

TRƯỜNG ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN

ĐỀ TÀI: HỆ THỐNG QUẢN LÝ BÁN/CHO THUÊ NHÀ

NHÓM 6

ĐOÀN NAM THUẬN - 18126034

LÊ THỊ ANH THI – 18126033

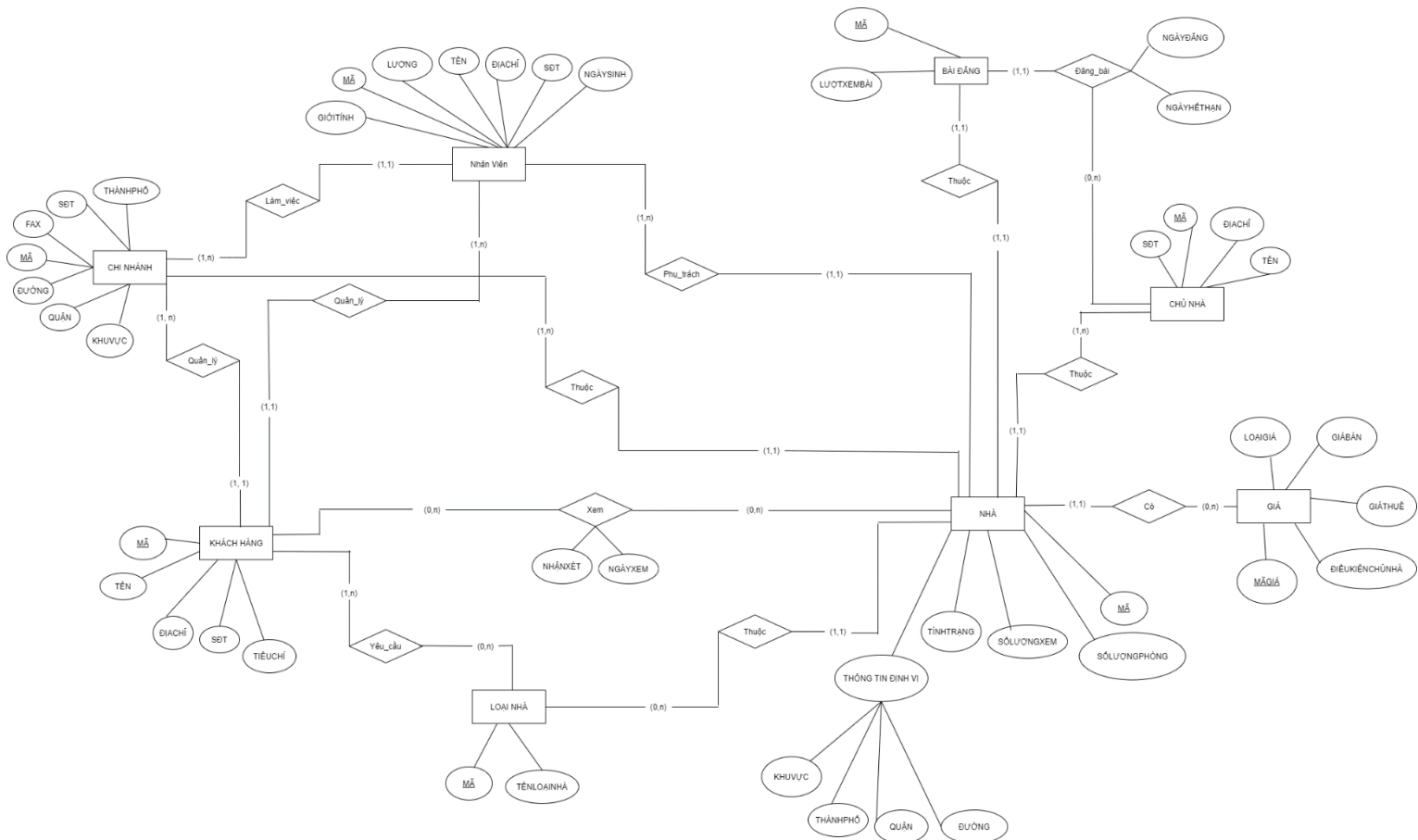
LƯƠNG LONG HÀ - 18126014

MÔN HỌC: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Hồ Chí Minh – 2021

I. Thiết kế và cài đặt CSDL

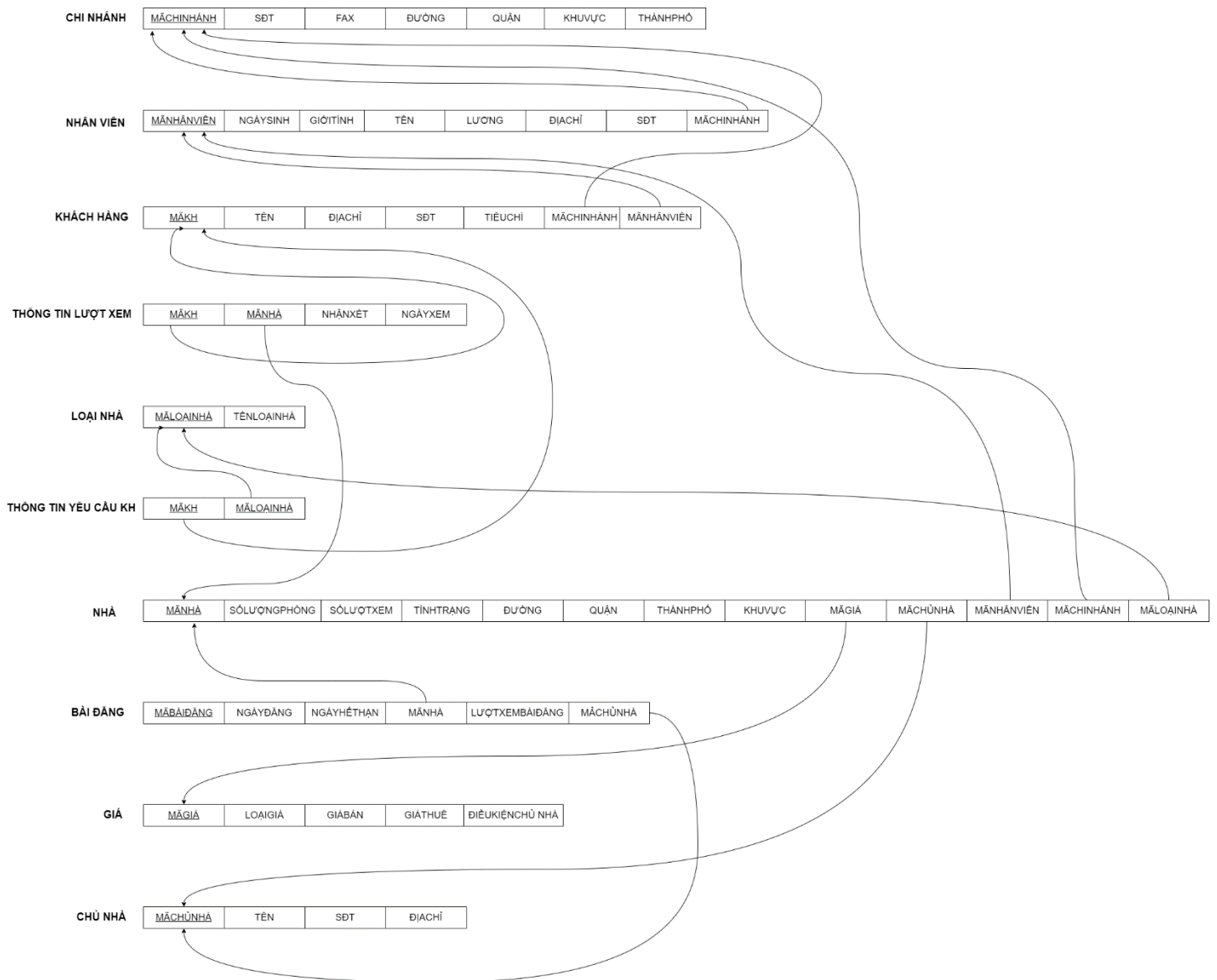
1. Bảng thiết kế ER



❖ Link hình HD:

<https://drive.google.com/file/d/1fT7Ypv00ixNM-WZRlgGuj6-Xrk2hlYoq/view?usp=sharing>

2. Lược đồ quan hệ



❖ **Link hình HD:**

<https://drive.google.com/file/d/1fT8Iq5fvYdFpx4dkfNNTsDJoUnLpDzvs/view?usp=sharing>

3. Các ràng buộc toàn vẹn

❖ CHI NHÁNH

1. Mỗi chi nhánh có một mã chi nhánh duy nhất

❖ NHÂN VIÊN

1. Mỗi nhân viên có một mã nhân viên duy nhất (MANHANVIEN)
2. Giới tính của nhân viên (GIOITINH) phải là “Nam” hoặc “Nữ”
3. Tên của nhân viên không được phép NULL (TEN)
4. Mỗi nhân viên có lương $\geq 1,000,000$
5. Nhân viên làm việc cho một chi nhánh (MACHINHANH)

❖ KHÁCH HÀNG

1. Mỗi khách hàng có một mã khách hàng duy nhất (MAKHACHHANG)
2. Tên của nhân viên không được phép NULL (TEN)
3. Mỗi khách hàng được một chi nhánh quản lý (MACHINHANH)
4. Mỗi khách hàng được quản lý bởi một nhân viên (MANHANVIEN)

