## SP\_ChuyenTien (Me)

|  |  |
| --- | --- |
| SP\_ChuyenTien (Sử dụng giao tác tập trung => vì nó chỉ thực hiện các thao tác cập nhật dữ liệu trên một cơ sở dữ liệu duy nhất mà không liên quan đến các cơ sở dữ liệu khác. Việc sử dụng giao tác tập trung giúp đảm bảo tính toàn vẹn và nhất quán của dữ liệu khi có nhiều người dùng cùng thực hiện các thao tác cập nhật dữ liệu trên cùng một cơ sở dữ liệu. | Giải Thích |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_ChuyenTien]  @sotkchuyen nvarchar(10),  @sotknhan nvarchar(10),  @sotien money,  @manv nvarchar(10)  AS  DECLARE @sodu money = (SELECT SODU FROM TaiKhoan WHERE SOTK = @sotkchuyen)  DECLARE @check int = 1  IF NOT EXISTS(SELECT \* FROM dbo.TaiKhoan AS TK WHERE TK.SOTK = @sotkchuyen)  BEGIN  SET @check = 2  SELECT @check  RETURN  END  IF NOT EXISTS(SELECT \* FROM dbo.TaiKhoan AS TK WHERE TK.SOTK = @sotknhan)  BEGIN  SET @check = 3  SELECT @check  RETURN  END  IF (@sotien <= @sodu)  BEGIN  SET XACT\_ABORT ON /\*lỗi tự động rollback \*/  BEGIN TRANSACTION  BEGIN TRY  INSERT INTO GD\_CHUYENTIEN(SOTK\_CHUYEN, NGAYGD, SOTIEN, SOTK\_NHAN, MANV)  VALUES(@sotkchuyen, GETDATE(), @sotien, @sotknhan, @manv)  UPDATE TaiKhoan  SET SODU = SODU - @sotien  WHERE SOTK = @sotkchuyen  UPDATE TaiKhoan  SET SODU = SODU + @sotien  WHERE SOTK = @sotknhan    COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  SET @check = 0  END CATCH  END  ELSE  SET @check = 4  SELECT @check | Chuỗi mã T-SQL trên là một stored procedure được đặt tên là **SP\_ChuyenTien**. Stored procedure này nhận vào 4 tham số đầu vào là **@sotkchuyen**, **@sotknhan**, **@sotien** và **@manv**.   * **@sotkchuyen**: số tài khoản của người gửi tiền (kiểu dữ liệu **nvarchar(10)**) * **@sotknhan**: số tài khoản của người nhận tiền (kiểu dữ liệu **nvarchar(10)**) * **@sotien**: số tiền cần chuyển (kiểu dữ liệu **money**) * **@manv**: mã nhân viên thực hiện giao dịch (kiểu dữ liệu **nvarchar(10)**)   Tiếp theo, stored procedure sử dụng câu lệnh **DECLARE** để khai báo biến **@sodu** có kiểu dữ liệu **money**. Biến này sẽ được sử dụng để lưu trữ số dư trong tài khoản của người gửi tiền.  Sau đó, stored procedure sử dụng biến **@check** có kiểu dữ liệu **int** để lưu trữ kết quả của giao dịch chuyển tiền. Biến này sẽ có các giá trị sau:   * **1**: giao dịch chuyển tiền thành công * **2**: tài khoản người gửi không tồn tại * **3**: tài khoản người nhận không tồn tại * **4**: số dư trong tài khoản người gửi không đủ để thực hiện giao dịch chuyển tiền * **0**: lỗi xảy ra trong quá trình thực hiện giao dịch chuyển tiền   Tiếp theo, stored procedure sử dụng câu lệnh **IF NOT EXISTS** để kiểm tra xem số tài khoản của người gửi và người nhận có tồn tại trong cơ sở dữ liệu hay không. Nếu một trong hai tài khoản không tồn tại, stored procedure sẽ gán giá trị **2** hoặc **3** cho biến **@check** tương ứng và trả về giá trị này thông qua câu lệnh **SELECT @check** ở cuối stored procedure.  Sau đó, stored procedure sử dụng câu lệnh **IF** để kiểm tra xem số dư trong tài khoản của người gửi có đủ để thực hiện giao dịch chuyển tiền hay không. Nếu số dư trong tài khoản người gửi không đủ, stored procedure sẽ gán giá trị **4** cho biến **@check** và trả về giá trị này thông qua câu lệnh **SELECT @check** ở cuối stored procedure.  Nếu số dư trong tài khoản của người gửi đủ để thực hiện giao dịch chuyển tiền, thì stored procedure sẽ tiếp tục thực hiện các bước sau:   1. Nó sẽ bật chế độ XACT\_ABORT để tự động rollback giao dịch nếu có lỗi xảy ra. 2. Sau đó, nó sẽ bắt đầu một giao dịch mới bằng lệnh BEGIN TRANSACTION. 3. Tiếp theo, nó sẽ thêm một bản ghi vào bảng GD\_CHUYENTIEN để lưu lại thông tin giao dịch chuyển tiền. 4. Nó sẽ cập nhật số dư của tài khoản người gửi bằng cách trừ đi số tiền chuyển. 5. Nó sẽ cập nhật số dư của tài khoản người nhận bằng cách cộng thêm số tiền chuyển. 6. Nếu các bước trên hoàn tất mà không có lỗi xảy ra, thì stored procedure sẽ commit giao dịch bằng lệnh COMMIT TRAN.   Nếu có lỗi xảy ra trong quá trình thực hiện các bước trên, stored procedure sẽ rollback giao dịch và trả về mã lỗi tương ứng để báo cho người dùng biết lý do tại sao chuyển tiền không thành công. |

