tìm kiếm khách hàng bằng SoDienThoai (SDT).

```
CREATE NONCLUSTERED INDEX IX_KHACHHANG_SDT  
ON KHACHHANG (SDT);
```

5.2 views

```
/*  
-- view lấy thông tin khách hàng - tài khoản  
    mục đích: tạo một bảng đơn giản cho nhân viên xem nhanh thông tin của khách hàng  
    và tài khoản họ sở hữu mà không cần join 4 bảng  
*/
```

```
create or alter view View_KHACHHANG as  
select  
    kh.TenKH,  
    kh.SDT,  
    tk.MaTK,  
    tk.SoDu,  
    ltk.TenLoaiTK,  
    ltk.LaiSuat,  
    cn.TenChiNhanh,  
    cn.DiaChi as DiaChiChiNhanh  
    from KHACHHANG as kh  
    inner join TAIKHOAN as tk  
        on kh.MaKH = tk.MaKH  
    inner join LOAITAIKHOAN as ltk  
        on tk.MaLoaiTK = ltk.MaLoaiTK  
    inner join CHINHANH as cn  
        on tk.MaChiNhanh = cn.MaChiNhanh
```

```
select * from View_KHACHHANG
```

```
/*  
view lịch sử giao dịch (đơn giản hóa & làm rõ nghĩa)  
mục đích: cung cấp một bản sao kê dễ đọc. Bảng GIAODICH gốc chỉ lưu các ma...,  
rất khó đọc
```

```

*/
create or alter view V_lichsugiaodich as
select
    gd.MaGiaoDich,
    gd.NgayGD,
    gd.SoTien,
    lgd.TenLoaiGD,
    gd.MaTK,
    nvgd.TenNV
    from GIAODICH as gd
    left join LOAIGIAODICH as lgd
        on gd.MaLoaiGD = lgd.MaLoaiGD
    left join NHANVIENGIAODICH as nvgd
        on gd.MaGiaoDich = nvgd.MaChiNhanh

select * from V_lichsugiaodich

/*
View Báo cáo Tổng hợp (Dùng cho Indexed View)
Mục đích: Tạo một báo cáo tổng hợp (dùng cho sếp hoặc quản lý) xem hiệu
suất của các chi nhánh.
*/
create or alter view V_BaoCao_TaiKhoan_ChiNhanh
with SCHEMABINDING
as
select
    tk.MaChiNhanh,
    COUNT_BIG(*) as tongsotk,
    sum(isnull(tk.SoDu, 0)) as tongsodu
    from [dbo].[TAIKHOAN] as tk
    group by
        tk.MaChiNhanh
go

create unique clustered index IDX_V_BaoCao_TaiKhoan_ChiNhanh
on V_BaoCao_TaiKhoan_ChiNhanh (MaChiNhanh);
go

select * from V_BaoCao_TaiKhoan_ChiNhanh

/*
View Bảo mật (Che giấu dữ liệu nhạy cảm)
Mục đích: Cung cấp cho một bộ phận (ví dụ: marketing hoặc phân tích) quyền xem dữ liệu khách hàng nhưng ẩn
đi thông tin cá nhân nhạy cảm
*/
CREATE VIEW V_ThongTinKhachHang_BaoMat AS
SELECT
    kh.MaKH,
    -- Giả sử TenKH là "Nguyễn Văn An", hàm này sẽ lấy "An"
    -- Nếu bạn muốn lấy họ "Nguyễn", dùng: LEFT(kh.TenKH, CHARINDEX(' ', kh.TenKH + ' ') - 1)
    -- RIGHT(kh.TenKH, CHARINDEX(' ', REVERSE(kh.TenKH) + ' ') - 1) AS TenKhachHang,
    tk.MaLoaiTK,
    tk.SoDu
FROM
    KHACHHANG AS kh
JOIN
    TAIKHOAN AS tk ON kh.MaKH = tk.MaKH;

select * from V_ThongTinKhachHang_BaoMat

```

5.3 Thủ tục (Stored Procedures)

```
/*
lab1: Chuyển Tiền
    -- Yêu cầu: Phải đảm bảo tiền được trừ ở tài khoản A VÀ cộng vào tài khoản B. Nếu một
                trong hai bước thất bại, cả giao dịch phải được hủy (ROLLBACK).
    -- Logic: Đây là một TRANSACTION kinh điển, nhận vào MaTKNguon, MaTKDich, SoTien

*/
CREATE OR ALTER PROCEDURE sp_ChuyenTien
    @MaTKNguon VARCHAR(20),
    @MaTKDich VARCHAR(20),
    @SoTien DECIMAL(18,2),
    @MaNV VARCHAR(10),
    @MaLoaiGD_Chuyen VARCHAR(10),
    @MaLoaiGD_Nhan VARCHAR(10),
    @TrangThai INT OUTPUT,
    @ThongBao NVARCHAR(4000) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    SET @TrangThai = 0;
    SET @ThongBao = N'Lỗi không xác định';

    BEGIN TRY
        -- 1. Kiểm tra đầu vào
        IF @SoTien <= 0
        BEGIN
            SET @ThongBao = N'Số tiền phải lớn hơn 0';
            RETURN;
        END;

        IF @MaTKNguon = @MaTKDich
        BEGIN
            SET @ThongBao = N'Tài khoản nguồn phải khác tài khoản đích';
            RETURN;
        END;

        IF NOT EXISTS (SELECT 1 FROM TAIKHOAN WHERE MaTK = @MaTKNguon)
        BEGIN
            SET @ThongBao = N'Tài khoản nguồn không tồn tại.';
            RETURN;
        END;

        IF NOT EXISTS (SELECT 1 FROM TAIKHOAN WHERE MaTK = @MaTKDich)
        BEGIN
            SET @ThongBao = N'Tài khoản đích không tồn tại.';
            RETURN;
        END;

        -- 2. Bắt đầu transaction
        BEGIN TRANSACTION;

        -- Lấy khóa ứng dụng để tránh race khi sinh MaGiaoDich
        DECLARE @rc INT;
        EXEC @rc = sp_getapplock @Resource = N'GIAODICH_ID_GENERATION',
            @LockMode = N'Exclusive',
            @LockTimeout = 10000,
            @LockOwner = N'Transaction';

        IF @rc < 0
        BEGIN
            ROLLBACK TRANSACTION;
            SET @ThongBao = N'Không thể lấy khóa để sinh mã giao dịch. Vui lòng thử lại.';

    END;
```

```

        RETURN;
    END;

DECLARE @SoDuNguon DECIMAL(18,2);

SELECT @SoDuNguon = SoDu
FROM TAIKHOAN WITH (UPDLOCK, ROWLOCK)
WHERE MaTK = @MaTKNguon;

IF @SoDuNguon < @SoTien
BEGIN
    ROLLBACK TRANSACTION;
    SET @ThongBao = N'Số dư tài khoản nguồn không đủ';
    RETURN;
END;

-- Cập nhật số dư
UPDATE TAIKHOAN
SET SoDu = SoDu - @SoTien
WHERE MaTK = @MaTKNguon;

IF @@ROWCOUNT = 0
BEGIN
    ROLLBACK TRANSACTION;
    SET @ThongBao = N'Lỗi khi trừ tiền tài khoản nguồn';
    RETURN;
END;

UPDATE TAIKHOAN
SET SoDu = SoDu + @SoTien
WHERE MaTK = @MaTKDich;

IF @@ROWCOUNT = 0
BEGIN
    ROLLBACK TRANSACTION;
    SET @ThongBao = N'Lỗi khi cộng tiền tài khoản đích';
    RETURN;
END;

-- Sinh MaGiaoDich (dạng GD00001, GD00002, ...)
DECLARE @MaxSuffix INT = 0;
SELECT @MaxSuffix = ISNULL(MAX(TRY_CAST(SUBSTRING(MaGiaoDich,3, 18) AS INT)), 0)
FROM GIAODICH WITH (TABLOCK); -- giữ khóa đọc toàn bảng trong transaction để an toàn hơn

DECLARE @Id1 INT = @MaxSuffix + 1;
DECLARE @Id2 INT = @MaxSuffix + 2;

DECLARE @MaGD_Chuyen VARCHAR(20) = 'GD' + RIGHT('00000' + CAST(@Id1 AS VARCHAR(10)), 5);
DECLARE @MaGD_Nhan  VARCHAR(20) = 'GD' + RIGHT('00000' + CAST(@Id2 AS VARCHAR(10)), 5);

DECLARE @Now DATETIME = GETDATE();

-- Ghi log giao dịch (ghi rõ MaGiaoDich)
INSERT INTO GIAODICH (MaGiaoDich, NgayGD, SoTien, MaNV, MaTK, MaLoaiGD)
VALUES
    (@MaGD_Chuyen, @Now, @SoTien, @MaNV, @MaTKNguon, @MaLoaiGD_Chuyen);

IF @@ROWCOUNT = 0
BEGIN
    ROLLBACK TRANSACTION;
    SET @ThongBao = N'Lỗi khi ghi giao dịch chuyển';
    RETURN;
END;

INSERT INTO GIAODICH (MaGiaoDich, NgayGD, SoTien, MaNV, MaTK, MaLoaiGD)
VALUES