❖ THONGTINLUOTXEM

1. Mỗi khách hàng (MAKH) có thể thể xem nhiều nhà khác nhau (MANHA) vào các ngày khác nhau (NGAYXEM)

❖ LOAINHA

1. Mỗi loại nhà có một mã loại nhà (MALOAINHA)
2. Tên của nhà không được phép NULL (TEN)

❖ THONGTINYEUCAUKH

1. Mỗi khách hàng có thể yêu cầu nhiều nhà loại khác nhau (MALOAINHA)

❖ NHA

1. Mỗi nhà có một mã nhà duy nhất (MANHA)
2. Mỗi nhà có một giá thuê, một giá bán (MAGIA)
3. Mỗi nhà thuộc quyền sở hữu của một chủ nhà (MACHUNHA)
4. Mỗi nhà được một nhân viên phụ trách (MANHANVIEN)
5. Mỗi nhà được một chi nhánh quản lý (MACHINHANH)
6. Mỗi nhà thuộc một về một loại nhà (MALOAINHA)
7. Mỗi nhà có số lượng phòng > 0 (SOLUONGPHONG)

❖ CHU NHA

1. Mỗi chủ nhà có một mã chủ nhà duy nhất (MACHUNHA)
2. Tên của chủ nhà (TEN) không được phép NULL

❖ BAI DANG

1. Mỗi bài đăng có một mã bài đăng (MABAIDANG)
2. Ngày đăng (NGAYDANG) phải luôn nhỏ hơn ngày hết hạn (NGAYHETHAN)
3. Mỗi bài đăng có thể đăng duy nhất về một nhà (MANHA) và thuộc về một chủ nhà (MACHUNHA)

❖ GIA

1. Mỗi giá nhà có một mã giá (MAGIA)
2. Loại giá (LOAIGIA) của giá nhà phải là “THUÊ” hoặc “BÁN”
3. Nếu loại giá (LOAIGIA) là “THUÊ” thì giá bán (GIABAN) bằng NULL và điều kiện chủ nhà (DIEUKIENCHUNHA) bằng NULL
4. Nếu loại giá (LOAIGIA) là “BÁN” thì giá thuê (GIATHUE) bằng NULL
5. Giá bán (GIABAN) và giá thuê (GIATHUE) phải luôn ≥ 0

4. Xác định các loại người dùng

- Ứng dụng này có các loại người dùng:

- Admin
- Nhân viên
- Chủ nhà
- Khách hàng

II. Xác định chức năng của hệ thống

	Tên chức năng	Mô tả chức năng	Người thực hiện
1.	Đặt lịch xem nhà	Trong ứng dụng, với mỗi nhà thì khách hàng có thể đặt lịch để xem nhà trực tiếp nhờ chức năng này. Sau khi khách hàng nhập vào các thông tin cần thiết như ngày xem, giờ xem, ... thì hệ thống sẽ thêm một lịch xem nhà gồm các thông tin về nhà, khách hàng, ngày xem, giờ xem, ... vào cơ sở dữ liệu.	Khách hàng
2.	Hủy lịch xem nhà	Hệ thống sẽ hủy một lịch xem nhà cụ thể do khách hàng yêu cầu.	Khách hàng
3.	Cập nhật lịch xem nhà	Hệ thống sẽ cập nhật lại thông tin của lịch xem nhà do khách hàng yêu cầu.	Khách hàng
4.	Xem thông tin của lịch xem nhà	Chức năng này cho phép người dùng xem thông tin về lịch xem nhà : - Khách hàng: xem lại thông tin lịch xem nhà mà họ đã đặt. - Chủ nhà: xem thông tin về lịch xem nhà mà những căn nhà thuộc quyền sở hữu của họ đã được đặt lịch xem. - Nhân viên: xem thông tin về lịch xem nhà mà những căn nhà do họ phụ trách được đặt lịch xem.	Chủ nhà, Khách hàng, Nhân viên
5.	Gửi yêu cầu thuê hoặc mua	Khách hàng sẽ gửi yêu cầu thông tin về nhà mà họ muốn thuê hoặc mua (vd: nhu cầu muốn mua hay thuê, loại nhà, số lượng phòng, địa chỉ, ...). Sau khi khách hàng gửi yêu cầu thành công thì hệ thống sẽ thêm một yêu cầu thuê/mua của khách hàng vào cơ sở dữ liệu.	Khách hàng
6.	Hủy yêu cầu thuê hoặc mua	Hệ thống sẽ hủy cụ thể một yêu cầu thuê hoặc mua nhà do khách yêu cầu.	Khách hàng
7.	Cập nhật yêu cầu thuê hoặc mua	Hệ thống sẽ cập nhật cụ thể một yêu cầu thuê hoặc mua nhà do khách yêu cầu.	Khách hàng


8.	Xem thông tin yêu cầu thuê/mua của khách hàng	<p>Chức năng này cho phép người dùng xem thông tin về yêu cầu thuê/mua nhà của khách hàng:</p> <ul style="list-style-type: none"> - Khách hàng: xem lại thông yêu cầu thuê/mua của họ. - Nhân viên: xem thông tin về yêu cầu thuê/mua của khách hàng mà nhân viên đó quản lý. 	Khách hàng, Nhân viên
9.	Thêm bài đăng	Với mỗi căn nhà, chủ nhà thực hiện đăng bài với các thông tin về nhà, giá thuê, giá bán, ngày đăng, ngày hết hạn, ... Sau khi chủ nhà đăng bài thành công thì hệ thống sẽ thêm một bài đăng vào cơ sở dữ liệu.	Chủ nhà
10.	Xóa bài đăng	Hệ thống sẽ xóa cụ thể một bài đăng theo yêu cầu của chủ nhà.	Chủ nhà, Quản trị viên
11.	Cập nhật bài đăng	Hệ thống sẽ cập nhật cụ thể một bài đăng theo yêu cầu của chủ nhà.	Chủ nhà
12.	Thêm khách hàng	Quá trình nhập thông tin khách hàng mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào	Khách hàng, Quản trị viên
13.	Xóa khách hàng	Xóa một thông tin khách hàng ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra khách hàng đó có tồn tại trong hệ thống hay không. Còn lịch xem nhà, thông tin yêu cầu thuê/mua tham chiếu đến hay không.	Quản trị viên
14.	Cập nhật thông tin khách hàng	Chỉnh sửa lại thông tin của khách hàng. Kiểm tra thông tin của khách hàng, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Khách hàng, Quản trị viên
15.	Xem thông tin khách hàng	Xem thông tin của một khách hàng cụ thể.	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên
16.	Thêm chủ nhà	Quá trình nhập thông tin chủ nhà mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào	Chủ nhà, Quản trị viên
17.	Xóa chủ nhà	Xóa một thông tin chủ nhà ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra chủ nhà đó có tồn tại trong hệ thống hay không. Còn nhà, bài đăng tham chiếu đến hay không.	Quản trị viên

18.	Cập nhật thông tin chủ nhà	Chỉnh sửa lại thông tin của chủ nhà. Kiểm tra thông tin của chủ nhà, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Chủ nhà, Quản trị viên
19.	Xem thông tin chủ nhà	Xem thông tin của một chủ nhà cụ thể	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên
20.	Thêm nhân viên	Quá trình nhập thông tin nhân viên mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào	Chủ nhà, Quản trị viên
21.	Xóa nhân viên	Xóa một thông tin nhân viên ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra nhân viên đó có tồn tại trong hệ thống hay không. Còn khách hàng, nhà tham chiếu đến hay không.	Quản trị viên
22.	Cập nhật thông tin nhân viên	Chỉnh sửa lại thông tin của nhân viên. Kiểm tra thông tin của nhân viên, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Chủ nhà, Quản trị viên
23.	Xem thông tin nhân viên	Xem thông tin của một nhân viên cụ thể	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên
24.	Thêm nhà	Quá trình nhập thông tin nhà mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào	Chủ nhà, Quản trị viên
25.	Xóa nhà	Xóa một thông tin nhà ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra nhà đó có tồn tại trong hệ thống hay không. Còn bài đăng, lịch xem nhà tham chiếu đến hay không.	Quản trị viên, Chủ nhà
26.	Cập nhật thông tin chủ nhà	Chỉnh sửa lại thông tin của nhà. Kiểm tra thông tin của nhà, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Chủ nhà, Quản trị viên
27.	Xem thông tin nhà	Xem thông tin của một nhà cụ thể	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên
28.	Thêm chi nhánh	Quá trình nhập thông tin chi nhánh mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào	Quản trị viên
29.	Xóa chi nhánh	Xóa một thông tin chi nhánh ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra chi nhánh đó có tồn tại trong hệ	Quản trị viên

