**d**

**BÁO CÁO ĐỒ ÁN THỰC HÀNH ĐỢT 4**

**MÔN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

GVHD:

GV. Hồ Thị Hoàng Vy

GV. Tiết Gia Hồng

# **THÔNG TIN NHÓM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ tên** | **Công việc** | **% Hoàn thành** |
| **1** | 18127022 | Phạm Ngọc Thùy Trang | * Viết báo cáo đợt 1 + chỉnh sửa đợt cuối * Sửa 4 lỗi tranh chấp * Code giao diện các lỗi tranh chấp | **100%** |
| **2** | 18127053 | Ngô Thị Thanh Thảo | * Sửa 2 lỗi tranh chấp * Chỉnh sửa báo cáo đợt 2 | **100%** |
| **3** | 18127055 | Hoàng Nguyên Trúc | * Code giao diện chức năng login/create tài khoản * Sửa 2 lỗi tranh chấp | **90%** |

Mục lục

[THÔNG TIN NHÓM 2](#_Toc60507936)

[NGUỒN THAM KHẢO 4](#_Toc60507937)

[CÀI ĐẶT TÌNH HUỐNG TRANH CHẤP 5](#_Toc60507938)

[I. Sinh viên thực hiện: Phạm Ngọc Thùy Trang 5](#_Toc60507939)

[1. Tình huống 1: 5](#_Toc60507940)

[2. Tình huống 2: 15](#_Toc60507941)

[3. Tình huống 3: 22](#_Toc60507942)

[8. Tình huống 8: 31](#_Toc60507943)

[II. Sinh viên thực hiện: Hoàng Nguyên Trúc 41](#_Toc60507944)

[2. Tình huống 4: 41](#_Toc60507945)

[3. Tình huống 5: 50](#_Toc60507946)

[4. Tình huống 6: 54](#_Toc60507947)

[III. Sinh viên thực hiện: Ngô Thị Thanh Thảo 59](#_Toc60507948)

[5. Tình huống 7: 59](#_Toc60507949)

# NGUỒN THAM KHẢO

[1]: <https://stackoverflow.com/questions/2065627/repeatable-read-am-i-understanding-this-right>

[2]: Slide lý thuyết của cô Phạm Thị Bạch Huệ

[3]: Script code demo của cô Hồ Thị Hoàng Vy

[5]: <https://csharp-video-tutorials.blogspot.com/p/sql-server.html>

[4]: <https://stackoverflow.com/questions/2065627/repeatable-read-am-i-understanding-this-right>

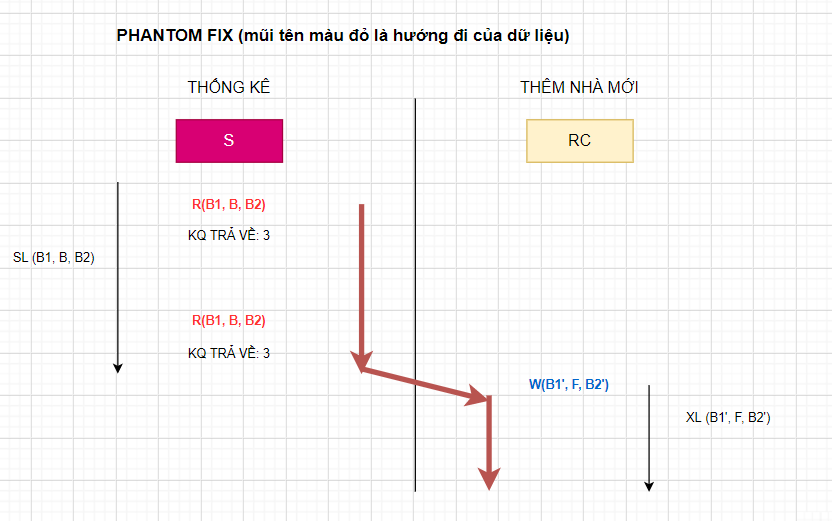
# CÀI ĐẶT TÌNH HUỐNG TRANH CHẤP

# Sinh viên thực hiện: Phạm Ngọc Thùy Trang

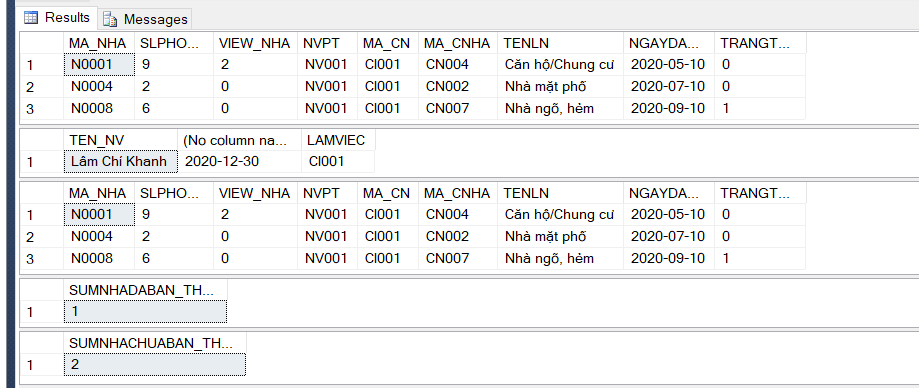
## Tình huống 1:

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Phantom***  T1 (User = NHANVIEN): Thống kê những nhà cần bán/cho thuê ở một địa điểm nhất định  T2 (User = CHUNHA): Thêm một nhà | | | |
| Thống kê những nhà cần bán/cho thuê ở một địa điểm nhất định | **Khóa** | Thêm một nhà | **Khóa** |
| ***Input:*** @MANV CHAR(5) = 'NV001'  ,@QUAN NVARCHAR(30) = NULL, @TPHO NVARCHAR(40) = 'TP.HCM' . Người dùng có thể dùng input mặc định hoặc thay đổi nếu muốn  ***Output***:  --Danh sách các nhà cần bán/cho thuê trong 3 tháng trở lại đây  -- Bài thống kê danh sách các nhà được đăng bán/cho thuê ở chi nhánh của nhân viên đó làm việc trong 3 tháng gồm có tên nhân viên lập thống kê, ngày báo cáo, danh sách, số lượng nhà đã bán, đã cho thuê, và tổng số lượng nhà chưa cho bán, cho thuê | ***Input***: @LOAINHA CHAR(5),  @NGAYHETHAN DATE,  @SLPHONG INTEGER,  @TINHTRANG NVARCHAR(100),  @GIABAN INTEGER,  @KHUVUC NVARCHAR(30),  @DUONG NVARCHAR(30),  @QUAN NVARCHAR(30),  @TPHO NVARCHAR(40),  @CHINHANHDK CHAR(5),  @DIEUKIENCHUNHA NVARCHAR(50)  ***Output***: Nếu nhà mới được thêm thành công thì  'Thêm nhà bán mới thành công',  Ngược lại nếu có lỗi xảy ra thì in ra lỗi tương ứng |
| SET TRANSACTION ISOLATION LEVEL SERIALIZABLE | SET TRANSACTION ISOLATION LEVEL READ COMMITTED |
| BEGIN TRAN |  |  |  |
| B1: Kiểm tra thông tin nhân viên có tồn tại hay không, nếu không tồn tại thì in câu lệnh báo lỗi và rollback transaction  IF (NOT EXISTS (SELECT \* FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV))  BEGIN  PRINT N'Không tồn tại mã nhân viên này'  ROLLBACK TRANSACTION;  RETURN;  END | **R(A)**  T1 xin khóa đọc trên bảng NHANVIEN với điều kiện Ma\_NV = @MANV và giữ khóa cho đến hết giao tác T1. T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong. Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng NHANVIEN (kể cả insert dòng mới) phải đợi cho đến khi T1 thực hiện xong |  |  |
| B2: Tìm ra chi nhánh mà nhân viên có @MANV đó đang làm việc  DECLARE @CHINHANH\_LAMVIEC CHAR(5) = (SELECT NHANVIEN.LAMVIEC FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV) | **R(A)** |  |  |
| B3: In danh sách các nhà cần thuê trong 3 tháng trở lại đây  IF(@QUAN IS NULL AND @TPHO IS NOT NULL)  BEGIN  SELECT N.MA\_NHA, N.SLPHONG, N.VIEW\_NHA, N.NVPT, N.MA\_CN, N.MA\_CNHA, LN.TENLN, N.NGAYDANG, N.TRANGTHAI  FROM NHA AS N, LOAINHA AS LN, DCHINHA AS DC  WHERE N.MA\_CN = @CHINHANH\_LAMVIEC AND LN.MA\_LN = N.MALN AND DC.MA\_NHA = N.MA\_NHA AND DC.TPHO = @TPHO  END | **R(B1, B, B2)**  T1 xin khóa đọc trên bảng NHA, LOAINHA, DCHINHA với điều kiện N.MA\_CN = @CHINHANH\_LAMVIEC AND LN.MA\_LN = N.MALN AND DC.MA\_NHA = N.MA\_NHA AND DC.TPHO = @TPHO và giữ khóa cho đến hết giao tác T1. T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong.  Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng NHA, LOAINHA, DCHINHA (kể cả insert thêm dòng mới)  phải đợi cho đến khi T1 thực hiện xong |  |  |
| B4: -- In thông tin nhân viên làm báo cáo  SELECT NHANVIEN.TEN\_NV, @NGAYBAOCAO, NHANVIEN.LAMVIEC  FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV; | **R(A)** |  |  |
| WAITFOR DELAY '00:00:06' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Tạo ra mã nhà mới bằng hàm và kiểm tra thông tin loại nhà insert vào có tồn tại hay không?  IF NOT EXISTS (SELECT \* FROM LOAINHA WHERE LOAINHA.MA\_LN = @LOAINHA)  BEGIN  PRINT N'Loại nhà này không tồn tại'  ROLLBACK TRANSACTION  RETURN;  END | **R(C)**  T2 xin khóa đọc trên bảng  LOAINHA  với điều kiện MA\_LN = @LOAINHA và nhả khóa sau khi kiểm tra xong. |
|  |  | B2: Kiểm tra chi nhánh đăng insert có tồn tại hay không?  IF NOT EXISTS (SELECT \* FROM CHINHANH WHERE CHINHANH.MA\_CN = @CHINHANHDK)  BEGIN  PRINT N'Chi nhánh này không tồn tại'  ROLLBACK TRANSACTION  RETURN;  END | **R(D)**  T2 xin khóa đọc trên bảng  CHINHANH  với điều kiện CHINHANH.MA\_CN = @CHINHANHDK và nhả khóa sau khi kiểm tra xong. |
|  |  | B3: Mặc định lấy mã nhân viên đầu tiên có làm việc tại chi nhánh được truyền vào. Set giá trị cho biến nhân viên phụ trách nhà  DECLARE @NVPHUTRACH CHAR(5) = (SELECT TOP 1 NHANVIEN.MA\_NV FROM NHANVIEN  WHERE NHANVIEN.LAMVIEC = @CHINHANHDK  ORDER BY NHANVIEN.MA\_NV ASC) | **R(E)**  T2 xin khóa đọc trên bảng  NHANVIEN  với điều kiện NHANVIEN.LAMVIEC = @CHINHANHDK và nhả khóa sau khi kiểm tra xong. |
|  |  | B4: Thêm dữ liệu mới vào bảng NHA, NHABAN, DCHINHA  INSERT INTO NHA(MA\_NHA,NGAYDANG,  NGAYHETHAN,  SLPHONG,TINHTRANG,  VIEW\_NHA,NVPT,MA\_CN,  MALN,MA\_CNHA,TRANGTHAI)  INSERT INTO NHABAN VALUES(@MANHAMOI, @GIABAN, @DIEUKIENCHUNHA);  INSERT INTO DCHINHA VALUES(@MANHAMOI, @KHUVUC, @DUONG, @QUAN, @TPHO);  PRINT N'Thêm nhà bán mới thành công' | **W(B1’, F, B2’)**  T2 xin khóa ghi trên các bảng NHA (B1’), NHABAN (F), DCHINHA (B2’)  Cho đến khi commit. Tuy nhiên, T2 phải đợi T1 commit xong mới có thể ghi mới vào các giá trị này |
|  |  | COMMIT |  |
| B5: -- Tiếp tục in danh sách các nhà lại một lần nữa  SELECT N.MA\_NHA, N.SLPHONG, N.VIEW\_NHA, N.NVPT, N.MA\_CN, N.MA\_CNHA, LN.TENLN, N.NGAYDANG, N.TRANGTHAI  FROM NHA AS N, LOAINHA AS LN, DCHINHA AS DC  WHERE N.MA\_CN = @CHINHANH\_LAMVIEC AND LN.MA\_LN = N.MALN AND DC.MA\_NHA = N.MA\_NHA AND DC.TPHO = @TPHO | **R(B1, B, B2)**  **Vì trước đó ta đã set isolation sang mức “Serializable” nên T2 vẫn không thể hành động được kể cả insert thêm một dòng dữ liệu mới. Chính vì vậy danh sách đọc được sẽ vẫn y chang như ban đầu** |  |  |
| B6: -- In tổng số lượng nhà đã cho bán, đã cho thuê  SELECT COUNT(\*) AS SUMNHADABAN\_THUE  FROM NHA, DCHINHA  WHERE NHA.MA\_CN = @CHINHANH\_LAMVIEC AND DCHINHA.MA\_NHA = NHA.MA\_NHA AND DCHINHA.TPHO = @TPHO AND NHA.TRANGTHAI = 1  GROUP BY NHA.TRANGTHAI | **R(B1, B2)**  T1 xin khóa đọc trên bảng NHA, DCHINHA với điều kiện NHA.MA\_CN = @CHINHANH\_LAMVIEC AND DCHINHA.MA\_NHA = NHA.MA\_NHA AND DCHINHA.TPHO = @TPHO AND NHA.TRANGTHAI = 1 và giữ khóa cho đến hết giao tác T1. T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong.  Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng NHA, DCHINHA (thậm chí insert dữ liệu mới) phải đợi cho đến khi T1 thực hiện xong |  |  |
| B7: -- In tổng số lượng nhà chưa cho bán, chưa cho thuê  SELECT COUNT(\*) AS SUMNHACHUABAN\_THUE  FROM NHA, DCHINHA  WHERE NHA.MA\_CN = @CHINHANH\_LAMVIEC AND DCHINHA.MA\_NHA = NHA.MA\_NHA AND DCHINHA.TPHO = @TPHO AND NHA.TRANGTHAI = 0  GROUP BY NHA.TRANGTHAI | **R(B1, B2)**  T1 xin khóa đọc trên bảng NHA, DCHINHA với điều kiện NHA.MA\_CN = @CHINHANH\_LAMVIEC AND DCHINHA.MA\_NHA = NHA.MA\_NHA AND DCHINHA.TPHO = @TPHO AND NHA.TRANGTHAI = 0 và giữ khóa cho đến hết giao tác T1.  **Vì trước đó ta đã set isolation sang mức “Serializable” nên T2 vẫn không thể hành động được kể cả insert thêm một dòng dữ liệu mới. Chính vì vậy danh sách đọc được sẽ vẫn y chang như ban đầu**  . |  |  |
| COMMIT |  |  |  |
|  |  |  |  |

Ta có thể hình dung quá trình sửa lỗi như sau:



Sau khi sửa lỗi thì ta có kết quả như sau. Sau khi set isolation level cho transaction “Thống kê những ngôi nhà ở chi nhánh CI001 tại TPHCM được đăng bán/cho thuê” ở mức cô lập “Serializable” 🡺 Transaction “Thêm một nhà bán mới ở thành phố HCM” sẽ bị block cho tới khi transaction thứ nhất thực hiện xong 🡺 Kết quả hiển thị ra 3 kết quả thay vì 4 🡺 điều đó chứng tỏ T2 không thể can thiệp vào các hành động của T1 khi T1 đang thực hiện



## Tình huống 2:

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Unrepeatable data***  T1 (User = KHACHHANG): Xem thông tin chi tiết nhà bán  T2 (User = CHUNHA): Cập nhật thông tin, chỉnh sửa của một nhà bán | | | |
| Xem thông tin chi tiết nhà bán | **Khóa** | Cập nhật thông tin, chỉnh sửa của một nhà bán | **Khóa** |
| ***Input:*** @MANHA CHAR(5), @MUAHAYKHONG BIT = 0  ***Output***: Thông tin chi tiết của nhà, bình luận về nhà, nếu @MUAHAYKHONG = 1 thì sẽ hiển thị thêm thông tin sơ lược về nhà đăng ký mua.  In ra dòng chữ “N'CHỨC NĂNG NÀY ĐÃ THỰC HIỆN THÀNH CÔNG' nếu chức năng không gặp phải bất cứ lỗi nào | ***Input***: @MANHA CHAR(5) = NULL,@MA\_LN CHAR(5) = NULL, @GIABAN INTEGER = NULL, @SLP INTEGER = NULL, @KHUVUC NVARCHAR(30) = NULL, @DUONG NVARCHAR(30) = NULL, @QUAN NVARCHAR(30) = NULL,@TPHO NVARCHAR(40) = NULL, @NHH DATE = NULL, @TINHTRANG NVARCHAR(100) = NULL, @CHINHANH CHAR(5) = NULL  ***Output***: Nhà bán sau khi đã được cập nhật, chỉnh sửa thông tin |
| SET TRANSACTION ISOLATION LEVEL SERIALIZABLE | SET TRANSACTION ISOLATION  LEVEL READ COMMITTED |
| BEGIN TRAN |  |  |  |
| B1: Khai báo các biến như :  @TENCNHA, @SLPHONG, @LOAINHA, @GIABAN, @TINHTRANG, @DKCHUNHA, @TRANGTHAI, @XEM, @NGAYDANGBAI, @NGAYHETHAN, @DIACHI, @MAKH, @BINHLUAN |  |  |  |
| B2: Set giá trị cho các biến, cụ thể như:  SET @DIACHI=(SELECT DCHINHA.DUONG FROM DCHINHA WHERE DCHINHA.MA\_NHA=@MANHA)+' '+(SELECT DCHINHA.QUAN FROM DCHINHA WHERE DCHINHA.MA\_NHA=@MANHA)+' '+(SELECT DCHINHA.TPHO FROM DCHINHA WHERE DCHINHA.MA\_NHA=@MANHA) | **R(A)**  T1 xin cấp khóa đọc cho bảng DCHINHA với điều kiện MA\_NHA = @MANHA.  T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong. Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng DCHINHA (thậm chí insert dòng dữ liệu mới) phải đợi cho đến khi T1 thực hiện xong |  |  |
| SET @LOAINHA=(SELECT LOAINHA.TENLN FROM LOAINHA,NHA WHERE NHA.MA\_NHA=@MANHA AND LOAINHA.MA\_LN=NHA.MALN) | **R(B)**  T1 xin cấp khóa đọc cho bảng LOAINHA,NHA với điều kiện NHA.MA\_NHA=  @MANHA  =@MANHA AND LOAINHA.MA\_LN  =NHA.MALN.  T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong. Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng NHA,LOAINHA (thậm chí insert dòng dữ liệu mới) phải đợi cho đến khi T1 thực hiện xong |  |  |
| SET @GIABAN=(SELECT NHABAN.GIABAN FROM NHABAN WHERE NHABAN.MA\_NHA=@MANHA) | **R(C)**  T1 xin cấp khóa đọc cho bảng NHABAN với điều kiện NHABAN.MA\_NHA=  @MANHA  T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong. Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng NHABAN (thậm chí insert dòng dữ liệu mới) phải đợi cho đến khi T1 thực hiện xong |  |  |
| …. Tương tự như vậy với quá trình set giá trị cho các biến khác |  |  |  |
| WAITFOR DELAY '00:00:06' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Kiểm tra xem nếu trường nhập vào không có giá trị NULL thì update giá trị.  Chẳng hạn, kiểm tra nếu @MA\_LN không bằng  NULL thì update lại mã loại nhà  IF @MA\_LN IS NOT NULL  BEGIN  UPDATE NHA SET MALN = @MA\_LN  WHERE MA\_NHA = @MANHA  END | **W(B’)**  T2 xin cấp khóa ghi cho bảng NHA với điều kiện MA\_NHA = @MANHA. Tuy nhiên, T2 phải đợi cho đến khi T1 commit (để nhả khóa) thì T2 mới xin được khóa ghi |
|  |  | B2: Kiểm tra xem nếu trường nhập của @GIABAN vào không có giá trị NULL, nếu không NULL sẽ kiểm tra xem giá bán có lẻ hay không nếu thỏa hai trường hợp trên thì update giá trị. Còn nếu không thỏa trường hợp kiểm tra giá bản có lẻ thì ROLLBACK TRANSACTION  IF @GIABAN IS NOT NULL  BEGIN  UPDATE NHABAN SET GIABAN = @GIABAN WHERE  MA\_NHA = @MANHA  IF @GIABAN%100!=0  BEGIN  ROLLBACK TRANSACTION  END  END | **W(C’)**  T2 xin cấp khóa ghi cho bảng NHABAN với điều kiện MA\_NHA = @MANHA.  Tuy nhiên, T2 phải đợi cho đến khi T1 commit (để nhả khóa) thì T2 mới xin được khóa ghi |
|  |  | …. Tương tự như vậy với quá trình kiểm tra và update giá trị cho các biến khác |  |
|  |  | COMMIT |  |
| B3 : Kiểm tra xem nếu @MUAHAYKHONG = 1 thì ta kiểm tra tiếp trạng thái nhà bán lại một lần nữa xem nhà đã bán hay chưa. Nếu chưa thì hiển thị thông tin sơ lược về nhà đăng ký mua  PRINT N'THÔNG TIN NHÀ BẠN ĐĂNG KÝ MUA SƠ LƯỢC GỒM'  SELECT NHA.MA\_NHA, CHUNHA.TEN\_CNHA, NHABAN.GIABAN  FROM NHA, CHUNHA, NHABAN WHERE NHABAN.MA\_NHA = @MANHA AND NHA.MA\_NHA = @MANHA AND CHUNHA.MA\_CNHA = NHA.MA\_CNHA  PRINT N'CHỨC NĂNG NÀY ĐÃ THỰC HIỆN THÀNH CÔNG' | **R(B, C)**  T1 xin cấp khóa đọc cho bảng NHABAN, NHA, CHUNHA với điều kiện  NHABAN.MA\_NHA = @MANHA AND NHA.MA\_NHA = @MANHA AND CHUNHA.MA\_CNHA = NHA.MA\_CNHA. Khóa sẽ được trả sau kh T1 commit |  |  |
| COMMIT | **Ta để ý rằng kết quả ở lần đọc 1 và lần đọc 2 của T1 là như nhau. (Vì T2 không thể can thiệp vào T1 và bị block cho đến khi T1 thực hiện xong)** |  |  |

