**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**🕯✡🕮🕮✡🕯**



**TIỂU LUẬN MÔN HỌC**

**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

***Đề tài:***

**QUẢN LÝ ĐẶT PHÒNG KHÁCH SẠN**

**GVHD: TS. Nguyễn Thành Sơn**

**Mã LHP:** **DBMS330284\_22\_2\_03CLC**

**Nhóm thực hiện: Nhóm 02**

**Học kỳ: II**

**Năm học: 2022-2023**

**Thành phố Hồ Chí Minh**

**Tháng 3 năm 2023**

## **DANH SÁCH SINH VIÊN THAM GIA**

**Mã học phần:** *DBMS330284\_22\_2\_03CLC*

**Nhóm: 02**

***Tên đề tài:*** *Quản lí đặt phòng khách sạn*

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **HỌ VÀ TÊN SINH VIÊN** | **MÃ SỐ SINH VIÊN** | **TỈ LỆ % HOÀN THÀNH** |
| 1 | Nguyễn Phú Thành | 21110299 | 100% |
| 2 | Nguyễn Hà Quỳnh Giao | 21110171 | 100% |
| 3 | Nguyễn Văn Hào | 21110175 | 100% |
| 4 | Võ Chí Khương | 21110221 | 100% |

***Ghi chú:***

Tỷ lệ %: Mức độ phần trăm hoàn thành của từng sinh viên tham gia.

***Trưởng nhóm:*** Nguyễn Phú Thành

***Nhận xét của giáo viên***

# ...…………………………………………………………………………………… …………………………………………………………………………………….... ……………………………………………………………………………………… ………………………………………………………………………………………

*Tp. Hồ Chí Minh - Tháng 2 năm 2023*

***MỤC LỤC***

[PHẦN MỞ ĐẦU 1](#_Toc128076641)

[**1. Lời mở đầu** 1](#_Toc128076642)

[**2. Mục đích, nhiệm vụ của đề tài** 1](#_Toc128076643)

[**4. Phạm vi thực hiện đề tài** 1](#_Toc128076644)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG 3](#_Toc128076645)

[**1.1 Đặc tả đề tài** 3](#_Toc128076646)

[**1.2 Nghiệp vụ bài toán** 3](#_Toc128076647)

[**1.3 Mô tả chức năng bài toán** 11](#_Toc128076648)

[KẾT LUẬN 78](#_Toc128076649)

# **PHẦN MỞ ĐẦU**

## **1. Lời mở đầu**

Hệ quản trị Cơ sở dữ liệu là môn học được đào tạo hầu hết các tại trường đại học cho các sinh viên theo học các chuyên ngành công nghệ thông tin đặc biệt là đối với chuyên ngành hệ thống thông tin và công nghệ phần mềm. Môn học trang bị cho sinh viên những kiến thức nền tảng về các giai đoạn phân tích thiết kế trong một dự án công nghệ phần mềm cũng như là cách thức thực hiện và triển khai một dự án cụ thể như thế nào,… Từ đó giúp sinh viên có cách nhìn rộng mở hơn về vai trò cực kì quan trọng của việc phân tích thiết kế khi phát triển phần mềm. Đến với môn học này, được sự cho phép của thầy trực tiếp giảng dạy – thầy Nguyễn Thành Sơn, nhóm chúng em lựa chọn đề chủ đề “Quản lý đặt phòng khách sạn” để làm đề tài thực hiện cho bài tiểu luận môn học của mình.

## **2. Mục đích, nhiệm vụ của đề tài**

Quản lý khách sạn là một khối nghiệp vụ hết sức phức tạp, đòi hỏi một hệ thống quản lý chặt chẽ, thuận tiện và có hiệu quả cao. Đề tài vừa có giá trị trong thực tế vừa có giá trị trong việc học tập của những sinh viên như chúng em. Vì sau khi thực hiện đề tài, sinh viên vừa nắm được vững hơn kiến thức môn học vừa bồi dưỡng cho bản thân thêm kiến thức về nghiệp vụ quản lý trong các doanh nghiệp, công ty góp phần nâng cao trình độ chuyên môn của mỗi cá nhân. Trong khuôn khổ bài tập lớn môn học này, mục tiêu chính yếu là để rèn luyện chúng em thực hiện hệ thống với các nhiệm vụ cơ bản: thực hiện việc đặt phòng, làm thủ tục nhận phòng cũng như trả phòng cho khách một cách nhanh chóng. Cung cấp các dịch vụ khi khách có yêu cầu, đem lại sự thoải mái hài lòng cho khách hàng. Bên cạnh đó là sự hổ trợ cho việc quản lý các danh mục trong hệ thống và thống kê báo cáo.

## **4. Phạm vi thực hiện đề tài**

Đề tài tập chủ yếu tập trung vào các nghiệp vụ chính của một khách sạn chứ không đi sâu vào chi tiết hay thực hiện nhiều nghiệp vụ quá phức tạp do vấn đề về thời gian 2 cũng như là kinh nghiệm vẫn còn ít. Tuy nhiên, với những nghiệp vụ trong đề tài đã đủ để triển khai ra một dự án thực tế dành cho các khách sạn, doanh nghiệp vừa và nhỏ

# **CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG**

## ***1.1 Đặc tả đề tài***

Một khách sạn cần xây dựng một chương trình quản lý đặt phòng khách sạn một cách có hệ thống và thuận tiện với người sử dụng.

Khách sạn sẽ có nhiều nhân viên lễ tân. Mỗi nhân viên sẽ có các thông tin: Mã nhân viên, tên, giới tính, ngày sinh, địa chỉ, số điện thoại, gmail, CCCD. Mỗi nhân viên sẽ được cấp một tài khoản để đăng nhập vào phần mềm, tài khoản sẽ bao gồm: Tên đăng nhập và mật khẩu.

Khách hàng có nhu cầu ở khách sạn sẽ gọi điện hoặc tới trực tiếp để đặt phòng. Nhân viên sẽ sử dụng phần mềm để ghi nhận thông tin đặt phòng bao gồm: Ngày nhận phòng, số ngày ở dự kiến, thời gian tới check-in (quy định là 14h), tiền cọc, tình trạng cọc, thông tin khách hàng, thông tin phòng.

Thông tin khách hàng bao gồm: Mã khách hàng, tên khách hàng, giới tính, địa chỉ, số điện thoại, CCCD. **Một khách hàng có thể đặt được nhiều phòng.** Sau khi khách hàng đã nhận phòng thành công thì trở thành khách hàng chính thức

Một phòng sẽ bao gồm: Mã phòng, tình trạng phòng. Khách sạn bao gồm 8 loại phòng (Cụ thể dưới phần nghiệp vụ). Mỗi loại phòng sẽ có mã loại phòng, tên loại phòng, đơn giá, giảm giá phòng (nếu có).

Trong khi liên hệ đặt phòng khách sạn hoặc quá trình ở tại khách sạn, mỗi phòng muốn có các yêu cầu về dịch vụ đi kèm sẽ liên hệ nhân viên lễ tân. Dịch vụ sẽ bao gồm: Mã dịch vụ, tên dịch vụ, đơn giá, giảm giá(nếu có). **Mỗi phòng có thể sử dụng nhiều dịch vụ.** Mỗi phiếu sử dụng dịch vụ sẽ bao gồm Số lượng dịch vụ sử dụng, ngày sử dụng.

Khi khách hàng trả phòng, nhân viên sẽ xuất hóa đơn bao gồm: Mã hóa đơn, tổng tiền, ngày lập hóa đơn, mã khách hàng, phương thức thanh toán. Nhân viên sẽ tính tổng tiền từ các phiếu chi tiết hóa đơn của hóa đơn đó bằng hệ thống. **Một hóa đơn có thể có nhiều phiếu chi tiết hóa đơn.**

Mỗi phiếu chi tiết hóa đơn sẽ bao gồm: Mã chi tiết hóa đơn, mã hóa đơn, tổng số ngày ở. Một phiếu chi tiết hóa đơn là của 1 phòng.

## ***1.2 Nghiệp vụ bài toán***

### ***1.2.1. Quy trình đặt phòng khách sạn***

Khách hàng cung cấp thông tin đặt phòng (Ngày check-in, ngày check-out, họ và tên, số điện thoại, CCCD hoặc hộ chiếu, email, số phòng đặt, loại phòng muốn đặt, số lượng người ở và một số yêu cầu khác). Hệ thống sẽ tiến hành tính giá trị đơn đặt phòng của khách hàng và xác nhận giá tiền phòng với khách.

**Kiểm tra khả năng đáp ứng của khách sạn:** Số lượng phòng trống mà khách yêu cầu có đáp ứng đủ không?

**Nhập thông tin vào hệ thống:**

Nhân viên lễ tân nhập thông tin đặt phòng vào hệ thống đặt phòng trên máy tính theo mẫu

[Online]:

**Thanh toán:** qua thẻ ngân hàng thông qua internet banking, thanh toán bằng thẻ tín dụng, thanh toán bằng các ví điện tử hoặc tại các điểm thu tiền, ...Với sự đa dạng về hình thức thanh toán giúp cho khách hàng cũng thuận tiện hơn. có thể bổ sung thêm một số khuyến mãi cho người dùng. Sau khi xác nhận thanh toán thành công, khách sạn sẽ gửi biên lai cho khách hàng qua email.

**Khách hàng xác nhận và lưu thông tin đặt phòng:**

**[Offline]:**

Sau khi nhận đặt phòng nhân viên xác nhận lại thông tin đặt phòng của khách theo quy định của khách sạn để tránh sai sót, sau đó mọi thông tin của khách, thông tin về đặt phòng sẽ được lưu lại

**[Online]:**

**Xác nhận đặt phòng:** Sau khi bạn đã hoàn tất thủ tục thanh toán trong **quá trình đặt phòng khách sạn online**, hệ thống sẽ gửi lại cho bạn thông báo xác nhận đã đặt phòng: biên lai xác nhận đặt phòng là một trong những thông tin cần thiết cho quá trình check-in nhận phòng tại khách sạn.

Kiểm tra biên lai xác nhận:  Ngày check-in, ngày check-out, giá phòng đã thỏa thuận, số lượng phòng, thông tin người đặt phòng,... Khi mọi thứ đã chính xác, nghĩa là việc **quy trình đặt phòng khách sạn online** của bạn đã hoàn tất.

**Khách hàng trả tiền cọc đặt phòng:**

**[Offline]:**

Trường hợp khách hàng tới đặt phòng và check-in liền thì không cần đặt cọc

**[Online]:**

Sau khi đặt phòng thành công, khách hàng sẽ chuyển tiền cọc (30% tiền thuê phòng) cho khách sạn. Tiền cọc sẽ được trừ vào khi khách hàng nhận hóa đơn. Nếu như sau khi đặt phòng thành công mà quá 30 phút chưa chuyển tiền cọc thì sẽ hủy phiếu đặt phòng.

**Một số lỗi thường gặp khi đặt phòng online:** Đặt sai ngày dẫn đến khi đến khách sạn vào ngày mong muốn sẽ gặp trường hợp full phòng, sai số lượng phòng không đủ sử dụng, sai vị trí khách sạn sai loại phòng, sai khách sạn [có thể do trùng tên].

**Nếu khách hàng phát hiện thông tin đặt phòng sai lệch.** Khách hàng khi nhận thấy sự cố này nên phản hồi trực tiếp với lễ tân khách sạn để kịp thời thay đổi thông tin đặt phòng.

=> Thông tin đặt hàng sẽ được lưu giữ và cập nhật đầy đủ trong hệ thống máy tính của khách sạn

**Quy định trong trường hợp khách hủy phòng (tùy thuộc vào mỗi khách sạn):**

Điều kiện hủy phòng:

- Quý khách được hủy đặt phòng miễn phí nếu thông báo cho khách sạn trước 5 ngày hoặc trong trường hợp bất khả kháng như thiên tai, bệnh dịch, tai nạn.

**- Mùa cao điểm**

+ Hủy trước 10 ngày (Trừ Thứ 7, Chủ nhật, Lễ Tết) so với ngày nhận phòng: Miễn phí

+ Hủy trong vòng 10 ngày (Trừ Thứ 7, Chủ nhật, Lễ Tết) so với ngày nhận phòng hoặc không đến: Tính 100% tổng tiền phòng

**- Mùa thấp điểm**

+ Hủy trước 06 ngày (Trừ Thứ 7, Chủ nhật, Lễ Tết) so với ngày nhận phòng: Không tính phí

+ Hủy trong vòng 06 ngày (Trừ Thứ 7, Chủ nhật, Lễ Tết) so với ngày nhận phòng hoặc không đến: Tính 100% tổng tiền phòng

* Khi hủy phòng thành công và trong thời gian quy đình, nhân viên sẽ hoàn lại tiền cho khách hàng từ thông tin ngân hàng mà khách hàng cung cấp hoặc báo khách hàng có thể tới lấy trực tiếp nếu trường hợp không thể chuyển qua ngân hàng

**Liên lạc với khách để khẳng định lại thông tin**

Trước một khoảng thời gian nhất định trước ngày nhận phòng của khách (thông thường là 15 ngày đối với khách theo đoàn và 3 ngày đối với khách lẻ), khách sạn liên hệ với khách hàng để xác nhận lại thông tin về việc đặt phòng của khách hàng để xác nhận các trường hợp hủy bỏ hoặc thay đổi thông tin.

**Hằng ngày nhân viên phải cập nhật và tổng hợp tình hình giao dịch trong ngày để khách sạn nắm được tình *hình khách, chuẩn bị đầy đủ hồ sơ thủ tục cũng như phục vụ khách một các tốt nhất.***

***1.2.2. Quy trình check-in nhận phòng***

***Thời gian check-in:*** *2.pm, thời gian lưu trú tính 1 ngày từ 2.pm đến 12.pm hôm sau*

***Xác định khách hàng***

*Sẽ có hai dạng khách hàng:*

* *Chưa đặt trước: Giới thiệu các loại phòng còn trống cho khách, nếu khách hàng đồng ý thì làm thủ tục check-in.*
* *Đã đặt trước: tiến hành kiểm tra và đối chiếu các thông tin của khách với thông tin trong hệ thống. Nếu thông tin trùng khớp, thì nhắc lại thông tin về ngày nhận phòng, các thông tin dịch vụ yêu cầu trước đó và các thông tin liên quan để tránh nhầm lẫn rồi tiến hành làm thủ tục check-in cho khách. Trong trường hợp, Nếu thông tin của khách không tồn tại thì nên mời khách ngồi chờ để kiểm tra lại thông tin xem lỗi do phía khách sạn hay phía khách hàng.*

***Yêu cầu xuất trình giấy tờ:*** *Hộ chiếu/CMND (bắt buộc) hoặc các giấy tờ tùy thân khác. Khách hàng sẽ nhận lại giấy tờ khi check out và thanh toán tiền phòng.*

***Thông báo cho các bộ phận có liên quan chuẩn bị phòng và chào đón khách***

***Kiểm tra thông tin của khách hàng trong hệ thống và xác nhận lại thông tin lần nữa với khách hàng:*** *Loại phòng, giá phòng, số phòng, hình thức thanh toán…*

***Đề nghị khách hàng điền thông tin và ký vào phiếu xác nhận.***

***Giao chìa khóa cho khách hàng và thông báo về quy định nếu làm mất chìa khóa phòng.***

***Một số tình huống xảy ra trong quá trình check-in:***

* *Đối với trường hợp nhận phòng sớm:*

*Check-in từ 5h00 – 9h00: Tính thêm 50% giá phóng.*

*Check-in từ 9h – 14h00: Tính thêm 30% giá phòng.*

* *Trường hợp trả phòng trễ*

*Từ 12h – 15h: Phụ thu 30% giá phòng.*

*Từ 15h – 18h: Phụ thụ 50% giá phòng.*

*Sau 18h00: Phụ thu 100% giá phòng.*

* *Trong trường hợp khách không đến:*

*[Online]: giữ phòng đến 12 pm ngày hôm sau (giờ check-out)*

*Liên hệ với khách hàng thông báo về việc khách đặt phòng nhưng không đến*

*Sau khi hoàn tất thỏa thuận mức phí như quy định (100%), lễ tân thực hiện chuyển đổi tình trạng buồng sang (phòng trống sạch) – bán phòng*

*[Offline]:*

*Một số trường hợp khách hàng đăng ký thông tin không thực, chuyển trạng thái phòng sang phòng trống (do không có cọc đảm bảo).*

***1.2.3. Thông tin về các loại phòng***

Khi khách hàng muốn đặt phòng, nhân viên lễ tân sẽ tìm kiếm phòng còn trống đáp ứng theo ý muốn của khách hàng(về giá cả hoặc loại phòng). Hiện tại khách sạn bao gồm các loại phòng:

* Phòng Standard (STD) – số lượng: 3 phòng
* Phòng Standard là phòng tiêu chuẩn trong khách sạn, loại phòng đơn giản nhất với những thiết bị tối thiểu, có diện tích nhỏ, ở tầng thấp, không có view hoặc view không đẹp. Vì vậy nên có mức giá thấp nhất trong khách sạn.
* Mức giá: 375.000 VND/đêm
* Phòng Superior (SUP) – số lượng: 3 phòng
* Phòng Superior đây là loại phòng chất lượng cao hơn STD với diện tích lớn hơn, được trang bị nhiều thiết bị tiện nghi và có view đẹp. Vì chất lượng tốt hơn nên mức giá thuê phòng SUP cũng sẽ cao hơn STD
* Mức giá: 525.000 VND/đêm
* Phòng Deluxe (DLX) – số lượng: 3 phòng
* Phòng Deluxe loại phòng này thường ở trên cao với chất lượng tốt hơn phòng superior. Phòng Deluxe có diện tích rộng, tầm nhìn đẹp với các trang thiết bị cao cấp và vì thế mức giá ở phòng này cũng cao hơn DLX và SUP.
* Mức giá: 885.000 VND/đêm
* Phòng Suite (SUT) – số lượng: 3 phòng
* Phòng Suite là loại phòng cao cấp nhất trong khách sạn. Phòng Suite thường ở trên tầng cao, được trang thiết bị cao cấp và các dịch vụ đặc biệt kèm theo. Khách sạn sẽ thiết kế phòng Suite có phòng khách và phòng ngủ riêng biệt, có ban công với view đẹp nhất khách sạn. Còn ở những khách sạn cao cấp hơn và tăng độ VIP để cho thuê với mức giá cao hơn, các khách sạn thường đặt tên cho loại phòng Suite là : phòng tổng thống, phòng hoàng gia…
* Mức giá: 1.999.000 VND/đêm
* Connecting room – số lượng: 2 phòng
* Loại phòng này thường được thiết kế theo dạng gia đình (phòng Family) hoặc khách hàng đi theo nhóm.
* Mức giá: 1.500.000 VND/đêm

**Bên cạnh đó, khách sạn còn được phân theo loại giường ngủ:**

* Twin bed room (TWB )
* Phòng có 2 giường đơn cho 2 người ngủ
* Mức giá: 700.000 VND/đêm
* Double bed room (DOB)
* Phòng 1 giường lớn cho 2 người ngủ
* Mức giá: 445.000 VND/đêm
* Triple bed room (TRB)
* Phòng 3 giường nhỏ hoặc 1 giường lớn + 1 giường lớn cho 3 người ngủ
* Mức giá: 900.000 VND/đêm

Khi khách hàng đã hoàn thành việc đặt phòng, trạng thái của phòng đó sẽ được cập nhật từ “Trống” sang “Đã đặt”. Trạng thái phòng sẽ chỉ được thay đổi trở lại thành “Trống” khi khách hàng đã thực sự thanh toán toàn bộ chi phí trên hóa đơn.

* Trường hợp khách hàng muốn gia hạn thêm ngày ở:
* Nếu còn trong thời gian thuê phòng, khách hàng có nhu cầu gia hạn thêm ngày nghỉ của phòng thì nhân viên sẽ tiến hành cập nhật thêm thông tin về số ngày ở của khách hàng.
* Trường hợp khách hàng sau khi thanh toán hóa đơn xong mới có nhu cầu gia hạn phòng, nếu đã có khách hàng khác đặt trước thì nhân viên sẽ giải quyết ưu tiên theo thứ tự trước sau

***1.2.4. Thông tin về các dịch vụ***

Khi khách hàng đặt phòng khách sạn, nhân viên lễ tân sẽ giới thiệu các dịch vụ và ghi nhận thông tin dịch vụ khách hàng đăng ký (nếu có). Khách hàng có thể yêu cầu dịch vụ khi tới check-in hoặc đang ở khách sạn.

Dịch vụ của khách sạn được có các loại dưới đây

* Dịch vụ Buffet: Khách sạn sẽ có các phiếu buffe cho khách hàng theo từng thời gian(Sáng, trưa, tối) với giá là 300.000 VNĐ/người
* Dịch vụ xe đưa đón sân bay: Khách sạn sẽ có xe riêng tới sân bay đón khách hàng hoặc chở khách hàng từ khách sạn ra tới sân bay (Khoảng cách từ khách sạn đến sân bay là 5km). Giá 200.000 VNĐ/người
* Dịch vụ giặt, ủi là: Khách hàng có thể yêu cầu giặt là quần áo nếu có nhu cầu. Thời gian trả đồ là 4 tiếng, giá 40.000/5kg
* Dịch vụ Spa: Khách hàng có thể trải nghiệm thư giãn toàn thân tại các phòng Spa của khách sạn như Massage body, Massage toàn thân, bấm huyệt, đá nóng, xông hơi, dưỡng da. Giá 450.000/giờ
* Dịch vụ café, nước uống, mì tôm: Phục vụ những khách hàng muốn uống nước, cà phê hoặc ăn mì tôm tạ phòng. Giá 20.000/chai nước, 15.000/gói cà phê, 20.000/gói mì tôm

Tiền sử dụng dịch vụ sẽ được nhân viên làm các phiếu sử dụng dịch vụ cho khách hàng và tiền sẽ được cộng vào tổng tiền ở hóa đơn.

***1.2.5. Thông tin về cách tính tiền Bill***

Một khách hàng sau khi check-out sẽ nhận được hóa đơn từ nhân viên lễ tân.

Tổng tiền = Tổng tiền của các chi tiết hóa đơn

Mỗi chi tiết hóa đơn sẽ của 1 phòng.

