**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

**QUẢN LÝ KHÁCH SẠN**

***Giáo viên hướng dẫn:***

**TS. HOÀNG LONG**

***Sinh viên thực hiện:***

**Nguyễn Huỳnh Anh Trực 17110247**

**Đoàn Quốc Hùng 17110154**

**THÀNH PHỐ HỒ CHÍ MINH – 10/12/2019**

**MỤC LỤC**

[LỜI NÓI ĐẦU 1](#_Toc26852166)

[I. GIỚI THIỆU 2](#_Toc26852167)

[*1.* *Lý do xây dựng phần mềm* 2](#_Toc26852168)

[*2.* *Mục tiêu của phần mềm* 2](#_Toc26852169)

[*3.* *Chức năng cơ bản* 2](#_Toc26852170)

[II. MÔ TẢ PHẦN MỀM VÀ DỮ LIỆU 3](#_Toc26852171)

[*1.* *Mục tiêu đề tài* 3](#_Toc26852172)

[*2.* *Đối tượng người dùng* 3](#_Toc26852173)

[*3.* *Mô tả.* 4](#_Toc26852174)

[*4.* *Mô hình ERD* 4](#_Toc26852175)

[*5.* *Cơ sở dữ liệu* 4](#_Toc26852176)

[III. MÔ TẢ CẤU TRÚC CÁC BẢNG 12](#_Toc26852177)

[IV. DANH SÁCH CÁC STORED PROCEDURE: 13](#_Toc26852178)

[V. DANH SÁCH CÁC FUNCTION 65](#_Toc26852179)

[VI. DANH SÁCH CÁC TRIGGER 67](#_Toc26852180)

[VII. GIAO DIỆN 71](#_Toc26852181)

[VIII. TỔNG KẾT 76](#_Toc26852182)

[*1.* *Nhận xét* 76](#_Toc26852183)

[*2.* *Khuyết điểm* 76](#_Toc26852184)

[*3.* *Hướng phát triển* 76](#_Toc26852185)

# LỜI NÓI ĐẦU

Thế giới đang ngày một phát triển vượt trội với những công nghệ kĩ thuật không ngừng ra đời. Những phát minh, những sáng tạo đang ngày một sinh sôi và dần biến đổi cả thế giới. Chỉ một cái nhắm mắt thì trên thế giới đâu đó đã cho đời một công nghệ mới và chúng ta sẽ phải học hỏi từng ngày để đi kịp với thời đại. Có người nói rằng: ”Công nghệ là thứ gần gũi nhất với ma thuật còn tồn tại trên thế giới này.”. Những phát minh chung quanh chúng ta bắt nguồn từ những ý tưởng, những việc mà con người không thể thực hiện một cách hoàn toàn chính xác.

Ngay từ xa xưa việc quản lý một đội ngũ hay một tổ chức nào đó đã là việc phổ biến trong xã hội. Người ta không thể nào nhớ hết tất cả thông các thành viên trong tổ chức đó cũng như tính toán lương bổng, từ đó việc ghi chép thông tin được ra đời. Và dĩ nhiên việc ghi chép đó hầu như là trên giấy bút thời bấy giờ. Trải qua một giai đoạn tưởng chừng như không có vấn đề gì xảy ra, người ta phát hiện rằng việc ghi chép thông tin ra giấy tốn rất nhiều tài nguyên về giấy, mực. Song cũng dễ bị thất thoát thông tin dữ liệu, việc tính toán như lương bổng cũng không thể hoàn toàn chính xác. Thật may mắn là sau đó máy móc thiết bị phần mềm đang được chú ý phát triển. Từ đó, yêu cầu phải có một hệ thống, một chương trình có chức năng quản lý, tính toán thay cho con người đáp ứng nhu cầu về thông tin phải thật nhanh chóng và chính xác. Chỉ cần các thao tác nhỏ thì họ có thể quản lý tối ưu tổ chức của mình. Ứng dụng Quản lý khách sạn sau đây là một ví dụ minh họa cho phần mềm giúp quản lý khách sạn một cách nhanh chóng qua các thao tác đặt phòng, thuê phòng, tính giá phòng, khách hàng, v.v…

# GIỚI THIỆU

## *Lý do xây dựng phần mềm*

* Quản lý khách sạn là một trong những việc phải có trong kinh doanh mô hình khách sạn. Nó giúp cho người dùng có thể đặt phòng, trả phòng, thanh toán, đồng thời quản lý cả nhân viên mà cụ thể là tiếp tân. Đây là một công việc vô cùng quan trọng cho việc kinh doanh của khách sạn và cũng đảm bảo sự thoải mái cho khách hàng khi đến thuê phòng.
* Phần mềm quản lý khách sạn giúp cho những việc nêu trên được thực thi một cách nhanh chóng. Giúp cho việc tính toán giá của hóa đơn được chính xác thay vì chúng ta phải tính toán thì giờ đã có phần mềm làm thay điều đó.
* Ngoài ra phần mềm Quản lý khách sạn còn cho biết tình hình của khách sạn, tình trạng các phòng thuê.

## *Mục tiêu của phần mềm*

* Các mục tiêu chính của phần mềm:
  + Xây dựng một ứng dụng có thể lưu trữ và cập nhật thông tin khách hàng, nhân viên, phòng và dịch vụ.
  + Xây dựng một ứng dụng có chức năng cho khách hàng đặt phòng, nhận phòng và xuất hóa đơn khi khách hàng thanh toán trả phòng.
  + Chức năng lưu thông tin khách hàng kèm theo phòng và dịch vụ mà khách hàng đó thuê được và tính ra giá tiền mà khách hàng đó phải trả theo từng mục tương ứng.
  + Chức năng thống kê doanh thu đặt được trong các tháng lân cận

## *Chức năng cơ bản*

* Đầu tiên hệ thống sẽ yêu cầu người dùng đăng nhập vào hệ thống, thông tin người dùng và mật khẩu được lưu trong bảng nhân viên. Sẽ có 2 đối tương người dùng chính: tiếp tân và administrator.
* Với admin là toàn quyền sử dụng ứng dụng và nhân viên sẽ bị hạn chế một số quyền trong ứng dụng. Trong đó:

Admin: Có toàn quyền chỉnh sửa,thêm xóa, tác động đến các bảng trong cơ sỡ dữ liệu, có quyền được truy cập vào chức năng phân tích thống kê doanh thu của khách sạn, cũng như có quyền thực hiện các thao tác chính trong khách sạn bao gồm Booking, Checkin và Checkout

Tiếp tân: Chỉ có quyền thực hiện 3 thao tác Booking, Checkin và Checkout cho khách

* Khi đăng nhập thành công, sẽ hiện ra các nút chức năng như xuất bảng phòng, bảng nhân viên, bảng dịch vị, bảng khách hàng,…
* Chức năng tìm kiếm cho nhân viên khi muốn tìm kiếm một thông tin thông qua một thuộc tính nào đó.
* Những chức năng cơ bản như thêm, sửa, xóa dòng trong bảng.
* Thống kê tổng doanh thu đạt được trong tháng và năm.

# MÔ TẢ PHẦN MỀM VÀ DỮ LIỆU

## *Mục tiêu đề tài*

* Giúp ban quản trị có thể theo dõi tình hình kinh doanh của khách sạn, doanh thu lớn hay nhỏ qua đó có thể đưa ra được nhận xét của mình qua thông tin đó.
* Cho phép người dùng có thể tìm kiếm thông tin một cách chính xác.
* Giúp cập nhật thông tin khách sạn về các mục kinh doanh một cách nhanh chóng nhất.
* Thực hiện các nhiệm vụ đặt phòng cho khách hàng, làm thủ tục nhận phòng cũng như trả phòng cho khách hàng một cách nhanh chóng, cung cấp cho khách hàng các khoảng dịch vụ khi khách hàng có yêu cầu, giúp cho khách hàng có một sự thoải mái và hài lòng.

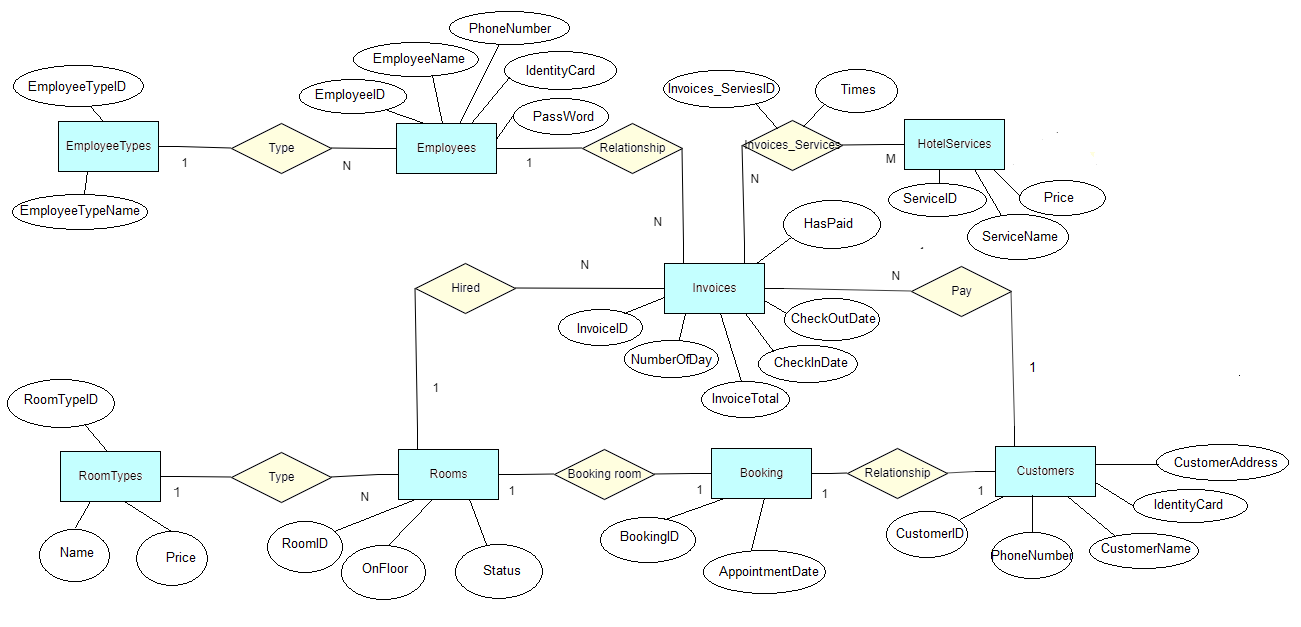
## *Đối tượng người dùng*

* Ban quản trị: có toàn quyền sử dụng phần mềm, có thể thêm sửa xóa các mục, thống kê doanh thu khách sạn, cũng như có thể thao tác đặt phòng, nhận phòng và thanh toán phòng cho khách.
* Nhân viên: Bị hạn chế một số quyền hạn khi sử dụng phần mềm, chỉ có thể thao tác đặt phòng, nhận phòng và xuất hóa đơn cho khách hàng.

## *Mô tả.*

* Trong một khách sạn có một số lượng phòng nhất định, có thể thêm phòng tùy vào người quản trị viên, một phòng có một loại phòng, một loại phòng có thể thuộc nhiều phòng, một phòng sẽ có một mục tình trạng phòng để người sử dụng có thể biết phòng đó đã được thuê, đã được đặt trước hay vẫn còn trống. Tương tự với mục nhân viên, một nhân viên có một loại nhân viên và một loại nhân viên có thể có nhiều nhân viên. Có một bảng đặt phòng, mỗi phòng trong 1 thời điểm chỉ có thể được đặt bởi 1 khách hàng. Bảng hóa đơn gồm mã phòng, mã nhân viên tạo hóa đơn, mã khách hàng cụ thể. Ngoài ra, có mục dịch vụ cho khách hàng sử dụng khi cần thiết. Khi thuê khách sạn, khách hàng có thể thuê nhiều dịch vụ để sử dụng. Khi nhận phòng thì hóa đơn sẽ lưu lại ngày vào của khách hàng, khi trả phòng, ngày hiện tại sẽ là ngày ra của khách hàng, căn cứ vào số ngày đó để tính giá tiền phòng cho khách, tiền dịch vụ sẽ tính và cộng vào sau đó xuất ra giá tiền tổng cho khách hàng phải chi trả.

