**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---o0o---



**BÁO CÁO ĐỒ ÁN: NỘI DUNG TUẦN 6, 7**

MÔN HỌC: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

**Nhóm sinh viên: Nguyễn Quang Phú – 19127507**

**Lâm Hoàng Phúc – 19127512**

**Võ Đình Phúc - 19127515**

**Lớp: 19HTTT1**

**TP.HỒ CHÍ MINH – THÁNG 11/2021**

**Mục lục**

**[1.](#_Toc88507525)****[Thông tin nhóm](#_Toc88507525)** [3](#_Toc88507525)

**[2.](#_Toc88507526)****[Phân công công việc](#_Toc88507526)** [3](#_Toc88507526)

**[3.](#_Toc88507527)****[Nội dung báo cáo](#_Toc88507527)** [4](#_Toc88507527)

*[3.1](#_Toc88507528)**[Mô tả chức năng](#_Toc88507528)* [4](#_Toc88507528)

*[3.2](#_Toc88507529)**[Mô tả các thủ tục](#_Toc88507529)* [5](#_Toc88507529)

*[3.3](#_Toc88507530)**[Mô tả lỗi tranh chấp, deadlock](#_Toc88507530)* [9](#_Toc88507530)

*[3.4 Thiết kế giao diện](#_Toc88507531)***[Error! Bookmark not defined.](#_Toc88507531)**

# **Thông tin nhóm**

|  |  |  |
| --- | --- | --- |
| **Mã số sinh viên** | **Họ và tên** | **Email** |
| **19127507** | Nguyễn Quang Phú | 19127507@student.hcmus.edu.vn |
| **19127512** | Lâm Hoàng Phúc | 19127512@student.hcmus.edu.vn |
| **19127515** | Võ Đình Phúc | 19127515@student.hcmus.edu.vn |

# **Phân công công việc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Công việc** | **Chi tiết công việc** | **Phân công** | **Hoàn thành** |
| 1 | Viết mô tả chức năng của các phân hệ, phân chia công việc cho các thành viên | | 19127515 | 100% |
| **1** | Viết các thủ tục trong hệ thống | Phân hệ nhân viên và quản trị viên | 19127515 | 100% |
| Phân hệ khách hàng và tài xế | 19127512 | 100% |
| Phân hệ đối tác | 19127507 | 100% |
| **2** | Tìm lỗi tranh chấp, deadlock, viết script, hình ảnh minh họa | Mỗi thành viên phải tìm được từ 3 lỗi trở lên, lưu script, hình ảnh minh họa vào thư mục cá nhân. | 19127512 | 100% |
| 19127507 | 100% |
| 19127515 | 100% |
| Tổng hợp lỗi, hoàn thành script (cuộc họp qua Zoom) | Nhóm | 100% |
| **3** | Thiết kế giao diện | Thực hiện trong buổi họp qua Zoom (cùng ngày với buổi họp tổng hợp lỗi) | Nhóm | 0% |
| **4** | Hoàn thành báo cáo | | 19127507  (50%) | 100% |
| 19127515  (50%) | 100% |
| **5** | Nộp bài | | 19127515 | 100% |

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ và tên** | **Mức độ đóng góp** |
| **19127515** | Võ Đình Phúc | 34% |
| **19127507** | Nguyễn Quang Phú | 33% |
| **19127512** | Lâm Hoàng Phúc | 33% |

# **Nội dung báo cáo**

* 1. *Mô tả chức năng*

|  |  |
| --- | --- |
| **Phân hệ** | **Các chức năng** |
| **Đối tác** | Đăng kí làm đối tác, đăng kí hợp đồng, đăng kí chi nhánh cho hợp đồng, đối tác thêm – xóa - sửa thông tin sản phẩm và chi nhánh có cung cấp sản phẩm này, đối tác xem thông tin đơn hàng và cập nhật tình trạng của đơn hàng. |
| **Khách hàng** | Đăng kí khách hàng, xem danh sách đối tác, xem danh sách sản phẩm của đối tác. Khách hàng chọn sản phẩm, số lượng tương ứng, hình thức thanh toán và địa chỉ giao hàng. Hệ thống tiến hành tính phí gồm: phí sản phẩm và phí vận chuyển. Khách hàng xác nhận đơn hàng. Khách hàng được phép xem tình trạng đơn hàng. |
| **Tài xế** | Tài có thể đăng ký tài làm tài xế của hệ thống. Sau khi đăng kí tài xế thì tài xế có thể xem danh sách đơn hàng theo khu vực mình đã chọn, nhận một đơn hàng, cập nhật trạng thái đơn hàng và xem thông tin của những đơn hàng mình đã nhận. |
| **Nhân viên** |  |
| **Quản trị** |  |

* 1. *Mô tả các thủ tục*

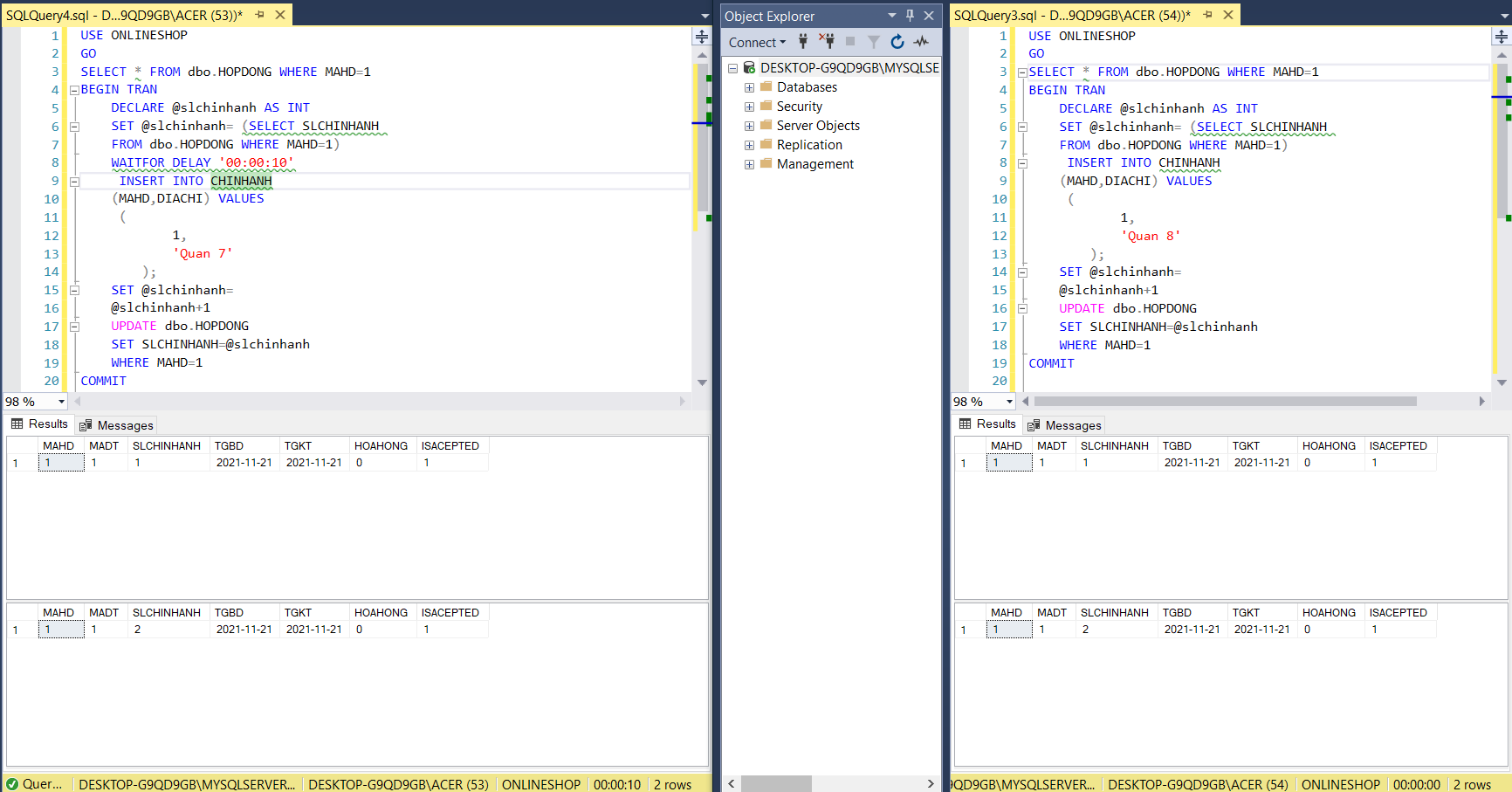
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thủ tục** | **Tham số** | **Mô tả** |
| **Phân hệ đối tác** | | | |
| **1** | XEM\_DS\_DONHANG\_DT | @MADT | Xem danh sách đơn hàng tồn tại của một đối tác. |
| **2** | XEM\_TT\_DONHANG\_DT | @MADT | Xem danh sách đơn hàng và chi tiết của đơn hàng của một đối tác. |
| **3** | XEM\_DS\_CHINHANH | @MASP,  @MADT | Xem danh sách chi nhánh cung cấp sản phẩm đã chọn. |
| **4** | DANGKI\_DT | @USERNAME\_DT  @PASS\_DT  @SDT\_DT  @EMAIL\_DT                    @DIACHI\_DT                   @MALH  @MAKV  @TENDT  @DAIDIEN | Hệ thống sẽ tạo một tài khoản cho đối tác sau đó, insert một đối tác với thông tin do người dùng nhập vào bảng đối tác. |
| **5** | DANGKI\_HOPDONG | @MADT  @TGBD  @TGKT  @HOAHONG | Tạo một hợp đồng cho đối tác với các thông tin do người dùng nhập vào, với trạng thái kích hoạt mặc định là 0. |
| **6** | DANGKI\_CHINHANH\_HOPDONG | @MAHD  @DIACHI | Thêm một chi nhánh vào hợp đồng với thông tin do người dùng nhập vào, sau đó thực hiện tăng số lượng chi nhánh của đối tác trong hợp đồng. |
| **7** | GIAHAN\_HOPDONG | @MAHD  @TGKT  @HOAHONG                  @ISACEPTED | Gia hạn hợp đồng với thông tin do người dùng nhập vào |
| **8** | THEM\_SANPHAM | @MALH  @TENSP | Thêm sản phẩm với thông tin do đối tác nhập vào |
| **9** | SUA\_SANPHAM | @MASP  @MALH  @TENSP | Sửa sản phẩm với thông tin do đối tác nhập vào |
| **10** | XOA\_SANPHAM | @MASP | Xóa sản phẩm đã chọn đồng thời xóa các QLSANPHAM của sản phẩm đó |
| **11** | THEM\_CHINHANH\_SP | @MADT  @MASP  @MACN  @GIASP  @SLCUNGCAP | Tạo QLSANPHAM để thêm các thông tin về chi nhánh và đối tác cung cấp theo thông tin do người dùng nhập |
| **12** | SUA\_CHINHANH\_SP | @MADT  @MASP  @MACN  @GIASP  @SLCUNGCAP | Sửa QLSANPHAM theo thông tin mà người dùng đã nhập |
| **13** | XOA\_CHINHANH\_SP | @MADT  @MASP | Xóa QLSANPHAM của đối tác đã chọn với một sản phẩm đã chọn |
| **14** | CAPNHAT\_DONHANG\_DT | @MADH  @TRANGTHAISHIP         @TRANGTHAITTOAN | Cập nhật trạng thái của đơn hàng theo người dùng nhập |
| **Phân hệ khách hàng** | | | |
| **1** | DANGKI\_KH | @USERNAME\_KH  @PASS\_KH                     @SDT\_KH                      @EMAIL\_KH                    @DIACHI\_KH | Đăng ký tài khoản cho khách hàng trong bảng tài khoản bằng proc TAO\_TAIKHOAN, sau đó hệ thống thực hiện thêm một khách hàng vào bảng khách hàng. |
| **2** | XEM\_DS\_DT |  | Cho phép khách hàng xem danh sách đối tác của hệ thống. |
| **3** | XEM\_DS\_SP | @MADT | Sau khi khách hàng đã chọn được đối tác, khách hàng có thể xem danh sách sản phẩm được cung cấp bởi đối tác này. |
| **4** | TAO\_DONHANG | @MAKH  @MADT  @MATX  @HINHTHUCTT  @NGAYTAO  @DIACHIGH  @PHISP  @PHISHIP  @TONGTIEN  @TRANGTHAISHIP  @TRANGTHAITTOAN  @MADH OUTPUT | Chức năng tạo một đơn hàng, khi khách hàng tiến hành chọn sản phẩm và những thông tin cần thiết và hệ thống sẽ tạo một đơn hàng cho khách hàng. |
| **5** | THEMSANPHAM | @MADH  @MASP  @GIASP  @SLSP  @MADT | Đối với mỗi loại sản phẩm được khách hàng thêm vào đơn hàng của mình thì hệ thống sẽ tạo ra một quản lí đơn hàng để quản lí sản phẩm này trong đơn hàng của khách hàng ( Giả sử một khách hàng được phép đặt nhiều sản phẩm của một đối tác). Đồng thời trừ số sản phẩm đặt vào số lượng cung cấp sản phẩm này của đối tác. |
| **6** | DATHANG | @MAKH  @MASP  @MADT  @SLSP  @HINHTHUCTT @DIACHIGH  @PHISHIP | Khách hàng sau khi chọn được sản phẩm và số lượng tương ứng cũng như một số thông tin cần thiết thì hệ thống thực hiện tạo một đơn hàng cho khách hàng.  Kiểm tra nếu số lượng sản phẩm cung cấp nhỏ hơn số lượng sản phẩm đặt thì cho phép khách hàng thêm sản phẩm.  Thêm sản phẩm này vào đơn hàng cho khách hàng. |
| **7** | XACNHAN\_DONHANG | @MADH | Khi  khách hàng xác nhận đơn hàng  thì hệ thống tiến hành set lại trạng thái cho đơn hàng là đã thanh toán. |
| **8** | XEM\_TTHAI\_DONHANG | @MADH | Khách hàng được phép xem trạng thái ship của đơn hàn sau khi xác nhận đơn hàng. |
| **Phân hệ tài xế** | | | |
| **1** | DANGKI\_TAIXE | @USERNAME\_TX  @PASS\_TX  @SDT\_TX  @EMAIL\_TX  @DIACHI\_TX  @MAKV  @CMND  @DIACHI  @BIENSO  @STK  @NGANHANG | Đầu tiên hệ thống tiến hành tạo một tài khoản cho tài xế, sau đó thêm một đối tượng tài xế vào bảng tài xế với những thông tin được nhập vào. |
| **2** | HIENTHI\_DSDH | @MAKV | Cho phép tài xế xem danh sách đơn hàng của những đối tác  trong khu vực mình đã đăng kí. Và những đơn hàng này phải thỏa điều kiện là đã được thanh toán và mã tài xế là null. |
| **3** | NHAN\_DONHANG | @MADH  @MATX | Khi tài xế nhận đơn hàng thì hệ thống sẽ gắn mã tài xế này vào đơn hàng. |
| **4** | CAPNHAT\_DONHANG\_TX | @MADH  @TRANGTHAISHIP | Tài xế được phép cập nhật lại trạng thái ship của đơn hàng, khi giao hàng. |
| **5** | XEM\_DS\_DONHANG\_DANHAN | @MATX | Tài xế được phép xem thông tin của những đơn hàng mình đã nhận. |
| **Phân hệ nhân viên và quản trị viên** | | | |
| **1** | XEM\_DSHD\_DADUYET | @MADT | Xem danh sách những hợp đồng của đối tác đã được xét duyệt |
| **2** | XEM\_DSHD\_CHUADUYET | @MADT | Xem danh sách những hợp đồng đối tác đã đăng kí nhưng chưa thực hiện xét duyệt |
| **3** | THONGBAO\_GIAHAN | @MADT | Thực hiện tìm ra danh sách những hợp đồng sắp đáo hạn của đối tác thông qua mã đối tác. Thời gian thông mặc định là 10 ngày trước khi đáo hạn. |
| **4** | DUYET\_HOPDONG | @MADT | Nhân viên thực hiện tìm ra danh sách những hợp đồng đã lập của đối tác những chưa thông qua xét duyệt, sau đó duyệt những hợp đồng này, và thông báo thời hạn đến đối tác. |
| **5** | TAO\_TAIKHOAN | @ISSTAFF  @ISSUPERUSER  @USERNAME  @PASS  @SDT  @EMAIL  @DIACHI  @ISACTIVE   @MATK | Tạo một tài khoản dựa vào những thông tin người dùng nhập vào. |
| **6** | CAPNHAT\_TAIKHOAN | @MATK  @USERNAME  @PASS  @ISSTAFF  @ISSUPPERUSER  @SDT  @EMAIL  @DIACHI | Cập nhật tài khoản người dùng dựa trên username của người dùng, và những thông tin và người dùng muốn thay đổi. |
| **7** | XEM\_DS\_TK |  | Nhân viên xem những tài khoản hiện tại có trong hệ thống. |
| **8** | XOA\_TK\_NV\_AD | @USERNAME | Nhân viên thực hiện xóa một tài khoản của admin hoặc nhân viên khác, thông qua username. |
| **9** | KHOA\_TK | @USERNAME | Nhân viên thực hiện khóa một tài khoản bất kì, thông qua username |
| **10** | KICHHOAT\_TK | @USERNAME | Nhân viên thực hiện kích hoạt một tài khoản bất kì trong hệ thống thông qua username. |
| **11** | CAPQUYEN\_DULIEU | @USERNAME | Người quản trị thực hiện cấp quyền cho một tài khoản bất kì trong hệ thống thông qua user |