```

```

(@MaGD_Nhan, @Now, @SoTien, @MaNV, @MaTKDich, @MaLoaiGD_Nhan);

IF @@ROWCOUNT = 0
BEGIN
    ROLLBACK TRANSACTION;
    SET @ThongBao = N'Lỗi khi ghi giao dịch nhận';
    RETURN;
END;

-- Commit và trả kết quả
COMMIT TRANSACTION;

SET @TrangThai = 1;
SET @ThongBao = N'Chuyển tiền thành công. Giao dịch chuyển #' +
    + @MaGD_Chuyen +
    + N', nhận #' +
    + @MaGD_Nhan + N':';

END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    SET @TrangThai = 0;
    SET @ThongBao = N'Lỗi hệ thống: ' + ERROR_MESSAGE();
END CATCH
END;
GO

DECLARE @trangthai INT, @thongbao NVARCHAR(4000);

EXEC sp_ChuyenTien
    @MaTKGuon = 'TK1002',
    @MaTKDich = 'TK1003',
    @SoTien = 200000,
    @MaNV = 'NV001',
    @MaLoaiGD_Chuyen = 'CKN', -- hoặc mã loại phù hợp (VARCHAR)
    @MaLoaiGD_Nhan = 'NT',
    @TrangThai = @trangthai OUTPUT,
    @ThongBao = @thongbao OUTPUT;

SELECT @trangthai AS TrangThai, @thongbao AS ThongBao;

```

```

/*
lab2 Rút tiền:
Yêu cầu: Phải kiểm tra số dư trong TAIKHOAN trước khi rút. Nếu đủ tiền,
trừ tiền (UPDATE TAIKHOAN) và ghi lại lịch sử (INSERT GIAODICH).

Logic: Đây cũng là một TRANSACTION để đảm bảo 2 hành động xảy ra cùng lúc.
*/

```

```

CREATE OR ALTER PROCEDURE sp_RutTien
    @MaTK NVARCHAR(10),
    @SoTien DECIMAL(18,2),
    @MaNV VARCHAR(10),
    @MaLoaiGD INT,      -- mã loại giao dịch rút tiền
    @TrangThai INT OUT,
    @ThongBao NVARCHAR(4000) OUT
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    SET @TrangThai = 0;

```

```

SET @ThongBao = N'Lỗi không xác định.';

BEGIN TRY
    -- 1 Kiểm tra dữ liệu đầu vào
    IF @SoTien <= 0
    BEGIN
        SET @ThongBao = N'Số tiền phải lớn hơn 0';
        RETURN;
    END

    IF NOT EXISTS (SELECT 1 FROM TAIKHOAN WHERE MaTK = @MaTK)
    BEGIN
        SET @ThongBao = N'Tài khoản không tồn tại';
        RETURN;
    END

    DECLARE @SoDu DECIMAL(18,2);
    SELECT @SoDu = SoDu
    FROM TAIKHOAN WITH (UPDLOCK, ROWLOCK)
    WHERE MaTK = @MaTK;

    IF @SoDu < @SoTien
    BEGIN
        SET @ThongBao = N'Số dư không đủ để rút.';
        RETURN;
    END

    -- 2 Bắt đầu transaction
    BEGIN TRANSACTION;

    -- 3 Trù tiền
    UPDATE TAIKHOAN
    SET SoDu = SoDu - @SoTien
    WHERE MaTK = @MaTK;

    IF @@ROWCOUNT = 0
    BEGIN
        ROLLBACK TRANSACTION;
        SET @ThongBao = N'Lỗi khi trù tiền tài khoản.';
        RETURN;
    END

    -- 4 Sinh MaGiaoDich thủ công
    DECLARE @MaxSuffix INT;
    SELECT @MaxSuffix = ISNULL(MAX(TRY_CAST(SUBSTRING(MaGiaoDich,3,10) AS INT)),0)
    FROM GIAODICH WITH (TABLOCK); -- khóa toàn bảng để tránh trùng

    DECLARE @MaGiaoDich VARCHAR(20);
    SET @MaGiaoDich = 'GD' + RIGHT('00000' + CAST(@MaxSuffix + 1 AS VARCHAR(10)), 5);

    -- 5 Ghi lịch sử giao dịch
    INSERT INTO GIAODICH(MaGiaoDich, MaTK, MaLoaiGD, SoTien, MaNV, NgayGD)
    VALUES (@MaGiaoDich, @MaTK, CAST(@MaLoaiGD AS VARCHAR(10)), @SoTien, @MaNV,
    GETDATE());

    IF @@ROWCOUNT = 0
    BEGIN
        ROLLBACK TRANSACTION;
        SET @ThongBao = N'Lỗi khi ghi lịch sử giao dịch.';
        RETURN;
    END

    -- 6 Commit transaction
    COMMIT TRANSACTION;

    SET @TrangThai = 1;