## *Mô hình ERD*



## *Cơ sở dữ liệu*

|  |  |  |
| --- | --- | --- |
| STT | Phần Script | Chức năng |
| 1 | IF DB\_ID('HotelDB') IS NOT NULL DROP DATABASE HotelDB;  go  CREATE DATABASE HotelDB;  go | Xóa database khi đã tồn tại nó để tạo một database mới. Nhằm mục đích để sửa database khi cần thiết. |
| 2 | CREATE TABLE RoomTypes  (  RoomTypeID TINYINT IDENTITY NOT NULL,  Name NVARCHAR(100) not null,  Price smallmoney not null,  )  GO | Tạo bảng loại phòng với tên và giá. Mỗi loại phòng có một giá phòng khác nhau. |
| 3 | CREATE TABLE Customers  (  CustomerID INT IDENTITY(1,1) NOT NULL,  CustomerName NVARCHAR(30) not null,  IdentityCard VARCHAR(9) not null,  PhoneNumber NVARCHAR(10) null,  CustomerAddress NVARCHAR(40) null,  )  go | Tạo bảng khách hàng để lưu trữ thông tin khách hàng gồm tên, chứng minh nhân dân, số điện thoại, địa chỉ để liên lạc khi cần thiết. |
| 4 | CREATE TABLE HotelServices  (  ServiceID INT IDENTITY(1,1) NOT NULL,  ServiceName NVARCHAR(20) not null,  Price smallmoney not null,  )  go | Tạo bảng dịch vụ với mã, tên và giá tiền dịch vụ cho khách hàng sử dụng khi cần thiết. |
| 5 | CREATE TABLE EmployeeTypes  (  EmployeeTypeID TINYINT IDENTITY(1,1) NOT null,  EmployeeTypeName NVARCHAR(30) not null,  )  GO | Tạo bảng loại nhân viên gồm tên nhân loại nhân viên, vd như tiếp tân, thu ngân,... |
| 6 | CREATE TABLE Employees  (  EmployeeID int IDENTITY(1000,1) NOT null,  EmployeeName NVARCHAR(30) not null,  EmployeeTypeID TINYINT not null,  PhoneNumber VARCHAR(10) not null,  IdentityCard VARCHAR(9) not null,  [Password] NVARCHAR(20) not null,  )  GO | Tạo bảng nhân viên để lưu trữ thông tin nhân viên gồm mã, tên, mã loại nhân viên, số điện thoại, chứng minh nhân dân và một mật khẩu để đăng nhập vào ứng dụng quản lý. |
| 7 | CREATE TABLE Booking  (  BookingID int IDENTITY(1,1) NOT null,  CustomerID INT UNIQUE NOT NULL,  RoomID int unique NOT NULL,  AppoinmentDate smalldatetime not null,  )  go | Tạo bảng đặt phòng để lưu thông tin lần đặt trước phòng cụ thể gồm mã, mã của khách hàng, mã phòng được đặt, ngày đặt phòng. |
| 8 | CREATE TABLE Invoices  (  InvoiceID INT IDENTITY(1,1) NOT null,  CustomerID INT not null,  RoomID int not null,  NumberOfDay TINYINT NULL,  EmployeeID INT not null,  InvoiceTotal SMALLMONEY not null,  CheckInDate smalldatetime not null,  CheckOutDate SMALLDATETIME null,  HasPaid BIT NOT NULL DEFAULT 0  )  go | Tạo bảng hóa đơn để khách hàng thanh toán giá tiền khi trả phòng. Bảng hóa đơn sẽ được chèn trực tiếp khi khách hàng thuê phòng trực tiếp “tại quầy”.  Đối với khách hàng đặt phòng trước thì thông tin đặt phòng đó sẽ vào bảng đặt phòng (Booking). Sau đó thông tin từ bảng đặt vào sẽ vào hóa đơn khi khách hàng đó nhận phòng và sẽ thanh toán tương tự. |
| 9 | CREATE TABLE Invoices\_Services  (  Invoices\_Services\_ID int IDENTITY(1,1) NOT null,  InvoiceID INT NOT NULL,  ServiceID INT NOT NULL,  [Times] INT NULL DEFAULT(1),  )  go | Tạo bảng hóa đơn dịch vụ gồm mã, mã hóa đơn. Với mỗi hóa đơn khách hàng thì khách hàng có thể thuê nhiều dịch vụ để sử dụng và sẽ được thanh toán cùng với hóa đơn và xuất hóa đơn. |
| 10 | ALTER TABLE dbo.Customers  WITH Check  ADD CONSTRAINT UK\_Customers UNIQUE(IdentityCard);  GO  ALTER TABLE dbo.Employees  WITH Check  ADD CONSTRAINT UK\_Employees UNIQUE(IdentityCard)  GO  ALTER TABLE dbo.Rooms  WITH CHECK  ADD CONSTRAINT FK\_Rooms\_RoomTypes  FOREIGN KEY(RoomTypeID)  REFERENCES dbo.RoomTypes(RoomTypeID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Employees  WITH CHECK  ADD CONSTRAINT FK\_Employees\_EmployeeTypes  FOREIGN KEY(EmployeeTypeID)  REFERENCES dbo.EmployeeTypes(EmployeeTypeID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Booking  WITH CHECK  ADD CONSTRAINT FK\_Booking\_Customers  FOREIGN KEY(CustomerID)  REFERENCES dbo.Customers(CustomerID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Booking  WITH CHECK  ADD CONSTRAINT FK\_Booking\_Rooms  FOREIGN KEY(RoomID)  REFERENCES dbo.Rooms(RoomID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Invoices  WITH CHECK  ADD CONSTRAINT FK\_Invoices\_Customers  FOREIGN KEY(CustomerID)  REFERENCES dbo.Customers(CustomerID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Invoices  WITH CHECK  ADD CONSTRAINT FK\_Invoices\_Room  FOREIGN KEY(RoomID)  REFERENCES dbo.Rooms(RoomID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Invoices\_Services  WITH CHECK  ADD CONSTRAINT FK\_InvoicesService\_Service  FOREIGN KEY(ServiceID)  REFERENCES dbo.HotelServices(ServiceID)  ON DELETE CASCADE;  go  ALTER TABLE dbo.Invoices\_Services  WITH CHECK  ADD CONSTRAINT FK\_InvoicesService\_Invoice  FOREIGN KEY(InvoiceID)  REFERENCES dbo.Invoices(InvoiceID)  ON DELETE CASCADE;  GO  ALTER TABLE dbo.Invoices  WITH CHECK  ADD CONSTRAINT FK\_InvoicesService\_Employee  FOREIGN KEY(EmployeeID)  REFERENCES dbo.Employees(EmployeeID)  ON DELETE CASCADE;  go  ALTER TABLE dbo.Booking  add CONSTRAINT Check\_BookingAppointmentDate CHECK ( AppoinmentDate >= CAST(CAST(GETDATE() AS DATE) AS SMALLDATETIME));  GO  ALTER TABLE dbo.Invoices  ADD CONSTRAINT Check\_InvoicesEndDate CHECK (CheckOutDate >= CheckInDate);  GO  ALTER TABLE dbo.Invoices  ADD CONSTRAINT Check\_InvoicesNumberOfDay CHECK (NumberOfDay >= 0);  GO  ALTER TABLE dbo.Invoices  ADD CONSTRAINT Check\_InvoicesInvoiceTotal CHECK (InvoiceTotal >= 0);  GO | Tạo ràng buộc các bảng lại với nhau. |

# MÔ TẢ CẤU TRÚC CÁC BẢNG

Bảng 2: Mô tả các field trong bảng InvoiceService

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | InvoiceServiceID | Int | Khóa chính của bảng InvoiceService được đặt identity |
| 2 | InvoiceID | Int | Khóa ngoại chỉ đến hóa đơn tương ứng với 1 phòng |
| 3 | ServiceID | Int | Khóa ngoại chỉ đến dịch vụ được phòng đó sử dung |
| 4 | Times | int | Số lần sử dụng dịch vụ đó của phòng đó |

Bảng 3: Mô tả các field trong bảng Booking

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | BookingID | Int | Khóa chính của bảng Booking được đặt identity |
| 2 | CustomerID | Int | Khóa ngoại chỉ đến người đặt phòng |
| 3 | RoomID | Int | Khóa ngoại chỉ đến phòng được đặt |
| 4 | AppointmentDate | Smalldatetime | Ngày dự định đến lấy phòng |

Bảng 4: Mô tả các field trong bảng Employees

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | EmployeeID | INT | Khóa chính của bảng Employee được đặt identity |
| 2 | EmployeeName | Nvarchar(30) | Tên của nhân viên |
| 3 | EmployeeTypeID | Tinyint | Khóa ngoại chỉ loại nhân viên |
| 4 | PhoneNumber | Varchar(10) | Số điện thoại của nhân viên |
| 5 | IdentityCard | Varchar(9) | Số chứng minh nhân dân của nhân viên, trường này có ràng buộc unique |
| 6 | Password | Nvarchar(20) | Mật khẩu đăng nhập của nhân viên |

Bảng 5: Mô tả các field trong bảng RoomType

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | RoomTypeID | TINYINT | Khóa chính cho bảng RoomTypes, được đặt identity |
| 2 | Name | NVARCHAR(100) | Tên của loại phòng |
| 3 | Price | smallmoney | Giá phòng trong 1 ngày |

Bảng 6: Mô tả các field trong bảng Rooms

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | RoomID | int | Khóa chính cho bảng Rooms, được đặt identity |
| 2 | RoomTypeID | tinyint | Khóa ngoại từ bảng RoomTypes, cho biết loại phòng |
| 3 | OnFloor | Int | Lưu trữ thông tin về phòng thuộc lầu nào ( mỗi lầu chỉ có cao nhất 5 phòng ) |
| 4 | Status | Int | Lưu trữ thông tin về trạng thái phòng, tương ứng 0 là phòng trống, 1 là đang được thuê và 2 là phòng đã được đặt |

Bảng 7: Mô tả các field trong bảng Customer

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | CustomerID | int | Khóa chính của bảng customer được đặc identity |
| 2 | CustomerName | NVARCHAR(30) | Tên của khách hàng |
| 3 | IdentityCard | VARCHAR(9) | Số chứng minh nhân dân của khách hàng, trường này được ràng buộc là unique |
| 4 | Phone Number | Nvarchar(10) | Số điện thoại của khách hàng |
| 5 | CustomerAddress | NVARCHAR(40) | Địa chỉ của khách hàng, thuộc tính này không bắt buộc |

Bảng 7: Mô tả các field trong bảng HotelServices

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | ServiceUD | int | Khóa chính cho bảng HotelService được đặt identity |
| 2 | ServiceName | nvarchar(30) | Tên của dịch vụ. |
| 3 | Price | Smallmoney | Giá thành 1 lần sử dụng dịch vụ |

Bảng 8: Mô tả các field trong bảng Invoices

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | InvoiceID | int | Khóa chính cho bảng Invoices và được đặt identity |
| 2 | CustomerID | int | Khóa ngoại trỏ đến khách hàng sử dụng |
| 3 | RoomID | Int | Khóa ngoại trỏ đến phòng mà khách thuê |
| 4 | EmployeeID | int | Khóa ngoại trỏ đến nhân viên đã làm thủ tục nhận phòng cho khách |
| 5 | InvoiceTotal | Smallmoney | Giá tiền khi trả phòng của khách bao gồm cả tiền phòng và tiền dịch vụ |
| 6 | Checkindate | Smalldatetime | Ngày khách nhận phòng |
| 7 | Checkoutdate | Smalldatetime | Ngày khách trả phòng |
| 8 | HasPaid | BIT | Thuộc tính phân loại hóa đơn đã thanh toán hay chưa |

Bảng 9: Mô tả các field trong bảng EmployeeType

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Mục đích |
| 1 | EmployeeTypeID | int | Khóa chính của EmployeeTypeID được đặt identity |
| 2 | EmployeeTypeName | Nvarchar(30) | Tên của loại nhân viên |

