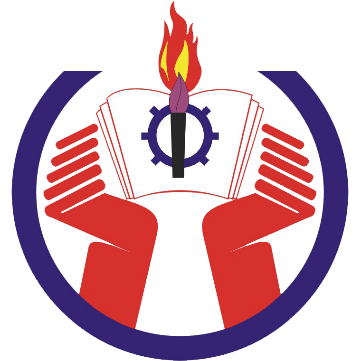


**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH QUALITY TRAINING**

**🕯✡🕮🕮✡🕯**

****

**COURSE PROJECT**

**DATABASE MANAGEMENT SYSTEM**

**CONVENIENCE STORE MANAGEMENT**

**Lecturer: Mr. Nguyen Thanh Son**

**Class: DBMS330284E\_21\_2\_01CLC**

**Member: Group 6**

**Ho Chi Minh, May 17th 2022**

**MEMBER OF GROUP 6**

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **ID** |
| 1 | Nguyễn Huỳnh Thanh Toàn | 20110420 |
| 2 | Bùi Ngọc Ánh | 20110354 |
| 3 | Lê Y Thiện | 20110403 |

**EVALUATION AND SCORE**

**EVALUATION:**

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

**SCORE**:………………………

**TABLE OF CONTENTS**

[PART 1. SYSTEM DESCRIPTION 7](#_Toc104243937)

[1.1. Application overview 7](#_Toc104243938)

[1.2. Functions of application 7](#_Toc104243939)

[1.3. System analysis and design 8](#_Toc104243940)

[1.4. Expected interface 9](#_Toc104243941)

[PART 2. DATABASE ANALYSIS AND DESIGN 11](#_Toc104243942)

[2.1. Model ERD 11](#_Toc104243943)

[2.2. Model relationships between tables 11](#_Toc104243944)

[PART 3. SYSTEM CONFIGURATION AND INSTALLATION 13](#_Toc104243945)

[3.1. Database Creation and Constraints 13](#_Toc104243946)

[3.2. Creating statements in Stored Procedure 15](#_Toc104243947)

[3.2.1. Add account 15](#_Toc104243948)

[3.2.2. Add employee 15](#_Toc104243949)

[3.2.3. Update employee 15](#_Toc104243950)

[3.2.4. Delete employee 16](#_Toc104243951)

[3.2.5. List employee 16](#_Toc104243952)

[3.2.6. Find employee 16](#_Toc104243953)

[3.2.7. Add customer 16](#_Toc104243954)

[3.2.8. Add product 16](#_Toc104243955)

[3.2.9. Update product 17](#_Toc104243956)

[3.2.10. Delete product 17](#_Toc104243957)

[3.2.11. Add invoice 17](#_Toc104243958)

[3.2.12. Update invoice 17](#_Toc104243959)

[3.2.13. Delete invoice 18](#_Toc104243960)

[3.2.14. Add detail 18](#_Toc104243961)

[3.2.15. Update detail 18](#_Toc104243962)

[3.2.16. Delete detail 18](#_Toc104243963)

[3.2.17. Add manufacturer 18](#_Toc104243964)

[3.2.18. Update manufacturer 18](#_Toc104243965)

[3.2.19. Delete manufacturer 18](#_Toc104243966)

[3.2.20. Add stock 19](#_Toc104243967)

[3.2.21. Update stock 19](#_Toc104243968)

[3.2.22. Delete stock 19](#_Toc104243969)

[3.2.23. Add types 19](#_Toc104243970)

[3.2.24. Update type 19](#_Toc104243971)

[3.2.25. Delete type 19](#_Toc104243972)

[3.3. Database connection 20](#_Toc104243973)

[3.4. Decentralization 20](#_Toc104243974)

[3.4.1. Database table for authorization 21](#_Toc104243975)

[3.4.2. Decentralized code on the database 21](#_Toc104243976)

[3.4.3. Decentralized code on C# 22](#_Toc104243977)

[3.5. Triggers and transactions 22](#_Toc104243978)

[3.5.1. Settings to create login accounts according to user permissions 22](#_Toc104243979)

[3.5.2. Install login account updates according to user permissions 23](#_Toc104243980)

[3.5.3. Setting to delete login accounts according to user permissions 24](#_Toc104243981)

[3.5.4. Setting to update total pay for invoice 24](#_Toc104243982)

[3.5.5. Setting to decrease total pay for invoice 24](#_Toc104243983)

[3.5.6. Setting to check the invoice exists or not 24](#_Toc104243984)

[3.5.7. Setting to update the invoice 25](#_Toc104243985)

[3.6. Function 25](#_Toc104243986)

[3.6.1. Total number of employees 25](#_Toc104243987)

[3.6.2. Total number of customers 25](#_Toc104243988)

[3.6.3. Total number of products 26](#_Toc104243989)

[3.6.4. Check account 26](#_Toc104243990)

[3.6.5. Find product by name and type 26](#_Toc104243991)

[3.6.6. Find customer 26](#_Toc104243992)

[3.6.7. Show detail of invoice 26](#_Toc104243993)

[3.6.8. Show detail 27](#_Toc104243994)

[3.6.9. Auto show information of product 27](#_Toc104243995)

[3.6.10. Total pay of invoice 27](#_Toc104243996)

[3.6.11. Search employees by id 27](#_Toc104243997)

[3.6.12. Employee gender statistics 27](#_Toc104243998)

[3.7. View 27](#_Toc104243999)

[3.7.1. View for products 27](#_Toc104244000)

[3.7.2. View for customers 28](#_Toc104244001)

[3.7.3. View for type 28](#_Toc104244002)

[3.7.4. View for stock 28](#_Toc104244003)

SYSTEM DESCRIPTION

Application overview

One of the urgent requirements today is to put information technology into management in order to reduce human labor, save time, increase accuracy, narrow storage space, and avoid loss of storage space, exit and ensure the safety of data.