		thống hay không. Còn nhân viên, khách hàng, nhà tham chiếu đến hay không.	
30.	Cập nhật thông tin chi nhánh	Chỉnh sửa lại thông tin của chi nhánh. Kiểm tra thông tin của chi nhánh, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Quản trị viên
31.	Xem thông tin chi nhánh	Xem thông tin của một chi nhánh cụ thể	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên
32.	Thêm loại nhà	Quá trình nhập thông tin loại nhà mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào	Quản trị viên
33.	Xóa loại nhà	Xóa một loại nhà ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra loại nhà đó có tồn tại trong hệ thống hay không. Còn giá trị nào tham chiếu đến hay không.	Quản trị viên
34.	Cập nhật thông tin loại nhà	Chỉnh sửa lại thông tin của loại nhà. Kiểm tra thông tin của loại nhà, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Quản trị viên
35.	Xem thông tin loại nhà	Xem thông tin của một loại nhà cụ thể	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên
36.	Thêm giá nhà	Quá trình nhập giá nhà mới vào cơ sở dữ liệu, chạy các bước kiểm tra dữ liệu đầu vào trước khi thêm vào.	Chủ nhà
37.	Xóa giá nhà	Xóa một giá nhà ra khỏi hệ thống. Trước khi xóa, hệ thống sẽ kiểm tra giá nhà đó có tồn tại trong hệ thống hay không. Còn nhà nào tham chiếu đến hay không.	Chủ nhà
38.	Cập nhật giá nhà	Chỉnh sửa lại giá nhà của một nhà. Kiểm tra thông tin của giá nhà, kiểm tra tính hợp lệ của các dữ liệu đầu vào.	Chủ nhà
39.	Xem giá nhà	Xem thông tin của một giá nhà cụ thể	Quản trị viên, Khách hàng, Chủ nhà, Nhân viên

III. Thiết kế giao diện

❖ Trang chủ



[Tìm mua](#) [Tìm bán](#) [Đăng nhập](#) [Đăng tin](#)

Thông tin nhà

Mua

Địa chỉ

Khu vực

Đúng nhà, đúng giá, đúng thời điểm

Loại nhà

Khoảng giá

Số phòng

Quận/Huyện

Phường/Xã

Đường

[Tìm kiếm](#)

Nhà đang bán

Xem tất cả >>

Tên nhà

Giá

Tên nhà

Giá

Tên nhà

Giá

Tên nhà

Giá

Tên nhà

Giá

Nhà đang cho thuê

Xem tất cả >>

Tên nhà

Giá

Tên nhà

Giá

Tên nhà


Giá

Tên nhà

Giá

Tên nhà

Giá



Admin


Đăng xuất

- Quản lý khách hàng
- Quản lý chủ nhà
- Quản lý nhà
- Quản lý loại nhà
- Quản lý bài đăng

Danh sách khách hàng


+ Thêm khách hàng

❖ Nhân viên



Tên NV

Đăng xuất

 Tên NV

Chi nhánh 1

• Trang chủ

Quản lý nhà

Quản lý khách hàng

Danh sách lịch xem nhà

Danh sách yêu cầu của khách hàng

Hỗ trợ

Điện thoại 09748293312

Email admin@dreamhouse.com

Lịch xem nhà sắp tới

Danh sách yêu cầu của khách hàng

❖ Khách hàng



[Tìm mua](#)

[Tìm bán](#)



Tên KH

[Đăng tin](#)

Trang chủ / Tên nhà

[Quay lại](#) ↩

Hình ảnh nhà

Thông tin chủ nhà

Tên nhà

Thông tin chi tiết nhà

Đặt lịch xem nhà

IV. Các lỗi tranh chấp đồng thời và phương pháp xử lý

1. Lỗi Dirty read

1.1. Dirty read 01.

- **Mô tả kịch bản lỗi:** Trong hệ thống, khi chủ nhà đang cập nhật thông tin nhà, cụ thể là cập nhật loại nhà. Nhưng vì sự cố nên không cập nhật thành công, hệ thống phải rollback lại trạng thái ban đầu. Nhưng cùng lúc đó lại có một khách hàng đang xem danh sách nhà theo loại nhà, tức là lấy dữ liệu đã xử lý trước khi thực hiện rollback.

- **Chi tiết về transaction:**

*** **Transaction 1: capNhatThongTinNha (T1)**

Input: @maNha, @soLuongPhong, @maLoaiNha, @tinhTrang, @duong, @quan, @thanhPho, @khuVuc, @maGia, @maChuNha, @maNhanVien, @maChiNhanh, maLoaiNha

Output: Dòng được cập nhật

1. Kiểm tra sự tồn tại của nhà (@maNha)
2. Kiểm tra sự tồn tại của loại nhà (@maLoaiNha)
3. Cập nhật loại nhà
4. Rollback

*** **Transaction 2: thongKeNhaTheoLoaiNha (T2)**

Input: @maLoaiNha

Output: Các dòng thỏa điều kiện

1. Kiểm tra sự tồn tại của loại nhà (@maLoaiNha)
2. Load danh sách nhà theo loại nhà
3. Commit transaction

ERROR01: DIRTY READ

Trans t	capNhatLoaiNha(T1)	thongKeNhaTheoLoaiNha (T2)
t0	Kiểm tra sự tồn tại của nhà	
t1	Kiểm tra sự tồn tại của loại nhà	Kiểm tra sự tồn tại của loại nhà
t2	Cập nhật nhà	Load danh sách nhà theo loại nhà
t3	Processing	Commit transaction
t4	Processing ...	
t5	Rollback	

- **Mô phỏng lỗi:**

- Dữ liệu ban đầu: Thông tin loại nhà của nhà có *MaNha* = 001 là 001

MaNha	SoLuon...	SoLuotX...	TinhTrang	Duong	Quan	ThanhPho	KhuVuc	MaGia	MaChuN...	MaNhan...	MaChiN...	MaLoaiNha
001	2	3	NULL	An Dươn...	5	HCM	TPHCM	001	001	001	001	001

- Trường hợp lỗi:

+ T1 đang thực hiện update *MaLoaiNha* = 004 của nhà «001», tuy nhiên transaction T1 bị rơi vào trường hợp rollback => thuộc tính *MaLoaiNha* = 001

+ T2 đang xem thông tin nhà «001» trong khi T1 đang update, mặc dù T1 thực hiện không thành công nhưng T2 vẫn hiển thị được *MaLoaiNha* = 004 => Lỗi Dirty Read

- **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T2 là **READ COMMITED**, nhằm mục đích chỉ đọc những dữ liệu đã commit.

ERROR01: DIRTY READ

Trans t	capNhatLoaiNha (T1)	Khóa	thongKeNhaTheoLoaiNha (T2)	Khóa
t0	Kiểm tra sự tồn tại của nhà		Set → READ COMMITED	
t1	Kiểm tra sự tồn tại của loại nhà		Kiểm tra sự tồn tại của loại nhà	
t2	Cập nhật nhà	T1: Xin khóa X SQL: Cấp khóa X	Load danh sách nhà theo loại nhà	T2: Xin khóa S

				SQL: Không cấp khóa S do T1 đang giữ khóa X
t3	Processing		Commit transaction	
t4	Processing ...			
t5	Rollback			

1.2. Dirty read 02.

- **Mô tả kịch bản lỗi:** Khách hàng đang cập nhật lại thông tin lịch đến xem trực tiếp một căn nhà, cụ thể là khách hàng muốn đổi ngày xem. Nhưng vì sự cố nên không cập nhật thành công, hệ thống phải rollback lại trạng thái ban đầu. Cùng lúc đó thì chủ nhà của căn nhà đó vào xem lịch xem nhà.

- **Chi tiết về transaction:**

*** **Transaction 1: capNhatLichXemNha (T1)**

- Cập nhật thông tin lịch xem nhà trực tiếp : UPDATE(THONGTINLUOTXEM)
- *Input: @maKhachHang, @maNha, @ngayXem*
Output: Dòng được cập nhật.
 1. Kiểm tra sự tồn tại của khách hàng (@maKhachHang)
 2. Kiểm tra sự tồn tại của nhà (@maNha)
 3. Kiểm tra ngày xem hợp lệ (@ngayXem)
 4. Cập nhật lịch xem nhà
 5. Rollback

*** **Transaction 2: xemThongTinLichXemNha (T2)**

- Xem thông tin lịch xem nhà trực tiếp : READ(THONGTINLUOTXEM)
- *Input: @maNha*
Output: Các dòng thỏa điều kiện
 1. Kiểm tra sự tồn tại của nhà
 2. Load danh sách lịch xem nhà
 3. Commit transaction.

ERROR02: DIRTY READ

Trans t	capNhatLichXemNha (T1)	xemThongTinLichXemNha(T2)
t0	Kiểm tra sự tồn tại của khách hàng	
t1	Kiểm tra sự tồn tại của nhà	
t2	Kiểm tra ngày xem hợp lệ	
t3	Cập nhật lịch xem nhà	Kiểm tra sự tồn tại của nhà
t4	Processing	Load danh sách lịch xem nhà
t5	Processing....	Commit transaction
t6	Rollback	

- **Mô phỏng lỗi:**

*** **Dữ liệu ban đầu:**

- Thông tin lượt xem của nhà có *MaNha* = "ho_2" và khách hàng có *MaKH* = "cli_2" là có *NgayXem* = "2021-05-01".
- Tức là vào ngày 01/05/2021, nhà "ho_002" và khách hàng "cli_2" có lịch xem nhà trực tiếp.