## SP\_GoiRut (Me)

|  |  |
| --- | --- |
| SP\_GoiRut sử dụng giao tác tập trung => mọi thao tác đều được thực hiện trong một đơn vị giao tác duy nhất (câu lệnh INSERT và UPDATE được thực hiện trong một giao tác duy nhất bằng cách sử dụng lệnh BEGIN TRANSACTION và COMMIT TRAN) để đảm bảo tính nhất quán và toàn vẹn. | Giải thích |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_GDGoiRut]  @sotk nvarchar(10),  @loaigd nvarchar(2),  @sotien money,  @manv nvarchar(10)  AS  DECLARE @sodu money = (SELECT SODU FROM TaiKhoan WHERE SOTK = @sotk)  DECLARE @check int = 1  IF NOT EXISTS(SELECT \* FROM dbo.TaiKhoan AS TK WHERE TK.SOTK = @sotk)  BEGIN  SET @check = -1  SELECT @check  RETURN  END  IF (@loaigd = 'GT')  BEGIN  SET XACT\_ABORT ON  BEGIN TRANSACTION  BEGIN TRY  INSERT INTO GD\_GOIRUT(SOTK, LOAIGD, NGAYGD, SOTIEN, MANV)  VALUES(@sotk, @loaigd, GETDATE(), @sotien, @manv)  UPDATE TaiKhoan  SET SODU = SODU + @sotien  WHERE SOTK = @sotk  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  SET @check = 0  END CATCH  END  ELSE  BEGIN  IF (@sotien > @sodu)  SET @check = 2  ELSE  BEGIN  SET XACT\_ABORT ON  BEGIN TRANSACTION  BEGIN TRY  INSERT INTO GD\_GOIRUT(SOTK, LOAIGD, NGAYGD, SOTIEN, MANV)  VALUES(@sotk, @loaigd, GETDATE(), @sotien, @manv)  UPDATE TaiKhoan  SET SODU = SODU - @sotien  WHERE SOTK = @sotk  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  SET @check = 0  END CATCH  END  END  SELECT @check | stored procedure **SP\_GDGoiRut** được sử dụng để thực hiện các giao dịch gởi hoặc rút tiền từ tài khoản ngân hàng.  Đầu tiên, stored procedure sẽ kiểm tra xem tài khoản người dùng có tồn tại trong cơ sở dữ liệu hay không. Nếu không, giá trị của biến **@check** sẽ được đặt là -1 và trả về để báo lỗi.  Nếu loại giao dịch là "GT" (gửi tiền), tiền sẽ được thêm vào tài khoản và ghi nhận giao dịch trong bảng GD\_GOIRUT. Nếu giao dịch thành công, stored procedure sẽ commit transaction. Nếu giao dịch thất bại, nó sẽ rollback transaction và giá trị của biến @check sẽ được đặt là 0 để báo lỗi.  Nếu loại giao dịch là "RT" (rút tiền), stored procedure sẽ kiểm tra xem số dư trong tài khoản có đủ để thực hiện giao dịch rút tiền hay không. Nếu số dư trong tài khoản không đủ, giá trị của biến **@check** sẽ được đặt là 2 và trả về để báo lỗi. Nếu số dư trong tài khoản đủ để thực hiện giao dịch, tiền sẽ được rút khỏi tài khoản và ghi nhận giao dịch trong bảng GD\_GOIRUT. Nếu giao dịch thành công, stored procedure sẽ commit transaction. Nếu giao dịch thất bại, nó sẽ rollback transaction và giá trị của biến **@check** sẽ được đặt là 0 để báo lỗi.  Cuối cùng, stored procedure sẽ trả về giá trị của biến **@check**, để cho biết kết quả của giao dịch. Giá trị -1 nếu tài khoản không tồn tại, giá trị 0 nếu giao dịch thất bại, giá trị 1 nếu giao dịch thành công và giá trị 2 nếu số dư không đủ để thực hiện giao dịch rút tiền.  Trường hợp thứ hai, nếu loại giao dịch là "RT" (rút tiền), thì trước khi thực hiện giao dịch, stored procedure sẽ kiểm tra xem số dư trong tài khoản của khách hàng đó có đủ để rút số tiền đó hay không. Nếu số dư đủ, sẽ thực hiện giao dịch rút tiền và cập nhật số dư trong tài khoản của khách hàng, ngược lại nếu số dư không đủ, sẽ trả về giá trị 2, đồng thời không thực hiện giao dịch và không cập nhật số dư.  Nếu trong quá trình thực hiện giao dịch có bất kỳ lỗi nào, stored procedure sẽ rollback toàn bộ giao dịch và trả về giá trị 0.  Khi đó, nếu thực hiện lệnh **EXEC [dbo].[SP\_GDGoiRut] '1234567890', 'GT', 100000, 'NV001'**, stored procedure sẽ thực hiện các bước sau:   1. Kiểm tra xem tài khoản với số tài khoản **1234567890** có tồn tại trong bảng **TaiKhoan** không. Nếu không tồn tại, stored procedure sẽ trả về giá trị **-1** và dừng thực thi. Nếu tồn tại, stored procedure sẽ tiếp tục. 2. Kiểm tra loại giao dịch. Vì giá trị của biến **@loaigd** là **'GT'** (gửi tiền), nên stored procedure sẽ thực hiện các bước trong khối **IF (@loaigd = 'GT')**. 3. Stored procedure bật chế độ **XACT\_ABORT ON** và bắt đầu một transaction. 4. Stored procedure thực hiện thêm một bản ghi mới vào bảng **GD\_GOIRUT** với các giá trị tương ứng. 5. Stored procedure thực hiện cập nhật số dư của tài khoản **1234567890** trong bảng **TaiKhoan** bằng cách thêm giá trị **100000** vào số dư hiện tại của tài khoản đó. 6. Stored procedure commit transaction. 7. Stored procedure trả về giá trị **1**, biểu thị giao dịch đã được thực hiện thành công.   Vì vậy, nếu thực hiện lệnh **EXEC [dbo].[SP\_GDGoiRut] '1234567890', 'GT', 100000, 'NV001'**, stored procedure sẽ trả về giá trị **1**. |