* 1. *Mô tả lỗi tranh chấp, deadlock*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case** | **Loại lỗi** | **Kịch bản** | **T1** | **T2** |
| **Testcase1** | Lost update | Xét giao tác Đăng kí chi nhánh cho hợp đồng.  T1: Đọc số lượng chi nhánh sau đó cập nhật số lượng chi nhánh.  T1: Đọc số lượng chi nhánh sau đó cập nhật số lượng chi nhánh. Giả sử hai giao tác này cùng thực hiện trên 1 hợp đồng. | DECLARE @slchinhanh AS INT  SET @slchinhanh = (SELECT SLCHINHANH  FROM dbo.HOPDONG WHERE MAHD=1) |  |
|  | DECLARE @slchinhanh AS INT  SET @slchinhanh= (SELECT SLCHINHANH  FROM dbo.HOPDONG WHERE MAHD=1) |
| INSERT INTO CHINHANH  (MAHD,DIACHI) VALUES(1,'Tp hcm'); |  |
|  | INSERT INTO CHINHANH  (MAHD,DIACHI) VALUES (1,'quan 8'); |
| SET @slchinhanh =  @slchinhanh+1  UPDATE dbo.HOPDONG  SET SLCHINHANH = @slchinhanh  WHERE MAHD=1 |  |
| COMMIT |  |
|  | SET @slchinhanh=  @slchinhanh+1  UPDATE dbo.HOPDONG  SET SLCHINHANH=@slchinhanh  WHERE MAHD=1 |
|  | COMMIT |
| **Giải thích:** Sau khi T1 đọc được số lượng chi nhánh trên bảng hợp đồng thì cùng lúc nào T2 cũng đi vào hệ thống và đọc số lượng chi nhánh tương tự,(vì cả 2 đều phát Slock nên không bị ngăn cản),sau đó cả hai thực hiện insert chi nhánh vào bảng chi nhánh và update lại số lượng sản phẩm trên bảng hợp đồng, dẫn đến số lượng chi nhánh bị ghi đè, chỉ tăng lên 1 trong khi có hai chi nhánh được thêm vào. | | | | |
| **Giải quyết:** Để giải quyết vấn đề lost update trên ta có thể set mức cô lập cho giao tác là repeatable read, tuy nhiên nếu set cho giao tác mức cô lập này rất có thể sẽ gây ra deadlock do, T1 và T2 sẽ chờ đợi tài nguyên của nhau. Thay vào đó, vì hai giao tác này thực hiện đọc trước và ghi sau, nên ở thao tác đọc ban đầu để lấy số lượng hiện tại của chi nhánh chúng ta có thể cấp khóa Xlock trên bảng chi nhánh cho giao tác đọc luôn thay vì là Slock như mặc định của SQL Server, điều này giúp cho giao tác chạy trước ngăn giao tác chạy sau đọc số lượng chi nhánh khi nó chưa chạy xong. | | | | |
| **Testcase2** | Unrepeatable read | Nhân viên thực hiện xem danh sách hợp đồng của đối tác sau đó gửi những thông báo đến đối tác đối với những hợp đồng sắp đến hạn, tuy nhiên trong lúc nhân viên đang thực hiện xem thì khách hàng thực hiện gia hạn một số hợp đồng, dẫn đến danh sách thông báo gia hạn sau đó bị sai so với ban đầu dự định của nhân viên. | XEM\_DSHD\_DADUYET (ĐỐI TÁC A) |  |
|  | GIAHAN\_HD (ĐỐI TÁC A) |
|  | COMMIT |
| THONGBAO\_GIAHAN (ĐÔI TÁC A) |  |
| COMMIT |  |
| **Giải thích:** Nhân viên thứ nhất thực hiện xem danh sách hợp đồng chưa duyệt của đối tác A, tuy nhiên sau đó đối tác A thực hiện gia hạn hợp đồng (vì T1 sinh SL nhưng nhả ra ngay sau đó nên T2 có thể ghi được trên bảng hợp đồng) sau đó T2 commit T1 tiếp tục thực hiện thông báo gia hạn đến đối tác A thì phát hiện danh sách thông báo đã sai. | | | | |
| **Giải quyết:** Để giải quyết vấn đề Unrepeatable read trên t chỉ cần set mức cô lập cho thao tác T1 là repeatable read, giúp cho T1 giữ khóa Slock trên bảng hợp đồng cho đến khi giao tác kết thúc, và giao tác T2 sẽ không thể ghi cho đến khi T1 commit. | | | | |
| **Testcase3** | Dirty read | Nhân viên thứ nhất thực hiện xem danh sách hợp đồng chưa duyệt của đối tác, sau đó thực hiện duyệt một số hợp đồng, trong khi đang thực hiện duyệt thì nhân viên thứ hai đi vào xem danh sách hợp đồng đã duyệt của đối tác để thông báo hạn đến cho khách hàng. Tuy nhiên sau đó nhân viên thứ nhất gặp lỗi phải rollback, dẫn đến danh sách thông báo bị sai. | XEM\_DSHD  \_CHUADUYET  (ĐỐI TÁC A)  DUYET\_HD  (ĐỐI TÁC A) |  |
|  | XEM\_DSHD\_DADUYET (ĐỐI TÁC A) |
| ROLLBACK |  |
|  | COMMIT |
| **Giải thích:** T1 thực hiện đọc trên bảng hợp đồng, sau đó thực hiện ghi trên bảng hợp đồng, tuy nhiên T2 sau đó đọc trên bảng hợp đồng, và T1 rollback ngay sau đó, dẫn đến dữ liệu T2 đọc bị sai. | | | | |
| **Giải quyết:** Để giải quyết vấn đề Dirty read trên, ta thực hiện set mức cô lập cho giao tác T2 là read commited để đảm bảo rằng T2 chỉ đọc khi T1 thực hiện xong. | | | | |
| **Testcase4** | Phantom read | Nhân viên thứ nhất thực hiện xem danh sách hợp đồng chưa duyệt của đối tác A  Đối tác A thực hiện đăng ký hợp đồng mới.  Nhân viên thứ nhất duyệt hợp đồng cho đối tác A, và thông báo thời hạn đối với những hợp đồng mới duyệt.  => Sau khi duyệt thì nhân viên phát hiện ra một hợp đồng khác của đối tác, thay vì những hợp đồng như danh sách ban đầu. | XEM\_DSHD\_CHUADUYET (ĐT A) |  |
| DUYET\_HOPDONG (ĐT A) |  |
|  | DANGKI\_HOPDONG (ĐỐI TÁC A) |
|  | COMMIT |
| XEM\_DSHD\_CHUADUYET (ĐT A) |  |
| COMMIT |  |
| **Giải thích:** T1 đọc và ghi trên bảng hợp đồng => T1 sinh XL trên bảng hợp đồng, những vì T2 thực hiện Insert trên bảng hợp đồng nên không bị chặn bởi XL, sau đó T1 đọc lại bảng hợp đồng thì dữ liệu bị dư ra do T2 insert. | | | | |
| **Giải quyết:** Để giải quyết Phantom read trên chúng ta thực hiện set mức cô lập cho giao tác T1 là Serializable, vì T2 là thao tác insert nên mặc dùng T1 đang giữ khóa thì T2 vẫn có thể ghi lên bản hợp đồng, vì vậy để tránh trường hợp này thì chúng ta set mức cô lập cho T1 là Serializable để ngăn T2 insert chen ngang. | | | | |
| **Testcase5** | Lost update | Hai khách cùng đặt sản phẩm của một đối tác với số lượng sản phẩm là 10, và số lượng sản phẩm cung cấp còn lại của đối tác là 15. Cả hai khách hàng kiểm tra số lượng sản phẩm cùng lúc, nên đều pass qua được điều kiện của hệ thống. Sau đó cả 2 cùng đặt và đặt được hàng. Tuy nhiên, số lượng sản phẩm còn lại của đối tác bị giảm thành -5. | TAO\_DONHANG @MADH, @MADT, @MATX, @HINHTHUCTT, @ NGAYTAO, @DIACHI, @PHISP, @PHISHIP,,@TONGTIEN, @ TRANGTHAISHIP, @MADH. |  |
|  | TAO\_DONHANG @MADH, @MADT, @MATX, @HINHTHUCTT, @ NGAYTAO, @DIACHI, @PHISP, @PHISHIP,@TONGTIEN, @ TRANGTHAISHIP, @MADH. |
| DECLARE @SLSPCC AS INT  SELECT @SLSPCC= SLCUNGCAP FROM dbo.QLSANPHAM WHERE MADT=@MADT AND MASP=@MASP |  |
|  | DECLARE @SLSPCC AS INT  SELECT @SLSPCC= SLCUNGCAP FROM dbo.QLSANPHAM WHERE MADT=@MADT AND MASP=@MASP |
| IF (@SLSP>@SLSPCC)  ROLLBACK |  |
|  | IF (@SLSP>@SLSPC)  ROLLBACK |
|  | THEMSANPHAM @MADH, @MASP, @GIASP, @SLSP, @MADT |
|  | COMMIT |
| THEMSANPHAM @MADH, @MASP, @GIASP, @SLSP, @MADT |  |
| COMMIT |  |
| **Giải thích:** T1 thực hiện đọc trên bảng QLSANPHAM và thấy số lượng sản phẩm hợp lệ, sau đó T2 cũng đọc trên bảng hợp đồng và cũng thấy dữ liệu hợp lệ (vì cả hai đều sinh ra SL và nhả ra ngay sau đọc nên không bị xung đột). Sau đó cả hai lần lượt Update số lượng sản phẩm lên bảng QLSANPHAM thì dẫn đến SLSP bị âm vì tổng số lượng nhỏ hơn tổng số được đặt. | | | | |
| **Giải quyết:** Đối với trường hợp lost update trên để giải quyết vấn đề trên thì chúng ta có thể set mức cô lập cho hai bảng Serializable, tuy nhiên việc set mức cô lập này rất có thể sẽ gây ra conversion deadlock bởi vì, T1 thực hiện đọc và giữ khóa đọc sau đó ghi trên bảng QLSanPham và T2 cũng làm điều tương tự. Vì vậy để giải quyết trường hợp này và tránh deadlock thì chúng ta sử dụng có chế khóa UPDLOCK khi thực hiện đọc trên bảng QLSanPham ở cả hai giao tác để đảm bảo khi T1 vào trước và thực hiện UPDLOCK nếu T2 đi vào và cũng thực hiện việc tương tự thì UPDLOCK sẽ ngăn cản T2 vì 2 UPDLOCK không tương thích với nhau. | | | | |
| **Testcase6** | Deadlock | Khách hàng thực hiện đặt hàng và xác nhận đơn hàng, sau đó khách hàng khách hàng tiếp tục xem danh sách sản phẩm của đối tác, tuy nhiên ngay sau khi khách hàng xác nhận đơn hàng thì đối tác thực hiện sửa một số sản phẩm, sau đó đối tác xem trạng thái của đơn hàng mà khách hàng mới xác nhận | XACNHAN\_DONHANG 1 |  |
|  | SUA\_SANPHAM 1 1 ‘GA’ |
| XEM\_DS\_SP 1 |  |
|  | XEM\_TTHAI\_DONHANG\_DT 1 |
| COMMIT |  |
|  | COMMIT |
| **Giải thích:** Deadlock vì sau khi khách hàng thực hiên xác nhận đơn hàng thì T1 đã XL trên bảng DONHANG, sau đó đối tác thực hiện sửa sản phẩm thì T2 đã XL trên bảng sản phẩm. Sau đó T1 yêu cầu đọc bảng SANPHAM phải đợi, tương tự T2 yêu cầu đọc bảng DONHANG cũng phải đợi, dẫn đến deadlock | | | | |
| **Testcase7** | Deadlock | Đối tác T1 thực hiện sửa thông tin sản phẩm, sau đó thực hiện chỉnh sửa chi nhánh của sản phẩm này, nhưng sau khi T1 sửa thông tin sản phẩm thì đối tác T2 thực hiện sửa thông tin chi nhánh của 1 sản phẩm, sau đó mới sửa thông tin của sản phẩm. | SUA\_SANPHAM 1, 1, ‘THITGA’ |  |
|  | SUA\_CHINHANH\_SP 2, 2, 1, 120, 100 |
| SUA\_CHINHANH\_SP 1, 1, 1, 100, 100 |  |
|  | SUA\_SANPHAM 2, 2, ‘MAYTINH’ |
| COMMIT |  |
|  | COMMIT |
| **Giải thích:** Đầu tiên đối tác T1 thực hiện sửa sản phẩm nên đã giữ khóa Xlock trên bảng SANPHAM sau đó T2 thực hiện sửa chi nhanh sản phẩm nên giữ khóa XLock trên bảng QLSanPham, sau đó T1 lại thực hiện sửa chi nhánh sản phẩm, nên bắt buộc phát sinh XLcok trên bảng QLSanPham những T2 lại đang giữ khóa trên bảng này, nên T1 phải đợi, sau đó T2 thực hiện sửa sản phẩm nên yêu cầu XLock trên bảng SanPham tuy nhiên T1 lại đang XLock trên bảng này nên T2 phải đợi T1, từ đấy dẫn đến hai giao tác này đợi lẫn nhau. | | | | |
| **Testcase8** | Lost update | Hai tài xế cùng nhận cùng thực hiện nhận một đơn hàng, tài xế thứ nhất thực hiện kiểm tra đơn hàng đấy thấy rằng đơn hàng chưa được tài xế nào nhận, sau đó tài xế 2 vào và cũng kiểm tra và cũng hợp lệ, sau đó hai tài xế cùng nhận đơn hàng này, dẫn đến sẽ có một tài xế bị ghi đè kết quả | IF NOT EXISTS (SELECT \* FROM DONHANG WHERE MATX IS NULL AND MADH=1)  ROLLBACK |  |
|  | IF NOT EXISTS (SELECT \* FROM DONHANG WHERE MATX IS NULL AND MADH=1)  ROLLBACK |
| UPDATE dbo.DONHANG SET MATX=1 WHERE MADH=1 |  |
| COMMIT |  |
|  | UPDATE dbo.DONHANG SET MATX=2WHERE MADH=1 |
|  | COMMIT |
| **Giải thích:** T1 thực hiện hiện đọc trên bảng DONHANG được cấp khóa SLock tuy nhiên sau đó nhả khóa ngay vì mức cô lập lúc này chỉ là Read Commited, sau đó T2 cũng thực tương tự như T1, sau đó T1 thực hiện Update trên bảng DONHANG và T2 cũng làm điều tương tự dẫn đến lost update. | | | | |
| **Giải quyết:** chúng ta set isolation level repeatable read cho giao tác này để khi T1 đọc thì sẽ giữ khóa đến cuối giao tác, tránh T2 vào hệ thống sau đó. | | | | |
| **Testcase9** | Dirty Read | Khách hàng T1 tạo một đơn hàng cho đối tác đó, sau đó T2 xem danh sách đơn hàng của đối tác, nhưng T1 xảy ra lỗi (số lượng sản phẩm đặt lớn hơn số lượng sản phẩm còn lại) nên rollback sau khi tạo đơn hàng. | DATHANG |  |
|  | XEM\_DS\_DONHANG\_DT |
| ROLLBACK |  |
|  | COMMIT |
| **Giải thích:** T1 thực hiện tạo đơn hàng (thao tác insert), lúc này T2 thực hiện đọc trên bảng đơn hàng (vì T1 insert nên không phát sinh khóa -> T2 được cấp SL), sau đó T1 kiểm tra số lượng sản phẩm đặt thì thấy lớn hơn số lượng sản phẩm còn lại, nên rollback dẫn đến T2 đọc dữ liệu bẩn. | | | | |
| **Giải quyết:** Để giải quyết vấn đề trên chúng ta chỉ cần set mức cô lập cho giao tác T2 là Read Commited để đảm bảo rằng T2 chỉ đọc khi T1 đã thực hiện xong. | | | | |
| **Testcase10** | Phantom Read | T1: nhân viên thực hiện xem danh sách tài khoản, sau đó thực hiện cập nhật thông tin tài khoản của khách hàng A, sau đó T2 khách hàng B đăng kí tài khoản,  sau đó T1 nhân viên kiểm tra lại danh sách tài khoản, nhận thấy trong danh sách xuất hiện tài khoản B. | XEM\_DS\_TK |  |
| CAPNHAT\_TAIKHOAN 1 |  |
|  | DANGKI\_KH |
| XEM\_DS\_TK |  |
|  | COMMIT |
| COMMIT |  |
| **Giải thích:** T1 đọc và update trên bảng TAIKHOAN -> XLOCK trên TAIKHOAN. T2 thực hiện insert một tài khoản vào bảng TAIKHOAN và KHACHANG (thao tác insert không bị ngăn cản bởi XLOCK) dẫn đến sau đó T2 đọc dữ liệu lại thì thấy trong hệ thống xuất hiện một tài khoản nữa. | | | | |
| **Giải quyết:** vì T2 thực hiện insert nên để giải quyết vấn đề này thì chúng ta set mức cô lập cho T1 là Serializable để đảm bảo khi T1 chưa thực hiện xong thì T2 không được insert. | | | | |
| **Testcase11** | Unrepeatable Read | T1: Nhân viên thực hiên xem danh sách đã duyệt của đối tác 1.  T2: nhân viên thực hiện duyệt hợp đồng của đối tác 1.  T1: Thực hiện thông báo gia hạn hợp đồng của đối tác 1  => Nhưng danh sách hợp đồng sắp đáo hạn của đối tác 1 có thay đổi, vì nhân viên T2 đã thực hiện duyệt thêm hợp đồng cho đối tác 1 | XEM\_DSHD\_DADUYET (1) |  |
|  | DUYET\_HOPDONG (1) |
|  | COMMIT |
| THONGBAO\_GIAHAN( 1) |  |
| COMMIT |  |
| **Giải thích:** T1 một đọc trên bảng HOPDONG, sau đó T2 update trên bảng HOPDONG ( phát sinh XL nhưng không bị ngăn cản vì SL của T1 trên HOPDONG đã được unlock sau khi T1 đọc xong), sau đó T2 rollback và T1 tiếp tục đọc trên bảng HOPDONG thì dữ liệu có thay đổi. | | | | |
| **Giải quyết:** để giải quyết vấn đề unrepeatable read trên thì chúng ta thực hiện set mức cô lập cho T1 ở mức repeatable read, để đảm bảo khi T1 đọc và nhận được khóa Slock thì sẽ giữ khóa Slock đến cuối giao tác, từ đó T2 không thể thực hiện Update trên bảng hợp đồng khi T1 chưa hoàn thành. | | | | |