	MaKH	MaNha	NhanXet	NgayXem
1	cli_1	ho_0	NULL	2021-05-05
2	cli_2	ho_2	NULL	2021-05-01
3	cli_3	ho_1	NULL	2021-05-02
4	cli_3	ho_3	NULL	2021-04-15
5	cli_4	ho_2	NULL	2021-04-29

*** **Trường hợp lỗi:**

- **T1:** đang thực hiện cập nhật thông tin lượt xem, cụ thể là cập nhật *NgayXem* = "2021-05-09" của nhà "ho_2" và khách hàng "cli_2", tuy nhiên T1 bị rơi vào trường hợp rollback => thuộc tính *NgayXem* = "2021-05-01".
- **T2:** đang xem thông tin lượt xem nhà, cụ thể là thông tin lượt xem của nhà "ho_2" và khách hàng "cli_2", trong khi T1 đang update, mặc dù T1 thực hiện không thành công nhưng T2 vẫn hiển thị được *NgayXem* = "2021-05-09"

⇒ **Lỗi Dirty Read**

Transaction 1

Transaction 2

Transaction1_error....-ATNSISDL(thi (53))

```

use QuanLyThueBanNha
go

---- DEMO DIRTY READ
----- TRANSACTION 1 : CẬP NHẬT THÔNG TIN LƯỢT XEM -----
alter procedure CapNhatThongTinLuotXem
    @maKH char(10),
    @maNha char(10),
    @nhanXet nvarchar(100) = null,
    @ngayXem date = null,
    @error nvarchar(100) out
as
begin tran
    -- Kiểm tra khách hàng có tồn tại không
    if (len(isnull(@maKH, '')) <> 0)
    begin
        declare @isKH int
        exec @isKH = KiemTraKhachHangTonTai @maKH
        if (@isKH = 0)
        begin
            set @error = N'Lỗi: Khách hàng không tồn tại'
            raiserror(N'Lỗi: Khách hàng không tồn tại', 0, 0)
            rollback tran
            return
        end
    end

    -- Kiểm tra nhà có tồn tại không
    if (len(isnull(@maNha, '')) <> 0)
    begin
        declare @isNha int
        exec @isNha = KiemTraNhaTonTai @maNha
        if (@isNha = 0)
        begin
            set @error = N'Lỗi: Nhà không tồn tại'
            raiserror(N'Lỗi: Nhà không tồn tại', 0, 0)
            rollback tran
            return
        end
    end

    -- Kiểm tra ngày xem
    if (@ngayXem is null)
    begin
        set @ngayXem = (select NgayXem from ThongTinLuotXem where MaKH = @maKH and MaNha = @maNha)
    end

    -- Kiểm tra nhận xét
    if (len(isnull(@nhanXet, '')) = 0)
    begin
        set @nhanXet = (select NhanXet from ThongTinLuotXem where MaKH = @maKH and MaNha = @maNha)
    end

    -- Update
    update ThongTinLuotXem
    set NhanXet = @nhanXet, NgayXem = @ngayXem
    where MaKH = @maKH and MaNha = @maNha

    -- Có tính rollback để demo Dirty Read
    waitfor delay '00:00:10'
    rollback tran
    print 'Rollback !!!!!'
    return

    if (@@ERROR <> 0)
    begin
        set @error = N'Lỗi: Không thể cập nhật'
        raiserror(N'Lỗi: Không thể cập nhật', 0, 0)
        rollback
    end
    commit tran

-- Transaction 1 : Cập nhật thông tin lượt xem
declare @errorOutput nvarchar(50)
exec CapNhatThongTinLuotXem
    @maKH = 'cli_2',
    @maNha = 'ho_2',
    @ngayXem = '5/9/2021',
    @error = @errorOutput output
        
```

74 %

Messages

(1 row affected)
Rollback !!!!!

Completion time: 2021-04-08T15:36:42.2897095+07:00

Transaction2_error....-ATNSISDL(thi (56))

```

use QuanLyThueBanNha
go

---- DEMO DIRTY READ
----- TRANSACTION 2 : XEM THÔNG TIN LƯỢT XEM -----
alter procedure XemThongTinLuotXem
    @maKH char(10),
    @maNha char(10),
    @error varchar(50) out
as
begin tran
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
    -- Kiểm tra khách hàng có tồn tại không
    declare @isKH int
    exec @isKH = KiemTraKhachHangTonTai @maKH
    if (@isKH = 0)
    begin
        set @error = N'Lỗi: Khách hàng không tồn tại'
        raiserror(N'Lỗi: Khách hàng không tồn tại', 0, 0)
        rollback tran
        return
    end

    -- Kiểm tra nhà có tồn tại không
    declare @isNha int
    exec @isNha = KiemTraNhaTonTai @maNha
    if (@isNha = 0)
    begin
        set @error = N'Lỗi: Nhà không tồn tại'
        raiserror(N'Lỗi: Nhà không tồn tại', 0, 0)
        rollback tran
        return
    end

    Select * from ThongTinLuotXem where MaKH = @maKH and MaNha = @maNha
    if (@@ERROR <> 0)
    begin
        set @error = N'Lỗi: Lấy thông tin lượt xem'
        raiserror(N'Lỗi: Lấy thông tin lượt xem', 0, 0)
        rollback tran
        return
    end
    commit tran
go

--- Transaction 2 : Xem thông tin lượt xem
declare @errorOutput nvarchar(50)
exec XemThongTinLuotXem
    @maKH = 'cli_2',
    @maNha = 'ho_2',
    @error = @errorOutput output
        
```

74 %

Results

	MaKH	MaNha	NhanXet	NgayXem
1	cli_2	ho_2	NULL	2021-05-09

Messages

- **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T2 là *READ COMMITED*, nhằm mục đích chỉ đọc những dữ liệu đã commit.

ERROR02: DIRTY READ				
Trans t	capNhatLichXemNha (T1)	Khóa	xemThongTinLichXemNha(T2)	Khóa
t0	Kiểm tra sự tồn tại của khách hàng		Set → READ COMMITED	
t1	Kiểm tra sự tồn tại của loại nhà			
t2	Kiểm tra ngày xem hợp lệ			
t3	Cập nhật lịch xem nhà	T1: Xin khóa X SQL: Cấp khóa X	Kiểm tra sự tồn tại của nhà	
t4	Processing ...		Load danh sách lịch xem nhà	T2: Xin khóa S SQL: Không cấp khóa S do T1 đang giữ khóa X
t5	Processing ...		Commit transaction	
t6	Rollback			

1.3. Dirty read 03.

- **Mô tả kịch bản lỗi:** Chủ nhà đang cập nhật thông tin cá nhân. Nhưng vì sự cố nên không cập nhật thành công, hệ thống phải rollback lại trạng thái ban đầu. Cùng lúc đó, có khách hàng đang xem thông tin của chủ nhà đó.

*** **Transaction 1: capNhatThongTinChuNha (T1)**

Input: @MaChuNha, @ten, @sdt, @diaChi

Output: Dòng được cập nhật

*** **Transaction 2: xemThongTinChuNha (T2)**

Input: @MaChuNha

Output: Dòng thỏa điều kiện

ERROR03: DIRTY READ		
Trans t	capNhatThongTinChuNha (T1)	xemThongTinChuNha (T2)
t0	Kiểm tra sự tồn tại của chủ nhà	
t1	Cập nhật thông tin chủ nhà	Xem thông tin chủ nhà
t2	Processing	Commit transaction
t3	Processing ...	
t4	Rollback	

- **Mô phỏng lỗi:**

*** **Dữ liệu ban đầu:**

- Thông tin chủ nhà có *MaChuNha* = "own_0" là

	MaChuNha	Ten	SDT	DiaChi
1	own_0	Nguyễn Duy Khánh	0987556754	TPHCM
2	own_1	Phạm Minh Huy	0987564453	TPHCM
3	own_2	Nguyễn Ngọc Trâm	0977665654	TPHCM
4	own_3	Trần Mỹ Chi	0912453889	TPHCM
5	own_4	Phan Anh Thư	0912456754	TPHCM

***** Trường hợp lỗi:**

- **T1:** đang thực hiện cập nhật thông tin chủ nhà, cụ thể là cập nhật *SDT* = "0987665437" của chủ nhà "own_0". Tuy nhiên T1 bị rơi vào trường hợp rollback => thuộc tính *NgayXem* = "0987556754".
- **T2:** đang xem thông tin chủ nhà, cụ thể là xem thông tin của chủ nhà "own_0" trong khi T1 đang update, mặc dù T1 thực hiện không thành công nhưng T2 vẫn hiển thị được *SDT* = "0987665437".

⇒ **Lỗi Dirty Read**

• **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T2 là **READ COMMITED**, nhằm mục đích chỉ đọc những dữ liệu đã commit.