Putting information technology into the "Convenience Store Management Application" for management is essential, because we spend very little time but get high efficiency, accuracy and convenience quickly. The software can meet the needs of convenience stores, helping to make sales and sales statistics quickly and efficiently.

The application helps users have the most intuitive view of data information. With the manager will be able to manage personnel and products easily without spending too much effort and time. With employees, they will see accurate information about products and functions related to their personal business. Users only need to correctly manipulate the pre-formatted items on the interface, the program ensures the most accurate data is retrieved.

With the "Convenience Store Management Application", we will conduct all operations right on a computer or laptop and can check out much more quickly and conveniently.

Functions of application

Build convenience store management software to ensure the following functions:

* Add, edit, delete, update employees if there is a change. Search for employees by Employee ID. Calculate total employees.
* Add, edit, update customers if there is a change. Search for customers by phone number. Calculate total customers.
* Add, edit, delete, update products if there is a change. Search products by ID and type. Total products.
* Add supply.
* Create and update new invoices. Search invoices by Invoice ID and by price greater than 50000.
* Add, edit, delete, update detailed invoices
* Decentralize management and employee login.
* Statistics of revenue, profit by day and number of male and female employees.

System analysis and design

The subject area applies in the convenience store. The functionality focuses on the management of the necessary information of the convenience store. The application is suitable for stores with many employees, managers, etc.

A convenience store needs a database to manage, including the following information:

* **Employee:** Each employee has personal information such as employee name (E-Name), employee's address (E-Address), employee's position (E-Position), employee's phone number (E- Phone), employee's photo (E-Image), employee’s birthday (E-Birth), employee’s gender (E-Gender), salary (E-Salary) and an employee code (E-ID) used to distinguish it from other employees. An employee is verified by logging in and an employee can confirm multiple invoices.
* **Account:** An employee is identified by logging into his or her account. An account will have an account name (Username), password (Password), employee code (E-ID), email of the account (A-Email) and state of the account (Active). An employee will only have one account.
* **Invoice:** An invoice will be issued by an employee. An invoice consists of the total value of the invoice (I-TotalPay), the invoice printing date (I-Date) and an invoice code (I-ID) used to distinguish between bills. An invoice will be paid by a customer. An invoice will show information including the price (D-Price) and quantity (D-Amount) of one or more products.
* **Customer:** A customer has personal information such as customer name (C-Name), customer phone number (C-Phone), total payment amount (C-TotalPay) and a customer code (C-ID) used to distinguish from other customers. The name of the customer may be null because there are some customers who have not registered their information before paying the invoices. A customer can pay one or more bills at the same time.
* **Product:** A product is recorded by one or more invoices. A product has information such as product name (P-Name), unit name (Unit-Name), price of the product (P-Price), image of the product (P-Image) and a product code (P-ID) used to distinguish one product from another. A product will belong to a certain product category. A product can be supplied or supplied by a warehouse or a manufacturer
* **Type:** A type can include many different products. A type consists of two pieces of information is a type name (T-Name) and a type code (T-ID) used to distinguish it from another type.
* **Stock:** A stock can hold many products. Information of a stock including the batch ID (Batch ID), import date (ImportDate) and quantity of the item (AmountOfProduct)
* **Manufacture:** A manufacturer can produce many products. A manufacturer consists of two pieces of information is the manufacturer's name (M-Name) and a manufacturer's code (M-ID) used to distinguish it from other manufacturers.

Expected interface

The login page is used to assign access rights to the Convenience Store Management software including sales staff and store managers.

Employee account interface:

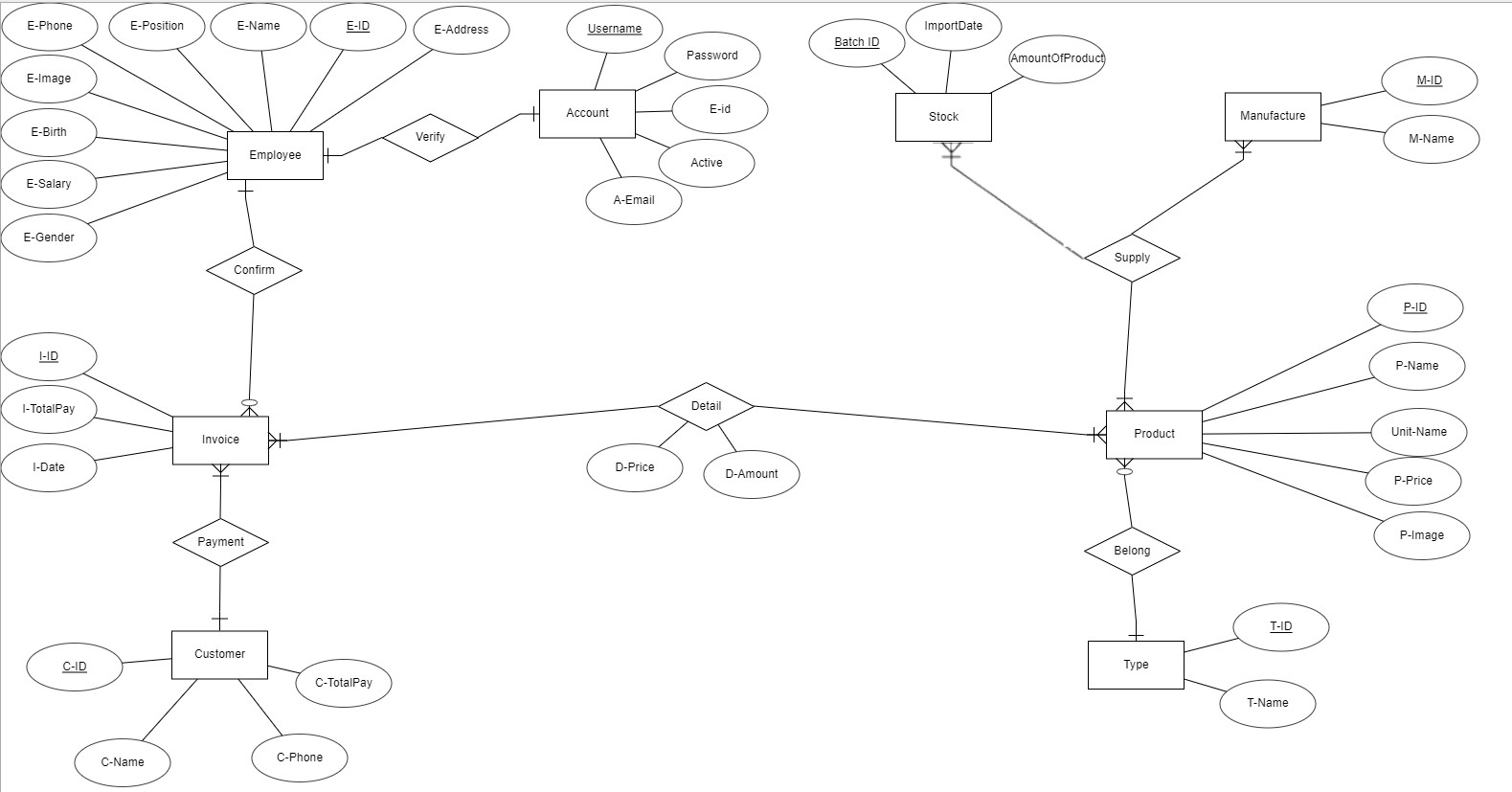
* Home page: helps users access user functions such as creating invoices and viewing product information,...
* Invoice creation page allows sales employee to update invoices in two ways, with customers and without customers.
* The product page allows employee to view list product information.
* Customer page allows employees to add customers

Manager account interface:

* Home page: helps users access user functions such as employee management, invoice management, product management, customer management see gender statistics, add supply, manufacture management, stock management, type management.
* The employee management page displays a list of employees and detailed information about the employee. Management can perform functions such as adding, editing, deleting, updating and searching for employee information in the store by name and id.
* The type management page displays a list of product types. Management can perform functions such as adding, removing and updating product types.
* The product management page displays the product list and detailed information about the product. Management can perform functions such as adding, editing, deleting, updating and searching product information by product name and type product.
* The stock management page allows to enter more information of new product batches such as batch id, import date, amount of product, edit or delete batches.
* Invoice creation page allows management staff to create new invoices and edit, remove invoices.
* Statistical management page showing male and female employees of the store.
* Management can perform functions such as adding, removing and updating manualfacture

DATABASE ANALYSIS AND DESIGN

Model ERD



Login(Username, Password, E-ID, Active, Email)

Employee (E-ID, E-Name, E-Position, E-Salary, E-Phone, E-Address, E-Image, E-Gender, E-Brith)

Invoice (I-ID, E-ID, C-ID, I-TotalPay, I-Date)

Customer (C-ID, C-Name, C-TotalPay, C-Phone)

Detail (I-ID, P-ID, D-Price, D-Amount)

Product (P-ID, P-Name, P-Price, T-ID, Unit-Name, P-image)

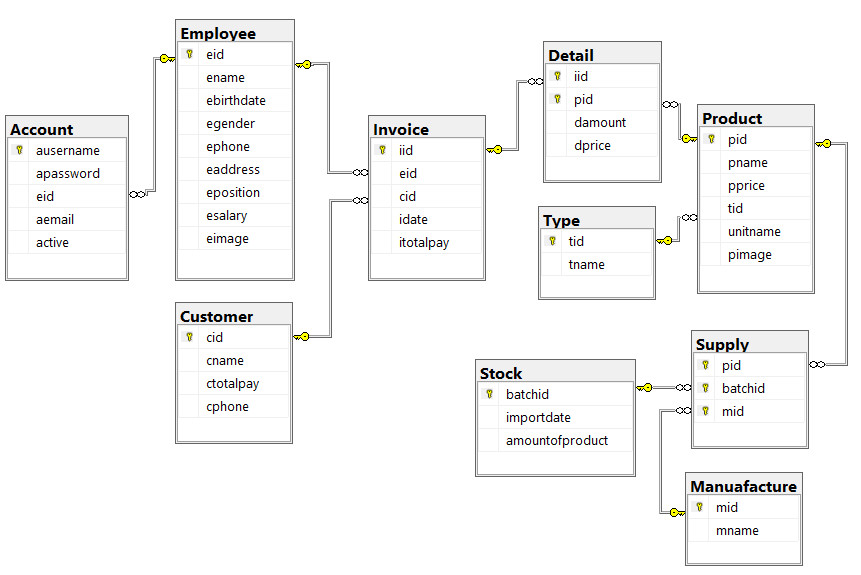
Type (T-ID, T-Name)

Manuafacture (M-ID, M-Name)