```

```

SET @ThongBao = N'Rút tiền thành công. Mã GD: ' + @MaGiaoDich;

END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    SET @TrangThai = 0;
    SET @ThongBao = N'Lỗi hệ thống: ' + ERROR_MESSAGE();
END CATCH
END
GO

declare @trangthai int, @thongbao Nvarchar(4000)
exec sp_RutTien
    @MaTk = 'TK1002',
    @sotien = 500000,
    @MaNV = 'NV001',
    @MaLoaiGD = 1,
    @TrangThai = @trangthai out,
    @ThongBao = @thongbao out

```

```
select @trangthai as TrangThai, @thongbao as ThongBao
```

```
/*
lab 3: Tạo Khách hàng & Mở Tài khoản:
```

Yêu cầu: Nhân viên cần một chức năng "Mở tài khoản" duy nhất.

Logic: SP này nhận vào TenKH, SDT, MaLoaiTK... Nó sẽ tự kiểm tra SDT đã tồn tại chưa.
Nếu chưa, INSERT vào KHACHHANG trước, sau đó INSERT vào TAIKHOAN.

```
*/
```

```

CREATE OR ALTER PROCEDURE sp_MoTaiKhoan
    @TenKH NVARCHAR(100),
    @SDT VARCHAR(15),
    @MaLoaiTK INT,
    @SoDuBanDau DECIMAL(18,2),
    @MaNV VARCHAR(10),
    @TrangThai INT OUT,
    @ThongBao NVARCHAR(4000) OUT
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @MaKH CHAR(10);
    DECLARE @MaTK CHAR(10);

    SET @TrangThai = 0;
    SET @ThongBao = N'Lỗi chưa xác định.';

    BEGIN TRY
        BEGIN TRANSACTION;

        -- Kiểm tra KH đã tồn tại chưa
        SELECT @MaKH = MaKH
        FROM KHACHHANG WITH (UPDLOCK)
        WHERE SDT = @SDT;

        -- Nếu chưa có -> tạo mới KH
        IF @MaKH IS NULL
        BEGIN
            DECLARE @MaxKH INT = ISNULL(
                (SELECT MAX(TRY_CAST(SUBSTRING(MaKH,3,10) AS INT)) FROM KHACHHANG WITH
                (TABLOCK)),
```

```

0
);
SET @MaKH = 'KH' + RIGHT('00000' + CAST(@MaxKH + 1 AS VARCHAR(10)), 5);

INSERT INTO KHACHHANG(MaKH, TenKH, SDT)
VALUES (@MaKH, @TenKH, @SDT);
END

-- 3 Sinh mã tài khoản mới
DECLARE @MaxTK INT = ISNULL(
    (SELECT MAX(TRY_CAST(SUBSTRING(MaTK,3,10) AS INT)) FROM TAIKHOAN WITH (TABLOCK)),
    0
);
SET @MaTK = 'TK' + RIGHT('00000' + CAST(@MaxTK + 1 AS VARCHAR(10)), 5);

-- 4 Tạo tài khoản
INSERT INTO TAIKHOAN(MaTK, MaKH, MaLoaiTK, SoDu, NgayMo)
VALUES (@MaTK, @MaKH, @MaLoaiTK, @SoDuBanDau, GETDATE());

COMMIT TRANSACTION;

SET @TrangThai = 1;
SET @ThongBao = N'Mở tài khoản thành công. Mã TK: ' + @MaTK + N', Mã KH: ' + @MaKH;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    SET @TrangThai = 0;
    SET @ThongBao = N'Lỗi hệ thống: ' + ERROR_MESSAGE();
END CATCH
END
GO

DECLARE @TrangThai INT, @ThongBao NVARCHAR(4000);
EXEC sp_MoTaiKhoan
    @TenKH = N'Nguyễn Văn A',
    @SDT = '0987123976',
    @MaLoaiTK = 1,
    @SoDuBanDau = 1000000,
    @MaNV = 'NV001',
    @TrangThai = @TrangThai OUTPUT,
    @ThongBao = @ThongBao OUTPUT;
SELECT @TrangThai AS TrangThai, @ThongBao AS ThongBao;

```

5.4 Hàm người dùng tự định nghĩa (User Functions)

```

use QLNH
/*
function user
*/

```

```
/*
```

lab 1: Lấy Tên Khách hàng (Scalar Function):

Yêu cầu: Cần một cách nhanh để lấy TenKH chỉ từ MaKH.

Logic: Hàm nhận vào MaKH, trả về TenKH (kiểu NVARCHAR). Dùng trong các câu SELECT để làm báo cáo dễ đọc hơn.

```

*/
create or alter function fn_laytenkh
(
    @makh varchar(10)

```

```

)
returns nvarchar(100)
as
begin
    declare @tenkh nvarchar(100);

    select @tenkh = TenKH
        from KHACHHANG
        where MaKH = @makh

    return @tenkh;
end;

```

```

-- cách gọi 1
select dbo.fn_laytenkh('KH001') as tenkh

```

```

/*
lab 2: Tính Lãi suất thực (Scalar Function):

```

Yêu cầu: Cần một hàm để tính số tiền lãi dự kiến cho một tài khoản.

Logic: Hàm nhận vào MaTK, trả về SoTienLai (kiểu DECIMAL). Bên trong, nó SELECT SoDu * LaiSuat / 100.

```
*/
```

```

create or alter function fn_tinhlaixuat
(
    @matk varchar(10)
)
returns decimal(18, 2)
as
begin
    declare @sotienlai decimal(18,2)

    select
        @sotienlai = (tk.SoDu * ltk.LaiSuat / 100)
        from TAIKHOAN tk
        inner join LOAITAIKHOAN ltk
            on tk.MaLoaiTK = ltk.MaLoaiTK
            where tk.MaTK = MaTK
    return @sotienlai
end

```

```
select dbo.fn_tinhlaixuat('TK003') as tienlaitk
```

```
/*
```

lab 3: Lấy Trạng thái Tài khoản (Scalar Function):

Yêu cầu: Cần biết tài khoản "Bình thường", "Bị âm tiền" hay "Sắp hết hạn".