ERROR03: DIRTY READ				
Trans t	capNhatThongTinChuNha (T1)	Khóa	xemThongTinChuNha (T2)	Khóa
t0	Kiểm tra sự tồn tại của chủ nhà		Set → READ COMMITED	
t1	Cập nhật thông tin chủ nhà	T1: Xin khóa X SQL: Cấp khóa X	Kiểm tra sự tồn tại của nhà	
t2	Processing ...		Load danh sách lịch xem nhà	T2: Xin khóa S SQL: Không cấp khóa S do T1 đang giữ khóa X
t3	Processing ...		Commit transaction	
t4	Rollback			

2. Lỗi Lost update

2.1. Lost update 01

- **Mô tả kịch bản lỗi:** Khi hai admin cùng cập nhật giá thuê cho cùng một ngôi nhà, khi đó giá thuê của ngôi nhà sau khi đã cập nhật bởi hai admin có thể xảy ra lỗi chằng chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: CapNhatGiaThue (T1)

Input: @maNha, @giaThem

Output: @giaThayDoi

*** Transaction 2: CapNhatGiaThue (T2)

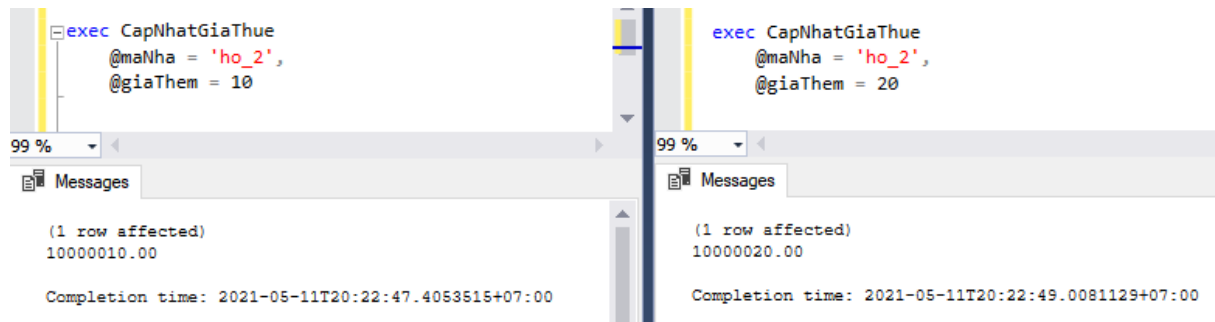
Input: @maNha, @giaThem

Output: @giaThayDoi

ERROR01: LOST UPDATE		
Trans t	CapNhatGiaThue (T1)	CapNhatGiaThue (T2)
t0	<code>set @gia = (Select GiaThue from Gia where MaGia = @maGia)</code>	
t1		<code>set @gia = (Select GiaThue from Gia where MaGia = @maGia)</code>
t2		<code>set @gia = @gia + @giaThem</code>
t3	<code>set @gia = @gia + @giaThem</code>	
t4	<code>update Gia set GiaThue = @gia where MaGia = @maGia</code>	
t5		<code>update Gia set GiaThue = @gia where MaGia = @maGia</code>
t6	<code>commit</code>	
t7		<code>commit</code>

****Trường hợp lỗi:** T1 sau khi thực hiện update nhưng chưa commit thì T2 thực hiện thao tác update trên cùng một ô dữ liệu.

Trường hợp bị lỗi với giá trị giá thuê ban đầu là 1,000,000 đồng. T1 (bên trái thực hiện trước) thì giá trị trả về là 1,000,0010 đồng. Sau đó, đến T2 thực hiện thêm 20 đồng vào vào giá thuê, nhưng giá trị trả về là 1,000,0020 đồng.

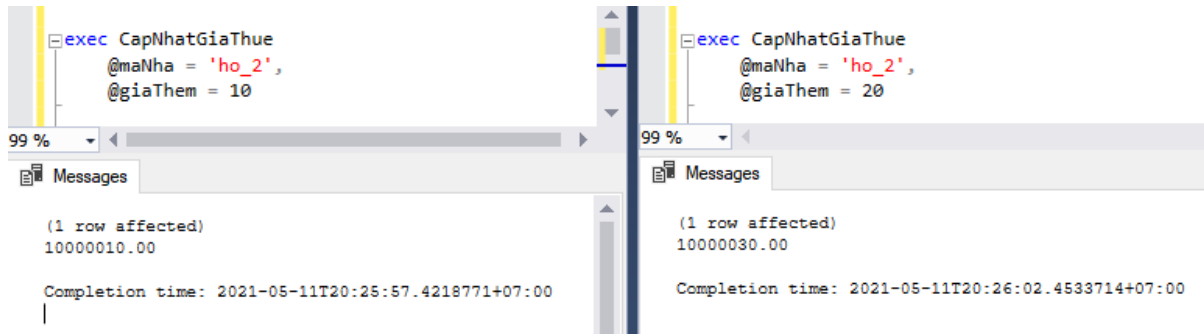


- **Phương pháp xử lý lỗi:**

Chúng ta sử dụng cơ chế khóa UPDLOCK và được giải thích bởi bảng sau:

ERROR01: LOST UPDATE				
Trans t	CapNhatGiaNha (T1)	Khóa	CapNhatGiaNha (T2)	Khóa
t0	set @luong = (Select Luong from NhanVien with (updlock) where MaNhanVien = @maNhanVien)	T1: Xin khóa U SQL: Cấp khóa U	Wait	T2: Xin khóa U SQL: Không cấp khóa U do T1 đang giữ khóa U
t1	set @gia = @gia + @giaThem			
t2	update Gia set GiaThue = @gia where MaGia = @maGia			
t3	commit			
t4			set @luong = (Select Luong from NhanVien with (updlock) where MaNhanVien = @maNhanVien)	
t5			set @gia = @gia + @giaThem	
t6			update Gia set GiaThue = @gia where MaGia = @maGia	
t7			commit	

**** Trường hợp chạy đúng giá trị giá thuê ban đầu là 1,000,000 đồng. T1 (bên trái) thực hiện trước thêm 10đ sau đó T2 (bên phải) cập nhật thêm 20đ thì giá trị cuối cùng là 1,000,030 đồng. Kết quả đúng như kỳ vọng!**



2.2. Lost update 02

- **Mô tả kịch bản lỗi:** Khi hai admin cùng cập nhật giá bán cho cùng một ngôi nhà, khi đó giá bán của ngôi nhà sau khi đã cập nhật bởi hai admin có thể xảy ra lỗi chằng chắp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: CapNhatGiaBan (T1)

Input: @maNha, @giaThem

Output: @giaThayDoi

*** Transaction 2: CapNhatGiaBan (T2)

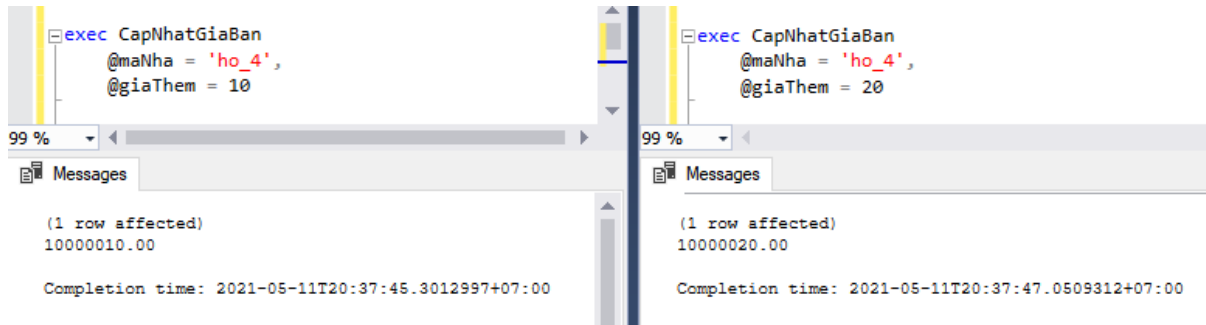
Input: @maNha, @giaThem

Output: @giaThayDoi

****Trường hợp lỗi:** T1 sau khi thực hiện update nhưng chưa commit thì T2 thực hiện thao tác update trên cùng một ô dữ liệu.

ERROR01: LOST UPDATE		
Trans t	CapNhatGiaBan (T1)	CapNhatGiaBan (T2)
t0	set @gia = (Select GiaBan from Gia where MaGia = @maGia)	
t1		set @gia = (Select GiaBan from Gia where MaGia = @maGia)
t2		set @gia = @gia + @giaThem
t3	set @gia = @gia + @giaThem	
t4	update Gia set GiaBan = @gia where MaGia = @maGia	
t5		update Gia set GiaBan = @gia where MaGia = @maGia
t6	commit	
t7		commit

Trường hợp bị lỗi với giá trị giá bán ban đầu là 1,000,000 đồng. T1 (bên trái thực hiện trước) thì giá trị trả về là 1,000,0010 đồng. Sau đó, đến T2 thực hiện thêm 20 đồng vào vào giá thuê, nhưng giá trị trả về là 1,000,0020 đồng.