# DANH SÁCH CÁC STORED PROCEDURE:

|  |  |  |
| --- | --- | --- |
| STT | Phần Script | Chức năng |
| 1 | if object\_id('sp\_Create') is not null  drop proc sp\_Create  go  create proc sp\_Create  (  @TableName NVARCHAR(30)  )  as  BEGIN  DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharacterLength NVARCHAR(max))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)    DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''sp\_Create'+@TableName+''') is not null  drop proc sp\_Create'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) =  ' Create procedure sp\_Create' + @TableName + '(';  DECLARE @cnt INT = 2  WHILE @cnt <= @count  BEGIN  DECLARE @ColName AS NVARCHAR(20) = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @ColType AS NVARCHAR(20) = (SELECT ColType FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @CharacterLength AS NVARCHAR(20) = (SELECT CharacterLength FROM @InfoColTempTable WHERE ID = @cnt);  IF(@ColType = 'nvarchar' or @ColType = 'varchar')  BEGIN  SET @sql += CHAR(13) +'@'+ @ColName +' '+@ColType+'('+@CharacterLength+') = NULL';  END  ELSE  SET @sql += CHAR(13) +'@'+ @ColName +' '+@ColType +' = NULL';  IF(@cnt < @count)  BEGIN  SET @sql += ',';  END  SET @cnt = @cnt +1;  END  SET @sql += Char(13) + ')' +  char(13) + 'as' + CHAR(13) +'BEGIN'+  char(13) + 'insert ' + @TableName +  char(13) + 'values ('  SET @cnt = 2  WHILE @cnt <= @count  BEGIN  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt );  SET @sql += '@' + @ColName ;  IF(@cnt < @count)  BEGIN  SET @sql += ',';  END  ELSE  BEGIN  SET @sql += ')';  END  SET @cnt = @cnt +1;  END  SET @sql += CHAR(13) +'END';  print @sql  exec sys.sp\_executesql @sql  END  go | 1 stored procedure sử dụng kỹ thuật dynamic sql để tạo các stored procedure chức năng create cho mỗi bảng trong cở sở dữ liệu., với tham số đầu vào là 1 tên bảng, câu sql động sẽ thực hiện việc tạo 1 stored procedure có chức năng create tương ứng cho bảng đó |
| 2 | if object\_id('sp\_Update') is not null  drop proc sp\_Update  go  create proc sp\_Update  (  @TableName NVARCHAR(30)  )  as  BEGIN  DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharacterLength NVARCHAR(max))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''sp\_Update'+@TableName+''') is not null  drop proc sp\_Update'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @sql AS NVARCHAR(max) = 'create procedure sp\_Update' + @TableName + '(';  DECLARE @cnt INT = 1  WHILE @cnt <= @count  BEGIN  DECLARE @ColName AS NVARCHAR(20) = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @ColType AS NVARCHAR(20) = (SELECT ColType FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @CharacterLength AS NVARCHAR(20) = (SELECT CharacterLength FROM @InfoColTempTable WHERE ID = @cnt);  IF(@ColType = 'nvarchar' or @ColType = 'varchar')  BEGIN  SET @sql += CHAR(13) +'@'+ @ColName +' '+@ColType+'('+@CharacterLength+') = NULL';  END  ELSE  SET @sql += CHAR(13) +'@'+ @ColName +' '+@ColType + '= NULL';  IF(@cnt < @count)  BEGIN  SET @sql += ',';  END  SET @cnt = @cnt +1;  END  SET @ColName = (select ColName from @InfoColTempTable where ID = 1);  SET @sql += Char(13) + ')' +  char(13) + 'as' + CHAR(13)+'BEGIN'+  CHAR(13) + 'if not exists (select \* from '+@TableName+' where '+@ColName+' = @'+@ColName+')  throw 50001, ''Invalid RoomTypeID.'' , 1;'+  char(13) + 'update ' + @TableName +  char(13) + 'set'  SET @cnt = 2  WHILE @cnt <= @count  BEGIN  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt );  SET @sql += char(13) + @ColName + '= @' + @ColName ;  IF(@cnt < @count)  BEGIN  SET @sql += ',';  END  ELSE  BEGIN  SET @ColName = (Select ColName from @InfoColTempTable where ID = 1);  SET @sql += char(13) + 'where ' + @ColName+ '= @' + @ColName + CHAR(13)+'END';  END  SET @cnt = @cnt +1;  END  print @sql  exec sys.sp\_executesql @sql  END  GO | 1 store procedure sử dụng kỹ thuật dynamic sql để tạo các stored procedure chức năng update cho mỗi bảng trong cở sở dữ liệu., với tham số đầu vào là 1 tên bảng, câu sql động sẽ thực hiện việc tạo 1 stored procedure có chức năng update tương ứng cho bảng đó |
| 3 | if object\_id('sp\_Delete') is not null  drop proc sp\_Delete  go  create proc sp\_Delete  (  @TableName NVARCHAR(30)  )  as  BEGIN  DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''sp\_Delete'+@TableName+''') is not null  drop proc sp\_Delete'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = 'create procedure sp\_Delete'+@TableName+'(';  DECLARE @ColName AS NVARCHAR(20) = (SELECT top (1) ColName FROM @InfoColTempTable ORDER BY ID ASC);  DECLARE @ColType AS NVARCHAR(20) = (SELECT top (1) ColType FROM @InfoColTempTable ORDER BY ID ASC);  SET @sql += CHAR(13) +'@'+ @ColName +' '+@ColType +  Char(13)+ ')';  SET @ColName = (select ColName from @InfoColTempTable where ID = 1);  SET @sql += char(13) + 'as' + CHAR(13)+'BEGIN'+  CHAR(13) + 'if not exists (select \* from '+@TableName+' where '+@ColName+' = @'+@ColName+')  throw 50001, ''Invalid RoomTypeID.'' , 1;'+  char(13)+ 'delete ' + @TableName +  char(13) +'where '  SET @ColName = (select top (1) ColName from @InfoColTempTable ORDER BY ID ASC);  SET @sql += @ColName+ ' = @' + @ColName+CHAR(13)+ 'END';  print @sql  exec sys.sp\_executesql  @stmt = @sql;  END  GO | 1 stored procedure sử dụng kỹ thuật dynamic sql để tạo các stored procedure chức năng delete cho mỗi bảng trong cở sở dữ liệu., với tham số đầu vào là 1 tên bảng, câu sql động sẽ thực hiện việc tạo 1 stored procedure có chức năng delete tương ứng cho bảng đó |
| 4 | if object\_id ('sp\_Create\_Undo\_Table') is not null  drop proc sp\_Create\_Undo\_Table  GO  create proc sp\_Create\_Undo\_Table  (  @TableName NVARCHAR(100)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''Undo\_'+@TableName+''') is not null  drop table Undo\_'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = 'create table Undo\_' + @TableName + char(13)+  '(' + char(13) +  'iID INT IDENTITY, ';  declare @cnt INT = 1;  while @cnt <= @count  begin  DECLARE @ColName as nvarchar(100) = (select ColName from @InfoColTempTable where ID = @cnt);  DECLARE @ColType as nvarchar(100) = (select ColType from @InfoColTempTable where ID = @cnt);  DECLARE @CharLength as nvarchar(100) = (select CharLength from @InfoColTempTable where ID = @cnt);  SET @sql += char(13) +@ColName + ' ' + @ColType ;  if(@ColType = 'nvarchar' or @ColType = 'varchar')  begin  SET @sql += '('+@CharLength+')';  end  if(@cnt <= @count)  begin  SET @sql += ',';  end  SET @cnt = @cnt +1;  end  SET @sql += char(13) + 'iStatus nvarchar(20)' + char(13) + ')';  print @sql;  exec sys.sp\_executesql @sql  END  GO | Hàm động tạo các bảng Undo cho các bảng, mỗi bảng trong CSDL tương ứng sẽ có 1 bảng Undo, khi các thao tác thêm sửa xóa được thực hiện, sẽ được lưu thao tác trong bảng Undo này. |
| 5 | if object\_id ('sp\_Create\_Redo\_Table') is not null  drop proc sp\_Create\_Redo\_Table  GO  create proc sp\_Create\_Redo\_Table  (  @TableName NVARCHAR(100)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''Redo\_'+@TableName+''') is not null  drop table Redo\_'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = 'create table Redo\_' + @TableName + char(13)+  '(' + char(13) +  'iID INT IDENTITY, ';  declare @cnt INT = 1;  while @cnt <= @count  begin  DECLARE @ColName as nvarchar(100) = (select ColName from @InfoColTempTable where ID = @cnt);  DECLARE @ColType as nvarchar(100) = (select ColType from @InfoColTempTable where ID = @cnt);  DECLARE @CharLength as nvarchar(100) = (select CharLength from @InfoColTempTable where ID = @cnt);  SET @sql += char(13) +@ColName + ' ' + @ColType ;  if(@ColType = 'nvarchar' or @ColType = 'varchar')  begin  SET @sql += '('+@CharLength+')';  end  if(@cnt <= @count)  begin  SET @sql += ',';  end  SET @cnt = @cnt +1;  end  SET @sql += char(13) + 'iStatus nvarchar(20)' + char(13) + ')';  print @sql;  exec sys.sp\_executesql @sql  END  GO | Hàm động tạo các bảng Redo cho các bảng, với mỗi bảng trong CSDL tương ứng sẽ có một bảng Redo. Khi thực hiện thao tác Undo thì thông tin cuối cùng trong bảng Undo sẽ được chèn vào bảng Redo trước khi xóa nhằm để thực hiện việc Redo 1 thao tác trước đó. |
| 6 | if object\_id('sp\_Create\_Trigger\_Delete') is not null  drop proc sp\_Create\_Trigger\_Delete  go  create proc sp\_Create\_Trigger\_Delete  (  @TableName NVARCHAR(100)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id('''+@TableName+'\_DELETE'') is not null  drop trigger '+@TableName+'\_DELETE;';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = 'create trigger ' + @TableName + '\_DELETE on ' + @TableName + char(13)+  'after delete' + char(13)+  'as' + char(13)+  'declare';  declare @cnt INT = 1;  while @cnt <= @count  begin  DECLARE @ColName as nvarchar(100) = (select ColName from @InfoColTempTable where ID = @cnt);  DECLARE @ColType as nvarchar(100) = (select ColType from @InfoColTempTable where ID = @cnt);  DECLARE @CharLength as nvarchar(100) = (select CharLength from @InfoColTempTable where ID = @cnt)  SET @sql += char(13) + '@'+@ColName + ' ' + @ColType;  if(@ColType = 'nvarchar' or @ColType = 'varchar')  begin  SET @sql += '('+@CharLength+')';  end  if(@cnt < @count)  begin  SET @sql += ',';  end  SET @cnt = @cnt +1;  end  SET @cnt = 1;  SET @sql += char(13) + 'select '  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += '@'+@ColName +' = ' +@ColName  if(@cnt < @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @sql += char(13) + 'from deleted'+char(13)+  'insert Undo\_' + @TableName + char(13)+  'values ('  SET @cnt = 1;  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += '@'+@ColName  if(@cnt <= @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @sql += '''deleted'');'  print @sql  exec sys.sp\_executesql @sql  END  go | Hàm động tạo ra các trigger after delete cho các bảng, với mỗi bảng trong CSDL sẽ có một trigger tương ứng. Trigger này có tác dụng lưu thông tin các hàng dữ liệu bị xóa và chèn nó vào bảng Undo tương ứng để có thể tiến hành thao tác Undo cho bảng. |