Logic: Hàm nhận vào MaTK, dùng CASE WHEN trên SoDu và NgayMo để trả về một chuỗi trạng thái (ví dụ: 'Hoạt động').

```
*/
```

```

create or alter function fn_laytrangthaitk
(
    @matk varchar(10)
)
returns nvarchar(100)
as
begin
    declare @trangthai nvarchar(100);

    select
        @trangthai = case

```

```

        when SoDu < 0 then 'đang nợ'
        when DATEDIFF(YEAR, NgayMo, GETDATE()) > 5 then N'tài khoản cũ sắp hết hạn'
        else N' bình thường'
    end
    from TAIKHOAN
    where MaTK = @matk
    return @trangthai
end

```

select dbo.fn_laytrangthaithit('TK002') as trangthai

/*

lab 4: Lấy tất cả Tài khoản của Khách (Inline Table-Valued Function):

Yêu cầu: Cần một cách để lấy tất cả tài khoản của một khách hàng (giống như một View có tham số).

Logic: Hàm nhận vào MaKH, trả về TABLE (kết quả của SELECT * FROM TAIKHOAN WHERE MaKH = @MaKH).

```

*/
create or alter function fn_laytkhach
(
    @makh varchar(10)
)
returns table
as
return
(
    select
    *
    from TAIKHOAN
    where MaKH = @makh
)

```

select * from fn_laytkhach('KH001');

/*

lab 5: Chuẩn hóa Tên (Scalar Function):

Yêu cầu: Dữ liệu nhập vào có thể là "nguyễn văn AN". Cần chuẩn hóa thành "Nguyễn Văn An".

Logic: Hàm nhận vào một chuỗi Ten, trả về chuỗi đã được chuẩn hóa.

```

*/
CREATE OR ALTER FUNCTION fn_chuanhoa
(
    @ten NVARCHAR(100)
)
RETURNS NVARCHAR(100)
AS
BEGIN
    DECLARE
        @tenchuanhoa NVARCHAR(100) = N"",
        @i INT = 1,
        @len INT,
        @ch NVARCHAR(1),
        @upcoming BIT = 1;

    -- Loại bỏ khoảng trắng thừa đầu cuối
    SET @ten = LTRIM(RTRIM(@ten));
    SET @len = LEN(@ten);

```

```

    WHILE @i <= @len
    BEGIN
        SET @ch = SUBSTRING(@ten, @i, 1);

```

```

IF @ch = N' '
BEGIN
    SET @tenchuanhoa += @ch;
    SET @upcoming = 1; -- sau khoảng trắng, chữ cái kế tiếp cần viết hoa
END
ELSE
BEGIN
    IF @upcoming = 1
        SET @tenchuanhoa += UPPER(@ch);
    ELSE
        SET @tenchuanhoa += LOWER(@ch);
    SET @upcoming = 0;
END

SET @i += 1;
END

RETURN @tenchuanhoa;
END;
GO
SELECT dbo.fn_chuanhoa(N'nguyễn văn AN') AS TenChuanHoa;

```

/*

Đề Lab 6 – Multi-statement Table-Valued Function

Đề bài:
 Viết hàm fn_TraCuuGiaoDichTheoKH nhận vào mã khách hàng (@MaKH)
 và trả về bảng gồm các giao dịch của khách đó,
 bao gồm:

Mã giao dịch
 Ngày giao dịch
 Loại giao dịch
 Số tiền
 Mã tài khoản
 Tên nhân viên thực hiện

Hàm phải:

Có thể trả về nhiều dòng.

Viết theo kiểu đa câu lệnh (multi-statement), nghĩa là phải dùng biến bảng (@result) trong thân hàm.

*/

```

create or alter function fn_tracuuugiaodichtheokh
(
    @makh varchar(10)
)
returns @Result table
(
    MaGiaoDich varchar(10),
    NgayGD datetime,
    TenLoaiGD nvarchar(100),
    SoTien decimal(18,2),
    MaTK varchar(10),
    TenNV nvarchar(100)
)
as
begin
    insert into @Result
        SELECT
            gd.MaGiaoDich,

```

```

gd.NgayGD,
lgd.TenLoaiGD,
gd.SoTien,
tk.MaTK,
nv.TenNV
FROM GIAODICH gd
INNER JOIN TAIKHOAN tk ON gd.MaTK = tk.MaTK
INNER JOIN LOAIGIAODICH lgd ON gd.MaLoaiGD = lgd.MaLoaiGD
LEFT JOIN NHANVIENGIAODICH nv ON gd.MaNV = nv.MaNV
WHERE tk.MaKH = @MaKH;
return;
end

select * from dbo.fn_tracuugiaodichtheokh('KH001')

/*
=====
§§ Lab 7 — Tra cứu giao dịch theo khoảng ngày
█ Đề bài:
Viết hàm fn_TraCuuGiaoDichTheoNgay nhận vào:

    @TuNgay DATE
    @DenNgay DATE

Trả về danh sách giao dịch trong khoảng thời gian đó, gồm:
MaGiaoDich, TenLoaiGD, SoTien, NgayGD, TenKH, TenNV, TenChiNhanh.

*/
create or alter function fn_TraCuuGiaoDichTheoNgay
(
    @tungay date,
    @dengay date
)
returns @result table
(
    MaGiaoDich varchar(10),
    TenLoaiGD nvarchar(100),
    SoTien decimal(18,2),
    NgayGD datetime,
    TenKH Nvarchar(100),
    TenNV nvarchar(100),
    TenChiNhanh nvarchar(100)
)
as
begin
    insert into @result
    SELECT
        gd.MaGiaoDich,
        lgd.TenLoaiGD,
        gd.SoTien,
        gd.NgayGD,
        kh.TenKH,
        nv.TenNV,
        cn.TenChiNhanh
    FROM GIAODICH gd
    INNER JOIN TAIKHOAN tk ON gd.MaTK = tk.MaTK
    INNER JOIN KHACHHANG kh ON tk.MaKH = kh.MaKH
    LEFT JOIN NHANVIENGIAODICH nv ON gd.MaNV = nv.MaNV
    LEFT JOIN CHINHANH cn ON tk.MaChiNhanh = cn.MaChiNhanh
    INNER JOIN LOAIGIAODICH lgd ON gd.MaLoaiGD = lgd.MaLoaiGD
    WHERE CAST(gd.NgayGD AS DATE) BETWEEN @TuNgay AND @DenNgay;
    return;
end

select * from fn_TraCuuGiaoDichTheoNgay('2023-10-20', '2023-10-25')

```

5.5 trigger

```
/*
```

Ghi Log thay đổi SDT (Auditing):

Yêu cầu: Phải ghi lại dấu vết mỗi khi ai đó thay đổi SDT của khách hàng vì lý do bảo mật.

