

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: THIẾT KẾ CƠ SỞ DỮ LIỆU CHO HỆ THỐNG QUẢN
LÝ TÀI CHÍNH**

Giáo viên hướng dẫn: Th.S Trần Thị Thanh Nhân

Sinh viên thực hiện:

Stt	Mã sv	Họ và tên	Lớp
1	1771020643	Hoàng văn Thi	CNTT 17-01
2	1771020092	Nguyễn Thanh Bình	CNTT 17-01
3	1771020256	Vũ Tuấn Hiệp	CNTT 17-01

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: THIẾT KẾ CƠ SỞ DỮ LIỆU CHO HỆ THỐNG QUẢN
LÝ TÀI CHÍNH**

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bảng Số	Bảng Chữ
1	1771020643	Hoàng văn Thi	25/09/2005		
2	1771020092	Nguyễn Thanh Bình	13/8/2005		
3	1771020256	Vũ Tuấn Hiệp	05/02/2005		

CÁN BỘ CHẤM THI 1

CÁN BỘ CHẤM THI 2

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Môn học Cơ Sở Dữ Liệu (CSDL) là một trong những lĩnh vực quan trọng trong ngành Công nghệ Thông tin. Với sự phát triển nhanh chóng của công nghệ, việc quản lý và khai thác dữ liệu ngày càng trở nên cần thiết. Cơ sở dữ liệu không chỉ giúp lưu trữ thông tin mà còn hỗ trợ trong việc truy xuất, phân tích và xử lý dữ liệu một cách hiệu quả.

Trong bối cảnh hiện nay, khi dữ liệu ngày càng lớn và phức tạp, việc hiểu rõ các khái niệm cơ bản về cơ sở dữ liệu là rất quan trọng. Môn học này sẽ cung cấp cho sinh viên những kiến thức nền tảng về mô hình dữ liệu, ngôn ngữ truy vấn (như SQL), thiết kế cơ sở dữ liệu và các hệ quản trị cơ sở dữ liệu phổ biến.

Chúng ta sẽ cùng nhau khám phá cách mà các hệ thống cơ sở dữ liệu hoạt động, từ việc lưu trữ thông tin cho đến việc tối ưu hóa truy vấn, đảm bảo tính toàn vẹn của dữ liệu và bảo mật thông tin. Qua đó, sinh viên sẽ có được những kỹ năng cần thiết để phát triển và quản lý các ứng dụng dữ liệu trong thực tế.

Hy vọng rằng môn học này sẽ mang lại cho các bạn những trải nghiệm bổ ích và kiến thức sâu sắc, giúp bạn tự tin trong việc làm việc với dữ liệu trong tương lai.

MỤC LỤC

LỜI NÓI ĐẦU	3
MỤC LỤC	4
MỤC LỤC HÌNH ẢNH	7
BẢNG CÁC TỪ VIẾT TẮT	10
CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI QUẢN LÝ TÀI CHÍNH SỬ DỤNG SQL SERVER	11
1.1. Giới thiệu đề tài	11
1.2. Mục tiêu đề tài	11
1.3. Các chức năng chính.....	11
1.3.1. Quản lý người dùng	11
1.3.2. Quản lý tài khoản tài chính.....	12
1.3.3. Quản lý giao dịch tài chính.....	12
1.3.4. Quản lý danh mục đầu tư.....	12
1.3.5. Quản lý ngân sách cá nhân	12
1.3.6. Quản lý các khoản vay	13
1.4. Công nghệ sử dụng	13
CHƯƠNG 2. Mô hình cơ sở dữ liệu quan hệ.....	14
2.1. Xác định các thực thể, thuộc tính và ràng buộc	14
2.1.1. Thực thể	14
2.1.2. Thuộc tính và ràng buộc	14
2.2. xây dựng các bảng	15
CHƯƠNG 3. Tạo cơ sở dữ liệu.....	17
3.1. Tạo database	17
3.1.1. Database Diagram	17

3.1.2. Cài đặt Hệ thống trên SQL Servers	17
3.2. chèn dữ liệu	20
3.3. In bảng dữ liệu	22
CHƯƠNG 4. XÂY DỰNG CÁC VIEW.....	26
4.1. Tạo các view	26
4.2. In thông tin view	29
CHƯƠNG 5. XÂY DỰNG CÁC PROCEDURE	34
5.1. Thêm người dùng mới	34
5.2. Cập nhật thông tin người dùng	35
5.3. Xóa người dùng (và các dữ liệu liên quan)	36
5.4. Thêm tài khoản ngân hàng	37
5.5. Gửi tiền vào tài khoản	38
5.6. Rút tiền từ tài khoản	39
5.7. Thêm giao dịch mới.....	40
5.8. Xem lịch sử giao dịch của người dùng.....	41
CHƯƠNG 6. XÂY DỰNG CÁC TRIGGER.....	42
6.1. Trigger cập nhật số dư tài khoản sau giao dịch	42
6.2. Trigger kiểm tra ngày mua đầu tư	43
6.3. Trigger cập nhật giá trị đầu tư sau khi thay đổi.....	44
6.4. Trigger cập nhật số tiền đã chi tiêu trong ngân sách	45
6.5. Trigger kiểm tra trạng thái khoản vay	46
6.6. Trigger cập nhật ngày đáo hạn khoản vay	47
6.7. Trigger ghi log giao dịch	48
6.8. Trigger xóa người dùng và tất cả dữ liệu liên quan (INSTEAD OF DELETE)	48

6.9. Trigger kiểm tra số dư trước khi giao dịch.....	50
6.10. Trigger kiểm tra ngân sách trước khi chi tiêu	50
6.11. Chạy các trigger.....	51
CHƯƠNG 7. PHÂN QUYỀN VÀ BẢO VỆ CƠ SỞ DỮ LIỆU	53
7.1. Tạo Login cho người dùng	53
7.2. Tạo Users trong CSDL QuanLyTaiChinh.....	54
7.3. Phân quyền	55
7.3.1. Phân quyền cho admin.....	55
7.3.2. Phân quyền cho nhân viên.....	55
7.3.3. Phân quyền cho khách hàng	57
7.3.4. Giới hạn dữ liệu bằng Row-Level Security (RLS).....	57
7.4. Kiểm tra quyền	59
DANH MỤC TÀI LIỆU THAM KHẢO	60

MỤC LỤC HÌNH ẢNH

Hình 1. Bảng Users Lưu trữ thông tin người dùng	15
Hình 2. Bảng Accounts lưu trữ thông tin tài khoản ngân hàng.....	15
Hình 3. Bảng lưu trữ thông tin giao dịch.....	15
Hình 4. Bảng lưu trữ đầu tư.....	16
Hình 5. Bảng lưu trữ ngân sách.....	16
Hình 6. Bảng lưu trữ các khoản vay.....	16
Hình 7. database Diagram Quản lý tài chính.....	17
Hình 8. Code khởi tạo cơ sở dữ liệu.....	17
Hình 9. Tạo bảng user	18
Hình 10. Tạo bảng accounts	18
Hình 11. Tạo bảng transactions	18
Hình 12. Code tạo bảng investments.....	19
Hình 13. Tạo bảng budget	19
Hình 14. Thêm dữ liệu bảng users.....	20
Hình 15. Thêm dữ liệu bảng accounts.....	20
Hình 16. Thêm dữ liệu bảng transactions.....	21
Hình 17. Nhập dữ liệu bảng investments	21
Hình 18. Nhập dữ liệu bảng budget.....	21
Hình 19. Nhập dữ liệu bảng loans	22
Hình 20. Dữ liệu có trong bảng users.....	22
Hình 21. Dữ liệu bảng accounts	23
Hình 22. Dữ liệu bảng transactions	23
Hình 23. Dữ liệu bảng investments.....	24
Hình 24. Dữ liệu bảng budget	24
Hình 25. Dữ liệu bảng loans.....	25
Hình 26. Code view UserAccounts	26
Hình 27. Code view UserTransactions.....	26
Hình 28. Code view UserTotalBalance	27
Hình 29. Code view UserInvestments.....	27

Hình 30. Code view UserBudgets	27
Hình 31. Code View UserLoans.....	28
Hình 32. View AccountTransactions.	28
Hình 33. Code view ProfitableInvestments.....	28
Hình 34. Code View RemainingBudget.	29
Hình 35. Code View ActiveLoans.....	29
Hình 36. In dữ liệu view UserAccounts	29
Hình 37. In dữ liệu view UserTransactions.....	30
Hình 38. In dữ liệu view UserTotalBalance.....	30
Hình 39. In dữ liệu view UserInvestments.....	31
Hình 40 In thông tin view UserBudgets.....	31
Hình 41. In thông tin view UserLoans.	32
Hình 42. in thông tin view AccountTransactions.....	32
Hình 43. In thông tin view ProfitableInvestments.	33
Hình 44. In thông tin view RemainingBudget.	33
Hình 45. In thông tin view ActiveLoans.	33
Hình 46. procedure thêm người mới	34
Hình 47. procedure cập nhật người dùng	35
Hình 48. Xóa người dùng procedure	36
Hình 49. procedure thêm tài khoản ngân hàng.....	37
Hình 50. procedure gửi tiền vào tài khoản	38
Hình 51. procedure rút tiền từ tài khoản.....	39
Hình 52. Procedure thêm giao dịch mới.....	40
Hình 53. Thêm giao dịch procedure xem lịch sử giao dịch người dùng	41
Hình 54. Trigger cập nhật số dư tài khoản sau giao dịch	42
Hình 55. Trigger kiểm tra ngày mua đầu tư	43
Hình 56. Trigger cập nhật giá trị đầu tư khi thay đổi	44
Hình 57. Trigger cập nhật số tiền đã chi tiêu trong ngân sách	45
Hình 58. Trigger kiểm tra trạng thái khoản vay	46
Hình 59. Trigger cập nhật đáo hạn khoản vay	47

Hình 60. trigger khi log giao dịch	48
Hình 61. trigger xóa người dùng và dữ liệu liên quan	49
Hình 62. Trigger kiểm tra số dư trước khi giao dịch.....	50
Hình 63. trigger kiểm tra ngân sách trước khi tiêu.....	51
Hình 64. Chạy các trigger.....	51
Hình 65. Tạo login người dùng	53
Hình 66. Tạo user cho người dùng	54
Hình 67. Phân quyền cho admin.....	55
Hình 68. Phân quyền cho nhân viên được xem,sửa,cập nhật.....	56
Hình 69. Cấp quyền select trên các view	56
Hình 70. Phân quyền cho khách hàng	57
Hình 71. Áp dụng RLS giới hạn dữ liệu của khách	58
Hình 72. Tạo chính sách bảo mật	58
Hình 73. Kiểm tra quyền	59

BẢNG CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	CSDL	Cơ sở dữ liệu
2	SQL	Structured Query Language

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI QUẢN LÝ TÀI CHÍNH SỬ DỤNG SQL SERVER

1.1. Giới thiệu đề tài

Quản lý tài chính cá nhân là một nhu cầu thiết yếu nhằm kiểm soát thu nhập, chi tiêu, tiết kiệm, đầu tư và các khoản vay một cách hiệu quả. Với sự phát triển của công nghệ, việc xây dựng một hệ thống quản lý tài chính giúp người dùng theo dõi tình trạng tài chính của mình một cách khoa học và dễ dàng hơn.

Hệ thống "Quản lý tài chính cá nhân" được thiết kế để giúp người dùng ghi nhận các giao dịch tài chính, lập ngân sách chi tiêu, quản lý tài khoản ngân hàng, theo dõi các khoản vay và đầu tư. Ứng dụng sử dụng SQL Server để lưu trữ dữ liệu, giúp đảm bảo tính bảo mật, truy vấn nhanh chóng và xử lý thông tin chính xác.

1.2. Mục tiêu đề tài

- **Tổ chức và lưu trữ dữ liệu tài chính:** Xây dựng một cơ sở dữ liệu có cấu trúc rõ ràng, an toàn và dễ mở rộng để lưu trữ thông tin tài chính cá nhân của người dùng.
- **Hỗ trợ quản lý tài chính toàn diện:** Cung cấp các công cụ để người dùng theo dõi tài khoản, giao dịch, đầu tư, ngân sách và các khoản vay một cách chi tiết và trực quan.
- **Tăng cường khả năng ra quyết định:** Giúp người dùng đưa ra các quyết định tài chính thông minh dựa trên dữ liệu được phân tích từ hệ thống.
- **Đảm bảo tính linh hoạt và bảo mật:** Đáp ứng nhu cầu cá nhân hóa của từng người dùng đồng thời bảo vệ thông tin nhạy cảm. Các chức năng chính

1.3. Các chức năng chính

1.3.1. Quản lý người dùng

- **Mục đích:** Lưu trữ và quản lý thông tin cá nhân của người dùng để xác định danh tính và hỗ trợ đăng nhập.
- **Chi tiết:**
 - Lưu thông tin như tên đăng nhập, mật khẩu, email, họ tên đầy đủ và ngày sinh.
 - Đảm bảo tính duy nhất của tên đăng nhập và email thông qua ràng buộc.
 - Sử dụng mã định danh người dùng làm khóa chính để liên kết với các bảng khác.

1.3.2. Quản lý tài khoản tài chính

- **Mục đích:** Theo dõi và quản lý các tài khoản ngân hàng của người dùng.
- **Chi tiết:**
 - Ghi nhận thông tin tài khoản bao gồm tên tài khoản, tên ngân hàng, số tài khoản và số dư.
 - Mỗi tài khoản được liên kết với một người dùng thông qua mã định danh người dùng.
 - Hỗ trợ kiểm tra số dư hiện tại và cập nhật khi có giao dịch.

1.3.3. Quản lý giao dịch tài chính

- **Mục đích:** Ghi chép và phân loại các giao dịch để người dùng nắm rõ dòng tiền.
- **Chi tiết:**
 - Lưu trữ thông tin giao dịch như ngày giao dịch, mô tả, số tiền, danh mục và tài khoản liên quan.
 - Liên kết giao dịch với người dùng và tài khoản thông qua các mã định danh.
 - Cho phép phân tích chi tiêu theo danh mục hoặc khoảng thời gian.

1.3.4. Quản lý danh mục đầu tư

- **Mục đích:** Theo dõi hiệu suất và giá trị của các khoản đầu tư.
- **Chi tiết:**
 - Quản lý thông tin đầu tư bao gồm tên khoản đầu tư, loại đầu tư, giá trị ban đầu, giá trị hiện tại và ngày mua.
 - Liên kết với người dùng qua mã định danh để cá nhân hóa danh mục đầu tư.
 - Hỗ trợ tính toán lợi nhuận hoặc lỗ dựa trên sự chênh lệch giữa giá trị hiện tại và giá trị ban đầu.

1.3.5. Quản lý ngân sách cá nhân

- **Mục đích:** Giúp người dùng lập kế hoạch và kiểm soát chi tiêu hàng tháng.
- **Chi tiết:**
 - Lưu trữ thông tin ngân sách theo tháng/năm, danh mục, số tiền phân bổ và số tiền đã chi.
 - Liên kết với người dùng qua mã định danh để quản lý ngân sách cá nhân.
 - Hỗ trợ so sánh giữa số tiền đã chi và số tiền phân bổ để cảnh báo vượt ngân sách.

1.3.6. Quản lý các khoản vay

Mục đích: Theo dõi và quản lý các khoản vay để đảm bảo thanh toán đúng hạn. Chi tiết: Ghi nhận thông tin khoản vay như tên người cho vay, số tiền vay, lãi suất, ngày bắt đầu, ngày đến hạn và trạng thái. Trạng thái khoản vay được giới hạn trong các giá trị: "Đang hoạt động", "Đã trả", "Quá hạn" thông qua ràng buộc. Liên kết với người dùng qua mã định danh để quản lý riêng biệt.

1.4. Công nghệ sử dụng

- **SQL Server:** Được sử dụng để lưu trữ và quản lý dữ liệu của hệ thống. SQL Server cung cấp:
 - Khả năng quản lý cơ sở dữ liệu quan hệ với hiệu suất cao.
 - Tính năng bảo mật mạnh mẽ để bảo vệ thông tin tài chính nhạy cảm.
 - Hỗ trợ các truy vấn phức tạp để phân tích dữ liệu tài chính.
 - Khả năng mở rộng khi số lượng người dùng hoặc dữ liệu tăng lên.

CHƯƠNG 2. Mô hình cơ sở dữ liệu quan hệ

2.1. Xác định các thực thể, thuộc tính và ràng buộc

Phần này trình bày các thực thể chính trong hệ thống Quản Lý Tài Sản (QuanLyTaiChinh), cùng với các thuộc tính của chúng và các ràng buộc được áp dụng để đảm bảo tính toàn vẹn dữ liệu.