Tổng tiền chi tiết hóa đơn = Tổng tiền phòng + Tổng tiền dịch vụ - tiền cọc

Tiền phòng = Giá phòng \* số ngày ở + phụ phí (nếu có)

Tiền dịch vụ = Giá dịch vụ \* số lượng dịch vụ đó (theo người, giờ, hoặc sản phẩm)

## ***1.3 Mô tả chức năng bài toán***

***Đăng nhập, đăng xuất, tạo tài khoản theo thông tin nhân viên có sẵn (mã nhân viên), mỗi nhân viên có một tài khoản khác nhau***

* Quản lý thông tin khách hàng (xóa theo mã khách hàng, cập nhật thông tin khách hàng, thêm khách hàng, tìm kiếm khách hàng theo ID,…)
* Quản lý thông tin đặt phòng (tìm thông tin đặt phòng theo tháng, theo ngày)
* Quản lý thông tin đặt dịch vụ (tìm thông tin đặt dịch vụ theo tháng, theo ngày
* Quản lý thông tin hóa đơn (tìm hóa đơn theo tháng, theo ngày)
* Quản lý tài khoản (thêm, sửa, xóa tài khoản theo mã số nhân viên, thay đổi mật khẩu tài khoản theo mã số nhân viên)
* Quản lý thông tin doanh thu (xem thông tin doanh thu theo ngày, tháng, quý,  năm)

# **CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

## ***2.1. Thiết kế cơ sở dữ liệu mức quan niệm***

Ảnh có chứa văn bản, ảnh chụp màn hình, đen và trắng, vòng tròn

Mô tả được tạo tự động

## ***2.2. Thiết kế cơ sở dữ liệu mức logic***

Từ sơ đồ thực thể kết hợp (ERD), ta có các lược đồ quan hệ:

* CUSTOMER (Customer\_id, Customer\_name, Gender, Birthday, Identify\_card, Phone, Mail, Customer\_address),
* OFFICIAL\_CUSTOMER(Customer\_id, Customer\_name, Gender, Birthday, Identify\_card, Phone, Mail, Customer\_address),
* ACCOUNT (Username, Pass, Employee\_id(FK)),
* EMPLOYEE (Employee\_id, Employee\_name, Gender, Birthday, Identify\_card, Phone, Mail, Employee\_address),
* TYPE\_ROOM (Type\_room\_id, Type\_room\_name, Unit),
* ROOM (Room\_id, Room\_status, Room\_type(FK)),
* SERVICE\_ROOM (Service\_room\_id, Service\_room\_name, Unit),
* BILL (Bill\_id, Pay\_time, Employee\_id(FK), Payment\_method, Totel\_money),
* DETAILS\_BILL (Details\_bill\_id, Total\_day, Bill\_id(FK)),
* DETAILS\_RESERVED (Customer\_id, Room\_id, Details\_bill\_id, Reserved\_day, Date\_check\_in, Check\_room\_received, Deposit, Check\_paid\_deposit, Date\_create),
* DETAILS\_USED\_SERVICE (Room\_id, Details\_bill\_id, Service\_room\_id, Number\_of\_dervice, Date\_used),
* TRACKING\_LOG (Id, Customer\_id, Customer\_name, Identify\_card, Operation, Updated\_at)

## ***2.3. Mô tả cơ sở dữ liệu***

### **2.3.1. Bảng CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Customer\_id | varchar(20) | Mã khách hàng để phân biệt các khách hàng |
| 2 | Customer\_name | nvarchar(50) | Tên khách hàng |
| 3 | Gender | nvarchar(3) | Giới tính |
| 4 | Birthday | date | Ngày sinh của khách hàng |
| 5 | Identify\_card | varchar(20) | CCCD của khách hang |
| 6 | Phone | varchar(10) | SĐT của khách hàng |
| 7 | Mail | varchar(50) | Email của khách hàng |
| 8 | Customer\_address | Nvarchar(255) | Địa chỉ của khách hàng |

### **2.3.2. Bảng ACCOUNT**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Username | varchar(20) | Tài khoản đăng nhập |
| 2 | Pass | varchar(20) | Mật khẩu |
| 3 | Employee\_id | varchar(20) | Mã nhân viên của tài khoản |

### **2.3.3. Bảng EMPLOYEE**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Employee\_id | varchar(20) | Mã nhân viên để phân biệt các nhân viên |
| 2 | Employee\_name | nvarchar(50) | Tên nhân viên |
| 3 | Gender | nvarchar(6) | Giới tính |
| 4 | Birthday | date | Ngày sinh của nhân viên |
| 5 | Identify\_card | varchar(20) | CCCD của nhân viên |
| 6 | Phone | Varchar(10) | SĐT của nhân viên |
| 7 | Mail | Varchar(50) | Email của nhân viên |
| 8 | Employee\_address | Nvarchar(255) | Địa chỉ của nhân viên |

### **2.3.4. Bảng TYPE\_ROOM**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | Type\_room\_id | varchar(20) | Mã loại phòng dung để phân biệt các loại phòng khác nhau |
| 2 | Type\_room\_name | nvarchar(50) | Tên của loại phòng |
| 3 | Unit | money | Đơn giá của mỗi loại phòng theo ngày |
| 4 | Discount\_room | float | Số phần tram giảm giá phòng |

### **2.3.5. Bảng ROOM**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Room\_id | Varchar(20) | Mã phòng để phân biệt các phòng khác nhau |
| 2 | Room\_status | bit | Tình trạng phòng |
| 3 | Room\_type | Varchar(20) | Mã loại phòng |

### **2.3.6. Bảng SERVICE\_ROOM**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Service\_room\_id | varchar(20) | Mã dịch vụ để phân biệt với các dịch vụ khác nhau |
| 2 | Service\_room\_name | nvarchar(50) | Tên của dịch vụ |
| 3 | Unit | Money | Đơn giá |
| 4 | Discount\_service | float | Số phần tram giảm giá dịch vụ |

### **2.3.7. Bảng BILL**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Bill\_id | varchar(20) | Mã hóa đơn để phân biệt các hóa đơn khác nhau |
| 2 | Pay\_time | Datetim | Thời gian thanh toán |
| 3 | Employee\_id | Varchar(20) | Mã nhân viên lập hóa đơn |
| 4 | Payment\_method | Varchar(20) | Phương thức thanh toán |
| 5 | Totel\_money | money | Tổng tiền cần thanh toán |

### **2.3.8. Bảng DETAILS\_BILL**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Details\_bill\_id | varchar(20) | Mã chi tiết hóa đơn của hóa đơn |
| 2 | Total\_day | Int | Tổng số ngày ở |
| 3 | Bill\_id | Varchar(20) | Mã hóa đơn |

### **2.3.9. Bảng DETAILS\_RESERVED**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Customer\_id | char(20) | Mã khách hàng |
| 2 | Room\_id | varchar(20) | Mã phòng |
| 3 | Details\_bill\_id | Varchar(20) | Mã chi tiết hóa đơn |
| 4 | Reserved\_day | Int | Số ngày đặt phòng |
| 5 | Date\_check\_in | Datetime | Ngày tới nhận phòng |
| 6 | Check\_room\_received | Bit | Kiểm tra đã tới nhận phòng  hay chưa |
| 7 | Deposit | Money | Tiền cọc |
| 8 | Check\_paid\_deposit | Bit | Kiểm tra đã cọc hay chưa |
| 9 | Date\_create | Datetime | Ngày tạo phiếu đặt phòng |

### **2.3.10. Bảng DETAILS\_USED\_SERVICE**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Room\_id | varchar(20) | Mã phòng |
| 2 | Details\_bill\_id | varchar(20) | Mã chi tiết hóa đơn |
| 3 | Service\_room\_id | varchar(20) | Mã dịch vụ |
| 4 | Number\_of\_dervice | Int | Số lượng dịch vụ |
| 5 | Date\_used | datetime | Ngày sử dụng dịch vụ |

### **2.3.11. Bảng TRACKING\_LOG**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Id | int | Số thứ tự |
| 2 | Customer\_id | varchar(20) | Mã khách hang |
| 3 | Customer\_name | nvarchar(50) | Tên khách hang |
| 4 | Identify\_card | Varchar(20) | CCCD của khách hàng |
| 5 | Operation | Char(3) | Thao tác(thêm, sửa, xóa) |
| 6 | Updated\_at | datetime | Thời gian thay đổi |

### **2.3.12. Bảng OFFICIAL\_CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ Liệu** | **Diễn giải** |
| 1 | Customer\_id | varchar(20) | Mã khách hàng để phân biệt các khách hàng |
| 2 | Customer\_name | nvarchar(50) | Tên khách hàng |
| 3 | Gender | nvarchar(3) | Giới tính |
| 4 | Birthday | date | Ngày sinh của khách hàng |
| 5 | Identify\_card | varchar(20) | CCCD của khách hang |
| 6 | Phone | varchar(10) | SĐT của khách hàng |
| 7 | Mail | varchar(50) | Email của khách hàng |
| 8 | Customer\_address | Nvarchar(255) | Địa chỉ của khách hàng |

## ***2.4. Các ràng buộc cần có***

|  |  |  |
| --- | --- | --- |
| **STT** | **TABLE** | **RÀNG BUỘC** |
| **1** | TYPE\_ROOM | -Ràng buộc khóa chính: Type\_room\_id |
| **2** | ROOM | - Ràng buộc khóa chính: Room\_id  - Ràng buộc khóa ngoại:  + Room\_type tham chiếu đến **TYPE\_ROOM** |
| **3** | EMPLOYEE | - Ràng buộc khóa chính: Employee\_id |
| **4** | ACCOUNT | -Ràng buộc khóa chính: Username  -Ràng buộc khóa ngoại:  + Employee\_id tham chiếu đến **EMPLOYEE** |
| **5** | CUSTOMER | - Ràng buộc khóa chính: Customer\_id |
| **6** | TRACKING\_LOG | - Ràng buộc khóa chính: Id |
| **7** | SERVICE\_ROOM | - Ràng buộc khóa chính: Service\_room\_id |
| **8** | BILL | - Ràng buộc khóa chính: Bill\_id  - Ràng buộc khóa ngoại:  + Employee\_id tham chiếu đến **EMPLOYEE** |
| **9** | DETAILS\_BILL | - Ràng buộc khóa chính: Details\_bill\_id  - Ràng buộc khóa ngoại:  + Bill\_id tham chiếu đến **BILL** |
| **10** | DETAILS\_RESERVED | - Ràng buộc tập khóa chính: Room\_id, Details\_bill\_id, Customer\_id  - Ràng buộc khóa ngoại:  + Room\_id tham chiếu đến **ROOM**  + Customer\_id tham chiếu đến **CUSTOMER**  + Details\_bill\_id tham chiếu đến **DETAILS\_BILL** |
| **11** | DETAILS\_USED\_SERVICE | - Ràng buộc tập khóa chính: Room\_id, Details\_bill\_id, Service\_room\_id  - Ràng buộc khóa ngoại:  + Room\_id tham chiếu đến **ROOM**  + Details\_bill\_id tham chiếu đến **DETAILS\_BILL**  + Service\_room\_id tham chiếu đến **SERVICE\_ROOM** |
| **12** | OFFCIAL\_CUSTOMER | - Ràng buộc khóa chính: Customer\_id |

## ***2.5. Mô tả cơ sở dữ liệu***

### **2.5.1. Tạo các bảng và các ràng buộc đơn giản**

***Bảng CUSTOMER***

CREATE TABLE CUSTOMER(

Customer\_id varchar(20) constraint PK\_customer primary key,

Customer\_name nvarchar(50) not null,

Gender nvarchar(6),

Birthday date check (DATEDIFF(year, Birthday, GETDATE())>=18),   
 --Khách hàng phải trên 18 tuổi

Identify\_card varchar(20) not null check(len(Identify\_Card)=12),

--CCCD đúng định dạng

Phone varchar(10) not null check(len(Phone)=10),

Mail varchar(50) check(Mail like'%@gmail.com'),

Customer\_address nvarchar(255)

);

***Bảng ACCOUNT***

CREATE TABLE ACCOUNT(

Username varchar(20) constraint PK\_account primary key,

Pass varchar(20) not null check(len(Pass)>=6),

Employee\_id varchar(20) constraint FK\_employee references EMPLOYEE(Employee\_id)

ON DELETE CASCADE -- Xóa tài khoản nhân viên thì thông tin nhân viên đó cũng bị xóa

);

***Bảng EMPLOYEE***

CREATE TABLE EMPLOYEE(

Employee\_id varchar(20) constraint PK\_employee primary key,

Employee\_name nvarchar(50) not null,

Gender nvarchar(6),

Birthday date check (DATEDIFF(year, Birthday, GETDATE())>=18), --Nhân viên phải trên 18 tuổi

Identify\_card varchar(20) not null check(len(Identify\_Card)=12), --CCCD đúng định dạng

Phone varchar(12) not null check(len(Phone)=10),

Mail varchar(50) check(Mail like'%@gmail.com'),

Employee\_address nvarchar(255)

);

***Bảng TYPE\_ROOM***

CREATE TABLE TYPE\_ROOM(

Type\_room\_id varchar(20) constraint PK\_typeroom primary key,

Type\_room\_name nvarchar(50),

Unit money, --VND

Discount\_room float

);

***Bảng ROOM***

CREATE TABLE ROOM(

Room\_id varchar(20) constraint PK\_room primary key,

Room\_status bit DEFAULT 0,

Room\_type varchar(20) constraint FK\_room references TYPE\_ROOM(Type\_room\_id)

);

***Bảng SERVICE\_ROOM***

CREATE TABLE SERVICE\_ROOM(

Service\_room\_id varchar(20) constraint PK\_service\_room primary key,

Service\_room\_name nvarchar(100),

Unit money, --VND

Discount\_service float --Giam gia

);

***Bảng BILL***

CREATE TABLE BILL(

Bill\_id varchar(20) constraint PK\_bill primary key,

Pay\_time datetime, --Xác định đã thanh toán, pay\_time chính là thời gian thanh toán hóa đơn

Employee\_id varchar(20) constraint FK references EMPLOYEE(Employee\_id),

Payment\_method nvarchar(20), -- Phương thức thanh toán

Total\_money money, --Tạo triggers tự động tính tổng tiền gộp số tiền dịch vụ và số tiền phòng, nhớ trừ ra

);

***Bảng DETAILS\_BILL***

CREATE TABLE DETAILS\_BILL(

Details\_bill\_id varchar(20) constraint PK\_details\_bill primary key,

Total\_day int,

Bill\_id varchar(20) constraint FK\_bill\_id references BILL(Bill\_id)

ON DELETE CASCADE -- Khi khách hàng hủy đặt phòng hoặc k tới nhận phòng thì xóa Bill\_ID và các dữ liệu liên quan đến BILL\_id này đều bị xóa

);

***Bảng DETAILS\_RESERVED***

CREATE TABLE DETAILS\_RESERVED(

Customer\_id varchar(20) constraint FK\_customer\_id references CUSTOMER(Customer\_id),

Room\_id varchar(20) constraint FK\_room\_id\_reserved references ROOM(room\_id),

Details\_bill\_id varchar(20) constraint FK\_details\_bill\_id\_reserved references DETAILS\_BILL(Details\_bill\_id)

ON DELETE CASCADE, -- Khi khách hàng hủy đặt phòng hoặc k tới nhận phòng thì xóa Bill\_ID và các dữ liệu liên quan đến BILL\_id này đều bị

Reserved\_day int,

Date\_check\_in datetime, --Thời gian check-in phải có rõ ngày tháng năm giờ phút giây

Constraint PK\_details\_reserved Primary Key (Customer\_id, Room\_id, Details\_bill\_id),

Check\_room\_received bit DEFAULT 0, --Để xác định được đã tới nhận phòng hay chưa

Deposit money, -- số tiền cọc 10% tiền phòng, nếu như thuê phòng trực tiếp thì deposit = null

Check\_paid\_deposit bit, --Check xem đã trả tiền cọc hay chưa, nếu như thuê phòng trực tiếp thì = null

Date\_create datetime,

);

***Bảng DETAILS\_USED\_SERVICE***

CREATE TABLE DETAILS\_USED\_SERVICE(

Room\_id varchar(20) constraint FK\_room\_id\_service references ROOM(room\_id),

Details\_bill\_id varchar(20) constraint FK\_details\_bill\_id\_service references DETAILS\_BILL(Details\_bill\_id)

ON DELETE CASCADE,

Service\_room\_id varchar(20) constraint FK\_service\_room\_id references SERVICE\_ROOM(Service\_room\_id),

Number\_of\_service int,

Date\_used datetime,

Constraint FK\_details\_used\_service Primary Key (Room\_id, Details\_bill\_id, Service\_room\_id)

);

***Bảng TRACKING\_LOG***

-- Bảng TRACKING\_LOG để ghi nhận lịch sử thay đổi khi thêm khách hàng hoặc xóa khách hàng nhằm tránh mất dữ liệu khách hàng đã từng đặt phòng

CREATE TABLE TRACKING\_LOG(

Id int DEFAULT 1 constraint PK\_trackinglog primary key,

Customer\_id varchar(20),

Customer\_name nvarchar(50),

Identify\_card varchar(20) not null check(len(Identify\_Card)=12),

Operation char(3),

Updated\_at datetime,

);

## 

## ***2.6. Triggers kiểm tra các ràng buộc***

***1. Trigger cập nhật tình trạng phòng khi có người đặt***

--1. Cập nhật tình trạng phòng khi có người đặt

|  |
| --- |
| create trigger Update\_status\_room\_reserved on Details\_Reserved  after insert as  begin  update Room  set room\_status=1  from Room  where Room.room\_id = (select room\_id from inserted )  end; |

-Phòng SUP002 trước khi có người đặt

Ảnh có chứa bàn

Mô tả được tạo tự động

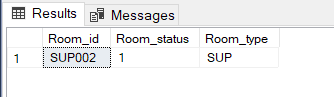
Ảnh có chứa văn bản

Mô tả được tạo tự động

-Sau khi chèn thêm vào chi tiết đặt phòng

Ảnh có chứa bàn

Mô tả được tạo tự động



***2. Trigger cập nhật tiền cọc phòng = 30% giá tiền phòng (đặt online)***

--2. Cập nhật tiền cọc phòng = 30% giá tiền phòng (đặt online)

|  |
| --- |
| create trigger Update\_deposit on DETAILS\_RESERVED  after insert  as  begin  declare @deposit money, @date\_create DateTime, @date\_checkin DateTime, @details\_bill\_id varchar(20);  select @date\_create = Date\_create, @date\_checkin = Date\_check\_in, @details\_bill\_id =Details\_bill\_id from inserted;      select @deposit = 0.3 \* Unit  from TYPE\_ROOM tr  join ROOM r on tr.Type\_room\_id = r.Room\_type  join inserted ins on ins.Room\_id = r.Room\_id;  --khi ngày tạo chi tiết đặt phòng khác với ngày check in thì đặt online -> cần tính cọc,  --còn đặt offline thì ngày tạo chi tiết đặt phòng cũng là ngày check in luôn -> không cần cọc -> gán bằng 0  if(@date\_create <> @date\_checkin)  update DETAILS\_RESERVED set Deposit = @deposit where DETAILS\_RESERVED.Details\_bill\_id = @details\_bill\_id;  else  update DETAILS\_RESERVED set Deposit = 0 where DETAILS\_RESERVED.Details\_bill\_id = @details\_bill\_id;  end; |

-Khi thêm vào chi tiết đặt phòng chưa dùng trigger

Ảnh có chứa bàn

Mô tả được tạo tự động

-Sau khi thực hiện chèn, trigger được thực hiện

Ảnh có chứa bàn

Mô tả được tạo tự động

***3. Trigger tính tổng tiền cho hóa đơn khi cập nhật bảng chi tiết sử dụng dịch vụ***

3. Trigger tính tổng tiền cho hóa đơn khi cập nhật bảng chi tiết sử dụng dịch vụ(không cần bảng chi tiết đặt phòng vì mình--đã có tổng số ngày ở rồi)

|  |
| --- |
| CREATE TRIGGER Calc\_total\_money\_service ON DETAILS\_USED\_SERVICE  AFTER INSERT, UPDATE  AS  BEGIN  --Khai báo biến  DECLARE @details\_bill\_id varchar(20), @money\_service money, @disco float, @number\_used\_old int = 0, @number\_used\_new int;  -- Lấy mã chi tiết hóa đơn, số lượng sử dụng dịch vụ mới được cập nhật từ hàng vừa chèn  SELECT @details\_bill\_id = Details\_bill\_id, @number\_used\_new = Number\_of\_service FROM inserted  --Lấy số lượng sử dụng dịch vụ cũ vừa được xóa  select @number\_used\_old = Number\_of\_service from deleted where deleted.Details\_bill\_id = @details\_bill\_id;  --Tính tiền dịch vụ  SET @money\_service = (SELECT SUM(Unit \* (@number\_used\_new - @number\_used\_old) \* (1 - Discount\_service) )  FROM SERVICE\_ROOM JOIN inserted ON SERVICE\_ROOM.Service\_room\_id = inserted.Service\_room\_id  WHERE inserted.Details\_bill\_id = @details\_bill\_id)  DECLARE @total\_money\_old money;  SELECT @total\_money\_old = Total\_money FROM BILL WHERE Bill\_id = (select Bill\_id from DETAILS\_BILL WHERE Details\_bill\_id = @details\_bill\_id)  -- Update the total\_money in Bill table using details\_bill\_id  UPDATE BILL  SET Total\_money = @total\_money\_old + @money\_service  WHERE Bill\_id = (SELECT bill\_id  FROM Details\_Bill  WHERE details\_bill\_id = @details\_bill\_id);  END; |

-Trước khi thêm vào chi tiết đặt phòng có mã Details\_bill\_id = Bill0005A của chi tiết hóa đơn liên kết với hóa đơn có mã Bill0005 chưa dùng trigger

Ảnh có chứa bàn

Mô tả được tạo tự động

-Sau khi thêm bản ghi có Details\_bill\_id = Bill0005A của khách hàng có mã KH0005 và có số ngày đã đặt trước là 1

Ảnh có chứa bàn

Mô tả được tạo tự động

-Phòng SUP002 thuộc loại phòng Superior đang giảm giá 30%, số tiền sẽ được điền vào hóa đơn là 2\*525000-0.3\*525000-0.3\*525000=420000

Ảnh có chứa bàn

Mô tả được tạo tự động

-Tiền phòng đã được cập nhật trong hóa đơn

Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

***4. Trigger cập nhật trạng thái phòng thành 0 khi thanh toán.***

--4. Cập nhật trạng thái phòng thành 0 khi thanh toán. Kiểm tra xem trong Bill mà ngày tạo khác null(đã thanh toán) thì kết bảng và đặt lại trạng thái phòng

|  |
| --- |
| create trigger Update\_status\_room\_checkouted on Bill  after update as  begin  update Room  set Room\_status = 0  where Room.room\_id in ( select room\_id  from Details\_Reserved  where details\_bill\_id in ( select Details\_bill\_id  from DETAILS\_BILL  where DETAILS\_BILL.Bill\_id = (select Bill\_id from inserted where inserted.Pay\_time is not null) ) )  end; |

-Trước khi cập nhật ngày tạo hóa đơn Pay\_time vào trong Bill, phòng SUP002 có trạng thái là 1

Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

-Sau khi cập nhật ngày trả Pay\_time, trạng thái phòng SUP002 đã chuyển thành 0

Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

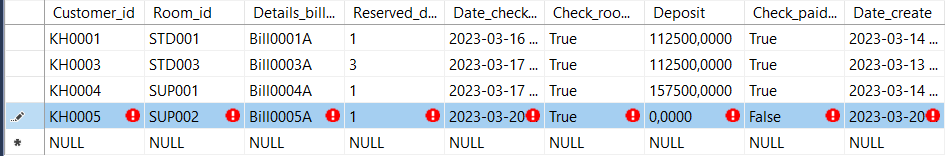
***5. Trigger cập nhật số ngày đã đặt trước cho tổng số ngày trong chi tiết hóa đơn khi đặt thành công hoặc gia hạn thành công***

--5. Cập nhật số ngày đã đặt trước cho tổng số ngày trong chi tiết hóa đơn

  khi đặt thành công hoặc gia hạn thành công

|  |
| --- |
| CREATE TRIGGER Update\_total\_day ON DETAILS\_RESERVED  AFTER INSERT, UPDATE  AS  BEGIN  -- Declare a variable to store the detail bill id of the inserted or updated row  DECLARE @details\_bill\_id varchar(20), @total\_day int;  -- Assign the value of detail bill id from the inserted table  SELECT @details\_bill\_id = Details\_bill\_id, @total\_day = Reserved\_day FROM inserted;    --cập nhật số ngày đã đặt trước cho tổng số ngày trong chi tiết hóa đơn  --khi mà đặt thành công hoặc gia hạn thành công  UPDATE DETAILS\_BILL  SET Total\_day = @total\_day  WHERE DETAILS\_BILL.Details\_bill\_id = @details\_bill\_id;  END; |

-Trước khi cập nhật ngày đã đặt trước cho phòng SUP002 bằng 1 có Details\_bill\_id là Bill0005A



Ảnh có chứa bàn

Mô tả được tạo tự động

-Sau khi chèn, Total\_day của chi tiết hóa đơn mã Bill0005A đã chuyển thành 1

Ảnh có chứa bàn

Mô tả được tạo tự động

***6. Trigger xóa khỏi chi tiết đặt phòng nếu sau 30 phút mà vẫn chưa chuyển tiền cọc(đặt online)***

--6. Xóa khỏi chi tiết đặt phòng nếu sau 30 phút mà vẫn chưa chuyển tiền cọc(đặt online)

|  |
| --- |
| create trigger Update\_status\_room\_deposited on DETAILS\_RESERVED  after update  as  begin  --Xóa khỏi bảng Details\_Reserved  delete from DETAILS\_RESERVED  where GETDATE()> DATEADD(minute, 30, Date\_create)    and Check\_paid\_deposit = 0    and Date\_create <> Date\_check\_in;  end; |