Logic: Tạo AFTER UPDATE Trigger trên bảng KHACHHANG. Nếu cột SDT bị thay đổi, INSERT vào bảng Log_ThayDoi (chứa MaKH, SDTCu, SDTMoi, NguoiSua, NgaySua).

```
*/
```

```
-- tạo bảng log để ghi vào
```

```
CREATE TABLE Log_ThayDoi (
    MaLog INT IDENTITY PRIMARY KEY,
    MaKH varchar(10),
    SDTCu NVARCHAR(15),
    SDTMoi NVARCHAR(15),
    NguoiSua NVARCHAR(100),
    NgaySua DATETIME
);
```

```
create or alter trigger trg_DML_thaysdt
on KHACHHANG
after update
as
begin
    set nocount on;

    if update(SDT)
    begin
        insert into Log_ThayDoi (MaKH, SDTCu, SDTMoi, NguoiSua, NgaySua)
        select
            d.MaKH,
            d.SDT as sdt_cu,
            i.SDT as sdt_moi,
            SUSER_SNAME() as nguoisua,
            GETDATE() as ngaysua
        from deleted d
        inner join inserted i on d.MaKH = i.MaKH
        where d.SDT <> i.SDT
    end
end

select * from KHACHHANG

update KHACHHANG
set SDT = '092548452'
where MaKH = 'KH001'

select * from Log_ThayDoi

SELECT kh.MaKH, kh.TenKH, kh.SDT, log.SDTCu, log.SDTMoi, log.NgaySua
FROM KHACHHANG kh
LEFT JOIN Log_ThayDoi log ON kh.MaKH = log.MaKH
ORDER BY log.NgaySua DESC;

SELECT * FROM sys.triggers;
sp_helptext 'trg_DML_ThaySDT';
```

```
/*
```

Ngăn chặn xóa Tài khoản còn tiền:

Yêu cầu: Không ai được phép DELETE một TAIKHOAN nếu SoDu của nó vẫn > 0.

Logic: Tạo FOR DELETE Trigger trên bảng TAIKHOAN. Kiểm tra SoDu của hàng đang bị xóa (từ bảng deleted), nếu > 0, báo lỗi RAISERROR và ROLLBACK.

*/

```
create or alter trigger trg_nganxoatk
on TAIKHOAN
after delete
as
begin
    set nocount on;
    if exists (
        select 1
        from deleted
        where SoDu > 0
    )
    begin
        -- báo lỗi và hủy thao tác
        raiserror('không được phép xóa tài khoản còn tiền!', 16, 1);
        rollback transaction
        return;
    end
end

select * from TAIKHOAN
delete from TAIKHOAN
where MaTK = 'TK1001'
```

/*

Ngăn giao dịch âm:

Yêu cầu: Đảm bảo không ai có thể INSERT một GIAODICH với SoTien là số âm.

Logic: Tạo FOR INSERT Trigger trên bảng GIAODICH. Kiểm tra SoTien của hàng vừa chèn (từ bảng inserted), nếu < 0, báo lỗi và ROLLBACK.

*/

```
create or alter trigger trg_checkgiaodich
on GIAODICH
for insert
as
begin
    set nocount on;

    if exists (
        select 1
        from inserted
        where SoTien < 0
    )
    begin
        raiserror('không được phép chèn giao dịch có số tiền âm', 16, 1);
        rollback transaction
        return;
    end
end

-- thử chèn giao dịch âm
INSERT INTO GIAODICH (MaGiaoDich, NgayGD, SoTien, MaNV, MaTK, MaLoaiGD)
VALUES ('GD00007', GETDATE(), 1000.00, 'NV002', 'TK1002', 'RT');

select * from GIAODICH
```

/*

Cập nhật TongSoDu (Denormalization):

Yêu cầu: Bảng KHACHHANG có một cột TongSoDu (tổng tiền của tất cả tài khoản). Cột này phải tự cập nhật.

Logic: Tao AFTER INSERT, UPDATE, DELETE Trigger trên bảng TAIKHOAN. Mỗi khi SoDu thay đổi, tính lại tổng SoDu của MaKH bị ảnh hưởng và UPDATE vào bảng KHACHHANG.

*/

```
ALTER TABLE KHACHHANG
ADD TongSoDu DECIMAL(18,2) DEFAULT 0;
```

```
create or alter trigger trg_UpdateTongSoDu
on TAIKHOAN
after insert, update, delete
as
begin
    set nocount on;

    declare @AffectedKH table (MaKH varchar(10) primary key);

    insert into @AffectedKH(MaKH)
    select distinct MaKH
    from inserted
    where MaKH is not null;

    insert into @AffectedKH(MaKH)
    select distinct MaKH
    from deleted
    where MaKH is not null
    and MaKH not in (select MaKH from @AffectedKH)

    update kh
    set TongSoDu = ISNULL(t.SoDuTong, 0)
    from KHACHHANG kh
    inner join (
        select MaKH, SUM(SoDu) as sodutong
        from TAIKHOAN
        where MaKH in (select MaKH from @AffectedKH)
        group by MaKH
    )t on kh.MaKH = t.MaKH

end

UPDATE TAIKHOAN
SET SoDu = 3000
WHERE MaTK = 'TK001';
SELECT MaKH, TongSoDu FROM KHACHHANG;
```

CHƯƠNG 6. BẢO MẬT VÀ QUẢN TRỊ CSDL

6.1 Quản lý người dùng và quyền quản lý

```
-- lệnh tạo login
create login develop
    with password = '1',          -- must_change nếu muốn đổi mật khẩu khi đăng nhập
    default_database = QLNH,    -- chỉ định database mặc định
    check_policy = off          -- tă kiểm tra chính sách bảo mật (chỉ nên dùng khi test)
go

create login phunnam
    with password = '2',
    default_database = QLNH,
    check_policy = off
go

use QLNH
go
-- tạo tên user theo tên [tenuser] và liên kết với login [tenlogin]
create user phuongnam for login develop
go

create user khachhang for login phunnam
go

-- cấp quyền grant (lệnh cho phép user làm gì đó)
grant select on CHINHANH to phuongnam

grant insert, update on GIAODICH to phuongnam

-- thu hồi quyền revoke (lấy lại quyền grand trước đó) hủy cả grant và deny
revoke update on GIAODICH to phuongnam

-- cấm quyền deny
/*
    deny cấm mạnh hơn revoke. deny là một lệnh cấm tuyệt đối ,ngay cả khi user đó
    thuộc một role(vai trò) có quyền, quyền deny luôn đc ưu tiên cao nhất
*/
deny delete on TAIKHOAN to phuongnam

-- phân vai trò role
create role nhanviennganhang;
go

grant select, insert, update, delete on KHACHHANG to nhanviennganhang
grant select, insert, update, delete on GIAODICH to nhanviennganhang

alter ROLE nhanviennganhang add member phuongnam

-- tạo schema
create schema HR;
create schema sales;
create schema finance;
go;

alter schema HR transfer dbo.GIAODICH
alter schema sales transfer dbo.KHACHHANG

grant select on schema::hr to phunnam
revoke select on schema::hr from phunnam
```