2.1.1. Thực thể

❖ **Hệ thống Quản Lý Tài Chính** bao gồm các thực thể chính sau:

1. **Users** (user_id, username, password, email, full_name, date_of_birth):
 - Đại diện cho người sử dụng hệ thống, là đối tượng chính thực hiện các hoạt động quản lý tài chính.
2. **Accounts** (account_id, user_id, account_name, bank_name, account_number, balance)
 - Đại diện cho các tài khoản ngân hàng mà người dùng sở hữu.
3. **Transactions** (transaction_id, user_id, transaction_date, description, amount, category, account_id)
 - Đại diện cho các giao dịch tài chính mà người dùng thực hiện.
4. **Investments** (investment_id, user_id, investment_name, investment_type, initial_investment, current_value, purchase_date)
 - Đại diện cho các khoản đầu tư mà người dùng thực hiện.
5. **Budget** (budget_id, user_id, month_year, category, allocated_amount, spent_amount)
 - Đại diện cho kế hoạch ngân sách hàng tháng của người dùng.
6. **Loans** (loan_id, user_id, lender, loan_amount, interest_rate, start_date, due_date, status)
 - Đại diện cho các khoản vay mà người dùng đang quản lý.

2.1.2. Thuộc tính và ràng buộc

❖ **Users** (user_id, username, password, email, full_name, date_of_birth)

❖ **Ràng buộc:**

- **user_id:** PK, IDENTITY(1,1).
- **username:** UNIQUE, NOT NULL.
- **password:** NOT NULL.
- **email:** UNIQUE, NOT NULL.

- **full_name:** NOT NULL.
- **date_of_birth:** NOT NULL.

2.2.xây dựng các bảng

❖ Xây dựng bảng user (người dùng):

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng Buộc	Ghi chú
1	user_id	INT	PK, IDENTITY(1,1)	Mã định danh người dùng
2	username	NVARCHAR(100)	UNIQUE, NOT NULL	Tên đăng nhập
3	password	NVARCHAR(255)	NOT NULL	Mật khẩu
4	email	NVARCHAR(255)	UNIQUE, NOT NULL	Địa chỉ email
5	full_name	NVARCHAR(255)	NOT NULL	Họ tên đầy đủ
6	date_of_birth	DATE	NOT NULL	Ngày sinh

Hình 1. Bảng Users Lưu trữ thông tin người dùng

❖ Xây dựng bảng Accounts (Tài khoản ngân hàng):

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng Buộc	Ghi chú
1	account_id	INT	PK, IDENTITY(1,1)	Mã định danh tài khoản
2	user_id	INT	FK, NOT NULL	Mã định danh người dùng
3	account_name	NVARCHAR(255)	NOT NULL	Tên tài khoản
4	bank_name	NVARCHAR(255)	NOT NULL	Tên ngân hàng
5	account_number	NVARCHAR(50)	UNIQUE, NOT NULL	Số tài khoản
6	balance	DECIMAL(18,2)	NOT NULL, DEFAULT 0	Số dư tài khoản

Hình 2. Bảng Accounts lưu trữ thông tin tài khoản ngân hàng

❖ Xây dựng bảng Transactions (Giao Dịch):

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng Buộc	Ghi chú
1	transaction_id	INT	PK, IDENTITY(1,1)	Mã định danh giao dịch
2	user_id	INT	FK, NOT NULL	Mã định danh người dùng
3	transaction_date	DATE	NOT NULL	Ngày giao dịch
4	description	NVARCHAR(500)		Mô tả giao dịch
5	amount	DECIMAL(18,2)	NOT NULL	Số tiền giao dịch
6	category	NVARCHAR(100)	NOT NULL	Danh mục giao dịch
7	account_id	INT	FK, NOT NULL	Mã định danh tài khoản

Hình 3. Bảng lưu trữ thông tin giao dịch

❖ Xây dựng bảng Investments (Đầu tư):

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng Buộc	Ghi chú
1	investment_id	INT	PK, IDENTITY(1,1)	Mã định danh đầu tư
2	user_id	INT	FK, NOT NULL	Mã định danh người dùng
3	investment_name	NVARCHAR(255)	NOT NULL	Tên khoản đầu tư
4	investment_type	NVARCHAR(100)	NOT NULL	Loại đầu tư
5	initial_investment	DECIMAL(18,2)	NOT NULL	Giá trị ban đầu
6	current_value	DECIMAL(18,2)	NOT NULL	Giá trị hiện tại
7	purchase_date	DATE	NOT NULL	Ngày mua

Hình 4. Bảng lưu trữ đầu tư

❖ **Xây dựng bảng Budget (Ngân Sách):**

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng Buộc	Ghi chú
1	budget_id	INT	PK, IDENTITY(1,1)	Mã định danh ngân sách
2	user_id	INT	FK, NOT NULL	Mã định danh người dùng
3	month_year	DATE	NOT NULL	Tháng/năm ngân sách
4	category	NVARCHAR(100)	NOT NULL	Danh mục ngân sách
5	allocated_amount	DECIMAL(18,2)	NOT NULL	Số tiền phân bổ
6	spent_amount	DECIMAL(18,2)	DEFAULT 0	Số tiền đã chi

Hình 5. Bảng lưu trữ ngân sách

❖ **Xây dựng bảng Loans (khoản vay):**

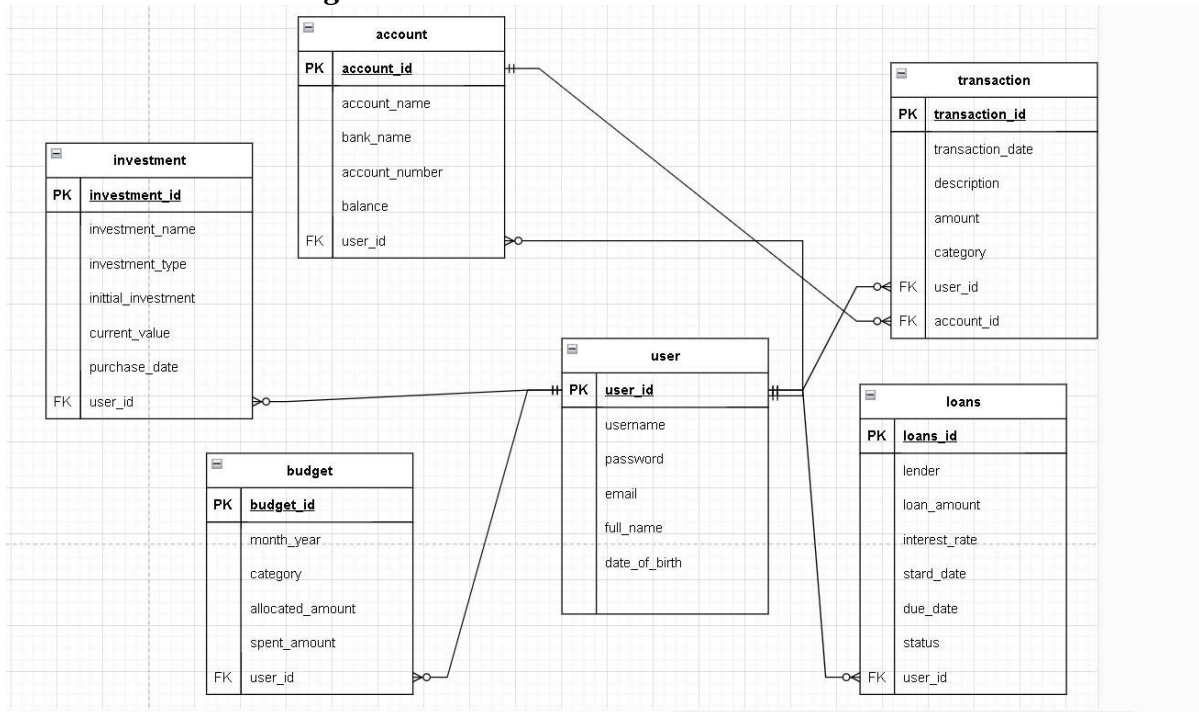
TT	Tên thuộc tính	Kiểu dữ liệu	Ràng Buộc	Ghi chú
1	loan_id	INT	PK, IDENTITY(1,1)	Mã định danh khoản vay
2	user_id	INT	FK, NOT NULL	Mã định danh người dùng
3	lender	NVARCHAR(255)	NOT NULL	Người cho vay
4	loan_amount	DECIMAL(18,2)	NOT NULL	Số tiền vay
5	interest_rate	DECIMAL(18,2)	NOT NULL	Lãi suất
6	start_date	DATE	NOT NULL	Ngày bắt đầu
7	due_date	DATE	NOT NULL	Ngày đến hạn
8	status	NVARCHAR(50)	CHECK	Trạng thái khoản vay

Hình 6. Bảng lưu trữ các khoản vay

CHƯƠNG 3. Tạo cơ sở dữ liệu

3.1. Tạo database

3.1.1. Database Diagram



Hình 7. database Diagram Quản lý tài chính

3.1.2. Cài đặt Hệ thống trên SQL Servers

❖ Khởi tạo Cơ sở dữ liệu:

```
CREATE DATABASE QuanLyTaiChinh
GO
USE QuanLyTaiChinh;
GO
```

Hình 8. Code khởi tạo cơ sở dữ liệu

❖ Tạo các bảng dữ liệu:

- Tạo Bảng user:

```
--Tao bang Users
CREATE TABLE users (
    user_id INT IDENTITY(1,1) PRIMARY KEY, -- khoa chinh
    username NVARCHAR(100) NOT NULL UNIQUE,
    password NVARCHAR(255) NOT NULL,
    email NVARCHAR(255) NOT NULL UNIQUE,
    full_name NVARCHAR(255) NOT NULL,
    date_of_birth DATE NOT NULL
);
```

KẾT QUẢ

user_id	username	password	email	full_name	date_of_birth
---------	----------	----------	-------	-----------	---------------

Hình 9. Tạo bảng user

- Tạo bảng Account:

--Tao bang accounts

```
CREATE TABLE accounts (  
    account_id INT IDENTITY(1,1) PRIMARY KEY, -- khoa chinh  
    user_id INT NOT NULL,  
    account_name NVARCHAR(255) NOT NULL,  
    bank_name NVARCHAR(255) NOT NULL,  
    account_number NVARCHAR(50) NOT NULL UNIQUE,  
    balance DECIMAL(18,2) NOT NULL DEFAULT 0,  
    FOREIGN KEY (user_id) REFERENCES users(user_id)  
);  
GO
```

KẾT QUẢ

account_id	user_id	account_name	bank_name	account_number	balance
------------	---------	--------------	-----------	----------------	---------

Hình 10. Tạo bảng accounts

- Tạo bảng transactions:

-- Tao bang transactions

```
CREATE TABLE transactions (  
    transaction_id INT IDENTITY(1,1) PRIMARY KEY, -- Khoa chinh  
    user_id INT NOT NULL,  
    transaction_date DATE NOT NULL,  
    description NVARCHAR(500),  
    amount DECIMAL(18,2) NOT NULL,  
    category NVARCHAR(100) NOT NULL,  
    account_id INT NOT NULL,  
    FOREIGN KEY (user_id) REFERENCES users(user_id),  
    FOREIGN KEY (account_id) REFERENCES accounts(account_id)  
);  
GO
```

KẾT QUẢ

transaction_id	user_id	transaction_date	description	amount	category	account_id
----------------	---------	------------------	-------------	--------	----------	------------

Hình 11. Tạo bảng transactions

- **Tạo Bảng investments:**

```
-- Tao bang investments
CREATE TABLE investments (
    investment_id INT IDENTITY(1,1) PRIMARY KEY, -- khoa chinh
    user_id INT NOT NULL,
    investment_name NVARCHAR(255) NOT NULL,
    investment_type NVARCHAR(100) NOT NULL,
    initial_investment DECIMAL(18,2) NOT NULL,
    current_value DECIMAL(18,2) NOT NULL,
    purchase_date DATE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
GO
```

KẾT QUẢ

investment_id	user_id	investment_name	investment_type	initial_investment	current_value	purchase_date
---------------	---------	-----------------	-----------------	--------------------	---------------	---------------

Hình 12. Code tạo bảng investments.

- **Tạo bảng budget:**

```
-- tao bang budget
CREATE TABLE budget (
    budget_id INT IDENTITY(1,1) PRIMARY KEY,
    user_id INT NOT NULL,
    month_year DATE NOT NULL,
    category NVARCHAR(100) NOT NULL,
    allocated_amount DECIMAL(18,2) NOT NULL,
    spent_amount DECIMAL(18,2) DEFAULT 0,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
GO
```

KẾT QUẢ

budget_id	user_id	month_year	category	allocated_amount	spent_amount
-----------	---------	------------	----------	------------------	--------------

Hình 13. Tạo bảng budget

- **Tạo Bảng loans:**

```
-- tao bang Loans
CREATE TABLE loans (
    loan_id INT IDENTITY(1,1) PRIMARY KEY,
    user_id INT NOT NULL,
    lender NVARCHAR(255) NOT NULL,
    loan_amount DECIMAL(18,2) NOT NULL,
    interest_rate DECIMAL(5,2) NOT NULL,
    start_date DATE NOT NULL,
    due_date DATE NOT NULL,
    status NVARCHAR(50) CHECK (status IN ('Active', 'Paid', 'Defaulted')),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
GO
```

KẾT QUẢ

loan_id	user_id	lender	loan_amount	interest_rate	start_date	due_date	status
---------	---------	--------	-------------	---------------	------------	----------	--------

3.2.chèn dữ liệu

❖ Nhập dữ liệu bảng users:

```
-- Thêm thông tin vào bảng người dùng.  
INSERT INTO users (username, password, email, full_name, date_of_birth) VALUES  
( 'user1', 'password123', 'user1@example.com', 'Nguyen Van A', '1990-01-01'),  
( 'user2', 'securepass', 'user2@example.com', 'Tran Thi B', '1985-05-15'),  
( 'user3', 'pass1234', 'user3@example.com', 'Le Hoang C', '1992-11-20'),  
( 'user4', 'strongpass', 'user4@example.com', 'Pham Thu D', '1988-03-10'),  
( 'user5', 'pass5678', 'user5@example.com', 'Vo Minh E', '1995-07-25'),  
( 'user6', 'pass9876', 'user6@example.com', 'Hoang Gia F', '1987-12-05'),  
( 'user7', 'pass4321', 'user7@example.com', 'Doan Ngoc G', '1991-09-18'),  
( 'user8', 'pass8765', 'user8@example.com', 'Bui Thanh H', '1989-06-30'),  
( 'user9', 'pass1122', 'user9@example.com', 'Truong My I', '1993-02-12'),  
( 'user10', 'pass3344', 'user10@example.com', 'Dang Van K', '1986-08-22'),  
( 'user11', 'pass5566', 'user11@example.com', 'Ly Thi L', '1994-04-08'),  
( 'user12', 'pass7788', 'user12@example.com', 'Vu Duc M', '1990-10-28'),  
( 'user13', 'pass9900', 'user13@example.com', 'Phan Kim N', '1987-01-15'),  
( 'user14', 'pass2233', 'user14@example.com', 'Ha Xuan O', '1992-05-03'),  
( 'user15', 'pass4455', 'user15@example.com', 'Cao Thuy P', '1988-11-19'),  
( 'user16', 'pass6677', 'user16@example.com', 'Dinh Quang Q', '1995-03-07'),  
( 'user17', 'pass8899', 'user17@example.com', 'Lam Bao R', '1989-09-24'),  
( 'user18', 'pass1212', 'user18@example.com', 'Mai Anh S', '1991-07-11'),  
( 'user19', 'pass3434', 'user19@example.com', 'Ngo Thanh T', '1993-12-29'),  
( 'user20', 'pass5656', 'user20@example.com', 'Quach Hong U', '1986-04-16'),  
( 'user21', 'pass7878', 'user21@example.com', 'Son Tung V', '1994-08-02'),  
( 'user22', 'pass9090', 'user22@example.com', 'Thai Binh W', '1990-02-27'),  
( 'user23', 'pass2323', 'user23@example.com', 'Uyen Chi X', '1987-06-14'),  
( 'user24', 'pass4545', 'user24@example.com', 'Vinh Duc Y', '1992-10-01'),
```

Hình 14. Thêm dữ liệu bảng users

❖ Nhập dữ liệu bảng accounts:

```
-- Thêm thông tin vào bảng tài khoản.  
INSERT INTO accounts (user_id, account_name, bank_name, account_number, balance) VALUES  
(1, 'Saving Account', 'Vietcombank', '1234567890', 5000000),  
(2, 'Checking Account', 'Techcombank', '0987654321', 10000000),  
(3, 'Salary Account', 'BIDV', '1122334455', 2500000),  
(4, 'Investment Account', 'ACB', '6677889900', 7500000),  
(5, 'Personal Account', 'Agribank', '5544332211', 3000000),  
(6, 'Business Account', 'VPBank', '9988776655', 15000000),  
(7, 'Joint Account', 'MBBank', '4455667788', 6000000),  
(8, 'Travel Account', 'Sacombank', '8899001122', 4000000),  
(9, 'Education Account', 'Shinhan Bank', '2211009988', 8000000),  
(10, 'Home Account', 'Eximbank', '7766554433', 12000000),  
(11, 'Saving Account 2', 'VietinBank', '3344556677', 5500000),  
(12, 'Checking Account 2', 'TPBank', '1122998877', 9000000),  
(13, 'Salary Account 2', 'OceanBank', '6655443322', 2800000),  
(14, 'Investment Account 2', 'LienVietPostBank', '9900112233', 7000000),  
(15, 'Personal Account 2', 'MSB', '4433221100', 3500000),  
(16, 'Business Account 2', 'Bac A Bank', '8877665544', 16000000),  
(17, 'Joint Account 2', 'GPBank', '2233445566', 6500000),  
(18, 'Travel Account 2', 'DongA Bank', '7788990011', 4500000),  
(19, 'Education Account 2', 'PVcomBank', '3322110099', 8500000),  
(20, 'Home Account 2', 'KienLongBank', '5566778899', 11000000),
```