## SP\_CheckCMND (Me)

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_CheckCMND]  @cmnd NVARCHAR(10)  AS  BEGIN  DECLARE @check int  IF EXISTS(SELECT \* FROM dbo.KhachHang WHERE dbo.KhachHang.CMND = @cmnd)  SET @check = 1 --Mã tồn tại ở phân mãnh hiện tại  ELSE IF EXISTS(SELECT \* FROM LINK0.NGANHANG.dbo.KhachHang AS KH WHERE KH.CMND = @cmnd)  SET @check = 2 --Mã tồn tại ở phân mãnh khác  ELSE  SET @check = 0  SELECT @check --Không bị trùng được thêm  END | 1. Dòng đầu tiên sử dụng câu lệnh USE để chuyển đổi đến cơ sở dữ liệu NGANHANG. 2. Tiếp theo, các câu lệnh SET được sử dụng để đặt các cờ ANSI\_NULLS và QUOTED\_IDENTIFIER trong quá trình thực thi stored procedure. 3. Dòng ALTER PROC sử dụng để định nghĩa một stored procedure với tên là SP\_CheckCMND. Stored procedure này sẽ nhận đầu vào là một chuỗi @cmnd có kiểu dữ liệu NVARCHAR và có độ dài tối đa là 10 ký tự. 4. Khối BEGIN...END chứa tất cả các câu lệnh trong stored procedure. 5. Dòng DECLARE @check int khai báo biến @check có kiểu dữ liệu INT để lưu trữ kết quả trả về từ stored procedure. 6. IF EXISTS... được sử dụng để kiểm tra sự tồn tại của CMND trong cùng cơ sở dữ liệu hiện tại. Nếu có, stored procedure sẽ đặt @check thành 1 để đại diện cho mã tồn tại trong phân mãnh hiện tại. 7. ELSE IF EXISTS... sử dụng để kiểm tra sự tồn tại của CMND trong một cơ sở dữ liệu khác. Nếu có, stored procedure sẽ đặt @check thành 2 để đại diện cho mã tồn tại trong một phân mãnh khác. 8. Nếu không có bất kỳ khớp nào, stored procedure sẽ đặt @check thành 0 để đại diện cho mã không trùng lặp và có thể được thêm mới. 9. Dòng cuối cùng sử dụng câu lệnh SELECT để trả về giá trị của biến @check như kết quả của stored procedure. |

## SP\_CheckMaNV

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_CheckMANV]  @maNV NCHAR(10)  AS  BEGIN  DECLARE @check int  IF EXISTS(SELECT \* FROM dbo.NhanVien WHERE dbo.NhanVien.MANV = @maNV)  SET @check = 1 --Mã tồn tại ở phân mãnh hiện tại  ELSE IF EXISTS(SELECT \* FROM LINK0.NGANHANG.dbo.NhanVien AS NV WHERE NV.MANV = @maNV)  SET @check = 2 --Mã tồn tại ở phân mãnh khác  ELSE  SET @check = 0  SELECT @check --Không bị trùng được thêm  END | Đây là một stored procedure để kiểm tra tính hợp lệ của mã nhân viên (MANV). Nó có hai tham số đầu vào: @maNV là mã nhân viên cần kiểm tra. Kết quả trả về được lưu trong biến @check và được chọn để xác định mã nhân viên có tồn tại trong cơ sở dữ liệu hay không.  Các câu lệnh trong stored procedure như sau:   1. USE [NGANHANG]: Đây là lệnh để chọn cơ sở dữ liệu mà stored procedure sẽ được thực thi. 2. SET ANSI\_NULLS ON: Đây là lệnh để thiết lập chế độ ANSI\_NULLS trên ON. Nó chỉ định rằng bất kỳ phép so sánh NULL nào trong stored procedure này đều trả về giá trị NULL. 3. SET QUOTED\_IDENTIFIER ON: Đây là lệnh để bật chế độ QUOTED\_IDENTIFIER trên ON. Nó chỉ định rằng tên đối tượng trong stored procedure này được bao quanh bởi dấu ngoặc kép ("), và tên đối tượng được bao quanh bởi dấu ngoặc kép ("), được xử lý như một chuỗi bình thường. 4. ALTER PROC [dbo].[SP\_CheckMANV]: Đây là câu lệnh để sửa đổi stored procedure có tên SP\_CheckMANV. Đây là một stored procedure tên đã được đặt trước đó, được tạo ra để kiểm tra tính hợp lệ của mã nhân viên. 5. BEGIN: Đây là lệnh để bắt đầu khối thực thi của stored procedure. 6. DECLARE @check int: Đây là lệnh để khai báo một biến có tên là @check có kiểu dữ liệu là int. 7. IF EXISTS(SELECT \* FROM dbo.NhanVien WHERE dbo.NhanVien.MANV = @maNV): Đây là lệnh để kiểm tra xem mã nhân viên được cung cấp có tồn tại trong bảng NhanVien của cơ sở dữ liệu hiện tại không. Nếu có, stored procedure sẽ thiết lập biến @check thành giá trị 1. 8. ELSE IF EXISTS(SELECT \* FROM LINK0.NGANHANG.dbo.NhanVien AS NV WHERE NV.MANV = @maNV): Đây là lệnh để kiểm tra xem mã nhân viên được cung cấp có tồn tại trong bảng NhanVien của cơ sở dữ liệu khác không. Nếu có, stored procedure sẽ thiết lập biến @check thành giá trị 2. 9. ELSE: Đây là lệnh thực hiện khi không tìm thấy bất kỳ mã nhân viên nào trong cơ sở dữ liệu. Trong trường hợp này, stored procedure sẽ thiết lập biến @check thành giá trị 0. 10. SELECT @check: Trả về giá trị của biến @check |