Ta có thể hình dung quá trình sửa lỗi như sau:

Graphical user interface, text, application

Description automatically generated

Sau khi sửa lỗi thì ta có kết quả như sau. Sau khi set isolation level cho transaction “Xem thông tin chi tiết nhà bán” ở mức cô lập “Serializable” 🡺 Transaction “Cập nhật, chỉnh sửa của một nhà bán” sẽ bị block cho tới khi transaction kia thực hiện xong 🡺 Kết quả hiển thị ra kết quả là 1 record thay vì 2 record (điều đó chứng tỏ T2 không thể can thiệp/chen ngang

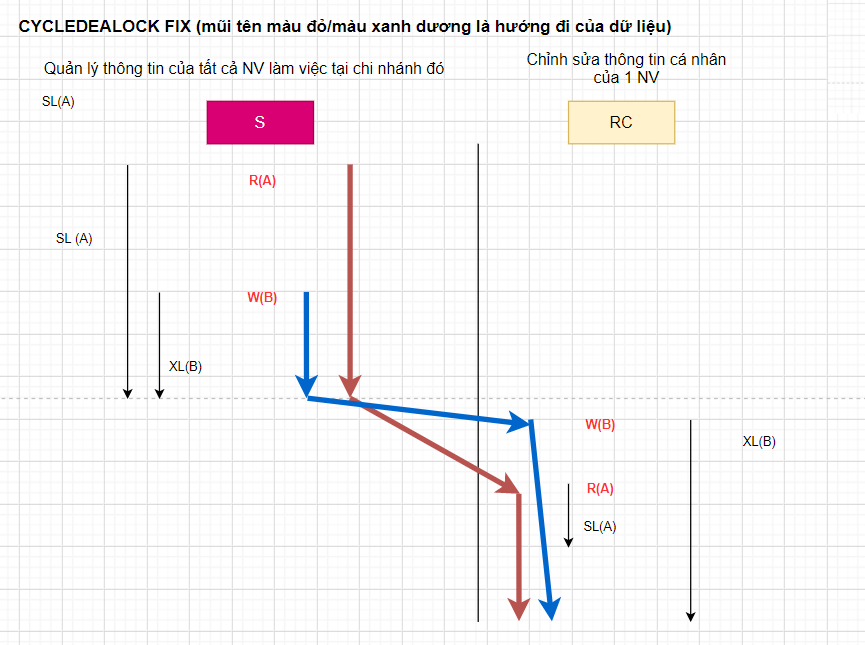
A picture containing chart

Description automatically generated

## Tình huống 3:

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Cycle Deadlock***  T1 (User = NHANVIENQUANLY): Quản lý thông tin của tất cả nhân viên làm việc tại chi nhánh đó  T2 (User = NHANVIEN): Chỉnh sửa thông tin cá nhân | | | |
| Quản lý thông tin của tất cả nhân viên làm việc tại chi nhánh đó | **Khóa** | Chỉnh sửa thông tin cá nhân | **Khóa** |
| ***Input:*** @NVQL CHAR(5),  @MANV CHAR(5) = NULL, /\*Gía trị mặc định là null\*/  @UPDATESALARY BIT = 0,  @LUONG FLOAT = 0  ***Output***: Thông tin của tất cả các nhân viên làm việc tại chi nhánh đó và lương của 1 nhân viên bất kỳ được cập nhật nếu có | ***Input***: @MANV CHAR(5), @TENNV NVARCHAR(50) = NULL, @SDT VARCHAR(15) = NULL, @DCHI NVARCHAR(250) = NULL, @DOB DATE = NULL  ***Output***: Thông tin đã được chỉnh sửa và dòng thông báo chúc mừng thành công hoặc thông báo lỗi nếu có lỗi |
| SET TRANSACTION ISOLATION LEVEL SERIALIZABLE |  |
| BEGIN TRAN |  |  |  |
| B1: Kiểm tra thông tin xem và không cho phép mã nhân viên truyền vào là 1 nhân viên quản lý khác  IF (EXISTS (SELECT \* FROM CHINHANH WHERE CHINHANH.NVQUANLY = @MANV))  BEGIN  PRINT N'Không được xóa/cập nhật  lương của nhân viên quản lý khác';  ROLLBACK TRANSACTION  RETURN;  END | **R(X)**  Xin khóa đọc trên bảng CHINHANH với điều kiện  CHINHANH.NVQUANLY = @MANV và chỉ trả sau khi commit. T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
| B2: Tìm chi nhánh của nhân viên quản lý được truyền vào  DECLARE @CHINHANHNVQL CHAR(5) = (SELECT NHANVIEN.LAMVIEC FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @NVQL) | **R(E)**  Xin khóa đọc trên bảng NHANVIEN với điều kiện  NHANVIEN.MA\_NV = @NVQL và giữ khóa cho tới khi commit . T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
| B3: Hiển thị danh sách các nhân viên do nhân viên quản lý đó quản lý khi **chưa** cập nhật lương  SELECT NHANVIEN.MA\_NV, NHANVIEN.TEN\_NV,  NHANVIEN.LUONG, NHANVIEN.LAMVIEC FROM NHANVIEN WHERE NHANVIEN.LAMVIEC = @CHINHANHNVQL | **R(A)**  Xin khóa đọc trên bảng NHANVIEN với điều kiện  NHANVIEN.LAMVIEC = @CHINHANHNVQL và giữ khóa cho tới khi commit. T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
| B4: Khi nhân viên đó muốn cập nhật lương thì kiểm tra nhân viên đó có tồn tại trong bảng nhân viên hay không?  IF(NOT EXISTS (SELECT \* FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV AND NHANVIEN.LAMVIEC = @CHINHANHNVQL))  BEGIN  PRINT N'Không tồn tại nhân  viên này trong bảng nhân viên'  ROLLBACK TRANSACTION  RETURN;  END | **R(D)**  Xin khóa đọc trên bảng NHANVIEN với điều kiện  NHANVIEN.MA\_NV = @MANV AND NHANVIEN.LAMVIEC = @CHINHANHNVQL và chỉ trả sau khi commit. T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
| WAITFOR DELAY '00:00:07' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Kiểm tra xem nhân viên truyền vào có tồn tại hay không  IF (NOT EXISTS (SELECT \* FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV))  BEGIN  PRINT 'Nhân viên này không tồn tại'  ROLLBACK TRANSACTION  RETURN;  END | **R(B)**  Xin khóa đọc trên bảng NHANVIEN với điều kiện  NHANVIEN.MA\_NV = @MANV và trả sau khi hoàn tất việc đọc dữ liệu. Tuy nhiên, T2 phải đợi cho đến khi T1 thực hiện xong mới được phép xin khóa đọc để đọc |
|  |  | B2: Cập nhật giá trị nếu trường vào không bằng NULL, chẳng hạn như  IF @TENNV IS NOT NULL  UPDATE NHANVIEN  SET NHANVIEN.TEN\_NV = @TENNV  WHERE NHANVIEN.MA\_NV = @MANV | **W(B)**  Xin khóa ghi trên bảng NHANVIEN với điều kiện  NHANVIEN.MA\_NV = @MANV và trả sau khi hoàn tất việc đọc dữ liệu. Tuy nhiên, T2 phải đợi cho đến khi T1 thực hiện xong mới được phép xin khóa ghi để ghi |
|  |  | B3: Tìm chi nhánh quản lý của nhân viên được truyền vào  DECLARE @CHINHANHNVQL CHAR(5) = (SELECT NHANVIEN.LAMVIEC FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV) | **R(B)** |
|  |  | B4: In ra thông tin sau khi chỉnh sửa  SELECT NHANVIEN.MA\_NV, NHANVIEN.TEN\_NV,  NHANVIEN.LUONG, NHANVIEN.LAMVIEC FROM NHANVIEN WHERE NHANVIEN.LAMVIEC = @CHINHANHNVQL AND NHANVIEN.MA\_NV = @MANV | **R(A)**  Xin khóa ghi trên bảng NHANVIEN với điều kiện  NHANVIEN.LAMVIEC = @CHINHANHNVQL AND NHANVIEN.MA\_NV = @MANV và sẽ nhả ra sau khi hoàn tất việc đọc A. Tuy nhiên, T2 phải đợi cho đến khi T1 thực hiện xong mới được phép xin khóa ghi để ghi |
|  |  | COMMIT |  |
| B5: Cập nhật lương cho nhân viên được chỉ định  UPDATE NHANVIEN SET NHANVIEN.LUONG = @LUONG  WHERE NHANVIEN.MA\_NV = @MANV | **W(B)**  Xin khóa GHI trên bảng NHANVIEN với điều kiện  NHANVIEN.MA\_NV = @MANV và giữ cho đến hết giao tác T1. T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
| B6: Hiển thị danh sách các nhân viên do nhân viên quản lý đó quản lý khi **đã** cập nhật lương  SELECT NHANVIEN.MA\_NV, NHANVIEN.TEN\_NV,NHANVIEN.LUONG, NHANVIEN.LAMVIEC FROM NHANVIEN WHERE NHANVIEN.LAMVIEC = @CHINHANHNVQL | **R(A)** |  |  |
| COMMIT |  |  |  |