Hình 15. Thêm dữ liệu bảng accounts

❖ Nhập dữ liệu bảng transactions:

```
-- Thêm dữ liệu bảng transactions
INSERT INTO transactions (user_id, transaction_date, description, amount, category, account_id) VALUES
(1, '2023-10-26', 'Grocery shopping', -150000, 'Food', 1),
(2, '2023-10-26', 'Salary deposit', 1000000, 'Income', 2),
(3, '2023-10-25', 'Online purchase', -500000, 'Shopping', 3),
(4, '2023-10-25', 'Investment return', 2000000, 'Investment', 4),
(5, '2023-10-24', 'Utility bill', -300000, 'Bills', 5),
(6, '2023-10-24', 'Business payment', 5000000, 'Business', 6),
(7, '2023-10-23', 'Rent payment', -2000000, 'Housing', 7),
(8, '2023-10-23', 'Travel expenses', -1000000, 'Travel', 8),
(9, '2023-10-22', 'Tuition fee', -1500000, 'Education', 9),
(10, '2023-10-22', 'Home repair', -800000, 'Home', 10),
(11, '2023-10-21', 'Savings transfer', -500000, 'Savings', 11),
(12, '2023-10-21', 'ATM withdrawal', -200000, 'Cash', 12),
(13, '2023-10-20', 'Online subscription', -100000, 'Subscriptions', 13),
(14, '2023-10-20', 'Dividend payment', 300000, 'Investment', 14),
(15, '2023-10-19', 'Personal expenses', -400000, 'Personal', 15),
(16, '2023-10-19', 'Business income', 2500000, 'Business', 16),
(17, '2023-10-18', 'Joint expense', -600000, 'Shared', 17),
(18, '2023-10-18', 'Travel booking', -1200000, 'Travel', 18),
(19, '2023-10-17', 'Education supplies', -700000, 'Education', 19),
(20, '2023-10-17', 'Home improvement', -1500000, 'Home', 20),
```

Hình 16. Thêm dữ liệu bảng transactions.

❖ Nhập dữ liệu bảng investments

```
-- thêm dữ liệu bảng investments
INSERT INTO investments (user_id, investment_name, investment_type, initial_investment, current_value, purchase_date) VALUES
(1, 'Stock A', 'Stocks', 5000000, 6000000, '2023-01-15'),
(2, 'Bond B', 'Bonds', 10000000, 10500000, '2023-02-20'),
(3, 'Mutual Fund C', 'Mutual Funds', 2500000, 3000000, '2023-03-25'),
(4, 'Real Estate D', 'Real Estate', 7500000, 8000000, '2023-04-30'),
(5, 'Gold E', 'Commodities', 3000000, 3200000, '2023-05-05'),
(6, 'Stock F', 'Stocks', 15000000, 16000000, '2023-06-10'),
(7, 'Bond G', 'Bonds', 6000000, 6200000, '2023-07-15'),
(8, 'Mutual Fund H', 'Mutual Funds', 4000000, 4300000, '2023-08-20'),
(9, 'Real Estate I', 'Real Estate', 8000000, 8500000, '2023-09-25'),
(10, 'Gold J', 'Commodities', 12000000, 12500000, '2023-10-30'),
(11, 'Stock K', 'Stocks', 5500000, 6500000, '2023-01-20'),
(12, 'Bond L', 'Bonds', 9000000, 9300000, '2023-02-25'),
(13, 'Mutual Fund M', 'Mutual Funds', 2800000, 3300000, '2023-03-30'),
(14, 'Real Estate N', 'Real Estate', 7000000, 7800000, '2023-04-05'),
(15, 'Gold O', 'Commodities', 3500000, 3600000, '2023-05-10'),
(16, 'Stock P', 'Stocks', 16000000, 17500000, '2023-06-15'),
(17, 'Bond Q', 'Bonds', 6500000, 6700000, '2023-07-20'),
(18, 'Mutual Fund R', 'Mutual Funds', 4500000, 4800000, '2023-08-25'),
(19, 'Real Estate S', 'Real Estate', 8500000, 9000000, '2023-09-30'),
(20, 'Gold T', 'Commodities', 11000000, 11800000, '2023-10-21'),
```

Hình 17. Nhập dữ liệu bảng investments

❖ Thêm dữ liệu bảng budget:

```
-- Thêm dữ liệu bảng budget
INSERT INTO budget (user_id, month_year, category, allocated_amount, spent_amount) VALUES
(1, '2023-10-01', 'Food', 2000000, 1500000),
(2, '2023-10-01', 'Income', 10000000, 10000000),
(3, '2023-10-01', 'Shopping', 1000000, 500000),
(4, '2023-10-01', 'Investment', 3000000, 2000000),
(5, '2023-10-01', 'Bills', 1000000, 300000),
(6, '2023-10-01', 'Business', 15000000, 5000000),
(7, '2023-10-01', 'Housing', 5000000, 2000000),
(8, '2023-10-01', 'Travel', 3000000, 1000000),
(9, '2023-10-01', 'Education', 2000000, 1500000),
(10, '2023-10-01', 'Home', 2500000, 800000),
(11, '2023-10-01', 'Savings', 1500000, 500000),
(12, '2023-10-01', 'Cash', 1000000, 200000),
(13, '2023-10-01', 'Subscriptions', 500000, 100000),
(14, '2023-10-01', 'Investment', 3000000, 300000),
(15, '2023-10-01', 'Personal', 1000000, 400000),
(16, '2023-10-01', 'Business', 15000000, 2500000),
(17, '2023-10-01', 'Shared', 1000000, 600000),
```

Hình 18. Nhập dữ liệu bảng budget

Thêm dữ liệu bảng loans:

```
INSERT INTO loans (user_id, lender, loan_amount, interest_rate, start_date, due_date, status) VALUES
(1, 'Bank A', 10000000, 5.0, '2023-01-01', '2024-01-01', 'Active'),
(2, 'Bank B', 20000000, 6.0, '2023-02-01', '2025-02-01', 'Active'),
(3, 'Bank C', 5000000, 4.5, '2023-03-01', '2024-03-01', 'Paid'),
(4, 'Bank D', 15000000, 5.5, '2023-04-01', '2025-04-01', 'Active'),
(5, 'Bank E', 8000000, 4.0, '2023-05-01', '2024-05-01', 'Active'),
(6, 'Bank F', 25000000, 6.5, '2023-06-01', '2026-06-01', 'Active'),
(7, 'Bank G', 12000000, 5.0, '2023-07-01', '2025-07-01', 'Active'),
(8, 'Bank H', 18000000, 5.8, '2023-08-01', '2026-08-01', 'Active'),
(9, 'Bank I', 7000000, 4.2, '2023-09-01', '2024-09-01', 'Paid'),
(10, 'Bank J', 22000000, 6.2, '2023-10-01', '2027-10-01', 'Active'),
(11, 'Bank K', 11000000, 5.2, '2023-01-15', '2024-01-15', 'Active'),
(12, 'Bank L', 19000000, 6.1, '2023-02-15', '2025-02-15', 'Active'),
(13, 'Bank M', 6000000, 4.6, '2023-03-15', '2024-03-15', 'Paid'),
(14, 'Bank N', 16000000, 5.6, '2023-04-15', '2025-04-15', 'Active'),
(15, 'Bank O', 9000000, 4.1, '2023-05-15', '2024-05-15', 'Active'),
(16, 'Bank P', 26000000, 6.6, '2023-06-15', '2026-06-15', 'Active'),
(17, 'Bank Q', 13000000, 5.1, '2023-07-15', '2025-07-15', 'Active'),
(18, 'Bank R', 20000000, 5.9, '2023-08-15', '2026-08-15', 'Active'),
(19, 'Bank S', 7500000, 4.3, '2023-09-15', '2024-09-15', 'Paid'),
```

Hình 19. Nhập dữ liệu bảng loans

3.3.In bảng dữ liệu

❖ Dữ liệu bảng user:

```
SELECT * FROM users; -- người dùng
```

KẾT QUẢ

	user_id	username	password	email	full_name	date_of_birth
1	1	user1	password123	user1@example.com	Nguyen Van A	1990-01-01
2	2	user2	securepass	user2@example.com	Tran Thi B	1985-05-15
3	3	user3	pass1234	user3@example.com	Le Hoang C	1992-11-20
4	4	user4	strongpass	user4@example.com	Pham Thu D	1988-03-10
5	5	user5	pass5678	user5@example.com	Vo Minh E	1995-07-25
6	6	user6	pass9876	user6@example.com	Hoang Gia F	1987-12-05
7	7	user7	pass4321	user7@example.com	Doan Ngoc G	1991-09-18
8	8	user8	pass8765	user8@example.com	Bui Thanh H	1989-06-30
9	9	user9	pass1122	user9@example.com	Truong My I	1993-02-12
10	10	user10	pass3344	user10@example.com	Dang Van K	1986-08-22
11	11	user11	pass5566	user11@example.com	Ly Thi L	1994-04-08
12	12	user12	pass7788	user12@example.com	Vu Duc M	1990-10-28
13	13	user13	pass9900	user13@example.com	Phan Kim N	1987-01-15
14	14	user14	pass2233	user14@example.com	Ha Xuan O	1992-05-03
15	15	user15	pass4455	user15@example.com	Cao Thuy P	1988-11-19
16	16	user16	pass6677	user16@example.com	Dinh Quang Q	1995-03-07

Hình 20. Dữ liệu có trong bảng users

❖ Dữ liệu bảng accounts:

SELECT * FROM accounts; -- tài khoản
KẾT QUẢ

	account_id	user_id	account_name	bank_name	account_number	balance
1	1	1	Saving Account	Vietcombank	1234567890	5850000.00
2	2	2	Checking Account	Techcombank	0987654321	10000000.00
3	3	3	Salary Account	BIDV	1122334455	2500000.00
4	4	4	Investment Account	ACB	6677889900	7500000.00
5	5	5	Personal Account	Agribank	5544332211	3000000.00
6	6	6	Business Account	VPBank	9988776655	15000000.00
7	7	7	Joint Account	MBBank	4455667788	6000000.00
8	8	8	Travel Account	Sacombank	8899001122	4000000.00
9	9	9	Education Account	Shinhan Bank	2211009988	8000000.00
10	10	10	Home Account	Eximbank	7766554433	12000000.00
11	11	11	Saving Account 2	VietinBank	3344556677	5500000.00
12	12	12	Checking Account 2	TPBank	1122998877	9000000.00
13	13	13	Salary Account 2	OceanBank	6655443322	2800000.00
14	14	14	Investment Account 2	LienVietPostBank	9900112233	7000000.00
15	15	15	Personal Account 2	MSB	4433221100	3500000.00

Hình 21. Dữ liệu bảng accounts

❖ Dữ liệu bảng transactions:

SELECT * FROM transactions; -- giao dịch
KẾT QUẢ

	transaction_id	user_id	transaction_date	description	amount	category	account_id
1	1	1	2023-10-26	Grocery shopping	-150000.00	Food	1
2	2	2	2023-10-26	Salary deposit	1000000.00	Income	2
3	3	3	2023-10-25	Online purchase	-500000.00	Shopping	3
4	4	4	2023-10-25	Investment return	2000000.00	Investment	4
5	5	5	2023-10-24	Utility bill	-300000.00	Bills	5
6	6	6	2023-10-24	Business payment	5000000.00	Business	6
7	7	7	2023-10-23	Rent payment	-2000000.00	Housing	7
8	8	8	2023-10-23	Travel expenses	-1000000.00	Travel	8
9	9	9	2023-10-22	Tuition fee	-1500000.00	Education	9
10	10	10	2023-10-22	Home repair	-800000.00	Home	10
11	11	11	2023-10-21	Savings transfer	-500000.00	Savings	11
12	12	12	2023-10-21	ATM withdrawal	-200000.00	Cash	12
13	13	13	2023-10-20	Online subscription	-100000.00	Subscriptions	13
14	14	14	2023-10-20	Dividend payment	300000.00	Investment	14
15	15	15	2023-10-19	Personal expenses	-400000.00	Personal	15
16	16	16	2023-10-19	Business income	2500000.00	Business	16

Hình 22. Dữ liệu bảng transactions

❖ Dữ liệu bảng investments:

```
SELECT * FROM investments; -- đầu tư
```

KẾT QUẢ

	investment_id	user_id	investment_name	investment_type	initial_investment	current_value	purchase_date
1	1	1	Stock A	Stocks	5000000.00	12000000.00	2023-01-15
2	2	2	Bond B	Bonds	10000000.00	10500000.00	2023-02-20
3	3	3	Mutual Fund C	Mutual Funds	2500000.00	3000000.00	2023-03-25
4	4	4	Real Estate D	Real Estate	7500000.00	8000000.00	2023-04-30
5	5	5	Gold E	Commodities	3000000.00	3200000.00	2023-05-05
6	6	6	Stock F	Stocks	15000000.00	16000000.00	2023-06-10
7	7	7	Bond G	Bonds	6000000.00	6200000.00	2023-07-15
8	8	8	Mutual Fund H	Mutual Funds	4000000.00	4300000.00	2023-08-20
9	9	9	Real Estate I	Real Estate	8000000.00	8500000.00	2023-09-25
10	10	10	Gold J	Commodities	12000000.00	12500000.00	2023-10-30
11	11	11	Stock K	Stocks	5500000.00	6500000.00	2023-01-20
12	12	12	Bond L	Bonds	9000000.00	9300000.00	2023-02-25
13	13	13	Mutual Fund M	Mutual Funds	2800000.00	3300000.00	2023-03-30
14	14	14	Real Estate N	Real Estate	7000000.00	7800000.00	2023-04-05
15	15	15	Gold O	Commodities	3500000.00	3600000.00	2023-05-10
16	16	16	Stock P	Stocks	16000000.00	17500000.00	2023-06-15
17	17	17	Bond Q	Bonds	6500000.00	6700000.00	2023-07-20

Hình 23. Dữ liệu bảng investments.

