**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**

**[MÃ ĐỒ ÁN]**

**QUẢN LÝ BÁN/ CHO THUÊ NHÀ**

**Mã nhóm** : **11**

**SVTH:** **Lê Bá Quốc – 18126008**

**Đỗ Nguyễn Minh Thành – 18126032**

**Thái Hoàng Nhân – 18126006**

**Tp HCM – 2021.**

# Phân công công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Công việc** | **% Hoàn thành** | **GV nhận xét** |
| 18126008 | Thiết kế CSDL, chức năng hệ thống và xử lý tranh chấp đồng thời | 100% |  |
| 18126032 | Thiết kế giao diện, demo xử lý tranh chấp | 50% |  |
| 18126006 | Thiết kế chức năng giao diện, cài đặt quyền người dùng, demo xử lý tranh chấp | 50% |  |

# Phần báo cáo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung** | **% Hoàn thành** | **MSSV tham gia** | **GV nhận xét** |
| 1 | Phần I 🡪 Phần 4 | 100% | 18126008, 18126032,  18126006 |  |
| 2 | Phần 5 | 50% | 18126032,  18126008,  18126032 |  |

# Phần tình huống truy xuất đồng thời

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung** | **% Hoàn thành** | **MSSV tham gia** | **GV nhận xét** |
| 1 | Thủ tục hệ thống | 100% |  |  |
| 2 | Mô tả tính huống tranh chấp đồng thời | 100% |  |  |
| 3 | Mô tả giải pháp xử lý | 100% |  |  |

# Phần chức năng hệ thống

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung** | **% Hoàn thành** | **MSSV tham gia** | **GV nhận xét** |
| 1 | Danh sách chức năng | 100% |  |  |
| 2 | Mô tả chi tiết chức năng | 100% |  |  |

# Phần nâng cao

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung** | **% Hoàn thành** | **MSSV tham gia** | **GV nhận xét** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

MỤC LỤC

[1 **PHẠM VI HỆ THỐNG** 4](#_Toc351574587)

[*1.1* *Giới thiệu* 4](#_Toc351574588)

[*1.2* *Phạm vi nghiệp vụ* 4](#_Toc351574589)

[*1.3* *Qui định của hệ thống* 4](#_Toc351574590)

[2 **MÔ TẢ CHỨC NĂNG HỆ THỐNG** 4](#_Toc351574591)

[*2.1* *Danh sách chức năng* 4](#_Toc351574592)

[*2.2* *Mô tả chi tiết chức năng* 4](#_Toc351574593)

[3 **MÔ TẢ CƠ SỞ DỮ LIỆU** 4](#_Toc351574594)

[*3.1* *Lược đồ cơ sở dữ liệu* 4](#_Toc351574595)

[*3.2* *Mô tả cơ sở dữ liệu* 4](#_Toc351574596)

[*3.3* *Ràng buộc của cơ sở dữ liệu* 5](#_Toc351574597)

[4 **MÔ TẢ TÌNH HUỐNG TRANH CHẤP ĐỒNG THỜI** 5](#_Toc351574598)

[*4.1* *Mô tả thủ tục của hệ thống* 5](#_Toc351574599)

[*4.2* *Mô tả tình huống tranh chấp đồng thời* 5](#_Toc351574600)

[*4.3* *Mô tả giải pháp xử lí* 5](#_Toc351574601)

[5 **ỨNG DỤNG DEMO** 5](#_Toc351574602)

[*5.1* *Mô tả cấu trúc ứng dụng* 5](#_Toc351574603)

[*5.2* *Tình huống tranh chấp đồng thời* 5](#_Toc351574604)

[5.3 Bộ dữ liệu test các tình huống 5](#_Toc351574605)

[6 **TỔNG KẾT** 6](#_Toc351574606)

# PHẠM VI HỆ THỐNG

## *Giới thiệu*

[Giới thiệu sơ bộ về hệ thống]

## *Phạm vi nghiệp vụ*

* Kỹ năng trong lập trình C#, sử dụng và lập trình SQL server, tìm và xủ lý các lỗi tranh chấp đồng thời trong hệ thống, các kỹ năng lập trình căn bản là những kỹ năng cần thiết để hoàn thành project này.

## *Qui định của hệ thống*

* Chính xác, nhanh và không có sự tranh chấp đồng thời giữa các chức năng, có giao diện để mô tả, trực quan hóa các xử lý tranh chấp.

# MÔ TẢ CHỨC NĂNG HỆ THỐNG

## *Danh sách chức năng*

* [Xác định các chức năng sẽ cài đặt trong hệ thống dựa vào phạm vi đã xác định ở trên (có thể dùng mô hình usecase)]

1. Đăng ký user.
2. Đăng nhập hệ thống.
3. Thêm dữ liệu.
4. Xóa dữ liệu.
5. Cập nhật dữ liệu.
6. Xem dữ liệu.

## *Mô tả chi tiết chức năng*

* [Đăng ký user]
  + Mục tiêu : tạo được user cho người dùng sủ dụng đăng nhập vào hệ thống.
  + Người dùng: khách hang, và người quản lý .
  + Đầu vào: họ tên, số điện thoại, địa chỉ, email, mật khẩu.
  + Đầu ra : user được phân quyền bởi hệ thống.
  + Các bước thực hiện: chi tiết từng bước.
    - Bước 1: Nhập thông tin vào ô đăng ký.
    - Bước 2: kiểm tra lại thông tin và tiến hành đăng ký.
* [Đăng nhập hệ thống]
  + Mục tiêu : đăng nhập vào hệ thống.
  + Người dùng: khách hang, và người quản lý .
  + Đầu vào: tên user và mật khẩu.
  + Đầu ra : đăng nhập được vào hệ thống.
  + Các bước thực hiện: chi tiết từng bước
    - Bước 1: nhập user và mật khẩu.
    - Bước 2: đăng nhập vào hệ thống.
* [Thêm dữ liệu]
  + Mục tiêu : Thêm được dữ liệu vào cơ sở .
  + Người dùng: khách hang (bị giới hạn ), và người quản lý.
  + Đầu vào: dữ liệu cần thêm, và thêm ở bảng nào.
  + Đầu ra : bảng đã được thêm dữ liệu.
  + Các bước thực hiện: chi tiết từng bước
    - Bước 1: xác định dữ liệu cần thêm.
    - Bước 2: kiểm tra lại thông tin và tiến hành thêm.
* [Xóa dữ liệu]
  + Mục tiêu : Xóa được dữ liệu vào cơ sở .
  + Người dùng: khách hang(bị giới hạn), và người quản lý.
  + Đầu vào: dữ liệu cần xóa, và xóa ở bảng nào.
  + Đầu ra : bảng đã được xóa dữ liệu.
  + Các bước thực hiện: chi tiết từng bước
    - Bước 1: xác định dữ liệu cần xóa.
    - Bước 2: kiểm tra lại thông tin và tiến hành xóa.
* [cập nhật dữ liệu]
  + Mục tiêu : Cập nhật được dữ liệu vào cơ sở .
  + Người dùng: khách hang(bị giới hạn), và người quản lý.
  + Đầu vào: dữ liệu cần cập nhật, và cập nhật ở bảng nào.
  + Đầu ra : bảng đã được cập nhật dữ liệu.
  + Các bước thực hiện: chi tiết từng bước
    - Bước 1: xác định dữ liệu cần cập nhật.
    - Bước 2: kiểm tra lại thông tin và tiến hành cập nhật.
* [Xem dữ liệu]
  + Mục tiêu : Xem được dữ liệu của hệ thống .
  + Người dùng: khách hang (bị giới hạn), và người quản lý.
  + Đầu vào: dữ liệu cần xem.
  + Đầu ra : bảng chứa dữ liệu đó.
  + Các bước thực hiện: chi tiết từng bước
    - Bước 1: xác định dữ liệu cần xem.
    - Bước 2: bấm vào button và xem được dữ liệu.

# MÔ TẢ CƠ SỞ DỮ LIỆU

## *Lược đồ cơ sở dữ liệu*

**Diagram, schematic

Description automatically generated**

## *Mô tả cơ sở dữ liệu*

## ChiNhanh(maCN,fax,SDT, duong, quan, khuVuc, thanhPho )

## ChuNha(maCN, SDT, diaChi, ten)

## KhachHang(maKH, ten, diaChi, SDT, tieuChi, maCN, loaiNha)

## LoaiNha(MaLN, tenLN)

## Nha(maN, tinhTrangNha, maNVQL, soLuongPhong, MaCN, ngayDB, ngayHH, luotXem, maLN, duong, quan, khuVuc, thanhPho, giaThue1T, giaBan, dieuKienBan)

## NhanVien(maNV, ten, ngaySinh, maCN, gioiTinh, SDT, luong, diaChi)

**A picture containing text, indoor, screenshot

Description automatically generated**

## *Ràng buộc của cơ sở dữ liệu*

## NOT NULL: các khóa chính, thông tin địa chỉ ( Chi nhánh, chủ nhà, nhà, khách hàng, nhân viên), toàn bộ thông tin nhà, thông tin nhân viên, khách hàng.

## UNIQUE: các khóa chính, SDT, fax.

## CHECK: Nhà(soLuongPhong > 0, luotXem >=0, giaThue1T > 0, giaBan > 0), NhanVien(luong > 0)

## DEFAULT: NhanVien(ten, ngaySinh, gioiTinh, money, diaChi), KhachHang(tieuChi, loaiNha), NHA(soLuongPhong, khuVuc, dieuKienBan).