Ta có thể hình dung quá trình sửa lỗi như sau:



Sau khi sửa lỗi thì ta có kết quả như sau:

Transaction 1 vẫn không có tên của nhân viên có mã là ‘NV013’ được cập nhật mà vẫn chạy đến hết giao tác. Điều đó có thể chứng minh việc T2 đã không thể chen ngang vào các hành động của T1

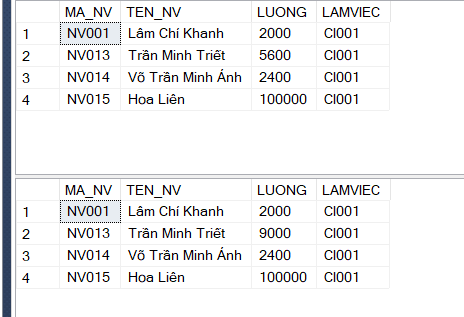
Transaction 2 lúc này đã đợi sau khi T1 xong mới có thể cập nhật lại tên của nhân viên có mã là ‘NV013’, khi này, lương của nhân viên này cũng đã được cập nhật lên thành 9000 (do bên T1 cập nhật), nên ta mới có kết quả như hình

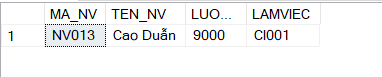
**Về vấn đề giải quyết DEADLOCK:**

* 1. Lý tưởng nhất: thiết kế DB để không bị deadlock. (Deadlock khác lock/blocking)
* 2. Work around/hard coding: Sử dụng các isolation level ở mức thấp hơn, nhưng có thể dẫn đến tình trạng dirty read, ta có thể tìm hiểu concurrency transaction entity framework để xử lý vấn đề này.
* 3. Sử dụng RCSI ( READ COMMITTED SNAPSHOT ISOLATION), có thể giảm deadlock tốt, nhưng sẽ tăng gánh nặng cho cơ sở dữ liệu tempdb. Tác dụng của nó là giảm blocking giữa các transaction mà vẫn đảm bảo tính toàn vẹn dữ liệu. Tuy nhiên cái giá kèm theo là cần thêm bộ nhớ để lưu bản sao của các bản ghi, và phần bộ nhớ này là cần cho mỗi transaction do đó có thể tăng lên rất lớn
* 4. Thay đổi thứ tự query trong các transaction để tránh deadlock (tùy case, không phải lúc nào cũng dùng được)
* 5. Tune queries, thêm index để transaction chạy nhanh nhất có thể, giảm tỷ lệ deadlock
* 6. Cách tồi nhất: deadlock priority. Cách này chỉ thực sự có hiệu quả khi ta có thể khống chế được transaction nào sẽ ưu tiên hơn. Trên thực tế khi có nhiều request vào db thì ta không thể quản lý được cái này.

Tóm lại, việc xử lý deadlock không có giải pháp toàn diện (trừ giải pháp 1). Cần linh hoạt tùy theo tính huống thực tế hệ thống để xử lý. Một transaction bị coi là victim khi nó dễ rollback nhất giữa 2 transaction.

Trong trường hợp này, em đã chọn cách set isolation level ở mức thấp hơn (“Chỉnh sửa thông tin cá nhân” sẽ có isolation level từ Serializable 🡪 Read Committed) (do cách 1 và cách 6 không khả thi, lý do em không áp dụng cách 3 là bởi em đã áp dụng cách 2 thành công, và thay vì phải tốn thêm không gian lưu trữ để lưu sao các bản ghi thì nếu có thể áp dụng cách 2 thành công thì cách 2 vẫn tốt hơn)