| 7 | if object\_id('sp\_Create\_Trigger\_Create') is not null  drop proc sp\_Create\_Trigger\_Create  go  create proc sp\_Create\_Trigger\_Create  (  @TableName NVARCHAR(100)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id('''+@TableName+'\_CREATE'') is not null  drop trigger '+@TableName+'\_CREATE;';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = 'create trigger ' + @TableName + '\_CREATE on ' + @TableName + char(13)+  'after insert' + char(13)+  'as' + char(13)+  'declare';  declare @cnt INT = 1;  while @cnt <= @count  begin  DECLARE @ColName as nvarchar(100) = (select ColName from @InfoColTempTable where ID = @cnt);  DECLARE @ColType as nvarchar(100) = (select ColType from @InfoColTempTable where ID = @cnt);  DECLARE @CharLength as nvarchar(100) = (select CharLength from @InfoColTempTable where ID = @cnt)  SET @sql += char(13) + '@'+@ColName + ' ' + @ColType;  if(@ColType = 'nvarchar' or @ColType = 'varchar')  begin  SET @sql += '('+@CharLength+')';  end  if(@cnt < @count)  begin  SET @sql += ',';  end  SET @cnt = @cnt +1;  end  SET @cnt = 1;  SET @sql += char(13) + 'select '  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += '@'+@ColName +' = ' +@ColName  if(@cnt < @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @sql += char(13) + 'from inserted'+char(13)+  'insert Undo\_' + @TableName + char(13)+  'values ('  SET @cnt = 1;  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += '@'+@ColName  if(@cnt <= @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @sql += '''inserted'');'  print @sql  exec sys.sp\_executesql @sql  END  go | Hàm động tạo ra các trigger after insert cho các bảng, với mỗi bảng trong CSDL sẽ có một trigger tương ứng. Trigger này có tác dụng lưu thông tin các hàng dữ liệu được chèn vào bảng và chèn nó vào bảng Undo tương ứng để có thể tiến hành thao tác Undo cho bảng. |
| 8 | if object\_id('sp\_Create\_Trigger\_Update') is not null  drop proc sp\_Create\_Trigger\_Update  go  create proc sp\_Create\_Trigger\_Update  (  @TableName NVARCHAR(100)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id('''+@TableName+'\_UPDATE'') is not null  drop trigger '+@TableName+'\_UPDATE;';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = 'create trigger ' + @TableName + '\_UPDATE on ' + @TableName + char(13)+  'instead of update' + char(13)+  'as' + char(13)+  'declare';  declare @cnt INT = 1;  while @cnt <= @count  begin  DECLARE @ColName as nvarchar(100) = (select ColName from @InfoColTempTable where ID = @cnt);  DECLARE @ColType as nvarchar(100) = (select ColType from @InfoColTempTable where ID = @cnt);  DECLARE @CharLength as nvarchar(100) = (select CharLength from @InfoColTempTable where ID = @cnt)  SET @sql += char(13) + '@'+@ColName + ' ' + @ColType;  if(@ColType = 'nvarchar' or @ColType = 'varchar')  begin  SET @sql += '('+@CharLength+')';  end  if(@cnt < @count)  begin  SET @sql += ',';  end  SET @cnt = @cnt +1;  end  SET @cnt = 1;  SET @sql += char(13) + 'select '  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += '@'+@ColName +' = ' +@ColName  if(@cnt < @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @sql += char(13) + 'from inserted'+char(13)+  'insert Undo\_' + @TableName + char(13)+  'select '  SET @cnt = 1;  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += @ColName  if(@cnt <= @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @ColName = (select ColName from @InfoColTempTable where ID = 1);  SET @sql += '''updated'' from '+@TableName+' where '+ @ColName +' = @' +@ColName+char(13)+  'update '+@TableName+' set ';  SET @cnt = 2;  while @cnt <= @count  begin  SET @ColName = (select ColName from @InfoColTempTable where ID = @cnt);  SET @sql += @ColName +' = @' +@ColName  if(@cnt < @count)  begin  SET @sql += ','  end  SET @cnt = @cnt +1  end  SET @ColName = (select ColName from @InfoColTempTable where ID = 1);  SET @sql += ' where ' + @ColName+ ' = @' + @ColName;  print @sql  exec sys.sp\_executesql @sql  END  go | Hàm động tạo ra các trigger instead of update cho các bảng, với mỗi bảng trong CSDL sẽ có một trigger tương ứng. Trigger này có tác dụng lưu thông tin các hàng dữ liệu trước khi được update và chèn nó vào bảng Undo tương ứng để có thể tiến hành thao tác Undo cho bảng. |
| 9 | use HotelDB;  if object\_id('sp\_Booking') is not null  drop proc sp\_Booking;  go  create proc sp\_Booking  (  @CustomerID INT = NULL,  @RoomID INT = NULL,  @AppoinmentDate SMALLDATETIME = NULL  )  AS  BEGIN  BEGIN TRY  BEGIN TRAN  insert into dbo.Booking(CustomerID, RoomID, AppoinmentDate)  values (@CustomerID, @RoomID, @AppoinmentDate);  UPDATE dbo.Rooms SET Status=2 WHERE RoomID=@RoomID;  COMMIT TRAN  END TRY  BEGIN CATCH  rollback tran;  throw 50001, 'Failed ', 1;  END CATCH  END  GO  use HotelDB;  if object\_id('sp\_Booking') is not null  drop proc sp\_Booking;  go  create proc sp\_Booking  (  @CustomerID INT = NULL,  @RoomID INT = NULL,  @AppoinmentDate SMALLDATETIME = NULL  )  AS  BEGIN  BEGIN TRY  BEGIN TRAN  insert into dbo.Booking(CustomerID, RoomID, AppoinmentDate)  values (@CustomerID, @RoomID, @AppoinmentDate);  UPDATE dbo.Rooms SET Status=2 WHERE RoomID=@RoomID;  COMMIT TRAN  END TRY  BEGIN CATCH  rollback tran;  throw 50001, 'Failed ', 1;  END CATCH  END  GO | Stored procedure dùng cho việc đặt phòng, nhận vào các tham số thích hợp để tạo ra một bản ghi trong bảng Booking, sử dụng transaction để thực hiện các thao tác thêm 1 bản ghi vào Booking và update trạng thái phòng được đặt, nhằm đảm bảo tính nguyên tố. |
| 10 | use HotelDB;  if object\_id('sp\_CheckIn') is not null  drop proc sp\_CheckIn;  go  create proc sp\_CheckIn  (  @CustomerID INT = NULL,  @RoomID INT = NULL,  @EmployeeID INT = NULL  )  AS  BEGIN  BEGIN TRY  BEGIN TRAN  insert into dbo.Invoices (CustomerID, RoomID, EmployeeID, InvoiceTotal, CheckInDate, CheckOutDate,HasPaid)  VALUES (@CustomerID, @RoomID, null, @EmployeeID, CAST(CAST(GETDATE() AS DATE) AS SMALLDATETIME),NULL,0)  UPDATE dbo.Rooms SET Status = 1 WHERE RoomID = @RoomID;    IF( EXISTS(SELECT BookingID FROM dbo.Booking WHERE CustomerID=@CustomerID AND RoomID = RoomID))  BEGIN  DECLARE @ID AS INT = (SELECT BookingID FROM dbo.Booking WHERE CustomerID=@CustomerID AND RoomID = RoomID);  DELETE dbo.Booking WHERE BookingID = @ID;  END  COMMIT TRAN  END TRY  BEGIN CATCH  rollback tran;  throw 50001, 'Failed ', 1;  END CATCH  END  GO | Stored procedure dùng cho việc nhận phòng, nhận vào các tham số thích hợp để tạo ra một bản ghi trong bảng Invoice, sử dụng transaction để thực hiện các thao tác thêm 1 bản ghi vào vào Invoice với các giá trị mặc định và update trạng thái phòng được đặt cũng như nếu tồn tại 1 Booking cho phòng tương ứng thì sẽ xóa Booking đó, nhằm đảm bảo tính nguyên tố. |
| 11 | use HotelDB;  if object\_id('sp\_CheckOut') is not null  drop proc sp\_CheckOut;  go  create proc sp\_CheckOut  (  @RoomID INT = NULL  )  AS  BEGIN  BEGIN TRY  BEGIN TRAN  UPDATE dbo.Invoices SET InvoiceTotal = dbo.fn\_RoomPrice(@RoomID) + dbo.fn\_ServicePrice(@RoomID), CheckOutDate = CAST(CAST(GETDATE() AS DATE) AS SMALLDATETIME), HasPaid = 1 WHERE RoomID = @RoomID AND HasPaid = 0;  UPDATE dbo.Rooms SET Status = 0 WHERE RoomID = @RoomID;  COMMIT TRAN  END TRY  BEGIN CATCH  rollback tran;  throw 50001, 'Failed ', 1;  END CATCH  END  GO | Stored procedure dùng cho việc trả phòng, nhận vào các tham số thích hợp để sửa một bản ghi trong bảng invoices, sử dụng transaction để thực hiện các thao tác sửa bản ghi trong Invoice thành HasPaid và cập nhật lại số tiền tổng cộng của invoice cũng như ngày trả phòng và update trạng thái phòng được đặt, nhằm đảm bảo tính nguyên tố. |
| 12 | use HotelDB;  if object\_id('sp\_Create\_Undo') is not null  drop proc sp\_Create\_Undo;  go  CREATE PROC sp\_Create\_Undo  (  @TableName NVARCHAR(30)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''sp\_Undo\_'+@TableName+''') is not null  drop proc sp\_Undo\_'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = N'create procedure sp\_Undo\_'+@TableName+char(13)+  'as'+char(13)+  'BEGIN'+char(13)+  'SET IDENTITY\_INSERT '+@TableName+ ' ON'+char(13)+  'exec (''DISABLE TRIGGER '+@TableName+'\_Create ON '+@TableName+''')'+ char(13)+  'exec (''DISABLE TRIGGER '+@TableName+'\_Delete ON '+@TableName+''')' + char(13)+  'exec (''DISABLE TRIGGER '+@TableName+'\_Update ON '+@TableName+''')'+ char(13)+      'declare';  DECLARE @cnt INT = 1;  WHILE @cnt <= @count  BEGIN  DECLARE @ColName AS NVARCHAR(20) = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @ColType AS NVARCHAR(20) = (SELECT ColType FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @CharLength AS NVARCHAR(20) = (SELECT CharLength FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += CHAR(13) +' @'+ @ColName +' '+@ColType ;  if(@ColType = 'varchar' or @ColType = 'nvarchar')  begin  SET @sql += '('+@CharLength+')';  end  SET @sql += ', '  SET @cnt = @cnt + 1;  END;  SET @sql += CHAR(13) + ' @iStatus varchar(10);' + CHAR(13);  SET @sql += 'select top 1';  SET @cnt = 1;  WHILE @cnt <= @count  BEGIN  set @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += ' @'+ @ColName +'='+@ColName+',';  SET @cnt = @cnt + 1;  END;  SET @sql +='@iStatus = iStatus from Undo\_'+@TableName+' order by iID desc'+char(13);    SET @ColName = (select ColName from @InfoColTempTable where ID = 1)  SET @sql += 'IF((select top 1 iStatus from Undo\_'+@TableName+' order by iID desc) = ''inserted'')'+char(13)+  'begin'+ char(13)+'begin tran'+char(13)+  ' delete '+ @TableName +char(13)+  ' where '+ @ColName +' = @' + @ColName+ char(13)+  ' insert Redo\_'+ @TableName + char(13)+  ' values ('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName + ', ';  SET @cnt = @cnt + 1;  END  SET @sql += '@iStatus);' +char(13)+  ' with t as (select top 1 \* from Undo\_'+@TableName+' order by iID desc)'+char(13)+  ' delete from t'+char(13)+'commit tran'+char(13)+  'end'+char(13)+  'else if((select top 1 iStatus from Undo\_'+@TableName+' order by iID desc) = ''deleted'')'+char(13)+  'begin'+char(13)+'begin tran'+char(13)+  ' insert '+ @TableName+'('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += @ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql += ')' +char(13)+  ' values ('  SET @cnt =1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql += ')' +char(13)+  ' insert Redo\_'+ @TableName+char(13)+  ' values ('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName + ', ';  SET @cnt = @cnt + 1;  END  SET @sql += '@iStatus);' +char(13)+  ' with t as (select top 1 \* from Undo\_'+@TableName+' order by iID desc )'+char(13)+  ' delete from t;'+char(13)+'commit tran'+char(13)+  'end'+char(13)+  'else if((select top 1 iStatus from Undo\_'+@TableName+' order by iID desc) = ''updated'')'+char(13)+  'begin'+char(13)+'begin tran'+char(13)+  ' insert Redo\_'+ @TableName +char(13)+  ' select '  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += @ColName + ', ';  SET @cnt = @cnt + 1;  END  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = 1);  SET @sql += '''updated'' from '+@TableName+ ' where '+ @ColName + ' = @'+@ColName+';'+char(13)+  ' delete '+ @TableName+ char(13)+  ' where ' + @ColName + ' = @'+@ColName+';'+char(13)+  ' insert '+ @TableName+'(';  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += @ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql+=')'+char(13)+  ' values('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql += ');' +char(13)+  ' with t as (select top 1 \* from Undo\_'+@TableName+' order by iID desc )'+char(13)+  ' delete from t;' + char(13)+'commit tran'+char(13)+  'end'+char(13)+  'exec (''ENABLE TRIGGER '+@TableName+'\_Create ON ' + @TableName+ ''')'+char(13)+  'exec (''ENABLE TRIGGER '+@TableName+'\_Delete ON ' + @TableName+ ''')'+char(13)+  'exec (''ENABLE TRIGGER '+@TableName+'\_Update ON ' + @TableName+ ''')'+char(13)+  'SET IDENTITY\_INSERT '+@TableName+ ' OFF'+char(13)+  'END';  print @sql  exec sys.sp\_executesql @sql  END  go | Store procedure dùng kỹ thuật sql động để tạo ra các store procedure tương ứng với chức năng undo ở mỗi bảng |