Stock (Batch-ID, ImportDate, AmountOfProduct)

Supply ( M-ID, P-ID, Batch-ID)

Model relationships between tables



SYSTEM CONFIGURATION AND INSTALLATION

Database Creation and Constraints

CREATE DATABASE StoreManagement

GO

USE StoreManagement

GO

--Phan Code Tao Bang

--Nhan Vien

CREATE TABLE [dbo].[Employee] (

[eid] nchar(20) NOT NULL,

[ename] nchar(40) NOT NULL,

[ebirthdate] date NULL,

[egender] nchar(15) NULL,

[ephone] nchar(15) NULL,

[eaddress] nchar(50) NULL,

[eposition] nchar(40) NULL,

[esalary] float NULL,

[eimage] image NULL,

CONSTRAINT [PK\_Employee] PRIMARY KEY CLUSTERED ([eid] ASC),

)

Go

CREATE TABLE [Account] (

[ausername] nchar(15) NOT NULL,

[apassword] int NOT NULL,

[eid] nchar(20) NOT NULL,

[aemail] VARCHAR(100) NULL,

[active] BIT NULL,

CONSTRAINT [PK\_Account] PRIMARY KEY CLUSTERED ([ausername] ASC),

CONSTRAINT [FK\_Account\_Employee\_eid] FOREIGN KEY ([eid]) REFERENCES [Employee] ([eid]) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT [CK\_Account\_aemail] CHECK ([aemail] like '%\_@%\_.\_%')

)

Go

--Khach Hang

CREATE TABLE [Customer] (

[cid] nchar(20) NOT NULL,

[cname] nchar(40) NULL,

[ctotalpay] float NULL,

[cphone] nchar(15) NOT NULL,

CONSTRAINT [PK\_Customer] PRIMARY KEY CLUSTERED ([cid] ASC)

)

Go

--Loai San Pham

CREATE TABLE [Type] (

[tid] nchar(20) NOT NULL,

[tname] nchar(20) NOT NULL,

CONSTRAINT [PK\_Type] PRIMARY KEY CLUSTERED ([tid] ASC),

)

Go

--Kho Hang

CREATE TABLE [Stock] (

[batchid] nchar(20) NOT NULL,

[importdate] date NULL,

[amountofproduct] int NULL,

CONSTRAINT [PK\_Stock] PRIMARY KEY CLUSTERED ([batchid] ASC),

)

Go

--Nha San Xuat

CREATE TABLE [Manuafacture] (

mid nchar(20) NOT NULL,

mname nchar(40) NULL,

CONSTRAINT [PK\_Manuafacture] PRIMARY KEY CLUSTERED ([mid] ASC),

)

Go

--San Pham

CREATE TABLE [Product] (

[pid] nchar(20) NOT NULL,

[pname] nchar(30) NOT NULL,

[pprice] float NOT NULL,

[tid] nchar(20) NOT NULL,

[unitname] nchar(20) NULL,

[pimage] image NULL,

CONSTRAINT [PK\_Product] PRIMARY KEY CLUSTERED ([pid] ASC),

CONSTRAINT [FK\_Product\_Type\_tid] FOREIGN KEY ([tid]) REFERENCES [Type] ([tid]) ON UPDATE CASCADE

)

Go

--Hoa Don

CREATE TABLE [Invoice] (

iid nchar(20) NOT NULL,

eid nchar(20) NOT NULL,

cid nchar(20) NULL,

idate datetime NULL,

itotalpay float NULL,

CONSTRAINT [PK\_Invoice] PRIMARY KEY CLUSTERED ([iid] ASC),

CONSTRAINT [FK\_Invoice\_Employee\_eid] FOREIGN KEY ([eid]) REFERENCES [Employee] ([eid]) ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT [FK\_Invoice\_Customer\_cid] FOREIGN KEY ([cid]) REFERENCES [Customer] ([cid]) ON UPDATE CASCADE ON DELETE CASCADE

)

Go

--Chi Tiet Hoa Don

CREATE TABLE [Detail] (

iid nchar(20) NOT NULL,

pid nchar(20) NOT NULL,

dprice float NULL,

damount int NULL,

CONSTRAINT [PK\_Detail] PRIMARY KEY CLUSTERED ([iid] ASC, [pid]),

CONSTRAINT [FK\_Detail\_Invoice\_iid] FOREIGN KEY ([iid]) REFERENCES [Invoice] ([iid]) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT [FK\_Detail\_Product\_pid] FOREIGN KEY ([pid]) REFERENCES [Product] ([pid]) ON DELETE CASCADE ON UPDATE CASCADE

)

Go

--Quan he 3 ngoi cung cap san pham

CREATE TABLE [Supply] (

[pid] nchar(20) NOT NULL,

[batchid] nchar(20) NOT NULL,

[mid] nchar(20) NOT NULL,

CONSTRAINT [PK\_Supply] PRIMARY KEY CLUSTERED ([pid] ASC, [batchid], [mid]),

CONSTRAINT [FK\_Supply\_Supply\_batchid] FOREIGN KEY ([batchid]) REFERENCES [Stock] ([batchid]) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT [FK\_Supply\_Supply\_mid] FOREIGN KEY ([mid]) REFERENCES [Manuafacture] ([mid]) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT [FK\_Supply\_Supply\_pid] FOREIGN KEY ([pid]) REFERENCES [Product] ([pid]) ON DELETE CASCADE ON UPDATE CASCADE

)

Go

Creating statements in Stored Procedure

Add account

CREATE PROC [dbo].[SP\_AddAccount]