# MÔ TẢ TÌNH HUỐNG TRANH CHẤP ĐỒNG THỜI

## *Mô tả thủ tục của hệ thống*

* Tên thủ tục : Phân công nhân viên
  + Mục đích sử dụng thủ tục: phân công nhân viên đến chi nhánh để làm việc
  + Ý nghĩa các tham số:
    - maNV: mã nhân viên
    - maCN: mã chi nhánh
  + Các bước đặt tả thủ tục:
    - Kiểm tra sự tồn tại của nhân viên
    - Phân công chi nhánh
* Tên thủ tục : Thêm Nhà
  + Mục đích sử dụng thủ tục: thêm nhà vào database
  + Ý nghĩa các tham số:
    - Tất cả tham số trong bảng Nha
  + Các bước đặt tả thủ tục:
    - Insert into Nha values (…)
* Tên thủ tục : Update chủ nhà
  + Mục đích sử dụng thủ tục: thay đổi chủ nhà
  + Ý nghĩa các tham số:
    - maN: mã nhà
    - maCN : mã chủ nhà
  + Các bước đặt tả thủ tục:
    - Kiểm tra nhà tồn tại hay không
    - Đổi chủ
* Tên thủ tục : Xóa nhà
  + Mục đích sử dụng thủ tục: xóa những nhà không còn hoạt động
  + Ý nghĩa các tham số:
    - maN: mã nhà
  + Các bước đặt tả thủ tục:
    - Kiểm tra sự tồn tại của nhà
    - Xóa nhà ra khỏi database

## *Mô tả tình huống tranh chấp đồng thời và giải pháp xử lý*

**Lỗi dirty read 1:**

* Nhân viên đang tư vấn thuê nhà cho khách hàng, nhưng chủ nhà lại thay đổi giá thuê nhưng chưa commit. Hệ thống không kịp cập nhật lại giá nhà cho nhân viên.

Transaction 1: kiemtraThongTinNha(T1)

Input : @maNVPhuTrach, @maNha

Output : Danh sách các nhà được nhân viên này phụ trách.

* Check xem mã nhân viên phụ trách có tồn tại hay không.(@maNVPhuTrach)
* Check xem mã nhà có tồn tại trong hệ thống hay không (@maNha)
* Xuất danh sách các nhà – bao gồm giá tiền cũ.
* Commit tran.

Transaction 2: updateThongTinNha(T2)

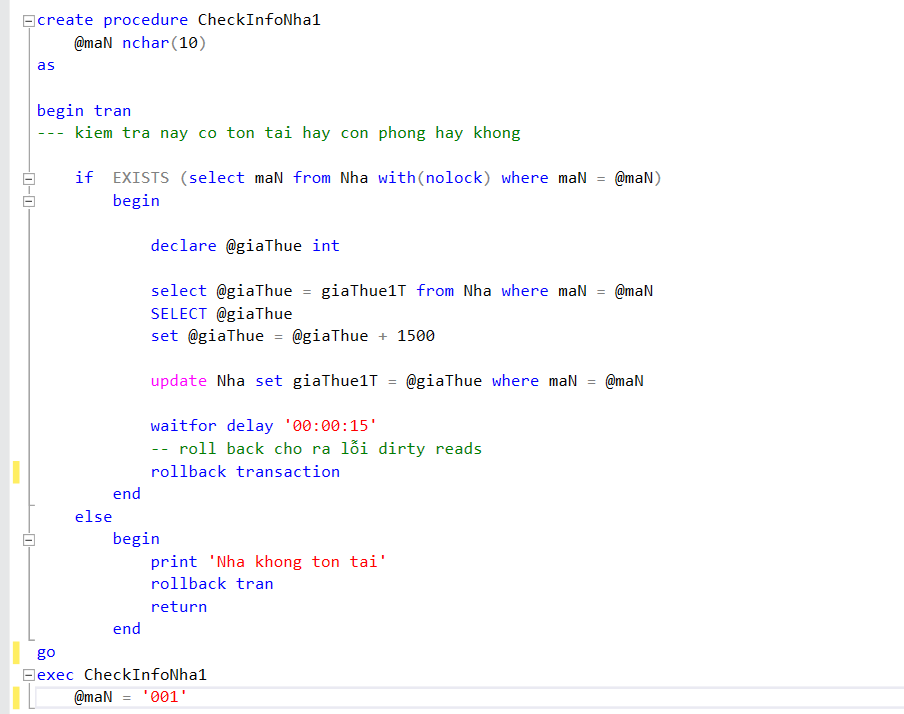
Input : @maNha, @giaThueMotThang

Output : Thông tin về nhà mới được cập nhật( VD : giá nhà cho thuê vừa được thay đổi )

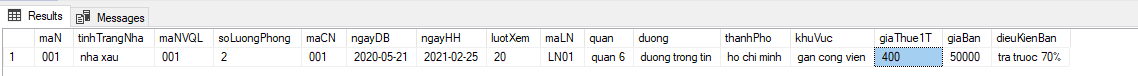
1. Check mà nhà có tồn tại :@maNha.
2. Check giá thuê mới có hợp lệ - kiểm tra xem chủ nhà có nhập giá tiền phù hợp với kiểu dữ liệu đã được lập trình hay không.
3. Cập nhật lại giá tiền mới của nhà.
4. Hiển thị nhà vừa cập nhật giá tiền.

|  |  |  |
| --- | --- | --- |
|  | T2(Nhan Vien) | T1 |
| 1 | Begin tran | Begin tran |
| 2 |  | Update giaThueMotThang |
| 3 | Select \* from Nha where maNVPhuTrach=… |  |
| 4 | Xuất danh sách nhà. |  |
| 5 |  | Running… |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

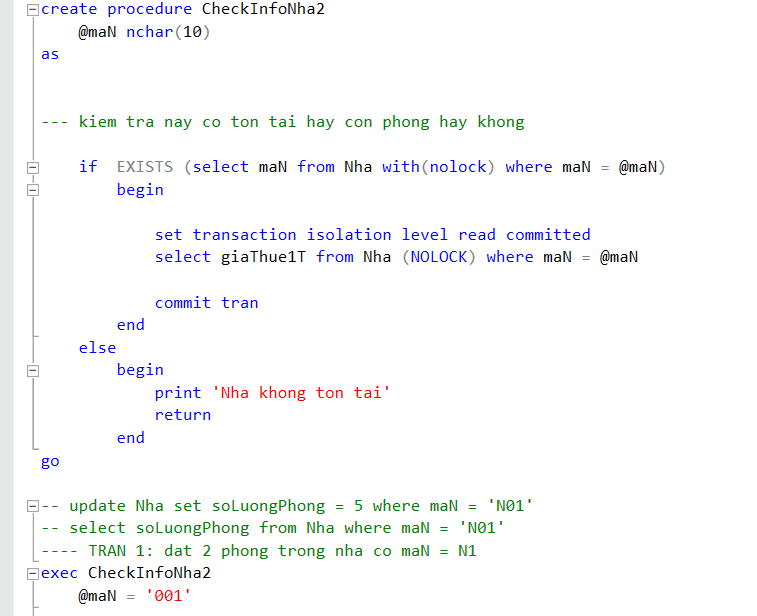
Mô phỏng:

TRANSACTION 1:

Kết quả :



TRANSACTION 2:



Kết quả :



Cách giải quyết lỗi : **SỬ DỤNG READ COMMITTED**

Khi T1( chủ nhà ) thực hiện thao tác update, T1 sẽ xin khóa X(SQL cho phép ), trong lúc này khi T2(khách hàng) xin khóa S thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

Điều này giúp người dùng tránh được việc sử dụng lại dữ liệu rác mà chủ nhà chưa commit.

|  |  |  |
| --- | --- | --- |
|  | T2(Nhan Vien) | T1 |
| 1 | Begin tran | Begin tran |
| 2 | SET TRAN ISOLATION LEVEL READ COMMITTED | Update giaThueMotThang |
| 3 |  | Xin khóa **X**, SQL cấp khóa **X** |
| 4 | Select \* from Nha where maNVPhuTrach=… (T2 xin khóa S nhưng SQL không cấp phép do T1 đang giữ khóa X). |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  | Running… |
| 8 | Commit tran | Commit tran |

**Lỗi dirty read 2:**

* Chủ nhà thay đổi tình trạng nhà nhưng chưa commit, trong lúc khách hàng đã lựa chọn được căn nhà này. Transaction 1: timnhaTheoTinhTrang(T1)

Input : @maKH, @tinhTrangNha

Output : Danh sách các nhà phù hợp với tình trang của khách hàng yêu cầu.

* Check xem mã khách hàng có tồn tại hay không.(@maKH)
* Check xem tình trạng nhà có tồn tại trong hệ thống không (@tinhTrangNha)
* Xuất danh sách các nhà – bao gồm mã nhà, tình trạng nhà ,….
* Commit tran.

Transaction 2: updateTinhTrangNha(T2)

Input : @maNha, @tinhTrangNha

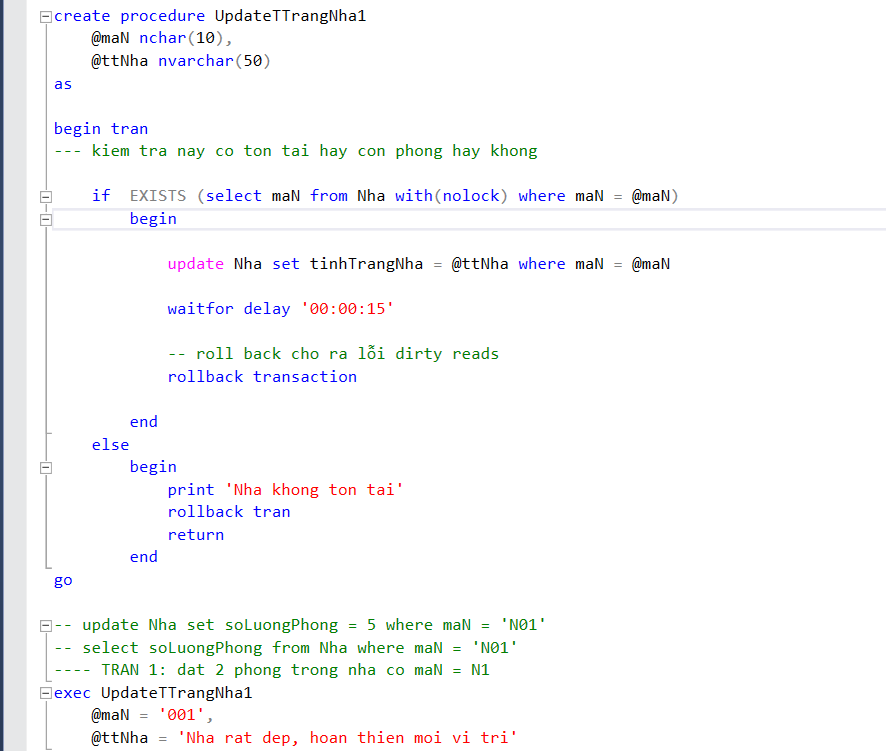
Output : Thông tin về nhà mới được cập nhật( VD : tình trạng nhà vừa được thay đổi )