## *Kết quả thực thi lỗi*

***Testcase1:***



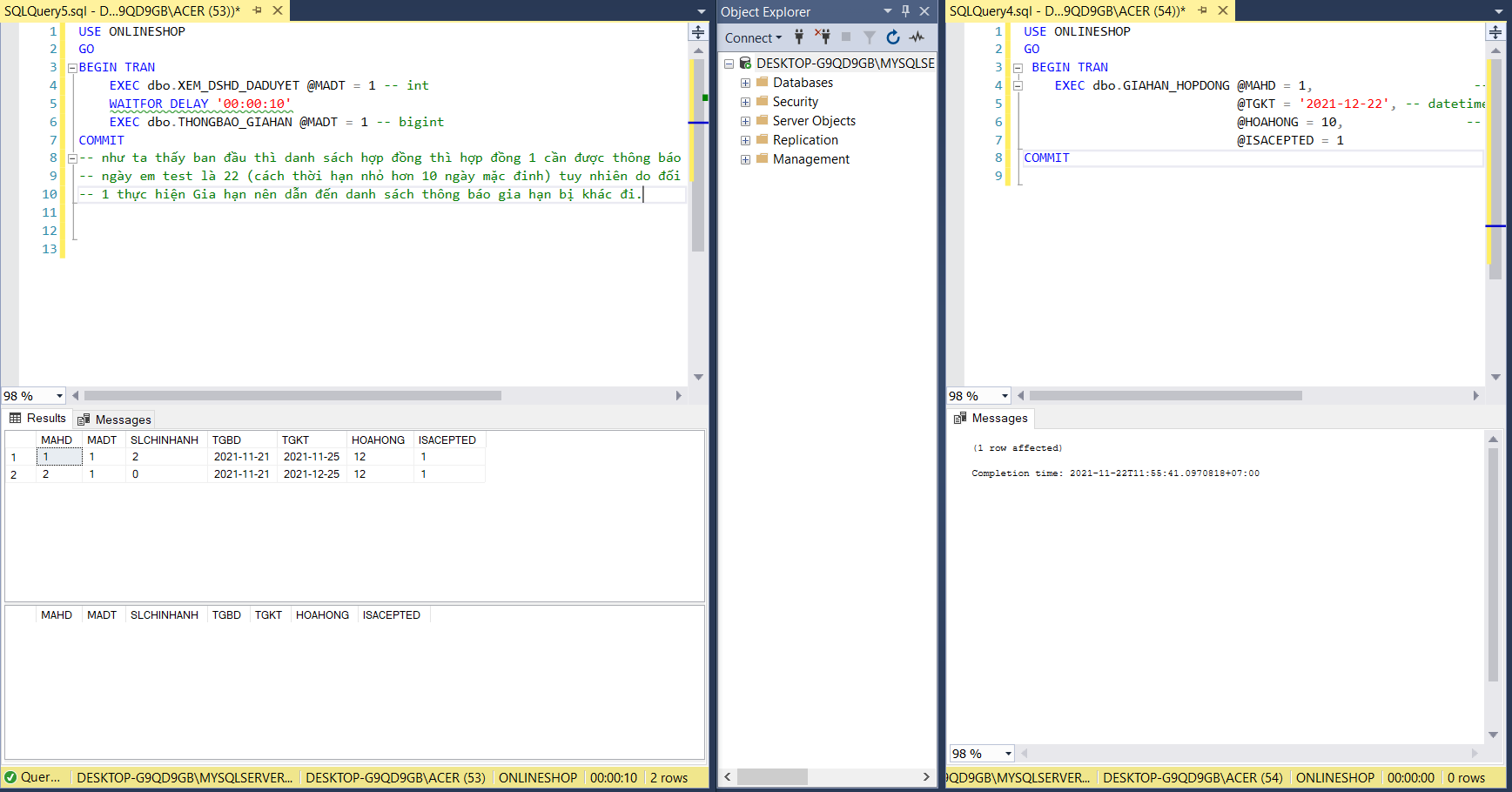
Ta thấy rằng ban đầu số lượng chi nhánh của hợp đồng 1 là 1:



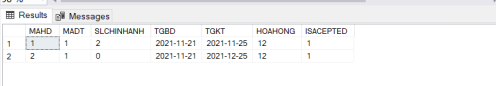
Tuy nhiên, sau khi cả hai giao tác cũng thực hiện đăng kí chi nhánh cho hợp đồng này thì số chi nhánh chỉ tăng lên 1, vì vậy đã rơi vào lost update.