@user NCHAR(15),

@pass INT,

@type NCHAR(20),

@mail VARCHAR(100),

@status BIT

as

begin

INSERT INTO Account(ausername,apassword,eid,aemail,active) VALUES(@user,@pass,@type,@mail,@status)

end Go

Add employee

Create Proc SP\_AddEmployee

@id nchar(20),

@name nchar(40),

@birthdate date,

@gender nchar(15),

@phone nchar(15),

@address nchar(50),

@position nchar(40),

@salary float,

@image image

As

Begin

if(year(getdate())-year(@birthdate)>18)

begin

INSERT INTO Employee (eid, ename, ebirthdate, egender, ephone, eaddress, eposition, esalary, eimage)

Values(@id,@name, @birthdate, @gender, @phone, @address, @position, @salary,@image)

end

else begin

Print('Invalid Age') Rollback TRANSACTION

end

End

Go

Update employee

Create Proc SP\_UpdateEmployee

@id nchar(20),

@name nchar(40),

@birthdate date,

@gender nchar(15),

@phone nchar(15),

@address nchar(50),

@position nchar(40),

@salary float,

@image image

As

Begin

if(year(getdate())-year(@birthdate)>18)

begin

Update Employee SET ename=@name, ebirthdate=@birthdate, egender=@gender, ephone=@phone, eaddress=@address,eposition=@position, esalary=@salary, eimage=@image WHERE Employee.eid=@id

end

else begin

print('Invalid Age') Rollback TRAN

end

End

Go

Delete employee

Create Proc SP\_DeleteEmployee

@id nchar(20)

As

Begin Delete From Employee Where Employee.eid=@id

end

Go

List employee

Create Proc SP\_ListEmployee

As

Begin

Select eid AS [Employee ID],ename AS [Name],ebirthdate AS [Birthdate],egender AS [Gender],ephone AS [Phone],eaddress AS [Address],eposition AS [Position],esalary AS [Salary],eimage AS [Image] From Employee

end

Go

Find employee

Create Proc SP\_FindEmployee

@id nchar(20)

As

Begin

Select eid From Employee Where eid=@id

End

Go

Add customer

Create Proc SP\_AddCustomer

@c nchar(20),

@name nchar(40),

@total float,

@phone nchar(15)

As

Begin

if(@phone like '0%' AND Len(@phone) = 10)

begin

INSERT INTO Customer (cid, cname, ctotalpay, cphone) Values (@c,@name,@total,@phone)

end

else begin

Print('Invalid Phone') Rollback TRANSACTION

end

End

Go

Add product

Create Proc SP\_AddProduct

@id nchar(20),

@name nchar(30),

@price float,

@typeid nchar(20),

@unitname nchar(20),

@image image

As

Begin

INSERT INTO Product (pid, pname, pprice, tid,unitname,pimage) Values (@id,@name,@price,@typeid,@unitname,@image)

End

Go

Update product

Create Proc SP\_UpdateProduct

@id nchar(20),

@name nchar(30),

@price float,

@typeid nchar(20),

@unitname nchar(20),

@image image

As

Begin Update Product Set pname=@name, pprice=@price,tid=@typeid,unitname=@unitname,pimage=@image Where pid=@id

End

go

Delete product

Create Proc SP\_DeleteProduct

@id nchar(20)

As

Begin Delete From Product Where pid=@id

End

Go

Add invoice

Create Proc SP\_AddInvoice

@i nchar(20),

@e nchar(20),

@c nchar(20),

@date datetime,

@total float

AS

Begin

INSERT INTO Invoice (iid, eid, cid, idate, itotalamount) VALUES (@i,@e, @c, @date, @total)

End

Go

Update invoice

Create Proc SP\_UpdateInvoice

@i nchar(20),

@e nchar(20),

@c nchar(20),

@date datetime,

@total float

As

Begin Update Invoice Set eid=@e,cid=@c,idate=@date,itotalamount=@total WHERE iid=@i

End

Go

Delete invoice

Create Proc SP\_DeleteInvoice

@i nchar(20)

As

Begin DELETE FROM Invoice WHERE iid =@i

End

Go

Add detail

Create Proc SP\_AddDetail

@id nchar(20),

@pid nchar(20),

@price float,

@amount int

As

Begin INSERT INTO Detail (iid, pid, dprice, damount) VALUES (@id,@pid, @price, @amount)

End

Go

Update detail

Create Proc SP\_UpdateDetail

@id nchar(20),

@pid nchar(20),

@price float,

@amount int

As

Begin Update Detail Set pid=@pid,dprice=@price,damount=@amount Where iid=@id

End

Go

Delete detail

Create Proc [dbo].[SP\_DeleteDetail]

@id nchar(20),

@pid NCHAR(20)

AS

Begin Delete From Detail Where iid=@id AND pid =@pid

End

Go

Add manufacturer

Create Proc SP\_AddManuafacture

@mid nchar(20),

@name nchar(40)

As

Begin INSERT INTO Manuafacture (mid, mname) VALUES (@mid,@name)

End

Go

Update manufacturer

Create Proc SP\_UpdateManuafacture

@mid nchar(20),

@name nchar(40)

AS

Begin Update Manuafacture Set mname=@name Where mid=@mid

End

Go

Delete manufacturer

Create Proc SP\_DeleteManuafacture

@mid nchar(20)

As

Begin Delete From Manuafacture Where mid=@mid

End

Go

Add stock

Create Proc SP\_AddStock

@batid nchar(20),

@date date,

@amount int

As

Begin INSERT INTO Stock (batchid,importdate,amountofproduct) Values (@batid,@date,@amount)