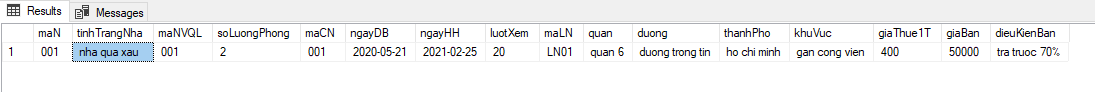
* Check mà nhà có tồn tại :@maNha.
* Check xem tình trạng nhà có hợp lệ không.
* Cập nhật tình trạng nhà.
* Hiển thị nhà vừa tình trạng.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2 |
| 1 | Begin tran |  |
| 2 | Select \* from Nha where tinhTrangNha=… | Begin tran |
| 3 | Running… |  |
| 4 | Xuất danh sách nhà. | Update tinhTrangNha |
| 5 |  | Running… |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

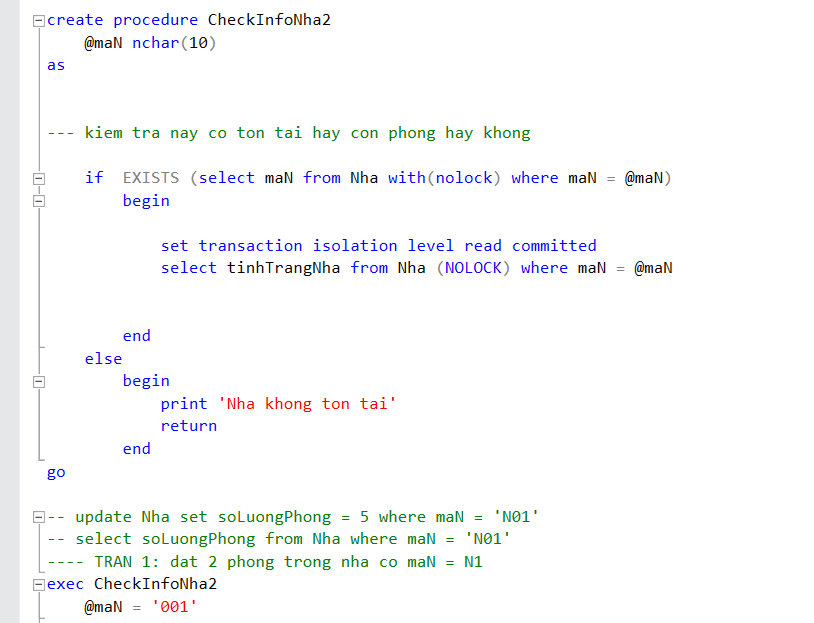
Mô phỏng lỗi:

TRACSACTION 1:





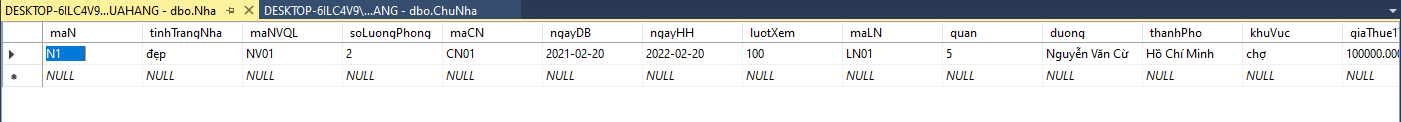
TRANSACTION 2:



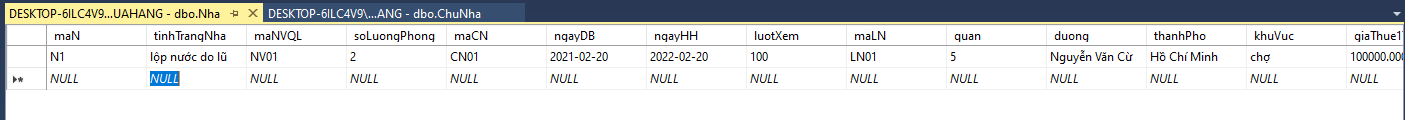
Kết quả :



Chẳng hạn lúc đầu khi khách hàng lấy dữ liệu của nhà về với yêu cầu: tình trang nhà đẹp.



Tuy nhiên trong lúc đó do mưa lũ khiến nhà bị lộp, mà chủ nhà không kịp cập nhật lại:



* Lỗi dirty read.

Cách giải quyết lỗi : **SỬ DỤNG REPEATABLE READ**

Khi T1( khách hàng ) thực hiện thao tác select, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2(chủ nhà) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2 |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ | Begin tran |
| 2 | Select \* from Nha where tinhTrangNha=…  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 |  |
| 3 | Running… |  |
| 4 | Xuất danh sách nhà. | Update tinhTrangNha  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |
| 5 |  | Running… |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi dirty read 3:**

* Chủ nhà trao quyền quản lý nhà cho người thân nhưng chưa cập nhật lại thông tin cá nhân + số tiền thuê mà người thân mong muốn nhưng lại chưa commit ,… Trong lúc đó khách hàng liên lạc thuê nhà thông qua chủ nhà cũ.

Transaction 1: timNha (T1)

Input : @maKH, @maChuNha,@diaChi,@sDTChuNha

Output : Danh sách các nhà phù hợp với tình trang của khách hàng yêu cầu.

* Check xem mã khách hàng có tồn tại hay không.(@maKH)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Check xem địa chỉ có tồn tại hay không (@diaChi)
* Check số điện thoại của chủ nhà (@sdtChuNha)
* Xuất danh sách các nhà – bao gồm mã nhà, tình trạng nhà ,….
* Commit tran.

Transaction 2: updateThongTinChuNha(T2)

Input : @maNha ,@tenChuNha,@sdtChuNha

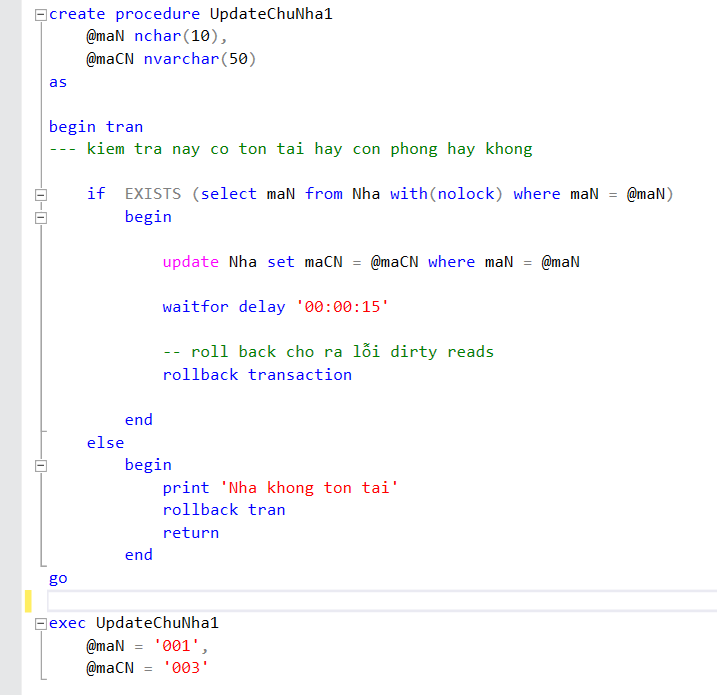
Output : Thông tin về nhà mới được cập nhật( VD : tên chủ nhà mới, số điện thoại chủ nhà mới )

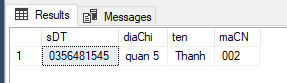
* Check mà nhà có tồn tại :@maNha.
* Check xem tên và số điện thoại mới có hợp lệ không(@tenChuNha,@sdtChuNha)
* Cập nhật thông tin chủ nhà.
* Hiển thị nhà vừa cập nhật.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2 |
| 1 | Begin tran |  |
| 2 | Select \* from Nha | Begin tran |
| 3 | Running… |  |
| 4 | Xuất danh sách nhà bao gồm thông tin chủ nhà. | Update thongtinChuNha |
| 5 |  | Running… |
| 6 |  |  |
| 7 |  |  |
| 8 |  | Commit tran |

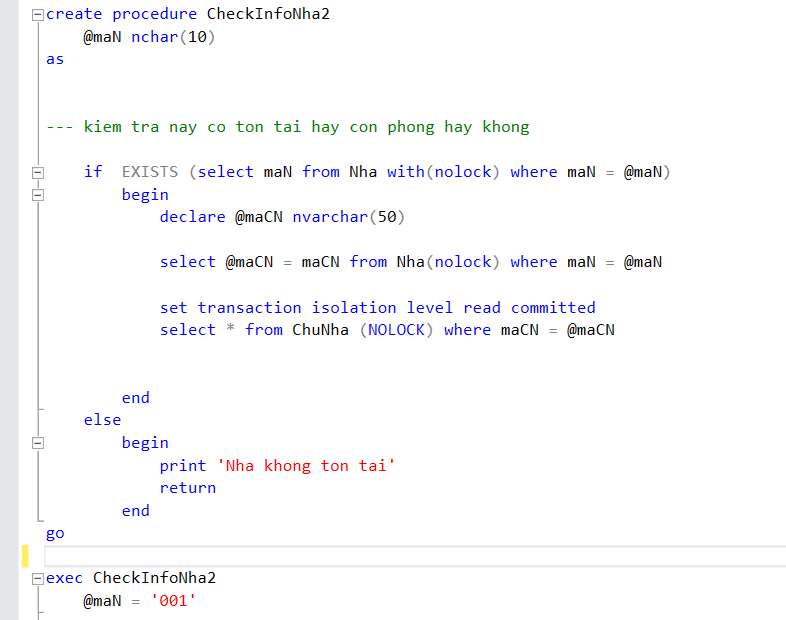
Mô phỏng lỗi :