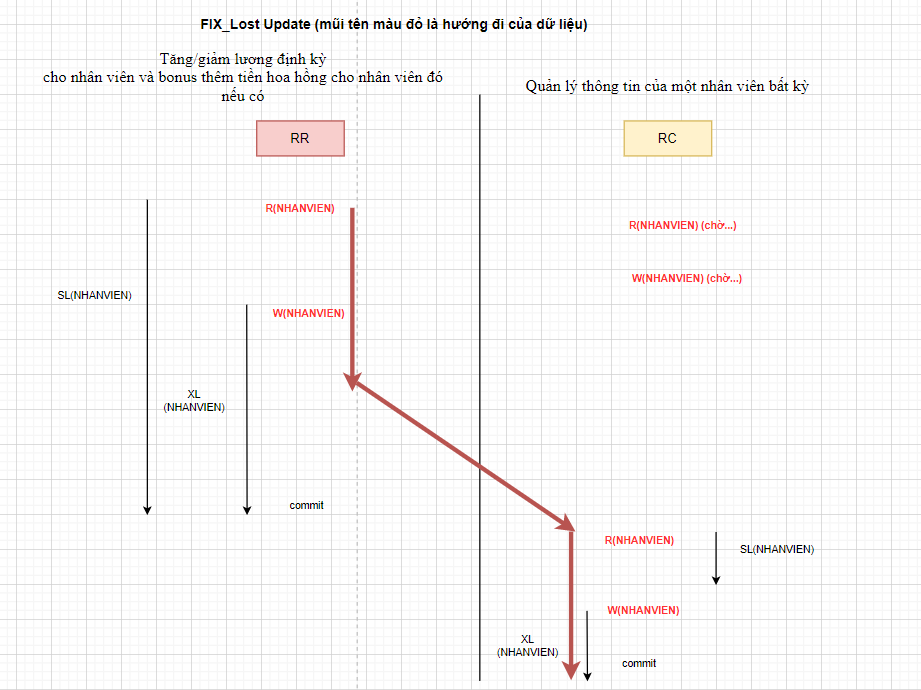




## Tình huống 8:

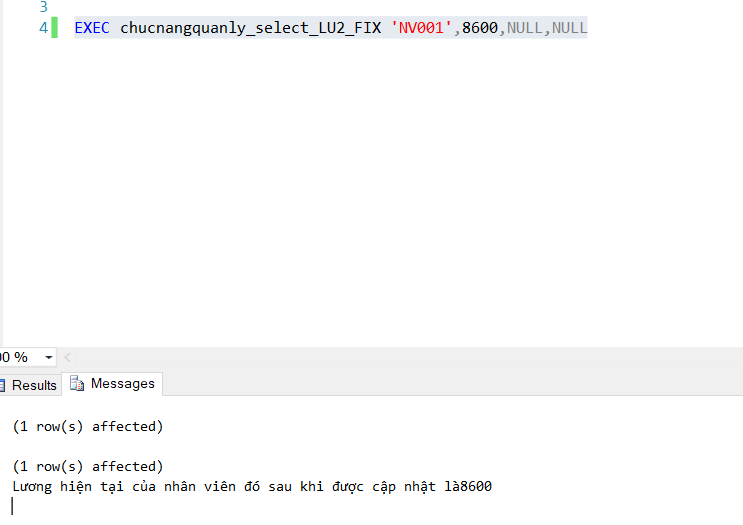
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Lost Update***  T1 (User = NHANVIENQUANLY): Tăng/giảm lương định kỳ cho nhân viên và bonus thêm tiền hoa hồng cho nhân viên đó nếu có  T2 (User = NHANVIEN): Quản lý thông tin của một nhân viên bất kỳ | | | |
| Tăng/giảm lương định kỳ cho nhân viên và bonus thêm tiền hoa hồng cho nhân viên đó nếu có | **Khóa** | Quản lý thông tin của một nhân viên bất kỳ | **Khóa** |
| ***Input:*** @NVQL CHAR(5),  @MANV CHAR(5) = NULL, /\*Gía trị mặc định là null\*/  @UPDATESALARY BIT = 0,  @LUONG FLOAT = 0  ***Output***: Lương nhân viên sau khi được cập nhật | ***Input***: @MANV CHAR(5), @LUONG FLOAT , @SDT VARCHAR(15), @DCHI\_NV NVARCHAR(250)  ***Output***: Thông tin của nhân viên (ở đây là lương) sau khi được cập nhật |
| SET TRANSACTION ISOLATION LEVEL REPEATABLE READ | SET TRANSACTION ISOLATION LEVEL REPEATABLE READ |
| BEGIN TRANSACTION |  |  |  |
| B1: Kiểm tra xem mã NV truyền vào có tồn tại hay không  IF (NOT EXISTS (SELECT \* FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV))  BEGIN  print N'Không tồn tại mã nhân viên này'  rollback transaction  return;  END | **R(NHANVIEN)**  T1 xin khóa đọc trên đơn vị dữ liệu là bảng NHANVIEN với điều kiện MA\_NV = @MANV và trả khóa sau khi T1 commit. T2 muốn xin khóa đọc trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit |  |  |
| B2: Tìm số lượng nhà mà nhân viên đó phụ trách và tiền lương hiện tại của nhân viên đó  DECLARE @SONHAPHUTRACH INT = (SELECT COUNT(MA\_NHA)  FROM NHA  WHERE NVPT = @MANV  GROUP BY NVPT)  DECLARE @LUONGNV FLOAT  SELECT @LUONGNV = LUONG  FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV  DECLARE @SOTIENBITRU FLOAT = 0; | **R(NHA)**  T1 xin khóa đọc trên đơn vị dữ liệu là bảng NHA và trả khóa sau khi đã đọc xong. T2 muốn xin khóa đọc trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit |  |  |
| /\*Nếu số nhà phụ trách lớn hơn 10 ==> thưởng 5000 \*/  /\*Nếu số nhà phụ trách < 10 && >= 5 ==> trừ 2000 \*/  /\*Nếu số nhà phụ trách < 5 && >= 2 ==> trừ 3000 \*/  /\*Nếu số nhà phụ trách < 2 ==> trừ 4500 \*/  WAITFOR DELAY '00:00:07' |  |  |  |
|  |  | BEGIN TRANSACTION |  |
|  |  | B1: In ra thông tin hiện tại của nhân viên có mã được truyền vào  select MA\_NV, TEN\_NV, SDT, DCHI\_NV, NGSINH, LUONG, USER\_NV, LAMVIEC  from NHANVIEN  where MA\_NV = @MANV | **R(NHANVIEN)**  T2 xin khóa đọc trên đơn vị dữ liệu là bảng NHANVIEN với điều kiện MA\_NV = @MANV và trả khóa sau khi đã T2 commit. Tuy nhiên T2 phải đợi T1 commit xong mới được phép đọc |
|  |  | B2: Kiểm tra xem các thông số truyền vào có bằng NULL hay không, nếu không thì cập nhật giá trị.  IF @LUONG IS NOT NULL  BEGIN  UPDATE NHANVIEN SET LUONG = @LUONG WHERE MA\_NV = @MANV | **W(NHANVIEN)**  T2 xin khóa ghi trên đơn vị dữ liệu là LƯƠNG trong bảng NHANVIEN với điều kiện MA\_NV = @MANV và trả khóa sau khi đã T2 commit. Tuy nhiên T2 phải đợi T1 commit xong mới được phép ghi |
|  |  | B3: Kiểm tra xem các thông số truyền vào có bằng NULL hay không, nếu không thì cập nhật giá trị.  DECLARE @LUONGHIENTAI FLOAT  SELECT @LUONGHIENTAI = (SELECT LUONG FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV)  PRINT N'Lương hiện tại của nhân viên đó sau khi được cập nhật là' + CONVERT(VARCHAR, @LUONGHIENTAI)  END | **R(NHANVIEN)** |
| B4: Kiểm tra xem tương ứng với số nhà phụ trách sẽ được tăng/giảm lương như thế nào cũng như cộng thêm tiền hoa hồng nếu có  IF(@SONHAPHUTRACH >= 10)  BEGIN  SET @SOTIENBITRU = 5000  SET @LUONGNV = @LUONGNV + @SOTIENBITRU + @BONUSTIP  ---Cập nhật lương  UPDATE NHANVIEN SET LUONG = @LUONGNV  WHERE NHANVIEN.MA\_NV = @MANV  END  ELSE IF (@SONHAPHUTRACH >=5 AND @SONHAPHUTRACH <10 )  BEGIN  SET @SOTIENBITRU = 2000  SET @LUONGNV = @LUONGNV + @BONUSTIP - @SOTIENBITRU  ---Cập nhật lương  UPDATE NHANVIEN SET LUONG = @LUONGNV  WHERE NHANVIEN.MA\_NV = @MANV  END  ELSE IF (@SONHAPHUTRACH >=2 AND @SONHAPHUTRACH < 5)  BEGIN  SET @SOTIENBITRU = 3000  SET @LUONGNV = @LUONGNV + @BONUSTIP - @SOTIENBITRU  ---Cập nhật lương  UPDATE NHANVIEN SET LUONG = @LUONGNV  WHERE NHANVIEN.MA\_NV = @MANV  END  ELSE IF (@SONHAPHUTRACH < 2)  BEGIN  SET @SOTIENBITRU = 4500  SET @LUONGNV = @LUONGNV + @BONUSTIP - @SOTIENBITRU  ---Cập nhật lương  UPDATE NHANVIEN SET LUONG = @LUONGNV  WHERE NHANVIEN.MA\_NV = @MANV  END  PRINT @LUONGNV | **W(NHANVIEN)**  T1 xin khóa ghi trên đơn vị dữ liệu là bảng NHANVIEN và trả khóa sau khi T1 commit. |  |  |
| commit transaction |  |  |  |
|  |  | B1: Lặp lại bước In ra thông tin hiện tại của nhân viên có mã được truyền vào  select MA\_NV, TEN\_NV, SDT, DCHI\_NV, NGSINH, LUONG, USER\_NV, LAMVIEC  from NHANVIEN  where MA\_NV = @MANV | **R(NHANVIEN)**  T2 xin khóa đọc trên đơn vị dữ liệu là bảng NHANVIEN với điều kiện MA\_NV = @MANV và trả khóa sau khi đã T2 commit. Lúc này do T1 đã commit nên T2 có thể xin khóa đọc |
|  |  | B2: Kiểm tra xem các thông số truyền vào có bằng NULL hay không, nếu không thì cập nhật giá trị.  IF @LUONG IS NOT NULL  BEGIN  UPDATE NHANVIEN SET LUONG = @LUONG WHERE MA\_NV = @MANV | **W(NHANVIEN)**  T2 xin khóa ghi trên đơn vị dữ liệu là LƯƠNG trong bảng NHANVIEN với điều kiện MA\_NV = @MANV và trả khóa sau khi đã T2 commit. Do T1 đã commit nên T2 có thể xin khóa ghi để ghi |
|  |  | B3: Kiểm tra xem các thông số truyền vào có bằng NULL hay không, nếu không thì cập nhật giá trị.  DECLARE @LUONGHIENTAI FLOAT  SELECT @LUONGHIENTAI = (SELECT LUONG FROM NHANVIEN WHERE NHANVIEN.MA\_NV = @MANV)  PRINT N'Lương hiện tại của nhân viên đó sau khi được cập nhật là' + CONVERT(VARCHAR, @LUONGHIENTAI)  END | **R(NHANVIEN)** |
|  |  | commit transaction |  |

Ta có thể hình dung việc sửa lỗi như sau:



Ta có kết quả sau khi sửa lỗi sẽ là:

Vì T1 (tăng/giảm lương cho nhân viên) có set isolation level là RR nên T2 (set isolation là RC) phải đợi cho đến khi T1 commit/rollback mới được phép xin khóa để đọc/ghi trên đơn vị dữ liệu là NHANVIEN. Chính vì lẽ đó mà mặc dù T1 begin trước T2, nhưng kết quả cuối cùng lại là kết quả được cập nhật của T2





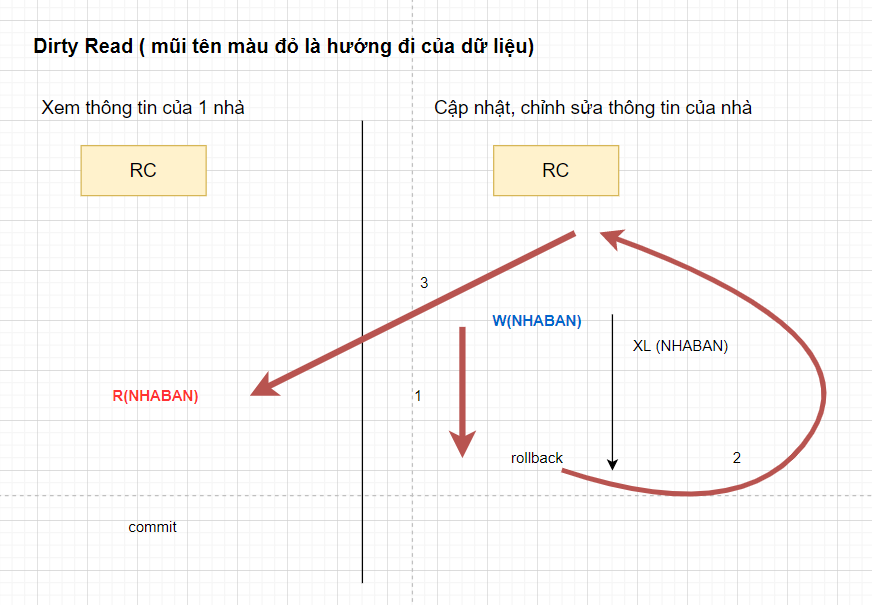
Ban đầu nhóm có ý định sẽ set isolation level transaction cho cả T1 và T2 để khi cả 2 cùng đồng thời chạy sẽ cho random việc chọn Ti làm deadlock victim, thế nhưng cách này thật sự nguy hiểm và không hợp lý. Vì suy cho cùng, chức năng “Quản lý thông tin của 1 nhân viên nhất định” vẫn cần hơn và có nhu cầu dùng nhiều hơn so với chức năng còn lại, nên nếu phải chọn 1 trong 2 giá trị của 1 trong 2 T được cập nhật, nhóm vẫn quyết định dùng giá trị của T2