❖ Dữ liệu bảng budget:

```
SELECT * FROM budget; -- ngân sách
```

KẾT QUẢ

	budget_id	user_id	month_year	category	allocated_amount	spent_amount
1	1	1	2023-10-01	Food	2000000.00	3000000.00
2	2	2	2023-10-01	Income	10000000.00	10000000.00
3	3	3	2023-10-01	Shopping	1000000.00	500000.00
4	4	4	2023-10-01	Investment	3000000.00	2000000.00
5	5	5	2023-10-01	Bills	1000000.00	300000.00
6	6	6	2023-10-01	Business	15000000.00	5000000.00
7	7	7	2023-10-01	Housing	5000000.00	2000000.00
8	8	8	2023-10-01	Travel	3000000.00	1000000.00
9	9	9	2023-10-01	Education	2000000.00	1500000.00
10	10	10	2023-10-01	Home	2500000.00	800000.00
11	11	11	2023-10-01	Savings	1500000.00	500000.00
12	12	12	2023-10-01	Cash	1000000.00	200000.00
13	13	13	2023-10-01	Subscriptions	500000.00	100000.00
14	14	14	2023-10-01	Investment	3000000.00	300000.00
15	15	15	2023-10-01	Personal	1000000.00	400000.00
16	16	16	2023-10-01	Business	15000000.00	2500000.00
17	17	17	2023-10-01	Food	2000000.00	3000000.00

Hình 24. Dữ liệu bảng budget

❖ Dữ liệu bảng loans:

```
SELECT * FROM loans; -- khoản vay
```

KẾT QUẢ

	loan_id	user_id	lender	loan_amount	interest_rate	start_date	due_date	status
1	1	1	Bank A	10000000.00	5.00	2023-01-01	2024-02-01	Paid
2	2	2	Bank B	20000000.00	6.00	2023-02-01	2025-02-01	Active
3	3	3	Bank C	5000000.00	4.50	2023-03-01	2024-03-01	Paid
4	4	4	Bank D	15000000.00	5.50	2023-04-01	2025-04-01	Active
5	5	5	Bank E	8000000.00	4.00	2023-05-01	2024-05-01	Active
6	6	6	Bank F	25000000.00	6.50	2023-06-01	2026-06-01	Active
7	7	7	Bank G	12000000.00	5.00	2023-07-01	2025-07-01	Active
8	8	8	Bank H	18000000.00	5.80	2023-08-01	2026-08-01	Active
9	9	9	Bank I	7000000.00	4.20	2023-09-01	2024-09-01	Paid
10	10	10	Bank J	22000000.00	6.20	2023-10-01	2027-10-01	Active
11	11	11	Bank K	11000000.00	5.20	2023-01-15	2024-01-15	Active
12	12	12	Bank L	19000000.00	6.10	2023-02-15	2025-02-15	Active
13	13	13	Bank M	6000000.00	4.60	2023-03-15	2024-03-15	Paid
14	14	14	Bank N	16000000.00	5.60	2023-04-15	2025-04-15	Active
15	15	15	Bank O	9000000.00	4.10	2023-05-15	2024-05-15	Active
16	16	16	Bank P	26000000.00	6.60	2023-06-15	2026-06-15	Active
17	17	17	Bank Q	13000000.00	5.10	2023-07-15	2025-07-15	Active

Hình 25. Dữ liệu bảng loans

CHƯƠNG 4. XÂY DỰNG CÁC VIEW

4.1. Tạo các view

❖ View kết hợp thông tin từ bảng users và accounts để hiển thị tên người dùng:

```
-- View này kết hợp thông tin từ bảng users và accounts để hiển thị tên người dùng,  
-- tên đầy đủ, tên tài khoản, tên ngân hàng, số tài khoản và số dư.  
CREATE VIEW UserAccounts AS  
SELECT u.user_id, u.username, u.full_name, a.account_name, a.bank_name, a.account_number, a.balance  
FROM users u  
JOIN accounts a ON u.user_id = a.user_id;  
GO
```

Hình 26. Code view UserAccounts

- CREATE VIEW UserAccounts AS: Lệnh này tạo một view có tên là UserAccounts.
- SELECT u.user_id, u.username, u.full_name, a.account_name, a.bank_name, a.account_number, a.balance: Lệnh này chọn các cột sau:
 - u.user_id: Mã định danh người dùng từ bảng users.
 - u.username: Tên người dùng từ bảng users.
 - u.full_name: Tên đầy đủ của người dùng từ bảng users.
 - a.account_name: Tên tài khoản từ bảng accounts.
 - a.bank_name: Tên ngân hàng từ bảng accounts.
 - a.account_number: Số tài khoản từ bảng accounts.
 - a.balance: Số dư tài khoản từ bảng accounts.
- FROM users u JOIN accounts a ON u.user_id = a.user_id: Lệnh này kết hợp dữ liệu từ bảng users (được đặt bí danh là u) và bảng accounts (được đặt bí danh là a) dựa trên cột user_id chung.

❖ View giao dịch theo người dùng

```
-- View giao dịch theo người dùng  
-- View này hiển thị các giao dịch của mỗi người dùng, bao gồm ngày giao dịch,  
-- mô tả, số tiền và danh mục.  
CREATE VIEW UserTransactions AS  
SELECT u.username, t.transaction_date, t.description, t.amount, t.category  
FROM users u  
JOIN transactions t ON u.user_id = t.user_id;  
GO
```

Hình 27. Code view UserTransactions

- Tương tự như UserAccounts, view này kết hợp dữ liệu từ bảng users và transactions dựa trên user_id.
- Nó chọn username, transaction_date, description, amount, và category từ hai bảng.

❖ View tổng số dư tài khoản theo người dùng

```
-- =====  
-- View tổng số dư tài khoản theo người dùng  
-- View này tính toán tổng số dư của tất cả các tài khoản của mỗi người dùng.  
CREATE VIEW UserTotalBalance AS  
SELECT u.username, SUM(a.balance) AS total_balance  
FROM users u  
JOIN accounts a ON u.user_id = a.user_id  
GROUP BY u.username;  
GO
```

Hình 28. Code view UserTotalBalance

- View này sử dụng hàm SUM(a.balance) để tính tổng số dư của tất cả các tài khoản của mỗi người dùng.
- GROUP BY u.username nhóm kết quả theo tên người dùng, đảm bảo rằng tổng số dư được tính cho từng người dùng riêng biệt.

❖ View đầu tư theo người dùng

```
-- =====  
-- View đầu tư theo người dùng  
-- View này hiển thị thông tin đầu tư của mỗi người dùng, bao gồm tên đầu tư,  
-- loại đầu tư, số tiền đầu tư ban đầu, giá trị hiện tại và ngày mua.  
CREATE VIEW UserInvestments AS  
SELECT u.username, i.investment_name, i.investment_type, i.initial_investment, i.current_value, i.purchase_date  
FROM users u  
JOIN investments i ON u.user_id = i.user_id;  
GO
```

Hình 29. Code view UserInvestments.

- Kết hợp dữ liệu từ bảng users và investments dựa trên user_id.
- Chọn thông tin đầu tư của từng người dùng.

❖ View ngân sách theo người dùng và danh mục

```
-- =====  
-- View ngân sách theo người dùng và danh mục  
-- View này hiển thị thông tin ngân sách của mỗi người dùng, bao gồm tháng/năm,  
-- danh mục, số tiền được phân bổ và số tiền đã chi tiêu.  
CREATE VIEW UserBudgets AS  
SELECT u.username, b.month_year, b.category, b.allocated_amount, b.spent_amount  
FROM users u  
JOIN budget b ON u.user_id = b.user_id;  
GO
```

Hình 30. Code view UserBudgets

- Kết hợp dữ liệu từ bảng users và budget dựa trên user_id.
- Chọn thông tin ngân sách của từng người dùng.

❖ View khoản vay theo người dùng

```
-- View khoản vay theo người dùng
-- View này hiển thị thông tin khoản vay của mỗi người dùng, bao gồm người cho vay,
-- số tiền vay, lãi suất, ngày bắt đầu, ngày đáo hạn và trạng thái.
CREATE VIEW UserLoans AS
SELECT u.username, l.lender, l.loan_amount, l.interest_rate, l.start_date, l.due_date, l.status
FROM users u
JOIN loans l ON u.user_id = l.user_id;
GO
```

Hình 31. Code View UserLoans

- Kết hợp dữ liệu từ bảng users và loans dựa trên user_id.
- Chọn thông tin khoản vay của từng người dùng.

❖ View giao dịch theo tài khoản

```
-- =====
-- View giao dịch theo tài khoản
-- View này hiển thị các giao dịch của mỗi tài khoản.
CREATE VIEW AccountTransactions AS
SELECT a.account_name, t.transaction_date, t.description, t.amount, t.category
FROM accounts a
JOIN transactions t ON a.account_id = t.account_id;
GO
```

Hình 32. View AccountTransactions.

- Kết hợp dữ liệu từ bảng accounts và transactions dựa trên account_id.
- Chọn thông tin giao dịch của từng tài khoản.

❖ View đầu tư sinh lời

```
-- =====
-- View đầu tư sinh lời
-- View này chỉ hiển thị các khoản đầu tư có giá trị hiện tại lớn hơn giá trị đầu tư ban đầu.
CREATE VIEW ProfitableInvestments AS
SELECT u.username, i.investment_name, i.investment_type, i.initial_investment, i.current_value, i.purchase_date
FROM users u
JOIN investments i ON u.user_id = i.user_id
WHERE i.current_value > i.initial_investment;
GO
```

Hình 33. Code view ProfitableInvestments

- Tương tự như UserInvestments, nhưng có thêm mệnh đề WHERE i.current_value > i.initial_investment để lọc chỉ các khoản đầu tư sinh lời.

❖ View ngân sách còn lại

```
-- =====
-- View ngân sách còn lại
-- View này tính toán và hiển thị số tiền còn lại trong ngân sách của mỗi người dùng cho từng danh mục.
CREATE VIEW RemainingBudget AS
SELECT u.username, b.month_year, b.category, (b.allocated_amount - b.spent_amount) AS remaining_amount
FROM users u
JOIN budget b ON u.user_id = b.user_id;
GO
```

Hình 34. Code View RemainingBudget.

- Tính toán số tiền còn lại trong ngân sách bằng cách lấy allocated_amount trừ đi spent_amount.

❖ View khoản vay đang hoạt động

```
-- =====
-- View khoản vay đang hoạt động
-- View này chỉ hiển thị các khoản vay đang trong trạng thái "Active".
CREATE VIEW ActiveLoans AS
SELECT u.username, l.lender, l.loan_amount, l.interest_rate, l.start_date, l.due_date
FROM users u
JOIN loans l ON u.user_id = l.user_id
WHERE l.status = 'Active';
GO
-- =====
```

Hình 35. Code View ActiveLoans.

- Lọc chỉ các khoản vay có trạng thái "Active".

4.2. In thông tin view

```
-- =====
-- Các câu lệnh để chạy VIEW
SELECT * FROM UserAccounts;
```

.09 %

	user_id	username	full_name	account_name	bank_name	account_number	balance
1	1	user1	Nguyen Van A	Saving Account	Vietcombank	1234567890	5850000.00
2	2	user2	Tran Thi B	Checking Account	Techcombank	0987654321	10000000.00
3	3	user3	Le Hoang C	Salary Account	BIDV	1122334455	2500000.00
4	4	user4	Pham Thu D	Investment Account	ACB	6677889900	7500000.00
5	5	user5	Vo Minh E	Personal Account	Agribank	5544332211	3000000.00
6	6	user6	Hoang Gia F	Business Account	VPBank	9988776655	15000000.00
7	7	user7	Doan Ngoc G	Joint Account	MBBank	4455667788	6000000.00
8	8	user8	Bui Thanh H	Travel Account	Sacombank	8899001122	4000000.00
9	9	user9	Truong My I	Education Account	Shinhan Bank	2211009988	8000000.00
10	10	user10	Dang Van K	Home Account	Eximbank	7766554433	12000000.00
11	11	user11	Ly Thi L	Saving Account 2	VietinBank	3344556677	5500000.00
12	12	user12	Vu Duc M	Checking Account 2	TPBank	1122998877	9000000.00
13	13	user13	Phan Kim N	Salary Account 2	OceanBank	6655443322	2800000.00
14	14	user14	Ha Xuan O	Investment Account 2	LienVietPostBank	9900112233	7000000.00
15	15	user15	Cao Thuy P	Personal Account 2	MSB	4433221100	3500000.00
16	16	user16	Dinh Quang Q	Business Account 2	Bac A Bank	8877665544	16000000.00
17	17	user17	Lam Bao R	Joint Account 2	GPBank	2233445566	6500000.00

Hình 36. In dữ liệu view UserAccounts

SELECT * FROM UserTransactions;

109 %

Results Messages

	username	transaction_date	description	amount	category
1	user1	2023-10-26	Grocery shopping	-150000.00	Food
2	user2	2023-10-26	Salary deposit	1000000.00	Income
3	user3	2023-10-25	Online purchase	-500000.00	Shopping
4	user4	2023-10-25	Investment return	2000000.00	Investment
5	user5	2023-10-24	Utility bill	-300000.00	Bills
6	user6	2023-10-24	Business payment	5000000.00	Business
7	user7	2023-10-23	Rent payment	-2000000.00	Housing
8	user8	2023-10-23	Travel expenses	-1000000.00	Travel
9	user9	2023-10-22	Tuition fee	-1500000.00	Education
10	user10	2023-10-22	Home repair	-800000.00	Home
11	user11	2023-10-21	Savings transfer	-500000.00	Savings
12	user12	2023-10-21	ATM withdrawal	-200000.00	Cash
13	user13	2023-10-20	Online subscription	-100000.00	Subscriptions
14	user14	2023-10-20	Dividend payment	300000.00	Investment
15	user15	2023-10-19	Personal expenses	-400000.00	Personal
16	user16	2023-10-19	Business income	2500000.00	Business
17	user17	2023-10-18	Joint expense	-600000.00	Shared
18	user18	2023-10-18	Travel booking	-1200000.00	Travel
19	user19	2023-10-17	Education supplies	-700000.00	Education
20	user20	2023-10-17	Home improvem	-1500000.00	Home

Hình 37. In dữ liệu view UserTransactions.

SELECT * FROM UserTotalBalance;

109 %

Results Messages

	username	total_balance
1	user1	6855000.00
2	user10	12000000.00
3	user11	5500000.00
4	user12	9000000.00
5	user13	2800000.00
6	user14	7000000.00
7	user15	3500000.00
8	user16	16000000.00
9	user17	6500000.00
10	user18	4500000.00
11	user19	8500000.00
12	user2	10010000.00
13	user20	11000000.00
14	user21	5200000.00
15	user22	9500000.00
16	user23	2600000.00

Hình 38. In dữ liệu view UserTotalBalance.

SELECT * FROM UserInvestments;

109 %

Results Messages

	username	investment_name	investment_type	initial_investment	current_value	purchase_date
1	user1	Stock A	Stocks	5000000.00	12000000.00	2023-01-15
2	user2	Bond B	Bonds	10000000.00	10500000.00	2023-02-20
3	user3	Mutual Fund C	Mutual Funds	2500000.00	3000000.00	2023-03-25
4	user4	Real Estate D	Real Estate	7500000.00	8000000.00	2023-04-30
5	user5	Gold E	Commodities	3000000.00	3200000.00	2023-05-05
6	user6	Stock F	Stocks	15000000.00	16000000.00	2023-06-10
7	user7	Bond G	Bonds	6000000.00	6200000.00	2023-07-15
8	user8	Mutual Fund H	Mutual Funds	4000000.00	4300000.00	2023-08-20
9	user9	Real Estate I	Real Estate	8000000.00	8500000.00	2023-09-25
10	user10	Gold J	Commodities	12000000.00	12500000.00	2023-10-30
11	user11	Stock K	Stocks	5500000.00	6500000.00	2023-01-20
12	user12	Bond L	Bonds	9000000.00	9300000.00	2023-02-25
13	user13	Mutual Fund M	Mutual Funds	2800000.00	3300000.00	2023-03-30
14	user14	Real Estate N	Real Estate	7000000.00	7800000.00	2023-04-05
15	user15	Gold O	Commodities	3500000.00	3600000.00	2023-05-10
16	user16	Stock P	Stocks	16000000.00	17500000.00	2023-06-15
17	user17	Bond Q	Bonds	6500000.00	6700000.00	2023-07-20
18	user18	Mutual Fund R	Mutual Funds	4500000.00	4800000.00	2023-08-25
19	user19	Real Estate S	Real Estate	8500000.00	9000000.00	2023-09-30
20	user20	Gold T	Commodities	11000000.00	11800000.00	2023-10-21
21	user1	Stock A	Stocks	5000000.00	6000000.00	2023-01-15
22	user2	Bond B	Bonds	10000000.00	10500000.00	2023-02-20
23	user3	Mutual Fund C	Mutual Funds	2500000.00	3000000.00	2023-03-25
24	user4	Real Estate D	Real Estate	7500000.00	8000000.00	2023-04-30
25	user5	Gold E	Commodities	3000000.00	3200000.00	2023-05-05

Hình 39. In dữ liệu view UserInvestments.

SELECT * FROM UserBudgets;

109 %

Results Messages

	username	month_year	category	allocated_amount	spent_amount
1	user1	2023-10-01	Food	2000000.00	3000000.00
2	user2	2023-10-01	Income	10000000.00	10000000.00
3	user3	2023-10-01	Shopping	1000000.00	500000.00
4	user4	2023-10-01	Investment	3000000.00	2000000.00
5	user5	2023-10-01	Bills	1000000.00	300000.00
6	user6	2023-10-01	Business	15000000.00	5000000.00
7	user7	2023-10-01	Housing	5000000.00	2000000.00
8	user8	2023-10-01	Travel	3000000.00	1000000.00
9	user9	2023-10-01	Education	2000000.00	1500000.00
10	user10	2023-10-01	Home	2500000.00	800000.00
11	user11	2023-10-01	Savings	1500000.00	500000.00
12	user12	2023-10-01	Cash	1000000.00	200000.00
13	user13	2023-10-01	Subscriptions	500000.00	100000.00
14	user14	2023-10-01	Investment	3000000.00	300000.00
15	user15	2023-10-01	Personal	1000000.00	400000.00
16	user16	2023-10-01	Business	15000000.00	2500000.00
17	user17	2023-10-01	Shared	1000000.00	600000.00
18	user18	2023-10-01	Travel	3000000.00	1200000.00
19	user19	2023-10-01	Education	2000000.00	700000.00

Hình 40 In thông tin view UserBudgets.