***Testcase2:***



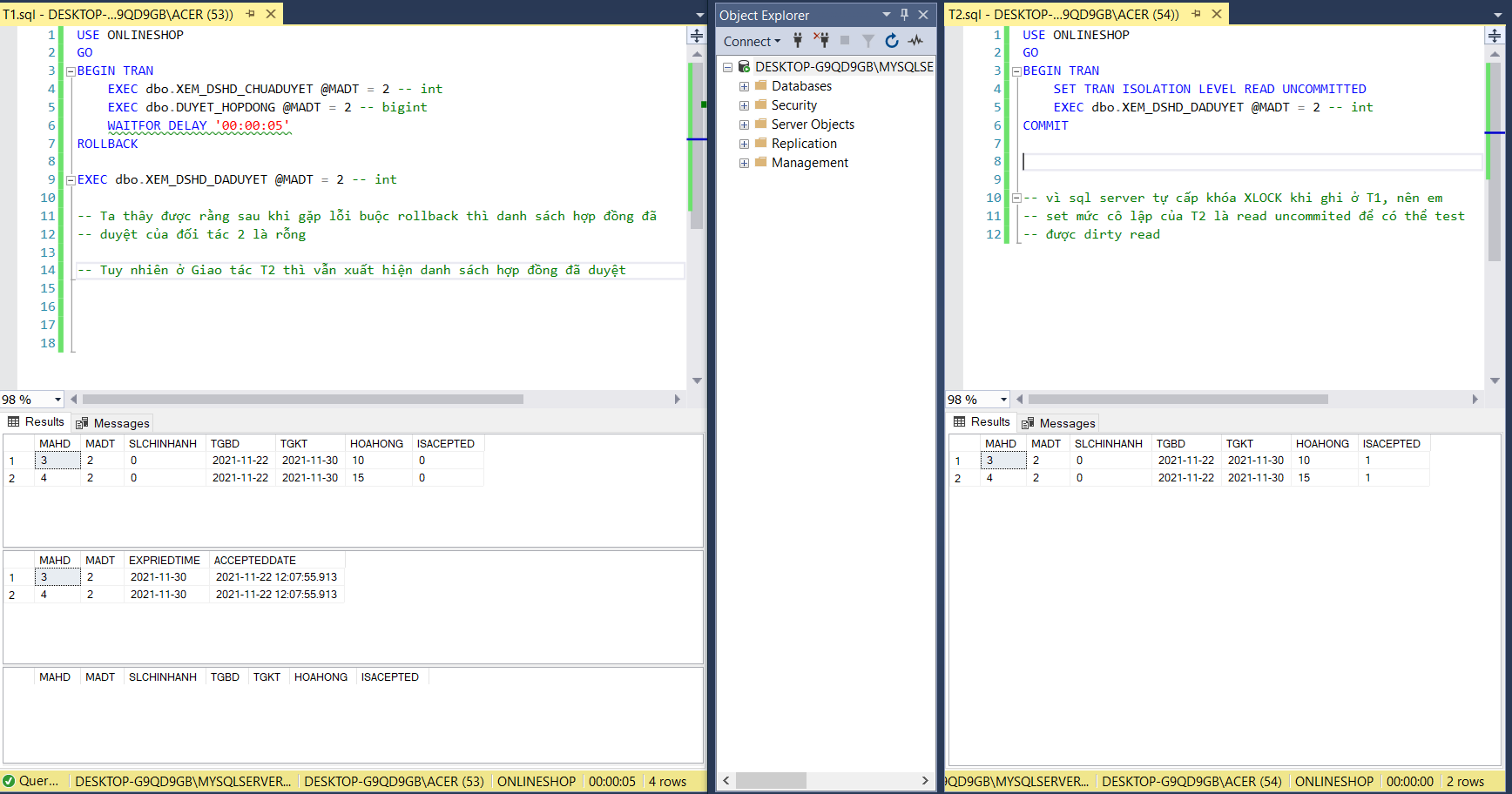
Ta thấy rằng, ban đầu số lượng hợp đồng cần thông báo gia hạn của đối tác 1 là hợp đồng 1, vì hợp đồng 1 còn 2 ngày nữa hết hạn (test vào ngày 22/11/2021)



Tuy nhiên do sau đó Đối tác 1 thực hiện gia hạn hợp đồng (T2), nên dẫn đến danh sách thông báo gia hạn của đối tác T1 bị rỗng.



Testcase3:

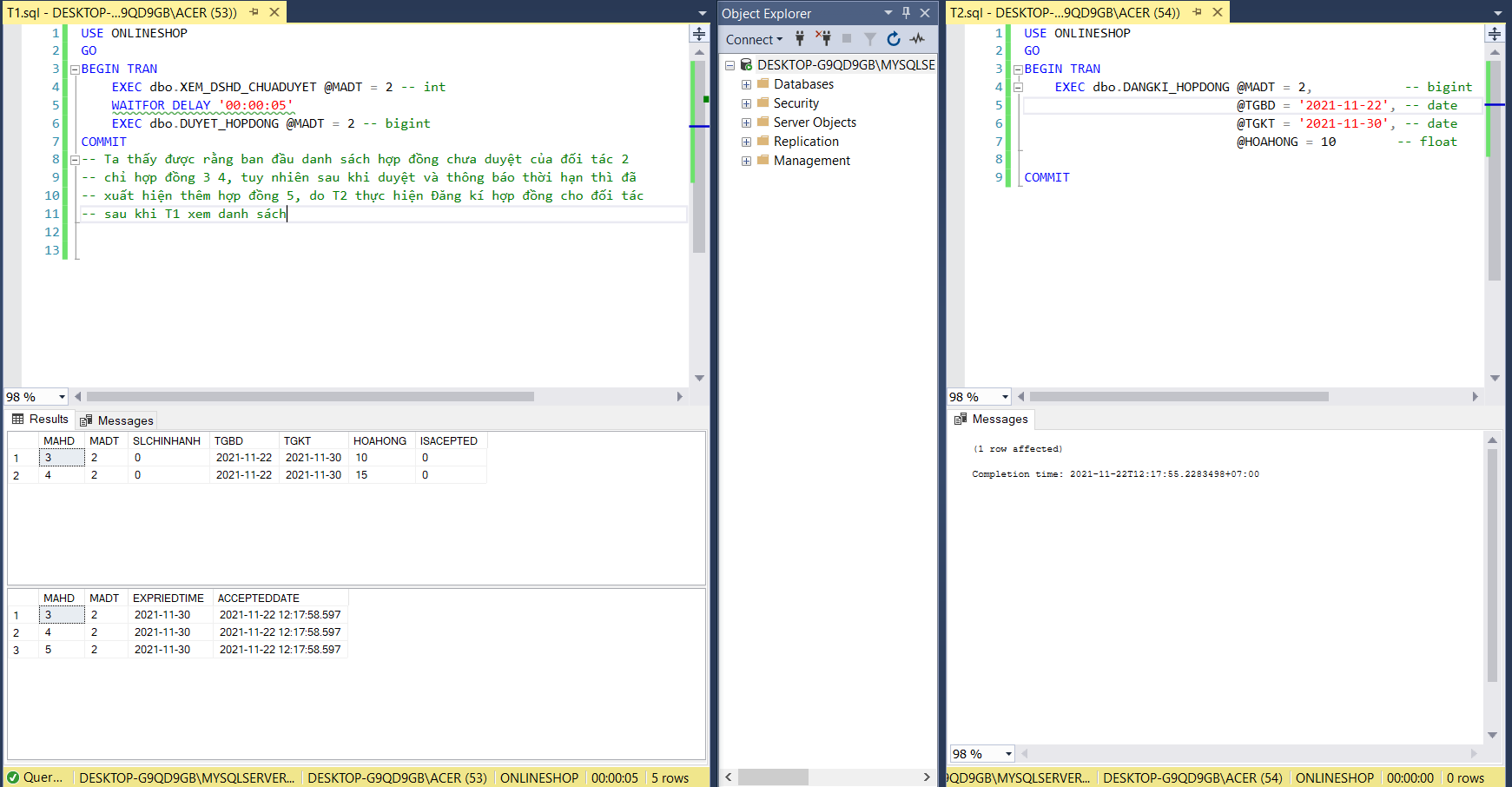


Danh sách hợp đồng đã duyệt

Danh sách hợp đồng đã duyệt

Ta thấy rằng danh danh sách hợp đồng đã duyệt của đối tác 2 là rỗng (do T1 rollback), tuy nhiên T2 vẫn đọc được danh sách hợp đồng đã duyệt của đối tác 2, do T2 đọc khi T1 chưa thực hiện rollback.