## SP\_ChuyenCN

|  |  |
| --- | --- |
| SP\_ChuyenCN sử dụng giao tác phân tán, chắc chắn rằng msdtc ở trạng thái running  Kiểm tra trạng thái msdtc: sc query msdtc bật msdtc: net start msdtc |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_ChuyenCN]  @manv nvarchar(10),  @macngo nvarchar(10)  AS  DECLARE @check int  DECLARE @manvmax nvarchar(10) = (RIGHT((SELECT TOP 1 MANV FROM LINK1.NGANHANG.DBO.NhanVien ORDER BY MANV DESC), 2))  IF (@manvmax < 10)  SET @manvmax = CONCAT('NV', '0', @manvmax + 1)  ELSE  SET @manvmax = CONCAT('NV', @manvmax + 1)  SET XACT\_ABORT ON  BEGIN DISTRIBUTED TRANSACTION  BEGIN TRY  IF EXISTS (SELECT \* FROM LINK1.NGANHANG.DBO.NhanVien  WHERE SODT = (SELECT SODT FROM NhanVien WHERE MANV = @manv))  BEGIN  UPDATE NhanVien  SET TrangThaiXoa = 1  WHERE MANV = @manv  UPDATE LINK1.NGANHANG.DBO.NhanVien  SET TrangThaiXoa = 0  WHERE SODT = (SELECT SODT FROM NhanVien WHERE MANV = @manv)  END  ELSE  BEGIN  INSERT INTO LINK1.NGANHANG.DBO.NhanVien(MANV, HO, TEN,CMND, DIACHI, PHAI, SODT, MACN, TrangThaiXoa)  SELECT @manvmax AS MANV, HO, TEN,CMND, DIACHI, PHAI, SODT, @macngo AS MACN, 0 FROM NhanVien WHERE MANV = @manv  UPDATE NhanVien  SET TrangThaiXoa = 1  WHERE MANV = @manv  END  IF (@@TRANCOUNT > 0)  BEGIN  COMMIT TRANSACTION  SET @check = 1  END  END TRY  BEGIN CATCH  IF (@@TRANCOUNT > 0)  BEGIN  ROLLBACK TRANSACTION  SET @check = 0  END  END CATCH  SELECT @check |  |

## SP\_DangNhap

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_DangNhap]  @tenLogin nvarchar(50)  AS  BEGIN  DECLARE @tenUser nvarchar(50)  SELECT @tenUser = NAME FROM sys.sysusers WHERE sid = SUSER\_SID(@tenLogin)    SELECT USERNAME = @tenUser,  HOTEN = (SELECT HO + ' ' + TEN FROM NHANVIEN WHERE MANV = @tenUser),  TENNHOM = NAME  FROM sys.sysusers  WHERE UID = (SELECT GROUPUID  FROM SYS.SYSMEMBERS  WHERE MEMBERUID = (SELECT UID FROM sys.sysusers  WHERE NAME = @tenUser))  END | Stored procedure này có một tham số đầu vào là **@tenLogin** là tên đăng nhập vào hệ thống.  Đầu tiên, stored procedure sẽ lấy tên người dùng từ tên đăng nhập truyền vào bằng câu lệnh **SELECT @tenUser = NAME FROM sys.sysusers WHERE sid = SUSER\_SID(@tenLogin)**. Sau đó, stored procedure sẽ trả về các thông tin của người dùng đó bao gồm tên người dùng, họ tên và tên nhóm.  Để lấy họ tên của người dùng, stored procedure sử dụng câu lệnh **SELECT HO + ' ' + TEN FROM NHANVIEN WHERE MANV = @tenUser** trong đó **NHANVIEN** là bảng chứa thông tin về nhân viên trong hệ thống ngân hàng.  Để lấy tên nhóm của người dùng, stored procedure sử dụng câu lệnh **SELECT NAME FROM sys.sysusers WHERE UID = (SELECT GROUPUID FROM SYS.SYSMEMBERS WHERE MEMBERUID = (SELECT UID FROM sys.sysusers WHERE NAME = @tenUser))**. Ở đây, **sys.sysusers** và **SYS.SYSMEMBERS** là các bảng hệ thống của SQL Server, được sử dụng để lấy thông tin về các nhóm người dùng trong hệ thống  Khi ANSI\_NULLS được thiết lập thành ON, các giá trị NULL trong các phép so sánh sẽ được coi là không bằng nhau và không thuộc về bất kỳ một loại dữ liệu nào. Nếu không có tùy chọn này, các phép so sánh với NULL sẽ trả về giá trị không xác định.  Trong Stored Procedure, khi chúng ta sử dụng tùy chọn SET ANSI\_NULLS ON, SQL Server sẽ bắt buộc các phép so sánh với NULL phải được đối xử như một giá trị không xác định và phải sử dụng phép toán IS NULL hoặc IS NOT NULL để kiểm tra giá trị NULL. Điều này sẽ giúp tránh các sai sót trong xử lý dữ liệu và đảm bảo tính đúng đắn của Stored Procedure. |