-Khi thêm vào chi tiết đặt phòng mà ngày giờ hiện tại lớn hơn ngày tạo Date\_create 30 phút đồng thời thuộc tính check\_paid\_deposit = 0 thì sẽ tự động xóa chi tiết đặt phòng

Ảnh có chứa văn bản

Mô tả được tạo tự động

-Bản ghi đã bị xóa và không xuất hiện trong bảng DETAILS\_RESERVED

Ảnh có chứa văn bản

Mô tả được tạo tự động

***7. Trigger cập nhật tình trạng phòng và xóa hóa đơn khi xóa chi tiết đặt phòng (do không chuyển cọc, do không nhận phòng)***

--7. Cập nhật tình trạng phòng và xóa hóa đơn khi xóa chi tiết đặt phòng (do không chuyển cọc, do không nhận phòng)

|  |
| --- |
| create trigger Delete\_bill\_and\_details\_bill on DETAILS\_RESERVED  after delete  as  begin    update ROOM  set Room\_status = 0  from Room join deleted on Room.Room\_id = deleted.Room\_id  where ROOM.Room\_id = deleted.Room\_id;  --xóa khỏi bảng Bill và Details\_Bill  delete from BILL where BILL.Bill\_id in (select Bill\_id  from DETAILS\_BILL join deleted on  deleted.Details\_bill\_id = DETAILS\_BILL.Details\_bill\_id);  delete from DETAILS\_BILL where DETAILS\_BILL.Details\_bill\_id in (select old.Details\_bill\_id from deleted as old);    --không cần xóa trong Details\_Used\_Service vì khi xóa Details\_bill\_id trong Details\_Bill sẽ tự  --động xóa (do delete cascade)  end; |

***8. Trigger cập nhật tiền phòng vào hóa đơn mỗi khi thêm, cập nhật ngày đặt phòng(trường hợp khách hàng không sử dụng dịch vụ) cho từng chi tiết đặt phòng)***

--8. Cập nhật tiền phòng vào hóa đơn khi thêm, cập nhật ngày đặt phòng(trường hợp khách hàng không sử dụng dịch vụ)

|  |
| --- |
| CREATE TRIGGER Update\_room\_money ON DETAILS\_RESERVED  AFTER INSERT, UPDATE  AS  BEGIN  DECLARE @details\_bill\_id varchar(20), @total\_day int, @price money, @disco float, @total\_money money = 0.0000, @total\_day\_old int = 0;  SELECT @details\_bill\_id = inserted.Details\_bill\_id, @total\_day = inserted.Reserved\_day FROM inserted;  SELECT @price = Unit, @disco =Discount\_room FROM TYPE\_ROOM JOIN ROOM ON TYPE\_ROOM.Type\_room\_id = ROOM.Room\_type  JOIN DETAILS\_RESERVED ON ROOM.Room\_id = DETAILS\_RESERVED.Room\_id  WHERE Details\_Reserved.details\_bill\_id = @details\_bill\_id;  select @total\_day\_old = Reserved\_day from deleted where deleted.Details\_bill\_id = @details\_bill\_id;  SELECT @total\_money = Total\_money FROM BILL WHERE BILL.Bill\_id = (select Bill\_id from DETAILS\_BILL where DETAILS\_BILL.Details\_bill\_id = @details\_bill\_id) and Total\_money is not null  --Cập nhật tiền phòng  if(@total\_day > @total\_day\_old)  UPDATE BILL  SET Total\_money = @total\_money + @price \* (1 - @disco) \* (@total\_day - @total\_day\_old) - (select Deposit from DETAILS\_RESERVED where Details\_bill\_id = @details\_bill\_id)  WHERE Bill\_id = (SELECT bill\_id  FROM Details\_Bill  WHERE details\_bill\_id = @details\_bill\_id);  END; |

-Trước khi thêm vào chi tiết đặt phòng có mã Details\_bill\_id = Bill0005A của chi tiết hóa đơn liên kết với hóa đơn có mã Bill0005 chưa dùng trigger

Ảnh có chứa bàn

Mô tả được tạo tự động

-Sau khi thêm bản ghi có Details\_bill\_id = Bill0005A của khách hàng có mã KH0005 và có số ngày đã đặt trước là 1

Ảnh có chứa bàn

Mô tả được tạo tự động

-Phòng SUP002 thuộc loại phòng Superior đang giảm giá 30%, số tiền sẽ được điền vào hóa đơn là 2\*525000-0.3\*525000-0.3\*525000=420000

Ảnh có chứa bàn

Mô tả được tạo tự động

-Tiền phòng đã được cập nhật trong hóa đơn

Ảnh có chứa bàn

Mô tả được tạo tự động

***9. Trigger tính thêm tiền vào trong hóa đơn nếu check in sớm hơn 14h (đối với khách hàng đặt phòng online)***

--9. Tính thêm tiền vào trong hóa đơn nếu check in sớm hơn 14h (đối với khách hàng đặt phòng online)

|  |
| --- |
| create trigger Update\_total\_money\_checkin\_early\_onl on DETAILS\_RESERVED  after update  as  begin  --khai báo biến  declare @date\_checkin\_new DateTime, @date\_checkin\_old DateTime, @room\_id varchar(20), @price money, @total\_money money = 0.0000, @details\_bill\_id varchar(20), @bill\_id varchar(20);  --lấy các giá trị cho biến  select @date\_checkin\_new = Date\_check\_in, @room\_id = Room\_id, @details\_bill\_id = Details\_bill\_id from inserted  select @date\_checkin\_old = Date\_check\_in from deleted  select @bill\_id = Bill\_id from DETAILS\_BILL where DETAILS\_BILL.Details\_bill\_id = @details\_bill\_id  select @price = Unit from TYPE\_ROOM join ROOM on TYPE\_ROOM.Type\_room\_id = ROOM.Room\_type where ROOM.Room\_id = @room\_id  select @total\_money = Total\_money from BILL where Bill\_id = @bill\_id and Total\_money is not null  if ( @date\_checkin\_new <> @date\_checkin\_old and DATEPART(HOUR, @date\_checkin\_new) >= 5 and DATEPART(HOUR, @date\_checkin\_new) <= 8 )--kiểm tra giờ checkin mới chèn hoặc cập nhật nếu từ 5-9 thì tính thêm 50% giá phòng  begin  update BILL  set Total\_money = @total\_money + 0.5 \* @price  where Bill\_id = @bill\_id  end  else if ( @date\_checkin\_new <> @date\_checkin\_old and DATEPART(HOUR, @date\_checkin\_new) >= 9 and DATEPART(HOUR, @date\_checkin\_new) <= 13 )----kiểm tra giờ checkin mới chèn hoặc cập nhật nếu từ 9-14 thì tính thêm 30% giá phòng  begin  update BILL  set Total\_money = @total\_money + 0.3 \* @price  where Bill\_id = @bill\_id  end  end; |

-Chi tiết hóa đơn mã Details\_bill\_id = Bill0005A đã bị xóa khỏi bảng

Ảnh có chứa bàn

Mô tả được tạo tự động

-Hóa đơn có liên kết với chi tiết hóa đơn đã bị xóa

Ảnh có chứa bàn

Mô tả được tạo tự động

-Phòng đã được cập nhật trạng thái thành 0

Ảnh có chứa bàn

Mô tả được tạo tự động

***10. Trigger tính thêm tiền vào trong hóa đơn nếu check in sớm hơn 14h (đối với khách hàng đặt phòng offline)***

--10. Tính thêm tiền vào trong hóa đơn nếu check in sớm hơn 14h (đối với khách hàng đặt phòng offline)

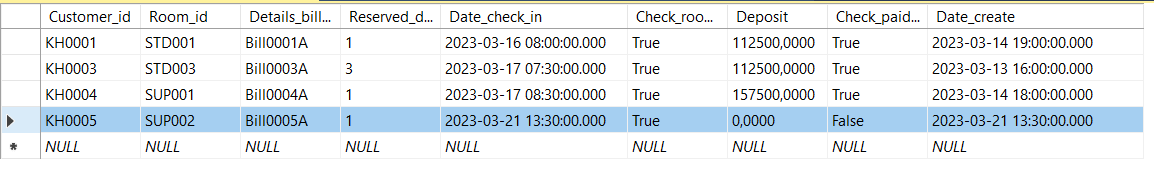
|  |
| --- |
| create trigger Update\_total\_money\_checkin\_early\_off on DETAILS\_RESERVED  after insert  as  begin  --khai báo biến  declare @date\_checkin DateTime, @room\_id varchar(20), @price money, @total\_money money = 0.0000, @details\_bill\_id varchar(20), @bill\_id varchar(20);  --lấy các giá trị cho biến  select @date\_checkin= Date\_check\_in, @room\_id = Room\_id, @details\_bill\_id = Details\_bill\_id from inserted  select @bill\_id = Bill\_id from DETAILS\_BILL where DETAILS\_BILL.Details\_bill\_id = @details\_bill\_id  select @price = Unit from TYPE\_ROOM join ROOM on TYPE\_ROOM.Type\_room\_id = ROOM.Room\_type where ROOM.Room\_id = @room\_id  select @total\_money = Total\_money from BILL where Bill\_id = @bill\_id and Total\_money is not null  if ( DATEPART(HOUR, @date\_checkin) >= 5 and DATEPART(HOUR, @date\_checkin) <= 8 )--kiểm tra giờ checkin mới chèn hoặc cập nhật nếu từ 5-9 thì tính thêm 50% giá phòng  begin  update BILL  set Total\_money = @total\_money + 0.5 \* @price  where Bill\_id = Bill\_id  end  else if ( DATEPART(HOUR, @date\_checkin) >= 9 and DATEPART(HOUR, @date\_checkin) <= 13 )----kiểm tra giờ checkin mới chèn hoặc cập nhật nếu từ 9-14 thì tính thêm 30% giá phòng  begin  update BILL  set Total\_money = @total\_money + 0.3 \* @price  where Bill\_id = @bill\_id  end  end; |

-Trước khi khách đặt offline có chi tiết đặt phòng thì tổng tiền trong hóa đơn sẽ NULL

Ảnh có chứa bàn

Mô tả được tạo tự động

Sau khi thêm vào chi tiết đặt phòng với ngày check in là 2023-03-21 lúc 13 giờ 30 phút, trong hóa đơn sẽ có tổng tiền phòng và tiền phạt check in sớm: (525000 \* 1(tổng tiền phòng) – 525000 \* 0,3(tiền giảm giá phòng 30%) + 525000 \* 0,3 (tiền phạt checkin sớm) = 525000)



Ảnh có chứa bàn

Mô tả được tạo tự động

***11. Trigger ghi nhận lịch sử khi thêm khách hàng***

|  |
| --- |
| --Trigger ghi nhận lịch sử khi thêm khách hàng  ALTER TRIGGER TrackingLogCustomer\_Ins  ON OFFICIAL\_CUSTOMER  AFTER INSERT  AS  declare @Cusomer\_id varchar(20), @Customer\_name nvarchar(50), @Identify\_card varchar(20), @Update\_at Datetime, @Operation char(3);  DECLARE @maxID int = (SELECT COALESCE(MAX(ID),0) FROM TRACKING\_LOG)  BEGIN  SET NOCOUNT ON;  SELECT  @Cusomer\_id = Official\_customer\_id,  @Customer\_name = Customer\_name,  @Identify\_card=Identify\_card  FROM inserted  INSERT INTO TRACKING\_LOG(  ID,  Customer\_id,  Customer\_name,  Identify\_card,  Operation,  Updated\_at  )  values(@maxID+1,@Cusomer\_id,@Customer\_name,@Identify\_card,'INS',GETDATE())  END |

***12. Trigger ghi nhận lịch sử khi xóa khách hàng***

|  |
| --- |
| --Trigger ghi nhận lịch sử khi xóa khách hàng----------------  GO  CREATE TRIGGER TrackingLogCustomer\_Del  ON OFFICIAL\_CUSTOMER  AFTER DELETE  AS  declare @Cusomer\_id varchar(20), @Customer\_name nvarchar(50), @Identify\_card varchar(20), @Update\_at Datetime, @Operation char(3);  DECLARE @maxID int = (SELECT COALESCE(MAX(ID),0) FROM TRACKING\_LOG)  BEGIN  SET NOCOUNT ON;  SELECT  @Cusomer\_id = Official\_customer\_id,  @Customer\_name = Customer\_name,  @Identify\_card=Identify\_card  FROM deleted  INSERT INTO TRACKING\_LOG(  ID,  Customer\_id,  Customer\_name,  Identify\_card,  Operation,  Updated\_at  )  values(@maxID+1,@Cusomer\_id,@Customer\_name,@Identify\_card,'DEL',GETDATE())  END |

***13. Trigger ghi nhận lịch sử khi cập nhật khách hàng***

|  |
| --- |
| ------------Trigger ghi nhận lịch sử khi cập nhật khách hàng-------------  GO  CREATE TRIGGER TrackingLogCustomer\_Upd  ON OFFICIAL\_CUSTOMER  AFTER UPDATE  AS  declare @Cusomer\_id varchar(20), @Customer\_name nvarchar(50), @Identify\_card varchar(20), @Update\_at Datetime, @Operation char(3);  DECLARE @maxID int = (SELECT COALESCE(MAX(ID),0) FROM TRACKING\_LOG)  BEGIN  SET NOCOUNT ON;  SELECT  @Cusomer\_id = Official\_customer\_id,  @Customer\_name = Customer\_name,  @Identify\_card=Identify\_card  FROM deleted  INSERT INTO TRACKING\_LOG(  ID,  Customer\_id,  Customer\_name,  Identify\_card,  Operation,  Updated\_at  )  values(@maxID+1,@Cusomer\_id,@Customer\_name,@Identify\_card,'UPD',GETDATE())  END |

***14. Trigger thông báo khi xóa BILL***

|  |
| --- |
| -----------Triggers thông báo khi xóa BILL- Thông báo là Đã xóa Bill------------  GO  CREATE TRIGGER Noti\_Delete\_BILL on BILL  FOR DELETE  AS  BEGIN  DECLARE @Bill\_id varchar(20)  SELECT @Bill\_id = ol.Bill\_id  FROM  deleted AS ol  IF (@Bill\_id NOT IN (SELECT Bill\_id FROM BILL))  PRINT N'Đã xóa Hóa đơn có mã' + RTRIM(@Bill\_id)  END |

***15. Trigger thông báo khi thêm BILL***

|  |
| --- |
| -----------Triggers thông báo khi thêm BILL- Thông báo là Đã thêm Bill------------  GO  CREATE TRIGGER Noti\_Insert\_BILL on BILL  FOR INSERT  AS  BEGIN  DECLARE @Bill\_id varchar(20)  SELECT @Bill\_id = ol.Bill\_id  FROM  inserted AS ol  IF (@Bill\_id IN (SELECT Bill\_id FROM BILL))  PRINT N'Đã thêm Hóa đơn có mã ' + RTRIM(@Bill\_id)  END |

***16. Trigger thông báo khi cập nhật BILL***

|  |
| --- |
| -----------Triggers thông báo khi cập nhật BILL- Thông báo là Đã cập nhật Bill---------  GO  CREATE TRIGGER Noti\_Update\_BILL on BILL  FOR UPDATE  AS  BEGIN  DECLARE @Bill\_id varchar(20)  SELECT @Bill\_id = ol.Bill\_id  FROM  deleted AS ol  IF (@Bill\_id IN (SELECT Bill\_id FROM BILL))  PRINT N'Đã cập nhật Hóa đơn có mã ' + RTRIM(@Bill\_id)  END |

***17. Trigger thông báo khi thêm DETAIL BILL***

|  |
| --- |
| ---------Triggers thông báo khi thêm DETAIL BILL- Thông báo là Đã thêm DETAILS\_BILL---  GO  CREATE TRIGGER Noti\_Insert\_DETAIL\_BILL on DETAILS\_BILL  FOR INSERT  AS  BEGIN  DECLARE @DetailBill\_id varchar(20), @Bill\_id varchar(20)  SELECT @DetailBill\_id = ol.Details\_bill\_id, @Bill\_id=ol.Bill\_id  FROM  inserted AS ol  IF (@DetailBill\_id IN (SELECT Details\_bill\_id FROM DETAILS\_BILL))  PRINT N'Đã thêm Chi tiết Hóa đơn của Hóa đơn có mã ' + RTRIM(@Bill\_id)  END |

***18. Trigger thông báo khi cập nhật Details\_BILL ID***

|  |
| --- |
| -----------Triggers thông báo khi cập nhật Details\_BILL - Thông báo là Đã cập nhật Detail\_Bill-----  GO  CREATE TRIGGER Noti\_Update\_DETAIL\_BILL on DETAILS\_BILL  FOR UPDATE  AS  BEGIN  DECLARE @DetailBill\_id varchar(20), @Bill\_id varchar(20)  SELECT @DetailBill\_id = ol.Details\_bill\_id, @Bill\_id=ol.Bill\_id  FROM  deleted AS ol  IF (@DetailBill\_id IN (SELECT Details\_bill\_id FROM DETAILS\_BILL))  PRINT N'Đã cập nhật Chi tiết Hóa đơn của Hóa đơn có mã ' + RTRIM(@Bill\_id)  END |

***19. Trigger thông báo khi xóa Customer***

|  |
| --- |
| -------------Triggers thông báo khi xóa Customer------------------  GO  CREATE TRIGGER Noti\_Delete\_CUS on CUSTOMER  FOR DELETE  AS  BEGIN  DECLARE @Customer\_id varchar(20), @Customer\_name nvarchar(50)  SELECT @Customer\_id=ol.Customer\_id, @Customer\_name=ol.Customer\_name  FROM deleted AS ol  IF (@Customer\_id NOT IN (SELECT Customer\_id FROM CUSTOMER))  PRINT N'Đã xóa thông tin Khách hàng ' + RTRIM(@Customer\_name)  END |

***20. Trigger thông báo khi thêm Customer***

|  |
| --- |
| -------------Triggers thông báo khi thêm Customer------------------  CREATE TRIGGER Noti\_Insert\_CUS on CUSTOMER  FOR INSERT  AS  BEGIN  DECLARE @Customer\_id varchar(20), @Customer\_name nvarchar(50)  SELECT @Customer\_id = new.Customer\_id, @Customer\_name = new.Customer\_name  FROM inserted AS new  IF (@Customer\_id IN (SELECT Customer\_id FROM CUSTOMER))  PRINT N'Đã thêm thông tin Khách hàng ' + RTRIM(@Customer\_name)  END |

***21. Trigger thông báo khi thêm Customer mà Customer đã tồn tại***

|  |
| --- |
| -------------Triggers thông báo khi thêm Customer mà Customer đã tồn tại---------------  CREATE TRIGGER Noti\_Update\_CUS ON CUSTOMER  FOR INSERT  AS  BEGIN  IF (SELECT COUNT(\*) FROM CUSTOMER c JOIN inserted i ON c.Identify\_card = i.Identify\_card) > 1  BEGIN  RAISERROR('Thông tin khách hàng đã tồn tại trước đó!', 16, 1);  ROLLBACK TRANSACTION;  END  END |

***22. Trigger thông báo khi thêm Phiếu đặt phòng thành công***

|  |
| --- |
| ---------------------Trigger thông báo khi thêm Phiếu đặt phòng thành công---------------  GO  CREATE TRIGGER Noti\_Insert\_Reserved\_Room on DETAILS\_RESERVED  FOR INSERT  AS  BEGIN  DECLARE @Room\_id varchar(20)  SELECT @Room\_id=ol.Room\_id  FROM inserted AS ol  IF (@Room\_id IN (SELECT Room\_id FROM DETAILS\_RESERVED))  PRINT N'Đã thêm phiếu đặt phòng cho phòng ' + RTRIM(@Room\_id) + N' thành công'  END |

***23. Trigger thông báo khi cập nhật Phiếu đặt phòng thành công***

|  |
| --- |
| ----------------Trigger thông báo khi cập nhật Phiếu đặt phòng thành công----------  GO  CREATE TRIGGER Noti\_Update\_Reserved\_Room on DETAILS\_RESERVED  FOR UPDATE  AS  BEGIN  DECLARE @Room\_id varchar(20)  SELECT @Room\_id=ol.Room\_id  FROM deleted AS ol  IF (@Room\_id IN (SELECT Room\_id FROM DETAILS\_RESERVED))  PRINT N'Đã cập nhật phiếu đặt phòng cho phòng ' + RTRIM(@Room\_id) + N' thành công'  END |

***24. Trigger thông báo khi thêm Phiếu sử dụng dịch vụ thành công***

|  |
| --- |
| -----------------Trigger thông báo khi thêm Phiếu sử dụng dịch vụ thành công---------------  GO  CREATE TRIGGER Noti\_Insert\_Use\_Service on DETAILS\_USED\_SERVICE  FOR INSERT  AS  BEGIN  DECLARE @Room\_id varchar(20)  SELECT @Room\_id=ol.Room\_id  FROM inserted AS ol  IF (@Room\_id IN (SELECT Room\_id FROM DETAILS\_USED\_SERVICE))  PRINT N'Đã thêm phiếu sử dụng dịch vụ cho phòng ' + RTRIM(@Room\_id) + N' thành công'  END |

***25. Trigger thông báo khi cập nhật Phiếu sử dụng dịch vụ thành công***

|  |
| --- |
| ------------Trigger thông báo khi cập nhật Phiếu sử dụng dịch vụ thành công---------------  GO  CREATE TRIGGER Noti\_Update\_Use\_Service on DETAILS\_USED\_SERVICE  FOR UPDATE  AS  BEGIN  DECLARE @Room\_id varchar(20)  SELECT @Room\_id=ol.Room\_id  FROM deleted AS ol  IF (@Room\_id IN (SELECT Room\_id FROM DETAILS\_USED\_SERVICE))  PRINT N'Đã cập nhật phiếu sử dụng dịch vụ cho phòng ' + RTRIM(@Room\_id) + N' thành công'  END |

***26. Trigger cập nhật khách hàng chính thức khi cập nhật khách hàng***

|  |
| --- |
| CREATE TRIGGER Update\_Official\_Customer  ON CUSTOMER  AFTER UPDATE  AS  BEGIN  UPDATE OFFICIAL\_CUSTOMER  SET OFFICIAL\_CUSTOMER.Customer\_name=inserted.Customer\_name,  OFFICIAL\_CUSTOMER.Gender=inserted.Gender,  OFFICIAL\_CUSTOMER.Birthday=inserted.Birthday,  OFFICIAL\_CUSTOMER.Identify\_card=inserted.Identify\_card,  OFFICIAL\_CUSTOMER.Phone=inserted.Phone,  OFFICIAL\_CUSTOMER.Mail=inserted.Mail,  OFFICIAL\_CUSTOMER.Customer\_address=inserted.Customer\_address  FROM inserted  WHERE Official\_customer\_id=inserted.Customer\_id;  END; |

***27. Trigger thêm khách hàng chính thức khi khách hàng đã nhận phòng***

|  |
| --- |
| CREATE TRIGGER Insert\_Official\_Customer  ON DETAILS\_RESERVED  AFTER INSERT, UPDATE  AS  BEGIN  INSERT INTO OFFICIAL\_CUSTOMER (Customer\_id, Customer\_name, Gender, Birthday, Identify\_card, Phone, Mail, Customer\_address)  SELECT CUSTOMER.Customer\_id, Customer\_name, Gender, Birthday, Identify\_card, Phone, Mail, Customer\_address  FROM inserted JOIN CUSTOMER on inserted.Customer\_id=CUSTOMER.Customer\_id  WHERE check\_room\_received = 1;  END; |