***Testcase4:***

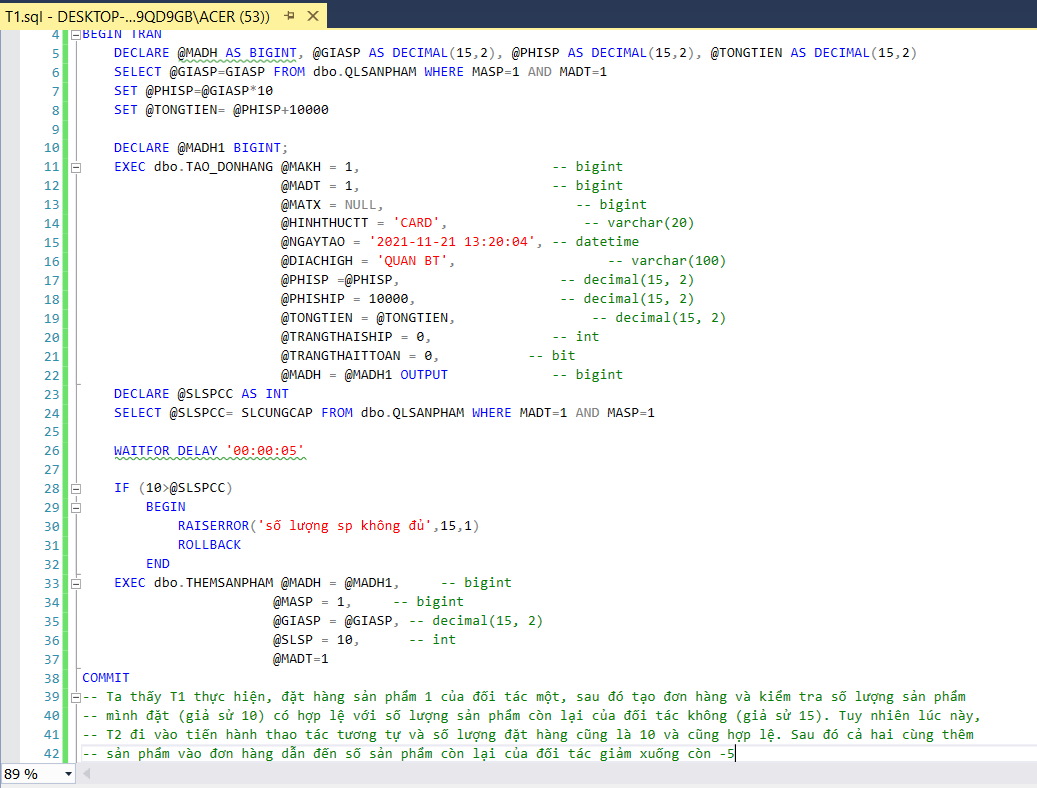


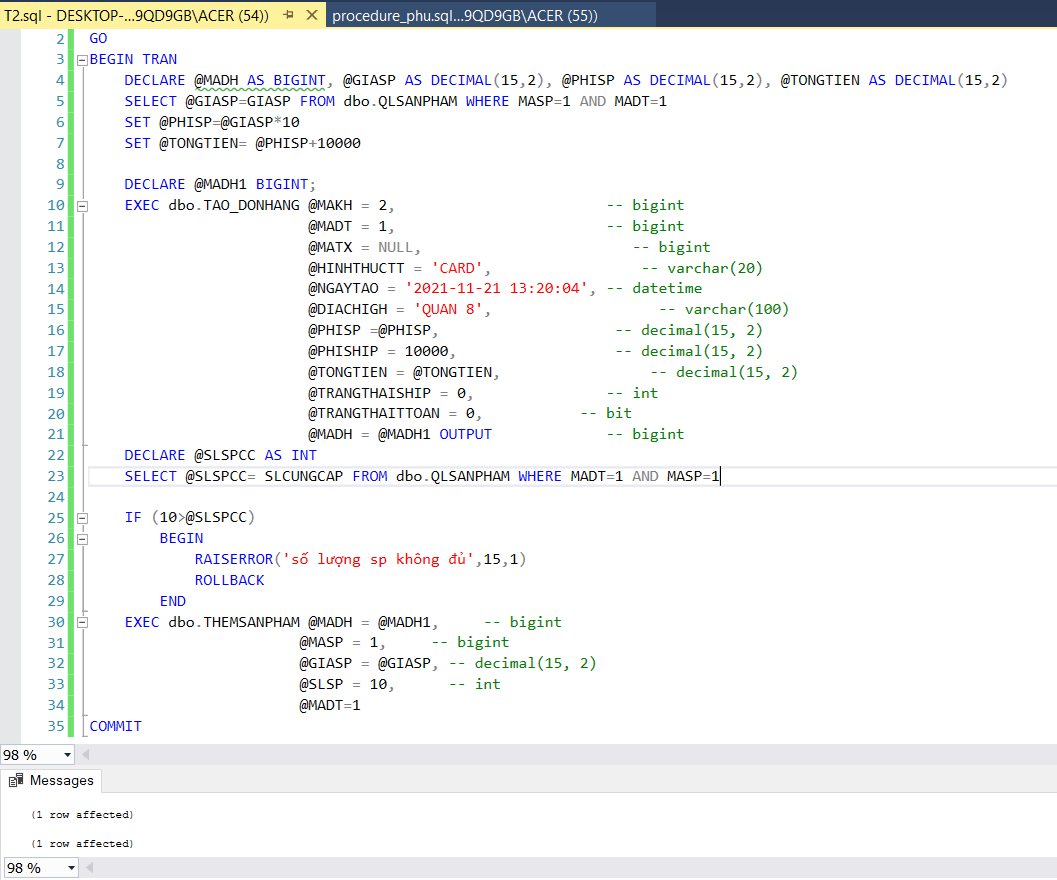
Danh sách hợp đồng được duyệt

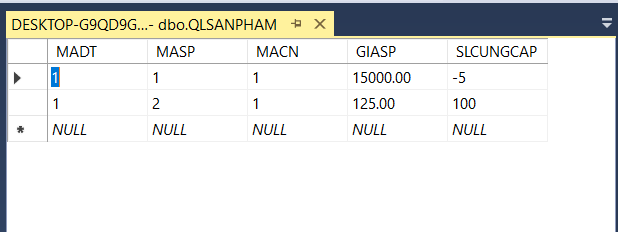
Danh sách hợp đồng cần duyệt

Ta thấy rằng danh sách hợp đồng cần duyệt và danh sách hợp đồng được duyệt của 2 lần đọc ở T1 khác nhau, do trong T1 đang đọc và duyệt hợp đồng, thì T2 thực hiện đăng kí hợp đồng.

***Testcase5:***

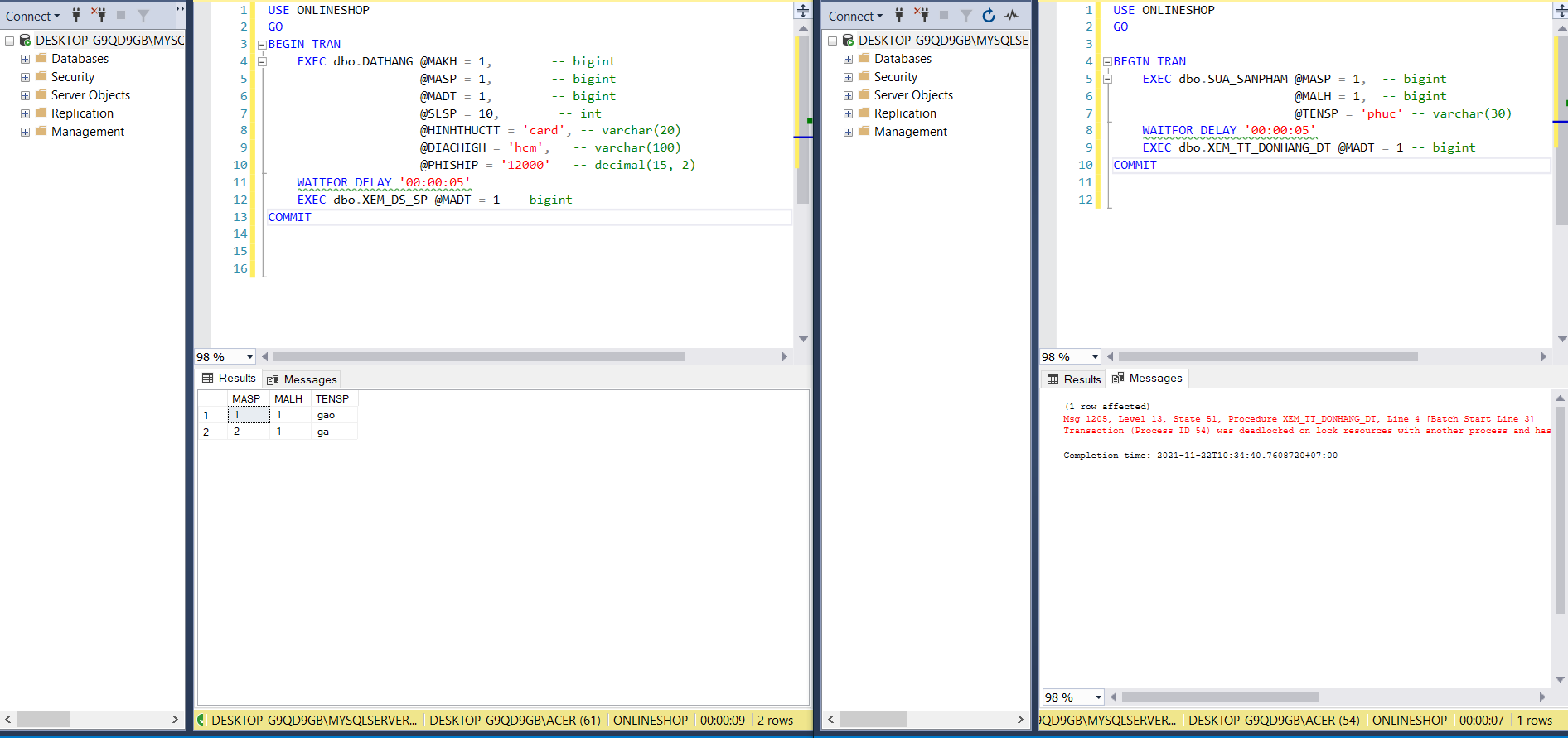






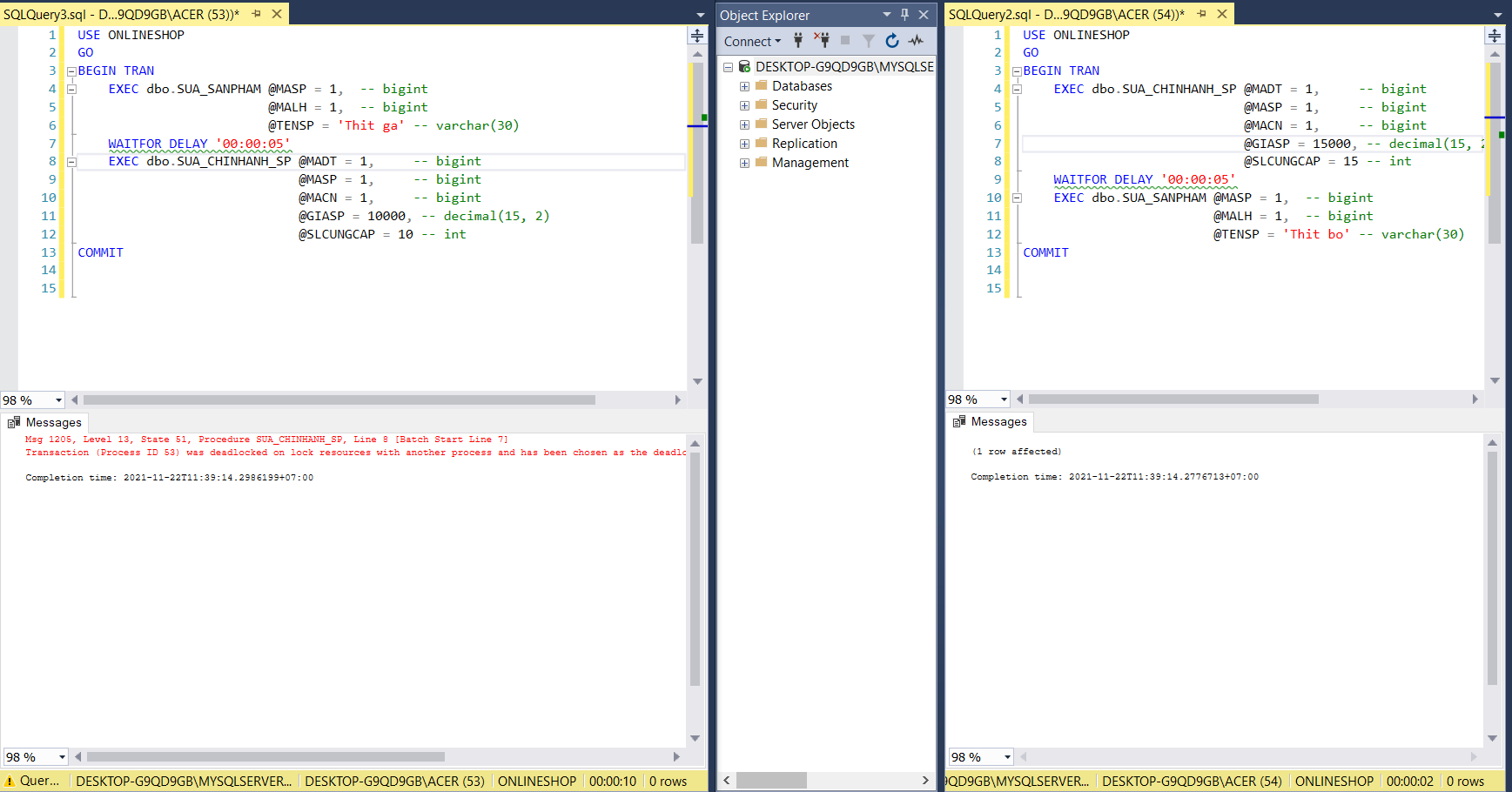
Ở trong test case trên thì số lượng sản phẩm 1 của đối tác 1 ban đầu là 15, sau đó T1 và T2 cùng vào hệ thống và thực hiện đặt hàng với sản phẩm 1 của đối tác 1 với số lượng 10, và cả hai đều qua được giai đoạn kiểm tra số lượng sản phẩm, do cả hai đều nhỏ hơn 15, tuy nhiên sau đó cả hai cùng thực hiện thêm sản phẩm thì số lượng sản phẩm còn lại của sản phẩm 1 của đối tác 1 bị giảm xuống -5.