SELECT * FROM UserLoans;

109 %

Results Messages

	username	lender	loan_amount	interest_rate	start_date	due_date	status
1	user1	Bank A	10000000.00	5.00	2023-01-01	2024-02-01	Paid
2	user2	Bank B	20000000.00	6.00	2023-02-01	2025-02-01	Active
3	user3	Bank C	5000000.00	4.50	2023-03-01	2024-03-01	Paid
4	user4	Bank D	15000000.00	5.50	2023-04-01	2025-04-01	Active
5	user5	Bank E	8000000.00	4.00	2023-05-01	2024-05-01	Active
6	user6	Bank F	25000000.00	6.50	2023-06-01	2026-06-01	Active
7	user7	Bank G	12000000.00	5.00	2023-07-01	2025-07-01	Active
8	user8	Bank H	18000000.00	5.80	2023-08-01	2026-08-01	Active
9	user9	Bank I	7000000.00	4.20	2023-09-01	2024-09-01	Paid
10	user10	Bank J	22000000.00	6.20	2023-10-01	2027-10-01	Active
11	user11	Bank K	11000000.00	5.20	2023-01-15	2024-01-15	Active
12	user12	Bank L	19000000.00	6.10	2023-02-15	2025-02-15	Active
13	user13	Bank M	6000000.00	4.60	2023-03-15	2024-03-15	Paid
14	user14	Bank N	16000000.00	5.60	2023-04-15	2025-04-15	Active
15	user15	Bank O	9000000.00	4.10	2023-05-15	2024-05-15	Active
16	user16	Bank P	26000000.00	6.60	2023-06-15	2026-06-15	Active
17	user17	Bank Q	13000000.00	5.10	2023-07-15	2025-07-15	Active
18	user18	Bank R	20000000.00	5.90	2023-08-15	2026-08-15	Active
19	user19	Bank S	7500000.00	4.30	2023-09-15	2024-09-15	Paid

Hình 41. In thông tin view UserLoans.

SELECT * FROM AccountTransactions;

109 %

Results Messages

	account_name	transaction_date	description	amount	category
1	Saving Account	2023-10-26	Grocery shopping	-150000.00	Food
2	Checking Account	2023-10-26	Salary deposit	1000000.00	Income
3	Salary Account	2023-10-25	Online purchase	-500000.00	Shopping
4	Investment Account	2023-10-25	Investment return	2000000.00	Investment
5	Personal Account	2023-10-24	Utility bill	-300000.00	Bills
6	Business Account	2023-10-24	Business payment	5000000.00	Business
7	Joint Account	2023-10-23	Rent payment	-2000000.00	Housing
8	Travel Account	2023-10-23	Travel expenses	-1000000.00	Travel
9	Education Account	2023-10-22	Tuition fee	-1500000.00	Education
10	Home Account	2023-10-22	Home repair	-800000.00	Home
11	Saving Account 2	2023-10-21	Savings transfer	-500000.00	Savings
12	Checking Account 2	2023-10-21	ATM withdrawal	-200000.00	Cash
13	Salary Account 2	2023-10-20	Online subscription	-100000.00	Subscriptions
14	Investment Account 2	2023-10-20	Dividend payment	300000.00	Investment
15	Personal Account 2	2023-10-19	Personal expenses	-400000.00	Personal
16	Business Account 2	2023-10-19	Business income	2500000.00	Business
17	Joint Account 2	2023-10-18	Joint expense	-600000.00	Shared
18	Travel Account 2	2023-10-18	Travel booking	-1200000.00	Travel
19	Education Account 2	2023-10-17	Education supplies	-700000.00	Education
20	Home Account 2	2023-10-17	Home improvement	-1500000.00	Home

Hình 42. in thông tin view AccountTransactions.

SELECT * FROM ProfitableInvestments;

	username	investment_name	investment_type	initial_investment	current_value	purchase_date
1	user1	Stock A	Stocks	5000000.00	12000000.00	2023-01-15
2	user2	Bond B	Bonds	10000000.00	10500000.00	2023-02-20
3	user3	Mutual Fund C	Mutual Funds	2500000.00	3000000.00	2023-03-25
4	user4	Real Estate D	Real Estate	7500000.00	8000000.00	2023-04-30
5	user5	Gold E	Commodities	3000000.00	3200000.00	2023-05-05
6	user6	Stock F	Stocks	15000000.00	16000000.00	2023-06-10
7	user7	Bond G	Bonds	6000000.00	6200000.00	2023-07-15
8	user8	Mutual Fund H	Mutual Funds	4000000.00	4300000.00	2023-08-20
9	user9	Real Estate I	Real Estate	8000000.00	8500000.00	2023-09-25
10	user10	Gold J	Commodities	12000000.00	12500000.00	2023-10-30
11	user11	Stock K	Stocks	5500000.00	6500000.00	2023-01-20
12	user12	Bond L	Bonds	9000000.00	9300000.00	2023-02-25
13	user13	Mutual Fund M	Mutual Funds	2800000.00	3300000.00	2023-03-30
14	user14	Real Estate N	Real Estate	7000000.00	7800000.00	2023-04-05
15	user15	Gold O	Commodities	3500000.00	3600000.00	2023-05-10
16	user16	Stock P	Stocks	16000000.00	17500000.00	2023-06-15
17	user17	Bond Q	Bonds	6500000.00	6700000.00	2023-07-20

Hình 43. In thông tin view ProfitableInvestments.

SELECT * FROM RemainingBudget;

	username	month_year	category	remaining_amount
1	user1	2023-10-01	Food	-1000000.00
2	user2	2023-10-01	Income	0.00
3	user3	2023-10-01	Shopping	500000.00
4	user4	2023-10-01	Investment	1000000.00
5	user5	2023-10-01	Bills	700000.00
6	user6	2023-10-01	Business	10000000.00
7	user7	2023-10-01	Housing	3000000.00
8	user8	2023-10-01	Travel	2000000.00
9	user9	2023-10-01	Education	500000.00
10	user10	2023-10-01	Home	1700000.00
11	user11	2023-10-01	Savings	1000000.00
12	user12	2023-10-01	Cash	800000.00
13	user13	2023-10-01	Subscriptions	400000.00
14	user14	2023-10-01	Investment	2700000.00
15	user15	2023-10-01	Personal	600000.00

Hình 44. In thông tin view RemainingBudget.

SELECT * FROM ActiveLoans;

	username	lender	loan_amount	interest_rate	start_date	due_date
1	user2	Bank B	20000000.00	6.00	2023-02-01	2025-02-01
2	user4	Bank D	15000000.00	5.50	2023-04-01	2025-04-01
3	user5	Bank E	8000000.00	4.00	2023-05-01	2024-05-01
4	user6	Bank F	25000000.00	6.50	2023-06-01	2026-06-01
5	user7	Bank G	12000000.00	5.00	2023-07-01	2025-07-01
6	user8	Bank H	18000000.00	5.80	2023-08-01	2026-08-01
7	user10	Bank J	22000000.00	6.20	2023-10-01	2027-10-01
8	user11	Bank K	11000000.00	5.20	2023-01-15	2024-01-15
9	user12	Bank L	19000000.00	6.10	2023-02-15	2025-02-15
10	user14	Bank N	16000000.00	5.60	2023-04-15	2025-04-15
11	user15	Bank O	9000000.00	4.10	2023-05-15	2024-05-15
12	user16	Bank P	26000000.00	6.60	2023-06-15	2026-06-15
13	user17	Bank Q	13000000.00	5.10	2023-07-15	2025-07-15
14	user18	Bank R	20000000.00	5.90	2023-08-15	2026-08-15
15	user20	Bank T	23000000.00	6.30	2023-10-15	2027-10-15

Hình 45. In thông tin view ActiveLoans.

CHƯƠNG 5. XÂY DỰNG CÁC PROCEDURE

5.1. Thêm người dùng mới

```
CREATE PROCEDURE AddUser
    @username NVARCHAR(100),
    @password NVARCHAR(255),
    @email NVARCHAR(255),
    @full_name NVARCHAR(255),
    @date_of_birth DATE
AS
BEGIN
    INSERT INTO users (username, password, email, full_name, date_of_birth)
    VALUES (@username, @password, @email, @full_name, @date_of_birth);
END;
GO
DROP PROCEDURE AddUser
```

Commands completed successfully.

Completion time: 2025-03-03T09:04:35.6948317+07:00

Hình 46. procedure thêm người mới

- CREATE PROCEDURE AddUser: Tạo một thủ tục lưu trữ (Stored Procedure) có tên AddUser.
- Khai báo tham số:
 - @username NVARCHAR(100): Tên người dùng (kiểu chuỗi Unicode, tối đa 100 ký tự).
 - @password NVARCHAR(255): Mật khẩu của người dùng (kiểu chuỗi Unicode, tối đa 255 ký tự).
 - @email NVARCHAR(255): Email của người dùng.
 - @full_name NVARCHAR(255): Họ và tên của người dùng.
 - @date_of_birth DATE: Ngày sinh của người dùng.
- Lệnh INSERT INTO users (...) VALUES (...):
- Chèn một dòng mới vào bảng users với các giá trị được truyền vào từ các tham số của Stored Procedure.
- Xóa Stored Procedure: DROP PROCEDURE AddUser;

5.2. Cập nhật thông tin người dùng

```
3 CREATE PROCEDURE UpdateUser
    @user_id INT,
    @email NVARCHAR(255),
    @full_name NVARCHAR(255)
AS
3 BEGIN
3     UPDATE users
    SET email = @email, full_name = @full_name
    WHERE user_id = @user_id;
- END;
- GO

Commands completed successfully.

Completion time: 2025-03-03T09:06:04.4700323+07:00
```

Hình 47. procedure cập nhật người dùng

- CREATE PROCEDURE UpdateUser: Tạo một Stored Procedure có tên UpdateUser.
- Danh sách tham số đầu vào:
 - @user_id INT: ID của người dùng cần cập nhật.
 - @email NVARCHAR(255): Email mới của người dùng.
 - @full_name NVARCHAR(255): Họ và tên mới của người dùng.
- Câu lệnh UPDATE dùng để cập nhật dữ liệu trong bảng users.
- Cập nhật cột email và full_name với các giá trị mới được truyền vào từ tham số @email và @full_name.
- Điều kiện WHERE user_id = @user_id giúp đảm bảo chỉ cập nhật dữ liệu cho người dùng có user_id tương ứng.
- END;; Đánh dấu kết thúc nội dung của Stored Procedure.
- GO: Xác nhận việc thực thi lệnh trong SQL Server.

5.3.Xóa người dùng (và các dữ liệu liên quan)

```
CREATE PROCEDURE DeleteUser
    @user_id INT
AS
BEGIN
    DELETE FROM transactions WHERE user_id = @user_id;
    DELETE FROM accounts WHERE user_id = @user_id;
    DELETE FROM investments WHERE user_id = @user_id;
    DELETE FROM budget WHERE user_id = @user_id;
    DELETE FROM loans WHERE user_id = @user_id;
    DELETE FROM users WHERE user_id = @user_id;
END;
GO

Commands completed successfully.

Completion time: 2025-03-03T09:07:07.2786839+07:00
```

Hình 48. Xóa người dùng procedure

- CREATE PROCEDURE DeleteUser: Tạo một Stored Procedure có tên DeleteUser.
- Tham số đầu vào:
 - @user_id INT: ID của người dùng cần xóa.
- Xóa tất cả dữ liệu liên quan đến user_id trong các bảng khác:
 - transactions: Giao dịch của người dùng.
 - accounts: Tài khoản liên quan.
 - investments: Khoản đầu tư của người dùng.
 - budget: Ngân sách do người dùng quản lý.
 - loans: Khoản vay của người dùng.
- Lý do phải xóa trước khi xóa người dùng:
 - Nếu có ràng buộc khóa ngoại (FOREIGN KEY) giữa users và các bảng trên, ta phải xóa dữ liệu liên quan trước khi xóa người dùng, nếu không sẽ xảy ra lỗi ràng buộc.
- Xóa người dùng trong bảng : DELETE FROM users WHERE user_id = @user_id;
 - Sau khi tất cả dữ liệu liên quan đã bị xóa, thực hiện xóa người dùng khỏi bảng users.
- Kết thúc Stored Procedure :
END;

5.4. Thêm tài khoản ngân hàng

```
CREATE PROCEDURE AddAccount
    @user_id INT,
    @account_name NVARCHAR(255),
    @bank_name NVARCHAR(255),
    @account_number NVARCHAR(50),
    @balance DECIMAL(18,2)
AS
BEGIN
    INSERT INTO accounts (user_id, account_name, bank_name, account_number, balance)
    VALUES (@user_id, @account_name, @bank_name, @account_number, @balance);
END;
GO
```

Hình 49. procedure thêm tài khoản ngân hàng

- CREATE PROCEDURE AddAccount: Tạo một Stored Procedure có tên AddAccount.
- Danh sách tham số đầu vào:
 - @user_id INT: ID của người dùng sở hữu tài khoản.
 - @account_name NVARCHAR(255): Tên tài khoản (ví dụ: "Tài khoản tiết kiệm").
 - @bank_name NVARCHAR(255): Tên ngân hàng.
 - @account_number NVARCHAR(50): Số tài khoản ngân hàng.
 - @balance DECIMAL(18,2): Số dư ban đầu của tài khoản.

- Chèn dữ liệu vào bảng accounts

INSERT INTO accounts (user_id, account_name, bank_name, account_number, balance)

VALUES (@user_id, @account_name, @bank_name, @account_number, @balance);

- Thêm một tài khoản mới vào bảng accounts.
- Các giá trị cột sẽ nhận dữ liệu từ tham số đầu vào.
- Kết thúc Stored Procedure

END;

GO

5.5. Gửi tiền vào tài khoản

```
CREATE PROCEDURE DepositMoney
    @account_id INT,
    @amount DECIMAL(18,2)
AS
BEGIN
    UPDATE accounts
    SET balance = balance + @amount
    WHERE account_id = @account_id;
END;
GO
```

Commands completed successfully.

Completion time: 2025-03-03T09:19:46.7504367+07:00

Hình 50. procedure gửi tiền vào tài khoản

- CREATE PROCEDURE DepositMoney: Tạo một Stored Procedure có tên DepositMoney.
- Danh sách tham số đầu vào:
 - @account_id INT: ID của tài khoản cần nạp tiền.
 - @amount DECIMAL(18,2): Số tiền cần nạp (định dạng số thập phân với tối đa 18 chữ số, trong đó 2 chữ số sau dấu thập phân).
- Cập nhật số dư (balance) của tài khoản có account_id được chỉ định.
- Cộng thêm số tiền nạp (@amount) vào số dư hiện tại.
- END;: Kết thúc Stored Procedure.
- GO: Thực thi trong SQL Server.

5.6. Rút tiền từ tài khoản

```
CREATE PROCEDURE WithdrawMoney
    @account_id INT,
    @amount DECIMAL(18,2)
AS
BEGIN
    IF EXISTS (SELECT 1 FROM accounts WHERE account_id = @account_id AND balance >= @amount)
    BEGIN
        UPDATE accounts
        SET balance = balance - @amount
        WHERE account_id = @account_id;
    END
    ELSE
    BEGIN
        PRINT 'Số dư không đủ';
    END
END;
GO

Commands completed successfully.

Completion time: 2025-03-03T09:25:46.0139545+07:00
```

Hình 51. procedure rút tiền từ tài khoản

- CREATE PROCEDURE WithdrawMoney: Tạo một Stored Procedure có tên WithdrawMoney.
- Danh sách tham số đầu vào:
 - @account_id INT: ID của tài khoản cần rút tiền.
 - @amount DECIMAL(18,2): Số tiền muốn rút (có tối đa 2 chữ số thập phân).
- Kiểm tra xem tài khoản có tồn tại không (account_id có trong bảng accounts).
- Kiểm tra số dư có đủ để rút không (balance >= @amount).
- Nếu tài khoản tồn tại và có đủ tiền, tiếp tục thực hiện rút tiền.
- Nếu điều kiện kiểm tra số dư hợp lệ, cập nhật bảng accounts bằng cách trừ số tiền rút (@amount) khỏi số dư (balance).
- Nếu số dư không đủ, in ra thông báo "Số dư không đủ".
- Lệnh PRINT chỉ hiển thị thông báo trong SQL Server, không chặn hoàn toàn giao dịch.
- END;: Kết thúc Stored Procedure.
- GO: Thực thi trong SQL Server.

5.7. Thêm giao dịch mới



```
CREATE PROCEDURE AddTransaction
    @user_id INT,
    @transaction_date DATE,
    @description NVARCHAR(500),
    @amount DECIMAL(18,2),
    @category NVARCHAR(100),
    @account_id INT
AS
BEGIN
    INSERT INTO transactions (user_id, transaction_date, description, amount, category, account_id)
    VALUES (@user_id, @transaction_date, @description, @amount, @category, @account_id);
END;
GO
```

parameter @transaction_date date

Commands completed successfully.

