##### **ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

##### **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

****

**BÁO CÁO ĐỒ ÁN THỰC HÀNH**

**MÔN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**GVHD: Hồ Thị Hoàng Vy**

### Nhóm 07:

### 18120213 Võ Đại Nam

### 18120214 Lê Ngọc Bảo Ngân

### 18120215 Vũ Yến Ngọc

### 18120217 Nguyễn Trần Ái Nguyên

### 18120227 Phạm Văn Minh Phương

# THÔNG TIN NHÓM

*(Cô có thể theo dõi tiến độ thực hiện đồ án nhóm cập nhật trên* [*Link GitHub Nhóm 07*](https://github.com/K18HCMUS-HQT-CSDL/DoAnHQT)*)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ tên** | **Công việc** | **% Hoàn thành** |
| **1** | 18120213 | Võ Đại Nam | * Cài đặt tình huống tranh chấp: Dirty Read * Cài đặt tình huống tranh chấp: Lost Update | 100% |
| **2** | 18120214 | Lê Ngọc Bảo Ngân | * Cài đặt tình huống tranh chấp: Cycle DeadLock * Cài đặt tình huống tranh chấp: Conversion DeadLock | 100% |
| **3** | 18120215 | Vũ Yến Ngọc | * Cài đặt tình huống tranh chấp: Dirty Read * Cài đặt tình huống tranh chấp: Lost Update | 100% |
| **4** | 18120217 | Nguyễn Trần Ái Nguyên | * Cài đặt tình huống tranh chấp: Unrepeatable Read * Cài đặt tình huống tranh chấp: Phantom | 100% |
| **5** | 18120227 | Phạm Văn Minh Phương | * Cài đặt tình huống tranh chấp: Unrepeatable Read * Cài đặt tình huống tranh chấp: Phantom | 100% |

MỤC LỤC

[THÔNG TIN NHÓM 2](#_Toc61771088)

[CÀI ĐẶT TÌNH HUỐNG TRANH CHẤP 5](#_Toc61771089)

[I. Sinh viên thực hiện: 18120213 5](#_Toc61771090)

[1. Tình huống 1: Dirty Read 5](#_Toc61771091)

[2. Tình huống 2: Lost Update 6](#_Toc61771092)

[II. Sinh viên thực hiện: 18120214 8](#_Toc61771093)

[1. Tình huống 3: Conversion DeadLock 8](#_Toc61771094)

[2. Tình huống 4: Cycle DeadLock 9](#_Toc61771095)

[III. Sinh viên thực hiện: 18120215 10](#_Toc61771096)

[1. Tình huống 5: Dirty Read 10](#_Toc61771097)

[2. Tình huống 6: Lost Update 12](#_Toc61771098)

[IV. Sinh viên thực hiện: 18120217 13](#_Toc61771099)

[1. Tình huống 7: Unrepeatable Read 13](#_Toc61771100)

[2. Tình huống 8: Phantom 14](#_Toc61771101)

[V. Sinh viên thực hiện: 18120227 16](#_Toc61771102)

[1. Tình huống 9: Unrepeatable Read 16](#_Toc61771103)

[2. Tình huống 10: Phantom 17](#_Toc61771104)

[XỬ LÝ TÌNH HUỐNG TRANH CHẤP 20](#_Toc61771105)

[I. Sinh viên thực hiện: 18120213 20](#_Toc61771106)

[1. Tình huống 1: Dirty Read 20](#_Toc61771107)

[2. Tình huống 2: Lost Update 21](#_Toc61771108)

[II. Sinh viên thực hiện: 18120215 25](#_Toc61771109)

[1. Tình huống 5: Dirty Read 25](#_Toc61771110)

[2. Tình huống 6: Lost Update 26](#_Toc61771111)

[III. Sinh viên thực hiện: 18120217 29](#_Toc61771112)

[1. Tình huống 7: Unrepeateale Read 29](#_Toc61771113)

[2. Tình huống 8: Phantom 30](#_Toc61771114)

[VI. Sinh viên thực hiện: 18120227 33](#_Toc61771115)

[1. Tình huống 9: Unrepeatable Read 33](#_Toc61771116)

[2. Tình huống 10: Phantom 34](#_Toc61771117)

# CÀI ĐẶT TÌNH HUỐNG TRANH CHẤP

# Sinh viên thực hiện: 18120213

## Tình huống 1: Dirty Read

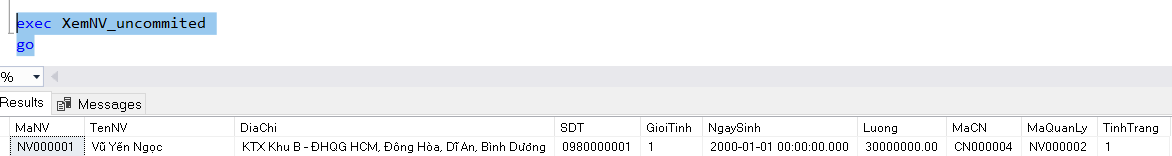
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Dirty Read***  T1 (User = CEO): thực hiện update chi nhánh và nhân viên quản lý cho 1 nhân viên  T2 (User = QLCN): thực hiện thống kê nhân viên tại các chi nhánh  Ví dụ: CEO chuyển công tác cho nhân viên với mã là NV000008, update Mã chi nhánh mới CN000004, update Mã nhân viên quản lý mới NV000002. Tại thời điểm đó, QLCN cũng đang thống kê nhân viên ở các chi nhánh. | | | |
| **sp\_ChuyenNV\_uncommited** | **Khóa** | **sp\_ChuyenNV\_uncommited** | **Khóa** |
| ***Input:*** @manv, @mcn, @mql  ***Output***: Cập nhập chi nhánh và nhân viên quản lý mới cho nhân viên đã chọn | ***Input***:  ***Output***: |
| SET TRANSACTION ISOLATION  LEVEL COMMITTED. | SET TRANSACTION ISOLATION  LEVEL UNCOMMITTED |
| BEGIN TRAN |  |  |  |
| B1: Cập nhập chi nhánh mới  update NhanVien  set MaCN=@mcn  where MaNV=@manv | **X**  **(Nhân viên)** |  |  |
| waitfor delay '00:00:10' |  | BEGIN TRAN |  |
|  |  | B1: Thống kê Nhân viên  select \* from NhanVien | **R**  **(Nhân Viên)**  Không cần xin khoá |
|  |  | COMMIT |  |
| B2: Cập nhập nhân viên quản lý mới  update NhanVien  set MaQuanLy=@mql  where MaNV=@manv | **X**  **(Nhân viên)** |  |  |
| B3: Thống kê Nhân viên  select \* from NhanVien where MaNV=@manv | **R**  **(Nhân viên)** Xin khóa tại bảng Nhân Viên |  |  |
| COMMIT |  |  |  |
| if(not exists(select \* from ChiNhanh where MaCN=@mcn) or not exists(select \* from NhanVien where MaNV=@manv))  begin  RAISERROR('Chuyen nhan vien khong thanh cong',1,1)  ROLLBACK TRAN sp\_ChuyenNV\_uncommited |  |  |  |

Kết quả: Dựa vào kết quả bên dưới ta thấy được đáng lẽ QLCN phải thấy được cả hai yếu tố update của CEO nhưng lại chỉ thấy được mã chi nhánh được cập nhập 🡺 Xuất hiện lỗi Dirty Read