## SP\_GetCNChuyen

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_GetCNChuyen]  @macn nvarchar(10)  AS  SELECT MACN, TENCN FROM LINK0.NGANHANG.DBO.ChiNhanh WHERE MACN <> @macn | * **USE [NGANHANG]** là câu lệnh để chọn cơ sở dữ liệu **NGANHANG** làm cơ sở dữ liệu hiện tại. Khi thực thi một stored procedure, cơ sở dữ liệu được thực thi trên sẽ là cơ sở dữ liệu hiện tại được chọn bằng câu lệnh này. * **SET ANSI\_NULLS ON** và **SET QUOTED\_IDENTIFIER ON** là hai lệnh cài đặt môi trường để đảm bảo tính nhất quán trong các phép so sánh NULL và đối với việc sử dụng các chuỗi được bao quanh bởi dấu ngoặc kép. * **ALTER PROC [dbo].[SP\_GetCNChuyen]** là câu lệnh để sửa đổi (ALTER) một stored procedure có tên **SP\_GetCNChuyen** trong schema **dbo**. Nó khai báo rằng stored procedure này sẽ có một tham số đầu vào là **@macn** kiểu **nvarchar(10)**. * **AS** là từ khóa đánh dấu bắt đầu phần thân của stored procedure. * Phần thân của stored procedure này chỉ có một câu lệnh truy vấn SQL, là lấy tất cả các chi nhánh khác với **@macn** từ bảng **ChiNhanh** của cơ sở dữ liệu, và trả về hai cột là **MACN** và **TENCN**. |

## SP\_LietKeKhachHang (Me)

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_LietKeKhachHang]  AS  SELECT CMND, HOTEN = HO + ' ' + TEN, DIACHI, PHAI, NGAYCAP, SODT  FROM KhachHang  ORDER BY TEN, HO | * USE [NGANHANG]: Câu lệnh này chỉ định database mà stored procedure sẽ thực hiện trên. * SET ANSI\_NULLS ON: Thiết lập ANSI\_NULLS trên, đảm bảo rằng các phép so sánh NULL sẽ được xử lý theo tiêu chuẩn ANSI. Khi ANSI\_NULLS ON, các giá trị NULL không được coi là bằng nhau, ví dụ: NULL = NULL sẽ trả về FALSE. * SET QUOTED\_IDENTIFIER ON: Thiết lập QUOTED\_IDENTIFIER trên, đảm bảo rằng chuỗi được đặt trong dấu ngoặc kép (" ") được coi là một chuỗi và không phải là một tên đối tượng. Điều này rất quan trọng để tránh lỗi cú pháp và bảo vệ hệ thống khỏi các cuộc tấn công SQL injection.   Sau đó, stored procedure thực hiện câu lệnh SELECT để liệt kê thông tin của các khách hàng, bao gồm:   * CMND: Số chứng minh nhân dân của khách hàng. * HOTEN: Họ tên của khách hàng, được kết hợp từ cột HO và cột TEN trong bảng KhachHang bằng dấu cách. * DIACHI: Địa chỉ của khách hàng. * PHAI: Giới tính của khách hàng. * NGAYCAP: Ngày cấp chứng minh nhân dân của khách hàng. * SODT: Số điện thoại của khách hàng.   Kết quả được sắp xếp theo bảng chữ cái theo tên, rồi đến họ nếu có tên trùng lặp. |