# Sinh viên thực hiện: Hoàng Nguyên Trúc

## Tình huống 4:

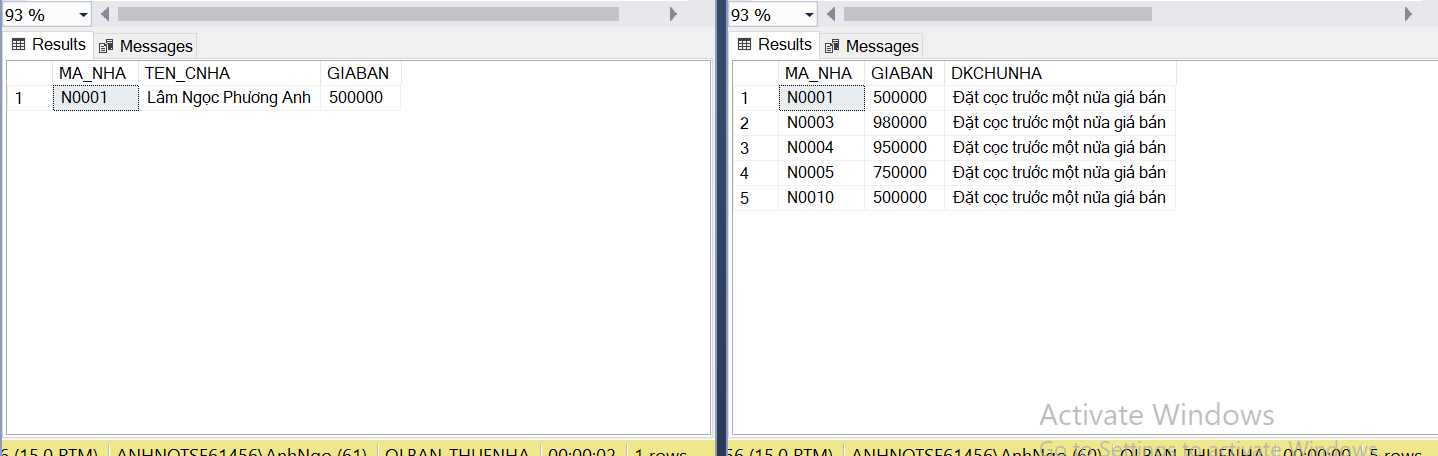
|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Dirty Read***  T1 (User = KHACHHANG): Xem thông tin chi tiết của một nhà  T2 (User = CHUNHA): Cập nhật, chỉnh sửa thông tin của nhà | | | |
| thongtinchitiet\_NB | **Khóa** | chinhsua\_loainha\_ban | **Khóa** |
| ***Input:*** @MANHA CHAR(5), @MUAHAYKHONG BIT = 0  ***Output***: Hiện ra thông tin chi tiết của nhà bán | ***Input***: @MANHA CHAR(5) = NULL,@MA\_LN CHAR(5) = NULL, @GIABAN INTEGER = NULL, @SLP INTEGER = NULL,  @KHUVUC NVARCHAR(30) = NULL, @DUONG NVARCHAR(30) = NULL, @QUAN NVARCHAR(30) = NULL,  @TPHO NVARCHAR(40) = NULL, @NHH DATE = NULL, @TINHTRANG NVARCHAR(100) = NULL, @CHINHANH CHAR(5) = NULL  ***Output***: Thông tin nhà bán được chỉnh sửa |
| SET TRANSACTION ISOLATION  LEVEL **READ COMMITTED** | SET TRANSACTION ISOLATION  LEVEL **READ COMMITTED** |
|  |  | BEGIN TRAN |  |
|  |  | **B1: Kiểm tra thông số truyền vào có NULL hay không, nếu không thì cập nhật giá trị**  **Chẳng hạn ở đây ta kiểm tra @MA\_LN có NULL hay không, nếu không thì cập nhật mã loại nhà**  IF @MA\_LN IS NOT NULL  BEGIN  UPDATE NHA SET MALN = @MA\_LN WHERE MA\_NHA = @MANHA  END | **W(NHABAN)**  Xin khóa ghi trên bảng NHÀ . T2 sẽ giữ khóa ghi này cho đến khi T2 commit |
|  |  | **B2: hẳng hạn ở đây ta kiểm tra @TINHTRANG có NULL hay không, nếu không thì cập nhật mã loại nhà**  IF @TINHTRANG IS NOT NULL  BEGIN  UPDATE NHA SET TINHTRANG = @TINHTRANG WHERE MA\_NHA = @MANHA  END | **W(NHABAN)** |
|  |  | **B3: Chẳng hạn ở đây ta kiểm tra @GIABAN có NULL hay không, nếu không thì cập nhật mã loại nhà. Tuy nhiên, nếu @GIABAN là một số không chia hết cho 100 thì rollback transaction**  IF @GIABAN IS NOT NULL  BEGIN  UPDATE NHABAN SET GIABAN = @GIABAN WHERE MA\_NHA = @MANHA  IF @GIABAN % 100 != 0 | **W(NHABAN)** |
|  |  | WAITFOR DELAY '00:00:10' |  |
| BEGIN TRAN |  |  |  |
| **B1: Khai báo các biến cần thiết**  DECLARE @TENCNHA NVARCHAR(20)  DECLARE @SLPHONG INTEGER  DECLARE @LOAINHA NVARCHAR(20)  DECLARE @GIABAN INTEGER  DECLARE @TINHTRANG NVARCHAR(100)  DECLARE @DKCHUNHA NVARCHAR(50)  DECLARE @TRANGTHAI BIT  DECLARE @XEM INTEGER  DECLARE @NGAYDANGBAI DATE  DECLARE @NGAYHETHAN DATE  DECLARE @DIACHI NVARCHAR(54) |  |  |  |
| **B2: Set giá trị cho các biến cần thiết**  SET @DIACHI=(SELECT DCHINHA.DUONG FROM DCHINHA WHERE DCHINHA.MA\_NHA=@MANHA)+' '+(SELECT DCHINHA.QUAN FROM DCHINHA WHERE DCHINHA.MA\_NHA=@MANHA)+' '+(SELECT DCHINHA.TPHO FROM DCHINHA WHERE DCHINHA.MA\_NHA=@MANHA)  SET @TENCNHA=(SELECT CHUNHA.TEN\_CNHA FROM CHUNHA,NHA WHERE NHA.MA\_NHA=@MANHA AND CHUNHA.MA\_CNHA=NHA.MA\_CNHA)  SET @SLPHONG=(SELECT NHA.SLPHONG FROM NHA WHERE NHA.MA\_NHA=@MANHA)  SET @LOAINHA=(SELECT LOAINHA.TENLN FROM LOAINHA,NHA WHERE NHA.MA\_NHA=@MANHA AND LOAINHA.MA\_LN=NHA.MALN)  SET @GIABAN=(SELECT NHABAN.GIABAN FROM NHABAN WHERE NHABAN.MA\_NHA=@MANHA)  SET @TINHTRANG=(SELECT NHA.TINHTRANG FROM NHA WHERE NHA.MA\_NHA=@MANHA)  SET @DKCHUNHA=(SELECT NHABAN.DKCHUNHA FROM NHABAN WHERE NHABAN.MA\_NHA=@MANHA)  SET @TRANGTHAI=(SELECT NHA.TRANGTHAI FROM NHA WHERE NHA.MA\_NHA=@MANHA)  SET @XEM=(SELECT NHA.VIEW\_NHA FROM NHA WHERE NHA.MA\_NHA=@MANHA)  SET @NGAYDANGBAI=(SELECT NHA.NGAYDANG FROM NHA WHERE NHA.MA\_NHA=@MANHA)  SET @NGAYHETHAN=(SELECT NHA.NGAYHETHAN FROM NHA WHERE NHA.MA\_NHA=@MANHA) | **R(NHA)**  **R(LOAINHA)**  **R(DCHINHA)**  **R(NHABAN)**  T1 xin khóa đọc trên các bảng NHA, LOAINHA, DCHINHA và nhả ra sau khi đọc xong. Riêng T1 muốn xin khóa đọc để đọc trên bảng NHABAN thì phải đợi T2 ghi xong mới được cấp khóa đọc trên bảng NHABAN |  |  |
| **B3: In ra màn hình các thông tin cần thiết**  PRINT(N'Xem :'+CAST(@XEM AS CHAR(3)))  PRINT(N'Ngày Đăng Bài: '+cast ( @NGAYDANGBAI as char(20)))  PRINT(N'Ngày Hết Hạn: '+ cast( @NGAYHETHAN as char(20)))  PRINT(N'Địa chỉ: '+@DIACHI)  PRINT(N'Chủ nhà: '+@TENCNHA)  PRINT(N'Số lượng phòng ở: '+ CAST( @SLPHONG as varchar(10)))  PRINT(N'Loại nhà:'+@LOAINHA)  PRINT(N'Giá bán: '+CAST(@GIABAN AS VARCHAR(10)))  PRINT(N'Tình trạng: '+@TINHTRANG)  PRINT(N'Điều kiện chủ nhà: '+@DKCHUNHA) | ***Lúc này T1 sẽ không đọc dữ liệu trước khi bị rollback của T2 nữa*** |  |  |
| **B4 : Kiểm tra nếu nhà đã mua rồi thì báo lỗi, ngược lại in thông tin chi tiết nhà sẽ mua**  IF (@MUAHAYKHONG = 1)  BEGIN  DECLARE @TRANGTHAINHA BIT = (SELECT NHA.TRANGTHAI FROM NHA WHERE NHA.MA\_NHA = @MANHA)  IF (@TRANGTHAINHA = 1)  BEGIN  PRINT N' NHÀ ĐÃ ĐƯỢC BÁN RỒI. KHÔNG THỂ MUA NỮA';  ROLLBACK TRANSACTION  RETURN;  END  ELSE BEGIN  PRINT N'THÔNG TIN NHÀ BẠN ĐĂNG KÝ MUA SƠ LƯỢC GỒM'  SELECT NHA.MA\_NHA, CHUNHA.TEN\_CNHA, NHABAN.GIABAN  FROM NHA, CHUNHA, NHABAN WHERE NHABAN.MA\_NHA = @MANHA AND NHA.MA\_NHA = @MANHA AND CHUNHA.MA\_CNHA = NHA.MA\_CNHA  END  END |  |  |  |
| COMMIT |  | rollback transaction  return; | **//Khi update dữ liệu GIABAN cho NHA, nhưng do GIABAN không thỏa điều kiện if nên điều này dẫn đến toàn dữ liệu sẽ bị ROLLBACK** |