* **User**: CEO



* **User**: QLCN

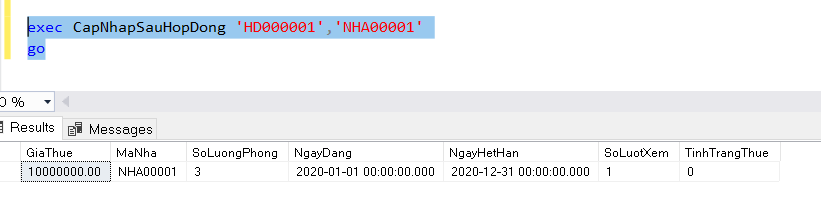


## Tình huống 2: Lost Update

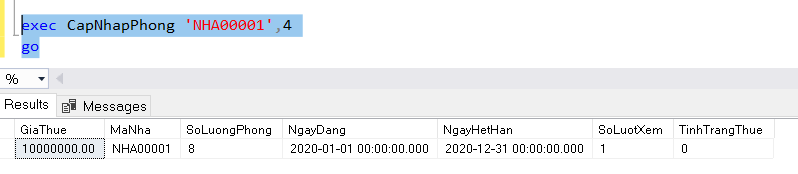
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Lost Update***  T1 (User = Nhân viên): thực hiện update số phòng sau khi hợp đồng thành công  T2 (User = Chủ nhà): thực hiện update số phòng còn lại mới  Ví dụ: Sau khi làm hợp đồng cho thuê phòng xong, nhân viên sẽ cập nhập lại số phòng= số phòng -1. Tại thời điểm đó chủ nhà xây thêm phòng cho thuê nên chủ nhà cập nhập lại số phòng của nhà mình. | | | |
| **sp\_ChuyenNV\_uncommited** | **Khóa** | **sp\_CapNhat** | **Khóa** |
| ***Input*** @mhd, @manha  ***Output***: Cập nhập Số phòng | ***Input***: @manha, @spt  ***Output***: Cập nhập số phòng |
| SET TRANSACTION ISOLATION  LEVEL COMMITTED. | SET TRANSACTION ISOLATION  LEVEL UNCOMMITTED |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu tại bảng Nhà thuê  declare @temp int  select @temp=SoLuongPhong from NhaThue where MaNha=@manha | **R**  **(Nhà thuê)**  Xin khóa đọc tại Nhà thuê |  |  |
| waitfor delay '00:00:10' |  | BEGIN TRAN |  |
|  |  | B1: Đọc dữ liệu tại bảng nhà thuê  declare @temp int  select @temp=SoLuongPhong from NhaThue where MaNha=@manha | **R**  **(Nhà thuê)**  Không cần xin khoá |
|  |  | B2: Cập nhập số phòng mới  update NhaThue  set SoLuongPhong=@temp+@spt  where MaNha=@manha | **X**  **(Nhà thuê)** |
|  |  | COMMIT |  |
| B2: Cập nhập số phòng mới  update NhaThue  set SoLuongPhong=@temp-1  where MaNha=@manha | **X**  **(Nhà thuê)** |  |  |
| B3: Thống kê lại số phòng  select \* from NhaThue where MaNha=@manha | **R**  **(Nhà thuê)** |  |  |
| COMMIT |  |  |  |
| if(not exists(select \* from HopDong where MaHD=@mhd) or not exists(select\* from NhaThue where MaNha=@manha))  begin  RAISERROR('Chuyen nhan vien khong thanh cong',1,1)  ROLLBACK TRAN sp\_CapNhapSau  end |  | if(not exists(select\* from NhaThue where MaNha=@manha))  begin  RAISERROR('Chuyen nhan vien khong thanh cong',1,1)  ROLLBACK TRAN sp\_CapNhap  end |  |

Kết quả: Dựa vào hình ở dưới ta thấy ở nhân viên số phòng là 3, ở chủ nhà số phòng là 8, điều này vô lý bởi vì số phòng đáng lẽ là 4+4-1=7 mà ở đây lại là 3 và 8 => Lost Update

* **User**: Nhân viên



* **User**: Chủ nhà

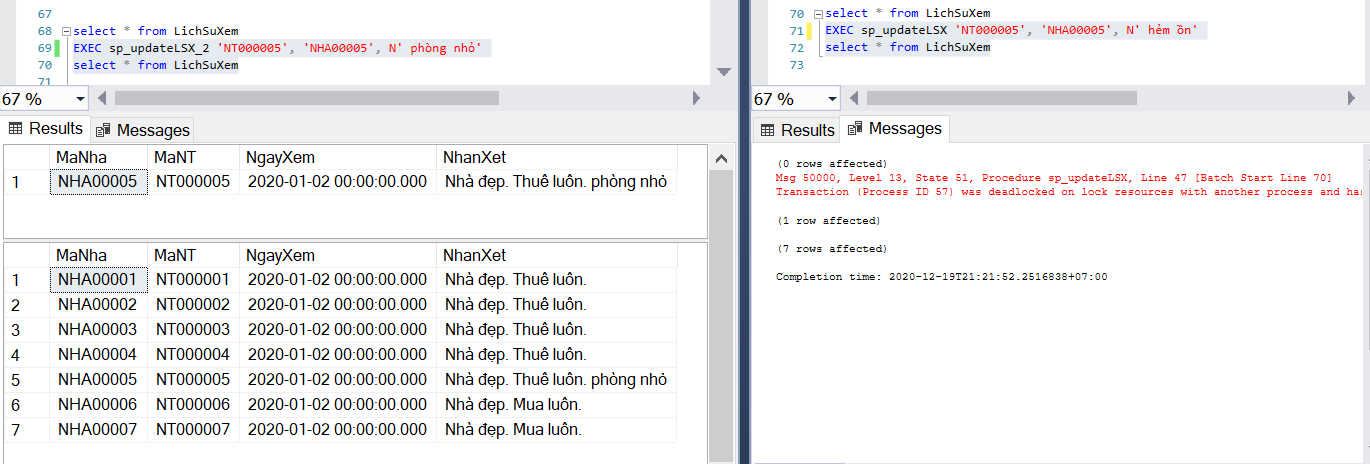


# Sinh viên thực hiện: 18120214

## Tình huống 3: Conversion DeadLock

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR02: *Conversion DeadLock***  T1 (User = NhanVien): cả 2 nhân viên đồng thời cập nhật nhận xét cùng 1 nhà trong LichSuXem  T2 (User = NhanVien): cả 2 nhân viên đồng thời cập nhật nhận xét cùng 1 nhà trong LichSuXem  Ví dụ trường hợp: Nhân viên 1 cập nhật nhật xét @NX = “hẻm ồn” cho @maNT = 'NT000006', @maNha ='NHA00006', đồng thời Nhân viên 2 cũng cập nhật cho @maNT = 'NT000006', @maNha ='NHA00006' nhưng @NX = ‘phòng nhỏ’ | | | |
| **sp\_updateLSX** | **Khóa** | **sp\_updateLSX\_2** | **Khóa** |
| ***Input:*** @maNT, @maNha, @NX  ***Output***: Cập nhật lịch sử xem | ***Input:*** @maNT, @maNha, @NX  ***Output***: Cập nhật lịch sử xem |
| SET TRAN ISOLATION LEVEL SERIALIZABLE | SET TRAN ISOLATION LEVEL SERIALIZABLE |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu tại bảng LichSuXem  DECLARE @nxet NVARCHAR(50) = (SELECT NhanXet FROM LichSuXem WHERE MaNT = @maNT and MaNha = @maNha) | **R(LichSuXem)**  Dọc Dữ liệu Luong trên 1 dòng của bảng LichSuXem |  |  |
| waitfor delay '00:00:7' |  | BEGIN TRAN |  |
|  |  | B1: Đọc dữ liệu tại bảng LichSuXem  DECLARE @nxet NVARCHAR(50) = (SELECT NhanXet FROM LichSuXem WHERE MaNT = @maNT and MaNha = @maNha) | **R**  **(LichSuXem)**  Không cần xin khoá |
|  |  | B2: B2: Cập nhập nhận xét  SET @nxet = @nxet + @NX  UPDATE LichSuXem SET NhanXet = @nxet WHERE MaNha = @maNha and MaNT = @maNT | **X**  **(LichSuXem)** |
|  |  | COMMIT |  |
| B2: Cập nhập nhận xét  SET @nxet = @nxet + @NX  UPDATE LichSuXem SET NhanXet = @nxet WHERE MaNha = @maNha and MaNT = @maNT | **X**  **(LichSuXem)** |  |  |
| RAISERROR  IF @@TRANCOUNT > 0  ROLLBACK TRAN |  |  |  |
| COMMIT |  |  |  |

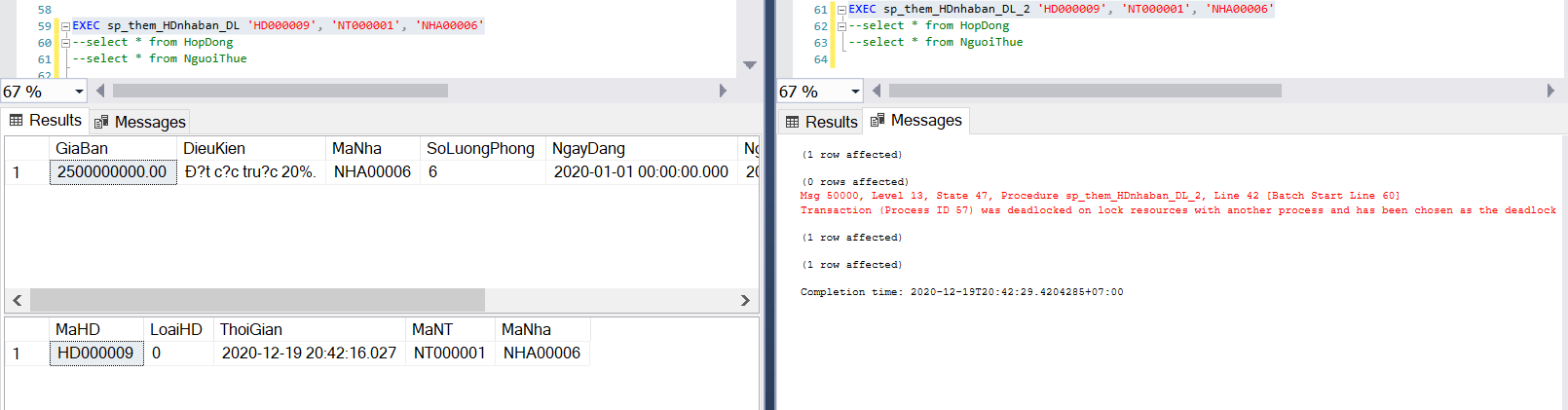
Kết quả: T1 bắt DeadLock, kết quả của bảng LichSuXem chỉ cập nhật nhận xét của T1