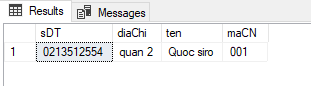
TRANSACTION 1 :



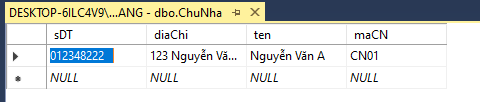
Kết quả : 

TRANSACTION 2 :

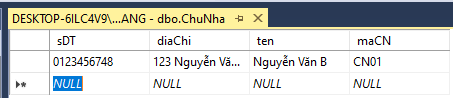


Kết quả : 

Ta có dữ liệu chủ nhà ban đầu do khách hàng yêu cầu :



Trong lúc đó chủ nhà cập nhật số điện thoại mà tên của chủ sở hữu mới :



* Lỗi dirty read

Giải quyết lỗi  : **SỬ DỤNG REPEATABLE READ**

Khi T1( khách hàng ) thực hiện thao tác select, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2(chủ nhà) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2 |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ |  |
| 2 | Select \* from Nha  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 | Begin tran |
| 3 | Running… |  |
| 4 | Xuất danh sách nhà bao gồm thông tin chủ nhà. | Update thongtinChuNha  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |
| 5 |  | Running… |
| 6 |  |  |
| 7 |  |  |
| 8 |  | Commit tran |

**Lỗi Lost update 1**

* Hai khách hàng cùng lúc muốn thuê một nhà, họ đã thành công hoàn thành hết thủ tục. Tuy nhiên số lượng phòng không kịp cập nhật.

Transaction 1: thueNha(T1)

Input : @maKH, @maNha

Output : Danh sách các nhà.

* Check xem mã nhà có tồn tại trong hệ thống hay không (@maNha)
* Xuất danh sách các nh.
* Commit tran.

Transaction 2: thueNha(T2)

Input : @maKH, @maNha

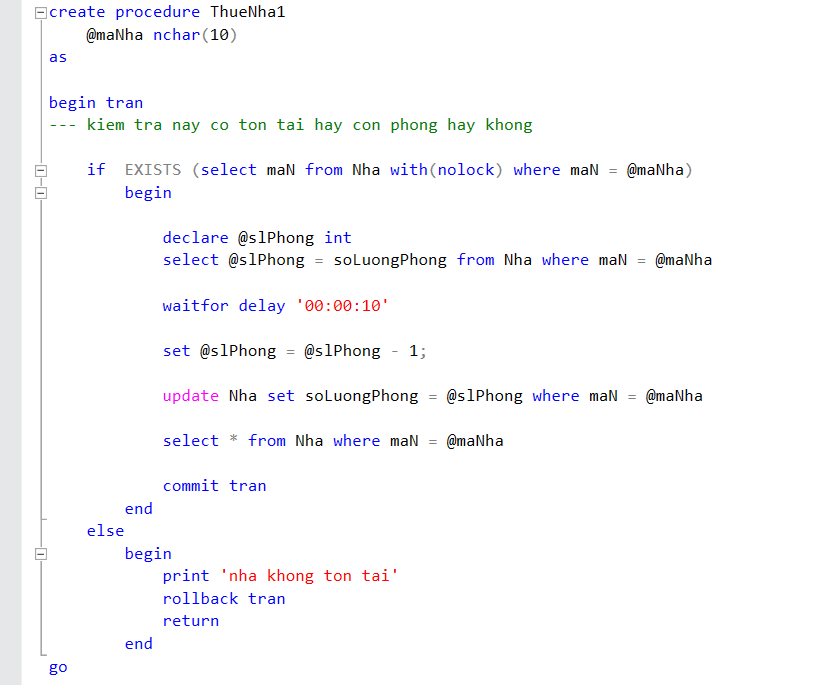
Output : Danh sách các nhà.

* Check xem mã nhà có tồn tại trong hệ thống hay không (@maNha)
* Xuất danh sách các nh.
* Commit tran.

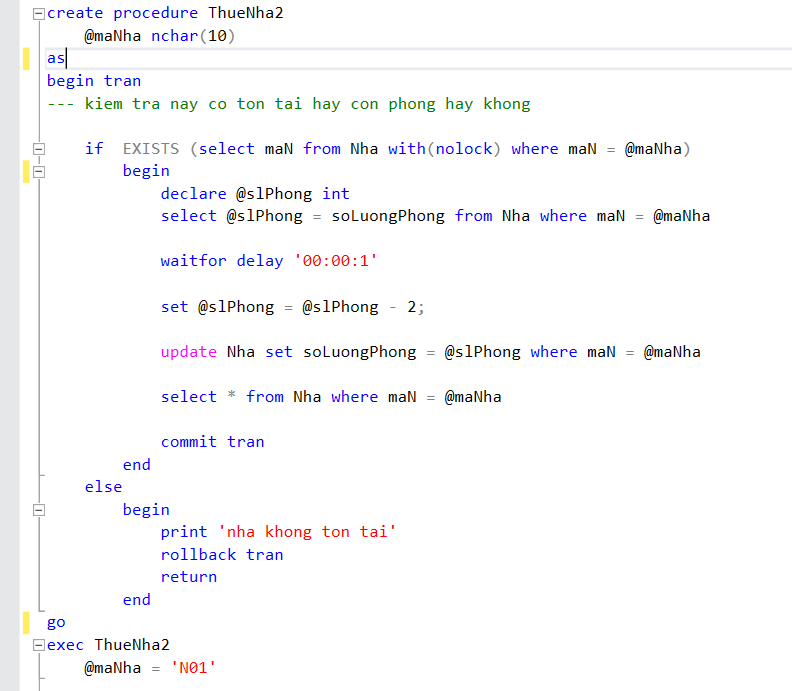
|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang 1) | T2(Khach Hang 2) |
| 1 | Begin tran | Begin tran |
| 2 | Select \* from Nha where maN = N1 | Select \* from Nha where maN=N1 |
| 3 | Running… | Running… |
| 4 | Hoàn tất thủ tục thuê. | Hoàn tất thủ tục thuê. |
| 5 | Update soLuongPhong -1. | Update soLuongPhong -1. |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

Mô phỏng:

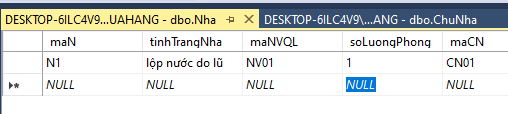
TRANSACTION 1:



TRANSACTION 2:



Chẳng hạn có bộ dữ liệu về nhà như sau:



Hai khách hàng cùng muốn thuê mã nhà N1 tuy nhiên số phòng chỉ có 1, sau khi cả hai thực hiện thanh toán thành công, thì soLuongPhong phải trừ đi 1, tuy nhiên kết quả sẽ chỉ ra là soLuongPhong = 0 trong khi soLuongPhong đáng lẽ phải =-1 => lỗi Lost update.

Giải quyết lỗi: **SỬ DỤNG REPEATABLE READ**

Khi T1( khách hàng 1) thực hiện thao tác select, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2(khách hàng 2) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang 1) | T2(Khach Hang 2) |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ | Begin tran |
| 2 | Select \* from Nha where maN = N1  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 | Select \* from Nha where maN=N1 |
| 3 | Running… | Running… |
| 4 | Hoàn tất thủ tục thuê. | Hoàn tất thủ tục thuê. |
| 5 | Update soLuongPhong -1. | Update soLuongPhong -1.  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

Ở T1 sau khi giữ khóa S và số lượng phòng trừ đi 1(VD: 2-1 = 1) thì lúc này T2 mới thực hiện việc cập nhật và số phòng mới trừ tiếp đi 1(1-1=0).

**Lỗi lost update 2:**

* Công ty A trong lúc nhầm lẫn đã phân công nhân viên B cùng lúc ở 2 chi nhánh.

Transaction 1: phanCongNV(T1)

Input : @maNV, @maChiNhanh

Output : Chi nhánh có nhân viên vừa được phân công.

* Check xem mã nhân viên phụ trách có tồn tại hay không.(@maNV)
* Check xem mã chi nhánh có tồn tại hay không.(@maChiNhanh)
* Insert nhân viên vào chi nhánh.
* Xuất danh sách chi nhánh có nhân viên vừa được phân công.
* Commit tran.

Transaction 2 : phanCongNV(T2)

Input : @maNV, @maChiNhanh

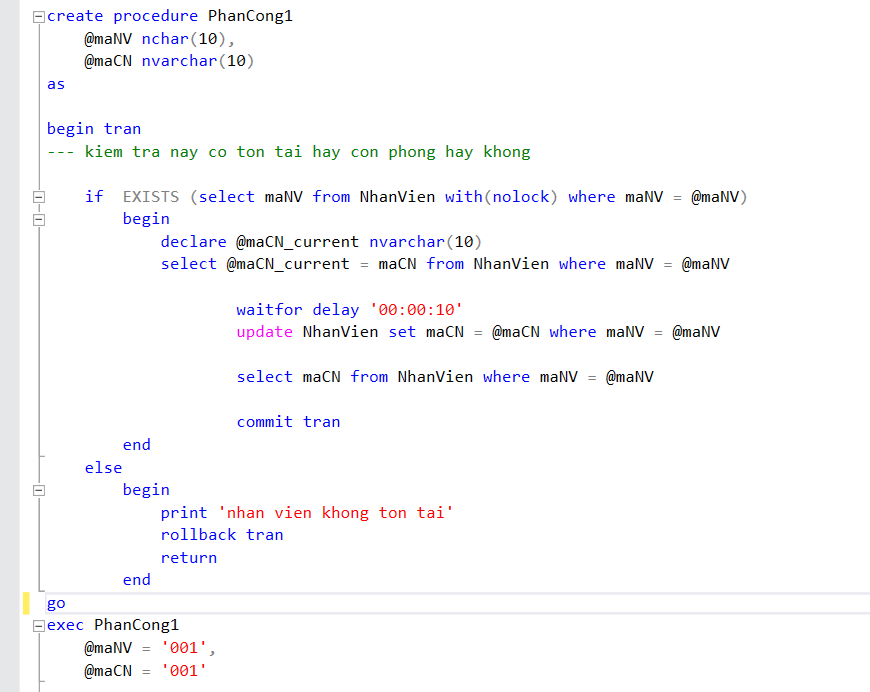
Output : Chi nhánh có nhân viên vừa được phân công.