End

Go

Update stock

Create Proc SP\_UpdateStock

@batid nchar(20),

@date date,

@amount int

As

Begin Update Stock Set importdate=@date,amountofproduct=@amount Where batchid=@batid

End

Go

Delete stock

Create Proc SP\_DeleteStock

@batid nchar(20)

As

Begin Delete From Stock Where batchid=@batid

End

Go

Add types

Create Proc SP\_AddTypes

@tid nchar(20),

@name nchar(20)

As

Begin INSERT INTO [Type] (tid,tname) Values (@tid,@name)

End

Go

Update type

Create Proc SP\_UpdateType

@tid nchar(20),

@name nchar(20)

As

Begin Update [Type] Set tname=@name Where tid=@tid

End

Go

Delete type

Create Proc SP\_DeleteType

@tid nchar(20)

As

Begin Delete From [Type] Where tid=@tid

End

Go

Database connection

public class MY\_DB

{

//Phan quyen doi voi Manager

SqlConnection con1 = new SqlConnection(@"Data Source=MSI;Initial Catalog=StoreManagement;Persist Security Info=True;User ID=managerAdmin;Password=1");

public SqlConnection getConnectionManager

{

get

{

return con1;

}

}

public void openConnectionManager()

{

if ((con1.State == ConnectionState.Closed))

{

con1.Open();

}

}

public void closeConnectionManager()

{

if ((con1.State == ConnectionState.Closed))

{

con1.Close();

}

}

//Phan quyen doi voi Employee

SqlConnection con2 = new SqlConnection(@"Data Source=MSI;Initial Catalog=StoreManagement;Persist Security Info=True;User ID=employeeUser;Password=1");

public SqlConnection getConnectionEmployee

{

get

{

return con2;

}

}

public void openConnectionEmployee()

{

if ((con2.State == ConnectionState.Closed))

{

con2.Open();

}

}

public void closeConnectionEmployee()

{

if ((con2.State == ConnectionState.Closed))

{

con2.Close();

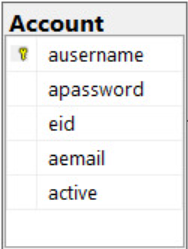
}

}

}

Decentralization

Database table for authorization



Decentralized code on the database

Create Role employee

--View for employee

GRANT SELECT ON V\_CheckAccount TO employee

GRANT SELECT ON V\_Product TO employee

GRANT SELECT ON V\_Customer TO employee

GRANT SELECT ON V\_Invoice TO employee--Function

GRANT SELECT ON V\_PersonalInfo TO employee

--Execute

GRANT EXECUTE ON SP\_UpdateInvoice TO employee

GRANT EXECUTE ON SP\_UpdateInvoiceNotCustomer TO employee

GRANT EXECUTE ON SP\_AddInvoice TO employee

GRANT EXECUTE ON SP\_AddCustomer TO employee

GRANT EXECUTE ON SP\_AddDetail TO employee

GRANT EXECUTE ON SP\_DeleteDetail TO employee

GRANT EXECUTE ON SP\_UpdateCustomer TO employee

GO

USE StoreManagement

GO

Create ROLE manager

GRANT SELECT ON V\_CheckAccount TO manager

GRANT SELECT ON V\_Product TO manager

GRANT SELECT ON V\_Customer TO manager

GRANT SELECT ON V\_PersonalInfo TO manager

GRANT SELECT ON V\_Invoice TO manager

GRANT EXECUTE ON SP\_AddInvoice TO manager

GRANT EXECUTE ON SP\_AddCustomer TO manager

GRANT EXECUTE ON SP\_AddDetail TO manager

GRANT EXECUTE ON SP\_UpdateCustomer TO manager

GRANT EXECUTE ON SP\_AddEmployee TO manager

GRANT EXECUTE ON SP\_AddManuafacture TO manager

GRANT EXECUTE ON SP\_AddProduct TO manager

GRANT EXECUTE ON SP\_AddStock TO manager

GRANT EXECUTE ON SP\_AddSupply TO manager

GRANT EXECUTE ON SP\_AddTypes TO manager

GRANT EXECUTE ON SP\_AddAccount TO manager

GRANT EXECUTE ON SP\_DeleteCustomer TO manager

GRANT EXECUTE ON SP\_DeleteDetail TO manager

GRANT EXECUTE ON SP\_DeleteEmployee TO manager

GRANT EXECUTE ON SP\_DeleteInvoice TO manager

GRANT EXECUTE ON SP\_DeleteManuafacture TO manager

GRANT EXECUTE ON SP\_DeleteProduct TO manager

GRANT EXECUTE ON SP\_DeleteStock TO manager

GRANT EXECUTE ON SP\_DeleteType TO manager

GRANT EXECUTE ON SP\_FindEmployee TO manager

GRANT EXECUTE ON SP\_ListEmployee TO manager

GRANT EXECUTE ON SP\_UpdateCustomer TO manager

GRANT EXECUTE ON SP\_UpdateDetail TO manager

GRANT EXECUTE ON SP\_UpdateEmployee TO manager

GRANT EXECUTE ON SP\_UpdateInvoice TO manager

GRANT EXECUTE ON SP\_UpdateManuafacture TO manager

GRANT EXECUTE ON SP\_UpdateProduct TO manager

GRANT EXECUTE ON SP\_UpdateStock TO manager

GRANT EXECUTE ON SP\_UpdateType TO manager

go

Decentralized code on C#

//Phan quyen tai khoan

SqlConnection con = new SqlConnection(@"Data Source=MSI;Initial Catalog=StoreManagement;Persist Security Info=True;User ID=" + user + ";Password=" + pass);