## Tình huống 4: Cycle DeadLock

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR04: *Cycle DeadLock***  T1 (User = NhanVien): Thêm 1 hợp đồng và cập nhật lại tình trạng của Nhà bán  T2 (User = NhanVien): Cập nhật tình trạng của Nhà bán và thêm 1 hợp đồng  Ví dụ trường hợp: Nhân viên 1 thêm 1 hợp đồng với @maHD= 'HD000009', @maNT= 'NT000001’ và @maNha='NHA00006' rồi sau đó tiến hành cập nhật tình trạng nhà đó thành đã bán, còn Nhân viên 2 thực hiện ngược lại | | | |
| **sp\_ChuyenNV\_uncommited** | **Khóa** | **sp\_CapNhap** | **Khóa** |
| ***Input*** @maNT, @maNha, @maHD  ***Output***: Cập nhật tình trạng nhà và thêm 1 hợp đồng | ***Input*** @maNT, @maNha, @maHD  ***Output***: Thêm 1 hợp đồng và cập nhật tình trạng nhà |
| BEGIN TRAN |  |  |  |
| B1: Cập nhật tình trạng nhà bán  DECLARE @tt BIT  SET @tt = '0'  UPDATE NhaBan SET TinhTrangBan = @tt WHERE MaNha = @maNha | **X**  **(Nhà bán)** |  |  |
| waitfor delay '00:00:07' |  | BEGIN TRAN |  |
|  |  | B1: Thêm 1 hợp đồng  INSERT INTO HopDong (MaHD, LoaiHD, ThoiGian, MaNT, MaNha) VALUES (@maHD, 0, GETDATE(), @maNT, @maNha) | **X(Hợp Đồng)** |
|  |  | B2: Cập nhập tình trạng nhà bán  DECLARE @tt BIT  SET @tt = '0'  UPDATE NhaBan SET TinhTrangBan = @tt WHERE MaNha = @maNha | **X**  **(Nhà bán)** |
|  |  | COMMIT |  |
| B2: Thêm 1 hợp đồng  INSERT INTO HopDong (MaHD, LoaiHD, ThoiGian, MaNT, MaNha) VALUES (@maHD, 0, GETDATE(), @maNT, @maNha) | **X(Hợp Đồng)** |  |  |
| RAISERROR  IF @@TRANCOUNT > 0  ROLLBACK TRAN |  |  |  |
| COMMIT |  |  |  |

Kết quả: T2 bắt DeadLock



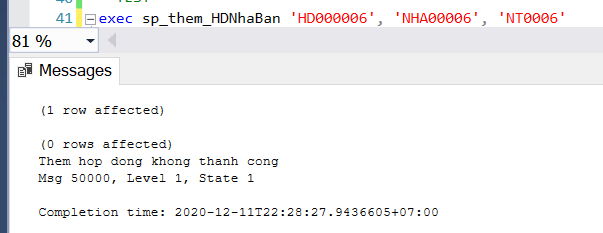
# Sinh viên thực hiện: 18120215

## Tình huống 5: Dirty Read

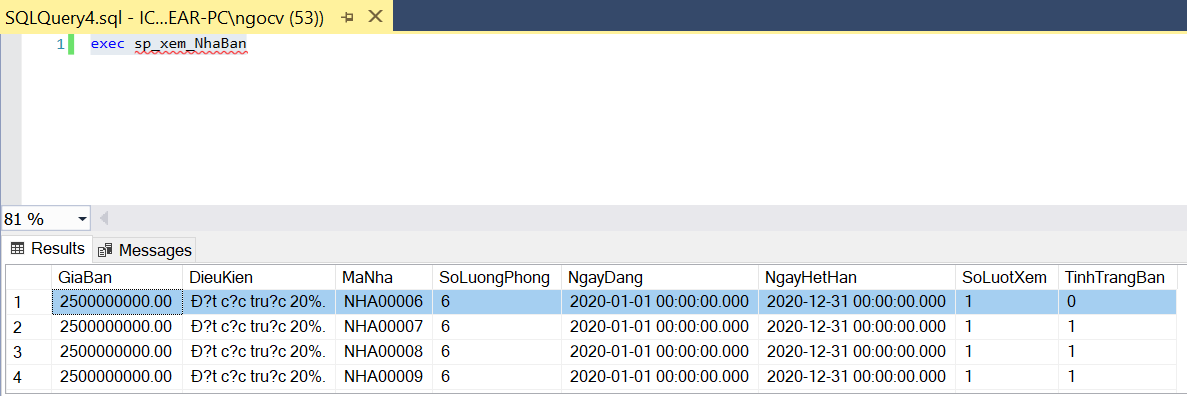
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR05: *Dirty Read***  T1 (User = NhanVien): Thực hiện thêm một hợp đồng (NV có mã là ‘NV000006’)  T2 (User = NguoiThue): Thực hiện thống kê NhaBan  Ví dụ: @maHD=’HD000006’, @maNha=’NHA00006’, @maNT=’NT06’  Ở VD này mã Người Thuê không hợp lệ làm cho T1 bị rollback 🡪 T2 select bảng nhà sẽ đọc dữ liệu sai | | | |
| **sp\_them\_HDNhaBan** | **Khóa** | **sp\_xem\_NhaBan** | **Khóa** |
| ***Input:*** @m*aHD, @maNha và @maNT*  ***Output***: Cập nhật lại tình trạng của căn nhà được bán và thêm 1 hợp đồng mới | ***Input***:  ***Output***: Danh sách NhaBan |
| SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Cập nhật tình trạng căn nhà được bán  update NhaBan set TinhTrangBan = 0 where MaNha=@maNha | **X(NhaBan)**  Xin khoá ghi trên bảng NhaBan |  |  |
| B2: Thêm 1 dòng mới vào bảng Hợp Đồng  insert into HopDong(MaHD,LoaiHD,ThoiGian,MaNT,MaNha) values (@maHD,0,GETDATE(),@maNT,@maNha) | **X(HopDong)**  Xin khoá ghi trên bảng HopDong |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Thống kê Nhân viên  select \* from NhaBan | **R(NhaBan)** |
|  |  | COMMIT |  |
| if not exists(select \* from NhaBan where MaNha=@maNha) or not exists(select \* from NguoiThue where MaNT=@maNT)  begin    RAISERROR('Them hop dong khong thanh cong',1,1)  ROLLBACK TRAN sp\_them\_HDNhaBan  end  else  COMMIT TRAN sp\_them\_HDNhaBan |  |  |  |

Kết quả: Hợp đồng không được thêm thành công tức ‘NHA00006’ chưa được bán (TinhTrangNha=1) nhưng khi T2 đọc dữ liệu thì tình trạng nhà của ‘NHA00006’ lại là 0 nghĩa là đã bán 🡪 Dirty Read

* **User**: Nhân viên



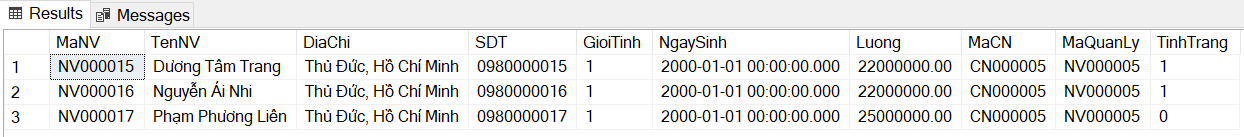
* **User**: Người thuê



## Tình huống 6: Lost Update

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR06: *Lost Update***  T1 (User = NhanVienQuanLy): Thực hiện tăng lương của 1 nhân viên trong chi nhánh lên 5tr  T2 (User = CEO): Thực hiện tăng hệ số lương của 1 chi nhánh  Ví dụ:  T1: NhanVienQuanLy chi nhánh có maCN=’CN000005’ thay đổi lương của nhân viên có maNV=’NV000017’ tăng lên 1 khoảng @Luong=5tr  T2: CEO thay đổi lương của toàn bộ chi nhánh có maCN=’CN000005’ lên 1.1 lần  Ở VD này khi NhanVienQuanLy cập nhật lương tăng 5tr tức Luong = 25tr cho nhân viên có MaNV=’NV000017’ sau đó CEO lại tăng lương thêm 1.1 lần thì lương sẽ là 27.5tr 🡪 nhưng ở đây do cả 2 đều ghi trên dữ liệu của nhân viên NV000015 nên mức lương của nó chỉ nhận cái commit sau cùng đó là 25tr và bị mất việc CEO update hệ số lên 1.1 | | | |
| **sp\_tang\_Luong\_nhanvien** | **Khóa** | **sp\_tang\_luong\_theochinhanh** | **Khóa** |
| ***Input:*** @m*aNV, @maCN và @Luong*  ***Output***: Cập nhật lại Luong của nhân viên đó là Luong= Luong+@Luong | ***Input***: @maCN, @heso  ***Output***: Luong của cả chi nhánh có mã là @maCN sẽ tăng lên @heso lần |
| SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Đọc Luong của Nhân viên cần tăng  declare @setluong money  select @setluong=Luong from NhanVien where MaNV=@maNV | **R(NhanVien)**  Dọc Dữ liệu Luong trên 1 dòng của bảng NhanVien |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B2: Tăng lương của Nhân viên thuộc Chi nhánh theo hệ số  update NhanVien set Luong = Luong\*@hesoLuong where MaCN=@maCN | **X(NhanVien)** |
|  |  | COMMIT |  |
| B2: Tăng lương của Nhân viên thuộc Chi nhánh lên một lượng là @Luong  set @setluong=@setluong+@Luong  update NhanVien set Luong = @setluong where MaNV=@maNV | **X(NhanVien)** |  |  |
| COMMIT |  |  |  |

Kết quả:



# Sinh viên thực hiện: 18120217

## Tình huống 7: Unrepeatable Read

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR07: *Unrepeatable Read***  T1 (User = Nhân viên): thực hiện thống kê Người thuê nhà thuộc chi nhánh  T2 (User = người thuê): người thuê NT000001 thực hiện update thông tin của mình  Ví dụ: @MaNT=’NT000001’@MaCN=’CN000001’ 🡪 T1 không đọc được update dữ liệu | | | |
| **sp\_Xem\_NguoiThue** | **Khóa** | **sp\_Update\_NguoiThue** | **Khóa** |
| ***Input:*** @MaCN  ***Output***: Danh sách người thuê | ***Input***: @MaNT, @TenNT  ***Output***: Cập nhật lại tên mới của người thuê |
| SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** |
| BEGIN TRAN |  |  |  |
| B1: thống kê danh sách người thuê  select \*  from NguoiThue  where MaCN=@MaCN | **R(NguoiThue)**  Xin khoá đọc trên bảng NguoiThue với đk MaCN=’ CN000001’ |  |  |
| WAITFOR DELAY '00:00:20' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | Update tên NguoiThue  update NguoiThue set TenNT = @TenNT where MaNT=@MaNT | **X(NguoiThue)**  Xin khóa ghi trên bảng NguoiThue |
|  |  | COMMIT |  |
| B2: thống kê danh sách người thuê  select \*  from NguoiThue  where MaCN=@MaCN  COMMIT |  |  |  |

Table

Description automatically generatedKết quả:

## Tình huống 8: Phantom

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Phantom***  T1 (User = ngườithue): thực hiện thống kê nhà thuê  T2 (User = chủ nhà): them 1 nha thuê  Vd: @MaNT=’NT000001’@MaCN=’ CN000001’ 🡪 T1 không đọc được update dữ liệu | | | |
| **sp\_Xem\_NhaThue** | **Khóa** | **Sp\_Them\_NhaThue** | **Khóa** |
| ***Input:***  ***Output***: Danh sách nhà thuê | ***Input***: @GiaThue,@MaNha,@SLphong,@Dang,@HetHan,  @HetHan,@SLXem,@TinhTrang  ***Output***: them 1 nhà thuê |
| SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: thống kê danh sách nhà thuê  SELECT \*  FROM NhaThue | **R(NhaThue)**  Xin khoá đọc trên bảng NhaThue |  |  |
| WAITFOR DELAY '00:00:20' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Thêm 1 dòng mới vào bảng NhaThue  INSERT INTO NhaThue(GiaThue,MaNha,SoLuongPhong, NgayDang, NgayHetHan, SoLuotXem, TinhTrangThue) VALUES | **X(NhaThue)**  Xin khóa ghi trên bảng NhaThue |
|  |  | B2: Kiểm tra thông tin mã NhaThue có trong danh sách Nha và có đúng chủ nhà hay không  IF NOT EXISTS(SELECT \* FROM Nha WHERE MaNha=@maNha AND MaChuNha = @maChuNha)  BEGIN    RAISERROR('Them nha khong thanh cong',1,1)  ROLLBACK TRAN sp\_them\_NhaThue  END  ELSE  COMMIT TRAN sp\_them\_NhaThue | **X(NhaThue)**  Xin khóa ghi trên bảng NhaThue |
| B2: thống kê danh sách nhà thuê  SELECT \*  FROM NhaThue  COMMIT |  |  |  |

Table

Description automatically generatedKết quả:

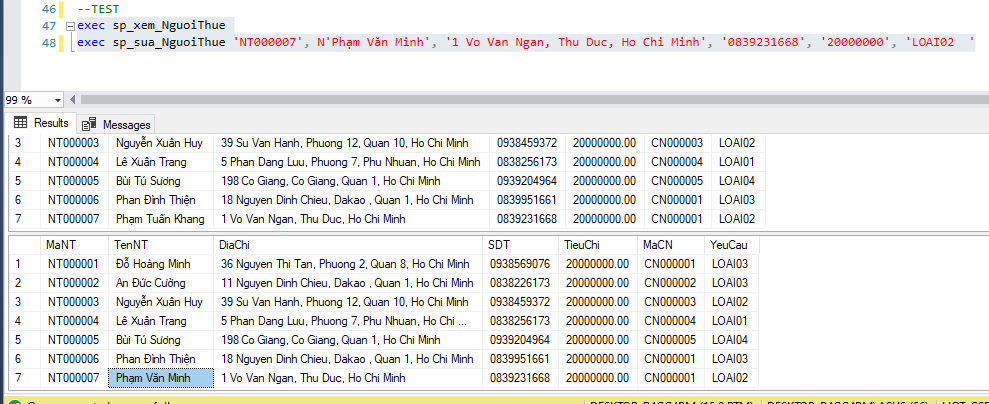
# Sinh viên thực hiện: 18120227

## Tình huống 9: Unrepeatable Read

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR09: *Unrepeatable Read***  T1 (User = NhanVien): Thực hiện xem danh sách NguoiThue  T2 (User = NguoiThue): Thực hiện cập nhật thông tin NguoiThue  Ví dụ: @maNT = ‘NT000001’ , @tenNT = N’Phạm Văn Minh’, @diaChi = ‘36 Nguyen Thi Tan, Phuong 2, Quan 8, Ho Chi Minh’, @sdt = ‘0938569076’, @tieuChi = ‘20000000’, @yeuCau = ‘LOAI03 ‘  Ở VD này một dòng dữ liệu trong NguoiThue được sửa khi đang được đọc bởi người dùng khiến lần đọc sau không còn thấy dữ liệu ban đầu nữa | | | |
| **sp\_xem\_NguoiThue** | **Khóa** | **sp\_sua\_NguoiThue** | **Khóa** |
| ***Input:***  ***Output***: Danh sách NguoiThue | ***Input***: @maNT, @tenNT, @diaChi, @sdt, @tieuChi, @yeuCau  ***Output***: Sửa 1 dòng trong bảng NguoiThue |
| SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu bảng NguoiThue  SELECT \* FROM NguoiThue | **R(NhaBan)** |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Kiểm tra thông tin mã NguoiThue có đúng không  IF NOT EXISTS(SELECT \* FROM NguoiThue WHERE MaNT = @maNT)  BEGIN  RAISERROR('Khong ton tai ma nguoi thue nay.',1,1)  ROLLBACK TRAN sp\_sua\_NguoiThue  END  ELSE |  |
|  |  | B2: Sửa dòng dữ liệu có MaNT = @maNT  UPDATE NguoiThue SET TenNT = @tenNT, DiaChi = @diaChi, SDT = @sdt, TieuChi = @tieuChi, YeuCau = @yeuCau WHERE MaNT = @maNT  COMMIT TRAN sp\_sua\_NguoiThue | **X(NguoiThue)** |
| B2: Đọc dữ liệu bảng NguoiThue  SELECT \* FROM NguoiThue | **R(NhaBan)** |  |  |
| COMMIT TRAN |  |  |  |
| ***Input:***  ***Output***: Danh sách NguoiThue |  | ***Input***: @maNT, @tenNT, @diaChi, @sdt, @tieuChi, @yeuCau  ***Output***: Sửa 1 dòng trong bảng NguoiThue |  |

Kết quả: Hai lần đọc của T1 cho ra 2 kết quả khác nhau tại dòng có MaNT = ‘NT000007’ với TenNT của lần 1 là Phạm Tuấn Khang, lần 2 là Phạm Văn Minh (mới được insert) à Unrepeatable Read