***Testcase6:***

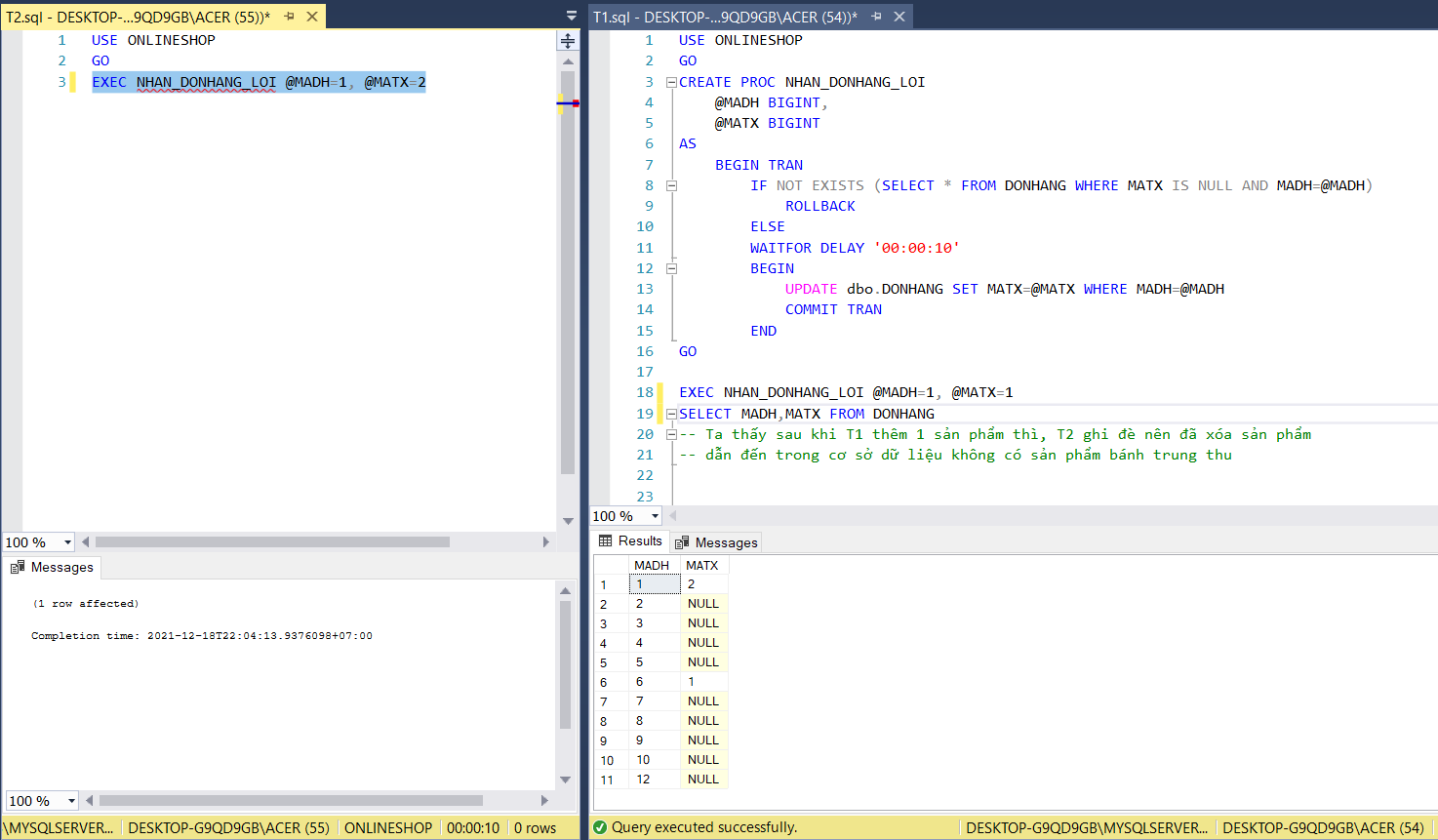


Deadlock do T1 giữ khóa trên bảng QLSanPham, sau đó yêu cầu tài nguyên trên bảng SanPham, mà bảng này bị T2 khóa XLock và T2 lại yêu cầu tài nguyên trên bảng QLSanPham đang bị T1 khóa, dẫn đến circle deadlock.

***Testcase7:***

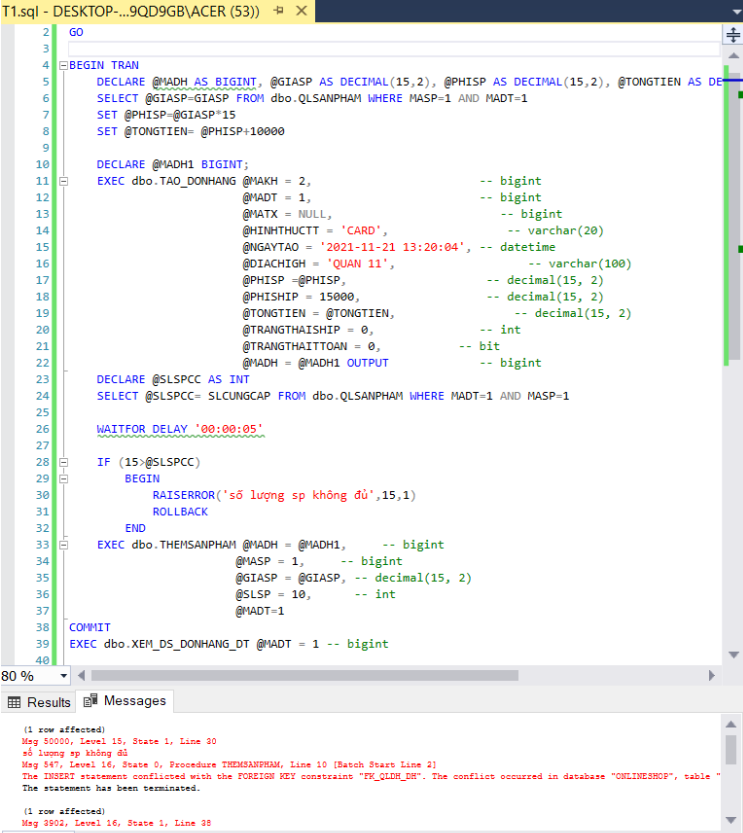


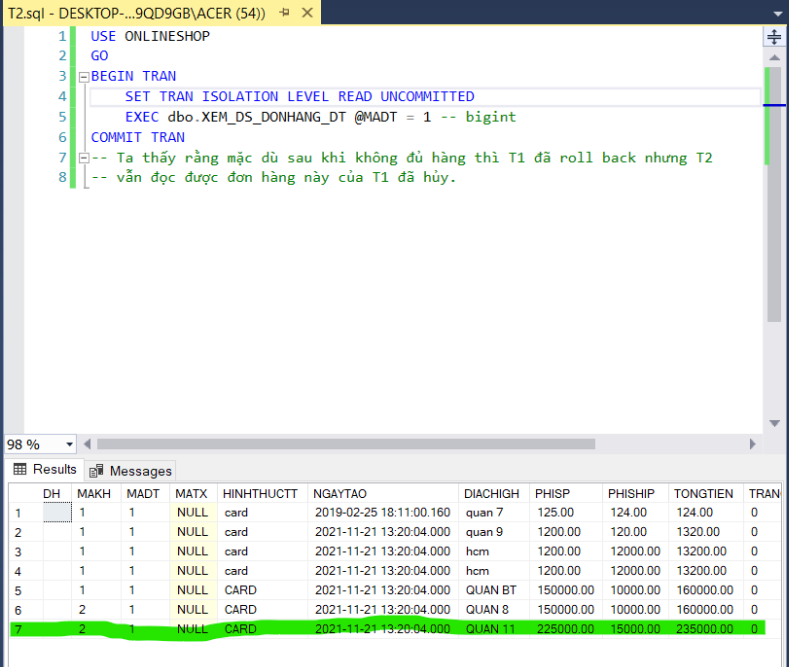
***Testcase8:***



Ta thấy mặc dù T1 thực hiện trước tuy nhiên T2 mới là người nhận được đơn hàng này.