* Check xem mã nhân viên phụ trách có tồn tại hay không.(@maNV)
* Check xem mã chi nhánh có tồn tại hay không.(@maChiNhanh)
* Insert nhân viên vào chi nhánh.
* Xuất danh sách chi nhánh có nhân viên vừa được phân công.
* Commit tran.

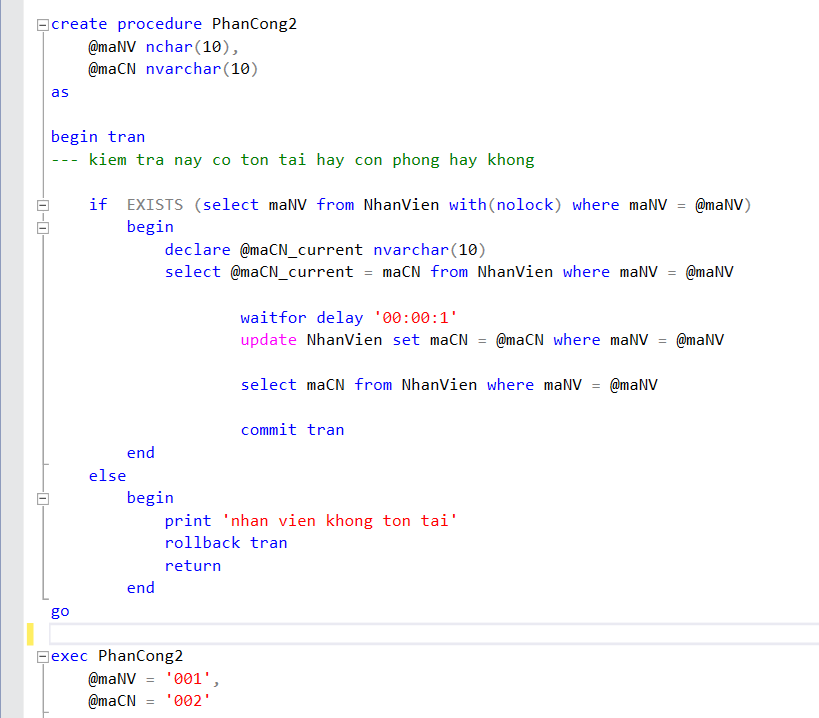
|  |  |  |
| --- | --- | --- |
|  | T1(Nhan Vien) | T2(Nhan Vien) |
| 1 | Begin tran | Begin tran |
| 2 | Insert NhanVienA to CN1 | Insert NhanVienA to CN2 |
| 3 | Running… | Running… |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

Mô phỏng lỗi :

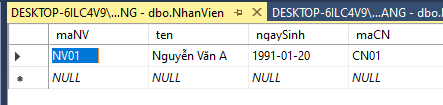
TRANSACTION 1:



TRANSACTION 2:



Nhân viên NV01 được phân công vào cả CN01



Và CN02:



* Lỗi lost update.

Giải quyết lỗi: **SỬ DỤNG Serializable**

Khi T1( Nhân viên 1) thực hiện thao tác insert, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( Nhân viên 2) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Nhan Vien) | T2(Nhan Vien) |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL SERIALIZABLE | Begin tran |
| 2 | Insert NhanVienA to CN1  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 | Insert NhanVienA to CN2  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |
| 3 | Running… | Running… |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi lost update 03:**

* Chủ nhà muốn cập nhật thông tin về nhà thuê mới tuy nhiên lại bấm nhầm cập nhật 2 lần.

Transaction 1: themNhaMoi(T1)

Input : @maNha, @maLoaiNha,@maChuNha

Output : Thêm thành công nhà cho thuê mới.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã loại nhà có tồn tại hay không.(@maLoaiNha)
* Thêm thành công nhà cho thuê mới..
* Commit tran.

Transaction 2: themNhaMoi(T2)

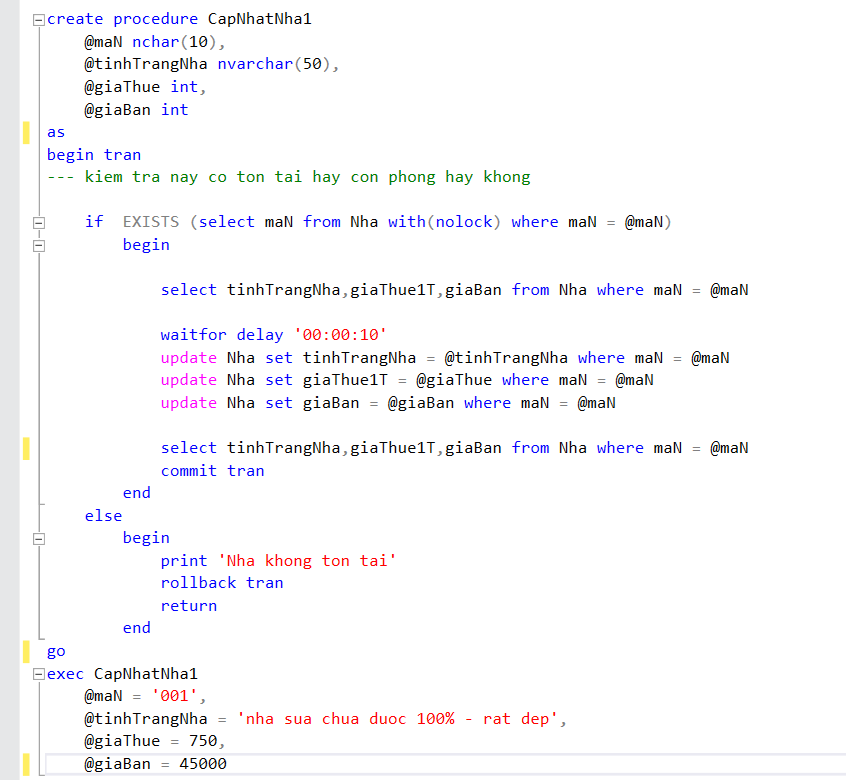
Input : @maNha, @maLoaiNha,@maChuNha

Output : Thêm thành công nhà cho thuê mới.

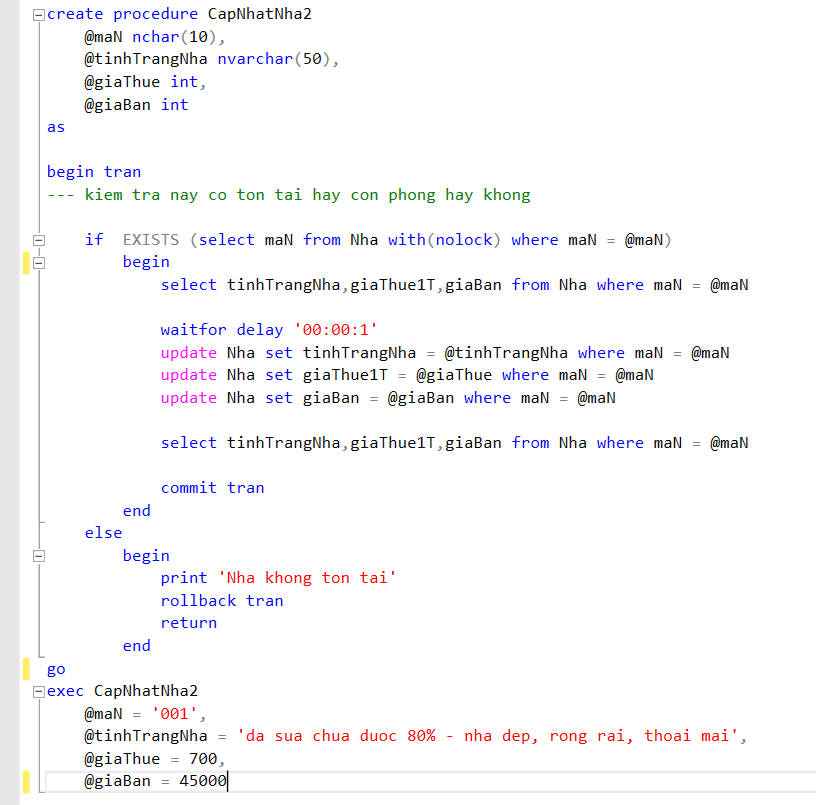
* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã loại nhà có tồn tại hay không.(@maLoaiNha)
* Thêm thành công nhà cho thuê mới..
* Commit tran.

|  |  |  |
| --- | --- | --- |
|  | T1(Nha 1 ) | T2(Nha 1) |
| 1 | Begin tran | Begin tran |
| 2 | Update N1 to Nha | Update N1 to Nha |
| 3 | Running… | Running… |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**TRANSACTION 1:**

****

**TRANSACTION 2:**

****

Giải quyết lỗi: **SỬ DỤNG REPEATABLE READ**

Khi T1( Chủ nhà) thực hiện thao tác update, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( chủ nhà ) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Nha 1 ) | T2(Nha 1) |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ | Begin tran |
| 2 | Update N1 to Nha  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 | Update N1 to Nha  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |
| 3 | Running… | Running… |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi Unrepeatable Read 1:**

* Khách ang đang xem thông tin của chủ nhà, trong lúc này chủ nhà thay đổi số điện thoại của mình và commit, sau khi khách ang xem lại thì số điện thoại đã thay đổi.