## SP\_SaoKeGD

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_SaoKeGD]  @tungay nvarchar(10),  @denngay nvarchar(10),  @sotk nvarchar(10)  AS  SET DATEFORMAT DMY  DECLARE @sodudau money = (SELECT SODU FROM TaiKhoan WHERE SOTK = @sotk)  DECLARE @ngaygd datetime, @loaigd nvarchar(2), @sotien money, @macn nvarchar(10)  DECLARE cursorGD CURSOR FOR  SELECT NGAYGD, LOAIGD, SOTIEN  FROM GD\_GOIRUT INNER JOIN NhanVien ON GD\_GOIRUT.MANV = NhanVien.MANV  WHERE SOTK = @sotk AND NGAYGD >= @tungay  UNION ALL  SELECT NGAYGD, LOAIGD = 'CT', SOTIEN  FROM GD\_CHUYENTIEN INNER JOIN NhanVien ON GD\_CHUYENTIEN.MANV = NhanVien.MANV  WHERE SOTK\_CHUYEN = @sotk AND NGAYGD >= @tungay  UNION ALL  SELECT NGAYGD, LOAIGD, SOTIEN  FROM LINK0.NGANHANG.DBO.GD\_GOIRUT INNER JOIN LINK0.NGANHANG.DBO.NhanVien ON GD\_GOIRUT.MANV = NhanVien.MANV  WHERE SOTK = @sotk AND NGAYGD >= @tungay AND MACN <> (SELECT TOP 1 MACN FROM NhanVien)  UNION ALL  SELECT NGAYGD, LOAIGD = 'CT', SOTIEN  FROM LINK0.NGANHANG.DBO.GD\_CHUYENTIEN INNER JOIN LINK0.NGANHANG.DBO.NhanVien ON GD\_CHUYENTIEN.MANV = NhanVien.MANV  WHERE SOTK\_CHUYEN = @sotk AND NGAYGD >= @tungay AND MACN <> (SELECT TOP 1 MACN FROM NhanVien)  --ORDER BY NGAYGD DESC  OPEN cursorGD  FETCH NEXT FROM cursorGD  INTO @ngaygd, @loaigd, @sotien  WHILE @@FETCH\_STATUS = 0  BEGIN  IF @loaigd = 'GT'  SET @sodudau -= @sotien  ELSE  SET @sodudau += @sotien  FETCH NEXT FROM cursorGD  INTO @ngaygd, @loaigd, @sotien  END  CLOSE cursorGD  DEALLOCATE cursorGD  CREATE TABLE #TAM (  SODUDAU money,  NGAYGD datetime,  LOAIGD nvarchar(2),  SOTIEN money,  SODUSAU money,  MACN nvarchar(10)  )  DECLARE cursorSaoKe CURSOR FOR  SELECT NGAYGD, LOAIGD, SOTIEN, MACN  FROM GD\_GOIRUT INNER JOIN NhanVien ON GD\_GOIRUT.MANV = NhanVien.MANV  WHERE SOTK = @sotk AND (NGAYGD >= @tungay AND NGAYGD < @denngay)  UNION ALL  SELECT NGAYGD, LOAIGD = 'CT', SOTIEN, MACN  FROM GD\_CHUYENTIEN INNER JOIN NhanVien ON GD\_CHUYENTIEN.MANV = NhanVien.MANV  WHERE SOTK\_CHUYEN = @sotk AND (NGAYGD >= @tungay AND NGAYGD < @denngay)  UNION ALL  SELECT NGAYGD, LOAIGD, SOTIEN, MACN  FROM LINK0.NGANHANG.DBO.GD\_GOIRUT INNER JOIN LINK0.NGANHANG.DBO.NhanVien ON GD\_GOIRUT.MANV = NhanVien.MANV  WHERE SOTK = @sotk AND (NGAYGD >= @tungay AND NGAYGD < @denngay) AND MACN <> (SELECT TOP 1 MACN FROM NhanVien)  UNION ALL  SELECT NGAYGD, LOAIGD = 'CT', SOTIEN, MACN  FROM LINK0.NGANHANG.DBO.GD\_CHUYENTIEN INNER JOIN LINK0.NGANHANG.DBO.NhanVien ON GD\_CHUYENTIEN.MANV = NhanVien.MANV  WHERE SOTK\_CHUYEN = @sotk AND (NGAYGD >= @tungay AND NGAYGD < @denngay) AND MACN <> (SELECT TOP 1 MACN FROM NhanVien)  ORDER BY NGAYGD  OPEN cursorSaoKe  FETCH NEXT FROM cursorSaoKe  INTO @ngaygd, @loaigd, @sotien, @macn  WHILE @@FETCH\_STATUS = 0  BEGIN  IF @loaigd = 'GT'  BEGIN  INSERT INTO #TAM  SELECT @sodudau, @ngaygd, @loaigd, @sotien, @sodudau + @sotien, @macn  SET @sodudau += @sotien  END  ELSE  BEGIN  INSERT INTO #TAM  SELECT @sodudau, @ngaygd, @loaigd, @sotien, @sodudau - @sotien, @macn  SET @sodudau -= @sotien  END  FETCH NEXT FROM cursorSaoKe  INTO @ngaygd, @loaigd, @sotien, @macn  END  CLOSE cursorSaoKe  DEALLOCATE cursorSaoKe  SELECT \* FROM #TAM |  |