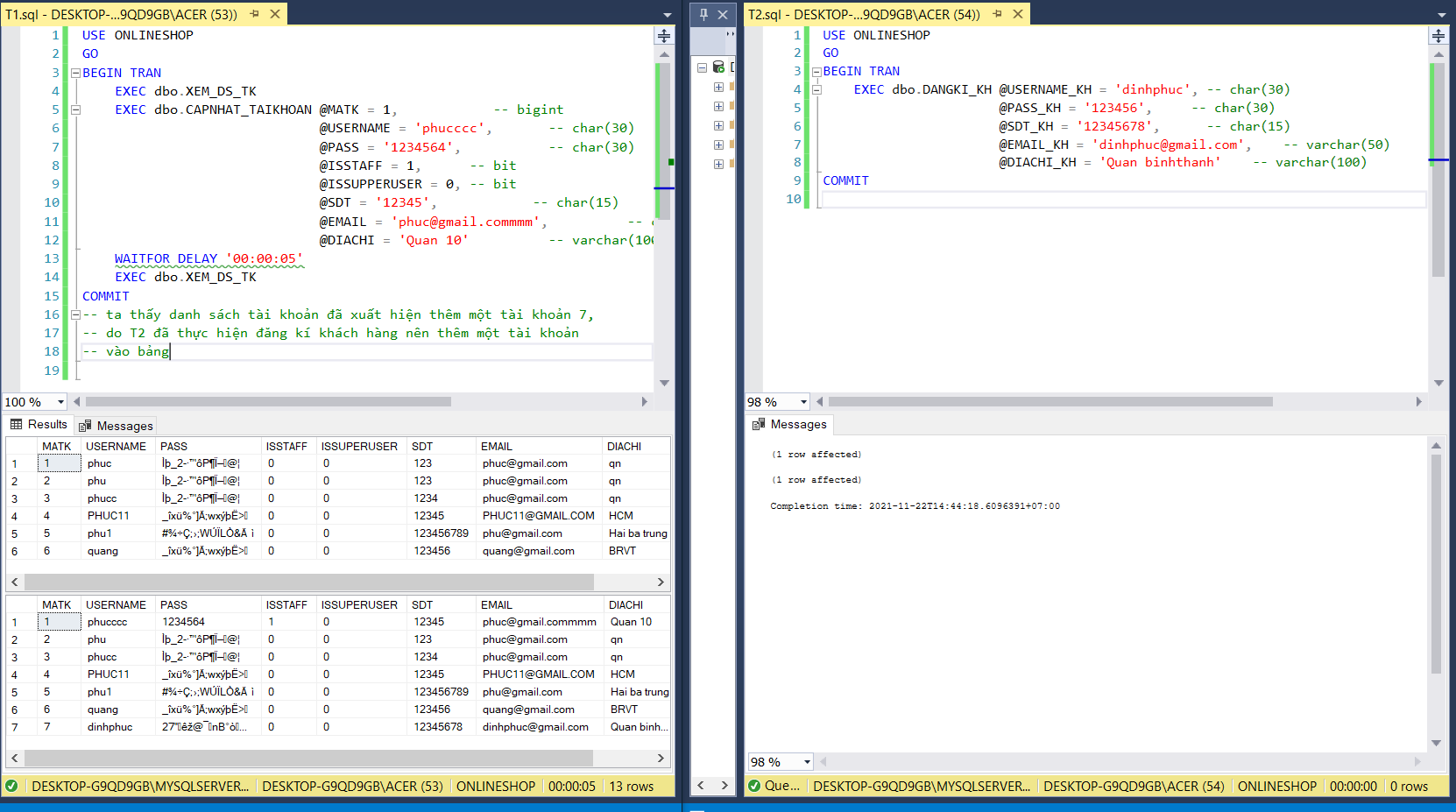
***Testcase9:***



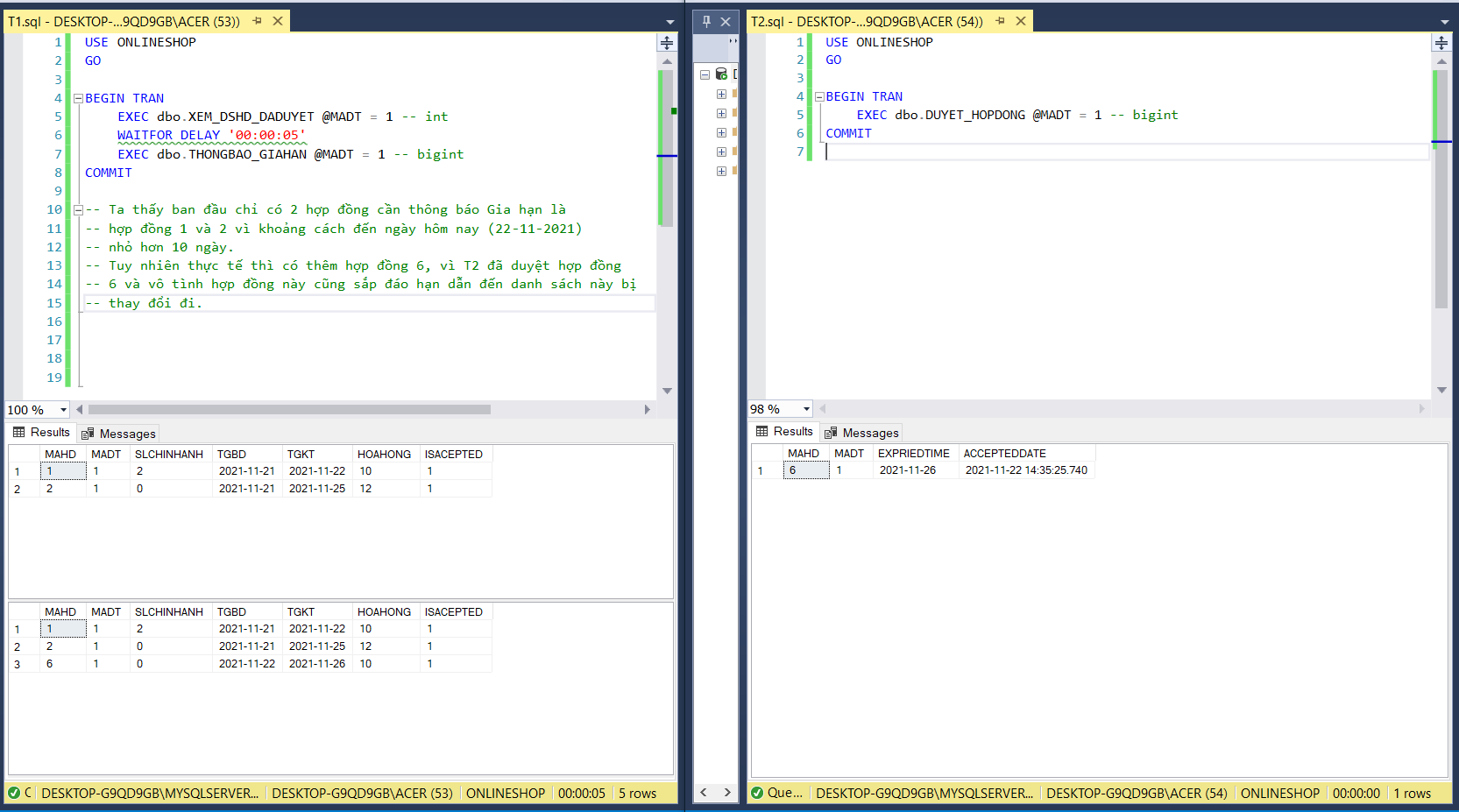


Ta thấy, rằng T1 thực hiện giao tác mua hàng, sản phẩm 1 của đối tác 1, với sô lượng hàng là 15 tuy nhiên trong hệ thống chỉ còn 10 sản phẩm, dẫn đến bị rollback, tuy nhiên trước khi rollback thì T2 đọc bảng đơn hàng, do T1 đã tạo đơn hàng trước đó nên T2 đọc bảng đơn hàng bị sai, rơi vào dirty read.

***Testcase10:***



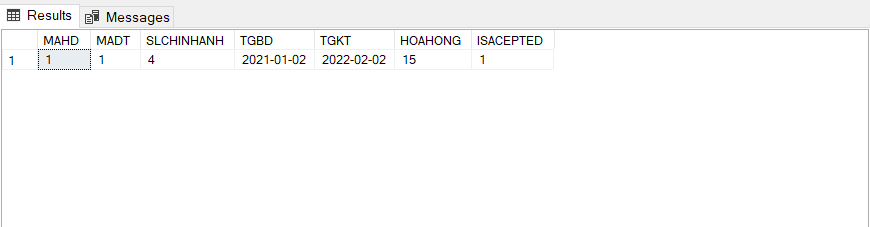
***Testcase11:***



* 1. Demo xử lý lỗi trên SQL:

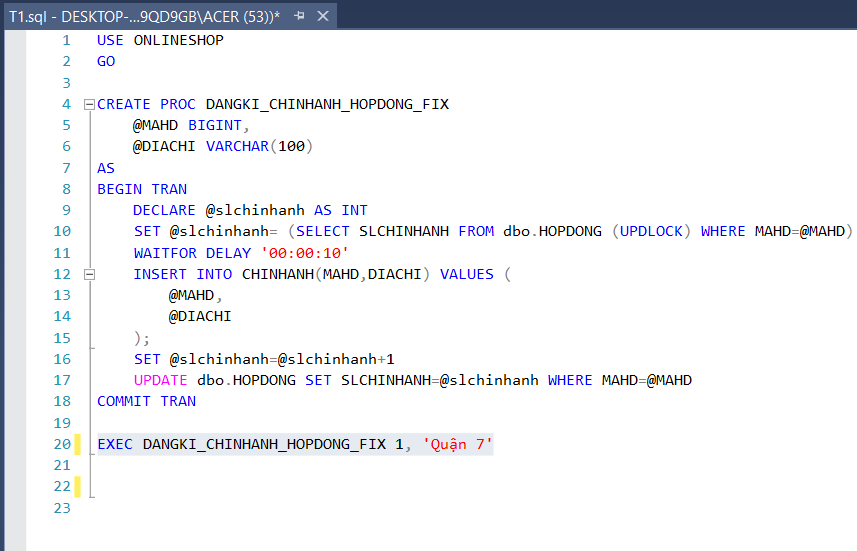
***Test case 1:***

Ban đầu:

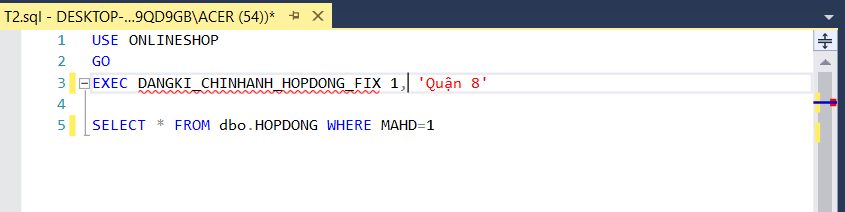


Số lượng chi nhánh ban đầu là 4.

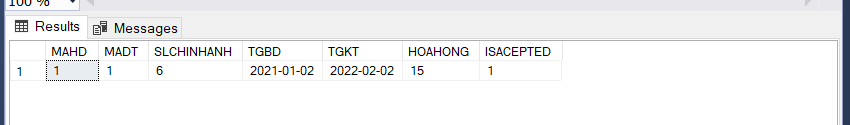
T1: thêm UPDLOCK cho giao tác



T2:

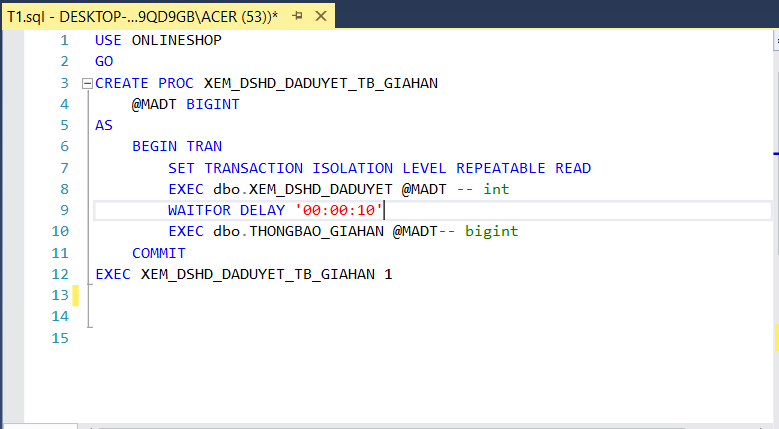


Số lượng hợp chi nhánh lúc sau đã tăng lên 6 (đúng)

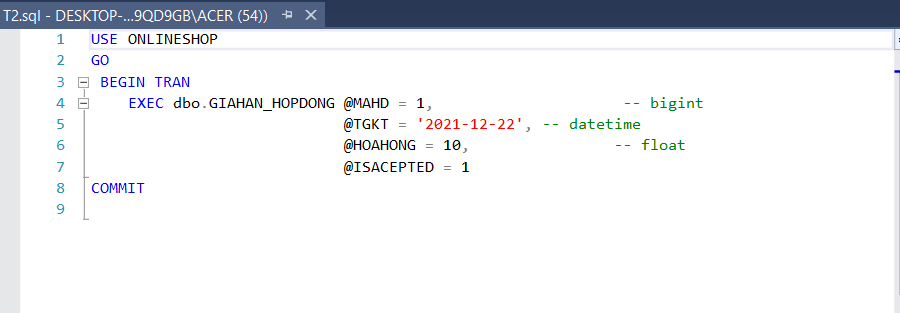


***Test case 2:***

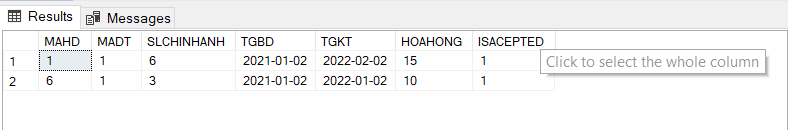
T1: SET TRANSACTION ISOLATION LEVEL REPEATABLE READ



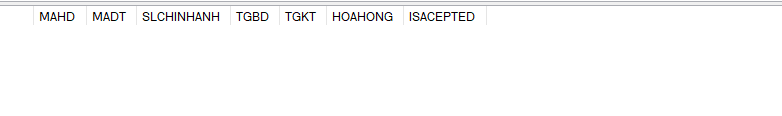
T2:



Danh sách hợp đồng chưa duyệt (T1 xem lần 1)



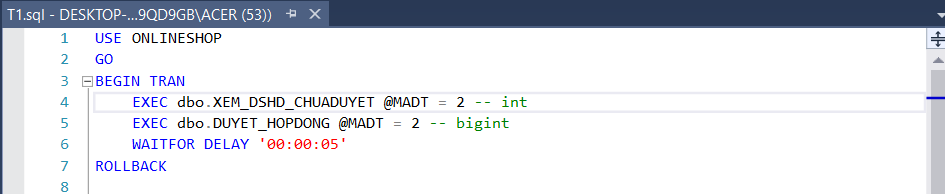
Danh sách thông báo gia hạn (T2 xem lần 2)



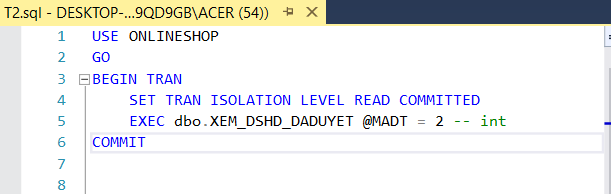
Ta thấy, vì thời gian hết hạn của hai hợp đồng đều còn hơn 10 ngày, nên danh danh sách thông báo gia hạn rỗng (đúng), mặc dù T2 thực hiện thêm hợp đồng có thời gian hết hạn là 22-12 (test ngày 18-12, nên dưới 10 ngày).

***Test case 3:***

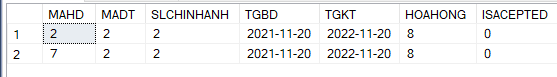
T1:



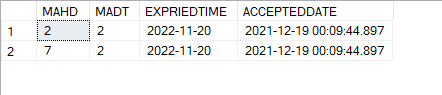
T2: set isolation level read committed



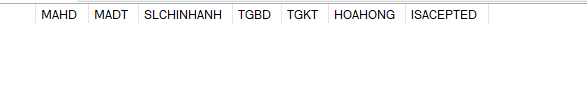
Danh sách hợp đồng chưa duyệt của đối tác 2 (T1)



Danh sách hợp đồng duyệt của đối tác 2(trước khi T1 rollback):



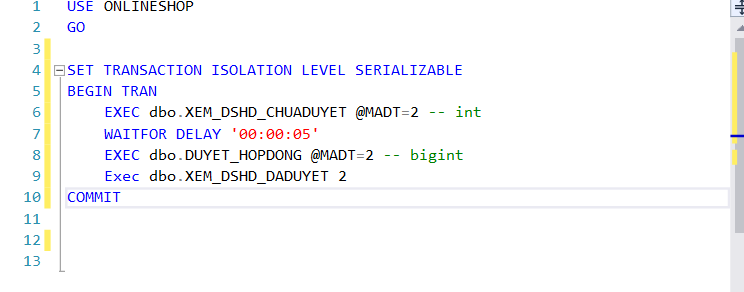
Tuy nhiên, T1 rollback nên danh sách hợp đồng đã duyệt của T2 đọc là rỗng (đúng):



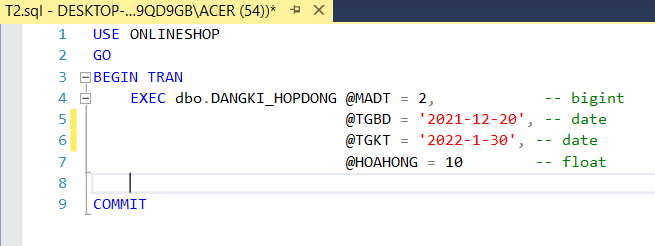
Vậy ta đã giải quyết được dirty read:

***Test case 4:***

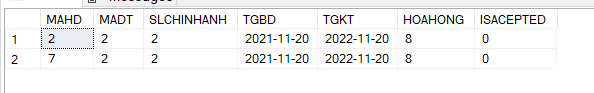
T1: SET TRANSACTION ISOLATION LEVEL SERIALIZABLE



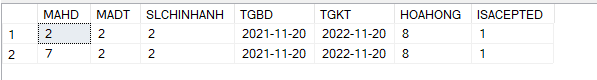
T2:



Danh sách hợp đồng chưa duyệt của đối tác 2 (lần đọc thứ nhất của T1)



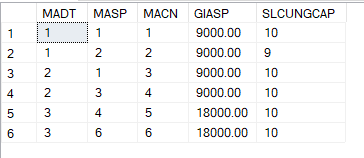
Danh sách hợp đồng đã duyệt của đối tác 2 (lần đọc thứ 2 của T1)



Mặc dù T2 đã insert thêm một hợp đồng tuy nhiên danh sách hợp đồng chưa duyệt và đã duyệt trong một giao tác vẫn đồng nhất, chứng ta chúng ta đã giải quyết được phantom read.