***28. Trigger cập nhật tiền vào hóa đơn khi cập nhật giảm giá dịch vụ***

|  |
| --- |
| create trigger Update\_Bill\_When\_Change\_Discount\_Service on SERVICE\_ROOM  after update  as  begin  declare @service\_id varchar(20), @dis\_new float, @dis\_old float, @price money, @date\_change DATE;  select @service\_id = Service\_room\_id, @price = Unit, @date\_change = convert(date, GETDATE()), @dis\_new = Discount\_service from inserted;  select @dis\_old = Discount\_service from deleted  --Chọn ra mã chi tiết hóa đơn, số lượng và số ngày đã thay đổi khi cập nhật giảm giá mới của dịch vụ đó  --chọn ra mã chi tiết hóa đơn, mã hóa đơn trong chi tiết hóa đơn của những hóa đơn chưa thanh toán và kết với bảng trên  --Lấy tiền dịch vụ sau khi thay đổi và mã hóa đơn rồi thực hiện cập nhật lại những hóa đơn nào có mã trùng với mã hóa đơn đó  update BILL  set Total\_money = Total\_money + money\_service\_changed\_dis  from BILL join (select sum(Number\_of\_service \* (@dis\_old - @dis\_new) \* @price) as money\_service\_changed\_dis, Bill\_id  from (select P.Details\_bill\_id, Bill\_id, Number\_of\_service, day\_changed  from (select DETAILS\_BILL.Details\_bill\_id, BILL.Bill\_id  from DETAILS\_BILL join BILL on DETAILS\_BILL.Bill\_id = BILL.Bill\_id  where BILL.Pay\_time is not null) as P  join (select dus.Details\_bill\_id, Number\_of\_service, DATEDIFF(day, convert(date, Date\_used), @date\_change) as day\_changed  from SERVICE\_ROOM sr join DETAILS\_USED\_SERVICE dus on sr.Service\_room\_id = dus.Service\_room\_id  where sr.Service\_room\_id = @service\_id) as Q  on P.Details\_bill\_id = Q.Details\_bill\_id) as R  where day\_changed >= 0  group by Bill\_id) as S  on BILL.Bill\_id = S.Bill\_id  where BILL.Bill\_id = S.Bill\_id  end |

***29. Trigger cập nhật tiền phòng vào hóa đơn khi cập nhật giảm giá phòng***

|  |
| --- |
| create trigger Update\_Bill\_When\_Change\_Discount\_Room on TYPE\_ROOM  after update  as  begin  declare @type\_room\_id varchar(20), @dis\_new float, @dis\_old float, @price money;  select @type\_room\_id = Type\_room\_id, @dis\_new = Discount\_room, @price = Unit from inserted;  select @dis\_old = Discount\_room from deleted;  update BILL  SET Total\_money = Total\_money + money\_dis\_room\_changed  from BILL join ( select sum(changed\_day \* (@dis\_old - @dis\_new) \* @price) as money\_dis\_room\_changed, Bill\_id --Tính tiền giảm giá mới  from DETAILS\_BILL join (select Details\_bill\_id, Reserved\_day - DATEDIFF(day, convert(date, Date\_check\_in), convert(date, GETDATE())) as changed\_day  from DETAILS\_RESERVED --Lấy ra mã chi tiết hóa đơn, số ngày thay đổi giảm giá của những phòng đang đặt thuộc loại phòng mà bị thay đổi giảm giá  where Room\_id in (select Room\_id --Lấy ra những mã phòng thuộc loại phòng mà bị thay đổi giảm giá  from ROOM  where Room\_type = @type\_room\_id)  and (Reserved\_day >= DATEDIFF(day, convert(date, Date\_check\_in), convert(date, GETDATE()))) ) as Q  on DETAILS\_BILL.Details\_bill\_id = Q.Details\_bill\_id  group by DETAILS\_BILL.Bill\_id ) as P  on BILL.Bill\_id = P.Bill\_id  where BILL.Bill\_id = P.Bill\_id  end |

## **2.7. Các VIEW**

### ***2.7.1 Các VIEW về Khách hàng***

***1. View chứa thông tin danh sách các khách hàng***

|  |
| --- |
| CREATE VIEW view\_CUSTOMER as SELECT \* FROM CUSTOMER |

Triển khai trên C#: Thực hiện gọi hàm trên lớp giao diện GUI

|  |
| --- |
| dtKhachHangChinhThuc = dbKhachHangChinhThuc.LayKhachHangChinhThuc().Tables[0];  CUSdataGridView.DataSource = dtKhachHangChinhThuc; |

Triển khai trên C#: Thực hiện gọi hàm trên lớp truy cập dữ liệu DB.KhachHang

|  |
| --- |
| public DataSet LayKhachHang()  {  return db.ExcuteQueryDataSet("select \* from view\_CUSTOMER", CommandType.Text, null);  } |

Hàm ExcuteQueryDataSet trên lớp truy xuất cơ sở dữ liệu DAL C# dung để trả về một tập dữ liệu bảng

|  |
| --- |
| public DataSet ExcuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p)  {  if (conn.State == ConnectionState.Open)  {  conn.Close();  }  conn.Open();  cmd.CommandText = strSQL;  cmd.CommandType = ct;  da = new SqlDataAdapter(cmd);  DataSet ds = new DataSet();  da.Fill(ds);  return ds;  } |

***2. View chứa thông tin danh sách các khách hàng chính thức***

|  |
| --- |
| CREATE VIEW view\_OFFICIAL\_CUSTOMER as SELECT \* FROM OFFICIAL\_CUSTOMER |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayKhachHangChinhThuc()  {  return db.ExcuteQueryDataSet("select \* from view\_OFFICIAL\_CUSTOMER", CommandType.Text, null);  } |

***4. View chứa thông tin khác hàng đã check in***

|  |
| --- |
| --View chứa thông tin khác hàng đã check in thì trong chi tiết đặt phòng giá trị Check\_room\_received được ghi nhận mang giá trị 1  CREATE VIEW CHECKED\_CUSTOMER AS  SELECT \*  FROM Customer  WHERE Customer.customer\_id IN (SELECT  DISTINCT customer\_id    FROM DETAILS\_RESERVED    WHERE Check\_room\_received ='1'); |

***5. View chứa thông tin khác hàng chưa check in***

|  |
| --- |
| --View chứa thông tin khác hàng chưa check in thì trong chi tiết đặt phòng giá trị Check\_room\_received được ghi nhận mang giá trị 0  CREATE VIEW UNCHECKED\_CUSTOMER AS  SELECT \*  FROM Customer  WHERE Customer.customer\_id IN (SELECT  DISTINCT customer\_id    FROM DETAILS\_RESERVED    WHERE Check\_room\_received = '0'); |

***6. View chứa thông tin về khách hàng đã thanh toán hóa đơn***

|  |
| --- |
| -- View chứa thông tin về khách hàng đã thanh toán hóa đơn, ở Bill giá trị Pay\_time là khác Null  CREATE VIEW PAID\_CUSTOMER AS  SELECT \*  FROM Customer  WHERE Customer.customer\_id IN (Select  DISTINCT customer\_id  From  DETAILS\_RESERVED  JOIN DETAILS\_BILL ON DETAILS\_RESERVED.Details\_bill\_id = DETAILS\_BILL.Details\_bill\_id  JOIN Bill ON Bill.Bill\_id = DETAILS\_BILL.Bill\_id  WHERE Bill.Pay\_time IS NOT NULL  ) |

***7. View chứa thông tin về khách hàng chưa thanh toán hóa đơn***

|  |
| --- |
| -- View chứa thông tin về khách hàng chưa thanh toán hóa đơn, ở Bill giá trị Pay\_time là Null  CREATE VIEW UNPAID\_CUSTOMER AS  SELECT \*  FROM Customer  WHERE Customer.customer\_id IN (Select  DISTINCT customer\_id  From  DETAILS\_RESERVED  JOIN DETAILS\_BILL ON DETAILS\_RESERVED.Details\_bill\_id = DETAILS\_BILL.Details\_bill\_id  JOIN Bill ON Bill.Bill\_id = DETAILS\_BILL.Bill\_id  WHERE Bill.Pay\_time IS NULL  ) |

***8. View chứa danh sách hoạt động trên khách hàng***

|  |
| --- |
| CREATE VIEW view\_TRACKING\_LOG as SELECT \* FROM TRACKING\_LOG |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayTrackingLog()  {  return db.ExcuteQueryDataSet("select \* from view\_TRACKING\_LOG", CommandType.Text, null);  } |

### ***2.7.2 Các VIEW về Doanh thu***

***1. View Doanh thu theo tháng – năm***

|  |
| --- |
| create view view\_SHOW\_REVENUE\_GROUPBY\_MONTH\_YEAR  as  (  select YEAR(Pay\_time) as [Năm], MONTH(Pay\_time) as [Tháng], sum(Total\_money) as [Doanh thu]  from view\_BILL  where Pay\_time is not null  group by YEAR(Pay\_time), MONTH(Pay\_time)  ) |

### ***2.7.3 Các VIEW về Phòng và Loại phòng***

***1. View danh sách phòng***

|  |
| --- |
| create view view\_ROOM as select \* from ROOM |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayPhong()  {  return db.ExcuteQueryDataSet("Select \* From view\_ROOM", CommandType.Text, null);  } |

***2. View danh sách loại phòng***

|  |
| --- |
| create view view\_TYPE\_ROOM as select \* from TYPE\_ROOM |

Triển khai trên C#:

|  |
| --- |
| public List<TypeRoom> GetAll()  {  return db.GetListModels<TypeRoom>(reader => new TypeRoom(reader), "Select \* From view\_TYPE\_ROOM", CommandType.Text);  } |

***3. View danh sách phòng theo loại phòng***

|  |
| --- |
| select \* from view\_Room\_With\_TypeName |

Triển khai trên C#:

|  |
| --- |
| public TypeRoom SearchItemWithName(string name)  {  return db.GetSingleObject<TypeRoom>(reader => new TypeRoom(reader), $"Select \* From view\_TYPE\_ROOM Where Type\_room\_name = '{name}'", CommandType.Text);  } |

### ***2.7.4 Các VIEW về đặt phòng***

***1. View danh sách phiếu đặt phòng***

|  |
| --- |
| CREATE VIEW view\_DETAILS\_RESERVED as SELECT \* FROM DETAILS\_RESERVED |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayChiTietDatPhong()  {  return db.ExcuteQueryDataSet("select \* from view\_DETAILS\_RESERVED", CommandType.Text, null);  } |

***2. View danh sách phiếu đặt phòng đã check in***

|  |
| --- |
| CREATE VIEW view\_DETAILS\_RESERVED\_CHECKED\_IN as SELECT \* FROM view\_DETAILS\_RESERVED WHERE (Check\_room\_received = 1) |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayChiTietDatPhongDaNhanPhong()  {  return db.ExcuteQueryDataSet("select \* from view\_DETAILS\_RESERVED\_CHECKED\_IN", CommandType.Text, null);  } |

***3. View danh sách phiếu đặt phòng chưa check in***

|  |
| --- |
| CREATE VIEW view\_DETAILS\_RESERVED\_NOT\_CHECK\_IN as SELECT \* FROM view\_DETAILS\_RESERVED WHERE (Check\_room\_received = 0) |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayChiTietDatPhongChuaNhanPhong()  {  return db.ExcuteQueryDataSet("select \* from view\_DETAILS\_RESERVED\_NOT\_CHECK\_IN", CommandType.Text, null);  } |

***4. View danh sách phiếu đặt phòng đã đặt cọc***

|  |
| --- |
| CREATE VIEW view\_DETAILS\_RESERVED\_PAID\_DEPOSIT as SELECT \* FROM view\_DETAILS\_RESERVED WHERE (Check\_paid\_deposit = 1) |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayChiTietDatPhongDaDatCoc()  {  return db.ExcuteQueryDataSet("select \* from view\_DETAILS\_RESERVED\_PAID\_DEPOSIT", CommandType.Text, null);  } |

***5. View danh sách phiếu đặt phòng chưa đặt cọc***

|  |
| --- |
| CREATE VIEW view\_DETAILS\_RESERVED\_UNPAID\_DEPOSIT as SELECT \* FROM view\_DETAILS\_RESERVED WHERE (Check\_paid\_deposit = 0) |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayChiTietDatPhongChuaDatCoc()  {  return db.ExcuteQueryDataSet("select \* from view\_DETAILS\_RESERVED\_UNPAID\_DEPOSIT", CommandType.Text, null);  } |

### ***2.7.5 Các VIEW về hóa đơn và chi tiết hóa đơn***

***1. View danh sách hóa đơn***

|  |
| --- |
| CREATE VIEW view\_BILL as SELECT \* FROM BILL |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayHoaDon()  {  return db.ExcuteQueryDataSet("select \* from view\_BILL", CommandType.Text, null);  } |

***2. View danh sách chi tiết hóa đơn***

|  |
| --- |
| CREATE VIEW view\_DETAILS\_BILL as SELECT \* FROM DETAILS\_BILL |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayChiTietHoaDon()  {  return db.ExcuteQueryDataSet("select \* from view\_DETAILS\_BILL", CommandType.Text, null);  } |

### ***2.7.6 Các VIEW về dịch vụ và sử dụng dịch vụ***

***1. View danh sách dịch vụ***

|  |
| --- |
| create view view\_SERVICE\_ROOM as select \* from SERVICE\_ROOM |

Triển khai trên C#:

|  |
| --- |
| public List<ServiceRoom> GetAll()  {  return db.GetListModels<ServiceRoom>(reader => new ServiceRoom(reader), "Select \* From view\_SERVICE\_ROOM", CommandType.Text);  } |

***2. View danh sách chi tiết sử dụng dịch vụ***

|  |
| --- |
| create view view\_DETAILS\_USED\_SERVICE as select \* from DETAILS\_USED\_SERVICE |

Triển khai trên C#:

|  |
| --- |
| public List<DetailsUsedService> GetALL()  {  return db.GetListModels<DetailsUsedService>(reader => new DetailsUsedService(reader), "Select \* From view\_DETAILS\_USED\_SERVICE", CommandType.Text);  } |

***3. View danh sách chi tiết sử dụng dịch vụ cho các phòng***

|  |
| --- |
| create view view\_DETAILS\_USED\_SERVICE\_BY\_ROOM  as  (  select dus.Room\_id, dus.Details\_bill\_id, db.Bill\_id, sr.Service\_room\_name, sr.Unit, dus.Number\_of\_service, dus.Date\_used  from DETAILS\_USED\_SERVICE as dus  join DETAILS\_BILL as db on dus.Details\_bill\_id = db.Details\_bill\_id  join DETAILS\_RESERVED dr on db.Details\_bill\_id = dr.Details\_bill\_id  join SERVICE\_ROOM sr on dus.Service\_room\_id = sr.Service\_room\_id  ) |

### ***2.7.6 Các VIEW về nhân viên***

***1. View danh sách nhân viên***

|  |
| --- |
| CREATE VIEW view\_EMPLOYEE as SELECT \* FROM EMPLOYEE |

Triển khai trên C#:

|  |
| --- |
| public DataSet LayNhanVien()  {  return db.ExcuteQueryDataSet("select \* from view\_EMPLOYEE", CommandType.Text, null);  } |

# **CHƯƠNG 3: THIẾT KẾ CHỨC NĂNG**

## ***3.1. Kết nối cơ sở dữ liệu***

Tạo constructor trong lớp truy cập dữ liệu:

|  |
| --- |
| public DAL(string user\_id, string password)  {  // Tạo chuỗi kết nối với user id và password  string ConnectionString = $"Data Source=DESKTOP-P25M6CJ\\SQLEXPRESS;Initial Catalog=QLDATPHONGKS;User ID=" + user\_id + ";Password=" + password;  // Tạo đối tượng SqlConnection với chuỗi kết nối  conn = new SqlConnection(ConnectionString);  cmd = conn.CreateCommand();  } |

Khởi tạo 1 class kết nối ở lớp xử lý logic dữ liệu (DBCurrentLogin) để trả về 1 đối tượng đang đăng nhập có chứa UserID và Password cho chuỗi kết nối

|  |
| --- |
| public class DBCurrentLogin  {  private static DBCurrentLogin LoginInfo = null;  public static DBCurrentLogin GetCurrentLoginInfo()  {  if (LoginInfo == null)  {  LoginInfo = new DBCurrentLogin();  }  return LoginInfo;  }  public string UserName { get; set; }  public string Password { get; set; }  } |

Các class khác trong lớp xử lý logic dữ liệu sẽ gọi các thuộc tính của DBCurrentLogin và thực hiện khởi tạo DAL. Lấy minh họa class DBAccount các class khác thực hiện tương tự

|  |
| --- |
| public class DBAccount  {  DAL db = null;  public DBAccount()  {  db = new DAL(DBCurrentLogin.GetCurrentLoginInfo().UserName,  DBCurrentLogin.GetCurrentLoginInfo().Password);  }  } |

Khi đó trong FormLogin đăng nhập, gọi ra DBCurrentLogin và truyền giá trị cho thuộc tính của nó từ các textbox chứa username và password, khi đó class DBCurrentLogin đã có giá trị nên các class còn lại trong lớp xử lý logic dữ liệu sẽ có được các thuộc tính và thực hiên khởi tạo DAL thành công.

|  |
| --- |
| DBCurrentLogin currentLogin =  DBCurrentLogin.GetCurrentLoginInfo();  currentLogin.UserName = usernameTextBox.Text;  currentLogin.Password = passwordTextBox.Text; |

## ***3.2. Danh sách các Stored Procedure***

### ***3.2.1 Nhóm các Stored Procedure thêm thông tin***

***1. Thêm thông tin khách hàng***

Triển khai trên SQL:

|  |
| --- |
| CREATE PROCEDURE ADD\_CUSTOMER  @Customer\_id varchar(20),  @Customer\_name nvarchar(50),  @Gender nvarchar(6),  @Birthday date,  @Identify\_card varchar(20),  @Phone varchar(10),  @Mail varchar(50),  @Customer\_address nvarchar(255)  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  INSERT INTO CUSTOMER(Customer\_id,Customer\_name,Gender,Birthday,Identify\_card,Phone,Mail,Customer\_address)  VALUES (@Customer\_id,@Customer\_name,@Gender,@Birthday,@Identify\_card,@Phone,@Mail,@Customer\_address)  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÁCH HÀNG ĐÃ TỒN TẠI', 25, 1)  END CATCH  END |

Triển khai trên C#: Thực hiện gọi hàm trên lớp giao diện GUI

|  |
| --- |
| bool f = dbKhachHang.Add\_Customer(ref err, IDCUStextBox.Text, NametextBox.Text, gender, birthday, cccdTextBox.Text, phonetextBox.Text, mailTextBox.Text, addresstextBox.Text); |

Triển khai trên C#: Thực hiện gọi hàm trên lớp xử lý logic dữ liệu DB.KhachHang

|  |
| --- |
| public bool Add\_Customer(ref string err, string Customer\_id, string Customer\_name, string Gender, DateTime Birthday, string Identify\_card, string phone, string mail, string Customer\_address)  {  return db.MyExecuteNonQuery("ADD\_CUSTOMER",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Customer\_id", Customer\_id),  new SqlParameter("@Customer\_name", Customer\_name),  new SqlParameter("@Gender", Gender),  new SqlParameter("@Birthday", Birthday),  new SqlParameter("@Identify\_card", Identify\_card),  new SqlParameter("@Phone", phone),  new SqlParameter("@mail", mail),  new SqlParameter("@Customer\_address", Customer\_address)  );  } |

Hàm MyExecuteNonQuery trên lớp truy xuất cơ sở dữ liệu DAL C# dùng để thực thi những truy vấn không trả về giá trị như: Insert, delete, update, stored procedure

|  |
| --- |
| public bool MyExecuteNonQuery(string strSQL, CommandType ct, ref string error, params SqlParameter[] param)  {  bool f = false;  try  {  if (conn.State == ConnectionState.Open)  {  conn.Close();  }  conn.Open();  cmd.Parameters.Clear();  cmd.CommandType = ct;  cmd.CommandText = strSQL;  foreach (SqlParameter p in param)  {  cmd.Parameters.Add(p);  }  cmd.ExecuteNonQuery();  f = true;  }  catch (SqlException ex)  {  error = ex.Message;  }  finally  {  conn.Close();  }  return f;  } |

***(Các Procedure phía dưới nếu có xài*** *db.MyExecuteNonQuery thì tương tự)*

***2. Thêm thông tin nhân viên***

|  |
| --- |
| CREATE PROCEDURE ADD\_EMPLOYEE  @Employee\_id varchar(20),  @Employee\_name nvarchar(50),  @Gender nvarchar(6),  @Birthday date,  @Identify\_card varchar(20),  @Phone varchar(10),  @Mail varchar(50),  @Employee\_address nvarchar(255)  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  INSERT INTO EMPLOYEE(Employee\_id,Employee\_name,Gender,Birthday,Identify\_card,Phone,Mail,Employee\_address)  VALUES (@Employee\_id,@Employee\_name,@Gender,@Birthday,@Identify\_card,@Phone,@Mail,@Employee\_address)  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÔNG THÊM ĐƯỢC NHÂN VIÊN', 25, 1)  END CATCH  END |

Triển khai trên C#:

|  |
| --- |
| public bool Add\_Employee(ref string err, string Employee\_id, string Employee\_name, string Gender, DateTime Birthday, string Identify\_card, string phone, string mail, string Employee\_address)  {  return db.MyExecuteNonQuery("ADD\_EMPLOYEE",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Employee\_id", Employee\_id),  new SqlParameter("@Employee\_name", Employee\_name),  new SqlParameter("@Gender", Gender),  new SqlParameter("@Birthday", Birthday),  new SqlParameter("@Identify\_card", Identify\_card),  new SqlParameter("@Phone", phone),  new SqlParameter("@mail", mail),  new SqlParameter("@Employee\_address", Employee\_address)  );  } |

***3. Thêm thông tin phòng***

|  |
| --- |
| CREATE PROCEDURE Add\_Room @id varchar(20) , @status bit, @type varchar(20)  AS  BEGIN  BEGIN TRAN  BEGIN TRY  Insert Into ROOM (room\_id, Room\_status, Room\_type) Values (@id, @status, @type);  COMMIT TRAN;  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÔNG THÊM PHÒNG ĐƯỢC!', 16, 1);  END CATCH  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Add(Room room, ref string err)  {  return db.MyExecuteNonQuery("Add\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", room.ID),  new SqlParameter("@status", room.RoomTypeStatus),  new SqlParameter("@type", room.RoomTypeID));  } |

***4. Thêm thông tin loại phòng***

|  |
| --- |
| CREATE PROC Add\_Type\_Room @id varchar(20), @name nvarchar(50), @unit money, @discount float  AS  BEGIN  begin tran  begin try  INSERT INTO TYPE\_ROOM (Type\_room\_id, Type\_room\_name, Unit, Discount\_room)  VALUES (@id, @name, @unit, @discount)  commit tran  end try  begin catch  rollback  raiserror('THÊM LOẠI PHÒNG MỚI KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Add(TypeRoom typeRoom, ref string err)  {  return db.MyExecuteNonQuery("Add\_Type\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", typeRoom.ID),  new SqlParameter("@name", typeRoom.Name),  new SqlParameter("@unit", typeRoom.Unit),  new SqlParameter("@discount", typeRoom.Discount));  } |

***5. Thêm thông tin đặt phòng***

|  |
| --- |
| CREATE PROCEDURE spAdd\_detail\_reserved  @customer\_id varchar(20), @room\_id varchar(20), @detail\_bill\_id varchar(20), @reserved\_day int,  @date\_checkin datetime, @check\_room\_received bit, @check\_paid\_deposit bit, @date\_create datetime  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  INSERT INTO DETAILS\_RESERVED (Customer\_id, Room\_id, Details\_bill\_id, Reserved\_day, Date\_check\_in, Check\_room\_received, Check\_paid\_deposit, Date\_create)  VALUES (@customer\_id, @room\_id, @detail\_bill\_id, @reserved\_day, @date\_checkin, @check\_room\_received, @check\_paid\_deposit, @date\_create)  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÔNG THÊM ĐƯỢC VÀO BẢNG CHI TIẾT ĐẶT PHÒNG!', 16, 1)  END CATCH  END; |