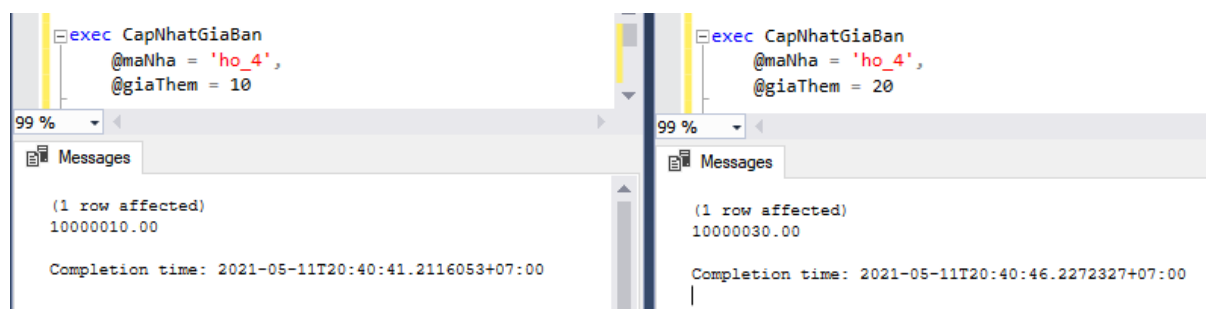


- Phương pháp xử lý lỗi:

Chúng ta sử dụng cơ chế khóa **UPDLOCK** và được giải thích bởi bảng sau:

ERROR02: LOST UPDATE				
Trans t	CapNhatGiaBan (T1)	Khóa	CapNhatGiaBan (T2)	Khóa
t0	set @gia = (Select GiaBan from Gia with (updlock) where MaGia = @maGia)	T1: Xin khóa U SQL: Cấp khóa U	Wait	T2: Xin khóa U SQL: Không cấp khóa U do T1 đang giữ khóa U
t1	set @gia = @gia + @giaThem			
t2	update Gia set GiaBan = @gia where MaGia = @maGia			
t3	commit			
t4			set @gia = (Select GiaBan from Gia with (updlock) where MaGia = @maGia)	
t5			set @gia = @gia + @giaThem	
t6			update Gia set GiaBan = @gia where MaGia = @maGia	
t7			commit	

****Trường hợp chạy đúng giá trị giá bán ban đầu là 1,000,000 đồng. T1 (bên trái) thực hiện trước thêm 10đ sau đó T2 (bên phải) cập nhật thêm 20đ thì giá trị cuối cùng là 1,000,030 đồng. Kết quả đúng như kỳ vọng!**



2.3. Lost update 03

- **Mô tả kịch bản lỗi** : Khi hai admin cùng cập nhật mức lương cho cùng một nhân viên, khi đó mức lương của nhân viên sau khi đã cập nhật bởi hai admin có thể xảy ra lỗi chằng chắp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: CapNhatLuongNhanVien (T1)

Input: @maNhanVien, @luongTang

Output: @luongCapNhat

*** Transaction 2: CapNhatLuongNhanVien (T2)

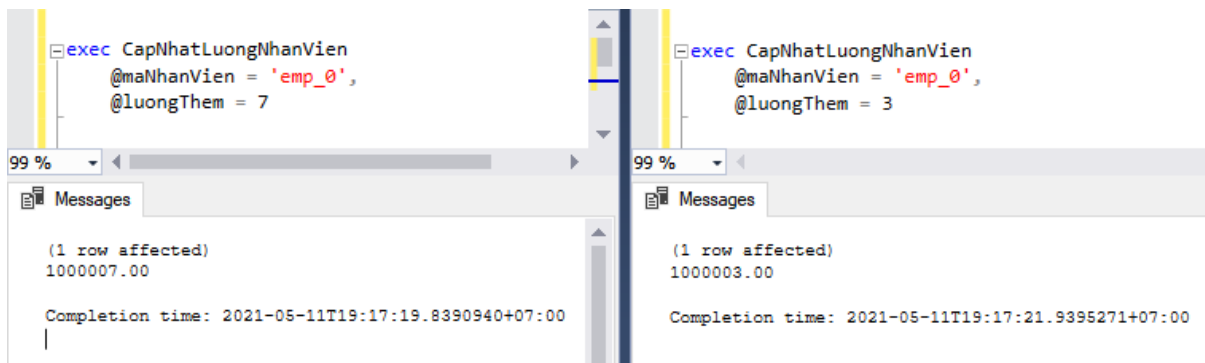
Input: @maNhanVien, @luongTang

Output: @luongCapNhat

****Trường hợp lỗi: T1 sau khi thực hiện update nhưng chưa commit thì T2 thực hiện thao tác update trên cùng một ô dữ liệu.**

ERROR03: LOST UPDATE		
Trans t	CapNhatLuongNhanVien (T1)	CapNhatLuongNhanVien (T2)
t0	<code>set @luong = (Select Luong from NhanVien where MaNhanVien = @maNhanVien)</code>	
t1		<code>set @luong = (Select Luong from NhanVien where MaNhanVien = @maNhanVien)</code>
t2		<code>set @luong = @luong + @luongThem</code>
t3	<code>set @luong = @luong + @luongThem</code>	
t4	<code>update NhanVien set Luong = @luong where MaNhanVien = @maNhanVien</code>	
t5		<code>update NhanVien set Luong = @luong where MaNhanVien = @maNhanVien</code>
t6	<code>commit</code>	
t7		<code>commit</code>

Trường hợp bị lỗi với giá trị lương nhân viên ban đầu là 1,000,000 đồng. T1 (bên trái thực hiện trước) thì giá trị trả về là 1,000,007. Sau đó, đến T2 thực hiện thêm 3đ vào thêm 3đ vào tài khoản, nhưng giá trị trả về là 1,000,003.

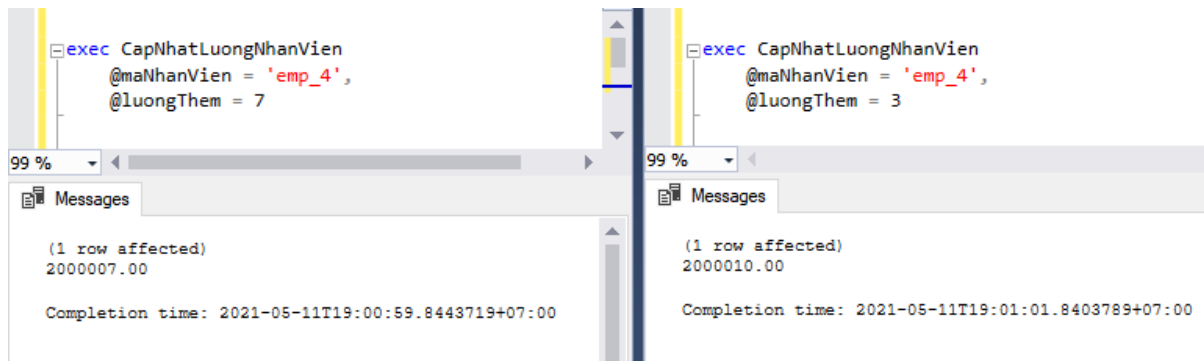


- Phương pháp xử lý lỗi:

Chúng ta sử dụng cơ chế khóa UPDLOCK và được giải thích bởi bảng sau:

ERROR03: LOST UPDATE				
Trans t	CapNhatLuongNhanVien (T1)	Khóa	CapNhatLuongNhanVien (T2)	Khóa
t0	<code>set @luong = (Select Luong from NhanVien where MaNhanVien = @maNhanVien)</code>	T1: Xin khóa U SQL: Cấp khóa U	Wait	T2: Xin khóa U SQL: Không cấp khóa U do T1 đang giữ khóa U
t1	<code>set @luong = @luong + @luongThem</code>			
t2	<code>update NhanVien set Luong = @luong where MaNhanVien = @maNhanVien</code>			
t3	<code>commit</code>			
t4			<code>set @luong = (Select Luong from NhanVien where MaNhanVien = @maNhanVien)</code>	
			<code>set @luong = @luong + @luongThem</code>	
			<code>update NhanVien set Luong = @luong where MaNhanVien = @maNhanVien</code>	
			<code>commit</code>	

****Trường hợp chạy đúng giá trị lương nhân viên ban đầu là 2,000,000 đồng. T1 (bên trái) thực hiện trước thêm 7đ sau đó T2 (bên phải) cập nhật thêm 3đ thì giá trị cuối cùng là 2,000,010 đồng. Kết quả đúng như kỳ vọng!**



3. Lỗi Unrepeatable read

3.1. Unrepeatable read 01.

- **Mô tả kịch bản lỗi:** Khi một admin 1 đang xem thông tin các nhân viên có mức lương trên mức lương chỉ định, thì admin 2 cập nhật mức lương của một nhân viên. Khi đó, kết quả trả về của admin 1 có thể không đúng như kỳ vọng vì phát sinh lỗi tranh chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: XemThongTinNhanVienTheoLuong (T1)

Input: @mucLuong

Output: @soLuong

*** Transaction 2: CapNhatLuongNhanVien (T2)

Input: @maNhanVien, @luongTang

Output: @luongCapNhat

****Trường hợp lỗi:** T1 thực hiện đếm số lượng nhân viên thỏa kiểu điện được định nghĩa, trước khi T1 xuất thông tin với số dòng được đếm thì T2 đã thực hiện thao tác update làm thay đổi kết quả của T1 trước đó, nên thông tin mà T1 xuất ra có thể không đúng như kỳ vọng.