| 13 | if object\_id('sp\_Create\_Redo') is not null  drop proc sp\_Create\_Redo;  go  CREATE PROC sp\_Create\_Redo  (  @TableName NVARCHAR(30)  )  AS  BEGIN    DECLARE @InfoColTempTable TABLE (ID INT IDENTITY(1,1) PRIMARY KEY,ColName sysname,ColType NVARCHAR(128), CharLength NVARCHAR(100))  INSERT INTO @InfoColTempTable  SELECT COLUMN\_NAME,DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.columns WHERE TABLE\_NAME= @TableName  DECLARE @count INT = (SELECT COUNT(ID)  FROM @InfoColTempTable)  DECLARE @delsql AS NVARCHAR(max) = ' if object\_id(''sp\_Redo\_'+@TableName+''') is not null  drop proc sp\_Redo\_'+@TableName+';';  PRINT @delsql;  EXEC sp\_executesql  @stmt = @delsql;  DECLARE @sql AS NVARCHAR(max) = N'create procedure sp\_Redo\_'+@TableName+char(13)+  'as'+char(13)+  'BEGIN'+char(13)+  'SET IDENTITY\_INSERT '+@TableName+ ' ON'+char(13)+  'exec (''DISABLE TRIGGER '+@TableName+'\_Create ON '+@TableName+''')'+ char(13)+  'exec (''DISABLE TRIGGER '+@TableName+'\_Delete ON '+@TableName+''')' + char(13)+  'exec (''DISABLE TRIGGER '+@TableName+'\_Update ON '+@TableName+''')'+ char(13)+      'declare';  DECLARE @cnt INT = 1;  WHILE @cnt <= @count  BEGIN  DECLARE @ColName AS NVARCHAR(20) = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @ColType AS NVARCHAR(20) = (SELECT ColType FROM @InfoColTempTable WHERE ID = @cnt);  DECLARE @CharLength AS NVARCHAR(20) = (SELECT CharLength FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += CHAR(13) +' @'+ @ColName +' '+@ColType ;  if(@ColType = 'varchar' or @ColType = 'nvarchar')  begin  SET @sql += '('+@CharLength+')';  end  SET @sql += ', '  SET @cnt = @cnt + 1;  END;  SET @sql += CHAR(13) + ' @iStatus varchar(10);' + CHAR(13);  SET @sql += 'select top 1';  SET @cnt = 1;  WHILE @cnt <= @count  BEGIN  set @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += ' @'+ @ColName +'='+@ColName+',';  SET @cnt = @cnt + 1;  END;  SET @sql +='@iStatus = iStatus from Redo\_'+@TableName+' order by iID desc'+char(13);    SET @ColName = (select ColName from @InfoColTempTable where ID = 1)  SET @sql += 'IF((select top 1 iStatus from Redo\_'+@TableName+' order by iID desc) = ''inserted'')'+char(13)+  'begin'+ char(13)+'Begin tran'+char(13)+  ' insert '+ @TableName+'('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += @ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql += ')' +char(13)+  ' values ('  SET @cnt =1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql += ')' +char(13)+  ' insert Undo\_'+ @TableName + char(13)+  ' values ('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName + ', ';  SET @cnt = @cnt + 1;  END  SET @sql += '@iStatus);' +char(13)+  ' with t as (select top 1 \* from Redo\_'+@TableName+' order by iID desc)'+char(13)+  ' delete from t'+char(13)+'commit tran'+char(13)+  'end'+char(13)+  'else if((select top 1 iStatus from Redo\_'+@TableName+' order by iID desc) = ''deleted'')'+char(13)+  'begin'+char(13)+'Begin tran'+char(13);  SET @ColName = (select ColName from @InfoColTempTable where ID = 1)  SET @sql += ' delete '+ @TableName +char(13)+  ' where '+ @ColName +' = @' + @ColName+ char(13)+  ' insert Undo\_'+ @TableName+char(13)+  ' values ('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName + ', ';  SET @cnt = @cnt + 1;  END  SET @sql += '@iStatus);' +char(13)+  ' with t as (select top 1 \* from Redo\_'+@TableName+' order by iID desc )'+char(13)+  ' delete from t;'+char(13)++'commit tran'+char(13)+  'end'+char(13)+  'else if((select top 1 iStatus from Redo\_'+@TableName+' order by iID desc) = ''updated'')'+char(13)+  'begin'+char(13)+'Begin tran'+char(13)+  ' insert Undo\_'+ @TableName +char(13)+  ' select '  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += @ColName + ', ';  SET @cnt = @cnt + 1;  END  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = 1);  SET @sql += '''updated'' from '+@TableName+ ' where '+ @ColName + ' = @'+@ColName+';'+char(13)+  ' delete '+ @TableName+ char(13)+  ' where ' + @ColName + ' = @'+@ColName+';'+char(13)+  ' insert '+ @TableName+'(';  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += @ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql+=')'+char(13)+  ' values('  SET @cnt = 1  WHILE @cnt <= @count  BEGIN  SET @ColName = (SELECT ColName FROM @InfoColTempTable WHERE ID = @cnt);  SET @sql += '@'+@ColName;  if(@cnt < @count)  begin  SET @sql += ', '  end  SET @cnt = @cnt + 1;  END  SET @sql += ');' +char(13)+  ' with t as (select top 1 \* from Redo\_'+@TableName+' order by iID desc )'+char(13)+  ' delete from t;' + char(13)+'commit tran'+char(13)+  'end'+char(13)+  'exec (''ENABLE TRIGGER '+@TableName+'\_Create ON ' + @TableName+ ''')'+char(13)+  'exec (''ENABLE TRIGGER '+@TableName+'\_Delete ON ' + @TableName+ ''')'+char(13)+  'exec (''ENABLE TRIGGER '+@TableName+'\_Update ON ' + @TableName+ ''')'+char(13)+  'SET IDENTITY\_INSERT '+@TableName+ ' OFF'+char(13)+  'END';  print @sql  exec sys.sp\_executesql @sql  END  GO | Tương tự sp\_Create\_Undo, là 1 store procedure dùng sql động để tạo các store procedure có chức năng redo cho mỗi bảng |
| 14 | if object\_id('Revenue') is not null  drop function Revenue  go  create function Revenue(@month nvarchar(50), @year nvarchar(50))  returns smallmoney  begin  if(@month != 'None')  begin  return(select sum(InvoiceTotal)  from Invoices  where FORMAT(CheckOutDate - 1,'MMM', 'en-US') = @month and  year(cast(CheckOutDate as nvarchar(50))) = @year)  end  return(select sum(InvoiceTotal)  from Invoices  where year(cast(CheckOutDate as nvarchar(50))) = @year)  end  go | Hàm tính tổng doanh thu trong các tháng và năm. |
| 15 | use HotelDB;  if object\_id('sp\_LoginEmployee') is not null  drop proc sp\_LoginEmployee;  go  create proc sp\_LoginEmployee  (  @Account int,  @Password NVARCHAR(20),  @EmployeeType TINYINT  )  as  BEGIN  DECLARE @id AS INT;  DECLARE @sql AS NVARCHAR(200) = 'SELECT @EmpID=EmployeeID FROM dbo.Employees WHERE IdentityCard = @acc AND PassWord = @pass AND EmployeeTypeID = @type';  DECLARE @params AS NVARCHAR(100)= '@acc int,@pass nvarchar(20),@type tinyint, @EmpID int output';  EXEC sys.sp\_executesql @sql, @params , @acc = @Account, @pass = @Password, @type = @EmployeeType, @EmpID = @id output  SELECT @id  END  go | Store procedure dùng để xác thực đăng nhập |
| 16 | use HotelDB;  if object\_id('sp\_LoginEmployee') is not null  drop proc sp\_LoginEmployee;  go  create proc sp\_LoginEmployee  (  @Account int,  @Password NVARCHAR(20),  @EmployeeType TINYINT  )  as  BEGIN  DECLARE @id AS INT;  DECLARE @sql AS NVARCHAR(200) = 'SELECT @EmpID=EmployeeID FROM dbo.Employees WHERE IdentityCard = @acc AND PassWord = @pass AND EmployeeTypeID = @type';  DECLARE @params AS NVARCHAR(100)= '@acc int,@pass nvarchar(20),@type tinyint, @EmpID int output';  EXEC sys.sp\_executesql @sql, @params , @acc = @Account, @pass = @Password, @type = @EmployeeType, @EmpID = @id OUTPUT;  SELECT @id  END  go | Store Proc dùng để xác thực đăng nhập sử dụng kỹ thuật Dynamic SQL |
| 17 | use HotelDB;  if object\_id('sp\_Revenue5Month') is not null  drop Proc sp\_Revenue5Month  go  create Proc sp\_Revenue5Month  AS  begin  DECLARE @month as nvarchar(20) = month(cast(getdate() as nvarchar(50)));  DECLARE @year as int = year(cast(getdate() as nvarchar(50)));  DECLARE @TempTable TABLE (ID INT identity(1,1) PRIMARY KEY, [Month] int, Revenue decimal);  Declare @cnt INT = 0;  If(@month<=4)  begin  while @cnt < @month  begin  INSERT INTO @TempTable  SELECT (@month-@cnt) as Month, SUM(InvoiceTotal) as Total  FROM Invoices WHERE month(cast(CheckOutDate as nvarchar(50))) = @month-@cnt and  year(cast(CheckOutDate as nvarchar(50))) = @year and HasPaid=1;  SET @cnt = @cnt +1;  end  end  else  begin  while @cnt < 5  begin  INSERT INTO @TempTable  SELECT (@month-@cnt) as Month, SUM(InvoiceTotal) as Total  FROM Invoices WHERE month(cast(CheckOutDate as nvarchar(50))) = @month-@cnt and  year(cast(CheckOutDate as nvarchar(50))) = @year and HasPaid=1;  SET @cnt = @cnt +1;  end  end  Select Month,Revenue from @TempTable;  end  go | Store Procedure dùng để xuất ra danh sách doanh thu trong từng tháng của 5 tháng gần nhất |
|  | USE HotelDB;  IF OBJECT\_ID(N'dbo.sp\_SearchCustomer', N'P') IS NOT NULL DROP PROC dbo.sp\_SearchCustomer;  GO  CREATE PROC dbo.sp\_SearchCustomer  (  @CustomerName AS nvarchar(30) = NULL,  @IdentityCard AS nvarchar(10) = NULL  )  AS  BEGIN  if @CustomerName is not null  SET @CustomerName = @CustomerName + '%';  if @IdentityCard is not null  SET @IdentityCard = @IdentityCard + '%'  DECLARE @sql AS NVARCHAR(1000);  SET @sql =  N'SELECT \*'  + N' FROM dbo.Customers where 1=1 '  + CASE WHEN @CustomerName IS NOT NULL THEN  N'AND CustomerName like @cna ' ELSE N'' END  + CASE WHEN @IdentityCard IS NOT NULL THEN  N'AND IdentityCard like @cid ' ELSE N'' END  print @sql;  EXEC sp\_executesql  @stmt = @sql,  @params = N'@cna AS nvarchar(30), @cid AS nvarchar(10)',  @cna = @CustomerName,  @cid = @IdentityCard  END  GO | Store Procedure dùng để thực hiện Dynamic Search ở bảng Customer, proc này được dùng trong các chức năng Booking và Checkin trên form |
|  |  |  |