Completion time: 2025-03-03T09:29:55.0842343+07:00

Hình 52. Procedure thêm giao dịch mới

- CREATE PROCEDURE AddTransaction: Tạo Stored Procedure với tên AddTransaction.
- Danh sách tham số đầu vào:
 - @user_id INT: ID của người thực hiện giao dịch.
 - @transaction_date DATE: Ngày thực hiện giao dịch.
 - @description NVARCHAR(500): Mô tả chi tiết giao dịch.
 - @amount DECIMAL(18,2): Số tiền giao dịch (có tối đa 18 chữ số, trong đó 2 chữ số sau dấu thập phân).
 - @category NVARCHAR(100): Danh mục của giao dịch (ví dụ: "Ăn uống", "Mua sắm", "Tiết kiệm").
 - @account_id INT: ID của tài khoản liên quan đến giao dịch.
- Chèn một dòng mới vào bảng transactions với dữ liệu từ các tham số truyền vào.
- Giá trị tương ứng:
 - user_id: ID người dùng.
 - transaction_date: Ngày giao dịch.
 - description: Mô tả giao dịch.
 - amount: Số tiền giao dịch.
 - category: Danh mục giao dịch.
 - account_id: Tài khoản liên quan.

5.8. Xem lịch sử giao dịch của người dùng

```
CREATE PROCEDURE GetUserTransactions
    @user_id INT
AS
BEGIN
    SELECT * FROM transactions WHERE user_id = @user_id ORDER BY transaction_date DESC;
END;
GO
```

Commands completed successfully.

Completion time: 2025-03-03T09:37:23.0222481+07:00

Hình 53. Thêm giao dịch procedure xem lịch sử giao dịch người dùng

- CREATE PROCEDURE GetUserTransactions: Tạo Stored Procedure có tên GetUserTransactions.
- Tham số đầu vào:

@user_id INT: ID của người dùng muốn truy vấn giao dịch.

CHƯƠNG 6. XÂY DỰNG CÁC TRIGGER

6.1.Trigger cập nhật số dư tài khoản sau giao dịch

❖ Chức năng:

- Sau khi một giao dịch mới được thêm vào bảng transactions, trigger này sẽ cập nhật số dư tài khoản (balance) trong bảng accounts.
- Số dư sẽ tăng hoặc giảm tùy thuộc vào giao dịch.

```
-- trigger 1: updateBalanceAfterTransaction trigger kiểm tra số dư trước khi giao dịch
CREATE TRIGGER UpdateBalanceAfterTransaction
ON transactions
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE a
    SET a.balance = a.balance + i.amount
    FROM accounts a
    JOIN inserted i ON a.account_id = i.account_id;
    INSERT INTO transaction_logs (transaction_id, user_id, transaction_date, description, amount, category, account_id, created_at)
    SELECT transaction_id, user_id, transaction_date, description, amount, category, account_id, GETDATE()
    FROM inserted;
END;
GO

INSERT INTO transactions (account_id, amount) VALUES (1, 100);
SELECT * FROM accounts WHERE account_id = 1;
```

Messages
Command completed successfully.

Hình 54. Trigger cập nhật số dư tài khoản sau giao dịch

- CREATE TRIGGER UpdateBalanceAfterTransaction ON transactions AFTER INSERT AS BEGIN ... END;; Định nghĩa trigger UpdateBalanceAfterTransaction kích hoạt sau khi một bản ghi được chèn vào bảng transactions.
- SET NOCOUNT ON;; Ngăn chặn việc trả về số dòng bị ảnh hưởng, giúp tăng hiệu suất.
- UPDATE a SET a.balance = a.balance + i.amount FROM accounts a JOIN inserted i ON a.account_id = i.account_id;; Cập nhật cột balance trong bảng accounts. Sử dụng JOIN với bảng inserted (bảng ảo chứa các bản ghi vừa được chèn) để lấy thông tin account_id và amount.
- INSERT INTO transaction_logs ... SELECT ... FROM inserted;; Ghi thông tin giao dịch vào bảng transaction_logs.
- INSERT INTO transactions (account_id, amount) VALUES (1, 100);; Chèn một bản ghi vào bảng transactions để kích hoạt trigger.
- SELECT * FROM accounts WHERE account_id = 1;; Truy vấn để kiểm tra số dư của tài khoản sau khi trigger được kích hoạt.
- SELECT * FROM sys.triggers WHERE name = 'UpdateBalanceAfterTransaction';; Truy vấn để kiểm tra sự tồn tại của trigger trong hệ thống.

6.2.Trigger kiểm tra ngày mua đầu tư

❖ Chức năng:

- Kiểm tra ngày mua (purchase_date) của một khoản đầu tư trong bảng investments.
- Nếu ngày mua lớn hơn ngày hiện tại (tức là trong tương lai), giao dịch sẽ bị hủy và báo lỗi "Purchase date cannot be in the future".

```
SELECT * FROM sys.triggers WHERE name = 'UpdateBalanceAfterTransaction';
-- Trigger 2: CheckInvestmentPurchaseDate Trigger kiểm tra ngày mua đầu tư
CREATE TRIGGER CheckInvestmentPurchaseDate
ON investments
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    IF EXISTS (SELECT 1 FROM inserted WHERE purchase_date > GETDATE())
    BEGIN
        THROW 50002, 'Purchase date cannot be in the future', 1;
        ROLLBACK TRANSACTION;
    END;
END;
GO-- Thử chèn ngày mua trong tương lai (lỗi)
INSERT INTO investments (purchase_date) VALUES ('2024-12-31');
-- Chèn ngày mua hợp lệ
INSERT INTO investments (purchase_date) VALUES ('2023-10-27');
```

Messages
Commands completed successfully.
Completion time: 2025-02-18T09:48:01.8270692+07:00

Hình 55. Trigger kiểm tra ngày mua đầu tư

- CREATE TRIGGER CheckInvestmentPurchaseDate ON investments AFTER INSERT AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được chèn vào bảng investments.
- IF EXISTS (SELECT 1 FROM inserted WHERE purchase_date > GETDATE()) BEGIN ... END;; Kiểm tra xem có bản ghi nào trong bảng inserted có purchase_date lớn hơn ngày hiện tại (GETDATE()) hay không.
- THROW 50002, 'Purchase date cannot be in the future', 1;; Nếu điều kiện trên đúng, ném ra lỗi với mã lỗi 50002 và thông báo "Purchase date cannot be in the future".
- ROLLBACK TRANSACTION;; Hủy bỏ giao dịch nếu có lỗi.
- INSERT INTO investments (purchase_date) VALUES ('2024-12-31');; Thử chèn ngày mua trong tương lai để kiểm tra trigger.
- INSERT INTO investments (purchase_date) VALUES ('2023-10-27');; Chèn ngày mua hợp lệ.
- SELECT * FROM investments;; Kiểm tra dữ liệu trong bảng investments.
- SELECT * FROM sys.triggers WHERE name = 'CheckInvestmentPurchaseDate';; Kiểm tra sự tồn tại của trigger.

6.3.Trigger cập nhật giá trị đầu tư sau khi thay đổi

❖ Chức năng:

- Khi giá trị hiện tại của khoản đầu tư (current_value) thay đổi, trigger này sẽ in ra thông báo "Investment value updated" để thông báo rằng giá trị đầu tư đã được cập nhật.

```
CREATE TRIGGER UpdateInvestmentValue
ON investments
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    IF UPDATE(current_value)
    BEGIN
        INSERT INTO investment_value_logs (investment_id, old_value, new_value, updated_at)
        SELECT i.investment_id, d.current_value, i.current_value, GETDATE()
        FROM inserted i
        JOIN deleted d ON i.investment_id = d.investment_id;
    END;
END;
GO

-- Trigger 4: UpdateBudgetSpentAmountAfterTransaction Trigger cập nhật số tiền đã chi tiêu trong
CREATE TRIGGER UpdateBudgetSpentAmountAfterTransaction
ON transactions
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    IF UPDATE(spent_amount)
    BEGIN
        INSERT INTO budget_spent_logs (transaction_id, old_spent_amount, new_spent_amount, updated_at)
        SELECT t.transaction_id, d.spent_amount, t.spent_amount, GETDATE()
        FROM inserted t
        JOIN deleted d ON t.transaction_id = d.transaction_id;
    END;
END;
GO
```

Messages

Commands completed successfully.

Completion time: 2025-09-18T09:48:57.6578048+07:00

Hình 56. Trigger cập nhật giá trị đầu tư khi thay đổi

- CREATE TRIGGER UpdateInvestmentValue ON investments AFTER UPDATE AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được cập nhật trong bảng investments.
- IF UPDATE(current_value) BEGIN ... END;; Kiểm tra xem cột current_value có được cập nhật hay không.
- INSERT INTO investment_value_logs ... SELECT ... FROM inserted i JOIN deleted d ON i.investment_id = d.investment_id;; Ghi log thông tin giá trị đầu tư cũ và mới vào bảng investment_value_logs. Sử dụng JOIN giữa bảng inserted (bản ghi mới) và deleted (bản ghi cũ) để lấy thông tin.

6.4.Trigger cập nhật số tiền đã chi tiêu trong ngân sách

❖ Chức năng:

- Khi một giao dịch mới được thêm vào bảng transactions, trigger này sẽ cập nhật số tiền đã chi tiêu (spent_amount) trong bảng budget.
- Nó lấy dữ liệu từ bảng liên kết budget_transactions để xác định ngân sách liên quan đến giao dịch.

```
-- Trigger 3: UpdateInvestmentValue Trigger cập nhật giá trị đầu tư sau khi thay đổi
CREATE TRIGGER UpdateInvestmentValue
ON investments
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    IF UPDATE(current_value)
    BEGIN
        INSERT INTO investment_value_logs (investment_id, old_value, new_value, updated_at)
        SELECT i.investment_id, d.current_value, i.current_value, GETDATE()
        FROM inserted i
        JOIN deleted d ON i.investment_id = d.investment_id;
    END;
END;
GO

-- Trigger 4: UpdateBudgetSpentAmountAfterTransaction Trigger cập nhật số tiền đã chi tiêu trong r
```

%

Messages

Commands completed successfully.

Completion time: 2025-03-18T09:48:57.6578048+07:00

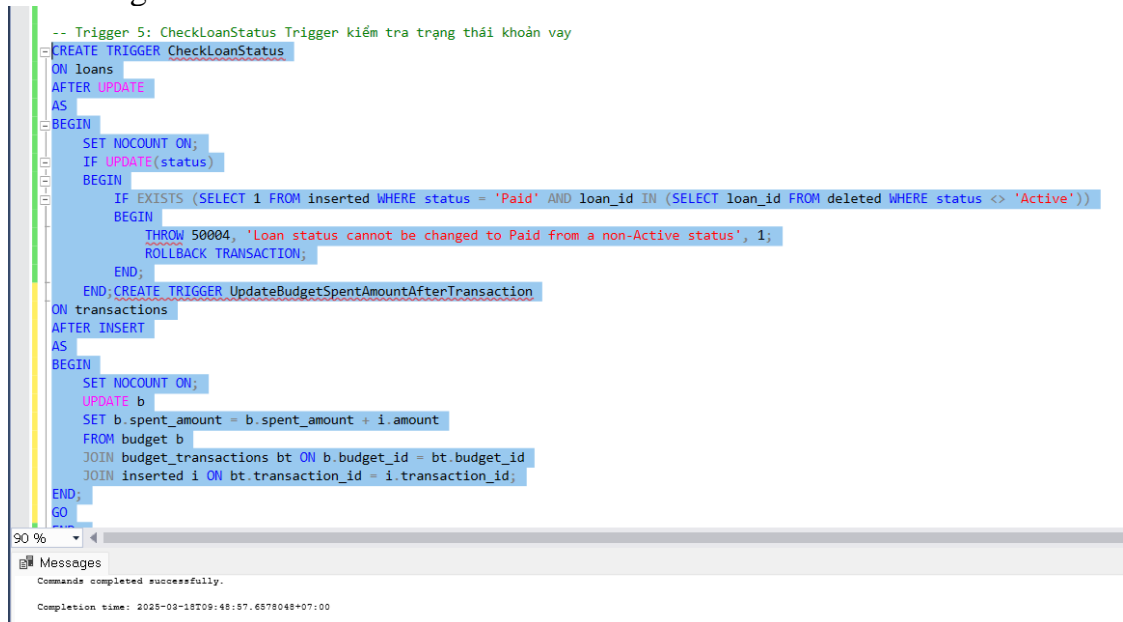
Hình 57. Trigger cập nhật số tiền đã chi tiêu trong ngân sách

- CREATE TRIGGER UpdateBudgetSpentAmountAfterTransaction ON transactions AFTER INSERT AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được chèn vào bảng transactions.
- UPDATE b SET b.spent_amount = b.spent_amount + i.amount FROM budget b JOIN budget_transactions bt ON b.budget_id = bt.budget_id JOIN inserted i ON bt.transaction_id = i.transaction_id;; Cập nhật cột spent_amount trong bảng budget. Sử dụng JOIN với bảng budget_transactions và inserted để lấy thông tin budget_id và amount.

6.5.Trigger kiểm tra trạng thái khoản vay

❖ Chức năng:

- Khi trạng thái (status) của một khoản vay trong bảng loans được cập nhật, trigger này sẽ kiểm tra nếu trạng thái mới là "Paid" nhưng trạng thái cũ không phải "Active".
- Nếu điều kiện trên xảy ra, giao dịch sẽ bị hủy và báo lỗi "Loan status cannot be changed to Paid from a non-Active status".



```
-- Trigger 5: CheckLoanStatus Trigger kiểm tra trạng thái khoản vay
CREATE TRIGGER CheckLoanStatus
ON loans
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    IF UPDATE(status)
    BEGIN
        IF EXISTS (SELECT 1 FROM inserted WHERE status = 'Paid' AND loan_id IN (SELECT loan_id FROM deleted WHERE status <> 'Active'))
        BEGIN
            THROW 50004, 'Loan status cannot be changed to Paid from a non-Active status', 1;
            ROLLBACK TRANSACTION;
        END;
    END;
END;
GO

-- Trigger 6: UpdateBudgetSpentAmountAfterTransaction
CREATE TRIGGER UpdateBudgetSpentAmountAfterTransaction
ON transactions
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE b
    SET b.spent_amount = b.spent_amount + i.amount
    FROM budget b
    JOIN budget_transactions bt ON b.budget_id = bt.budget_id
    JOIN inserted i ON bt.transaction_id = i.transaction_id;
END;
GO
```

90 %

Messages

Commands completed successfully.

Completion time: 2025-02-18T09:48:57.6578048+07:00

Hình 58. Trigger kiểm tra trạng thái khoản vay

- CREATE TRIGGER CheckLoanStatus ON loans AFTER UPDATE AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được cập nhật trong bảng loans.
- IF UPDATE(status) BEGIN ... END;; Kiểm tra xem cột status có được cập nhật hay không.
- IF EXISTS (SELECT 1 FROM inserted WHERE status = 'Paid' AND loan_id IN (SELECT loan_id FROM deleted WHERE status <> 'Active')) BEGIN ... END;; Kiểm tra xem có bản ghi nào được cập nhật có status là 'Paid' và loan_id không có trạng thái 'Active' trước đó hay không.
- THROW 50004, 'Loan status cannot be changed to Paid from a non-Active status', 1;; Nếu điều kiện trên đúng, ném ra lỗi.
- ROLLBACK TRANSACTION;; Hủy bỏ giao dịch nếu có lỗi.

6.6.Trigger cập nhật ngày đáo hạn khoản vay

❖ Chức năng:

- Khi ngày bắt đầu khoản vay (start_date) thay đổi, trigger này sẽ cập nhật ngày đáo hạn (due_date) bằng cách cộng thêm 1 năm kể từ ngày bắt đầu mới.

```
-- Trigger 6: UpdateLoanDueDate   Trigger cập nhật ngày đáo hạn
CREATE TRIGGER UpdateLoanDueDate
ON loans
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    IF UPDATE(start_date)
    BEGIN
        UPDATE loans
        SET due_date = DATEADD(year, 1, i.start_date)
        FROM inserted i
        WHERE loans.loan_id = i.loan_id;
    END;
END;
GO
```

Hình 59. Trigger cập nhật đáo hạn khoản vay

- CREATE TRIGGER UpdateLoanDueDate ON loans AFTER UPDATE AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được cập nhật trong bảng loans.
- IF UPDATE(start_date) BEGIN ... END;; Kiểm tra xem cột start_date có được cập nhật hay không.
- UPDATE loans SET due_date = DATEADD(year, 1, i.start_date) FROM inserted i WHERE loans.loan_id = i.loan_id;; Cập nhật cột due_date trong bảng loans. Sử dụng DATEADD(year, 1, i.start_date) để thêm một năm vào ngày bắt đầu khoản vay. FROM inserted i WHERE loans.loan_id = i.loan_id; để đảm bảo chỉ cập nhật ngày đáo hạn của khoản vay được cập nhật.