Ta có thể hình dung quá trình sửa lỗi như sau:



Sau khi sửa lỗi thì ta có kết quả như sau:

Ban đầu T2 chạy trước T1 để cập nhật thông tin của nhà (ở đây là giá bán), sau khi cập nhật, T2 nhận thấy giá bán được truyền vào không chia hết cho 100 nên rollback transaction và hủy những gì đã thay đổi trước đó. Vì T1 set isolation level là RC nên nó phải đợi T2 commit/rollback xong mới có thể cấp khóa đọc trên bảng NHABAN để đọc giá bán của nhà 🡺 Transaction 1 vẫn hiển thị thông tin giá bán là 500.000 khi chạy đến hết giao tác



## Tình huống 5:

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: Conversion Deadlock**  T1 (User = NHANVIEN): Chỉnh sửa tên của nhân viên (chức năng nhỏ).  T2 (User = QUANLY): Quản lý thông tin của một nhân viên bất kỳ | | | |
| chucnangquanly\_select | **Khóa** | Chucnangquanly\_select\_nv | **Khóa** |
| ***Input:*** @MANV CHAR(5), @LUONG FLOAT , @SDT VARCHAR(15), @DCHI\_NV NVARCHAR(250)  ***Output***: Thông tin của nhân viên | ***Input***: @MANV CHAR(5) , @SDT VARCHAR(15), @DCHI\_NV NVARCHAR(250)  ***Output***: Thông tin của nhân viên |
| SET TRANSACTION ISOLATION  LEVEL **SERIALIZABLE** | SET TRANSACTION ISOLATION  LEVEL **REPEATEABLE READ COMMITTED** |
| BEGIN TRAN |  |  |  |
| B1: Xem thông tin của nhân viên  select MA\_NV, TEN\_NV, SDT, DCHI\_NV, NGSINH, LUONG, USER\_NV, LAMVIEC  from NHANVIEN  where MA\_NV = @MANV | **R(NHANVIEN)**  Xin share lock để đọc trên bảng NHANVIEN với điều kiện NHANVIEN.MA\_NV= @MANV và giữ khóa cho tới khi commit . T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Xem thông tin của nhân viên  select MA\_NV, TEN\_NV, SDT, DCHI\_NV, NGSINH, LUONG, USER\_NV, LAMVIEC  from NHANVIEN  where MA\_NV = @MANV | **R(NHANVIEN)**  Xin shared lock  trên bảng NHANVIEN với điều kiện NHANVIEN.MA\_NV= @MANV và trả sau khi hoàn tất việc đọc dữ liệu. Tuy nhiên, T2 phải đợi cho đến khi T1 thực hiện xong mới được phép xin khóa đọc để đọc |
| WAITFOR DELAY '0:0:05' |  |  |  |
| B2: Cập nhật tên nhân viên  IF @SDT IS NOT NULL  BEGIN try  UPDATE NHANVIEN SET LUONG = @LUONG WHERE MA\_NV = @MANV  END try | **W(NHANVIEN)**  Xin khóa GHI trên bảng NHANVIEN với điều kiện NHANVIEN.MA\_NV= @MANV và giữ khóa cho tới khi commit . T2 muốn thực hiện các hành động trên đơn vị dữ liệu này phải đợi cho đến khi T1 commit (kể cả việc insert dữ liệu mới) |  |  |
|  |  |  |  |
|  |  | WAITFOR DELAY '0:0:05' |  |
|  |  | B2: Cập nhật tên nhân viên  begin try  UPDATE NHANVIEN SET SDT= @SDT WHERE MA\_NV = @MANV  end try | **W(NHANVIEN)**  Xin khóa ghi trên bảng NHANVIEN với điều kiện NHANVIEN.MA\_NV= @MANV và trả sau khi hoàn commit xong. Tuy nhiên, T2 phải đợi cho đến khi T1 thực hiện xong mới được phép xin khóa ghi để ghi |
|  |  | COMMIT |  |

Ta có thể hình dung quá trình sửa lỗi như sau:

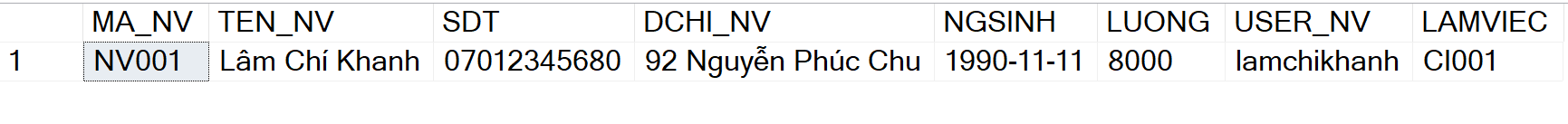
Diagram

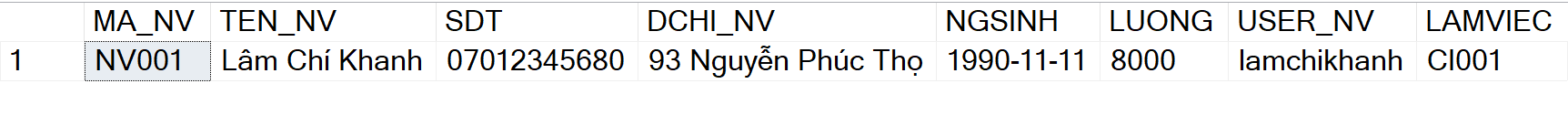
Description automatically generated

Sau khi sửa lỗi thì ta có kết quả như sau.

Transaction 1 địa chỉ vẫn là 92 Nguyễn Phúc Chu. Điều đó có thể chứng minh việc T2 đã không thể chen ngang vào các hành động của T1.

Transaction 2 lúc này đã đợi sau khi T1 xong mới có thể cập nhật lại địa chỉ của nhân viên có mã là ‘NV001’, khi này, lương của nhân viên này cũng đã được cập nhật lên thành 8000 (do bên T1 cập nhật), nên ta mới có kết quả như hình.





## Tình huống 6:

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Lost Update***  T1 (User = NHANVIEN): Chỉnh sửa thông tin cá nhân  T2 (User = CHUNHA): Lập hợp đồng thuê/bán nhà có đăng ký tại chi nhánh nơi nhân viên đó làm việc cho khách thuê/mua | | | |
| capnhat\_thongtin\_chunha | **Khóa** | laphopdong\_b2 | **Khóa** |
| ***Input:*** @machunha char(5), @ten NVARCHAR(50), @dchi NVARCHAR(250), @SDT VARCHAR(15), @PASS\_CNHA VARCHAR(50)  ***Output***: Thông tin của chủ nhà được cập nhật | ***Input***: @manha char(5), @manv char(5)  ***Output***: Thông tin nhân viên lập hợp đồng, mã hợp đồng, ngày kết thúc và bắt đầu của hợp đồng; thông tin chủ nhà; thông tin khách hàng |
|  | SET TRANSACTION ISOLATION  LEVEL **REPEATABLE READ** |
| BEGIN TRAN |  |  |  |
| WAITFOR DELAY '0:0:04' |  |  |  |
| B1: Cập nhật tên hoặc địa chỉ hoặc số điện thoại nếu giá trị truyền vào tương ứng khác NULL.  IF @ten is not null  BEGIN  UPDATE CHUNHA SET TEN\_CNHA = @ten WHERE MA\_CNHA = @machunha  END  IF @dchi is not null  BEGIN  UPDATE CHUNHA SET DCHI\_CNHA= @dchi WHERE MA\_CNHA = @machunha  END  IF @SDT IS NOT NULL  BEGIN  UPDATE CHUNHA SET SDT = @SDT WHERE MA\_CNHA = @machunha  END  IF @PASS\_CNHA IS NOT NULL  BEGIN  UPDATE CHUNHA SET PASS\_CNHA = @PASS\_CNHA WHERE MA\_CNHA = @machunha  END | **W(CHUNHA)**  Xin khoá ghi trên bảng CHUNHA với điều kiện MA\_CNHA = @machunha. T1 sẽ giữ khóa ghi này cho đến khi commit |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Khai báo và gán giá trị cho ngày bắt đầu và kết thúc của hợp đồng  declare @ngaybd date  declare @ngaykt date  set @ngaybd = getdate()  set @ngaykt = DATEADD(day,2,@ngaybd) |  |
|  |  | B2: Chọn thông tin cá nhân của chủ nhà và khách hàng  select TEN\_CNHA, DCHI\_CNHA, SDT  FROM CHUNHA  WHERE MA\_CNHA = 'CN001'  select TEN\_KH, DCHI\_KH,SDT  FROM KHACHHANG  WHERE MA\_KH = 'KH001' | **R(CHUNHA, KHACHHANG)**  Xin khoá đọc trên bảng CHUNHA, KHACHHANG với điều kiện MA\_CNHA = ‘CN001’, MA\_KH = ‘KH001’ và trả khóa sau khi T2 commit |
| COMMIT |  |  |  |
|  |  | WAITFOR DELAY '0:0:03' |  |
|  |  | B3: Chọn thông tin nhân viên, mã hợp đồng, ngày bắt đầu và kết thúc của hợp đồng  select TEN\_NV, MA\_HD, NGAYBD\_HD, NGAYKT\_HD  FROM NHANVIEN, HOPDONG  WHERE LAMVIEC = (select MA\_CN from NHA where MA\_NHA = @manha)  AND HOPDONG.MA\_NHA = @manha  AND NHANVIEN.MA\_NV = @manv | **R(NHANVIEN, HOPDONG, NHA)**  Xin khoá đọc trên bảng NHANVIEN, HOPDONG, NHA với điều kiện MA\_NHA = @manha, HOPDONG.MA\_NHA = @manha, NHANVIEN.MA\_NV = @manv. Trả khóa sau khi T2 commit |
|  |  | COMMIT |  |

Ta có thể hình dung quá trình sửa lỗi như sau:

Diagram

Description automatically generated

Sau khi sửa lỗi thì ta có kết quả như sau. Nhóm quyết định dùng cách 2 là hạ isolation level của T xuống thay vì dùng các cách khác

Transaction 2 đọc đúng giá trị số điện thoại được cập nhật bởi Transaction 1 thay vì đọc giá trị số điện thoại cũ trước khi được Transaction 1 cập nhật.