Transaction 1: thaydoiThongTin(T1)

Input : @maChuNha, @sDTChuNha

Output : Thêm thành công số điện thoại mới.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Thêm thành công số điện thoại ..
* Commit tran.

Transaction 2 : docThongTin(T2)

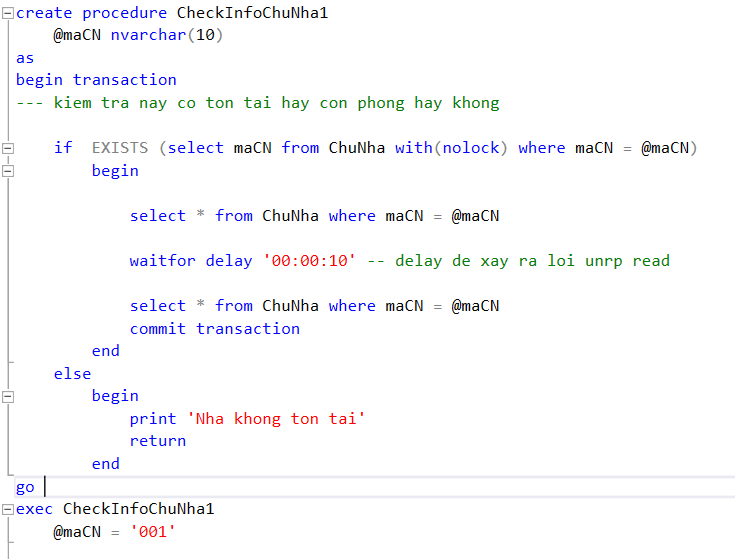
Input : @maNha, @maChuNha, @sDTChuNha

Output : Xuất thông tin của chủ nhà.

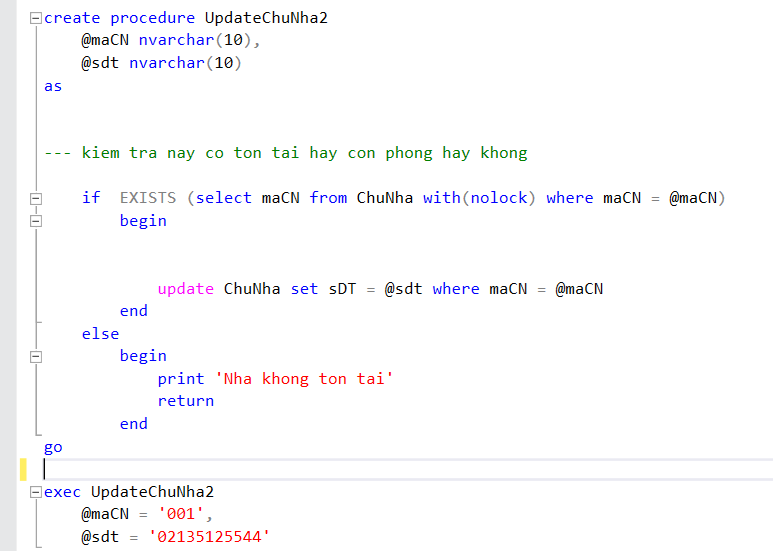
* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Commit tran.

|  |  |  |
| --- | --- | --- |
|  | T2(Khach Hang) | T1(Chu Nha 1) |
| 1 | Begin tran | Begin tran |
| 2 | Read sDTChuNha from ChuNha |  |
| 3 | Running… | update sDTChuNha to ChuNha |
| 4 | Read sDTChuNha from ChuNha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |
|  |  |  |

TRANSACTION 1:



TRANSACTION 2:



Giải quyết lỗi: **SỬ DỤNG REPEATABLE READ**

Khi T1( chủ nhà ) thực hiện thao tác update, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( chủ nhà ) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T2(Khach Hang) | T1(Chu Nha 1) |
| 1 | Begin tran | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ |
| 2 | Read sDTChuNha from ChuNha  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |  |
| 3 | Running… | update sDTChuNha to ChuNha  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 |
| 4 | Read sDTChuNha from ChuNha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi Unrepeatable Read 2:**

* Khách hàng đăng ký xem nhà thành công, tuy nhiên chủ nhà lại xóa nhà thuê ra khỏi danh sách.

Transaction 1: xoaNha(T1)

Input :@maNha, @maChuNha, @sDTChuNha

Output : Thêm thành công số điện thoại mới.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Xóa nhà khỏi hệ thống.
* Commit tran.

Transaction 2: dangKyThue(T2)

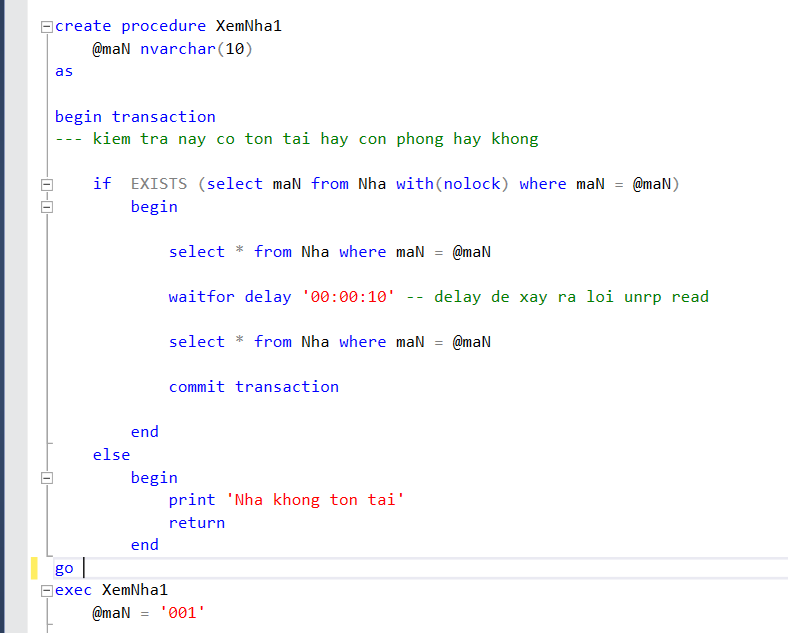
Input : @maNha, @maChuNha, @sDTChuNha

Output : Xuất thông tin của nhà.

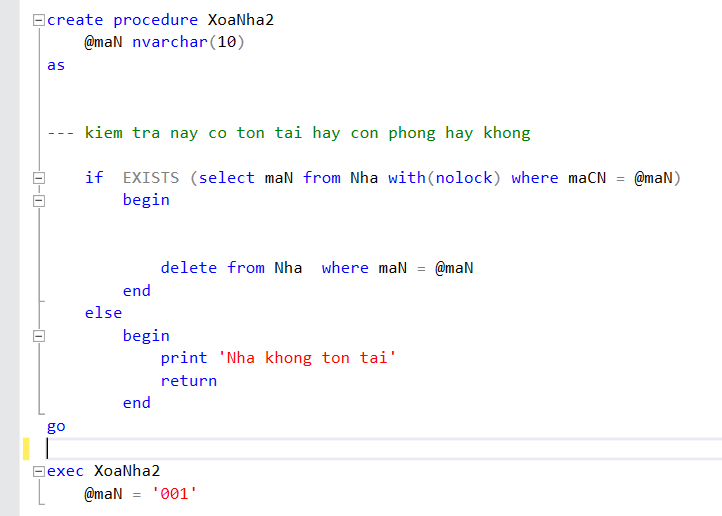
* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Commit tran.

|  |  |  |
| --- | --- | --- |
|  | T2(Khach Hang) | T1(Chu Nha 1) |
| 1 | Begin tran | Begin tran |
| 2 | Read N1 from Nha |  |
| 3 | Running… | Delete N1 from Nha |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

TRANSACTION 1:



TRANSACTION 2:



Giải quyết lỗi: **SỬ DỤNG REPEATABLE READ**

Khi T1( chủ nhà ) thực hiện thao tác delete, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( khách hàng ) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T2(Khach Hang) | T1(Chu Nha 1) |
| 1 | Begin tran | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ |
| 2 | Read N1 from Nha  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |  |
| 3 | Running… | Delete N1 from Nha  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi Unrepeatable Read 3:**

* Khách hàng đăng ký xem nhà với yêu cầu khu vực ngoại ô, tuy nhiên chủ nhà lại thay đổi lại từ ngoại ô thành chợ vì nhầm lẫn.

Transaction 1: doiKhuVuc(T1)

Input :@maNha, @maChuNha, @sDTChuNha

Output : Thay đổi thành công khu vực mới.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Thay đổi khu vực nhà thành gần chợ.
* Commit tran.

Transaction 2: dangKyThueTheoYeuCau(T2)

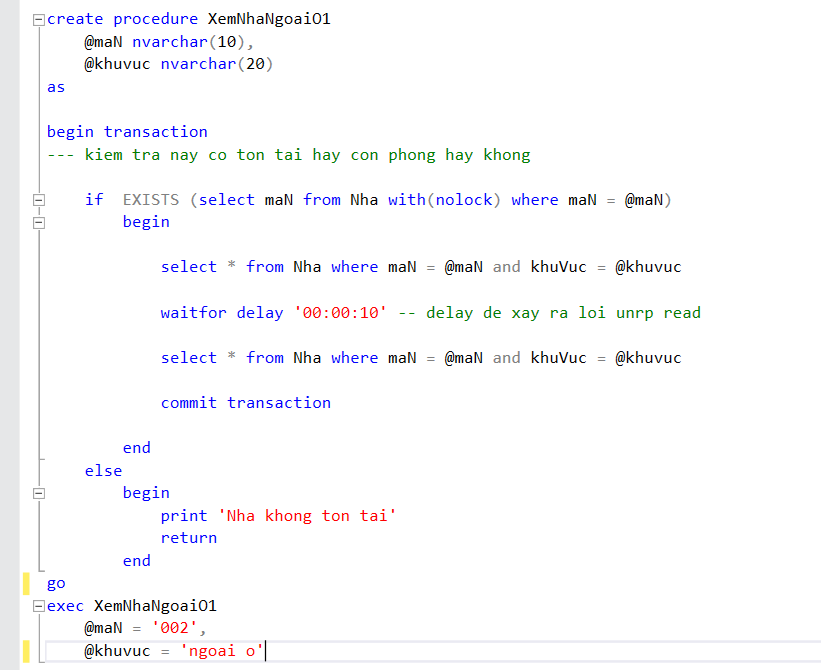
Input : @maNha, @maChuNha, @sDTChuNha

Output : Xuất thông tin của nhà.