6.7.Trigger ghi log giao dịch

❖ Chức năng:

- Sau khi một giao dịch mới được thêm vào bảng transactions, trigger này sẽ ghi lại thông tin giao dịch vào bảng transaction_logs.
- Nó lưu lại các thông tin như: mã giao dịch (transaction_id), người thực hiện (user_id), số tiền (amount), loại giao dịch (category), ngày giao dịch (transaction_date), v.v.

```
-- trigger LogTransaction trigger ghi log giao dịch
CREATE TRIGGER LogTransaction
ON transactions
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO transaction_logs (transaction_id, user_id, transaction_date, description, amount, category, account_id, created_at)
    SELECT transaction_id, user_id, transaction_date, description, amount, category, account_id, GETDATE()
    FROM inserted;
END;
GO
```

Hình 60. trigger khi log giao dịch

- CREATE TRIGGER LogTransaction ON transactions AFTER INSERT AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được chèn vào bảng transactions.
- INSERT INTO transaction_logs ... SELECT ... FROM inserted;; Ghi log thông tin giao dịch từ bảng inserted vào bảng transaction_logs. Sử dụng GETDATE() để lấy thời gian hiện tại.

6.8.Trigger xóa người dùng và tất cả dữ liệu liên quan (INSTEAD OF DELETE)

❖ Chức năng:

- Khi một người dùng bị xóa khỏi bảng users, trigger này sẽ tự động xóa tất cả các dữ liệu liên quan của người đó, bao gồm:
 - Tài khoản (accounts)
 - Giao dịch (transactions)
 - Khoản đầu tư (investments)
 - Ngân sách (budget)
 - Khoản vay (loans)
- Sau khi xóa tất cả các dữ liệu liên quan, người dùng sẽ được xóa khỏi bảng users.


```

-- Trigger 8: DeleteUserCascade Trigger xóa người dùng và các dữ liệu liên quan
CREATE TRIGGER DeleteUserCascade
ON users
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @deleted_user_id INT;
    SELECT @deleted_user_id = user_id FROM deleted;
    DELETE FROM accounts WHERE user_id = @deleted_user_id;
    DELETE FROM transactions WHERE user_id = @deleted_user_id;
    DELETE FROM investments WHERE user_id = @deleted_user_id;
    DELETE FROM budget WHERE user_id = @deleted_user_id;
    DELETE FROM loans WHERE user_id = @deleted_user_id;
    DELETE FROM users WHERE user_id = @deleted_user_id;
    INSERT INTO deleted_users (user_id, deleted_at) VALUES (@deleted_user_id, GETDATE());
END;
GO

```

Hình 61. trigger xóa người dùng và dữ liệu liên quan

- CREATE TRIGGER DeleteUserCascade ON users INSTEAD OF DELETE AS BEGIN ... END;; Định nghĩa trigger kích hoạt thay vì hành động xóa trên bảng users. Sử dụng INSTEAD OF để kiểm soát hoàn toàn quá trình xóa.
- DECLARE @deleted_user_id INT; SELECT @deleted_user_id = user_id FROM deleted;; Khai báo và gán giá trị user_id từ bảng deleted (bảng ảo chứa bản ghi bị xóa) cho biến @deleted_user_id.
- DELETE FROM accounts WHERE user_id = @deleted_user_id;; Xóa tất cả các bản ghi liên quan đến người dùng bị xóa trong bảng accounts.
- DELETE FROM transactions WHERE user_id = @deleted_user_id;; Xóa tất cả các bản ghi liên quan đến người dùng bị xóa trong bảng transactions.
- DELETE FROM investments WHERE user_id = @deleted_user_id;; Xóa tất cả các bản ghi liên quan đến người dùng bị xóa trong bảng investments.
- DELETE FROM budget WHERE user_id = @deleted_user_id;; Xóa tất cả các bản ghi liên quan đến người dùng bị xóa trong bảng budget.
- DELETE FROM loans WHERE user_id = @deleted_user_id;; Xóa tất cả các bản ghi liên quan đến người dùng bị xóa trong bảng loans.
- DELETE FROM users WHERE user_id = @deleted_user_id;; Xóa người dùng khỏi bảng users.
- INSERT INTO deleted_users (user_id, deleted_at) VALUES (@deleted_user_id, GETDATE());; Ghi log thông tin người dùng bị xóa vào bảng deleted_users.

6.9.Trigger kiểm tra số dư trước khi giao dịch

❖ Chức năng:

- Khi một giao dịch mới được thêm vào bảng transactions, trigger này sẽ kiểm tra số dư tài khoản (balance).
- Nếu giao dịch là rút tiền và số dư không đủ, giao dịch sẽ bị hủy và báo lỗi "Insufficient balance".

```
CREATE TRIGGER CheckBalanceBeforeTransaction
ON transactions
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    IF EXISTS (SELECT 1 FROM inserted i JOIN accounts a ON i.account_id = a.account_id WHERE a.balance - i.amount < 0 AND i.amount < 0)
    BEGIN
        THROW 50001, 'Insufficient balance', 1;
        ROLLBACK TRANSACTION;
    END;
END;
GO

-- Trigger 10: CheckBudgetBeforeSpending Trigger kiểm tra ngân sách trước khi chi tiêu
CREATE TRIGGER CheckBudgetBeforeSpending
```

Hình 62. Trigger kiểm tra số dư trước khi giao dịch

- CREATE TRIGGER CheckBalanceBeforeTransaction ON transactions AFTER INSERT AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được chèn vào bảng transactions.
- IF EXISTS (SELECT 1 FROM inserted i JOIN accounts a ON i.account_id = a.account_id WHERE a.balance - i.amount < 0 AND i.amount < 0) BEGIN ... END;; Kiểm tra xem có bản ghi nào trong bảng inserted có số dư tài khoản không đủ để thực hiện giao dịch rút tiền hay không.
- THROW 50001, 'Insufficient balance', 1;; Nếu điều kiện trên đúng, ném ra lỗi.
- ROLLBACK TRANSACTION;; Hủy bỏ giao dịch nếu có lỗi.

6.10.Trigger kiểm tra ngân sách trước khi chi tiêu

❖ Chức năng:

- Khi một giao dịch mới được thêm vào bảng transactions, trigger này sẽ kiểm tra nếu số tiền chi tiêu vượt quá hạn mức ngân sách (allocated_amount).
- Nếu số tiền đã chi tiêu cộng với số tiền mới vượt quá ngân sách, giao dịch sẽ bị hủy và báo lỗi "Budget exceeded".

```

-- Trigger 10: CheckBudgetBeforeSpending Trigger kiểm tra ngân sách trước khi chi tiêu
CREATE TRIGGER CheckBudgetBeforeSpending
ON transactions
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    IF EXISTS (SELECT 1 FROM inserted i JOIN budget_transactions bt ON i.transaction_id = bt.transaction_id JOIN budget b ON bt.budget_id = b.budget_id WHERE b.spent_amount + i.amount > b.allocated_amount) BEGIN
        THROW 50003, 'Budget exceeded', 1;
        ROLLBACK TRANSACTION;
    END;
END;
GO

DROP TRIGGER IF EXISTS UpdateBalanceAfterTransaction;
DROP TRIGGER IF EXISTS CheckInvestmentPurchaseDate;

```

Messages

Commands completed successfully.

Completion time: 2025-03-18T09:48:57.6578048+07:00

Hình 63. trigger kiểm tra ngân sách trước khi tiêu

- CREATE TRIGGER CheckBudgetBeforeSpending ON transactions AFTER INSERT AS BEGIN ... END;; Định nghĩa trigger kích hoạt sau khi một bản ghi được chèn vào bảng transactions.
- IF EXISTS (SELECT 1 FROM inserted i JOIN budget_transactions bt ON i.transaction_id = bt.transaction_id JOIN budget b ON bt.budget_id = b.budget_id WHERE b.spent_amount + i.amount > b.allocated_amount) BEGIN ... END;; Kiểm tra xem có bản ghi nào trong bảng inserted làm cho số tiền đã chi tiêu vượt quá ngân sách được cấp phát hay không.
- THROW 50003, 'Budget exceeded', 1;; Nếu điều kiện trên đúng, ném ra lỗi.
- ROLLBACK TRANSACTION;; Hủy bỏ giao dịch nếu có lỗi.

6.11. Chạy các trigger

```

-- Trigger 1: UpdateBalanceAfterTransaction
INSERT INTO transactions (account_id, amount) VALUES (1, 100);
SELECT * FROM accounts WHERE account_id = 1;
SELECT * FROM transaction_logs;

-- Trigger 2: CheckInvestmentPurchaseDate
INSERT INTO investments (purchase_date) VALUES ('2024-12-31'); -- Lỗi
INSERT INTO investments (purchase_date) VALUES ('2023-10-27');
SELECT * FROM investments;

-- Trigger 3: UpdateInvestmentValue
UPDATE investments SET current_value = 6000 WHERE investment_id = 1;
SELECT * FROM investment_value_logs;

-- Trigger 4: UpdateBudgetSpentAmountAfterTransaction
INSERT INTO transactions (account_id, amount) VALUES (1, 200);
INSERT INTO budget_transactions (budget_id, transaction_id) VALUES (1, 3);
SELECT * FROM budget WHERE budget_id = 1;

-- Trigger 5: CheckLoanStatus
UPDATE loans SET status = 'Paid' WHERE loan_id = 1; -- Lỗi
UPDATE loans SET status = 'Paid' WHERE loan_id = 2;
SELECT * FROM loans;

-- Trigger 6: UpdateLoanDueDate
UPDATE loans SET start_date = '2023-10-27' WHERE loan_id = 1;
SELECT * FROM loans WHERE loan_id = 1;

```

Hình 64. Chạy các trigger

```

-- Trigger 7: LogTransaction
INSERT INTO transactions (account_id, amount) VALUES (2, 500);
SELECT * FROM transaction_logs;

-- Trigger 8: DeleteUserCascade
DELETE FROM users WHERE user_id = 1;
SELECT * FROM users;
SELECT * FROM accounts;
SELECT * FROM transactions;
SELECT * FROM investment_value_logs;
SELECT * FROM budget;
SELECT * FROM loans;
SELECT * FROM deleted_users;

-- Trigger 9: CheckBalanceBeforeTransaction
INSERT INTO transactions (account_id, amount) VALUES (2, -6000); -- Lỗi
INSERT INTO transactions (account_id, amount) VALUES (2, -500);
SELECT * FROM accounts WHERE account_id = 2;

-- Trigger 10: CheckBudgetBeforeSpending
INSERT INTO transactions (account_id, amount) VALUES (1, 500);
INSERT INTO budget_transactions (budget_id, transaction_id) VALUES (1, 4); -- Lỗi
INSERT INTO transactions (account_id, amount) VALUES (1, 100);
INSERT INTO budget_transactions (budget_id, transaction_id) VALUES (1, 5);
SELECT * FROM budget WHERE budget_id = 1;

```

CHƯƠNG 7. PHÂN QUYỀN VÀ BẢO VỆ CƠ SỞ DỮ LIỆU

7.1. Tạo Login cho người dùng

+ Tạo login cho Quản lý

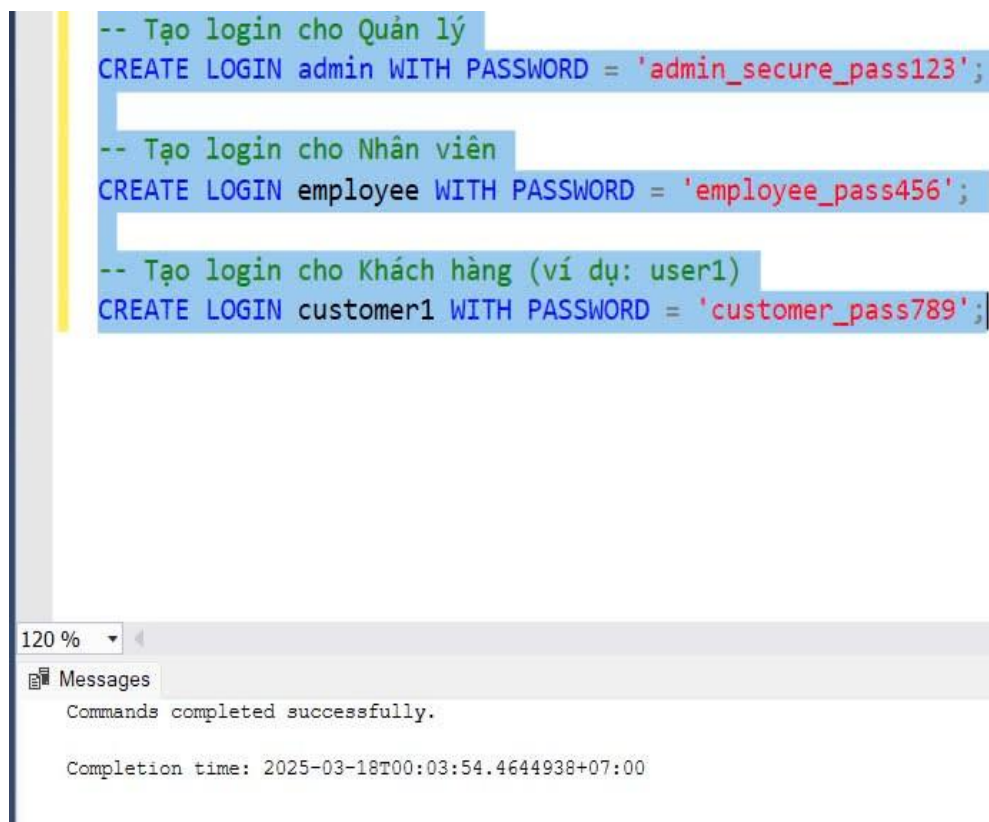
```
CREATE LOGIN admin WITH PASSWORD = 'admin_secure_pass123';
```

+ Tạo login cho Nhân viên

```
CREATE LOGIN employee WITH PASSWORD = 'employee_pass456';
```

+ Tạo login cho Khách hàng (vd: user1)

```
CREATE LOGIN customer1 WITH PASSWORD = 'customer_pass789'
```



```
-- Tạo login cho Quản lý
CREATE LOGIN admin WITH PASSWORD = 'admin_secure_pass123';

-- Tạo login cho Nhân viên
CREATE LOGIN employee WITH PASSWORD = 'employee_pass456';

-- Tạo login cho Khách hàng (ví dụ: user1)
CREATE LOGIN customer1 WITH PASSWORD = 'customer_pass789';
```

120 %

Messages

Commands completed successfully.

Completion time: 2025-03-18T00:03:54.4644938+07:00

Hình 65. Tạo login người dùng

-CREATE LOGIN admin: Tạo một tài khoản đăng nhập vào SQL Server có tên admin.
WITH PASSWORD = 'admin_secure_pass123': Thiết lập mật khẩu cho tài khoản admin.

-Tạo login employee với mật khẩu employee_pass456.

-Tạo login customer1 với mật khẩu customer_pass789

7.2. Tạo Users trong CSDL QuanLyTaiChinh

+ Sử dụng CSDL QuanLyTaiChinh

USE QuanLyTaiChinh;

GO

+ Tạo user cho admin

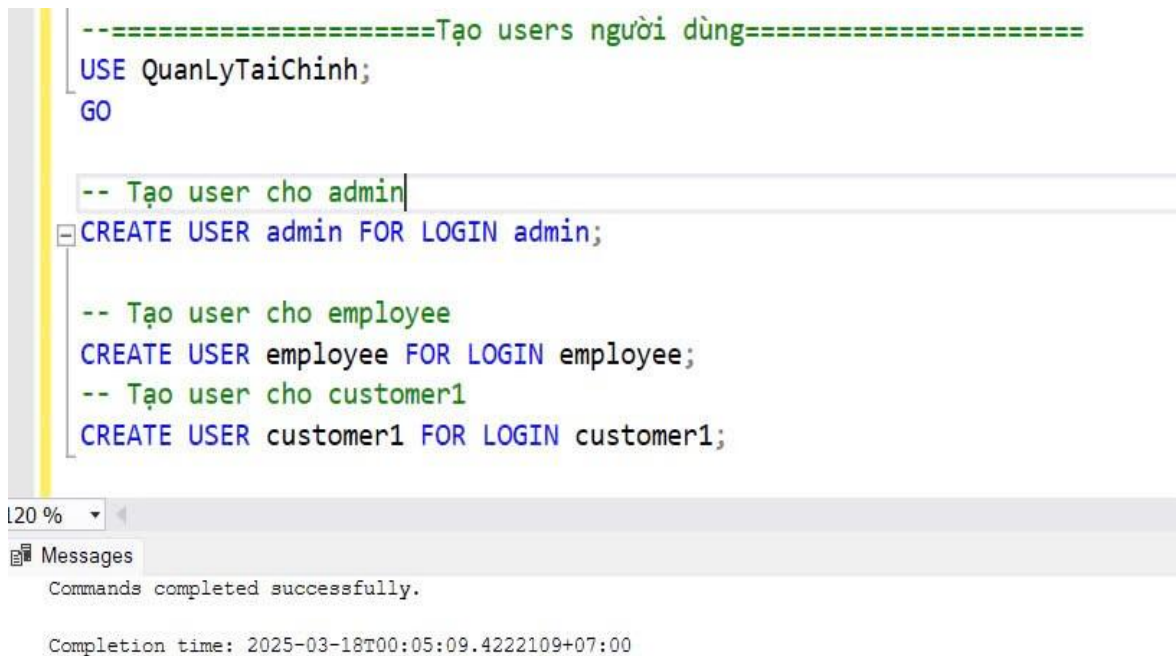
CREATE USER admin FOR LOGIN admin;

+ Tạo user cho employee

CREATE USER employee FOR LOGIN employee;

+ Tạo user cho customer1

CREATE USER customer1 FOR LOGIN customer1;



```
--=====Tạo users người dùng=====
USE QuanLyTaiChinh;
GO

-- Tạo user cho admin
CREATE USER admin FOR LOGIN admin;

-- Tạo user cho employee
CREATE USER employee FOR LOGIN employee;

-- Tạo user cho customer1
CREATE USER customer1 FOR LOGIN customer1;
```

Messages

Commands completed successfully.