Triển khai trên C#:

|  |
| --- |
| public bool ThemChiTietDatPhong(ref string err, string Customer\_id,  string Room\_id, string Detail\_bill\_id, string Reserved\_day, DateTime Date\_checkin,  bool Check\_room\_received, bool Check\_paid\_deposit, DateTime Date\_create)  {  return db.MyExecuteNonQuery("spAdd\_detail\_reserved",  CommandType.StoredProcedure, ref err,  new SqlParameter("@customer\_id", Customer\_id),  new SqlParameter("@room\_id", Room\_id),  new SqlParameter("@detail\_bill\_id", Detail\_bill\_id),  new SqlParameter("@reserved\_day", Reserved\_day),  new SqlParameter("@date\_checkin", Date\_checkin),  new SqlParameter("@check\_room\_received", Check\_room\_received),  new SqlParameter("@check\_paid\_deposit", Check\_paid\_deposit),  new SqlParameter("@date\_create", Date\_create));  } |

***6. Thêm thông tin hóa đơn***

|  |
| --- |
| CREATE PROCEDURE spAdd\_bill @bill\_id varchar(20)  AS  BEGIN  -- Bắt đầu một giao dịch  BEGIN TRANSACTION;  -- Kiểm tra xem mã hóa đơn đã tồn tại trong bảng invoice chưa  IF EXISTS  (  -- Lấy bản ghi có mã hóa đơn trùng với tham số đầu vào  SELECT 1  FROM BILL  WHERE Bill\_id = @bill\_id  )  BEGIN  -- Nếu đã tồn tại thì hủy giao dịch và in ra thông báo lỗi  ROLLBACK TRANSACTION;  RAISERROR ('MÃ HÓA ĐƠN ĐÃ TỒN TẠI! VUI LÒNG CHỌN MÃ KHÁC.', 16, 1)  END  ELSE  BEGIN  -- Nếu chưa tồn tại thì thêm vào bảng invoice một bản ghi mới chỉ với mã hóa đơn  INSERT INTO BILL(Bill\_id) VALUES (@bill\_id);  -- Kết thúc giao dịch và in ra kết quả  COMMIT TRANSACTION;  END  END |

Triển khai trên C#:

|  |
| --- |
| public bool Add\_Bill(ref string err, string Bill\_id)  {  return db.MyExecuteNonQuery("spAdd\_bill",  CommandType.StoredProcedure, ref err,  new SqlParameter("@bill\_id", Bill\_id));    } |

***7. Thêm thông tin chi tiết hóa đơn***

|  |
| --- |
| CREATE PROCEDURE spAdd\_detail\_bill @detail\_bill\_id varchar(20), @bill\_id varchar(20)  AS  BEGIN  -- Bắt đầu một giao dịch  BEGIN TRANSACTION;  -- Kiểm tra xem mã hóa đơn đã tồn tại trong bảng BILL chưa  IF EXISTS  (  -- Lấy bản ghi có mã hóa đơn trùng với tham số đầu vào  SELECT 1  FROM BILL  WHERE Bill\_id = @bill\_id  )  BEGIN  -- Nếu đã tồn tại thì kiểm tra xem mã chi tiết đơn đã tồn tại trong bảng invoice\_detail chưa  IF EXISTS  (  -- Lấy bản ghi có mã chi tiết đơn trùng với tham số đầu vào  SELECT 1  FROM DETAILS\_BILL  WHERE Details\_bill\_id = @detail\_bill\_id  )  BEGIN  -- Nếu đã tồn tại thì hủy giao dịch và in ra thông báo lỗi  ROLLBACK TRANSACTION;  RAISERROR ('MÃ CHI TIẾT HÓA ĐƠN ĐÃ TỒN TẠI! VUI LÒNG CHỌN MÃ KHÁC.', 16, 1)  END  ELSE  BEGIN  -- Nếu chưa tồn tại thì thêm vào bảng DETAILS\_BILL một bản ghi mới với mã chi tiết đơn và mã hóa đơn  INSERT INTO DETAILS\_BILL(Details\_bill\_id, Bill\_id) VALUES (@detail\_bill\_id, @bill\_id);  -- Kết thúc giao dịch và in ra kết quả  COMMIT TRANSACTION;  END;  END  ELSE  BEGIN  -- Nếu không tồn tại thì hủy giao dịch và in ra thông báo lỗi  ROLLBACK TRANSACTION;  RAISERROR ('MÃ HÓA ĐƠN KHÔNG TỒN TẠI! VUI LÒNG CHỌN MÃ HỢP LỆ.', 16, 1)  END  END |

Triển khai trên C#:

|  |
| --- |
| public bool ThemChiTietHoaDon(ref string err, string Detail\_bill\_id, string Bill\_id)  {  return db.MyExecuteNonQuery("spAdd\_detail\_bill",  CommandType.StoredProcedure, ref err,  new SqlParameter("@detail\_bill\_id", Detail\_bill\_id),  new SqlParameter("@bill\_id", Bill\_id));  } |

***8. Thêm thông tin dịch vụ***

|  |
| --- |
| CREATE PROC Add\_Service\_Room @id varchar(20), @name nvarchar(50), @unit money, @discount float  AS  BEGIN  begin tran  begin try  INSERT INTO SERVICE\_ROOM (Service\_room\_id, Service\_room\_name, Unit, Discount\_service) VALUES (@id, @name, @unit, @discount)  commit tran  end try  begin catch  rollback  raiserror('THÊM DỊCH VỤ MỚI KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Add(ServiceRoom serviceRoom, ref string err)  {  return db.MyExecuteNonQuery("Add\_Service\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", serviceRoom.ID),  new SqlParameter("@name", serviceRoom.Name),  new SqlParameter("@unit", serviceRoom.Unit),  new SqlParameter("@discount", serviceRoom.Discount));  } |

***9. Thêm thông tin chi tiết dịch vụ***

|  |
| --- |
| CREATE PROC Add\_Details\_Used\_Service  @room\_id varchar(20), @details\_bill\_id varchar(20),  @service\_room\_id varchar(20),  @number int, @date\_used datetime  AS  BEGIN  begin tran  begin try  INSERT INTO DETAILS\_USED\_SERVICE (Room\_id, Details\_bill\_id, Service\_room\_id, Number\_of\_service, Date\_used)  VALUES (@room\_id, @details\_bill\_id, @service\_room\_id, @number, @date\_used)  commit tran  end try  begin catch  rollback tran  raiserror('THÊM CHI TIẾT SỬ DỤNG DỊCH VỤ KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Add(DetailsUsedService detailsUsedService, ref string err)  {  return db.MyExecuteNonQuery("Add\_Details\_Used\_Service", CommandType.StoredProcedure, ref err,  new SqlParameter("@room\_id", detailsUsedService.RoomID), new SqlParameter("@details\_bill\_id", detailsUsedService.DetailsBillID),  new SqlParameter("@service\_room\_id", detailsUsedService.ServiceRoomID), new SqlParameter("@number", detailsUsedService.Number),  new SqlParameter("@date\_used", detailsUsedService.DateUsed));  } |

### ***3.2.2 Nhóm các Stored Procedure xóa thông tin***

***1. Xóa thông tin khách hàng***

|  |
| --- |
| CREATE PROCEDURE DELETE\_CUSTOMER  @Customer\_id varchar(20)  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  DELETE FROM CUSTOMER WHERE Customer\_id=@Customer\_id  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÔNG XÓA ĐƯỢC KHÁCH HÀNG', 25, 1)  END CATCH  END |

Triển khai trên C#:

|  |
| --- |
| public bool Delete\_Customer(ref string err, string Customer\_id)  {  return db.MyExecuteNonQuery("DELETE\_CUSTOMER",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Customer\_id", Customer\_id)  );  } |

***2. Xóa thông tin nhân viên***

|  |
| --- |
| CREATE PROCEDURE DELETE\_EMPLOYEE  @Employee\_id varchar(20)  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  DELETE FROM EMPLOYEE WHERE Employee\_id = @Employee\_id  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÔNG XÓA ĐƯỢC NHÂN VIÊN', 25, 1)  END CATCH  END |

Triển khai trên C#:

|  |
| --- |
| public bool Delete\_Employee(ref string err, string Employee\_id)  {  return db.MyExecuteNonQuery("DELETE\_EMPLOYEE",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Employee\_id", Employee\_id)  );  } |

***3. Xóa thông tin phòng***

|  |
| --- |
| CREATE PROC Delete\_Room @id varchar(20)  AS  BEGIN  begin tran  begin try  Delete  From ROOM  WHERE Room\_id = @id  commit tran  end try  begin catch  rollback tran  raiserror('XÓA PHÒNG THẤT BẠI!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Delete(string id, ref string err)  {  return db.MyExecuteNonQuery("Delete\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", id));  } |

***4. Xóa thông tin loại phòng***

|  |
| --- |
| CREATE PROC Delete\_Type\_Room @id varchar(20)  AS  BEGIN  begin tran  begin try  DELETE  FROM TYPE\_ROOM  WHERE Type\_room\_id = @id  commit tran  end try  begin catch  rollback  raiserror('XÓA LOẠI PHÒNG KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Delete(string id, ref string err)  {  return db.MyExecuteNonQuery("Delete\_Type\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", id));  } |

***6. Xóa thông tin đặt phòng***

|  |
| --- |
| create procedure spDelete\_detail\_reserved  @customer\_id varchar(20), @room\_id varchar(20), @detail\_bill\_id varchar(20)  as  begin  begin transaction  begin try  delete from DETAILS\_RESERVED where Details\_bill\_id = @detail\_bill\_id and Room\_id = @room\_id and Customer\_id = @customer\_id  commit tran  end try  begin catch  rollback tran  raiserror('KHÔNG XÓA ĐƯỢC CHI TIẾT ĐẶT PHÒNG!', 16, 1)  end catch  end |

Triển khai trên C#:

|  |
| --- |
| public bool XoaChiTietDatPhong(ref string err, string Customer\_id, string Room\_id, string Detail\_bill\_id)  {  return db.MyExecuteNonQuery("spDelete\_detail\_reserved",  CommandType.StoredProcedure, ref err,  new SqlParameter("@customer\_id", Customer\_id),  new SqlParameter("@room\_id", Room\_id),  new SqlParameter("@detail\_bill\_id", Detail\_bill\_id));  } |

***6. Xóa thông tin hóa đơn***

|  |
| --- |
| create procedure spDelete\_bill  @bill\_id varchar(20)  as  begin  begin transaction  begin try  delete from BILL where Bill\_id = @bill\_id  commit tran  end try  begin catch  rollback tran  raiserror('KHÔNG XÓA ĐƯỢC HÓA ĐƠN!', 16, 1)  end catch  end |

Triển khai trên C#:

|  |
| --- |
| public bool Delete\_Bill(ref string err, string Bill\_id)  {  return db.MyExecuteNonQuery("spDelete\_bill",  CommandType.StoredProcedure, ref err,  new SqlParameter("@bill\_id", Bill\_id));  } |

***7. Xóa thông tin chi tiết hóa đơn***

|  |
| --- |
| create procedure spDelete\_detail\_bill  @detail\_bill\_id varchar(20)  as  begin  begin transaction  begin try  delete from DETAILS\_BILL where Details\_bill\_id = @detail\_bill\_id  commit tran  end try  begin catch  rollback tran  raiserror('KHÔNG XÓA ĐƯỢC CHI TIẾT HÓA ĐƠN!', 16, 1)  end catch  end |

Triển khai trên C#:

|  |
| --- |
| public bool XoaChiTietHoaDon(ref string err, string Detail\_bill\_id)  {  return db.MyExecuteNonQuery("spDelete\_detail\_bill",  CommandType.StoredProcedure, ref err,  new SqlParameter("@detail\_bill\_id", Detail\_bill\_id));  } |

***8. Xóa thông tin dịch vụ***

|  |
| --- |
| CREATE PROC Delete\_Service\_Room @id varchar(20)  AS  BEGIN  begin tran  begin try  DELETE  FROM SERVICE\_ROOM  WHERE Service\_room\_id = @id  commit tran  end try  begin catch  rollback tran  raiserror('XÓA DỊCH VỤ KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Delete(string id, ref string err)  {  return db.MyExecuteNonQuery("Delete\_Service\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", id));  } |

***9. Xóa thông tin tài khoản, login và quyền của nhân viên***

|  |
| --- |
| ---------Procedure xóa tài khoản nhân viên khi xóa nhân viên---------  alter procedure spDelete\_account  @employee\_id varchar(20)  as  begin  begin try  declare @user\_name varchar(20), @stringSQL nvarchar(200), @SPID smallint, @SQL nvarchar(1000);  select @user\_name = Username from ACCOUNT where Employee\_id = @employee\_id  SELECT @SPID = session\_id FROM sys.dm\_exec\_sessions WHERE login\_name = @user\_name  SET @SQL = 'KILL ' + CAST(@SPID as varchar(4))  EXEC (@SQL)  begin tran  set @stringSQL = 'drop user ['+@user\_name+']';  EXEC(@stringSQL);  set @stringSQL = 'drop login ['+@user\_name+']';  EXEC(@stringSQL);  commit tran  end try  begin catch  rollback tran;  raiserror('KHÔNG XÓA ĐƯỢC USER VÀ TÀI KHOẢN', 16, 1)  end catch  end |

Triển khai trên C#:

|  |
| --- |
| public bool XoaTaiKhoanVaUser(ref string err, string employee\_id)  {  return db.MyExecuteNonQuery("spDelete\_account", CommandType.StoredProcedure, ref err,  new SqlParameter("@employee\_id", employee\_id));  } |

### ***3.2.3 Nhóm các Stored Procedure sửa thông tin***

***1. Sửa thông tin khách hàng***

|  |
| --- |
| CREATE PROCEDURE UPDATE\_CUSTOMER  @Customer\_id varchar(20),  @Customer\_name nvarchar(50),  @Gender nvarchar(6),  @Birthday date,  @Identify\_card varchar(20),  @Phone varchar(10),  @Mail varchar(50),  @Customer\_address nvarchar(255)  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  UPDATE CUSTOMER  SET  Customer\_name=@Customer\_name,  Gender=@Gender,  Birthday=@Birthday,  Identify\_card=@Identify\_card,  Phone=@Phone,  Mail=@Mail,  Customer\_address=@Customer\_address  WHERE Customer\_id = @Customer\_id  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('CẬP NHẬT THÔNG TIN KHÁCH HÀNG KHÔNG THÀNH CÔNG', 25, 1)  END CATCH  END |

Triển khai trên C#:

|  |
| --- |
| public bool Update\_Customer(ref string err, string Customer\_id, string Customer\_name, string Gender, DateTime Birthday, string Identify\_card, string phone, string mail, string Customer\_address)  {  return db.MyExecuteNonQuery("UPDATE\_CUSTOMER",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Customer\_id", Customer\_id),  new SqlParameter("@Customer\_name", Customer\_name),  new SqlParameter("@Gender", Gender),  new SqlParameter("@Birthday", Birthday),  new SqlParameter("@Identify\_card", Identify\_card),  new SqlParameter("@Phone", phone),  new SqlParameter("@mail", mail),  new SqlParameter("@Customer\_address", Customer\_address)  );  } |

***2. Sửa thông tin nhân viên***

|  |
| --- |
| CREATE PROCEDURE UPDATE\_EMPLOYEE  @Employee\_id varchar(20),  @Employee\_name nvarchar(50),  @Gender nvarchar(6),  @Birthday date,  @Identify\_card varchar(20),  @Phone varchar(10),  @Mail varchar(50),  @Employee\_address nvarchar(255)  AS  BEGIN  BEGIN TRANSACTION  BEGIN TRY  UPDATE EMPLOYEE  SET  Employee\_name=@Employee\_name,  Gender=@Gender,  Birthday=@Birthday,  Identify\_card=@Identify\_card,  Phone=@Phone,  Mail=@Mail,  Employee\_address=@Employee\_address  WHERE Employee\_id = @Employee\_id  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('CẬP NHẬT THÔNG TIN NHÂN VIÊN KHÔNG THÀNH CÔNG', 25, 1)  END CATCH  END |

Triển khai trên C#:

|  |
| --- |
| public bool Update\_Employee(ref string err, string Employee\_id, string Employee\_name, string Gender, DateTime Birthday, string Identify\_card, string phone, string mail, string Employee\_address)  {  return db.MyExecuteNonQuery("UPDATE\_EMPLOYEE",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Employee\_id", Employee\_id),  new SqlParameter("@Employee\_name", Employee\_name),  new SqlParameter("@Gender", Gender),  new SqlParameter("@Birthday", Birthday),  new SqlParameter("@Identify\_card", Identify\_card),  new SqlParameter("@Phone", phone),  new SqlParameter("@mail", mail),  new SqlParameter("@Employee\_address", Employee\_address)  );  } |

***3. Sửa thông tin phòng***

|  |
| --- |
| CREATE PROC Update\_Room @id varchar(20), @type varchar(20)  AS  BEGIN  begin tran  begin try  Update Room Set Room\_type = @type  Where Room\_id = @id  commit tran  end try  begin catch  rollback tran  raiserror('KHÔNG CẬP NHẬT PHÒNG ĐƯỢC', 16, 1);  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Update(string id, string type, ref string err)  {  return db.MyExecuteNonQuery("Update\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", id),  new SqlParameter("@type", type));  } |

***4. Sửa thông tin loại phòng***

|  |
| --- |
| CREATE PROC Update\_Type\_Room @id varchar(20), @name nvarchar(50), @unit money, @discount float  AS  BEGIN  begin tran  begin try  UPDATE TYPE\_ROOM SET Type\_room\_name = @name, Unit = @unit, Discount\_room = @discount  WHERE Type\_room\_id = @id  commit tran  end try  begin catch  rollback tran  raiserror('CẬP NHẬT LOẠI PHÒNG KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Update(TypeRoom typeRoom, ref string err)  {  return db.MyExecuteNonQuery("Update\_Type\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", typeRoom.ID),  new SqlParameter("@name", typeRoom.Name),  new SqlParameter("@unit", typeRoom.Unit),  new SqlParameter("@discount", typeRoom.Discount));  } |

***6. Sửa thông tin đặt phòng***

|  |
| --- |
| create procedure spModify\_reserved\_day  @customer\_id varchar(20), @room\_id varchar(20), @detail\_bill\_id varchar(20), @reserved\_day int,  @date\_checkin datetime, @check\_room\_received bit, @check\_paid\_deposit bit  as  begin  begin transaction  begin try  update DETAILS\_RESERVED  set Reserved\_day = @reserved\_day, Date\_check\_in = @date\_checkin, Check\_room\_received = @check\_room\_received,  Check\_paid\_deposit = @check\_paid\_deposit  where Customer\_id = @customer\_id and Room\_id = @room\_id and Details\_bill\_id = @detail\_bill\_id  commit tran  end try  begin catch  rollback tran  raiserror('CẬP NHẬT CHI TIẾT ĐẶT PHÒNG KHÔNG THÀNH CÔNG!', 16, 1)  end catch  end |

Triển khai trên C#:

|  |
| --- |
| public bool ChinhSuaChiTietDatPhong(ref string err, string Customer\_id,  string Room\_id, string Detail\_bill\_id, string Reserved\_day, DateTime Date\_checkin,  bool Check\_room\_received, bool Check\_paid\_deposit)  {  return db.MyExecuteNonQuery("spModify\_reserved\_day",  CommandType.StoredProcedure, ref err,  new SqlParameter("@customer\_id", Customer\_id),  new SqlParameter("@room\_id", Room\_id),  new SqlParameter("@detail\_bill\_id", Detail\_bill\_id),  new SqlParameter("@reserved\_day", Reserved\_day),  new SqlParameter("@date\_checkin", Date\_checkin),  new SqlParameter("@check\_room\_received", Check\_room\_received),  new SqlParameter("@check\_paid\_deposit", Check\_paid\_deposit));  } |

***7. Sửa thông tin hóa đơn***

|  |
| --- |
| create procedure spUpdate\_bill  @bill\_id varchar(20), @pay\_time datetime, @employee\_id varchar(20), @payment\_method nvarchar(20)  as  begin  begin transaction  begin try  update BILL  set Pay\_time = @pay\_time, Employee\_id = @employee\_id, Payment\_method = @payment\_method  where Bill\_id = @bill\_id  commit tran  end try  begin catch  rollback tran  raiserror('KHÔNG CẬP NHẬT ĐƯỢC HÓA ĐƠN!', 16, 1)  end catch  end |

Triển khai trên C#:

|  |
| --- |
| public bool Update\_Bill(ref string err, string Bill\_id, DateTime Pay\_time,  string Employee\_id, string Payment\_method)  {  return db.MyExecuteNonQuery("spUpdate\_bill",  CommandType.StoredProcedure, ref err,  new SqlParameter("@bill\_id", Bill\_id),  new SqlParameter("@pay\_time", Pay\_time),  new SqlParameter("@employee\_id", Employee\_id),  new SqlParameter("@payment\_method", Payment\_method));  } |

***8. Sửa thông tin dịch vụ***

|  |
| --- |
| CREATE PROC Update\_Service\_Room @id varchar(20), @name nvarchar(50), @unit money, @discount float  AS  BEGIN  begin tran  begin try  UPDATE SERVICE\_ROOM SET Service\_room\_name = @name, Unit = @unit, Discount\_service = @discount  WHERE Service\_room\_id = @id  commit tran  end try  begin catch  rollback tran  raiserror('CẬP NHẬT DỊCH VỤ KHÔNG THÀNH CÔNG!', 16, 1)  end catch  END; |

Triển khai trên C#:

|  |
| --- |
| public bool Update(ServiceRoom serviceRoom, ref string err)  {  return db.MyExecuteNonQuery("Update\_Service\_Room", CommandType.StoredProcedure, ref err,  new SqlParameter("@id", serviceRoom.ID),  new SqlParameter("@name", serviceRoom.Name),  new SqlParameter("@unit", serviceRoom.Unit),  new SqlParameter("@discount", serviceRoom.Discount));  } |

## ***3.3. Danh sách các Functions***

### ***3.3.1 Nhóm các Function tìm kiếm***

***1. Tìm kiếm thông tin nhân viên***

|  |
| --- |
| CREATE FUNCTION Search\_Employee(@string nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_EMPLOYEE  WHERE ((Employee\_id = @string)  OR (Identify\_card = @string) OR (Phone LIKE '%' +@string +'%')  OR (Mail LIKE '%' +@string +'%') OR (Employee\_address LIKE '%' +@string +'%'))) |

Triển khai trên C#: Thực hiện gọi hàm trên lớp giao diện GUI

|  |
| --- |
| dtNhanVien = dbNhanVien.LocNhanVien(txtKey.Text); |

Triển khai trên C#: Thực hiện gọi hàm trên lớp truy cập dữ liệu DB.NhanVien

|  |
| --- |
| public DataTable LocNhanVien(string strKey)  {  return db.ExcuteTableFunction("Search\_Employee", CommandType.Text,  new SqlParameter("@string", strKey));  } |