## SP\_ TaoTaiKhoan

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_TaoTaiKhoan]  @LGNAME nvarchar(10),  @PASS nvarchar(20),  @USERNAME nvarchar(10),  @ROLE nvarchar(20)  AS  BEGIN  DECLARE @check int = 0  DECLARE @RET INT  EXEC @RET = SP\_ADDLOGIN @LGNAME, @PASS, 'NGANHANG'  IF (@RET = 1) -- LOGIN NAME BI TRUNG  BEGIN  SET @check = 1  SELECT @check  RETURN 1  END  EXEC @RET = SP\_GRANTDBACCESS @LGNAME, @USERNAME  IF (@RET = 1) -- USER NAME BI TRUNG  BEGIN  EXEC SP\_DROPLOGIN @LGNAME  SET @check = 2  SELECT @check  RETURN 2  END  EXEC sp\_addrolemember @ROLE, @USERNAME  EXEC sp\_addsrvrolemember @LGNAME, 'SecurityAdmin'  SELECT @check  END | * Đầu tiên, nó sử dụng database **NGANHANG** để thực hiện stored procedure. * Sau đó, **SET ANSI\_NULLS ON** thiết lập để bật các quy tắc xử lý NULL trong stored procedure. Nó yêu cầu các quy tắc ANSI\_NULLS trong stored procedure và bật các quy tắc này để đảm bảo tính nhất quán và độ chính xác của dữ liệu. * **SET QUOTED\_IDENTIFIER ON** yêu cầu SQL Server xử lý các chuỗi được bao quanh bằng dấu ngoặc kép (" ") là các định danh hoặc chuỗi thay vì là các giá trị. Nó cũng đảm bảo rằng các câu lệnh được trình bày trong stored procedure tuân theo cú pháp chuẩn ANSI SQL. * **ALTER PROC** để chỉ định rằng đây là một stored procedure được sửa đổi. **SP\_TaoTaiKhoan** là tên stored procedure. * Sau đó, các tham số được khai báo để lấy giá trị đầu vào cho stored procedure. Các tham số đó là **@LGNAME**, **@PASS**, **@USERNAME**, và **@ROLE**. * Các biến **@check** và **@RET** được khai báo và gán giá trị ban đầu. * Stored procedure sử dụng **EXEC** để gọi các stored procedure khác như **SP\_ADDLOGIN**, **SP\_GRANTDBACCESS**, **SP\_DROPLOGIN**, **sp\_addrolemember**, và **sp\_addsrvrolemember**. Những stored procedure này được sử dụng để thêm login, grant database access, xóa login, thêm user vào role, và thêm login vào server role. * Nếu tên login hoặc tên user đã tồn tại trong database thì stored procedure sẽ trả về giá trị khác nhau cho biến **@check**. Nếu tên login hoặc tên user chưa tồn tại thì stored procedure sẽ tiếp tục thực hiện các câu lệnh tiếp theo để tạo tài khoản. Cuối cùng, giá trị của **@check** sẽ được trả về để thông báo về kết quả của stored procedure. |

## SP\_ MoTaiKhoan (Me)

|  |  |
| --- | --- |
| SP | GT |
| USE [NGANHANG]  GO SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_MoTaiKhoan] @cmnd NVARCHAR(10),  @macn NVARCHAR(10)  AS  BEGIN  DECLARE @NewSoTK nchar(9) = CAST(RAND()\*1000000000 AS INT)  WHILE (EXISTS(SELECT SOTK from [dbo].TaiKhoan WHERE [dbo].TaiKhoan.SOTK = @NewSoTK) AND EXISTS(select SOTK from LINK0.NGANHANG.dbo.TaiKhoan as NewSoTK WHERE NewSoTK.SOTK = @NewSoTK))  BEGIN  SET @NewSoTK = CAST(RAND()\*1000000000 AS INT)  END  INSERT INTO TaiKhoan(SOTK, CMND, SODU,MACN) VALUES (@NewSoTK,@cmnd, 0,@macn);  END | 1. [SP\_MoTaiKhoan] và nhận hai tham số đầu vào là @cmnd và @macn có kiểu dữ liệu là NVARCHAR(10). Một biến cục bộ @NewSoTK được khai báo và khởi tạo giá trị là một số ngẫu nhiên có 9 chữ số. Câu lệnh CAST(RAND()\*1000000000 AS INT) được sử dụng để sinh số ngẫu nhiên trong khoảng từ 0 đến 999,999,999. 2. Một vòng lặp WHILE được sử dụng để kiểm tra xem số tài khoản @NewSoTK đã tồn tại trong bảng [dbo].TaiKhoan và bảng LINK0.NGANHANG.dbo.TaiKhoan hay chưa. Nếu số tài khoản này đã tồn tại, vòng lặp sẽ tiếp tục sinh một số ngẫu nhiên khác cho @NewSoTK và kiểm tra lại cho đến khi tìm được một số tài khoản chưa tồn tại trong cả hai bảng. 3. Khi tìm được số tài khoản mới và không trùng với bất kỳ số tài khoản nào trong bảng [dbo].TaiKhoan và bảng LINK0.NGANHANG.dbo.TaiKhoan, một câu lệnh INSERT INTO được sử dụng để chèn một bản ghi mới vào bảng [dbo].TaiKhoan. Bản ghi mới này sẽ có giá trị số tài khoản là @NewSoTK, số CMND là @cmnd, số dư là 0, và mã chi nhánh là @macn.   Lệnh SET ANSI\_NULLS ON và SET QUOTED\_IDENTIFIER ON được sử dụng để đảm bảo rằng cấu hình ANSI\_NULLS và QUOTED\_IDENTIFIER đang được kích hoạt trong quá trình thực thi stored procedure. Điều này đảm bảo tính nhất quán và chính xác trong cách xử lý các giá trị NULL và xác định tên đối tượng trong câu lệnh SQL. |