Completion time: 2025-03-18T00:05:09.4222109+07:00

Hình 66. Tạo user cho người dùng

USE QuanLyTaiChinh;: Chuyển sang sử dụng cơ sở dữ liệu QuanLyTaiChinh.

Tạo user tên admin liên kết với login cùng tên.

Tạo user employee liên kết với login employee

Tạo user customer1 liên kết với login customer1.

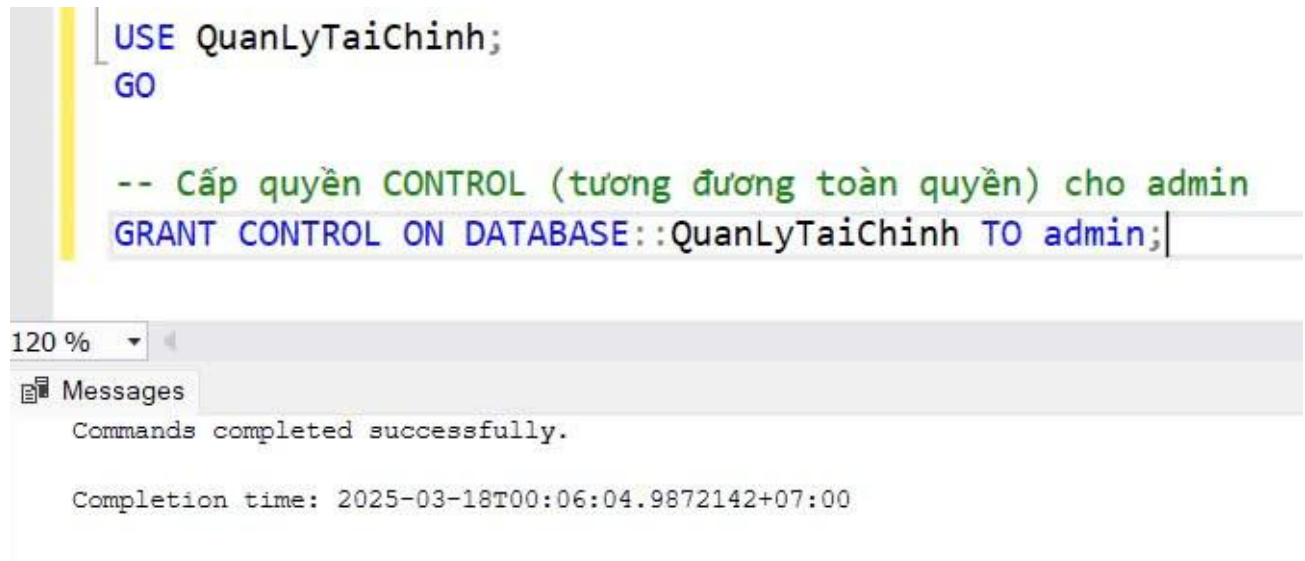
7.3. Phân quyền

7.3.1. Phân quyền cho admin

USE QuanLyTaiChinh;

GO

GRANT CONTROL ON DATABASE::QuanLyTaiChinh TO admin;

The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays the following T-SQL script:

```
USE QuanLyTaiChinh;  
GO  
  
-- Cấp quyền CONTROL (tương đương toàn quyền) cho admin  
GRANT CONTROL ON DATABASE::QuanLyTaiChinh TO admin;
```

 The bottom pane, titled 'Messages', shows the execution results: 'Commands completed successfully.' and 'Completion time: 2025-03-18T00:06:04.9872142+07:00'. The zoom level is set to 120%.

Hình 67. Phân quyền cho admin

GRANT CONTROL: Cấp toàn quyền kiểm soát cơ sở dữ liệu QuanLyTaiChinh cho admin.

7.3.2. Phân quyền cho nhân viên

USE QuanLyTaiChinh;

GO

GRANT SELECT, INSERT, UPDATE ON dbo.users TO employee;

GRANT SELECT, INSERT, UPDATE ON dbo.accounts TO employee;

GRANT SELECT, INSERT, UPDATE ON dbo.transactions TO employee;

GRANT SELECT, INSERT, UPDATE ON dbo.investments TO employee;

GRANT SELECT, INSERT, UPDATE ON dbo.budget TO employee;

GRANT SELECT, INSERT, UPDATE ON dbo.loans TO employee;


```

=====Phân Quyền cho nhân viên=====
USE QuanLyTaichinh;
GO

-- Quyền SELECT, INSERT, UPDATE trên các bảng
GRANT SELECT, INSERT, UPDATE ON dbo.users TO employee;
GRANT SELECT, INSERT, UPDATE ON dbo.accounts TO employee;
GRANT SELECT, INSERT, UPDATE ON dbo.transactions TO employee;
GRANT SELECT, INSERT, UPDATE ON dbo.investments TO employee;
GRANT SELECT, INSERT, UPDATE ON dbo.budget TO employee;
GRANT SELECT, INSERT, UPDATE ON dbo.loans TO employee;

```

Hình 68. Phân quyền cho nhân viên được xem,sửa,cập nhật

GRANT SELECT, INSERT, UPDATE: Cho phép nhân viên có thể đọc (SELECT), thêm (INSERT) và sửa (UPDATE) trên các bảng chỉ định

+ Quyền SELECT trên VIEWs

GRANT SELECT ON dbo.UserAccounts TO employee;

GRANT SELECT ON dbo.UserTransactions TO employee;

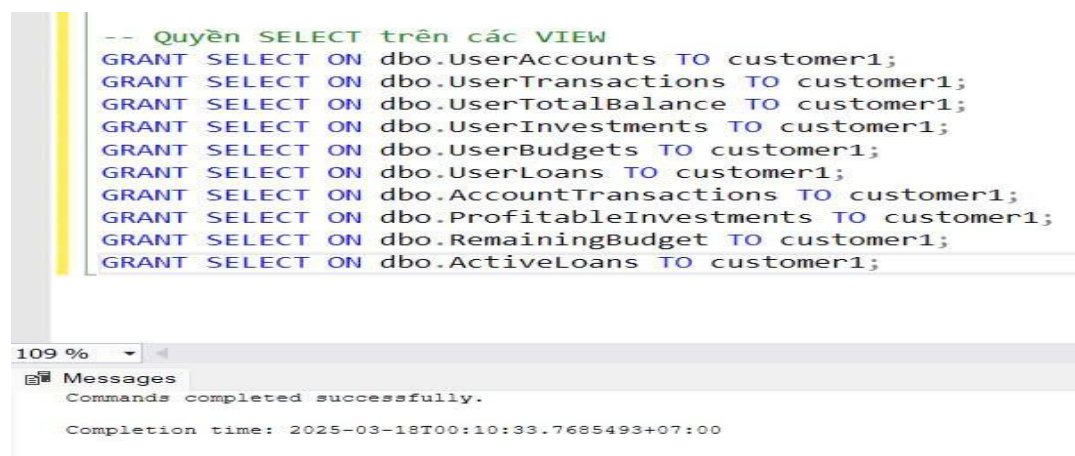
GRANT SELECT ON dbo.UserTotalBalance TO employee;

+ Quyền EXECUTE trên Stored Procedures

GRANT EXECUTE ON dbo.AddUser TO employee;

GRANT EXECUTE ON dbo.AddAccount TO employee;

GRANT EXECUTE ON dbo.AddTransaction TO employee;



Hình 69. Cấp quyền select trên các view

Nhân viên chỉ có quyền xem dữ liệu từ các VIEW trong CSDL.

GRANT EXECUTE: Cho phép nhân viên có quyền thực thi các stored procedure (AddUser, AddAccount, AddTransaction).

7.3.3. Phân quyền cho khách hàng

USE QuanLyTaiChinh;

GO

GRANT SELECT ON dbo.users TO customer1;

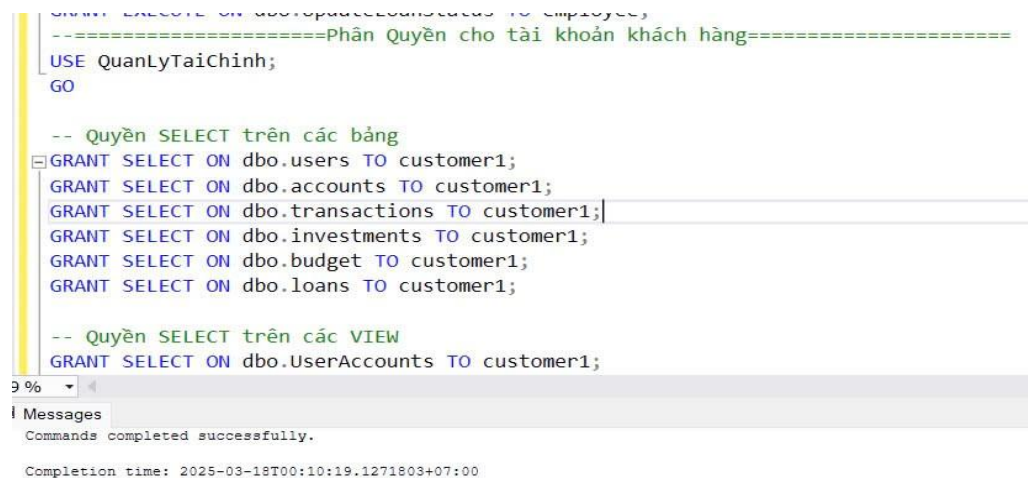
GRANT SELECT ON dbo.accounts TO customer1;

GRANT SELECT ON dbo.transactions TO customer1;

GRANT SELECT ON dbo.investments TO customer1;

GRANT SELECT ON dbo.budget TO customer1;

GRANT SELECT ON dbo.loans TO customer1;

The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a script with the following content:

```
GRANT EXECUTE ON dbo.sp_addaccount TO employee;  
-----Phân Quyền cho tài khoản khách hàng-----  
USE QuanLyTaiChinh;  
GO  
  
-- Quyền SELECT trên các bảng  
GRANT SELECT ON dbo.users TO customer1;  
GRANT SELECT ON dbo.accounts TO customer1;  
GRANT SELECT ON dbo.transactions TO customer1;  
GRANT SELECT ON dbo.investments TO customer1;  
GRANT SELECT ON dbo.budget TO customer1;  
GRANT SELECT ON dbo.loans TO customer1;  
  
-- Quyền SELECT trên các VIEW  
GRANT SELECT ON dbo.UserAccounts TO customer1;
```

 The bottom pane shows a message: "Commands completed successfully." and the completion time: "2025-03-18T00:10:19.1271803+07:00".

Hình 70. Phân quyền cho khách hàng

7.3.4. Giới hạn dữ liệu bằng Row-Level Security (RLS)

CREATE FUNCTION dbo.fn_securitypredicate(@user_id AS INT)

RETURNS TABLE

WITH SCHEMABINDING

AS

RETURN SELECT 1 AS fn_securitypredicate_result

WHERE @user_id = CAST(SESSION_CONTEXT(N'UserID') AS INT)

OR USER_NAME() = 'admin' OR USER_NAME() = 'employee';

GO

```

--Áp dụng Row-Level Security (RLS) để giới hạn dữ liệu của khách hàng:

-- Tạo hàm bảo mật để lọc dữ liệu theo user_id
CREATE FUNCTION dbo.fn_securitypredicate(@user_id AS INT)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN SELECT 1 AS fn_securitypredicate_result
WHERE @user_id = CAST(SESSION_CONTEXT(N'UserID') AS INT)
OR USER_NAME() = 'admin' OR USER_NAME() = 'employee';
GO

```

Hình 71. Áp dụng RLS giới hạn dữ liệu của khách

Tạo hàm bảo mật: Giới hạn dữ liệu hiển thị chỉ với user_id tương ứng.

Điều kiện:

- Nếu user_id của bản ghi trùng với SESSION_CONTEXT('UserID'), bản ghi sẽ được hiển thị.
- Nếu người dùng là admin hoặc employee, họ sẽ thấy toàn bộ dữ liệu.

+ Tạo chính sách bảo mật

```

-- Tạo chính sách bảo mật (security policy)
CREATE SECURITY POLICY UserFilter
ADD FILTER PREDICATE dbo.fn_securitypredicate(user_id) ON dbo.users,
ADD FILTER PREDICATE dbo.fn_securitypredicate(user_id) ON dbo.accounts,
ADD FILTER PREDICATE dbo.fn_securitypredicate(user_id) ON dbo.transactions,
ADD FILTER PREDICATE dbo.fn_securitypredicate(user_id) ON dbo.investments,
ADD FILTER PREDICATE dbo.fn_securitypredicate(user_id) ON dbo.budget,
ADD FILTER PREDICATE dbo.fn_securitypredicate(user_id) ON dbo.loans
WITH (STATE = ON);
GO

```

Đặt user_id vào SESSION_CONTEXT khi khách hàng đăng nhập (thực hiện trong ứng dụng)

Messages
Commands completed successfully.
Completion time: 2025-03-18T00:12:00.3986040+07:00

Hình 72. Tạo chính sách bảo mật

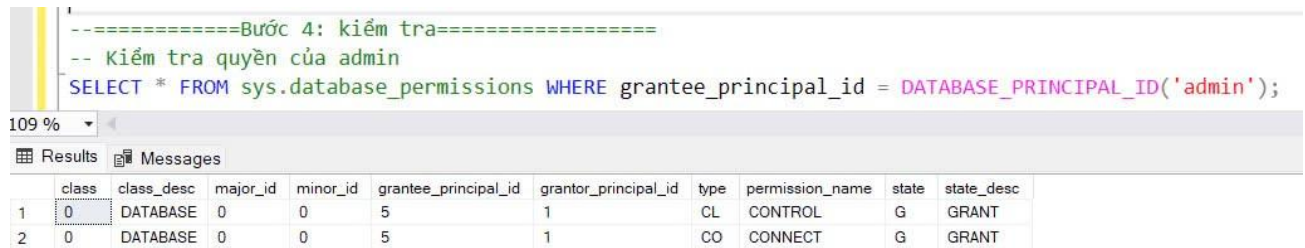
CREATE SECURITY POLICY: Áp dụng chính sách bảo mật lên nhiều bảng, giới hạn dữ liệu theo từng user_id.

Khi khách hàng đăng nhập, hệ thống sẽ lưu user_id vào **SESSION_CONTEXT**, giúp RLS hoạt động đúng.

7.4. Kiểm tra quyền

+ Kiểm tra quyền của admin

```
SELECT * FROM sys.database_permissions WHERE grantee_principal_id =  
DATABASE_PRINCIPAL_ID('admin');
```



The screenshot shows a SQL Server Enterprise Manager interface. At the top, a query window displays the following text:
-----BƯỚC 4: KIỂM TRA-----
-- Kiểm tra quyền của admin
SELECT * FROM sys.database_permissions WHERE grantee_principal_id = DATABASE_PRINCIPAL_ID('admin');
Below the query window, the 'Results' tab is active, showing a table with 10 columns: class, class_desc, major_id, minor_id, grantee_principal_id, grantor_principal_id, type, permission_name, state, and state_desc. The table contains two rows of data.

	class	class_desc	major_id	minor_id	grantee_principal_id	grantor_principal_id	type	permission_name	state	state_desc
1	0	DATABASE	0	0	5	1	CL	CONTROL	G	GRANT
2	0	DATABASE	0	0	5	1	CO	CONNECT	G	GRANT

Hình 73. Kiểm tra quyền

+ Kiểm tra quyền của employee

```
SELECT * FROM sys.database_permissions WHERE grantee_principal_id =  
DATABASE_PRINCIPAL_ID('employee');
```

+ Kiểm tra quyền của customer1

```
SELECT * FROM sys.database_permissions WHERE grantee_principal_id =  
DATABASE_PRINCIPAL_ID('customer1');
```

DANH MỤC TÀI LIỆU THAM KHẢO

- [1]. Đỗ Thị Minh Phụng, Hệ quản trị cơ sở dữ liệu SQL SERVER, Đại học Quốc Gia TP. Hồ Chí Minh.
- [2]. Nguyễn Thanh Quang, Hoàng Thanh Quang, *Tự học SQL SERVER 2000*, NXB Văn Hóa và Thông tin
- [3]. Phạm Hữu Khang, *SQL SERVER lập trình thủ tục và hàm 2005*, NXB Lao Động Xã Hội.