# DANH SÁCH CÁC FUNCTION

|  |  |  |
| --- | --- | --- |
| STT | Phần Script | Chức năng |
| 1 | use HotelDB;  if object\_id('fn\_RoomPrice') is not null  drop FUNCTION fn\_RoomPrice;  go  create FUNCTION fn\_RoomPrice  (  @RoomID INT = NULL  )  RETURNS Money  BEGIN  DECLARE @startdate AS SMALLDATETIME  SET @startdate = (SELECT CheckInDate FROM dbo.Invoices WHERE RoomID = @RoomID AND HasPaid = 0);  DECLARE @Price AS MONEY = ( SELECT c.Price FROM dbo.Invoices a INNER JOIN dbo.Rooms b ON a.RoomID = b.RoomID INNER JOIN dbo.RoomTypes c ON b.RoomTypeID = c.RoomTypeID WHERE a.RoomID = @RoomID AND a.HasPaid = 0);  DECLARE @pay AS MONEY = DATEDIFF(day, @startdate,getdate()) \* @Price  RETURN @pay;  END  go | Tính tiền dựa theo số ngày ở của 1 bản hợp đồng |
| 2 | use HotelDB;  if object\_id('fn\_ServicePrice') is not null  drop FUNCTION fn\_ServicePrice;  go  create FUNCTION fn\_ServicePrice  (  @RoomID INT = NULL  )  RETURNS Money  BEGIN  DECLARE @ID AS INT = (SELECT InvoiceID FROM dbo.Invoices WHERE RoomID= @RoomID AND HasPaid = 0);  DECLARE @pay AS MONEY =0;  SET @pay += (SELECT Sum(Times\*Price) AS Price FROM dbo.Invoices\_Services INNER JOIN dbo.HotelServices ON HotelServices.ServiceID = Invoices\_Services.ServiceID WHERE InvoiceID = @ID);  RETURN @pay;  END  GO | Tính tiền dựa trên những dịch vụ mà 1 họp đồng đã sử dụng |
| 3 | use HotelDB;  if object\_id('fn\_FindBooking') is not null  drop FUNCTION fn\_FindBooking;  go  create FUNCTION fn\_FindBooking  (  @RoomID INT = NULL  )  RETURNS TABLE  RETURN (SELECT BookingID,Booking.CustomerID,CustomerName,Booking.RoomID FROM ( Booking INNER JOIN dbo.Customers ON Booking.CustomerID = Customers.CustomerID ) INNER JOIN dbo.Rooms ON Booking.RoomID = Rooms.RoomID WHERE Booking.RoomID = 5);  go | Function có chức năng xuất ra 1 bảng tương ứng với Booking của 1 phòng. |
| 4 | if object\_id('fn\_Revenue') is not null  drop function fn\_Revenue  go  create function fn\_Revenue(@month nvarchar(50), @year nvarchar(50))  returns smallmoney  begin  if(@month != 'None')  begin  return(select sum(InvoiceTotal)  from Invoices  where FORMAT(CheckOutDate - 1,'MMM', 'en-US') = @month and  year(cast(CheckOutDate as nvarchar(50))) = @year)  end  return(select sum(InvoiceTotal)  from Invoices  where year(cast(CheckOutDate as nvarchar(50))) = @year)  end  go | Function dùng để tính doanh thu trong khoảng thời gian 1 tháng bất kỳ |

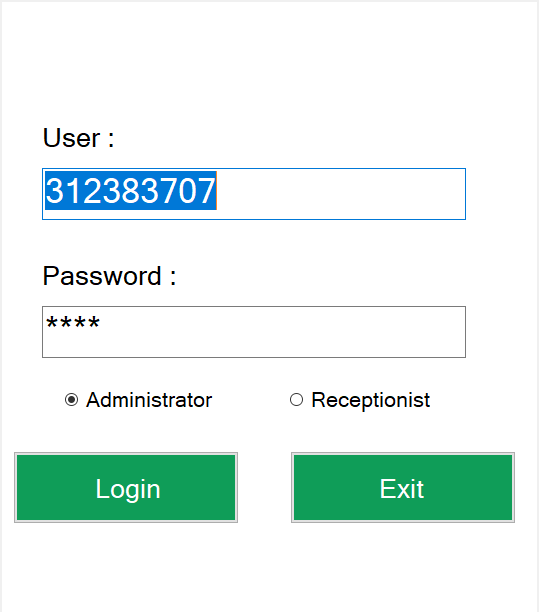
# DANH SÁCH CÁC TRIGGER

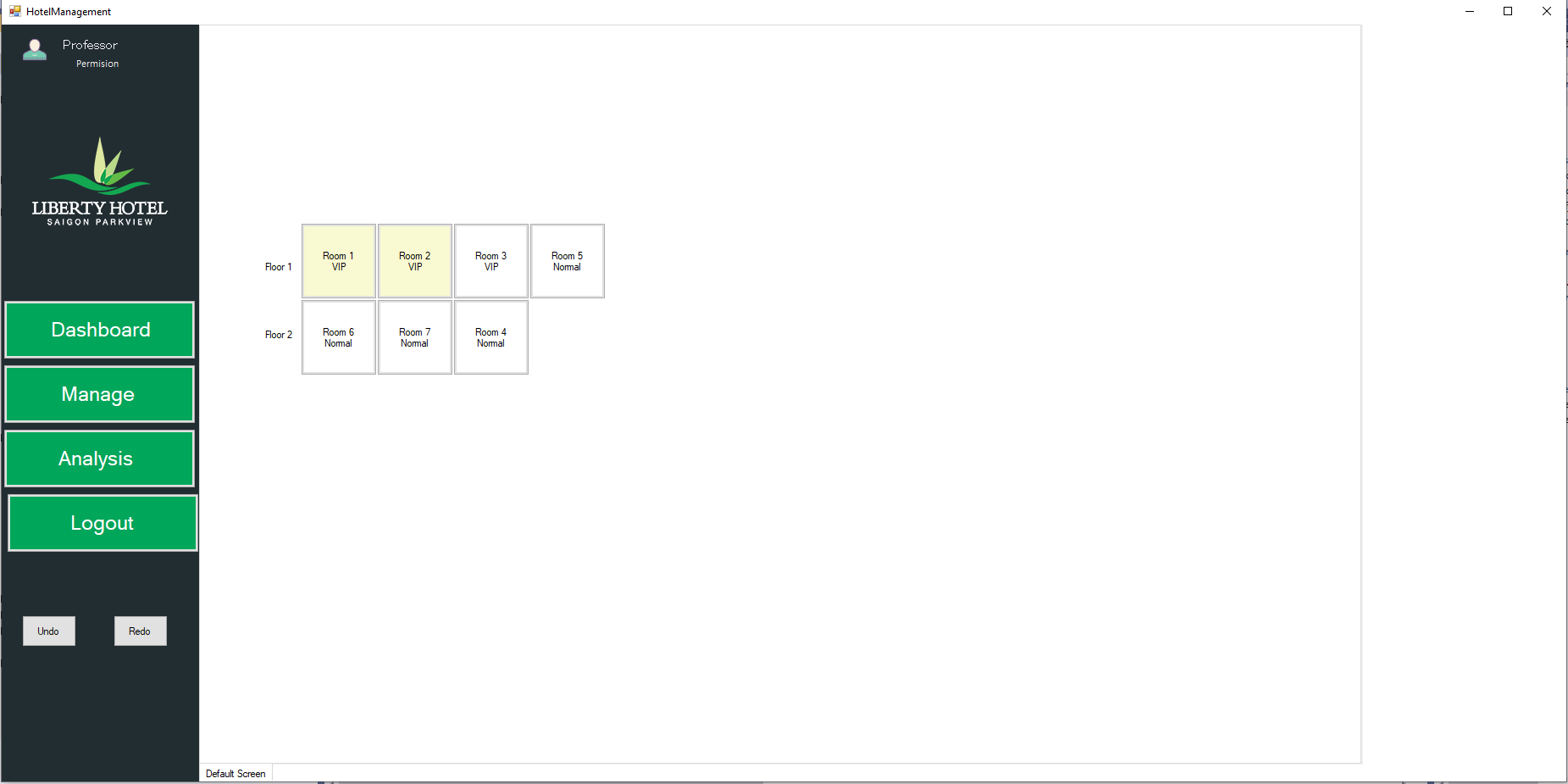
* Do việc tạo các trigger với một số lượng lớn các bảng là việc vô cùng khó khăn. Nên trong đề tài này, việc tạo các trigger cho các bảng sẽ thông qua một stored procedure động như: sp\_Create\_Trigger\_Create, sp\_Create\_Trigger\_Update, sp\_Create\_Trigger\_Delete. Những trigger được tạo ra với việc chạy các stored procedure động trên với biến truyền vào là tên của 1 bảng trong CSDL
* Những trigger này được tạo ra nhằm cho việc undo (trở về) và redo (không trở về) một giai đoạn làm việc trên ứng dụng. Dưới đây là trigger Rooms\_CREATE là ví dụ điển hình.