ERROR01: UNREPEATABLE READ		
Trans t	XemThongTinNhanVienTheoLuong	CapNhatLuongNhanVien
t0	<code>set @soLuong = (select count(MaNhanVien) from NhanVien where NhanVien.Luong > @mucLuong)</code>	
t1		<code>set @luong = (Select Luong from NhanVien where MaNhanVien = @maNhanVien)</code>
t2		<code>set @luong = @luong + @luongThem</code>
t3		<code>update NhanVien set Luong = @luong where MaNhanVien = @maNhanVien</code>
t4	<code>select * from NhanVien where NhanVien.Luong > @mucLuong</code>	
t5	<code>commit</code>	

t6		commit
----	--	--------

- **Mô phỏng lỗi:** Theo kết quả bên dưới, trong T1 kết quả về 4 nhân viên có mức lương trên 1,000,000 nhưng thông tin hiển thị thì có 5 dòng (tức 5 nhân viên). Do, trong lúc T1 đang thực hiện thì T2 thực hiện thao tác UPDATE mức lương của một nhân viên có mã 'emp_0'. Điều này dẫn đến lỗi tranh chấp đồng thời đưa ra kết quả không như kỳ vọng.

```

declare @soLuongNhanVien int
exec XemThongTinNhanVienTheoLuong
    @mucLuong = 1000000,
    @soLuong = @soLuongNhanVien out

```

99 %

Results Messages

	MaNhanVien	Ten	NgaySinh	GioiTinh	Luong
1	emp_0	Đoàn Nam Thuận	2000-01-01	Nữ	1000500
2	emp_1	Phạm Gia Bảo	2000-04-11	Nam	4000937
3	emp_2	Phạm Gia Bảo	2000-04-11	Nam	4000005
4	emp_3	Lê Minh Hùng	2000-05-07	Nam	5000000
5	emp_4	Lê Ngọc Mai	2000-01-04	Nữ	2000010

(5 rows affected)
4
Completion time: 2021-05-11T20:55:13.0531514+07:00

```

exec CapNhatLuongNhanVien
    @maNhanVien = 'emp_0',
    @luongThem = 500

```

99 %

Messages

(1 row affected)
1000500.00
Completion time: 2021-05-11T20:55:05.3295722+07:00

- **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T2 là **REPEATABLE READ**, nhằm mục đích thiết lập Share lock và giữ đến hết thao tác.

ERROR01: UNREPEATABLE READ				
Trans t	XemThongTinNhanVienTheoLuong(T1)	Khóa	CapNhatLuongNhanVien (T2)	Khóa
t0			Set → REPEATABLE READ	
t1	set @soLuong = (select count(MaNhanVien) from NhanVien where NhanVien.Luong > @mucLuong)	T1: Xin khóa S SQL: Cấp khóa S T1: giữ khóa S đến hết T1		
t2			set @luong = (Select Luong from NhanVien where MaNhanVien = @maNhanVien)	
t3	select * from NhanVien where NhanVien.Luong > @mucLuong		Không thể thực hiện update	T2: Xin khóa X SQL: Không cấp khóa X do
t4	commit			

				T1 đang giữ khóa S
t5			set @luong = @luong + @luongThem	
t6			update NhanVien set Luong = @luong where MaNhanVien = @maNhanVien	
t7			commit	

```

declare @soLuongNhanVien int

exec XemThongTinNhanVienTheoLuong
    @mucLuong = 1000000,
    @soLuong = @soLuongNhanVien out

print(@soLuongNhanVien)

```

99 %

Results Messages

	MaNhanVien	Ten	NgaySinh	GioiTinh	Luong
1	emp_1	Phạm Gia Bảo	2000-04-11	Nam	4000937
2	emp_2	Phạm Gia Bảo	2000-04-11	Nam	4000005
3	emp_3	Lê Minh Hùng	2000-05-07	Nam	5000000
4	emp_4	Lê Ngọc Mai	2000-01-04	Nữ	2000010

```

exec CapNhatLuongNhanVien
    @maNhanVien = 'emp_0',
    @luongThem = 500

```

99 %

Messages

Commands completed successfully.

Completion time: 2021-05-11T21:47:42.1586998+07:00

3.2. Unrepeatable read 02

- **Mô phỏng kịch bản lỗi:** Khi một khách hàng A đang xem thông tin các ngôi nhà có giá thuê dưới mức giá chỉ định, thì admin B cập nhật mức giá thuê của một ngôi nhà. Khi đó, kết quả trả về của khách hàng A có thể không đúng như kỳ vọng vì phát sinh lỗi tranh chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: ThôngTinNhaTheoGiaThue (T1)

Input: @giaThue

Output: @soLuong

*** Transaction 2: CapNhatGiaThue (T2)

Input: @maNha, @giaThem

****Trường hợp lỗi:** T1 thực hiện đếm số lượng ngôi nhà thỏa kiểu điện được định nghĩa, trước khi T1 xuất thông tin với số dòng được đếm thì T2 đã thực hiện thao tác update làm thay đổi kết quả của T1 trước đó, nên thông tin mà T1 xuất ra có thể không đúng như kỳ vọng.

ERROR02: UNREPEATABLE READ		
t	ThôngTinNhaTheoGiaThue(T1)	CapNhatGiaThue (T2)
t0	<pre>set @soLuong = (select count(MaNha) from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaThue < @giaThue)</pre>	
t1		<pre>set @gia = (Select GiaThue from Gia where MaGia = @maGia)</pre>
t2		<pre>set @gia = @gia + @giaThem</pre>
t3		<pre>update Gia set GiaThue = @gia where MaGia = @maGia</pre>
t4	<pre>select * from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaThue < @giaThue</pre>	
t5	<pre>commit</pre>	
t6		<pre>commit</pre>

```

declare @soLuongOut int
exec ThongTinNhaTheoGiaThue
    @giaThue = 4000000,
    @soLuong = @soLuongOut out

print(@soLuongOut)

```

```

exec CapNhatGiaThue
    @maNha = 'ho_2',
    @giaThem = 2000000

```

9 %

Results Messages

	MaNha	SoLuongPhong	SoLuotXem	TinhTrang
1	ho_4	2	NULL	Trống

(1 row affected)

2

99 %

Messages

(1 row affected)

5000000.00

Completion time: 2021-05-11T22:13:00.0829265+07:00

- **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T1 là **REPEATABLE READ**, nhằm mục đích thiết lập Share lock và giữ đến hết thao tác, cụ thể theo bảng sau:

ERROR02: UNREPEATABLE READ				
Trans t	xemThongTinNV (T1)	Khóa	capNhatThongTinNV (T2)	Khóa
t0	Set → REPEATABLE READ			
t1	set @soLuong = (select count(MaNha) from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaThue < @giaThue)	T1: Xin khóa S SQL: Cấp khóa S T1: giữ khóa S đến hết T1		
t2			set @gia = (Select GiaThue from Gia where MaGia = @maGia)	
t3	select * from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaThue < @giaThue		Không thể thực hiện update	T2: Xin khóa X SQL: Không cấp khóa X do T1 đang giữ khóa S
t4	commit			
t5			set @gia = @gia + @giaThem	

t6			<code>update Gia set GiaThue = @gia where MaGia = @maGia</code>	
t7			<code>commit</code>	

```

declare @soLuongOut int
exec ThongTinNhaTheoGiaThue
    @giaThue = 100000000,
    @soLuong = @soLuongOut out
print(@soLuongOut)

```

```

exec CapNhatGiaThue
    @maNha = 'ho_1',
    @giaThem = 100000000

```

99 %

Results Messages

(2 rows affected)

2

Completion time: 2021-05-11T22:04:01.818:

99 %

Messages

(1 row affected)

Completion time: 2021-05-11T22:05:05.9342979+07:00

3.3. Unrepeatable read 03

- **Mô tả kịch bản lỗi:** Khi một khách hàng A đang xem thông tin các ngôi nhà có giá bán dưới mức giá chỉ định, thì admin B cập nhật mức giá bán của một ngôi nhà. Khi đó, kết quả trả về của khách hàng A có thể không đúng như kỳ vọng vì phát sinh lỗi tranh chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: ThôngTinNhaTheoGiaBan (T1)

Input: @giaBan

Output: @soLuong

*** Transaction 2: CapNhatGiaBan (T2)

Input: @maNha, @giaThem

****Trường hợp lỗi:** T1 thực hiện đếm số lượng ngôi nhà thỏa kiểu điện được định nghĩa, trước khi T1 xuất thông tin với số dòng được đếm thì T2 đã thực hiện thao tác update làm thay đổi kết quả của T1 trước đó, nên thông tin mà T1 xuất ra có thể không đúng như kỳ vọng.