6.2 Bảo mật csdl

- BƯỚC 1: Tạo Chìa khóa chủ (Database Master Key) trong 'master'
- (Chìa khóa này sẽ bảo vệ Chứng chỉ ở bước 2)
- Nếu bạn đã có Master Key rồi, bước này sẽ báo lỗi, nhưng không sao, -- cứ tiếp tục sang bước 2.

```
USE master;
GO
--CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'phuongnam@3333';
GO
```

- BƯỚC 2: Tạo Chứng chỉ (Certificate) trong 'master'
- (Chứng chỉ này sẽ dùng để bảo vệ chìa khóa mã hóa CSDL)
- Đặt tên Chứng chỉ (TdeCert) theo ý bạn.

```
USE master;
GO
CREATE CERTIFICATE pnam
WITH SUBJECT = 'Chung Chi Dung De Ma Hoa TDE';
GO
```

- BƯỚC 3: Tạo Chìa khóa Mã hóa CSDL (Database Encryption Key - DEK)
- Chìa khóa này được lưu trong CSDL của bạn và được bảo vệ bởi
- Chứng chỉ (TdeCert) mà chúng ta vừa tạo ở bước 2.

```
USE [QLNH]; -- <-- THAY TÊN DB CỦA BẠN Ở ĐÂY
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE pnam;
GO
```

- BƯỚC 4: KÍCH HOẠT MÃ HÓA TDE
- **HÃY ĐỔI [TenDatabaseCuaBan] thành tên database của bạn (ví dụ: QLNH)**
- Đây là lúc quá trình mã hóa BẮT ĐẦU.
- Tùy CSDL lớn hay nhỏ, quá trình này có thể mất vài giây đến vài giờ.

```
ALTER DATABASE [QLNH]-- <-- THAY TÊN DB CỦA BẠN Ở ĐÂY
SET ENCRYPTION ON;
GO
```

```
/*
```

KIỂM TRA TRẠNG THÁI MÃ HÓA

```
*/
```

- Chạy lệnh này để xem trạng thái mã hóa.
- encryption_state = 3 là ĐÃ MÃ HÓA (ENCRYPTED)
- encryption_state = 2 là ĐANG MÃ HÓA (ENCRYPTION_IN_PROGRESS)
- encryption_state = 1 là CHƯA MÃ HÓA (UNENCRYPTED)

```
USE master;
GO
SELECT
    db.name,
    db.is_encrypted,
    dm.encryption_state,
    dm.percent_complete
FROM
```

```

sys.databases AS db
LEFT JOIN
    sys.dm_database_encryption_keys AS dm
    ON db.database_id = dm.database_id;
GO

/*
=====
BACKUP CHỨNG CHỈ (CỰC KỲ QUAN TRỌNG)
=====
*/
USE master;
GO

-- Backup Chứng chỉ và Khóa riêng (Private Key) ra file
-- Hãy lưu 2 file này (TdeCert.cer và TdeCert.pvk) ở một nơi an toàn
-- (ví dụ: USB, Google Drive,...)
BACKUP CERTIFICATE pnam
TO FILE = 'D:\Backup\TdeCert.cer' -- Bạn có thể đổi đường dẫn
WITH PRIVATE KEY (
    FILE = 'D:\Backup\TdeCert.pvk', -- Bạn có thể đổi đường dẫn
    ENCRYPTION BY PASSWORD = 'phuongnam@3333'
);
GO

USE master;
GO
SELECT
    db.name,
    db.is_encrypted,
    dm.encryption_state,
    dm.percent_complete
FROM sys.databases db
LEFT JOIN sys.dm_database_encryption_keys dm
    ON db.database_id = dm.database_id
WHERE db.name = 'QLNH';
GO

```

6.3 Quản lý sao lưu phục hồi

```

/*
-- back up file (full backup)
*/

backup database QLNH
to disk = 'E:\SQL_Backup\QLNH_Full_2025_11_13.bak'          -- đường dẫn nơi lưu file
with format,                                                 -- tạo mới file
      backup, định dạng lại thiết bị sao lưu
      init,                                                 -- ghi đè
      file backup nếu có
      name = 'BACKUP FULL 11-13-2025',                         -- tên mô tả cho bản backup
      medianame = 'quan li ngan hang backup',
      skip,
      stats = 10;                                              -- hiển
      thị tiến trình 10% một lần

/*
-- Restore từ file .bak thành database mới
*/
RESTORE DATABASE QLNH_Restore
FROM DISK = 'E:\SQL_Backup\QLNH_Full_2025_11_13.bak'          -- đường dẫn file backup

```

```

WITH
  MOVE 'QLNH' TO 'E:\SQLData\QLNH_Restore.mdf', -- Tên logical name của file dữ liệu
  (.mdf)
  MOVE 'QLNH_log' TO 'E:\SQLData\QLNH_Restore_log.ldf', -- Tên logical name của file log (.ldf)
  STATS = 10; -- Hiển thị tiến trình 10% mỗi lần

/*
Restore ghi đè database cũ
WITH REPLACE → cho phép ghi đè database cũ.

SINGLE_USER → đảm bảo không ai đang truy cập database khi restore.
*/
USE master;
ALTER DATABASE QLNH SET SINGLE_USER WITH ROLLBACK IMMEDIATE;

RESTORE DATABASE QLNH
FROM DISK = 'E:\SQL_Backup\QLNH_Full_2025_11_13.bak'
WITH REPLACE, STATS = 10;

ALTER DATABASE QLNH SET MULTI_USER;

RESTORE FILELISTONLY
FROM DISK = 'D:\Backup\QLNH_Full_2025_11_13.bak';

```

6.4 Quản lý hiệu suất csdl

-- tối ưu hóa hiệu suất csdl

```
/*
  6.4 Quản lý hiệu suất cơ sở dữ liệu
  1. Tối ưu hóa truy vấn (Query Optimization)
```

Mục tiêu: giảm thời gian thực thi và giảm tải cho máy chủ.

Kỹ thuật chính

Chọn lọc cột (SELECT cột cần thiết)
→ Không dùng SELECT *.

Lọc sớm ở WHERE
→ WHERE càng cụ thể, dữ liệu lấy ra càng ít → nhanh hơn.

Dùng JOIN đúng cách

Ưu tiên INNER JOIN khi có thể.

JOIN theo cột có index.

Tránh subquery lồng nhau phức tạp
→ Dùng JOIN hoặc CTE (WITH) khi phù hợp.

Sử dụng LIMIT / TOP khi chỉ cần một số dòng nhỏ.

Tránh hàm trên cột trong WHERE
(ví dụ WHERE YEAR(Ngay) = 2025 → chậm)
→ dùng WHERE Ngay >= '2025-01-01' AND Ngay < '2026-01-01'.

2. Tối ưu hóa chỉ mục (Index Optimization)

Mục tiêu: tăng tốc độ tìm kiếm, JOIN, ORDER BY, GROUP BY.