Triggers and transactions

Settings to create login accounts according to user permissions

CREATE TRIGGER CreateUserLogin ON Account

FOR INSERT

AS

BEGIN

DECLARE @user NCHAR(15), @password INT, @type NCHAR(20), @db\_name NVARCHAR(MAX), @active BIT

SET @db\_name = DB\_NAME()

SELECT @user = ausername, @password = apassword, @type = eid, @active = active

FROM inserted

EXEC('CREATE LOGIN [' + @user + '] WITH PASSWORD = '''+ @password +''', DEFAULT\_DATABASE=[' + @db\_name + ']')

EXEC('CREATE USER [' + @user + '] FOR LOGIN [' + @user + ']')

IF @type LIKE 'manager%'

BEGIN

EXEC sp\_addrolemember 'db\_owner', @user

EXEC sp\_addrolemember 'db\_accessadmin', @user

EXEC sp\_addrolemember 'db\_securityadmin', @user

EXEC sp\_addrolemember 'manager', @user

EXEC('USE master; GRANT ALTER ANY LOGIN TO [' + @user + '] WITH GRANT OPTION')

END

ELSE IF @type LIKE 'employee%'

BEGIN

EXEC sp\_addrolemember 'employee', @user

END

IF @active = 0

EXEC('ALTER LOGIN [' + @user + '] DISABLE')

ELSE

EXEC('ALTER LOGIN [' + @user + '] ENABLE')

END

GO

Install login account updates according to user permissions

CREATE TRIGGER UpdateUserLogin ON Account

FOR UPDATE

AS

BEGIN

DECLARE @old\_active BIT, @new\_active BIT, @old\_user NCHAR(15), @new\_user NCHAR(15), @old\_password INT, @new\_password INT, @type NCHAR(20), @db\_name NVARCHAR(MAX)

SET @db\_name = DB\_NAME()

SELECT @old\_user = ausername, @old\_password = apassword, @old\_active = active

FROM deleted

SELECT @new\_user = ausername, @new\_password = apassword, @type = eid, @new\_active = active

FROM inserted

IF (@new\_user = @old\_user AND @new\_password = @old\_password AND @old\_active = @new\_active)

RETURN

ELSE IF (@new\_user <> @old\_user) --Quan Ly

BEGIN

EXEC('DROP USER [' + @old\_user + ']')

EXEC('DROP LOGIN [' + @old\_user + ']')

EXEC('CREATE LOGIN [' + @new\_user + '] WITH PASSWORD = ''' + @new\_password + ''', DEFAULT\_DATABASE=[' + @db\_name + ']')

EXEC('CREATE USER [' + @new\_user + '] FOR LOGIN [' + @new\_user + ']')

IF @type LIKE 'manager%'

BEGIN

EXEC sp\_addrolemember 'db\_owner', @new\_user

EXEC sp\_addrolemember 'db\_accessadmin', @new\_user

EXEC sp\_addrolemember 'db\_securityadmin', @new\_user

EXEC sp\_addrolemember 'manager', @new\_user

EXEC('USE master; GRANT ALTER ANY LOGIN TO [' + @new\_user + '] WITH GRANT OPTION')

END

ELSE IF @type LIKE 'employee%'

BEGIN

EXEC sp\_addrolemember 'employee', @new\_user

EXEC('USE master; GRANT ALTER ANY LOGIN TO [' + @new\_user + '] WITH GRANT OPTION')

END

END

ELSE IF (@new\_password <> @old\_password) --Ca 2

BEGIN

EXEC('ALTER LOGIN [' + @new\_user + '] WITH PASSWORD = ''' + @new\_password + ''' OLD\_PASSWORD = ''' + @old\_password + '''')

END

ELSE --Quan Ly

BEGIN

IF @new\_active = 0

EXEC('ALTER LOGIN [' + @new\_user + '] DISABLE')

ELSE

EXEC('ALTER LOGIN [' + @new\_user + '] ENABLE')

END

END

GO

CREATE TRIGGER DeleteUserLogin ON Account

FOR DELETE

AS

BEGIN

DECLARE @user NCHAR(15)

SELECT @user = ausername

FROM deleted

EXEC('DROP USER [' + @user + ']')

EXEC('DROP LOGIN [' + @user + ']')

END

GO

Setting to delete login accounts according to user permissions

CREATE TRIGGER DeleteUserLogin ON Account

FOR DELETE

AS

BEGIN

DECLARE @user NCHAR(15)

SELECT @user = ausername

FROM deleted

EXEC('DROP USER [' + @user + ']')

EXEC('DROP LOGIN [' + @user + ']')

END

Setting to update total pay for invoice

CREATE TRIGGER [dbo].[Update\_TotalPay\_Invoice] ON [dbo].[Detail]

AFTER INSERT

AS

BEGIN

DECLARE @totalamount float, @iid nchar(20), @sum float

Select @iid=iid From inserted

SELECT @totalamount=damount\*dprice FROM inserted

Select @sum= Sum(itotalpay) From Invoice Where invoice.iid=@iid

UPDATE Invoice SET itotalpay= @sum + @totalamount Where Invoice.iid=@iid

END

GO

Setting to decrease total pay for invoice

CREATE TRIGGER [dbo].[Decrease\_TotalPay\_Invoice] ON [dbo].[Detail]