* **User**: Nhân viên

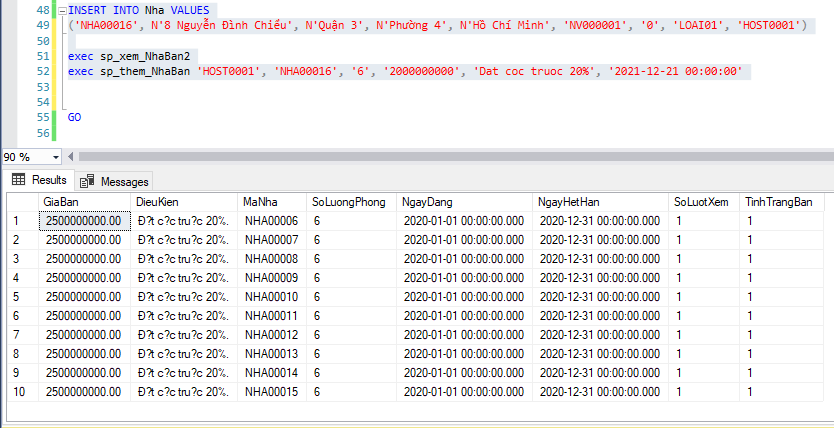


## Tình huống 10: Phantom

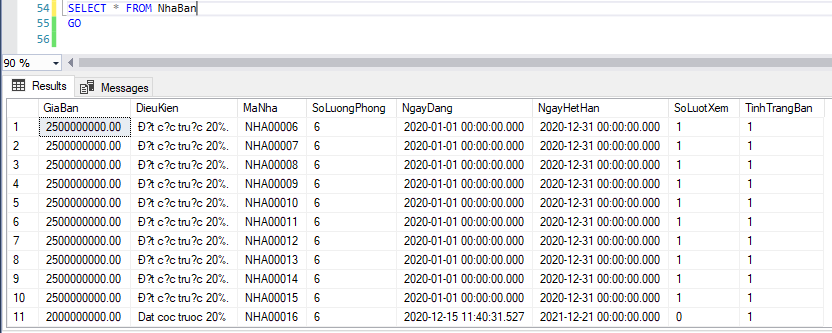
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR10: *Phantom***  T1 (User = NguoiThue): Thực hiện xem danh sách Nhà  T2 (User = ChuNha): Thực hiện thêm NhaBan  VD: @maChuNha = ‘HOST0001’, @maNha = ‘NHA00016’, @soLuongPhong = ‘6’, @giaBan = ‘2000000000’, @dieuKien=’Dat coc truoc 20%’, @ngayHetHan = ‘2021-12-21 00:00:00’  Ở VD này danh sách nhà bán được thêm 1 dòng trong khi đang được xem bởi người dùng khiến cho dữ liệu đọc lên bị sai (thiếu) | | | |
| **sp\_xem\_NhaBan** | **Khóa** | **sp\_them\_NhaBan** | **Khóa** |
| ***Input:***  ***Output***: Danh sách NhaBan | ***Input***: @maChuNha, @maNha, @soLuongPhong, @giaBan, @dieuKien, @ngayHetHan  ***Output***: Thêm 1 dòng trong danh sách NhaBan |
| SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu bảng NhaBan  SELECT \* FROM NhaBan |  |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Thêm 1 dòng mới vào bảng NhaBan  INSERT INTO NhaBan(MaNha,SoLuongPhong,GiaBan,DieuKien,NgayDang, NgayHetHan, SoLuotXem, TinhTrangBan) VALUES (@maNha, @soLuongPhong ,@giaBan, @dieuKien, GETDATE(), @ngayHetHan, 0, 1) | **X(NhaBan)** |
|  |  | B2: Kiểm tra thông tin mã NhaBan có trong danh sách Nha và có đúng chủ nhà hay không  IF NOT EXISTS(SELECT \* FROM Nha WHERE MaNha=@maNha AND MaChuNha = @maChuNha)  BEGIN    RAISERROR('Them nha ban khong thanh cong',1,1)  ROLLBACK TRAN sp\_them\_NhaBan  END  ELSE  COMMIT TRAN sp\_them\_NhaBan |  |
| B2: Đọc dữ liệu bảng NhaBan  SELECT \* FROM NhaBan |  |  |  |
| COMMIT TRAN |  |  |  |
| ***Input:***  ***Output***: Danh sách NhaBan |  | ***Input***: @maChuNha, @maNha, @soLuongPhong, @giaBan, @dieuKien, @ngayHetHan  ***Output***: Thêm 1 dòng trong danh sách NhaBan |  |
| SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** |  | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |  |

Kết quả: NhaBan được thêm 1 dòng dữ liệu (NHA00016) nhưng T1 lại không đọc được dòng dữ liệu mới thêm vào à Phantom

**User**: Người thuê



* **User**: Chủ nhà



# XỬ LÝ TÌNH HUỐNG TRANH CHẤP

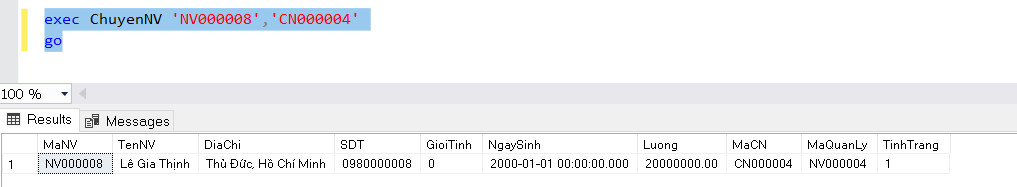
# Sinh viên thực hiện: 18120213

## Tình huống 1: Dirty Read

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Dirty read***  T1 (User = CEO): thực hiện update chi nhánh và nhân viên quản lý cho 1 nhân viên  T2 (User = QLCN): thực hiện thống kê nhân viên tại các chi nhánh  Ví dụ trường hợp: CEO chuyển công tác cho nhân viên với mã là NV000008, update Mã chi nhánh mới CN000004 và dựa trên Mã chi nhánh mới mà update Mã nhân viên quản lý theo chi nhánh đó. Tại thời điểm đó, QLCN cũng đang thống kê nhân viên ở các chi nhánh.  Khắc phục: vì uncommited sẽ gây ra Diry read nên chỉnh lại Isolation level của T2 thành Commited để không đọc phải dữ liệu rác và ta sẽ có kết quả đúng | | | |
| **sp\_ChuyenNV\_uncommited** | **Khóa** | **sp\_XemNV** | **Khóa** |
| ***Input:*** @manv, @mcn, @mql  ***Output***: Cập nhập chi nhánh và nhân viên quản lý mới cho nhân viên đã chọn | ***Input***:  ***Output***: |
| SET TRANSACTION ISOLATION  LEVEL COMMITTED. | SET TRANSACTION ISOLATION  LEVEL COMMITTED |
| BEGIN TRAN |  |  |  |
| B1: Cập nhập chi nhánh mới  update NhanVien  set MaCN=@mcn  where MaNV=@manv | **X(Nhân viên)**  Xin khóa trên nhân viên |  |  |
| B2: Cập nhập nhân viên quản lý mới  update NhanVien  set MaQuanLy=(select distinct MaNV from ChiNhanh where MaCN=@mcn) where MaNV=@manv |  |  |  |
| B3: Thống kê Nhân viên  select \* from NhanVien where MaNV=@manv | R(Nhân viên) |  |  |
| if(not exists(select \* from ChiNhanh where MaCN=@mcn) or not exists(select \* from NhanVien where MaNV=@manv))  begin  RAISERROR('Chuyen nhan vien khong thanh cong',1,1)  ROLLBACK TRAN sp\_ChuyenNV\_uncommited |  |  |  |
| COMMIT | Mở khóa ở bảng nhân viên |  |  |
|  |  | Begin Tran |  |
|  |  | B1: Thống kê nhân viên  select \* from NhanVien | *R(Nhân Viên)*  *Xia khóa ở bảng nhân viên* |
|  |  | Commit |  |

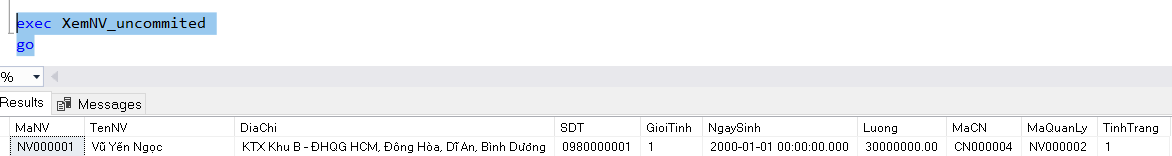
Kết quả: Dựa vào kết quả bên dưới ta thấy được đáng lẽ QLCN phải thấy được cả hai yếu tố update của CEO nhưng lần này sẽ thấy được mã chi nhánh lẫn mã nhân viên được cập nhập theo => Fix thành công lỗi Dirty read

User: CEO

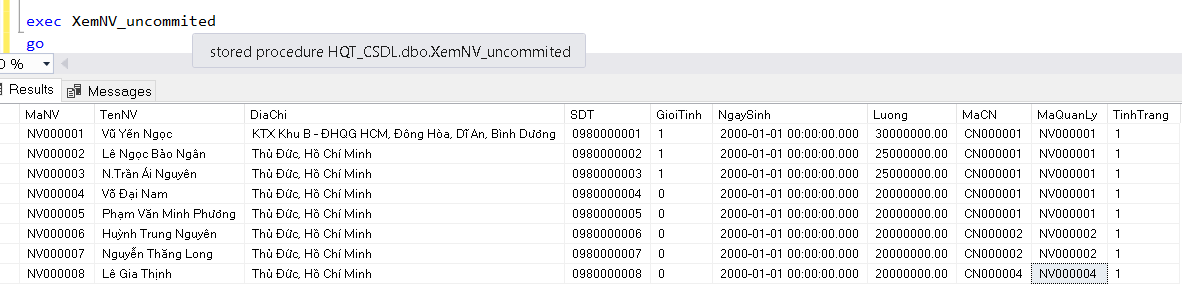


User: QLCN

Trước khi sửa:



Sau khi Sửa:

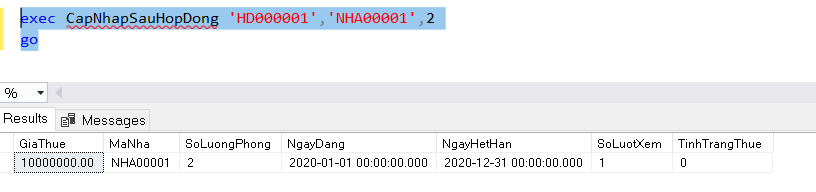


## 2. Tình huống 2: Lost Update

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR02: Lost Update**  T1 (User = Nhân viên): thực hiện update số phòng sau khi hợp đồng thành công  T2 (User = Chủ nhà): thực hiện update số phòng còn lại mới  Ví dụ trường hợp: Sau khi làm hợp đồng cho thuê phòng xong của căn nhà có mã là NHA00001, nhân viên sẻ cập nhập lại số phòng=4 – số phòng đã cho thuê theo hợp đồng (ở đây là 2 phòng). Tại thời điểm đó chủ nhà bổng dư phòng nên chủ nhà cập nhập thêm 4 phòng của nhà mình.  Khắc phục: Sử dụng Isolation level Repeatable Read để giải quyết Lost data  Sử dụng Uplock để giải quyết Deadlock ở T2 khi T2 đang chuẩn bị Update | | | |
| **sp\_CapNhapSau** | **Khóa** | **sp\_CapNhap** | **Khóa** |
| ***Input*** @mhd, @manha, @sphd  ***Output***: Cập nhập Số phòng | ***Input***: @manha, @spt  ***Output***: Cập nhập số phòng |
| SET TRANSACTION ISOLATION  LEVEL COMMITTED. | SET TRANSACTION ISOLATION  LEVEL UNCOMMITTED |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu tại bảng Nhà thuê  declare @temp int  select @temp=SoLuongPhong from NhaThue with(Uplock) where MaNha=@manha | **R(Nhà thuê)// Xin khóa đọc tại Nhà thuê** |  |  |
| B2: Cập nhập số phòng mới  update NhaThue  set SoLuongPhong=@temp-@sphd  where MaNha=@manha | X(Nhà thuê)//Xin khóa ghi |  |  |
| B3: Thống kê lại số phòng  select \* from NhaThue where MaNha=@manha | R(Nhà thuê) |  |  |
| if(not exists(select \* from HopDong where MaHD=@mhd) or not exists(select\* from NhaThue where MaNha=@manha))  begin  RAISERROR('Cap Nhap khong thanh cong',1,1)  ROLLBACK TRAN sp\_CapNhapSau  End |  |  |  |
| Commit tran | //Trả khóa lại |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Đọc dữ liệu tại bảng nhà thuê  declare @temp int  select @temp=SoLuongPhong from NhaThue with Uplock where MaNha=@manha | **R(Nhà thuê)**  //Xin khóa đọc |
|  |  | B2: Cập nhập số phòng mới  update NhaThue  set SoLuongPhong=@temp+@spt  where MaNha=@manha | X(Nhà thuê)// Xin khóa ghi |
|  |  | if(not exists(select\* from NhaThue where MaNha=@manha))  begin  RAISERROR('Cap Nhap khong thanh cong',1,1)  ROLLBACK TRAN sp\_CapNhap  End |  |
|  |  | COMMIT |  |

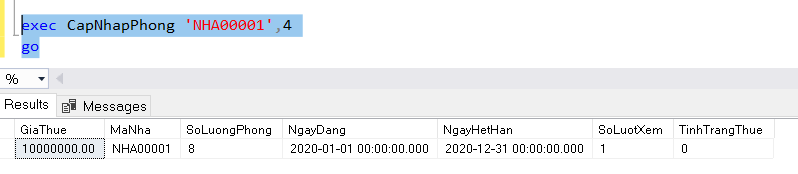
Kết quả: Dựa vào hình ở dưới ta thấy ở nhân viên số phòng là 2, ở chủ nhà số phòng là 8, điều này vô lý bởi vì số phòng đáng lẽ là 4+4-2=6 mà ở đây là 2 và 6 => Khắc phục Lost Update

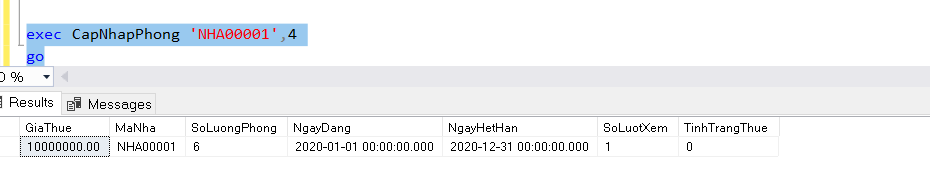
User: Nhân viên



User: Chủ nhà

Trước:



Sau:

//Chỗ này để nhét sửa tình huống tranh chấp của Ngân

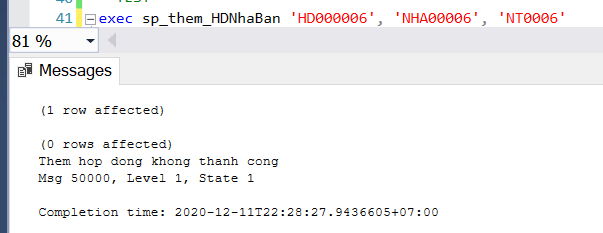
# Sinh viên thực hiện: 18120215

## Tình huống 5: Dirty Read

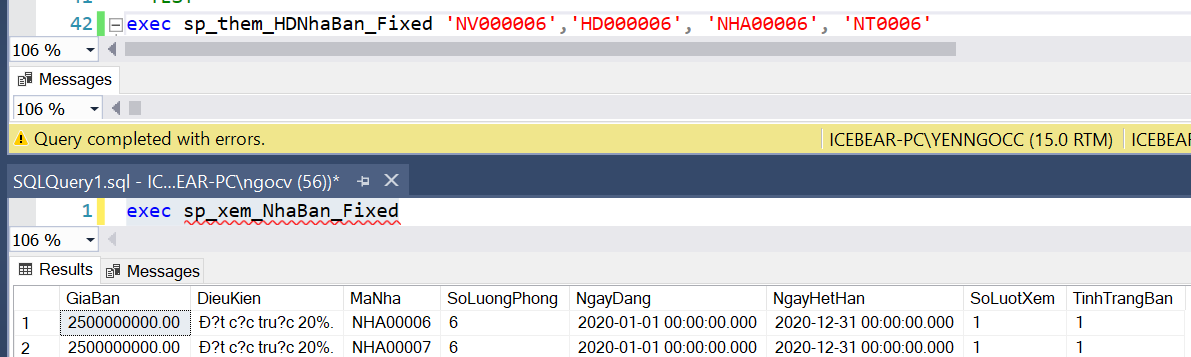
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR05: *Dirty read***  T1 (User = NhanVien): Thực hiện thêm một hợp đồng (NV có mã là ‘NV000006’)  T2 (User = NguoiThue): Thực hiện thống kê NhaBan  VD: @maHD=’HD000006’, @maNha=’NHA00006’, @maNT=’NT06’  Sửa lại mức cô lập để không bị lỗi Dirty Read -> T2 sẽ có mức cô lập mời là READ COMMITED để không đọc dữ liệu rác khi T1 bị rollback | | | |
| sp\_them\_HDNhaBan | **Khóa** | sp\_xem\_NhaBan | **Khóa** |
| ***Input:*** @m*aHD, @maNha và @maNT*  ***Output***: Cập nhật lại tình trạng của căn nhà được bán và thêm 1 hợp đồng mới | ***Input***:  ***Output***: Danh sách NhaBan |
| SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Cập nhật tình trạng căn nhà được bán  update NhaBan set TinhTrangBan = 0 where MaNha=@maNha | **X(NhaBan)**  Xin khoá ghi trên bảng NhaBan |  |  |
| B2: Thêm 1 dòng mới vào bảng Hợp Đồng  insert into HopDong(MaHD,LoaiHD,ThoiGian,MaNT,MaNha) values (@maHD,0,GETDATE(),@maNT,@maNha) | **X(HopDong)**  Xin khoá ghi trên bảng HopDong |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
| if not exists(select \* from NhaBan where MaNha=@maNha) or not exists(select \* from NguoiThue where MaNT=@maNT)  begin    RAISERROR('Them hop dong khong thanh cong',1,1)  ROLLBACK TRAN sp\_them\_HDNhaBan  end  else  COMMIT TRAN sp\_them\_HDNhaBan |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Thống kê Nhân viên  select \* from NhaBan | **R(NhaBan)**  Xin khóa đọc trên bảng NhaBan giữ đến hết lệnh |
|  |  | COMMIT |  |

Kết quả: Hợp đồng không được thêm thành công tức ‘NHA00006’ chưa được bán (TinhTrangNha=1) 🡪 T2 đợi T1 xxong thì mới đọc dữ liệu thì tình trạng nhà của ‘NHA00006’ 1 nghĩa là chưa bán 🡪 Đã sửa Dirty Read

T1:

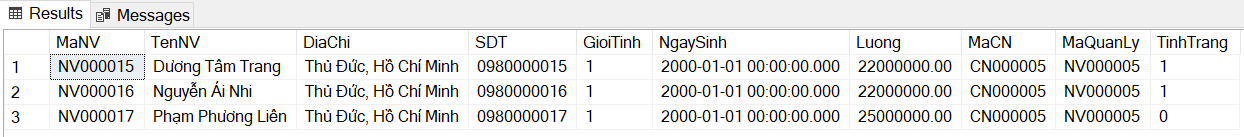


T2:

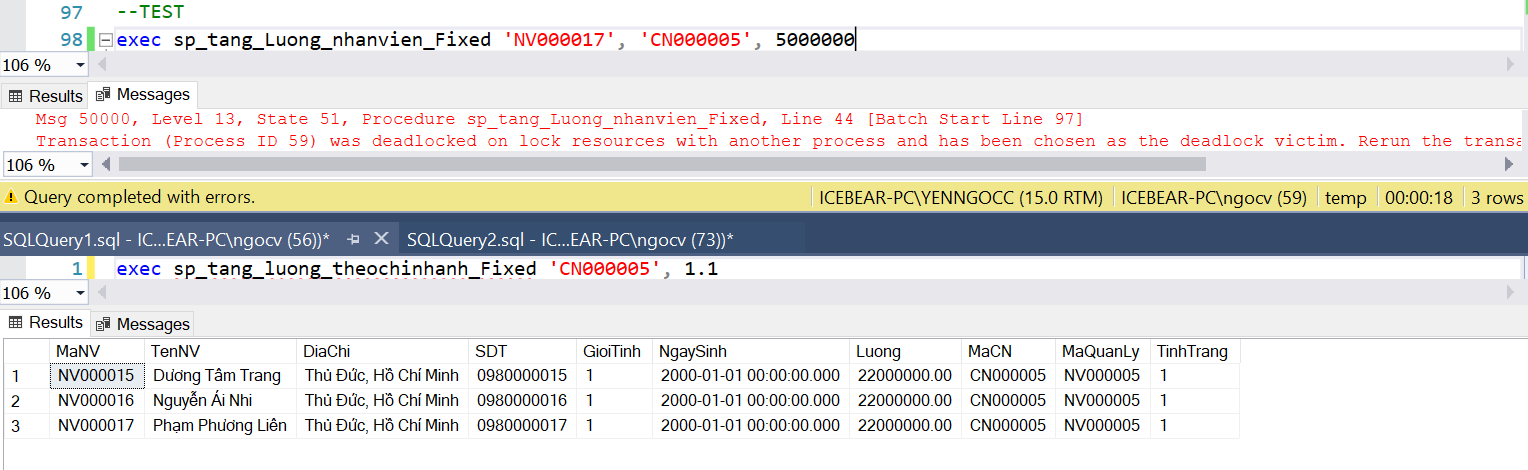


## Tình huống 6: Lost Update

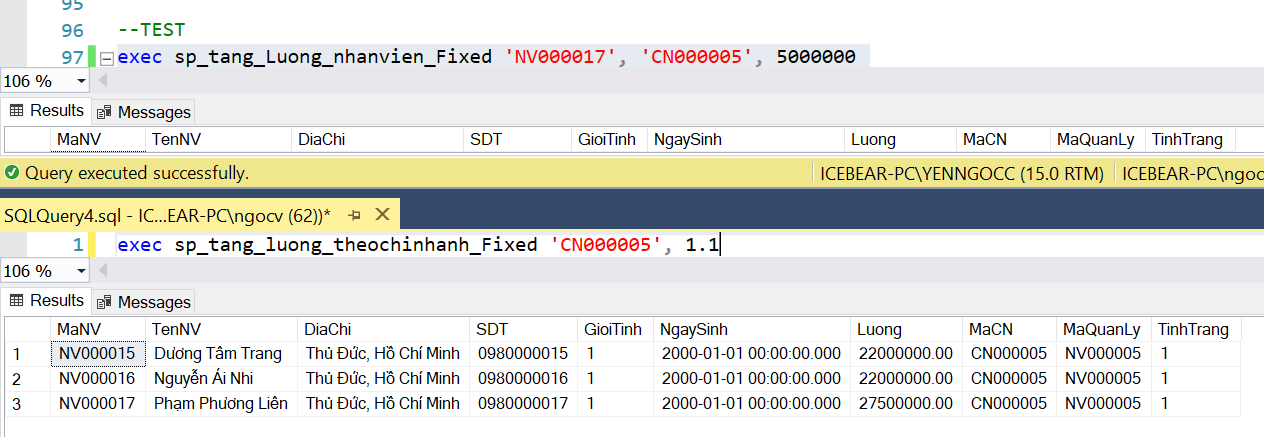
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR06: *Lost Update***  T1 (User = NhanVienQuanLy): Thực hiện tăng lương của 1 nhân viên trong chi nhánh lên 5tr  T2 (User = CEO): Thực hiện tăng hệ số lương của 1 chi nhánh  VD:  T1: NhanVienQuanLy chi nhánh có maCN=’CN000005’ thay đổi lương của nhân viên có maNV=’NV000017’ tăng lên 1 khoảng @Luong=5tr  T2: CEO thay đổi lương của toàn bộ chi nhánh có maCN=’CN000005’ lên 1.1 lần  Ở VD ta sẽ sửa lại mức cô lập ở T1 là UNREPEATABLE để việc đọc dữ liệu lên của giao tác T1 sẽ giữ khóa đọc suốt cả transaction 🡪 tránh việc đọc dữ liệu lên từ T2.  Sau khi sửa lỗi thì ta gặp một lỗi khác đó là DEADLOCK cụ thể là Conversion deadlock.  Để khắc phục ta thêm một UPD LOCK vào việc đọc dữ liệu của T1 từ bảng Nhân Viên để giữ khóa đọc và chuẩn bị được update lên khóa ghi 🡪 Để hệ thống không cung cấp khóa ghi cho T2 gây ra Deadlock. | | | |
| sp\_tang\_Luong\_nhanvien | **Khóa** | sp\_tang\_luong\_theochinhanh | **Khóa** |
| ***Input:*** @m*aNV, @maCN và @Luong*  ***Output***: Cập nhật lại Luong của nhân viên đó là Luong= Luong+@Luong | ***Input***: @maCN, @heso  ***Output***: Luong của cả chi nhánh có mã là @maCN sẽ tăng lên @heso lần |
| SET TRANSACTION ISOLATION  LEVEL **REPEATABLE READ** | SET TRANSACTION ISOLATION  LEVEL **REPEATABLE READ** |
| BEGIN TRAN |  |  |  |
| B1: Đọc Luong của Nhân viên cần tăng  declare @setluong money  select @setluong=Luong from NhanVien with (UPDLOCK) where MaNV=@maNV | **R(NhanVien)**  Lấy khóa đọc trên bảng NhanVien  đến hết T1 và báo với hệ thống rằng sẽ ghi dữ liệu |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
| B2: Tăng lương của Nhân viên thuộc Chi nhánh lên một lượng là @Luong  set @setluong=@setluong+@Luong  update NhanVien set Luong = @setluong where MaNV=@maNV | **X(NhanVien)**  Khóa ghi ở bảng NhanVien |  |  |
| COMMIT |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B2: Tăng lương của Nhân viên thuộc Chi nhánh theo hệ số  update NhanVien set Luong = Luong\*@hesoLuong where MaCN=@maCN | **X(NhanVien)**  Khóa ghi ở bảng NhanVien |
|  |  | COMMIT |  |

Kết quả:

Kết quả sau khi sửa mức cô lập bị Conversion Deadlock làm cho RollBack T1 chỉ thực hiện T2 🡪 do MS SQL Sever tự xử lý Deadlock:



Sau khi thêm UPDLOCK 🡪 Đã sửa và dữ liệu không bị mất việc update của cả 2 Transaction



# Sinh viên thực hiện: 18120217

## Tình huống 7: Unrepeateale Read

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR07: Unrepeatable *read***  T1 (User = nhân viên): thực hiện thống kê số người thuê của nhà mình  T2 (User = người thuê): người thuê NT000001 thực hiện update thông tin của mình  Vd: @MaNT=’NT000001’@MaCN=’CN000001’   * T1 không đọc được update dl | | | |
| **sp\_Xem\_NguoiThue** | **Khóa** | **sp\_Update\_NguoiThue** | **Khóa** |
| ***Input:*** @MaCN  ***Output***: Danh sách người thuê | ***Input***: @MaNT, @TenNT  ***Output***: Cập nhật lại tên mới của người thuê |
| SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** | SET TRANSACTION ISOLATION  LEVEL **READ UNCOMMITED** |
| BEGIN TRAN |  |  |  |
| B1: thống kê danh sách người thuê  select \*  from NguoiThue  where MaCN=@MaCN | **R(NguoiThue)**  Xin khoá đọc trên bảng NguoiThue với đk MaCN=’ CN000001’ |  |  |
| WAITFOR DELAY '00:00:20' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | Update tên NguoiThue  update NguoiThue set TenNT = @TenNT where MaNT=@MaNT | **X(NguoiThue)**  Xin khóa ghi trên bảng NguoiThue |
|  |  | COMMIT |  |
| B2: thống kê danh sách người thuê  select \*  from NguoiThue  where MaCN=@MaCN  COMMIT |  |  |  |

A picture containing application

Description automatically generated

Kết quả không đổi 🡪 T2 phải chờ T1 đọc xong mới cho update

Graphical user interface, text, application, email

Description automatically generated

## Tình huống 8: Phantom

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR08: *Phantom***  T1 (User = ngườithue): thực hiện thống kê nhà thuê  T2 (User = chủ nhà): them 1 nha thuê  Vd: @MaNT=’NT000001’@MaCN=’ CN000001’   * T1 không đọc được update dl | | | |
| **sp\_Xem\_NhaThue** | **Khóa** | **Sp\_Them\_NhaThue** | **Khóa** |
| ***Input:***  ***Output***: Danh sách nhà thuê | ***Input***: @GiaThue,@MaNha,@SLphong,@Dang,@HetHan,  @HetHan,@SLXem,@TinhTrang  ***Output***: them 1 nhà thuê |
| SET TRANSACTION ISOLATION  LEVEL SERIALIZABLE | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: thống kê danh sách nhà thuê  SELECT \*  FROM NhaThue | **R(NhaThue)**  Xin khoá đọc trên bảng NhaThue |  |  |
| WAITFOR DELAY '00:00:20' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Thêm 1 dòng mới vào bảng NhaThue  INSERT INTO NhaThue(GiaThue,MaNha,SoLuongPhong, NgayDang, NgayHetHan, SoLuotXem, TinhTrangThue) VALUES | **X(NhaThue)**  Xin khóa ghi trên bảng NhaThue |
|  |  | B2: Kiểm tra thông tin mã NhaThue có trong danh sách Nha và có đúng chủ nhà hay không  IF NOT EXISTS(SELECT \* FROM Nha WHERE MaNha=@maNha AND MaChuNha = @maChuNha)  BEGIN    RAISERROR('Them nha khong thanh cong',1,1)  ROLLBACK TRAN sp\_them\_NhaThue  END  ELSE  COMMIT TRAN sp\_them\_NhaThue | **X(NhaThue)**  Xin khóa ghi trên bảng NhaThue |
| B2: thống kê danh sách nhà thuê  SELECT \*  FROM NhaThue  COMMIT |  |  |  |