Khi nào cần index

Cột dùng trong WHERE

Cột dùng trong JOIN

Cột dùng trong ORDER BY / GROUP BY

Cột có dữ liệu lặp ít (selectivity cao)

Các loại index quan trọng

Clustered Index

Quyết định cách dữ liệu sắp xếp vật lý trong bảng

Thường đặt ở khóa chính (PK)

Non-clustered Index

Giúp tăng tốc truy vấn trên cột không phải PK

Composite Index (đa cột)

Chú ý thứ tự cột (tối ưu cho điều kiện từ trái sang phải)

Covering Index

Index chứa tất cả các cột truy vấn → SELECT không cần đọc bảng

Lưu ý: Index nhiều quá → INSERT/UPDATE/DELETE chậm.

3. Tối ưu hóa cấu trúc bảng và dữ liệu

Chuẩn hóa dữ liệu (3NF) để giảm trùng lặp → truy vấn nhanh hơn

Backup và bảo trì định kỳ

Rebuild/Reorganize Index

Update statistics

Partitioning

Chia bảng lớn (hàng triệu bản ghi) thành các phần nhỏ theo ngày/tháng → truy vấn nhanh hơn

Archiving

Chuyển dữ liệu cũ sang bảng khác giúp bảng chính nhỏ gọn

4. Tối ưu hóa tài nguyên và cấu hình hệ thống

Tối ưu RAM cho SQL Server / MySQL / PostgreSQL

Cấu hình số lượng kết nối tối đa

Bố trí ổ cứng:

SSD nhanh hơn HDD

Log và Data nên tách ổ

Caching

Query caching (MySQL)

Buffer Pool (SQL Server)

5. Giám sát hiệu suất (Monitoring)

Công cụ:

SQL Server: Execution Plan, Profiler, Extended Events

MySQL: EXPLAIN, Performance Schema

PostgreSQL: EXPLAIN ANALYZE

Theo dõi:

CPU – RAM – Disk I/O

Slow Query Log

Deadlock / Locking

Tình trạng index (fragmentation)

6. Cách thực hành tối ưu hóa trong thực tế

Bước 1: Dùng EXPLAIN / Execution Plan

→ Xem truy vấn đang quét bao nhiêu dòng.

Bước 2: Thêm hoặc chỉnh index

→ Đảm bảo toàn bộ truy vấn được “seek” thay vì “scan”.

Bước 3: Viết lại truy vấn cho tối ưu

→ Tránh phép toán không cần thiết.

Bước 4: Tối ưu bảng, partition, archive.

Bước 5: Theo dõi và bảo trì định kỳ.

*/

CHƯƠNG 7. KẾT LUẬN VÀ KHUYẾN NGHỊ

7.1 Tóm tắt kết quả

Dự án đã xây dựng thành công một hệ thống lõi (backend) vững chắc cho việc quản lý giao dịch ngân hàng cơ bản, với trọng tâm là đảm bảo **tính toàn vẹn dữ liệu, hiệu năng và an toàn trong môi trường đa người dùng**.

Các thành tựu cụ thể đạt được bao gồm:

- **Hoàn thiện các nghiệp vụ giao dịch cốt lõi:** Xây dựng thành công 03 Stored Procedure (SP) chính, đóng gói toàn bộ logic nghiệp vụ:
 - o sp_MoTaiKhoan: Tự động hóa quy trình nghiệp vụ phức tạp, kết hợp cả việc tạo khách hàng mới và mở tài khoản liên kết chỉ trong một lệnh gọi duy nhất.
 - o sp_ChuyenTien: Thực hiện logic chuyển tiền giữa hai tài khoản.
 - o sp_RutTien: Xử lý nghiệp vụ rút tiền khỏi một tài khoản.
- **Đảm bảo tính toàn vẹn dữ liệu tuyệt đối (Atomicity):**
 - o Tất cả các SP nghiệp vụ (Chuyển tiền, Rút tiền, Mở tài khoản) đều được bọc trong TRANSACTION.
 - o Điều này đảm bảo nguyên tắc "tất cả hoặc không có gì" (All or Nothing). Giao dịch chỉ thành công khi *tất cả* các bước (ví dụ: trừ tiền nguồn, cộng tiền đích, ghi log) đều hoàn tất. Nếu có bất kỳ lỗi nào xảy ra, ROLLBACK sẽ hủy bỏ mọi thay đổi, ngăn chặn triệt để các rủi ro như "mất tiền" hoặc sai lệch số dư.
- **Tối ưu hóa hiệu năng truy vấn (Performance):**
 - o Đã phân tích và tạo các NONCLUSTERED INDEX chiến lược trên các cột thường xuyên được sử dụng để tìm kiếm, lọc hoặc JOIN (ví dụ: KHACHHANG(SDT), GIAODICH(NgayGD), TAIKHOAN(MaKH)).
 - o Thành tựu này giúp tăng tốc độ phản hồi của hệ thống một cách đáng kể, đặc biệt khi cơ sở dữ liệu phát triển lớn hơn.
- **Xử lý đồng thời và chống xung đột dữ liệu (Concurrency):**
 - o Đã áp dụng các kỹ thuật khóa (UPDLOCK, ROWLOCK, TABLOCK, sp_getapplock) một cách hợp lý trong các transaction.
 - o Việc này giải quyết được bài toán "race condition" (khi nhiều giao dịch cố gắng sửa *cùng một* dữ liệu tại *cùng một thời điểm*), đảm bảo số dư tài khoản và mã giao dịch mới sinh ra luôn chính xác.
- **Thiết kế hệ thống linh hoạt và bảo mật:**
 - o Việc sử dụng Stored Procedure giúp che giấu logic xử lý phức tạp khỏi ứng dụng (frontend), tăng tính bảo mật và dễ bảo trì.
 - o Các SP được thiết kế với tham số OUTPUT (@TrangThai, @ThongBao) để trả về thông báo trạng thái rõ ràng, giúp ứng dụng dễ dàng tương tác và thông báo cho người dùng cuối.

7.2 Khuyến nghị

7.2.1 Khuyến nghị về Áp dụng và Triển khai

- **Xây dựng lớp API (Application Programming Interface):** Các Stored Procedure (SP) hiện là "trái tim" xử lý nghiệp vụ. Cần xây dựng một lớp API (ví dụ: sử dụng Node.js, .NET, Java) để làm cầu nối an toàn giữa ứng dụng (Mobile/Web) và cơ sở dữ liệu. Ứng dụng sẽ gọi API, và API sẽ gọi SP.
- **Thiết lập cơ chế Xác thực và Phân quyền:** Xây dựng hệ thống đăng nhập (Authentication) để xác định người dùng. Quan trọng hơn, cần phân quyền (Authorization) rõ ràng, ví dụ:
 - + **Khách hàng:** Chỉ được gọi sp_ChuyenTien, sp_RutTien cho chính tài khoản của mình.
 - + **Nhân viên (NV):** Được gọi sp_MoTaiKhoan cho khách hàng mới.
- **Kiểm thử tải (Stress Test):** Trước khi triển khai, cần thực hiện kiểm thử tải để đánh giá khả năng xử lý đồng thời (concurrency) của hệ thống. Cần đặc biệt kiểm tra các SP có TRANSACTION và khóa (lock) để đảm bảo không xảy ra "deadlock" (khóa chết) khi có hàng trăm giao dịch cùng lúc.