AFTER DELETE

AS

BEGIN

DECLARE @totalamount float

SELECT @totalamount=damount\*dprice FROM deleted

UPDATE Invoice SET itotalpay= itotalpay - @totalamount

END

GO

Setting to check the invoice exists or not

CREATE TRIGGER [dbo].[CheckInvoice\_Exist] ON [dbo].[Detail] INSTEAD OF INSERT

AS

BEGIN

Declare @tmp int, @tmp1 nchar(20)

Select @tmp1=iid from inserted

DECLARE @totalamount float

DECLARE @pid NCHAR(20), @dprice float, @damount int

SELECT @pid=pid, @dprice=dprice, @damount=damount FROM inserted

SELECT @totalamount = damount\*dprice FROM inserted

If Exists( Select iid From Invoice Where iid=@tmp1)

begin

if Exists(Select pid From Detail Where pid=@pid and iid=@tmp1)

begin

Update Detail Set damount=damount+@damount Where pid=@pid and iid=@tmp1

end

else

begin

INSERT INTO Detail(iid,pid,damount,dprice) VALUES(@tmp1,@pid,@damount,@dprice)

end

End

Else

BEGIN

DECLARE @date datetime

SET @date =getdate()

Insert Into Invoice(iid,eid,idate,itotalpay) Values(@tmp1,'manager1',@date,0)

INSERT INTO Detail(iid,pid,damount,dprice) VALUES(@tmp1,@pid,@damount,@dprice)

END

END

GO

Setting to update the invoice

CREATE TRIGGER [dbo].[UpdateInvoice] ON [dbo].[Detail]

AFTER Update

AS

BEGIN

Declare @damount int ,@dprice float,@iid nchar(20),@pid nchar(20),@sum float

Select @pid=pid,@iid=iid from inserted

Select @damount=damount,@dprice=dprice,@iid=iid From Detail Where pid=@pid and iid=@iid

Update Invoice Set itotalpay=0 Where Invoice.iid=@iid

Select @sum= Sum(itotalpay) From Invoice Where invoice.iid=@iid

UPDATE Invoice SET itotalpay= @sum + @damount\*@dprice Where Invoice.iid=@iid

END

Function

Total number of employees

CREATE FUNCTION FN\_CountEmployee()

RETURNS INT

AS

BEGIN

DECLARE @SL INT

SELECT @SL=Count(eid) FROM Employee

RETURN @SL

END

GO

Total number of customers

CREATE FUNCTION FN\_CountCustomer()

RETURNS INT

AS

BEGIN

DECLARE @SLKH INT

SELECT @SLKH=Count(cid) FROM Customer

RETURN @SLKH

END

GO

Total number of products

CREATE FUNCTION FN\_CountProducts()

RETURNS INT

AS

BEGIN

DECLARE @SLSP INT

SELECT @SLSP=Count(pid) FROM Product

RETURN @SLSP

END

GO

Check account

CREATE FUNCTION V\_CheckAccount(@user NCHAR(15),@pass INT)

RETURNS TABLE

AS

RETURN

(

select \* from Account where ausername = @user and apassword = @pass

)

GO

Find product by name and type

CREATE FUNCTION FN\_FindProduct(@string VARCHAR(MAX))

RETURNS TABLE

AS

RETURN

(

Select \* FROM V\_Product WHERE CONCAT([Product Name],[Type ID]) LIKE '%'+@string+'%'

)

GO

Find customer

CREATE FUNCTION FN\_FindCustomer(@Search NCHAR(15))

RETURNS TABLE

AS

RETURN

(

SELECT cid, cname FROM Customer WHERE cphone =@Search

)

GO

Show detail of invoice

Create Function FN\_ShowDetail(@iid nchar(20))

returns Table

As

Return (

Select \*From Detail Where iid=@iid

)

Go

Show detail

Auto show information of product

Create Function [dbo].[FN\_AutoShowProductInfo](@id NCHAR(20))

Returns Table

As

Return

(

Select pname,pprice,unitname, pimage FROM Product WHERE pid=@id

)

GO

Total pay of invoice

CREATE FUNCTION [dbo].[FN\_GetTotalPay](@iid NCHAR(20))

RETURNS float

AS

BEGIN

DECLARE @total float

SELECT @total=itotalpay FROM Invoice WHERE iid=@iid

RETURN @total

END

GO

Search employees by id

CREATE FUNCTION [dbo].[FN\_SearchEmployeeByID](@eid nchar(40) null)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM Employee

WHERE eid =@eid

)

GO

Employee gender statistics

CREATE FUNCTION Fn\_ListGender(@gen nchar(10))

RETURNS INT

AS

BEGIN

DECLARE @SL INT

SELECT @SL=Count(eid) FROM Employee Where egender=@gen

RETURN @SL

END

Go

View

View for products

--Thong tin san pham

CREATE VIEW V\_Product AS Select pid AS [Product ID],pname AS [Product Name],pprice AS [Price], tid AS [Type ID], unitname AS [Unit], pimage AS [Image] From [Product]

GO

--Thong tin san pham (ngoai tru tid va unitname)

CREATE VIEW V\_Product\_Invoice AS Select pid AS [Product ID],pname AS [Product Name],pprice AS [Price], pimage AS [Image] From [Product]

GO

View for customers

CREATE VIEW V\_Customer AS SELECT cid AS [ID], cname as [Name], cphone as [Phone] FROM [Customer]

go

View for type

CREATE VIEW V\_Type AS SELECT \* FROM [Type]

GO

View for stock

CREATE VIEW V\_Stock AS SELECT batchid AS [Batch ID], importdate AS [Import Date], amountofproduct [Amount Of Product] FROM [Stock]

GO