Table

Description automatically generated

Kết quả không đổi 🡪 t2 phải chờ T1

A picture containing graphical user interface

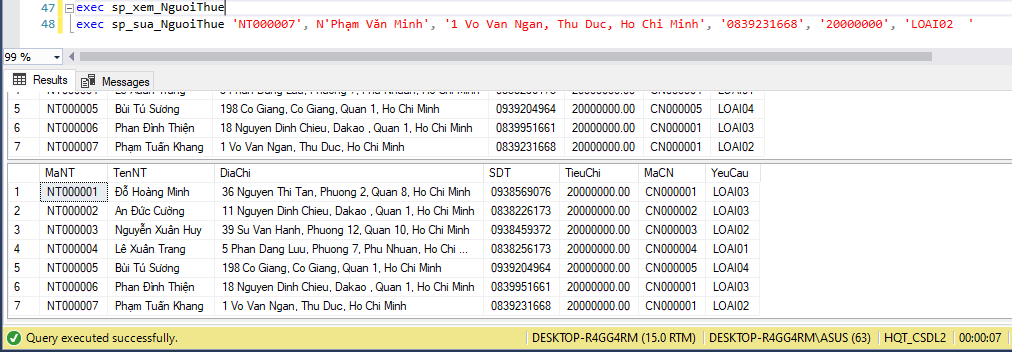
Description automatically generated

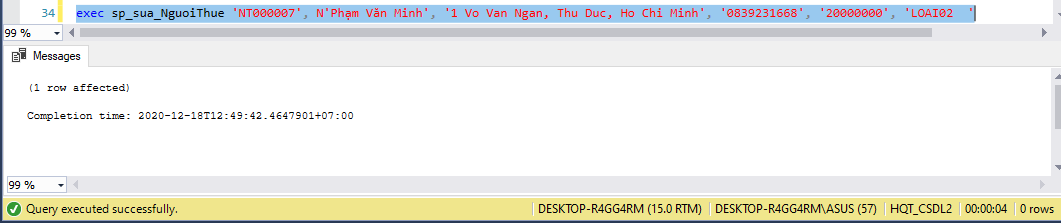
# Sinh viên thực hiện: 18120227

## Tình huống 9: Unrepeatable Read

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR09:  *Unrepeatable read***  T1 (User = NhanVien): Thực hiện xem danh sách NguoiThue  T2 (User = NguoiThue): Thực hiện cập nhật thông tin NguoiThue  VD: @maNT = ‘NT000001’ , @tenNT = N’Phạm Văn Minh’, @diaChi = ‘36 Nguyen Thi Tan, Phuong 2, Quan 8, Ho Chi Minh’, @sdt = ‘0938569076’, @tieuChi = ‘20000000’, @yeuCau = ‘LOAI03 ‘  Ở VD này nếu lỗi Unrepaeatable read xảy ra thì một dòng dữ liệu trong NguoiThue được sửa khi đang được đọc bởi người dùng khiến lần đọc sau không còn thấy dữ liệu ban đầu nữa. Thiết đặt Transaction Isolation Level thành Repeatable Read để ngăn lỗi. | | | |
| sp\_xem\_NguoiThue | **Khóa** | sp\_sua\_NguoiThue | **Khóa** |
| ***Input:***  ***Output***: Danh sách NguoiThue | ***Input***: @maNT, @tenNT, @diaChi, @sdt, @tieuChi, @yeuCau  ***Output***: Sửa 1 dòng trong bảng NguoiThue |
| SET TRANSACTION ISOLATION  LEVEL **REPEATABLE READ** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu bảng NguoiThue  SELECT \* FROM NguoiThue | **S(NguoiThue)** |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Kiểm tra thông tin mã NguoiThue có đúng không  IF NOT EXISTS(SELECT \* FROM NguoiThue WHERE MaNT = @maNT)  BEGIN  RAISERROR('Khong ton tai ma nguoi thue nay.',1,1)  ROLLBACK TRAN sp\_sua\_NguoiThue  END  ELSE |  |
|  |  | B2: Sửa dòng dữ liệu có MaNT = @maNT  UPDATE NguoiThue SET TenNT = @tenNT, DiaChi = @diaChi, SDT = @sdt, TieuChi = @tieuChi, YeuCau = @yeuCau WHERE MaNT = @maNT  COMMIT TRAN sp\_sua\_NguoiThue | **X(NguoiThue)** |
| B2: Đọc dữ liệu bảng NguoiThue  SELECT \* FROM NguoiThue | **S(NguoiThue)** |  |  |
| COMMIT TRAN |  |  |  |

Kết quả: Hai lần đọc của T1 cho ra 2 kết quả giống nhau tại dòng có MaNT = ‘NT000007’. T2 phải chờ T1 đọc xong mới được phép sửa dữ liệu trong bảng người thuê.

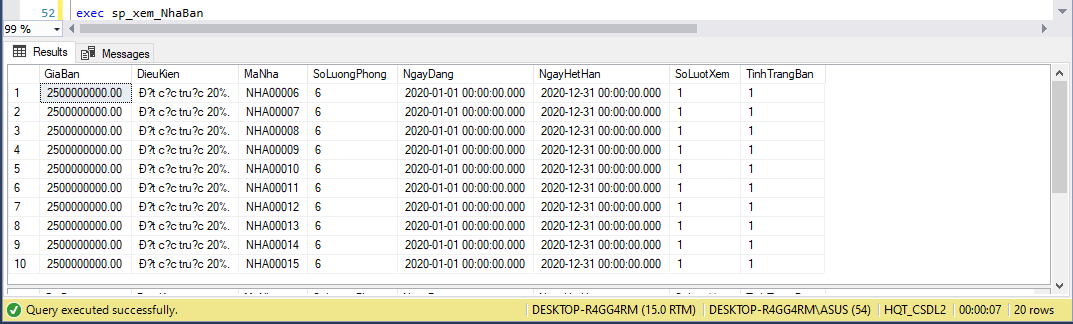
T1: 

T2: 

## Tình huống 10: Phantom

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR10:  *Phantom***  T1 (User = NguoiThue): Thực hiện xem danh sách Nhà  T2 (User = ChuNha): Thực hiện thêm NhaBan  VD: @maChuNha = ‘HOST0001’, @maNha = ‘NHA00016’, @soLuongPhong = ‘6’, @giaBan = ‘2000000000’, @dieuKien=’Dat coc truoc 20%’, @ngayHetHan = ‘2021-12-21 00:00:00’  Ở VD này, nếu lỗi Phantom xảy ra thì danh sách nhà bán được thêm 1 dòng trong khi đang được xem bởi người dùng khiến cho dữ liệu đọc lên bị sai (thiếu). Thiết đặt Transaction Isolation Level là Serializable tại T1(sp\_xem\_NhaBan) để ngăn lỗi. | | | |
| sp\_xem\_NhaBan | **Khóa** | sp\_them\_NhaBan | **Khóa** |
| ***Input:***  ***Output***: Danh sách NhaBan | ***Input***: @maChuNha, @maNha, @soLuongPhong, @giaBan, @dieuKien, @ngayHetHan  ***Output***: Thêm 1 dòng trong danh sách NhaBan |
| SET TRANSACTION ISOLATION  LEVEL **READ SERIALIZABLE** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITED** |
| BEGIN TRAN |  |  |  |
| B1: Đọc dữ liệu bảng NhaBan  SELECT \* FROM NhaBan | **R(NhaBan)** |  |  |
| WAITFOR DELAY '00:00: 07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Thêm 1 dòng mới vào bảng NhaBan  INSERT INTO NhaBan(MaNha,SoLuongPhong,GiaBan,DieuKien,NgayDang, NgayHetHan, SoLuotXem, TinhTrangBan) VALUES (@maNha, @soLuongPhong ,@giaBan, @dieuKien, GETDATE(), @ngayHetHan, 0, 1) | **X(NhaBan)** |
|  |  | B2: Kiểm tra thông tin mã NhaBan có trong danh sách Nha và có đúng chủ nhà hay không  IF NOT EXISTS(SELECT \* FROM Nha WHERE MaNha=@maNha AND MaChuNha = @maChuNha)  BEGIN    RAISERROR('Them nha ban khong thanh cong',1,1)  ROLLBACK TRAN sp\_them\_NhaBan  END  ELSE  COMMIT TRAN sp\_them\_NhaBan |  |
| B2: Đọc dữ liệu bảng NhaBan  SELECT \* FROM NhaBan | **R(NhaBan)** |  |  |
| COMMIT TRAN |  |  |  |

Kết quả: T2 phải chờ cho đến khi T1 commit thì mới được phép thêm dữ liệu trên bảng NhaBan

T1:

T2: Thời gian thực hiện dài hơn vì phải chờ T1.