7.2.2 Khuyến nghị Cải tiến Hệ thống (Ngắn hạn)

- **Cải tiến cơ chế sinh mã tự động:** Phương pháp MAX(ID) + 1 (như đang dùng để sinh MaGiaoDich và MaKH) là một **điểm tắc nghẽn (bottleneck)** nghiêm trọng khi hệ thống có nhiều người dùng.
 - + **Khuyến nghị:** Chuyển sang sử dụng SEQUENCE OBJECT (trong SQL Server) hoặc cột IDENTITY để SQL Server tự động sinh mã mới một cách an toàn và hiệu suất cao.
- **Bổ sung logic Phí và Hạn mức giao dịch:**
 - + **Phí (Fee):** Cần thêm logic trừ phí giao dịch trong sp_ChuyenTien và sp_RutTien. Số dư khả dụng phải được kiểm tra ($SoDu \geq @SoTien + @PhiGiaoDich$).
 - + **Hạn mức (Limit):** Cần kiểm tra hạn mức giao dịch/ngày của tài khoản trước khi cho phép thực hiện.
- **Phân quyền trong CSDL (Database Permissions):** Tạo các "Database Role" (ví dụ: AppRole). Chỉ cấp quyền EXECUTE trên các Stored Procedure cho Role này. Ứng dụng API sẽ kết nối CSDL bằng user thuộc AppRole. Điều này ngăn chặn tuyệt đối việc ứng dụng (hoặc hacker) có thể truy cập trực tiếp vào bảng (SELECT * FROM TAIKHOAN).

7.2.3 Khuyến nghị Phát triển và Nghiên cứu (Dài hạn)

- **Xây dựng hệ thống Báo cáo (Reporting & BI):** Dữ liệu từ bảng GIAODICH rất quý giá. Hướng phát triển tiếp theo là xây dựng các hệ thống báo cáo (Business Intelligence) để trực quan hóa dữ liệu, giúp ban lãnh đạo theo dõi các chỉ số như:

tổng tiền giao dịch theo ngày/tháng, số lượng khách hàng mới, chi nhánh hoạt động hiệu quả nhất...

- **Tích hợp hệ thống Thông báo (Notification):** Nghiên cứu tích hợp các dịch vụ bên thứ ba (như Twilio, Firebase) để gửi thông báo (SMS, Email, Push Notification) cho khách hàng ngay khi có biến động số dư.
- **Nghiên cứu hệ thống Phát hiện gian lận (Fraud Detection):** Đây là một hướng nghiên cứu nâng cao. Hệ thống có thể phân tích các hành vi giao dịch "bất thường" (ví dụ: một tài khoản đột nhiên chuyển tiền nhiều lần lúc 3 giờ sáng, hoặc rút tiền ở nhiều địa điểm khác nhau) để cảnh báo rủi ro cho người dùng.

KẾT LUẬN

Về ưu điểm, dự án đã xây dựng thành công một hệ thống lõi nghiệp vụ vững chắc, đóng gói toàn bộ logic xử lý phức tạp (như chuyển tiền, rút tiền, mở tài khoản) vào bên trong các Stored Procedure (SP). Điểm mạnh nổi bật nhất là việc áp dụng nghiêm ngặt cơ chế TRANSACTION kết hợp với TRY...CATCH. Điều này đảm bảo tính toàn vẹn dữ liệu (Atomicity) tuyệt đối, giúp hệ thống tự động ROLLBACK khi có lỗi, ngăn chặn hiệu quả các rủi ro sai lệch số dư—một yêu cầu sống còn của hệ thống tài chính.Thêm vào đó, việc chủ động sử dụng các gợi ý khóa (locking hints) như UPDLOCK và sp_getapplock cho thấy sự hiểu biết sâu sắc về bài toán xử lý đồng thời (concurrency), đảm bảo dữ liệu vẫn chính xác ngay cả khi có nhiều giao dịch xảy ra cùng lúc.

Về nhược điểm, hạn chế lớn nhất của hệ thống nằm ở cơ chế sinh mã (ID) thủ công. Phương pháp truy vấn MAX(ID) + 1 từ bảng, ngay cả khi được bảo vệ bằng TABLOCK, là một điểm tắc nghẽn (bottleneck) nghiêm trọng về hiệu năng. Giải pháp này không thể mở rộng và sẽ làm giảm đáng kể tốc độ xử lý chung khi hệ thống đối mặt với lượng truy cập lớn trong thực tế. Bên cạnh đó, các nghiệp vụ vẫn còn ở mức cơ bản, thiếu các logic quan trọng như cơ chế tính phí giao dịch, kiểm tra hạn mức chuyển/rút tiền, và chưa xây dựng một hệ thống phân quyền (Authorization) chi tiết để kiểm soát quyền thực thi các SP.

Về hướng phát triển, ưu tiên hàng đầu là phải cải tổ ngay lập tức cơ chế sinh mã bằng cách chuyển sang sử dụng SEQUENCE OBJECT hoặc cột IDENTITY để CSDL tự động quản lý việc này, giúp tối ưu hóa hiệu năng. Bước tiếp theo là xây dựng một lớp API (ví dụ: REST API) làm cầu nối tiêu chuẩn giữa ứng dụng (Web/Mobile) và cơ sở dữ liệu. Song song đó, cần triển khai một hệ thống phân quyền chặt chẽ, chỉ cấp quyền EXECUTE trên các SP cho tài khoản ứng dụng và thu hồi mọi quyền truy cập trực tiếp vào bảng. Về lâu dài, hệ thống có thể dễ dàng mở rộng bằng cách bổ sung các module quản lý phí, hạn mức và phát triển các hệ thống báo cáo (BI) dựa trên dữ liệu giao dịch.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1]. W3Schools (2025), *SQL Tutorial*, [Trực tuyến]. Available: <https://www.w3schools.com/sql>. (Truy cập ngày 16/11/2025).
- [2]. Bro Code (2021), *SQL Full Course for Beginners*, [Video trực tuyến]. Available: <https://www.youtube.com/watch?v=HXV3zeQKqGY>. (Truy cập ngày 16/11/2025).
- [3]. Data with Dbra (2025), *SQL & Database Tutorials Playlist*, [Video trực tuyến]. Available: <https://www.youtube.com/@datawithdbra>. (Truy cập ngày 16/11/2025).