ERROR02: UNREPEATABLE READ		
t	ThôngTinNhaTheoGiaBan(T1)	CapNhatGiaBan (T2)
t0	<pre>set @soLuong = (select count(MaNha) from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaBan < @giaBan)</pre>	
t1		<pre>set @gia = (Select GiaBan from Gia where MaGia = @maGia)</pre>
t2		<pre>set @gia = @gia + @giaThem</pre>
t3		<pre>update Gia set GiaBan = @gia where MaGia = @maGia</pre>
t4	<pre>select * from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaBan < @giaThue</pre>	
t5	<pre>commit</pre>	
t6		<pre>commit</pre>

- Mô phỏng lỗi:

*** Dữ liệu ban đầu:

- Danh sách nhà có giá thuê tối đa 10 triệu là

```
-- TRANSACTION 1 : XEM NHÀ CÓ GIÁ TỐI ĐA
declare @errorOutput nvarchar(50)
exec XemThongTinNhaGiaToiDa
    @gia = 10000000,
    @loaiGia = N'Thuê',
    @error = @errorOutput output
```

← sql

98 %

	MaNha	SoLuongPhong	SoLuotXem	TinhTrang	Duong	Quan	ThanhPho	KhuVuc	MaGia	MaChuNha	MaNh
1	ho_2	2	2	Trống	Lê Văn Sỹ	3	HCM	TPHCM	price_2	own_4	emp_
2	ho_4	2	NULL	Trống	Nguyễn Kim	10	HCM	TPHCM	price_4	own_1	emp_

← kết quả

- Phương pháp xử lý lỗi:

Đặt mức cô lập của T1 là REPEATABLE READ, nhằm mục đích thiết lập Share lock và giữ đến hết thao tác

ERROR03: UNREPEATABLE READ				
Trans t	ThongTinNhaTheoGiaBan (T1)	Khóa	CapNhatGiaBan (T2)	Khóa
t0	Set → REPEATABLE READ			
t1	<pre>set @soLuong = (select count(MaNha) from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaBan < @giaBan)</pre>	T1: Xin khóa S SQL: Cấp khóa S T1: giữ khóa S đến hết T1		
t2			<pre>set @gia = (Select GiaBan from Gia where MaGia = @maGia)</pre>	
t3	<pre>select * from Nha join Gia on Nha.MaGia = Gia.MaGia where Gia.GiaBan < @giaThue</pre>		Không được thực hiện update	T2: Xin khóa X SQL: Không cấp khóa X do T1 đang giữ khóa S
t4	commit			
t5			<pre>set @gia = @gia + @giaThem</pre>	

t6			<pre> update Gia set GiaBan = @gia where MaGia = @maGia </pre>	
t7			<pre> commit </pre>	

4. Lỗi Phantom

4.1. Phantom 01

- **Mô tả kịch bản lỗi:** Khi một admin A đang xem thông tin các chi nhánh (bao gồm số lượng các chi nhánh). Trong khi đó admin B cập nhật (THÊM) một chi nhánh mới vào database. Khi đó, kết quả trả về của admin A có thể không đúng như kỳ vọng vì phát sinh lỗi tranh chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: ThôngTinChiNhanh (T1)

Output: @soLuong

*** Transaction 2: ThemChiNhanh (T2)

Input: @SDT, @fax, @duong, @quan, @khuVuc, @thanhPho

Output: @error

****Trường hợp lỗi:** T1 thực hiện đếm số chi nhánh nhà thầu kiểu điện được định nghĩa, trước khi T1 xuất thông tin với số dòng được đếm thì T2 đã thực hiện thao tác INSERT làm thay đổi kết quả của T1 trước đó, nên thông tin mà T1 xuất ra có thể không đúng như kỳ vọng.

ERROR01: PHANTOM		
Trans t	ThôngTinChiNhanh (T1)	ThemChiNhanh(T2)
t0	<code>set @soLuong = (select count(MaChiNhanh) from ChiNhanh)</code>	
t1		<code>insert into ChiNhanh values (@maChiNhanh, @sdt, @fax, @duong, @quan, @khuVuc, @thanhPho)</code>
t2	<code>select * from ChiNhanh</code>	
t3	<code>commit</code>	
t4		<code>commit</code>

- **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T1 là **SERIALIZABLE**, nhằm mục đích không cho DELETE vào đơn vị dữ liệu mà nó đang được đọc, cụ thể được giải thích theo bảng sau:

ERROR01: PHANTOM				
Trans t	ThôngTinChiNhanh (T1)	Khóa	ThemChiNhanh(T2)	Khóa
t0	Set → SERIALIZABLE			
t1	<code>set @soLuong = (select count(MaChiNhanh) from ChiNhanh)</code>	T1: Xin khóa S SQL: Cấp khóa S		
t2	<code>select * from ChiNhanh</code>	T1: giữ khóa S đến hết T1	Không thể thực hiện insert	T2: Xin khóa X SQL: Không cấp khóa X do T1 đang giữ khóa S
t3	<code>commit</code>			
t4			<code>insert into ChiNhanh values (@maChiNhanh, @sdt, @fax, @duong, @quan, @khuVuc, @thanhPho)</code>	
t5			<code>commit</code>	

4.2. Phantom 02

- **Mô tả kịch bản lỗi:** Khi một khách hàng A đang xem thông tin các bài đăng (bao gồm số lượng các bài đăng). Trong khi đó chủ nhà B đăng thêm một bài đăng mới. Khi đó, kết quả trả về của khách hàng A có thể không đúng như kỳ vọng vì phát sinh lỗi tranh chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: ThôngTinBaiDang (T1)

Input: @loaiNha

Output: Dòng thỏa điều kiện

*** Transaction 2: ThemBaiDang (T2)

Input: @maNha, @ngayHetHan, @ngayDangBai

Output : @error

****Trường hợp lỗi:** T1 thực hiện đếm số lượng bài đăng nhà thỏa kiểu điện được định nghĩa, trước khi T1 xuất thông tin với số dòng được đếm thì T2 đã thực hiện thao tác INSERT làm thay đổi kết quả của T1 trước đó, nên thông tin mà T1 xuất ra có thể không đúng như kỳ vọng.

ERROR02: PHANTOM		
Trans t	ThôngTinBaiDang (T1)	ThemBaiDang (T2)
t0	<code>set @soLuong = (select count(MaBaiDang) from BaiDang)</code>	
t1		<code>insert into BaiDang(MaBaiDang, NgayDang, NgayHetHan, MaNha, LuotXemBaiDang, MaChuNha) values (@maBaiDang, @ngayDang, @ngayHetHan, @maNha, @luotXemBaiDang, @maChuNha)</code>
t2	<code>select * from BaiDang</code>	
t3	<code>commit</code>	
t4		<code>commit</code>

- Phương pháp xử lý lỗi:

Đặt mức cô lập của T1 là **SERIALIZABLE**, nhằm mục đích không cho INSERT vào đơn vị dữ liệu mà nó đang được đọc, cụ thể được giải thích theo bảng sau:

ERROR02: PHANTOM				
Trans t	ThôngTinBaiDang (T1)	Khóa	ThemBaiDang (T2)	Khóa
t0	Set → SERIALIZABLE			
t1	<code>set @soLuong = (select count(MaBaiDang) from BaiDang)</code>	T1: Xin khóa S SQL: Cấp khóa S		
t2	<code>select * from BaiDang</code>	T1: giữ khóa S đến hết T1	Không thể thực hiện insert	T2: Xin khóa X SQL: Không cấp khóa X do T1 đang giữ khóa S
t3	<code>commit</code>			
t4			<code>insert into BaiDang(MaBaiDang, NgayDang, NgayHetHan, MaNha, LuotXemBaiDang, MaChuNha)</code> <code>values (@maBaiDang, @ngayDang, @ngayHetHan, @maNha, @luotXemBaiDang, @maChuNha)</code>	
t5			<code>commit</code>	

4.3. Phantom 03

- **Mô tả kịch bản lỗi:** Khi một admin A đang xem thông tin các chủ nhà (bao gồm số lượng các chủ nhà). Trong khi đó admin B cập nhật (THÊM) một chủ nhà mới vào database. Khi đó, kết quả trả về của admin A có thể không đúng như kỳ vọng vì phát sinh lỗi tranh chấp đồng thời.

- **Chi tiết về transaction:**

*** Transaction 1: ThôngTinChuNha (T1)

Output: @soLuong

*** Transaction 2: ThemChuNha (T2)

Input: @ten, @SDT, @diaChi

Output : @error

****Trường hợp lỗi:** T1 thực hiện đếm số lượng chủ nhà thỏa kiểu điện được định nghĩa, trước khi T1 xuất thông tin với số dòng được đếm thì T2 đã thực hiện thao tác INSERT làm thay đổi kết quả của T1 trước đó, nên thông tin mà T1 xuất ra có thể không đúng như kỳ vọng.

ERROR03: PHANTOM		
Trans t	ThôngTinChuNha (T1)	ThemChuNha(T2)
t0	<code>set @soLuong = (select count(MaChuNha) from ChuNha)</code>	
t1		<code>insert into ChuNha (MaChuNha, Ten, SDT, DiaChi) values (@maChuNha, @ten, @sdt, @diaChi)</code>
t2	<code>select * from ChuNha</code>	
t3	<code>commit</code>	
t4		<code>commit</code>

- **Phương pháp xử lý lỗi:**

Đặt mức cô lập của T2 là **SERIALIZABLE**, nhằm mục đích không cho DELETE vào đơn vị dữ liệu mà nó đang được đọc, cụ thể được giải thích theo bảng sau:

ERROR03: PHANTOM				
Trans t	ThôngTinChuNha (T1)	Khóa	ThemChuNha (T2)	Khóa
t0	Set → SERIALIZABLE			
t1	<code>set @soLuong = (select count(MaChuNha) from ChuNha)</code>	T1: Xin khóa S SQL: Cấp khóa S		
t2	<code>select * from ChuNha</code>	T1: giữ khóa S đến hết T1	Không thể thực hiện insert	T2: Xin khóa X SQL: Không cấp khóa X do T1 đang giữ khóa S
t3	<code>commit</code>			
t4			<code>insert into ChuNha (MaChuNha, Ten, SDT, DiaChi) values (@maChuNha, @ten, @sdt, @diaChi)</code>	
t5			<code>commit</code>	