Hàm ExcuteTableFunction trên lớp truy xuất cơ sở dữ liệu DAL C# dùng để thực thi 1 Function trên SQL và trả về là 1 bảng

|  |
| --- |
| public DataTable ExcuteTableFunction(string strSQL, CommandType ct, params SqlParameter[] p)  {  DataTable dt = new DataTable();  try  {  if (conn.State == ConnectionState.Open)  {  conn.Close();  }  conn.Open();  cmd.Parameters.Clear();  string commandText = string.Format("SELECT \* FROM {0}(", strSQL);  if (p != null)  {  foreach (SqlParameter param in p)  {  commandText += param.ParameterName + ",";  cmd.Parameters.Add(param);  }  commandText = commandText.TrimEnd(',') + ")";  }  else  {  commandText += ")";  }  cmd.CommandText = commandText;  cmd.CommandType = ct;  da = new SqlDataAdapter(cmd);  da.Fill(dt);  }  catch(Exception ex)  {  MessageBox.Show("Lỗi: " + ex.Message);  }  finally  {  conn.Close();  }  return dt;  } |

***2. Tìm kiếm thông tin nhân viên theo tên***

|  |
| --- |
| CREATE FUNCTION Search\_Employee\_ByName(@string nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_EMPLOYEE  WHERE ( (Employee\_name LIKE '%' +@string +'%') )) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocNVTheoTen(string name)  {  return db.ExcuteTableFunction("Search\_Employee\_ByName", CommandType.Text,  new SqlParameter("@string", name));  } |

***3. Tìm kiếm thông tin nhân viên trong khoảng ngày sinh***

|  |
| --- |
| CREATE FUNCTION Search\_Employee\_By\_Birthday(@fromdate DateTime , @todate DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_EMPLOYEE  WHERE Birthday BETWEEN @fromdate AND @todate); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocNVTheoNgaySinh(DateTime fromdate, DateTime todate)  {  return db.ExcuteTableFunction("Search\_Employee\_By\_Birthday", CommandType.Text,  new SqlParameter("@fromdate", fromdate),  new SqlParameter("@todate", todate));  } |

***4. Tìm kiếm thông tin nhân viên trong khoảng ngày sinh và giới tính***

|  |
| --- |
| CREATE FUNCTION Search\_Employee\_By\_GenderAndBirthday(@gender nvarchar(50), @fromdate DateTime , @todate DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_EMPLOYEE  WHERE Gender=@gender AND Birthday BETWEEN @fromdate AND @todate); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocNVTheoGioiTinhVaNgaySinh(string gender, DateTime fromdate, DateTime todate)  {  return db.ExcuteTableFunction("Search\_Employee\_By\_GenderAndBirthday", CommandType.Text,  new SqlParameter("@gender", gender),  new SqlParameter("@fromdate", fromdate),  new SqlParameter("@todate", todate));  } |

***5. Tìm kiếm thông tin nhân viên theo giới tính***

|  |
| --- |
| CREATE FUNCTION Search\_Employee\_By\_Gender(@gender nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_EMPLOYEE  WHERE Gender=@gender); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocNVTheoGioiTinh(string gender)  {  return db.ExcuteTableFunction("Search\_Employee\_By\_Gender", CommandType.Text,  new SqlParameter("@gender", gender));  } |

***6. Tìm kiếm thông tin khách hàng***

|  |
| --- |
| CREATE FUNCTION Search\_Customer(@string nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_CUSTOMER  WHERE ((Customer\_id = @string)  OR (Identify\_card = @string) OR (Phone LIKE '%' +@string +'%')  OR (Mail LIKE '%' +@string +'%') OR (Customer\_address LIKE '%' +@string +'%'))) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocKhachHang(string strKey)  {  return db.ExcuteTableFunction("Search\_Customer", CommandType.Text,  new SqlParameter("@string", strKey));  } |

***7. Tìm kiếm thông tin khách hàng theo tên***

|  |
| --- |
| CREATE FUNCTION Search\_Customer\_ByName(@string nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_CUSTOMER  WHERE ( (Customer\_name LIKE '%' +@string +'%') )) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocKHTheoTen(string name)  {  return db.ExcuteTableFunction("Search\_Customer\_ByName", CommandType.Text,  new SqlParameter("@string", name));  } |

***8. Tìm kiếm thông tin khách hàng trong khoảng ngày sinh***

|  |
| --- |
| CREATE FUNCTION Search\_Customer\_By\_Birthday(@fromdate DateTime , @todate DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_CUSTOMER  WHERE Birthday BETWEEN @fromdate AND @todate); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocKHTheoNgaySinh(DateTime fromdate, DateTime todate)  {  return db.ExcuteTableFunction("Search\_Customer\_By\_Birthday", CommandType.Text,  new SqlParameter("@fromdate", fromdate),  new SqlParameter("@todate", todate));  } |

***9. Tìm kiếm thông tin khách hàng trong khoảng ngày sinh và giới tính***

|  |
| --- |
| CREATE FUNCTION Search\_Customer\_By\_GenderAndBirthday(@gender nvarchar(50), @fromdate DateTime , @todate DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_CUSTOMER  WHERE Gender=@gender AND Birthday BETWEEN @fromdate AND @todate); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocKHTheoGioiTinhVaNgaySinh(string gender, DateTime fromdate, DateTime todate)  {  return db.ExcuteTableFunction("Search\_Customer\_By\_GenderAndBirthday", CommandType.Text,  new SqlParameter("@gender", gender),  new SqlParameter("@fromdate", fromdate),  new SqlParameter("@todate", todate));  } |

***10. Tìm kiếm thông tin khách hàng theo giới tính***

|  |
| --- |
| CREATE FUNCTION Search\_Customer\_By\_Gender(@gender nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_CUSTOMER  WHERE Gender=@gender) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocKHTheoGioiTinh(string gender)  {  return db.ExcuteTableFunction("Search\_Customer\_By\_Gender", CommandType.Text,  new SqlParameter("@gender", gender));  } |

***11. Tìm kiếm thông tin phòng theo mã phòng và loại phòng***

|  |
| --- |
| CREATE FUNCTION FILTER\_ROOM(  @id varchar(20),  @type varchar(20)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM view\_Room\_With\_TypeName  WHERE (Room\_id LIKE '%' + @id +'%') AND (Type\_room\_name LIKE '%' + @type +'%')  ); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocPhongDaDatTheoNgayVaLoaiPhong(DateTime date\_start, DateTime date\_end, string type\_room)  {  return db.ExcuteTableFunction("ftFilter\_room\_reserved\_by\_type", CommandType.Text,  new SqlParameter("@date\_start", date\_start),  new SqlParameter("@date\_end", date\_end),  new SqlParameter("@type\_room", type\_room));  } |

***12. Tìm kiếm thông tin loại phòng theo giá tiền***

|  |
| --- |
| CREATE FUNCTION FILTER\_TYPE\_ROOM(  @min money,  @max money  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM view\_TYPE\_ROOM  WHERE Unit >= @min AND Unit <= @max  ); |

Triển khai trên C#:

|  |
| --- |
| public List<TypeRoom> Filter(int min, int max)  {  return db.GetListModels<TypeRoom>(reader => new TypeRoom(reader), $"SELECT \* FROM FILTER\_TYPE\_ROOM ({min}, {max})", CommandType.Text);  } |

***13. Tìm kiếm thông tin hóa đơn theo khách hàng***

|  |
| --- |
| CREATE FUNCTION ftGet\_bill\_by\_customer (@customer\_id varchar(20))  RETURNS TABLE  AS  RETURN  (  SELECT DISTINCT b.\*  FROM view\_BILL b  INNER JOIN view\_DETAILS\_BILL db on b.Bill\_id = db.Bill\_id  INNER JOIN view\_DETAILS\_RESERVED dr on db.Details\_bill\_id = dr.Details\_bill\_id  WHERE dr.Customer\_id = @customer\_id  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocHoaDonTheoMaKH(string strKey)  {  return db.ExcuteTableFunction("ftGet\_bill\_by\_customer", CommandType.Text,  new SqlParameter("@customer\_id", strKey));  } |

***14. Tìm kiếm thông tin hóa đơn***

|  |
| --- |
| CREATE FUNCTION ftSearch\_bill(@string nvarchar(50))  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM view\_BILL  WHERE ((Bill\_id = @string) OR (Pay\_time LIKE '%' +@string +'%')  OR (Employee\_id = @string) OR (Payment\_method LIKE '%' +@string +'%'))  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocHoaDon(string strKey)  {    return db.ExcuteTableFunction("ftSearch\_bill", CommandType.Text,  new SqlParameter("@string", strKey));  } |

***15. Tìm kiếm thông tin hóa đơn theo ngày thanh toán***

|  |
| --- |
| CREATE FUNCTION ftGet\_bill\_by\_pay\_time (@pay\_time DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_BILL  WHERE DATEDIFF(day, convert(date, Pay\_time), convert(date, @pay\_time)) = 0  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocHoaDonTheoNgayThanhToan(DateTime Pay\_time)  {  return db.ExcuteTableFunction("ftGet\_bill\_by\_pay\_time", CommandType.Text,  new SqlParameter("@pay\_time", Pay\_time));  } |

***16. Tìm kiếm chi tiết đặt phòng***

|  |
| --- |
| CREATE FUNCTION ftGet\_detail\_reserved (@string varchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_DETAILS\_RESERVED  WHERE ((Details\_bill\_id = @string) OR (Room\_id =@string)  OR (Customer\_id=@string ))  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocChiTietDatPhong(string strKey)  {  return db.ExcuteTableFunction("ftGet\_detail\_reserved", CommandType.Text,  new SqlParameter("@string", strKey));  } |

***17. Tìm kiếm chi tiết đặt phòng theo ngày đặt phòng***

|  |
| --- |
| CREATE FUNCTION ftGet\_detail\_reserved\_date\_create (@date\_create DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_DETAILS\_RESERVED  WHERE DATEDIFF(day, convert(date, Date\_create), convert(date, @date\_create)) = 0  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocChiTietDatPhongTheoNgayDat(DateTime strKey)  {  return db.ExcuteTableFunction("ftGet\_detail\_reserved\_date\_create", CommandType.Text,  new SqlParameter("@string", strKey));  } |

***18. Tìm kiếm chi tiết đặt phòng theo ngày nhận phòng***

|  |
| --- |
| CREATE FUNCTION ftGet\_detail\_reserved\_date\_check\_in (@date\_check\_in DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_DETAILS\_RESERVED  WHERE DATEDIFF(day, convert(date, Date\_check\_in), convert(date, @date\_check\_in)) = 0  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocChiTietDatPhongTheoNgayNhanPhong(DateTime strKey)  {  return db.ExcuteTableFunction("ftGet\_detail\_reserved\_date\_check\_in", CommandType.Text,  new SqlParameter("@string", strKey));  } |

***19. Tìm kiếm danh sách Tracking Log***

|  |
| --- |
| CREATE FUNCTION Search\_Tracking(@string nvarchar(50))  RETURNS TABLE AS RETURN  (  SELECT \*  FROM TRACKING\_LOG  WHERE ((Customer\_id = @string) OR (Customer\_name LIKE '%' +@string +'%')  OR (Identify\_card = @string +'%') OR (Operation LIKE '%' +@string +'%'))); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocTracking(string strKey)  {  return db.ExcuteTableFunction("Search\_Tracking", CommandType.Text,  new SqlParameter("@string", strKey));  } |

***20. Tìm kiếm danh sách Tracking Log theo ngày cập nhật***

|  |
| --- |
| CREATE FUNCTION Search\_Tracking\_By\_Updated(@fromdate DateTime , @todate DateTime)  RETURNS TABLE AS RETURN  (  SELECT \*  FROM view\_TRACKING\_LOG  WHERE Updated\_at BETWEEN @fromdate AND @todate); |

Triển khai trên C#:

|  |
| --- |
| public DataTable LocTrackingTheoNgayCapNhat(DateTime fromdate, DateTime todate)  {  return db.ExcuteTableFunction("Search\_Tracking\_By\_Updated", CommandType.Text,  new SqlParameter("@fromdate", fromdate),  new SqlParameter("@todate", todate));  } |

***21. Tìm kiếm chi tiết sử dụng dịch vụ theo giá tiền***

|  |
| --- |
| CREATE FUNCTION FILTER\_SERVICE(  @min money,  @max money  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM view\_SERVICE\_ROOM  WHERE Unit >= @min AND Unit <= @max  ); |

Triển khai trên C#: Thực hiện gọi hàm trên lớp giao diện GUI

|  |
| --- |
| private void btnFilter\_Click(object sender, EventArgs e)  {  gvListServiceRoom.DataSource = dbDichVu.Filter(int.Parse(txtMinUnit.Text), int.Parse(txtMaxUnit.Text));  } |

Triển khai trên C#: Thực hiện gọi hàm trên lớp truy cập dữ liệu DB.NhanVien

|  |
| --- |
| public List<ServiceRoom> Filter(int min, int max)  {  return db.GetListModels<ServiceRoom>(reader => new ServiceRoom(reader), $"SELECT \* FROM FILTER\_SERVICE ({min}, {max})", CommandType.Text);  } |

Hàm GetListModels trên lớp truy xuất cơ sở dữ liệu DAL C# dùng để lấy một danh sách đối tượng Service Room

|  |
| --- |
| public List<T> GetListModels<T>(Func<SqlDataReader, T> converter, string sqlStr, CommandType commandType)  {  List<T> list = new List<T>();  try  {  if (conn.State == ConnectionState.Open)  {  conn.Close();  }  conn.Open();  SqlCommand cmd = conn.CreateCommand();  cmd.CommandType = commandType;  cmd.CommandText = sqlStr;  SqlDataReader reader = cmd.ExecuteReader();  while (reader.Read())  list.Add(converter(reader));  cmd.Dispose();  reader.Close();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  finally  {  conn.Close();  }  return list;  } |

***22. Tìm kiếm chi tiết sử dụng dịch vụ***

|  |
| --- |
| CREATE FUNCTION FILTER\_DETAILS\_USED\_SERVICE(  @room\_id varchar(20), @details\_bill\_id varchar(20),  @service\_room\_id varchar(20),  @number\_min int, @number\_max int, @date\_used\_start datetime, @date\_used\_end datetime  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM view\_DETAILS\_USED\_SERVICE  WHERE (Room\_id = @room\_id OR @room\_id IS NULL)  AND (Details\_bill\_id = @details\_bill\_id OR @details\_bill\_id IS NULL)  AND (Service\_room\_id = @service\_room\_id OR @service\_room\_id IS NULL)  AND ((Date\_used >= @date\_used\_start AND Date\_used<=@date\_used\_end)  OR ( @date\_used\_start IS NULL AND @date\_used\_end IS NULL))  AND ((Number\_of\_service >= @number\_min AND Number\_of\_service<=@number\_max)  OR (@number\_min IS NULL AND @number\_max IS NULL))  ); |

Triển khai trên C#:

|  |
| --- |
| public List<ServiceRoom> Filter(int min, int max)  {  return db.GetListModels<ServiceRoom>(reader => new ServiceRoom(reader), $"SELECT \* FROM FILTER\_SERVICE ({min}, {max})", CommandType.Text);  } |

### ***3.3.2 Nhóm các Function tính giá trị lớn nhất***

***1. Tìm giá tiền loại phòng cao nhất***

|  |
| --- |
| CREATE FUNCTION Max\_Unit\_Type\_Room()  RETURNS INT  AS  BEGIN  DECLARE @maxValue INT;  SELECT @maxValue = MAX(Unit)  FROM view\_TYPE\_ROOM  RETURN @maxValue;  END; |

Triển khai trên C#: Thực hiện gọi hàm trên lớp giao diện GUI

|  |
| --- |
| txtMaxUnit.Text = dbLoaiPhong.GetMaxUnit().ToString(); |

Triển khai trên C#: Thực hiện gọi hàm trên lớp truy cập dữ liệu DB.LoaiPhong

|  |
| --- |
| public int GetMaxUnit()  {  return db.ExecuteScalarFunction<int>("[dbo].[Max\_Unit\_Type\_Room]", CommandType.Text, null);  } |

Hàm ExecuteScalarFunction trên lớp truy xuất cơ sở dữ liệu DAL C# dùng để trả về một giá trị

|  |
| --- |
| public T ExecuteScalarFunction<T>(string functionName, CommandType ct, params SqlParameter[] param)  {  T result = default(T);  try  {  if (conn.State == ConnectionState.Open)  {  conn.Close();  }  conn.Open();  cmd.Parameters.Clear();  string commandText = "SELECT " + functionName + "(";  if (param != null)  {  foreach (SqlParameter p in param)  {  commandText += p.ParameterName + ",";  cmd.Parameters.Add(p);  }  commandText = commandText.TrimEnd(',') + ")";  }  else  {  commandText += ")";  }  cmd.CommandText = commandText;  cmd.CommandType = ct;  object scalarValue = cmd.ExecuteScalar();  if (scalarValue != null && scalarValue != DBNull.Value)  {  result = (T)scalarValue;  }  }  catch (Exception ex)  {  MessageBox.Show("Lỗi: " + ex);  }  finally  {  conn.Close();  }  return result;  } |

***2. Tìm giá dịch vụ cao nhất***

|  |
| --- |
| CREATE FUNCTION Max\_Unit\_Service\_Room()  RETURNS INT  AS  BEGIN  DECLARE @maxValue INT;  SELECT @maxValue = MAX(Unit)  FROM SERVICE\_ROOM  RETURN @maxValue;  END; |

Triển khai trên C#:

|  |
| --- |
| public int GetMaxUnit()  {  //return Convert.ToInt32(dbConnection.GetSingleValueFromFunction("EXECUTE Max\_Unit\_Type\_Room", null));  return db.ExecuteScalarFunction<int>("[dbo].[Max\_Unit\_Service\_Room]", CommandType.Text, null);  } |

### ***3.3.3 Nhóm các Function tạo mã tự động***

***1. Function sinh mã khách hàng tự động khi thêm mới khách hàng***

|  |
| --- |
| CREATE FUNCTION ftGet\_next\_customer\_id()  RETURNS VARCHAR(20)  AS  BEGIN  DECLARE @s VARCHAR(20);  DECLARE @max\_num INT;  SELECT TOP 1 @max\_num = CAST(SUBSTRING([Customer\_id], 5, LEN([Customer\_id])) AS INT)  FROM view\_CUSTOMER  WHERE SUBSTRING([Customer\_id], 1, 5) = 'KH000'  ORDER BY CAST(SUBSTRING([Customer\_id], 5, LEN([Customer\_id])) AS INT) DESC;  IF (@max\_num IS NULL)  BEGIN  SET @s = 'KH0001';  END  ELSE  BEGIN  SET @max\_num = @max\_num + 1;  SET @s = 'KH000' + RIGHT(CAST(@max\_num AS NVARCHAR(MAX)), LEN(CAST(@max\_num AS NVARCHAR(MAX))));  END  RETURN @s;  END; |

Triển khai trên C#:

|  |
| --- |
| public string SinhMaKhachHang()  {  return db.ExecuteScalarFunction<string>("[dbo].[ftGet\_next\_customer\_id]", CommandType.Text, null);  } |

***2. Function sinh mã nhân viên tự động khi thêm mới nhân viên***

|  |
| --- |
| CREATE FUNCTION ftGet\_next\_employee\_id()  RETURNS VARCHAR(20)  AS  BEGIN  DECLARE @s VARCHAR(20);  DECLARE @max\_num INT;  SELECT TOP 1 @max\_num = CAST(SUBSTRING([Employee\_id], 3, LEN([Employee\_id])) AS INT)  FROM view\_EMPLOYEE  ORDER BY CAST(SUBSTRING([Employee\_id], 3, LEN([Employee\_id])) AS INT) DESC;  IF (@max\_num IS NULL)  BEGIN  SET @s = 'NV01';  END  ELSE  BEGIN  SET @max\_num = @max\_num + 1;  SET @s = 'NV0' + RIGHT(CAST(@max\_num AS NVARCHAR(MAX)), LEN(CAST(@max\_num AS NVARCHAR(MAX))));  END  RETURN @s;  END; |

Triển khai trên C#:

|  |
| --- |
| public string SinhMaNhanVien()  {  return db.ExecuteScalarFunction<string>("[dbo].[ftGet\_next\_employee\_id]", CommandType.Text, null);  } |

### ***3.3.4 Nhóm các Function khác***

***1. Hiển thị dịch vụ, số lượng sử dụng dịch vụ và ngày sử dụng dịch vụ của chi tiết hóa đơn***

|  |
| --- |
| create function ftShow\_detail\_used\_service (@detail\_bill\_id varchar(20), @bill\_id varchar(20))  returns table as return  (  select Room\_id, Service\_room\_name, Unit, Number\_of\_service, Date\_used  from view\_DETAILS\_USED\_SERVICE\_BY\_ROOM  where Details\_bill\_id = @detail\_bill\_id and Bill\_id = @bill\_id  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable HienThiDichVuDaSuDung(string detail\_bill\_id, string bill\_id)  {  return db.ExcuteTableFunction("ftShow\_detail\_used\_service", CommandType.Text,  new SqlParameter("@detail\_bill\_id", detail\_bill\_id),  new SqlParameter("@bill\_id", bill\_id));  } |

***2. Hiển thị doanh thu theo tháng, theo năm***

|  |
| --- |
| create function ftShow\_revenue\_by\_month\_of\_year(@year varchar(20))  returns table as return  (  select \*  from view\_SHOW\_REVENUE\_GROUPBY\_MONTH\_YEAR  where Năm = @year  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LayDoanhThuTheoThangCuaNam(string year)  {  return db.ExcuteTableFunction("ftShow\_revenue\_by\_month\_of\_year", CommandType.Text,  new SqlParameter("@year", year));  } |

***3. Kiểm tra tài khoản và mật khẩu(Đã có thì trả về mã nhân viên)***

|  |
| --- |
| create function ftCheck\_login (@user\_name varchar(20), @pass\_word varchar(20))  returns varchar(20) as  begin  declare @result varchar(20)  set @result = '';  if exists(select \* from ACCOUNT where Username = @user\_name and Pass = @pass\_word)  begin  select @result = Employee\_id from ACCOUNT where Username = @user\_name  end  return @result  end |

Triển khai trên C#:

|  |
| --- |
| public string KiemTraDangNhap(string user\_name, string pass\_word)  {  return db.ExecuteScalarFunction<string>("[dbo].[ftCheck\_login]", CommandType.Text,  new SqlParameter("@user\_name", user\_name),  new SqlParameter("@pass\_word", pass\_word));  } |

***4.Lấy tên tài khoản, mật khẩu và quyền của 1 nhân viên(truyền vào mã nhân viên)***

|  |
| --- |
| create function ftGet\_username\_pass\_role(@employee\_id varchar(20))  returns table as return  (  select \* from ACCOUNT where Employee\_id = @employee\_id  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LayTaiKhoanVaQuyen(string employee\_id)  {  return db.ExcuteTableFunction("[dbo].[ftGet\_username\_pass\_role]", CommandType.Text,  new SqlParameter("@employee\_id", employee\_id));  } |

***5.*** ***Lấy thông tin cá nhân của 1 nhân viên khi đăng nhập***

|  |
| --- |
| create function Get\_infor\_employee(@user\_name varchar(20), @pass\_word varchar(20))  returns table as return  (  select \*  from view\_EMPLOYEE  where Employee\_id = (select Employee\_id  from ACCOUNT  where Username = @user\_name and Pass = @pass\_word)  ) |

Triển khai trên C#:

|  |
| --- |
| public DataTable LayThongTin1NhanVien()  {  return db.ExcuteTableFunction("[dbo].[Get\_infor\_employee]", CommandType.Text,  new SqlParameter("@user\_name", DBCurrentLogin.GetCurrentLoginInfo().UserName),  new SqlParameter("@pass\_word", DBCurrentLogin.GetCurrentLoginInfo().Password));  } |