Text, Word

Description automatically generated

A picture containing scatter chart

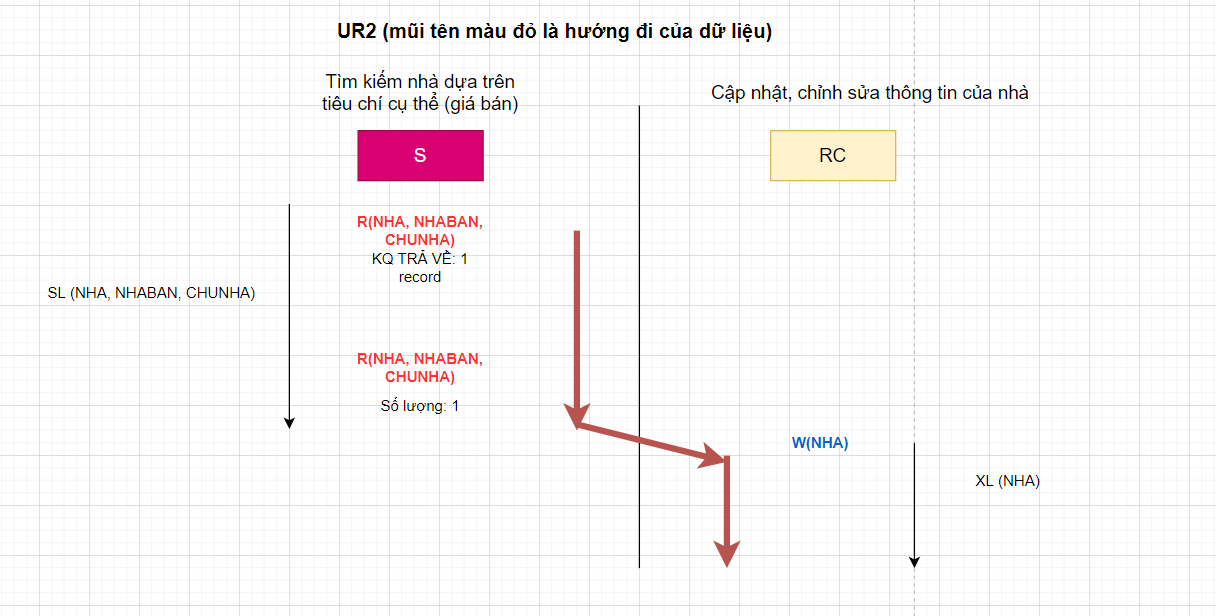
Description automatically generated

# Sinh viên thực hiện: Ngô Thị Thanh Thảo

## Tình huống 7:

|  |  |  |  |
| --- | --- | --- | --- |
| **ERR01: *Unrepeatable Data***  T1 (User = KHACHHANG): Tìm nhà dựa trên tiêu chí cụ thể  T2 (User = CHUNHA): Cập nhật, chỉnh sửa thông tin của nhà | | | |
| Tìm nhà dựa trên tiêu chí cụ thể | **Khóa** | Cập nhật, chỉnh sửa thông tin của nhà | **Khóa** |
| ***Input: @GIABAN INTEGER***  ***Output***: Thông tin nhà được tìm kiếm | ***Input***: @MANHA CHAR(5) = NULL,@MA\_LN CHAR(5) = NULL, @GIABAN INTEGER = NULL, @SLP INTEGER = NULL,  @KHUVUC NVARCHAR(30) = NULL, @DUONG NVARCHAR(30) = NULL, @QUAN NVARCHAR(30) = NULL,  @TPHO NVARCHAR(40) = NULL, @NHH DATE = NULL, @TINHTRANG NVARCHAR(100) = NULL, @CHINHANH CHAR(5) = NULL  ***Output***: Thông tin nhà bán được cập nhật |
| SET TRANSACTION ISOLATION LEVEL SERIALIZABLE |  |
| BEGIN TRAN |  |  |  |
| B1: Đọc thông tin của nhà thỏa điều kiện  SELECT NHA.MA\_NHA,CHUNHA.TEN\_CNHA AS CHUNHA, NHA.SLPHONG, NHA.VIEW\_NHA, NHABAN.GIABAN, NHA.NGAYDANG, NHA.NGAYHETHAN  FROM NHA, NHABAN, CHUNHA  WHERE NHA.MA\_NHA= NHABAN.MA\_NHA AND NHABAN.GIABAN <= @GIABAN AND NHA.MA\_CNHA = CHUNHA.MA\_CNHA | **R(NHA, NHABAN, CHUNHA)**  Xin khoá đọc trên bảng NHA, NHABAN, CHUNHA với điều kiện NHA.MA\_NHA = NHABAN.MA\_NHA, NHABAN.GIABAN <= @GIABAN AND NHA.MA\_CNHA = CHUNHA.MA\_CNHA. T2 muốn cập nhật trên đơn vị dữ liệu này phải đợi T1 thực hiện xong. Mọi hành động liên quan đến update, delete hay insert có liên quan đến bảng NHA, NHABAN, CHUNHA (thậm chí insert dòng dữ liệu mới) phải đợi cho đến khi T1 thực hiện xong |  |  |
| WAITFOR DELAY '00:00:06' |  |  |  |
|  |  | BEGIN TRAN |  |
|  |  | B1: Cập nhật thông tin của nhà bán  IF @MA\_LN IS NOT NULL  BEGIN  UPDATE NHA SET MALN = @MA\_LN WHERE MA\_NHA = @MANHA  END  IF @TINHTRANG IS NOT NULL  BEGIN  UPDATE NHA SET TINHTRANG = @TINHTRANG WHERE MA\_NHA = @MANHA  END  IF @GIABAN IS NOT NULL  BEGIN  UPDATE NHABAN SET GIABAN = @GIABAN WHERE MA\_NHA = @MANHA  IF @GIABAN % 100 != 0  begin  rollback transaction  return;  end  END  IF @KHUVUC IS NOT NULL  BEGIN  UPDATE DCHINHA SET KHUVUC = @KHUVUC WHERE MA\_NHA = @MANHA  END  IF @DUONG IS NOT NULL  BEGIN  UPDATE DCHINHA SET DUONG = @DUONG WHERE MA\_NHA = @MANHA  END  IF @QUAN IS NOT NULL  BEGIN  UPDATE DCHINHA SET QUAN = @QUAN WHERE MA\_NHA = @MANHA  END  IF @TPHO IS NOT NULL  BEGIN  UPDATE DCHINHA SET TPHO = @TPHO WHERE MA\_NHA = @MANHA  END  IF @NHH IS NOT NULL  BEGIN  UPDATE NHA SET NGAYHETHAN = @NHH WHERE MA\_NHA = @MANHA  END  IF @SLP IS NOT NULL  BEGIN  UPDATE NHA SET SLPHONG = @SLP WHERE MA\_NHA = @MANHA  END  IF @CHINHANH IS NOT NULL  BEGIN  UPDATE NHA SET MA\_CN = @CHINHANH WHERE MA\_NHA = @MANHA  END | **W(NHA)**  Xin khóa ghi trên bảng NHA với điều kiện NHA.MA\_NHA = @MANHA.    Tuy nhiên, T2 phải đợi cho đến khi T1 commit (để nhả khóa) thì T2 mới xin được khóa ghi |
|  |  | COMMIT |  |
| B2: In số lượng nhà thỏa điều kiện  PRINT N'Số lượng nhà thỏa điều kiện trên là'  SELECT COUNT(\*) AS SOLUONGNHATHOADK  FROM NHA,NHABAN,CHUNHA  WHERE NHA.MA\_NHA= NHABAN.MA\_NHA AND NHABAN.GIABAN <=@GIABAN AND NHA.MA\_CNHA=CHUNHA.MA\_CNHA | **R(NHA, NHABAN, CHUNHA)** |  |  |
| COMMIT | **Ta để ý rằng kết quả ở lần đọc 1 và lần đọc 2 của T1 là như nhau. (Vì T2 không thể can thiệp vào T1 và bị block cho đến khi T1 thực hiện xong** |  |  |
|  |  |  |  |

Ta có thể hình dung quá trình sửa lỗi như sau:



Sau khi sửa lỗi thì ta có kết quả như sau. Sau khi set isolation level cho transaction “tìm nhà dựa trên tiêu chí cụ thể” ở mức cô lập “Serializable” 🡺 Transaction “Cập nhật, chỉnh sửa thông tin của nhà” sẽ bị block cho tới khi transaction kia thực hiện xong 🡺 Kết quả hiển thị ra kết quả là 1 record thay vì 2 record (điều đó chứng tỏ T2 không thể can thiệp/chen ngang vào hành động của T1 mà phải đợi cho T1 commit xong để nhả các khóa trên đơn vị dữ liệu nó đã từng khóa ra thì T2 mới có thể hành động được)