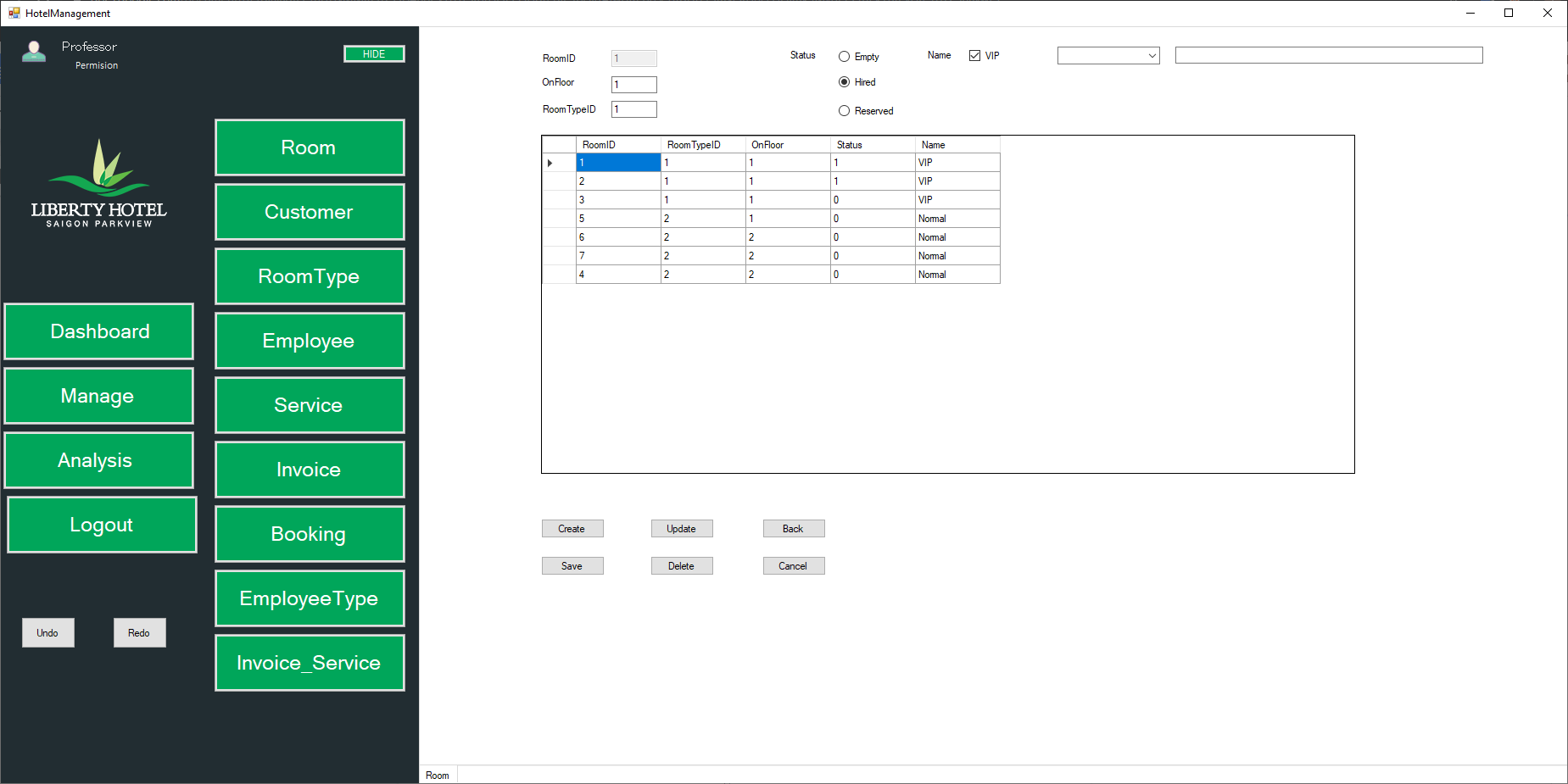
|  |  |  |
| --- | --- | --- |
| STT | Phần Script | Chức năng |
| 1 | if object\_id('Rooms\_CREATE') is not null  drop trigger Rooms\_CREATE;  create trigger Rooms\_CREATE on Rooms  after insert  as  declare  @RoomID int,  @RoomTypeID tinyint,  @OnFloor int,  @Status int  select @RoomID = RoomID,@RoomTypeID = RoomTypeID,@OnFloor = OnFloor,@Status = Status  from inserted  insert Undo\_Rooms  values (@RoomID,@RoomTypeID,@OnFloor,@Status,'inserted'); | Muốn Undo một hàng dữ liệu vào, ta tạo một trigger after insert. Sau khi insert, hệ thống sẽ lấy dữ liệu thêm vào bảng Undo với trạng thái là ‘inserted’ như một Stack cho việc Undo. |
| 2 | if object\_id('Rooms\_DELETE') is not null  drop trigger Rooms\_DELETE;  create trigger Rooms\_DELETE on Rooms  after delete  as  declare  @RoomID int,  @RoomTypeID tinyint,  @OnFloor int,  @Status int  select @RoomID = RoomID,@RoomTypeID = RoomTypeID,@OnFloor = OnFloor,@Status = Status  from deleted  insert Undo\_Rooms  values (@RoomID,@RoomTypeID,@OnFloor,@Status,'deleted'); | Tương tự như trên. Tạo một trigger sau khi xóa hàng dữ liệu của bảng Rooms, hệ thống sẽ lưu lại hàng vừa mới xóa và chèn vào bảng Undo với trạng thái “deleted” như là một Stack cho việc Undo. |
| 3 | if object\_id('Rooms\_UPDATE') is not null  drop trigger Rooms\_UPDATE;  create trigger Rooms\_UPDATE on Rooms  instead of update  as  declare  @RoomID int,  @RoomTypeID tinyint,  @OnFloor int,  @Status int  select @RoomID = RoomID,@RoomTypeID = RoomTypeID,@OnFloor = OnFloor,@Status = Status  from inserted  insert Undo\_Rooms  select RoomID,RoomTypeID,OnFloor,Status,'updated' from Rooms where RoomID = @RoomID  update Rooms set RoomTypeID = @RoomTypeID,OnFloor = @OnFloor,Status = @Status where RoomID = @RoomID | Khác với hai trigger trên, trigger cho update là dạng instead of. Thay vì update thông thường, thì trigger có tác dụng trước khi update, hệ thống sẽ lưu hàng mà người dùng chỉ định vào bảng Undo với trạng thái “updated” và sau đó mới tiến hành update nhằm lưu lại thông tin cũ trước khi update. |

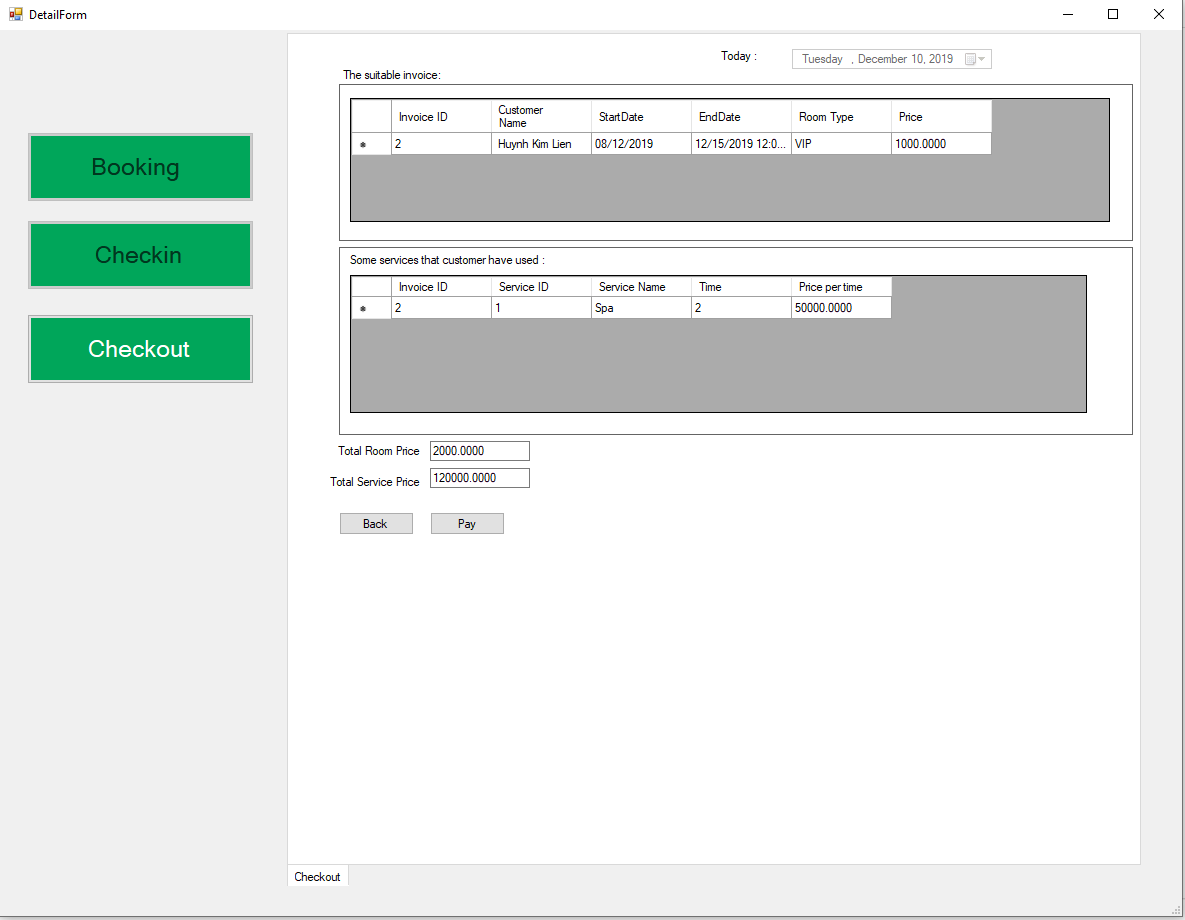
* Những trạng thái nêu trên khi chèn vào các bảng Undo để tiện lợi cho việc sử dụng Stack ứng với từng trường hợp trạng thái khác nhau.
* Các bảng còn lại đều được tạo trigger cho việc Undo và Redo ngoại trừ hai bảng Booking và Invoices vì dữ liệu 2 bảng này không thể được tạo một cách vô tội vạ mà phải được tạo ra khi có khách hàng đến thuê hoặc đặt phòng và tức nhiên các thao tác tạo xóa sửa trên 2 bảng này là không thể undo, redo xét về mặt ngữ nghĩa.