## ***3.4. Phân quyền trong phần mềm***

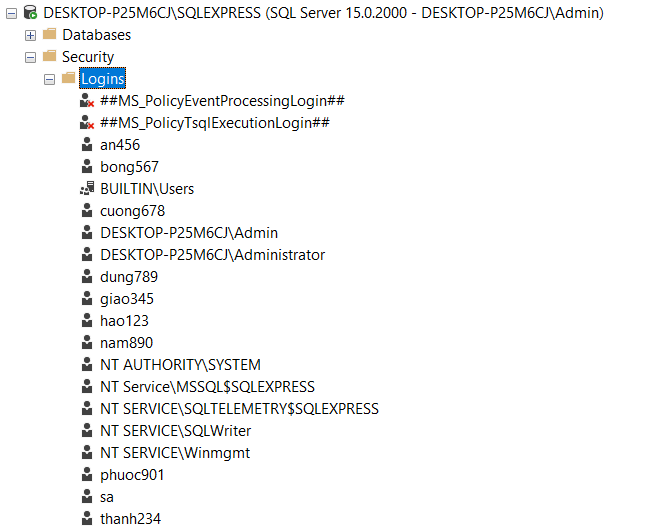
### ***3.4.1 Tạo user và roles khi thêm nhân viên***

Dùng Procedure để tự động tạo user khi thêm nhân viên vào hệ thống

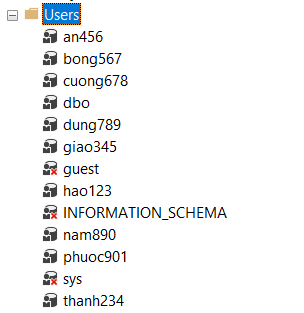
|  |
| --- |
| CREATE PROCEDURE spCreate\_account\_user\_login  @user\_name varchar(20), @pass varchar(20), @employee\_id varchar(20), @role varchar(20)  AS  BEGIN  BEGIN TRAN  BEGIN TRY  --Thêm tài khoản  INSERT INTO ACCOUNT(Username, Pass, Employee\_id, Role) values (@user\_name, @pass, @employee\_id, @role);  DECLARE @sqlString nvarchar(2000)  -- Tạo tài khoản login cho nhân viên, tên người dùng và mật khẩu là tài khoản được tạo trên bảng Account  SET @sqlString= 'CREATE LOGIN [' + @user\_name +'] WITH PASSWORD='''+ @pass +''',  DEFAULT\_DATABASE=[QLDATPHONGKS], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF'  EXEC (@sqlString)  -- Tạo tài khoản người dùng đối với nhân viên đó trên database (tên người dùng trùng với tên login)  SET @sqlString= 'CREATE USER ' + @user\_name +' FOR LOGIN '+ @user\_name  EXEC (@sqlString)  -- Thêm người dùng vào vai trò quyền tương ứng (Staff hoặc Manager(sysadmin))  IF(@role ='Sysadmin')  SET @sqlString = 'ALTER SERVER ROLE ' + @role + ' ADD MEMBER ' + @user\_name;  ELSE  SET @sqlString = 'ALTER ROLE '+ @role +' ADD MEMBER ' + @user\_name;  EXEC (@sqlString)  COMMIT TRAN  END TRY  BEGIN CATCH  ROLLBACK TRAN  RAISERROR('KHÔNG THỰC HIỆN VIỆC THÊM, TẠO USER CHO TÀI KHOẢN NHÂN VIÊN ĐƯỢC!', 16, 1)  END CATCH  END |

Các Account đã được có trong bảng Account trước đó

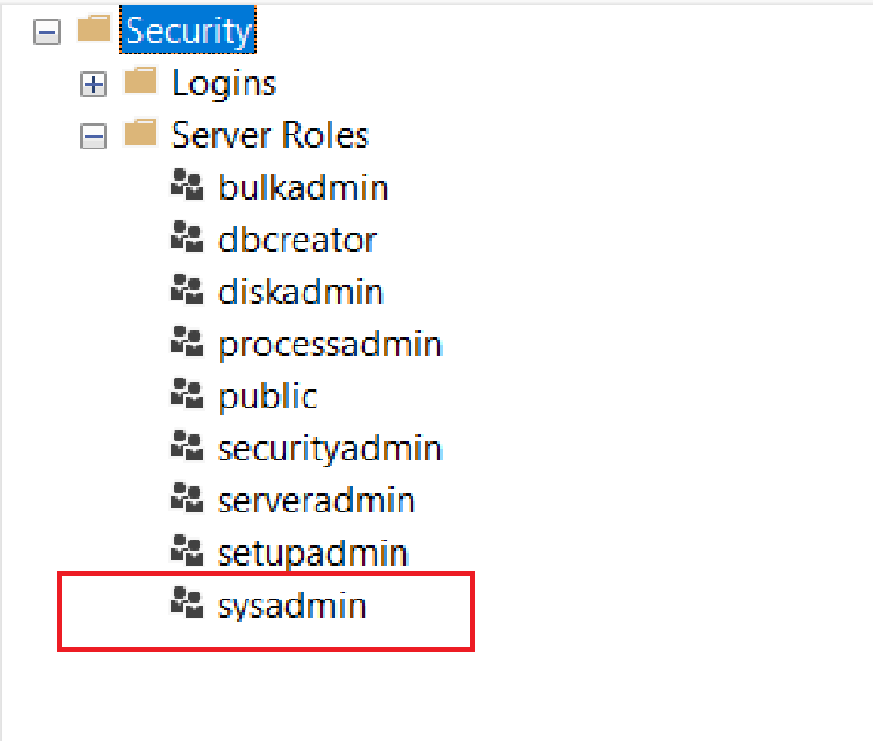
Danh sách các Account login



Danh sách các user

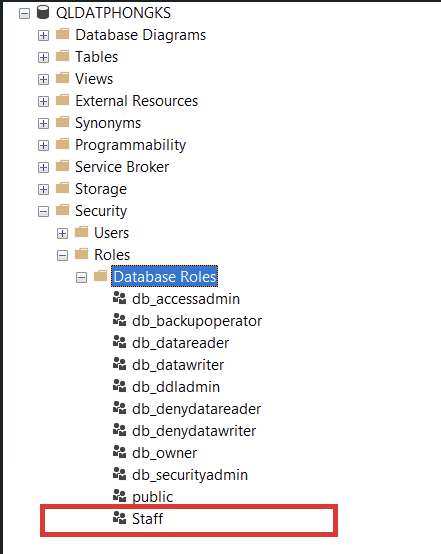


Role của Người quản trị viên là Systemadmin của hệ thống, có thể có tất cả các quyền



Tạo role cho nhân viên

|  |
| --- |
| ---------Tạo một vai trò giữ quyền có tên là Staff---------  CREATE ROLE Staff |



### ***3.4.1 Phân quyền nhân viên***

|  |
| --- |
| --------Gán các quyền trên các bảng cho role Staff---------  GRANT SELECT, REFERENCES ON ACCOUNT TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON BILL TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON DETAILS\_BILL TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON DETAILS\_RESERVED TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON CUSTOMER TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON OFFICIAL\_CUSTOMER TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON DETAILS\_USED\_SERVICE TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON ROOM TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON SERVICE\_ROOM TO Staff  GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON TRACKING\_LOG TO Staff |

|  |
| --- |
| -- Gán quyền thực thi trên các procedure, function cho role Staff  GRANT EXECUTE TO Staff  GRANT SELECT TO Staff  DENY UPDATE, INSERT, DELETE ON EMPLOYEE to Staff;  DENY EXECUTE ON UPDATE\_EMPLOYEE to Staff;  DENY EXECUTE ON ADD\_EMPLOYEE to Staff;  DENY EXECUTE ON DELETE\_EMPLOYEE to Staff;  DENY SELECT ON ftShow\_revenue\_by\_month\_of\_year to Staff;  DENY EXECUTE ON spCreate\_account\_user\_login to Staff  DENY SELECT ON ftGet\_username\_pass\_role to Staff  DENY SELECT, REFERENCES ON view\_EMPLOYEE TO Staff |

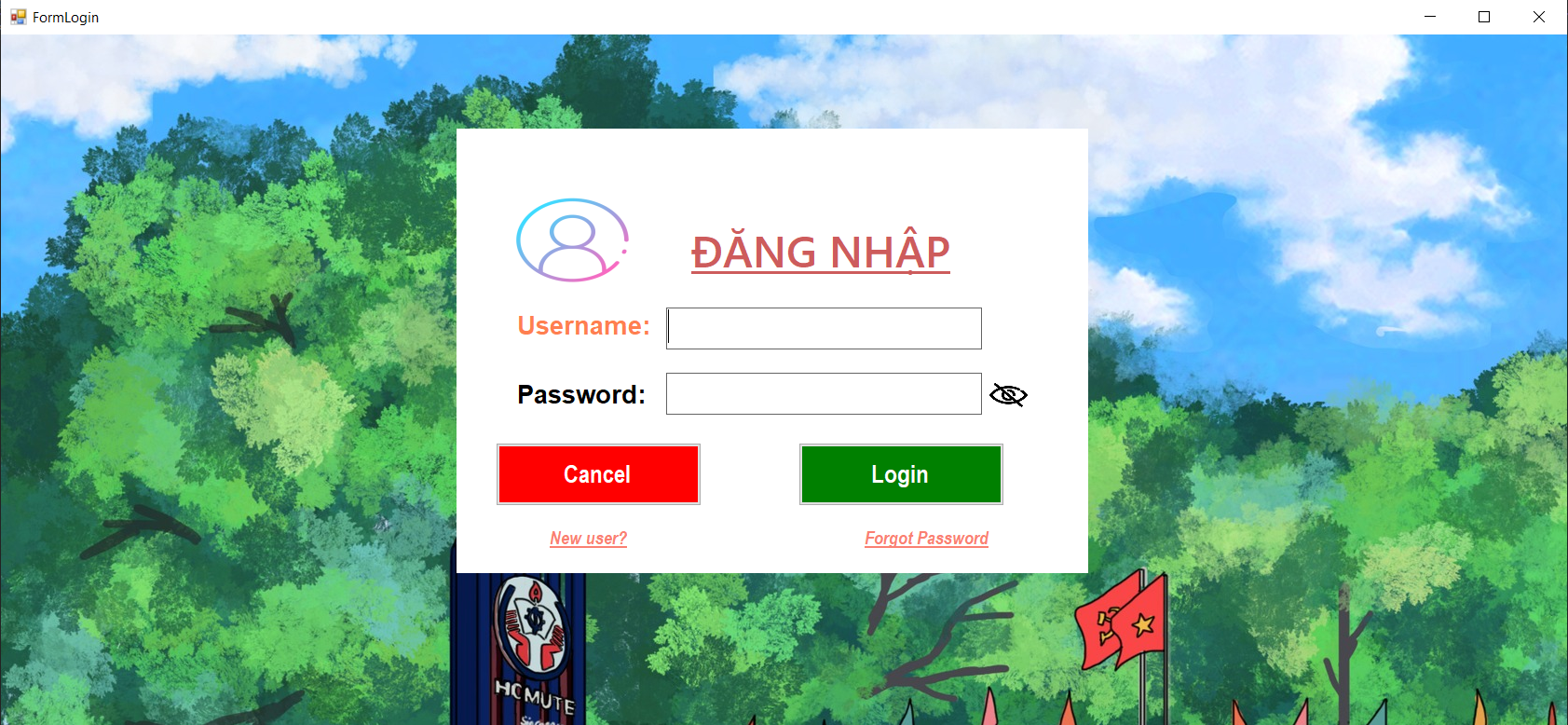
## ***3.5. Hướng dẫn sử dụng***

**Cài đặt chương trình**

* Bước 1: Restore Database
  + Chạy các file \*.sql theo thứ tự sau
    - **Schema.sql** để khởi tạo các bảng và các procedures, functions, views, triggers kèm theo
    - **Data.sql** để tạo dữ liệu mẫu
* Bước 2: Khởi chạy project
  + Mở file **DBMS\_UTEManagement.sln** để khởi động project bằng Visual Studio 2019
  + Chạy phần mềm: các tài khoản đăng nhập:
    - Quản trị viên:
      * Tài khoản: QuanTri
      * Mật khẩu: 123
    - Giảng viên:
      * Tài khoản: GiangVien
      * Mật khẩu: 123

## ***3.6. Chi tiết chương trình***

### ***3.6.1 Đăng nhập***



Nhập username và password cho tài khoản để đăng nhập. Tùy từng loại tài khoản (ứng với username, password đã nhập) mà sẽ có các quyền truy cập khác nhau

### ***3.6.2 Màn hình dashboard***

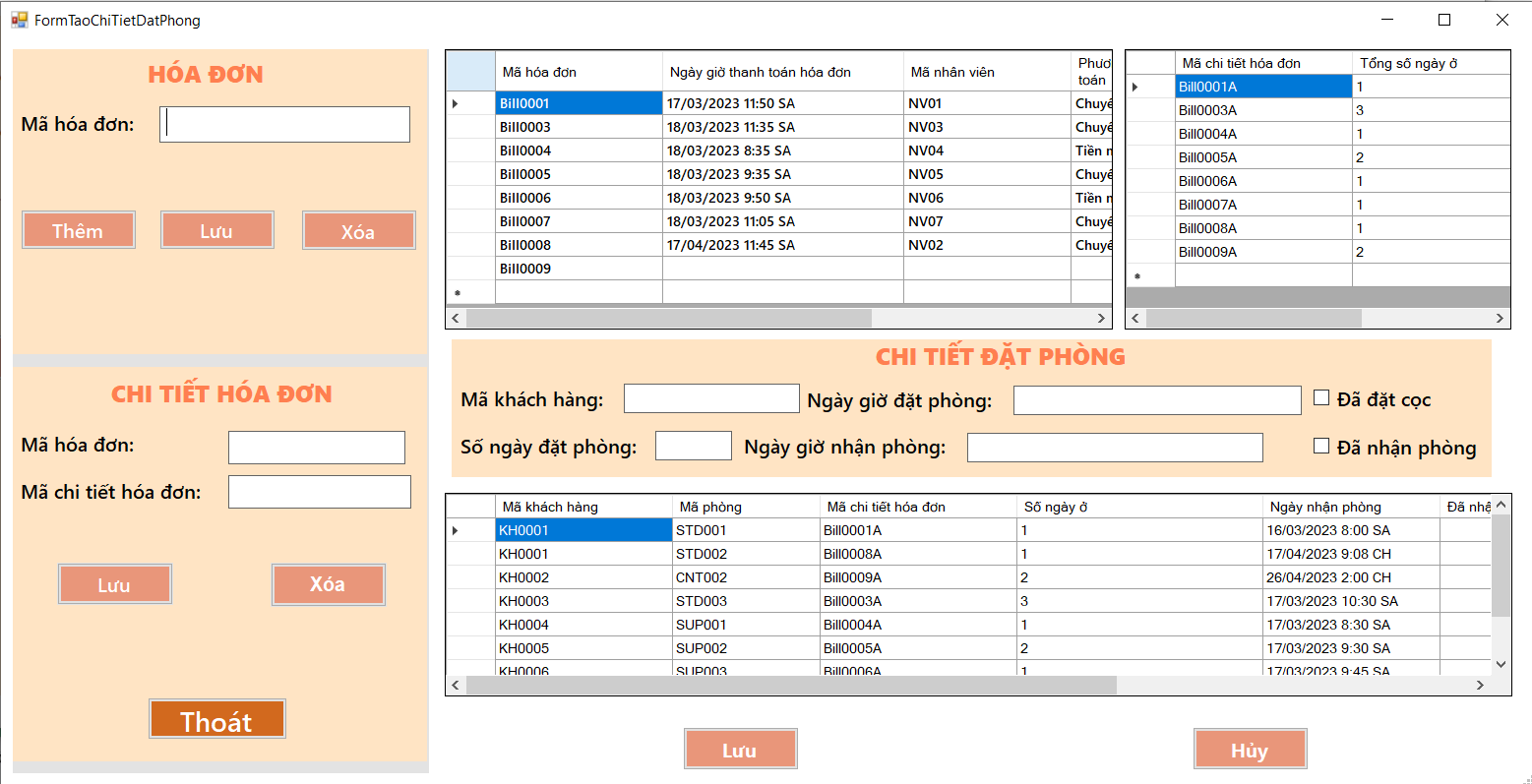
Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

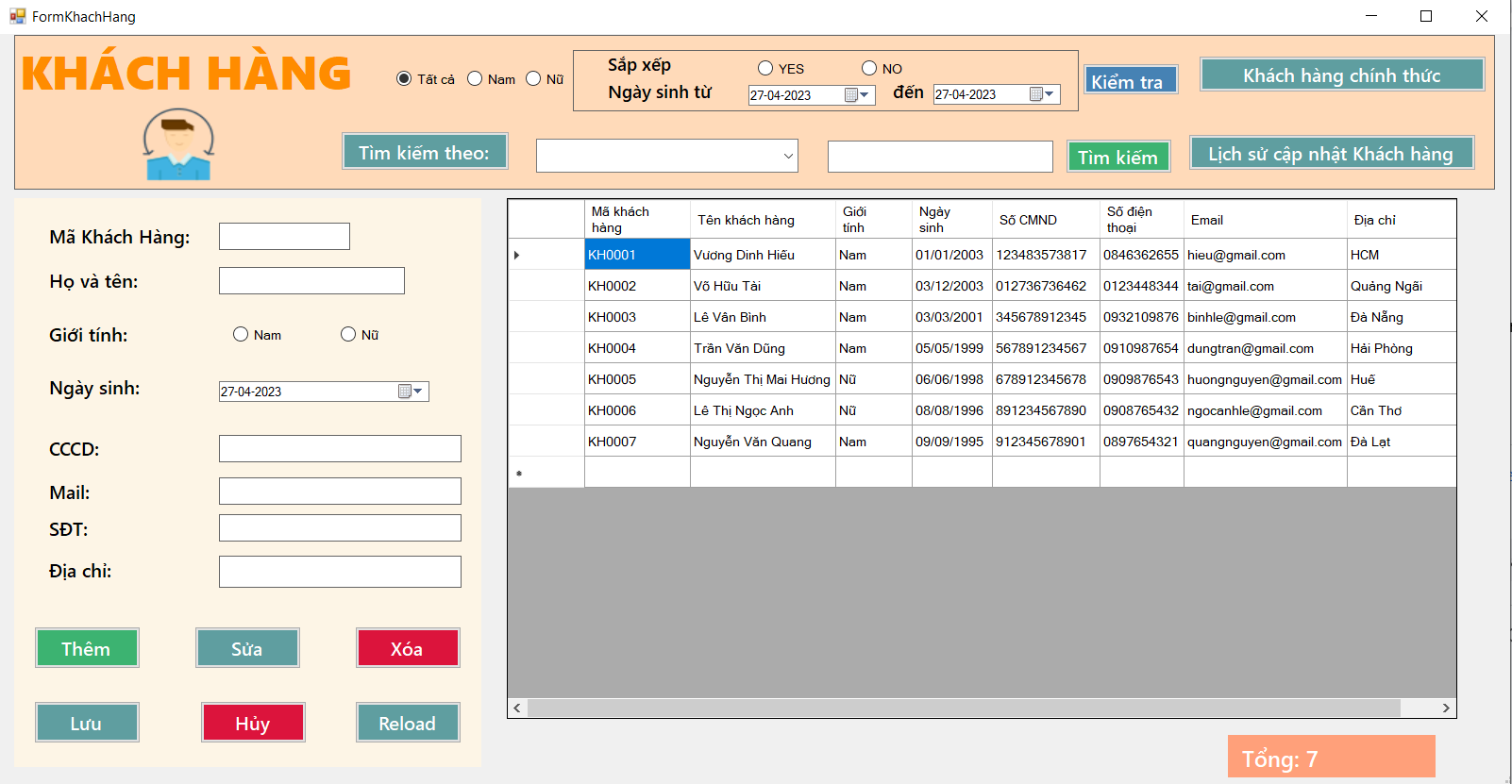
### ***3.6.3 Đặt phòng***



Sau khi nhấn icon sẽ hiện ra giao diện để lễ tân nhập thông tin đặt phòng

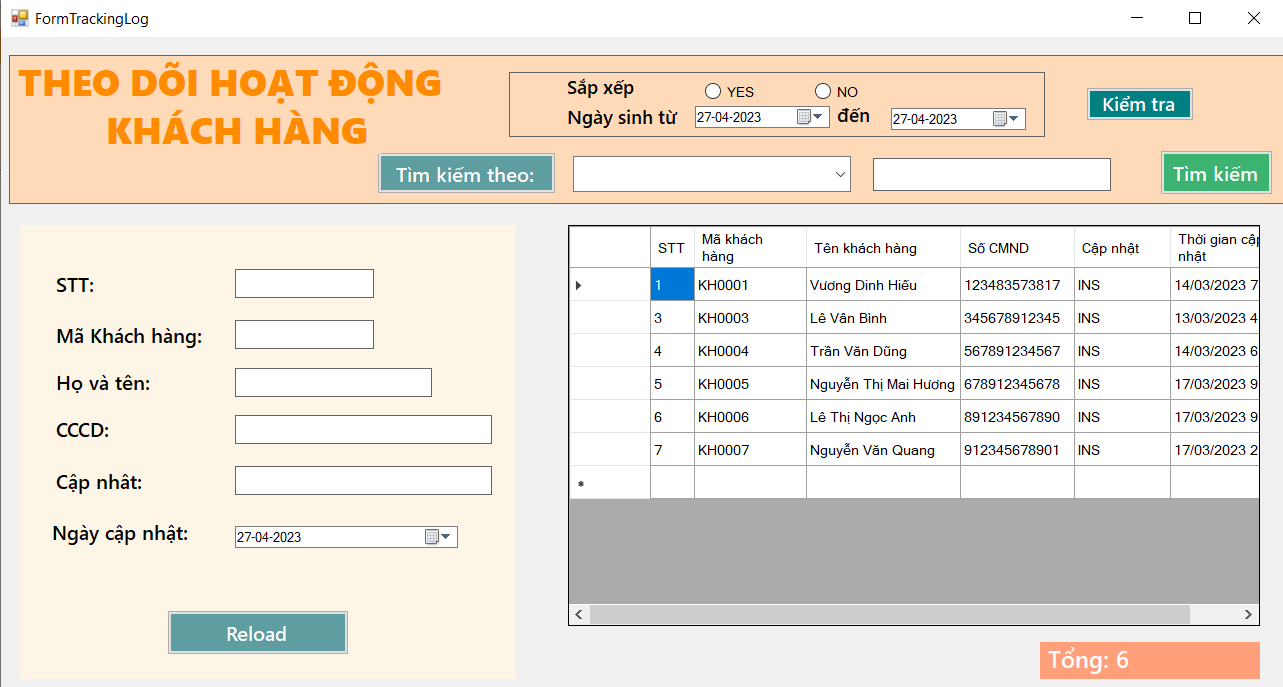


### ***3.6.4 Quản lý khách hàng***

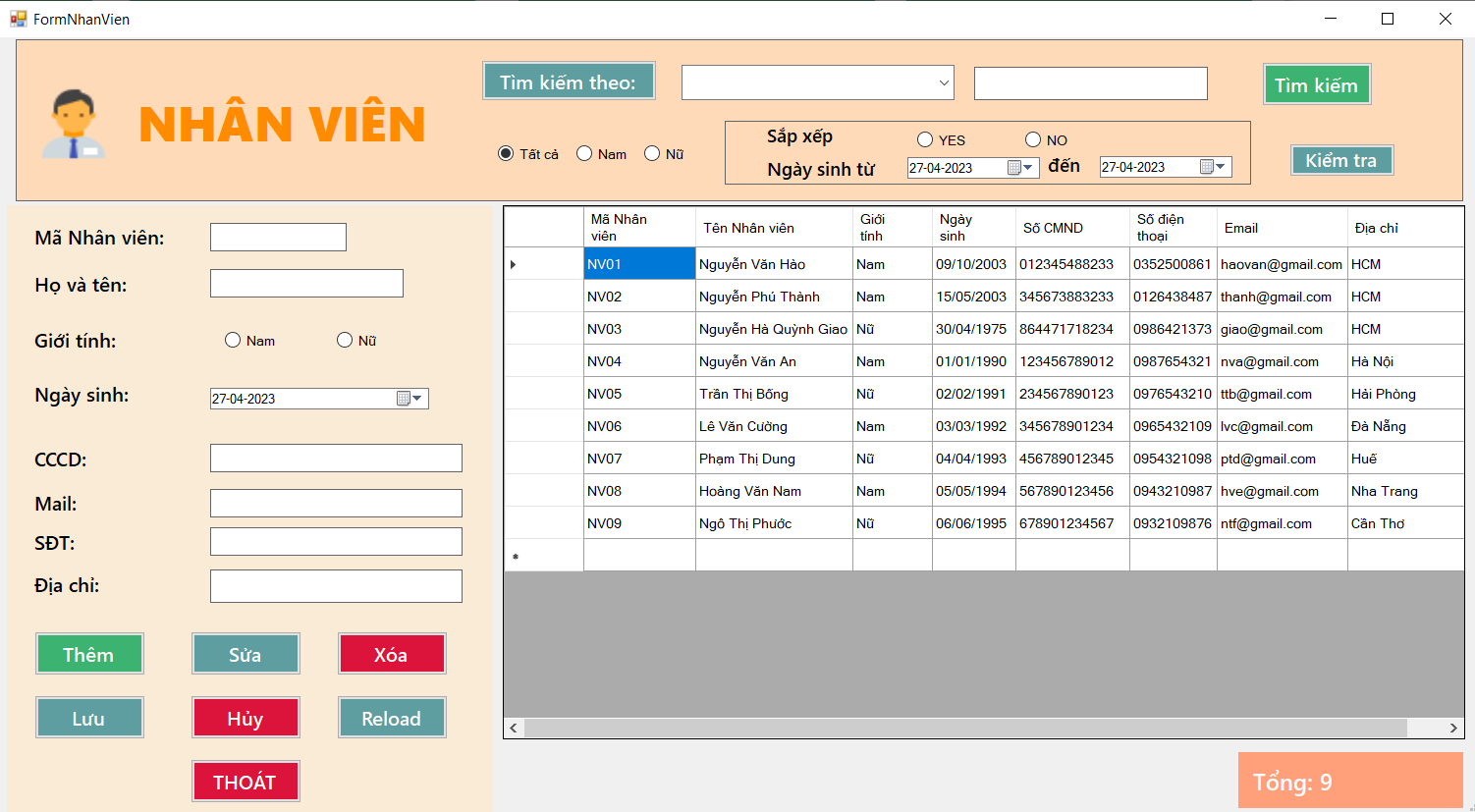


Sau khi nhấn vào “Khách hàng chính thức” sẽ hiện lên bảng khách hàng chính thức tức là khách hàng đã Check-in thành công