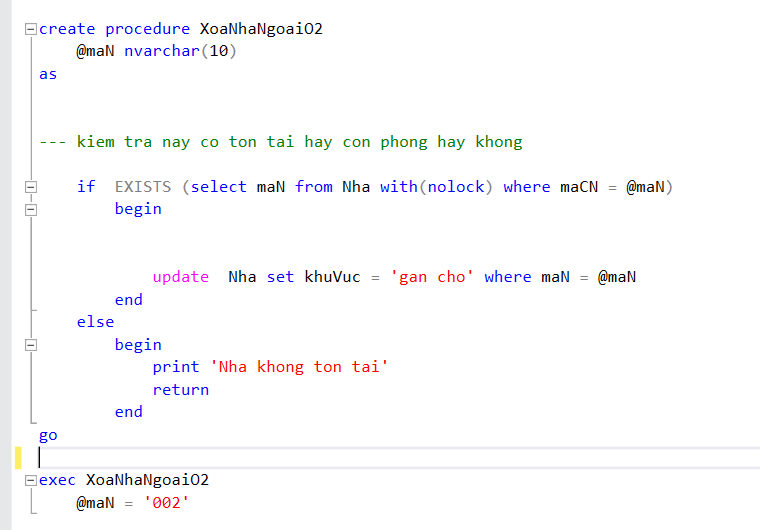
* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Check khu vực có gần ngoại ô
* Xuất danh sách các nha gần ngoại ô
* Commit tran.

|  |  |  |
| --- | --- | --- |
|  | T2(Khach Hang) | T1(Chu Nha 1) |
| 1 | Begin tran | Begin tran |
| 2 | Read \* from Nha where KhuVuc=”Ngoại ô” |  |
| 3 | Running… | Update N1 from Nha where KhuVuc=”gần chợ” |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

TRANSACTION 1:



TRANSACTION 2:



Giải quyết lỗi: **SỬ DỤNG REPEATABLE READ**

Khi T1( chủ nhà ) thực hiện thao tác update, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( khách hàng ) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T2(Khach Hang) | T1(Chu Nha 1) |
| 1 | Begin tran | Begin tran  SET TRAN ISOLATION LEVEL REPEATABLE READ |
| 2 | Read \* from Nha where KhuVuc=”Ngoại ô”  T2: xin khóa X, SQL không cấp khóa X do T1 đang giữ khóa. |  |
| 3 | Running… | Update N1 from Nha where KhuVuc=”gần chợ”  T1: giữ khóa S  SQL cấp khóa S  T1: giữ khóa cho đến khi hết T1 |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi Phantom 1:**

* Khách hàng yêu cầu nhà cho thuê phải có số phòng <2, tuy nhiên sau khi hoàn tất thủ tục chủ nhà lại xây dựng thêm 1 phòng.

Transaction 1: dangKyThueTheoYeuCau(T1)

Input : @maNha, @maChuNha, @sDTChuNha

Output : Xuất thông tin của nhà.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Check số phòng <2.
* Xuất danh sách các nhà theo yêu cầu.
* Commit tran.

Transaction 2: themPhong(T2)

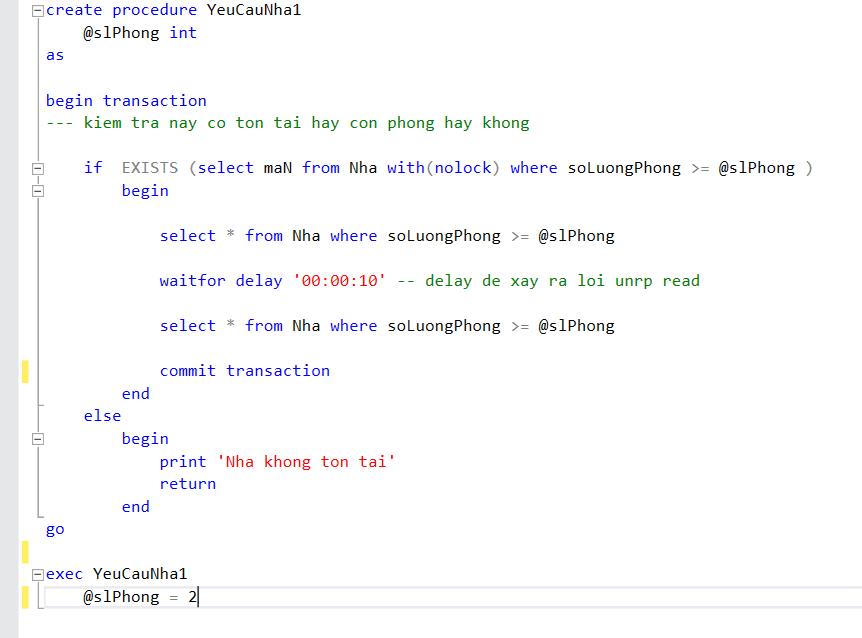
Input :@maNha, @maChuNha, @sDTChuNha

Output : Thay đổi thành công số phòng.

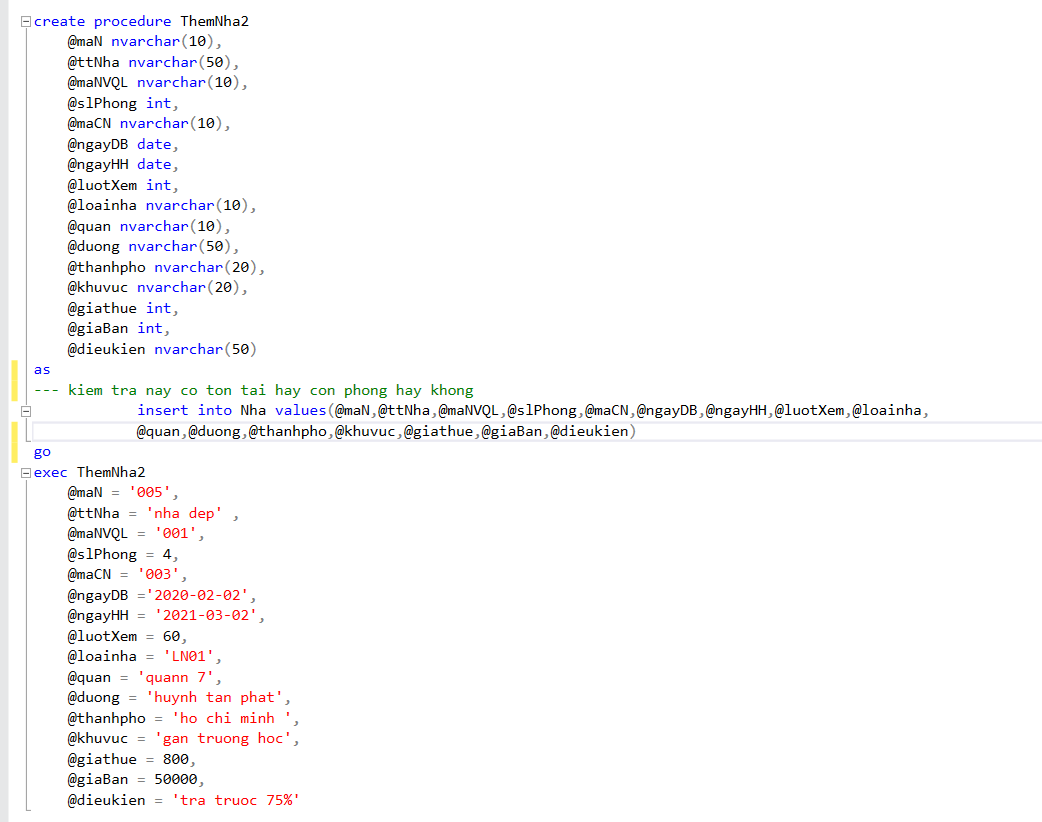
* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Insert số phòng trong nhà thêm 1 .
* Commit tran.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2(Chu Nha 1) |
| 1 | Begin tran | Begin tran |
| 2 | Select \* from Nha where soLuongPhong<2 |  |
| 3 | Running… | Insert N1 from Nha where soLuongPhong+=1 |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

TRANSACTION 1:



TRANSACTION 2:



Giải quyết lỗi: **SỬ DỤNG Serializable**

Khi T1( Khách hàng ) thực hiện thao tác tìm kiếm, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( chủ nhà) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2(Chu Nha 1) |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL SERIALIZABLE | Begin tran |
| 2 | Select \* from Nha where soLuongPhong<2  T1 : Xin khóa S  SQL : Cấp khóa S  T1 : giữ khóa S đến hết giao tác  Ngăn chèn dữ liệu vào tập đang khóa |  |
| 3 | Running… | Insert N1 from Nha where soLuongPhong+=1  T2: Xin khóa X  SQL : Không cấp khóa X do T1  ngăn không cho insert vào |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi Phantom 2:**

* Khách hàng select các căn nhà với giá tiền thuê 1 tháng <1.000.000Đ(tìm thấy 4 căn), tuy nhiên, khi đã commit thì chủ nhà thứ 4 lại tăng giá lên >1.000.000đ.

Transaction 1: dangKyThueTheoYeuCau(T1)

Input : @maNha, @maChuNha, @sDTChuNha

Output : Xuất thông tin của nhà.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Check giaThue1Thang <1000000.
* Xuất danh sách các nhà theo yêu cầu.
* Commit tran.