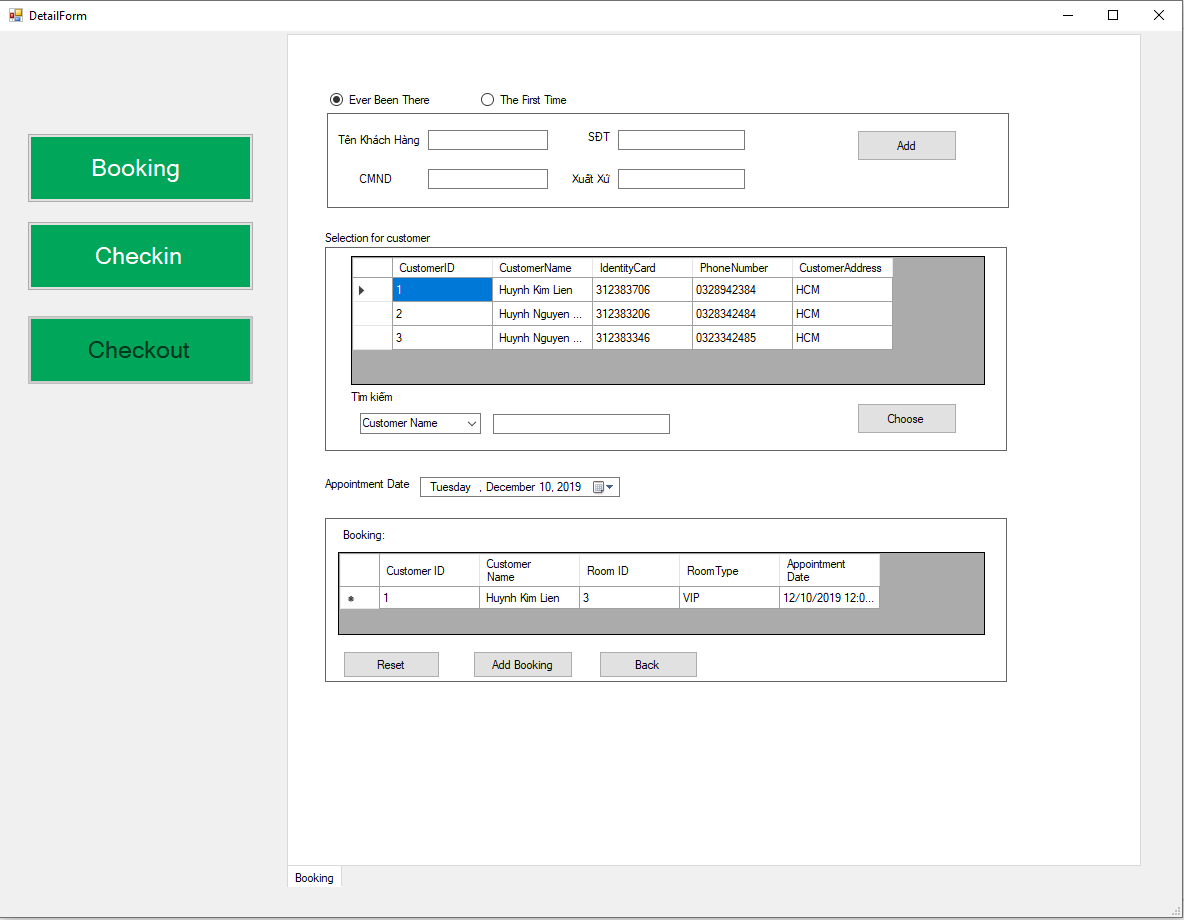
# GIAO DIỆN

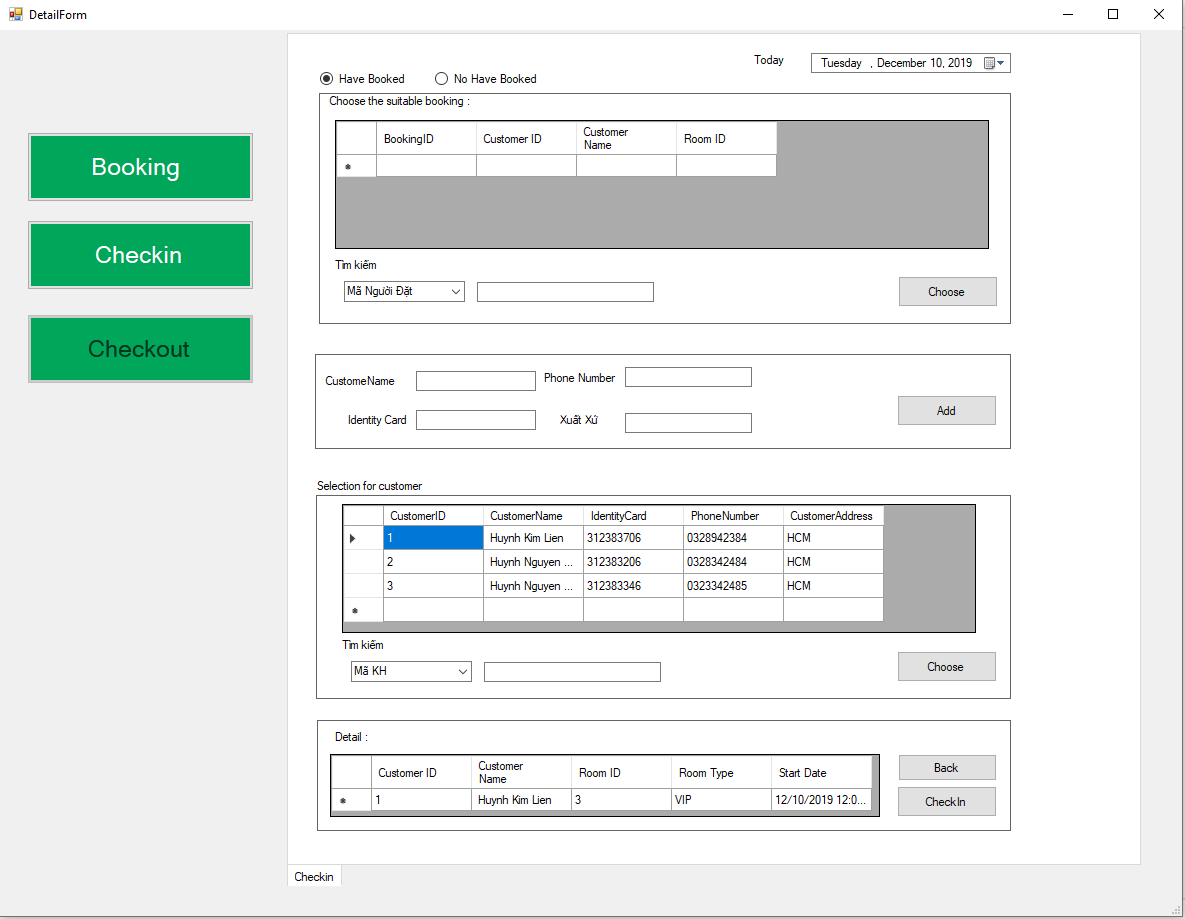


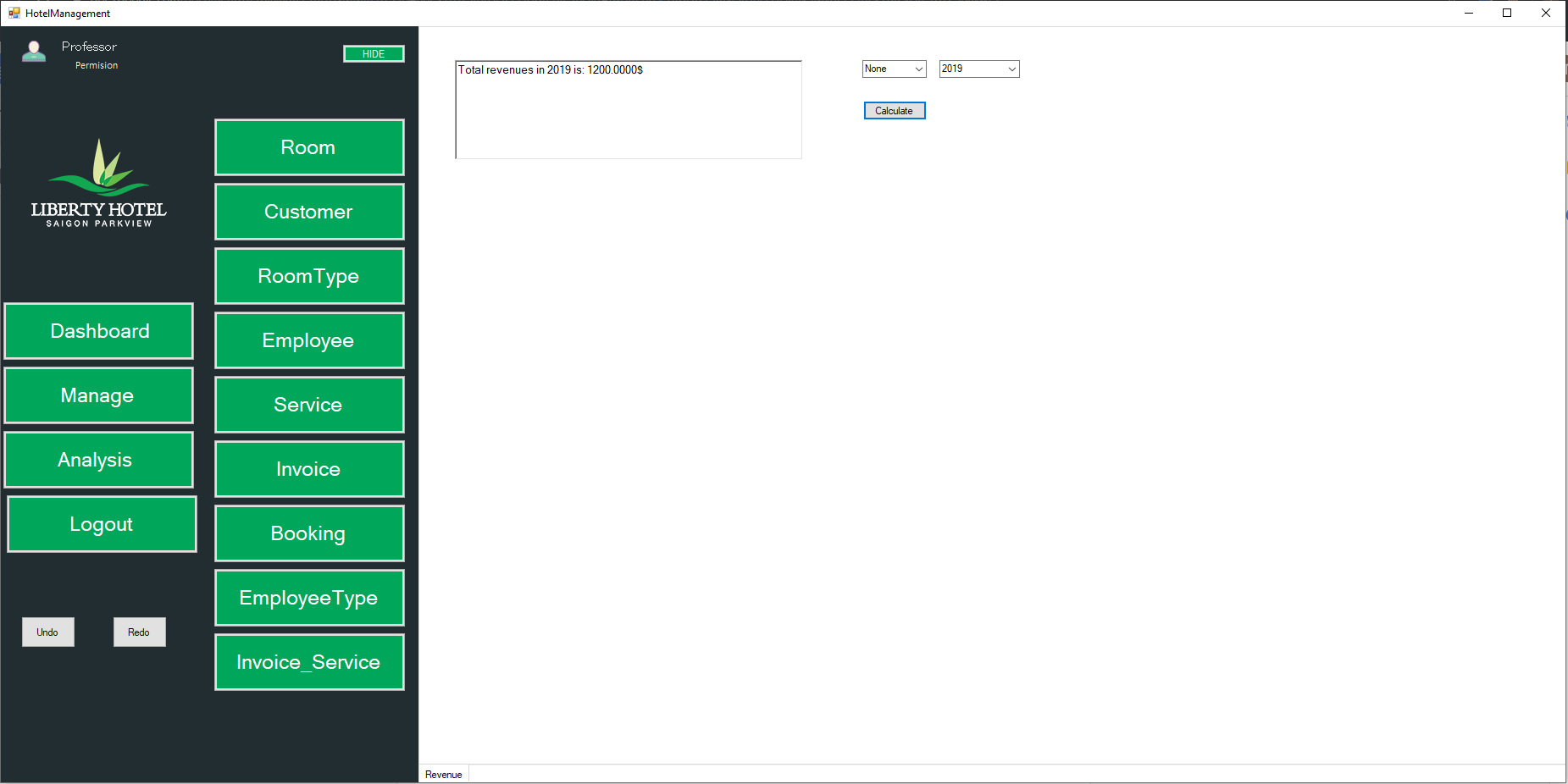












# TỔNG KẾT

# *Nhận xét*

* Nhóm đã hoàn thành yêu cầu khoảng hơn 80% mục tiêu đặt ra
* Trong quá trình làm bài tập nhóm đã học hỏi được nhiều thứ, một trong đó là cách sử dụng Dynamic SQL để giảm thiểu khối lượng công việc lặp lại của việc viết code, đồng thời qua bài tập nhóm cũng hiểu được cách sử dụng index để tối ưu hóa tốc độ, cũng như các thủ thuật, vận dụng kiến thức ở trên lớp vào thực tiễn ứng dụng.
* Có dùng kỹ thuật Dynamic SQL để sinh ra các store procedure cần thiết cho mỗi bảng trong cơ sở dữ liệu( như các store procedure create và update, delete ở các bảng cũng như undo redo, trigger ở các bảng có undo redo), việc này giúp giảm bớt việc phải viết lại những hàm na ná nhau, giúp cho việc bảo trì, cải thiện code trở nên thuận tiện
* Ứng dụng thực hiện chức năng undo redo, cũng như có các store procedure và function phù hợp cho các chức năng.
* Có sử dụng dynamic sql ( cách truyền tham số như chuẩn trong quá trình học )để tránh sql injection khi đăng nhập, cũng như dùng để thực hiện dynamic search.
* Nhóm có đánh index để cải thiện hiệu suất của ứng dụng.
* Giao diện người dùng tương đối đơn giản và thân thiện.
* Sử dụng transaction trong các khối lệnh có tính chất nguyên tố trong nhiều hàm để đảm bảo tính toàn vẹn.
* Có thuận lợi thì cũng có khó khăn, một trong những khó khăn lớn nhất của nhóm là khả năng làm việc nhóm chưa cao, dẫn đến tiến độ chậm, còn nhiều điều trong hệ quản trị cơ sỡ dữ liệu mà nhóm cảm thấy kiến thức còn thiếu và phải nghiên cứu thêm nhiều nữa, cũng như cách áp dụng các đối tượng Index và View của nhóm chưa được tối ưu.

# *Khuyết điểm*

* Sử dụng đến đối tượng View tương đối ít, chưa biết trường hợp nào đúng đắn nhất để dùng View hạn chế thông tin cần ẩn với người dùng.
* Vì ứng dụng chỉ là 1 ứng dụng window với cơ sỡ dữ liệu cục bộ ( chứ không phải mô hình chuyên nghiệp nhiều người dùng cùng truy xuất dữ liệu trên 1 server )nên việc sử dụng transaction chưa được tận dụng hiệu quả như công dụng đáng lý ra của nó.
* Chức năng dynamic search mới chỉ được áp dung cho đối tượng customer để thuận tiện cho các hoạt động Booking và Checkin, chứ chưa được viết trong phần quản trị tác động đến các bảng trong cơ sỡ dữ liệu
* Cơ sở dữ liệu còn đơn giản, chưa chi tiết để áp dụng vào thực tế, mô hình ứng dụng còn là ứng dụng trên window, chưa phải là một ứng dụng online.
* Chức năng undo redo chưa hoàn hảo, vì để hoàn hảo thì khi xóa 1 bản ghi trong 1 bảng được phụ thuộc bởi 1 bảng khác (B phụ thuộc vào A), thì với option cascade delete khi tạo các ràng buộc khóa ngoại sẽ dẫn đến xóa ở bảng phụ thuộc ( khi xóa 1 bản ghi ở A, thì ở B bản ghi tương ứng có khóa ngoại trùng với bản ghi bị xóa ở A cũng phải bị xóa), vì theo đuổi việc thực hiện theo cách dùng dynamic sql để tạo khuôn mẫu cho chức năng undo redo ở các bảng, nên nhóm đã chức năng undo redo khi xóa 1 bảng gốc, thì dữ liệu của bảng phụ thuộc cũng sẽ bị xõa theo, nhưng khi undo redo sẽ không quay về dữ liệ ở các bảng phụ thuộc đó.
* Không có cơ chế mã hóa hay bảo mật, phân quyền người dùng cũng đơn giản ( chỉ mới phân chia 2 dạng người dùng chính là tiếp tân và administrator )

# *Hướng phát triển*

* Thực hiện thêm các điều kiện kiểm tra ở các trigger và store proc phụ trách chức năng undo redo ở mỗi bảng, tùy trường hợp mà phải thực hiện ở bảng phụ thuộc trước, rồi mới tới bảng chính.
* Xây dựng ứng dụng thành 1 ứng dụng web phù hợp để thuận tiện cho người dùng trong việc quản lý và phát triển kinh doanh ( mô hình chuyên nghiệp nhiều người dùng 1 lúc)
* Mở rộng cỡ sỡ dữ liệu chi tiết và phức tạp hơn, gần với thực tiễn hơn nhiều nữa.
* Sử dụng View để hạn chế những thông cần cần thiết trước người dùng.
* Tối ưu các câu truy vấn đến mức kỹ càng.