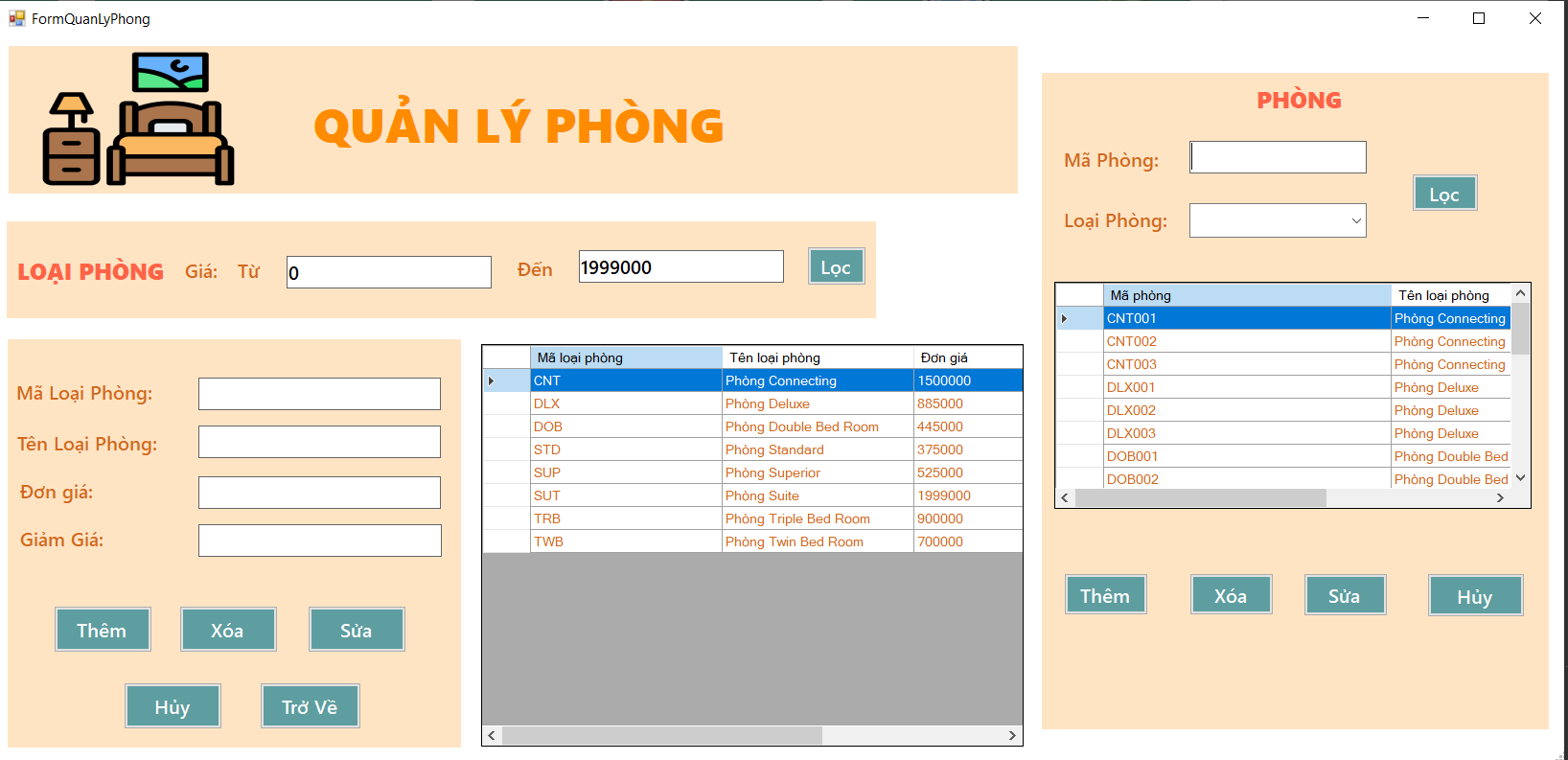
Sau khi nhấn vào “Lịch sử cập nhật khách hàng” sẽ hiện lên bảng theo dõi hành động với khách hàng



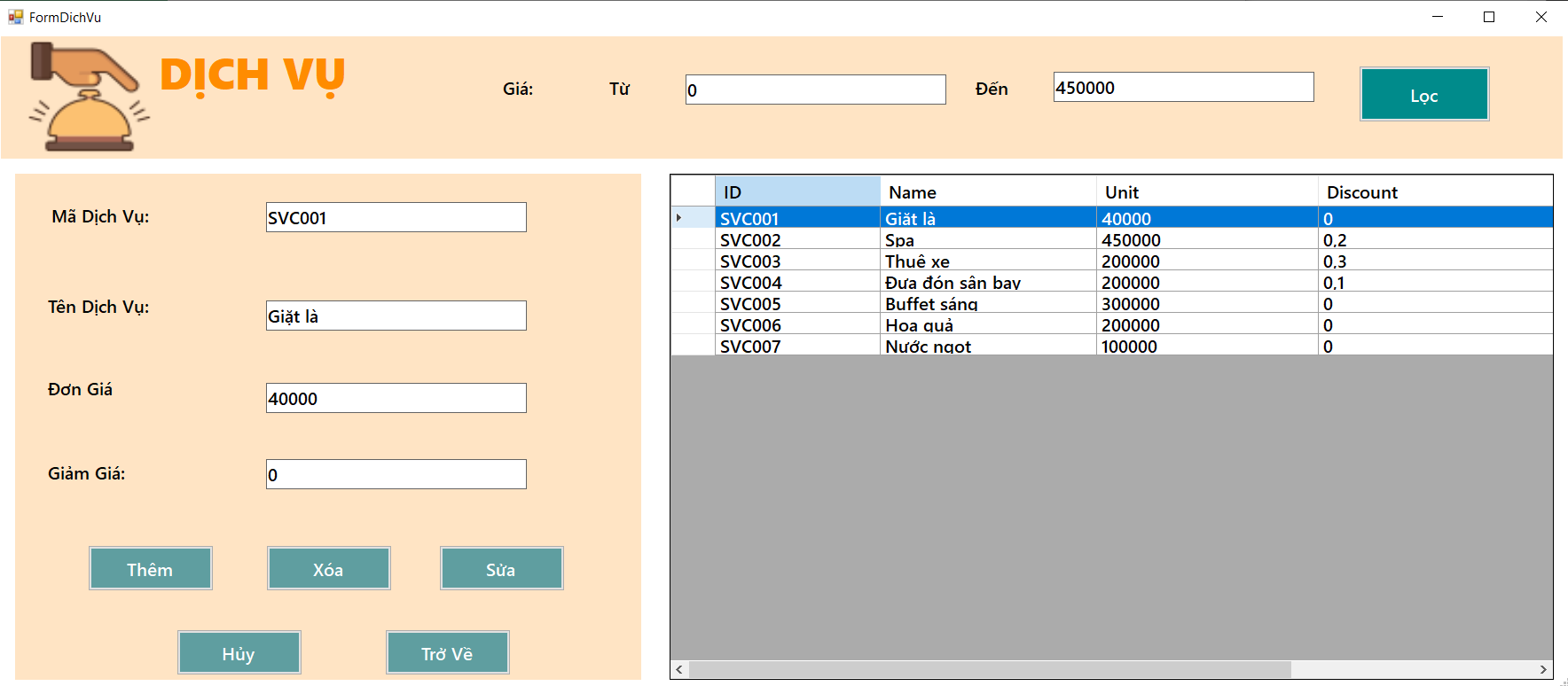
### ***3.6.5 Thông tin nhân viên***



### ***3.6.6 Quản lý phòng***



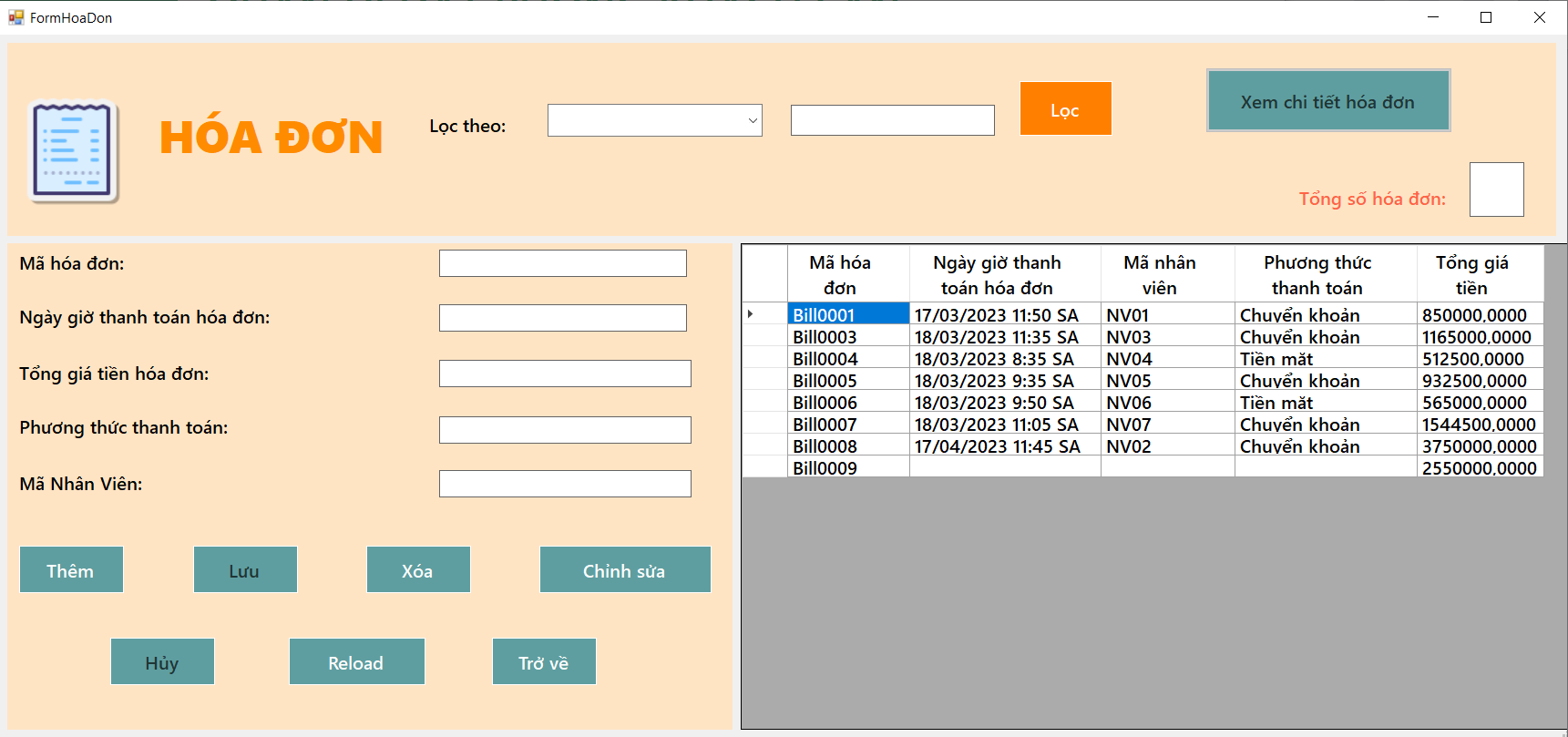
### ***3.6.7 Quản lý dịch vụ***



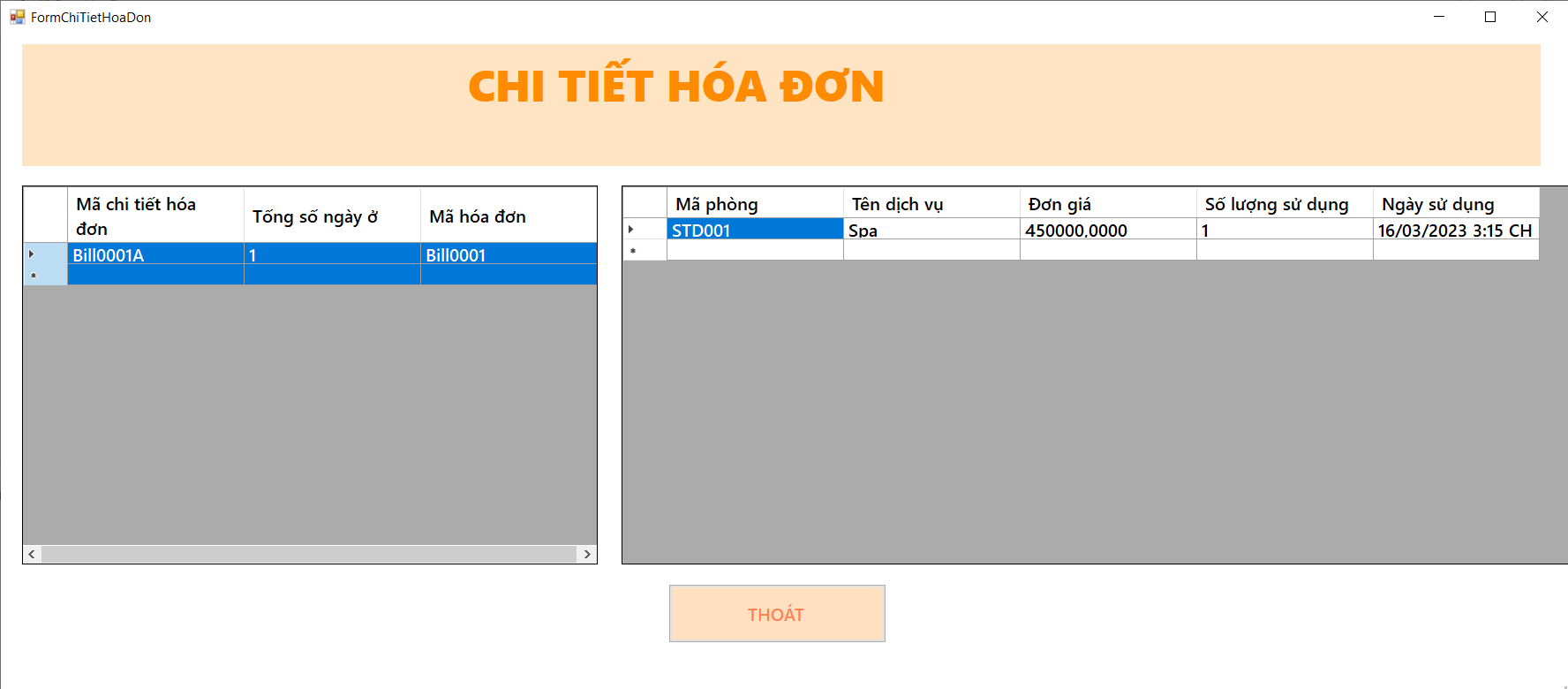
### ***3.6.8 Thống kê***



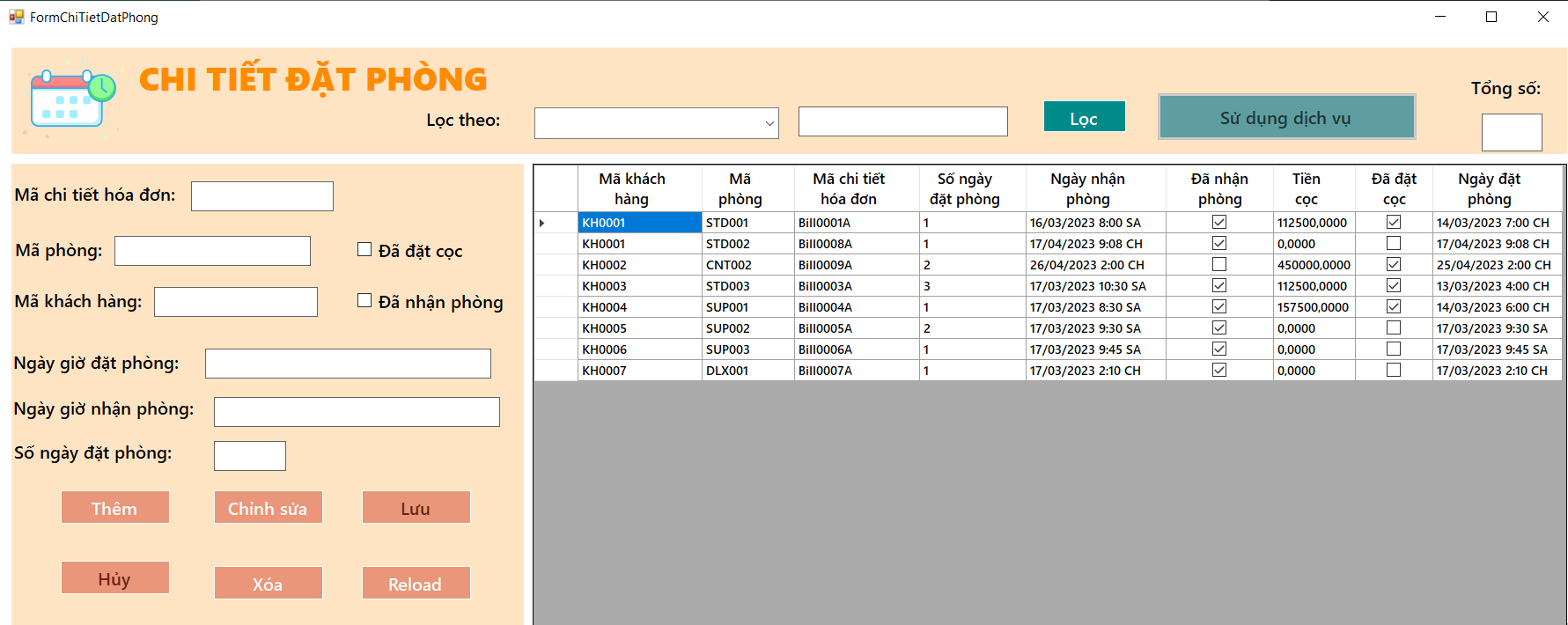
### ***3.6.9 Quản lý hóa đơn***



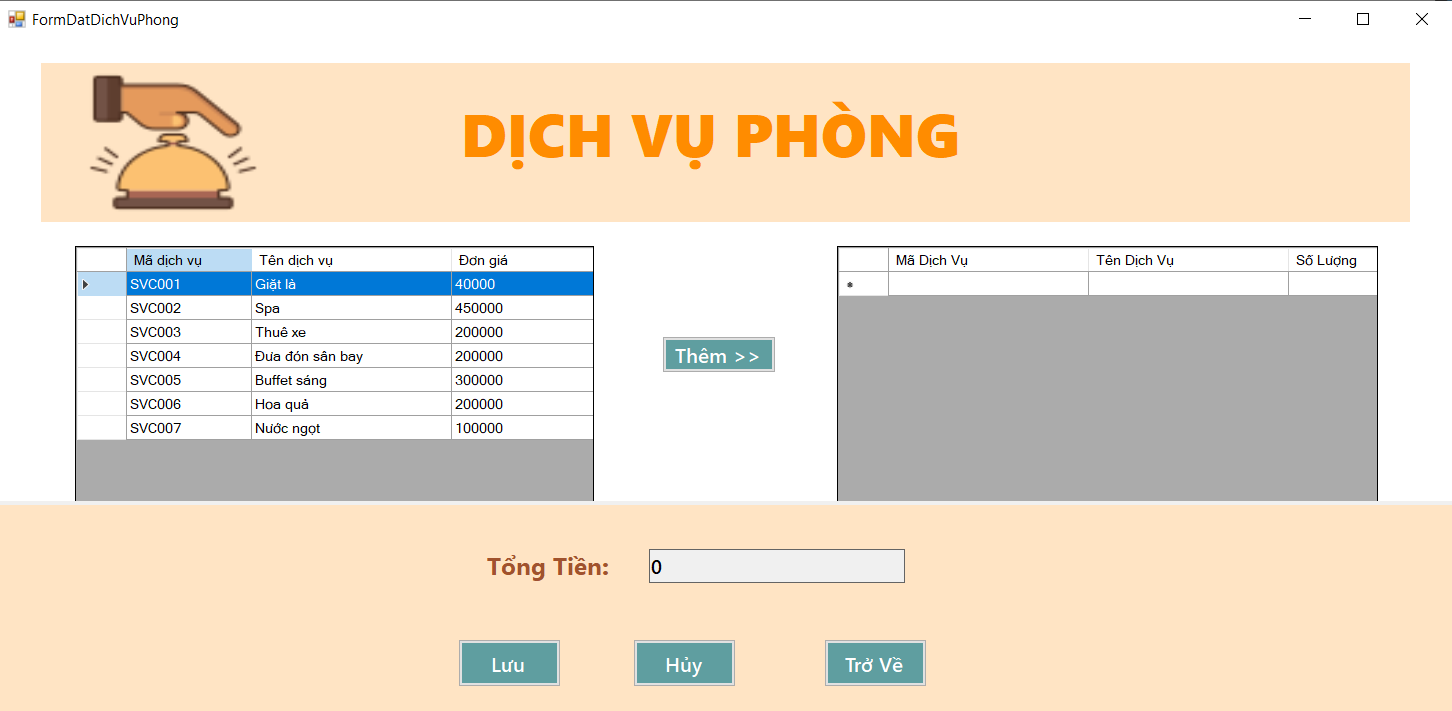
Sau khi nhấn vào “Xem chi tiết hóa đơn”



### ***3.6.10 Quản lý chi tiết đặt phòng***



Sau khi nhấn vào “Sử dụng dịch vụ” sẽ hiện lên bảng để lễ tân thêm các dịch vụ cho khách hàng



# **KẾT LUẬN**