## SP\_ LietKeTaiKhoan (Me)

|  |  |
| --- | --- |
|  |  |
| USE [NGANHANG]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_LietKeTaiKhoan] Script Date: 5/22/2023 12:56:16 PM \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROC [dbo].[SP\_LietKeTaiKhoan]  @tungay nvarchar(10),  @denngay nvarchar(10),  @all int  AS  SET DATEFORMAT DMY  IF (@all = 0)  SELECT SOTK, NGAYMOTK, TaiKhoan.MACN  FROM TaiKhoan INNER JOIN ChiNhanh ON TaiKhoan.MACN = ChiNhanh.MACN  WHERE (NGAYMOTK >= @tungay AND NGAYMOTK < @denngay)  ELSE  SELECT SOTK, NGAYMOTK, TaiKhoan.MACN  FROM TaiKhoan  WHERE (NGAYMOTK >= @tungay AND NGAYMOTK < @denngay) | 1. **USE [NGANHANG]**: Câu lệnh này xác định rằng các câu lệnh SQL sau đây sẽ được thực thi trong ngữ cảnh của cơ sở dữ liệu **[NGANHANG]**. Nó đặt cơ sở dữ liệu hiện tại mà các câu lệnh sẽ tác động. 2. **SET ANSI\_NULLS ON**: Câu lệnh này đặt tùy chọn ANSI\_NULLS thành ON. Khi tùy chọn này được bật, so sánh với giá trị NULL sẽ được coi là giá trị không xác định. Điều này có nghĩa là nếu bạn so sánh một giá trị với NULL, kết quả sẽ không xác định. 3. **SET QUOTED\_IDENTIFIER ON**: Câu lệnh này đặt tùy chọn QUOTED\_IDENTIFIER thành ON. Khi tùy chọn này được bật, dấu ngoặc kép (") được sử dụng để bao quanh các định danh chuỗi, chẳng hạn như tên bảng hoặc tên cột. Điều này giúp phân biệt giữa các định danh và giá trị chuỗi. 4. **ALTER PROC [dbo].[SP\_LietKeTaiKhoan]**: Câu lệnh này sửa đổi một Stored Procedure đã tồn tại có tên **[SP\_LietKeTaiKhoan]** trong schema **[dbo]**. Nó cho phép bạn thay đổi nội dung của Stored Procedure đã tồn tại mà không cần tạo lại từ đầu. 5. **@tungay nvarchar(10), @denngay nvarchar(10), @all int**: Đây là khai báo các tham số đầu vào cho Stored Procedure. Stored Procedure này có ba tham số là **@tungay**, **@denngay** và **@all**. Các tham số này sẽ được truyền giá trị khi gọi Stored Procedure. 6. **SET DATEFORMAT DMY**: Câu lệnh này đặt định dạng ngày tháng trong Stored Procedure thành ngày/tháng/năm (DD/MM/YYYY). Điều này đảm bảo rằng các ngày trong câu lệnh SELECT được hiểu đúng theo định dạng này. 7. **IF (@all = 0)**: Đây là câu lệnh điều kiện IF. Nó kiểm tra giá trị của biến **@all**. Nếu **@all** có giá trị bằng 0, câu lệnh SELECT đầu tiên sẽ được thực thi. Nếu không, câu lệnh SELECT thứ hai sẽ được thực thi. 8. Trong câu lệnh SELECT đầu tiên, dữ liệu được truy vấn từ hai bảng là **TaiKhoan** và **ChiNhanh** bằng cách sử dụng câu lệnh JOIN. Kết quả trả về sẽ bao gồm các cột **SOTK**, **NGAYMOTK** và **TaiKhoan.MACN** từ hai bảng này. Điều kiện WHERE sẽ lọc các bản ghi dựa trên ngày mở tài khoản. 9. Trong câu lệnh SELECT thứ hai, dữ liệu chỉ được truy vấn từ bảng **TaiKhoan**. Tương tự như câu lệnh SELECT đầu tiên, nó cũng lọc các bản ghi dựa trên ngày mở tài khoản. Tuy nhiên, trong trường hợp này, không có sự kết hợp với bảng **ChiNhanh** 10. **GO**: Đây là một từ khóa không phải là một câu lệnh SQL mà chỉ được sử dụng để phân tách các lô lệnh. Nó cho phép chạy các lệnh SQL theo từng lô một. Trong trường hợp này, nó được sử dụng để phân tách các phần của script. 11. Một Stored Procedure là một khối mã SQL được lưu trữ trong cơ sở dữ liệu và có thể được gọi để thực thi. Stored Procedure này có tên là **[SP\_LietKeTaiKhoan]** trong schema **[dbo]**. Nó nhận ba tham số đầu vào là **@tungay**, **@denngay** và **@all**. 12. Câu lệnh SELECT trong khối IF/ELSE truy xuất dữ liệu từ bảng **TaiKhoan** và kết quả trả về gồm các cột **SOTK**, **NGAYMOTK** và **TaiKhoan.MACN**. Điều kiện WHERE lọc các bản ghi dựa trên ngày mở tài khoản (**NGAYMOTK**) trong khoảng từ **@tungay** đến **@denngay**. 13. Trường hợp ELSE của câu lệnh IF chứa một câu lệnh SELECT tương tự, nhưng không có phần kết hợp với bảng **ChiNhanh**. Điều này có nghĩa là nó chỉ truy xuất dữ liệu từ bảng **TaiKhoan** và áp dụng điều kiện WHERE giống như trước đó.   Đoạn mã SQL này tạo ra một Stored Procedure có thể được gọi để truy xuất thông tin tài khoản từ bảng **TaiKhoan** trong cơ sở dữ liệu **[NGANHANG]**. Các tham số **@tungay** và **@denngay** được sử dụng để xác định khoảng thời gian ngày mở tài khoản. Tham số **@all** quyết định xem liệu tất cả các tài khoản hoặc chỉ một số tài khoản cụ thể sẽ được liệt kê. |