Transaction 2: giamGiaTienThue(T2)

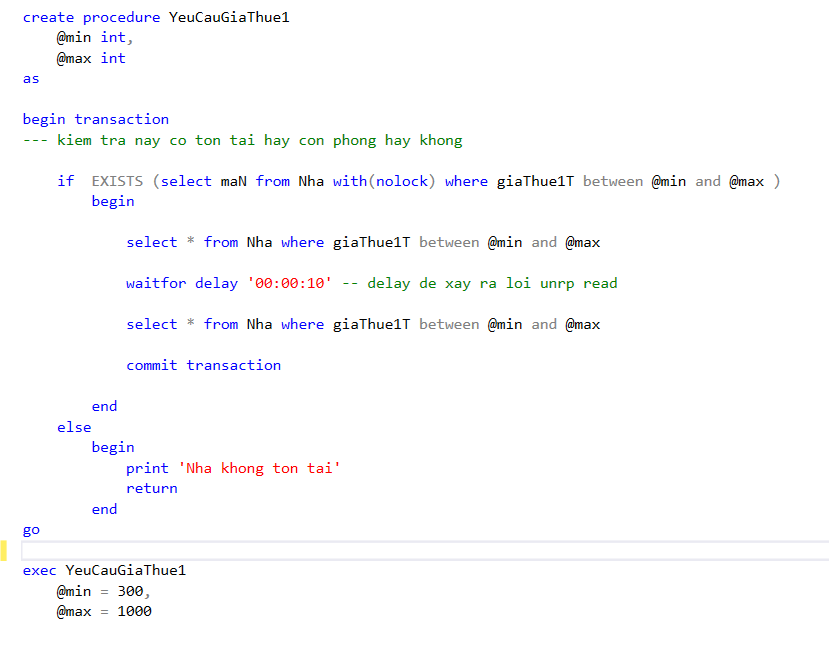
Input :@maNha, @maChuNha, @sDTChuNha

Output : Thay đổi thành công số tiền cho thuê.

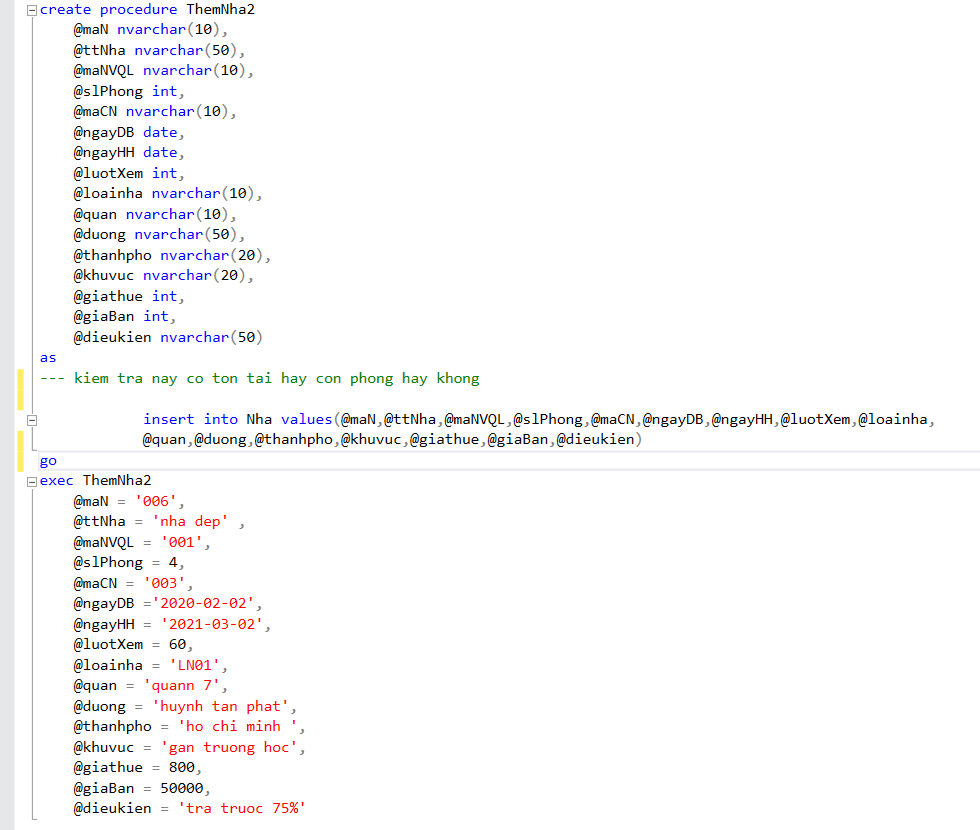
* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Thay đổi giá nhà theo tháng lên .
* Commit tran

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2(He Thong) |
|  |  |  |
| 1 | Begin tran | Begin tran |
| 2 | Read \* from Nha where giaThue1Thang>1000000 |  |
| 3 | Running… | Insert N1 from Nha where giaThue1Thang-- |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

TRANSACTION 1:



TRANSACTION 2:



Giải quyết lỗi: **SỬ DỤNG Serializable**

Khi T1( Khách hàng ) thực hiện thao tác tìm kiếm, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( chủ nhà) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2(He Thong) |
|  |  |  |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL SERIALIZABLE | Begin tran |
| 2 | Read \* from Nha where giaThue1Thang<1000000  T1 : Xin khóa S  SQL : Cấp khóa S  T1 : giữ khóa S đến hết giao tác  Ngăn chèn dữ liệu vào tập đang khóa |  |
| 3 | Running… | Insert giaThue1Thang++ from Nha where maN=”N1”  T2: Xin khóa X  SQL : Không cấp khóa X do T1 |
| 4 | Read N1 from Nha |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

**Lỗi Phantom 3:**

* Khách hàng đang xem những nhà ở khu vực thành phố Hồ Chi Minh, trong lúc đó chủ nhà vừa thêm vào danh sách thêm 1 nhà thuê.

Transaction 1: dangKyThueTheoYeuCau(T1)

Input : @maNha, @maChuNha, @sDTChuNha

Output : Xuất thông tin của nhà.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Check thành phố : Hồ Chí Minh.
* Xuất danh sách các nhà theo yêu cầu.
* Commit tran.

Transaction 2: themNha(T2)

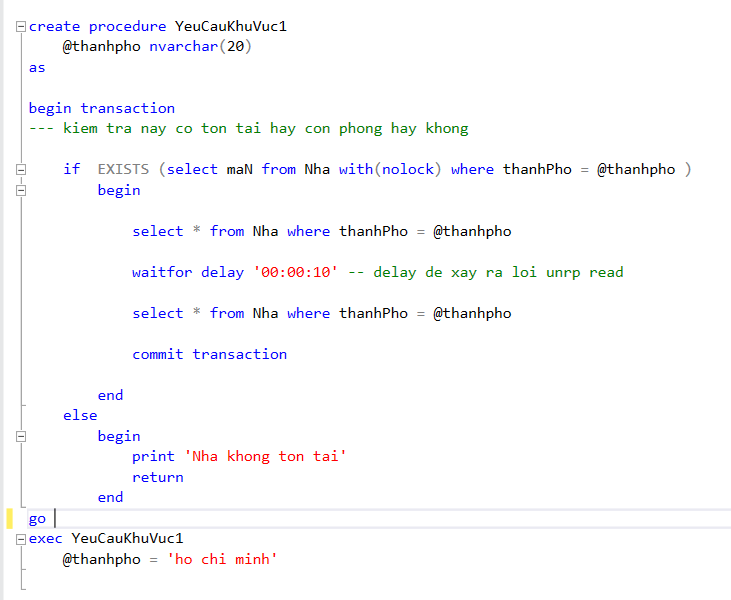
Input :@maNha, @maChuNha, @sDTChuNha

Output : Thay đổi thành công số phòng.

* Check xem mã nhà có tồn tại hay không.(@maNha)
* Check xem mã chủ nhà có tồn tại hay không.(@maChuNha)
* Thêm nhà cho thuê thuộc Thành Phố Hồ Chí Minh.
* Commit tran.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2(Chu Nha) |
| 1 | Begin tran | Begin tran |
| 2 | Read \* from Nha where thanhPho=”Hồ Chí Minh” |  |
| 3 | Running… | Insert N1 from Nha where thanhPho=”Hồ Chi Minh” |
| 4 | Read \* from Nha where thanhPho=”Hồ Chí Minh” |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

TRANSACTION 1:



TRANSACTION 2:



Giải quyết lỗi: **SỬ DỤNG Serializable**

Khi T1( Khách hàng ) thực hiện thao tác tìm kiếm, T1 sẽ xin khóa S(SQL cho phép ), trong lúc này khi T2( chủ nhà) xin khóa X thì SQL sẽ không cấp phép cho T2, vì T1 đang giữ khóa X.

|  |  |  |
| --- | --- | --- |
|  | T1(Khach Hang) | T2(Chu Nha) |
| 1 | Begin tran  SET TRAN ISOLATION LEVEL SERIALIZABLE | Begin tran |
| 2 | Read \* from Nha where thanhPho=”Hồ Chí Minh”  T1 : Xin khóa S  SQL : Cấp khóa S  T1 : giữ khóa S đến hết giao tác  Ngăn chèn dữ liệu vào tập đang khóa |  |
| 3 | Running… | Insert N1 from Nha where thanhPho=”Hồ Chi Minh”  T2: Xin khóa X  SQL : Không cấp khóa X do T1  ngăn không cho insert vào |
| 4 | Read \* from Nha where thanhPho=”Hồ Chí Minh” |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 | Commit tran | Commit tran |

# ỨNG DỤNG DEMO

## *Mô tả cấu trúc ứng dụng*

* [Mô tả sitemap của website]
* [Thông tin của link download project, phim…]

## *Tình huống tranh chấp đồng thời*

* [Với mỗi tình huống tranh chấp chụp hình giao diện của các chức năng gây tranh chấp (có dữ liệu test)]

## Bộ dữ liệu test các tình huống

* [Mô tả bộ dữ liệu test tình huống tranh chấp trên ứng dụng]
* Tình huống gây lỗi 1:
  + Chức năng gì
  + Dữ liệu là gì
  + Kết quả là gì

# TỔNG KẾT

* [Tự đánh giá kết quả đạt được và những gì chưa đạt được của nhóm]