***Test case 5:***

Số lượng cung cấp sản phẩm 1 của đối tác 1 là 10:



Ta thực hiện đặt UPDLOCK trên bảng QLSANPHAM cho giao tác đặt hàng, để đảm bảo hai khách hàng không thể kiểm tra cùng lúc.

Procedure:

CREATE PROC DATHANG\_FIX

@MAKH BIGINT,

@MASP BIGINT,

@MADT BIGINT,

@SLSP INT,

@HINHTHUCTT VARCHAR(20),

@DIACHIGH VARCHAR(100),

@PHISHIP DECIMAL(15,2)

AS

BEGIN TRAN

DECLARE @MADH AS BIGINT, @GIASP AS DECIMAL(15,2), @PHISP AS DECIMAL(15,2), @TONGTIEN AS DECIMAL(15,2)

SELECT @GIASP=GIASP FROM dbo.QLSANPHAM WHERE MASP=@MASP AND MADT=@MADT

SET @PHISP=@GIASP\*@SLSP

SET @TONGTIEN= @PHISP+@PHISHIP

DECLARE @MADH1 BIGINT;

EXEC dbo.TAO\_DONHANG @MAKH = @MAKH, -- bigint

@MADT = @MADT, -- bigint

@MATX = NULL, -- bigint

@HINHTHUCTT = @HINHTHUCTT, -- varchar(20)

@NGAYTAO = GETDATE, -- datetime

@DIACHIGH = @DIACHIGH, -- varchar(100)

@PHISP =@PHISP, -- decimal(15, 2)

@PHISHIP = @PHISHIP, -- decimal(15, 2)

@TONGTIEN = @TONGTIEN, -- decimal(15, 2)

@TRANGTHAISHIP = 0, -- int

@TRANGTHAITTOAN = 0, -- bit

@MADH = @MADH1 OUTPUT -- bigint

DECLARE @SLSPCC AS INT

SELECT @SLSPCC= SLCUNGCAP FROM dbo.QLSANPHAM (UPDLOCK) WHERE MADT=@MADT AND MASP=@MASP

WAITFOR DELAY '00:00:05'

print (@SLSP)

print(@SLSPCC)

IF (@SLSP>@SLSPCC)

rollback tran

ELSE

begin

EXEC dbo.THEMSANPHAM @MADH = @MADH1, -- bigint

@MASP = @MASP, -- bigint

@GIASP = @GIASP,

@SLSP = @SLSP, -- int

@MADT=@MADT

COMMIT tran

end

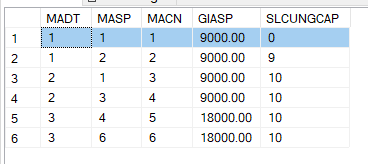
T1: Khách hàng 1 đặt 10 sản phẩm 1 của đối tác 1



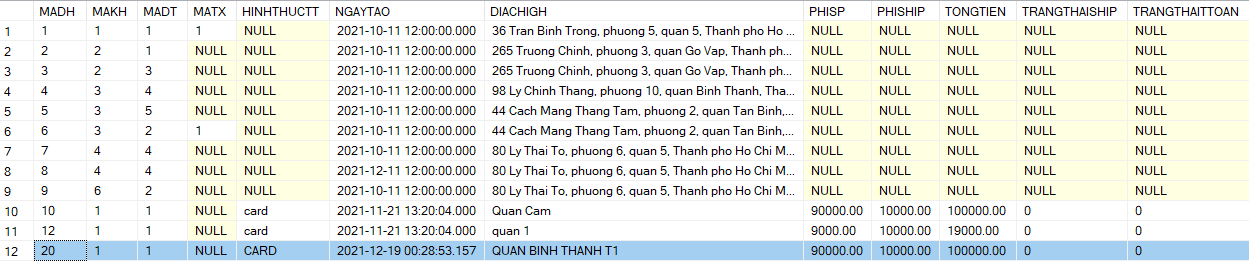
T2: Khách hàng 2 đặt 10 sản phẩm 1 của đối tác 1



Số lượng sản phẩm sau khi khách hàng đặt hàng:



Bảng đơn hàng:



T1 đặt được hàng vì T1 đến trước, T2 không đặt được hàng -> giải quyết được lost update

***Test case 6:***

***Test case 7:***

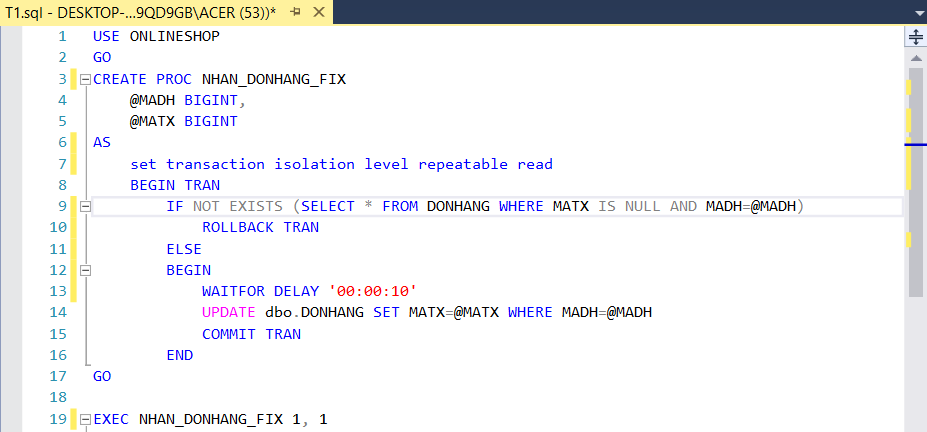
***Test case 8:***

Tình trạng đơn hàng 1 trước khi thực hiện nhận đơn hàng (MATX là null)

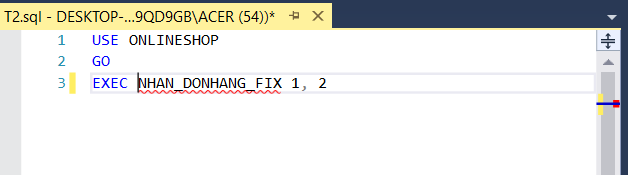


Thêm UPDLOCK cho giao tác nhận hàng

T1:



T2:



Sau khi T1 và T2 thực hiện nhận đơn hàng (T1 đến trước):



Vì Taixe 1 đến trước nên Taixe 1 nhận được đơn hàng này.

***Test case 9:***

Procedure đặt hàng:

CREATE PROC DATHANG\_FIX

@MAKH BIGINT,

@MASP BIGINT,

@MADT BIGINT,

@SLSP INT,

@HINHTHUCTT VARCHAR(20),

@DIACHIGH VARCHAR(100),

@PHISHIP DECIMAL(15,2)

AS

BEGIN TRAN

DECLARE @MADH AS BIGINT, @GIASP AS DECIMAL(15,2), @PHISP AS DECIMAL(15,2), @TONGTIEN AS DECIMAL(15,2)

SELECT @GIASP=GIASP FROM dbo.QLSANPHAM WHERE MASP=@MASP AND MADT=@MADT

SET @PHISP=@GIASP\*@SLSP

SET @TONGTIEN= @PHISP+@PHISHIP

DECLARE @MADH1 BIGINT;

EXEC dbo.TAO\_DONHANG @MAKH = @MAKH, -- bigint

@MADT = @MADT, -- bigint

@MATX = NULL, -- bigint

@HINHTHUCTT = @HINHTHUCTT, -- varchar(20)

@NGAYTAO = GETDATE, -- datetime

@DIACHIGH = @DIACHIGH, -- varchar(100)

@PHISP =@PHISP, -- decimal(15, 2)

@PHISHIP = @PHISHIP, -- decimal(15, 2)

@TONGTIEN = @TONGTIEN, -- decimal(15, 2)

@TRANGTHAISHIP = 0, -- int

@TRANGTHAITTOAN = 0, -- bit

@MADH = @MADH1 OUTPUT -- bigint

DECLARE @SLSPCC AS INT

SELECT @SLSPCC= SLCUNGCAP FROM dbo.QLSANPHAM (UPDLOCK) WHERE MADT=@MADT AND MASP=@MASP

WAITFOR DELAY '00:00:05'

print (@SLSP)

print(@SLSPCC)

IF (@SLSP>@SLSPCC)

rollback tran

ELSE

begin

EXEC dbo.THEMSANPHAM @MADH = @MADH1, -- bigint

@MASP = @MASP, -- bigint

@GIASP = @GIASP,

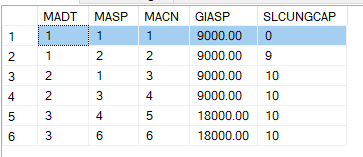
@SLSP = @SLSP, -- int

@MADT=@MADT

COMMIT tran

end

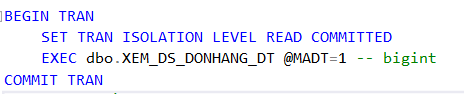
Số lượng của sản phẩm 1 của đối tác 1:



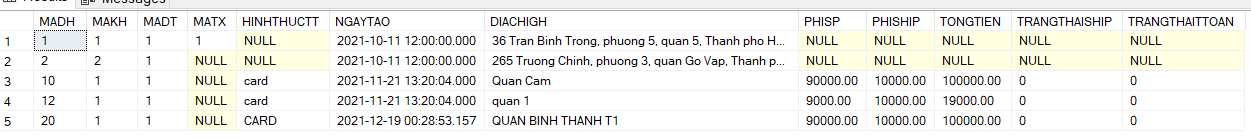
T1: khách hàng đặt sản phẩm 1 của đối tác 1 số lượng 10



T2: SET INSOLATION LEVEL READ COMMITTED



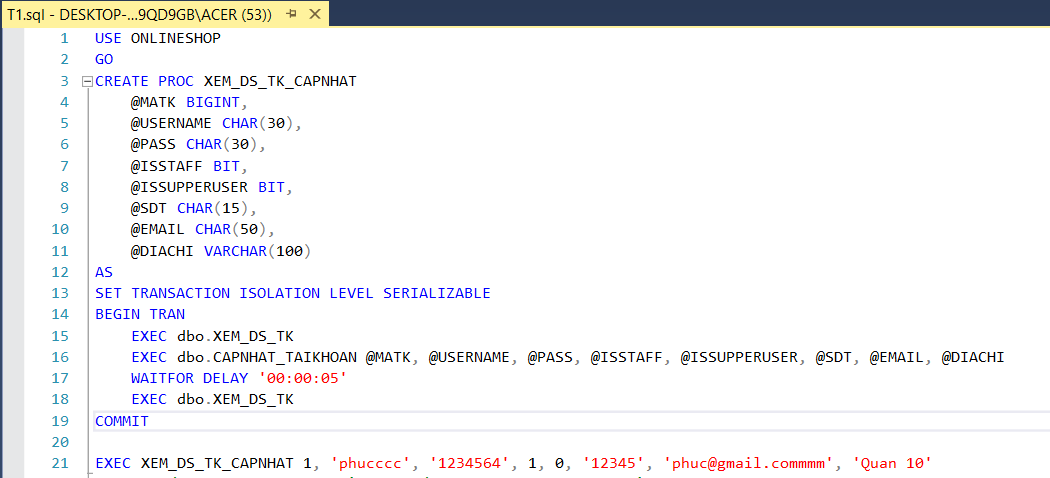
Bảng danh sách đơn hàng của giao tác T2 đọc được:



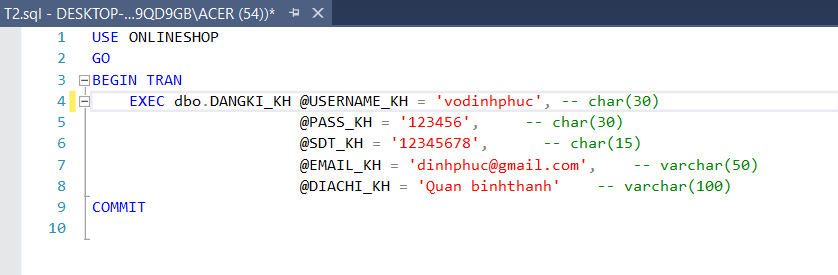
Vì số lượng sản phẩm mua (10) lớn hơn số lượng hàng còn lại (0), nên dẫn đến T1 rollback (hủy tạo đơn hàng), dẫn đến T2 không đọc được đơn hàng đã hủy của T1 => chúng ta đã giải quyết được dirty read.

***Test case 10:***

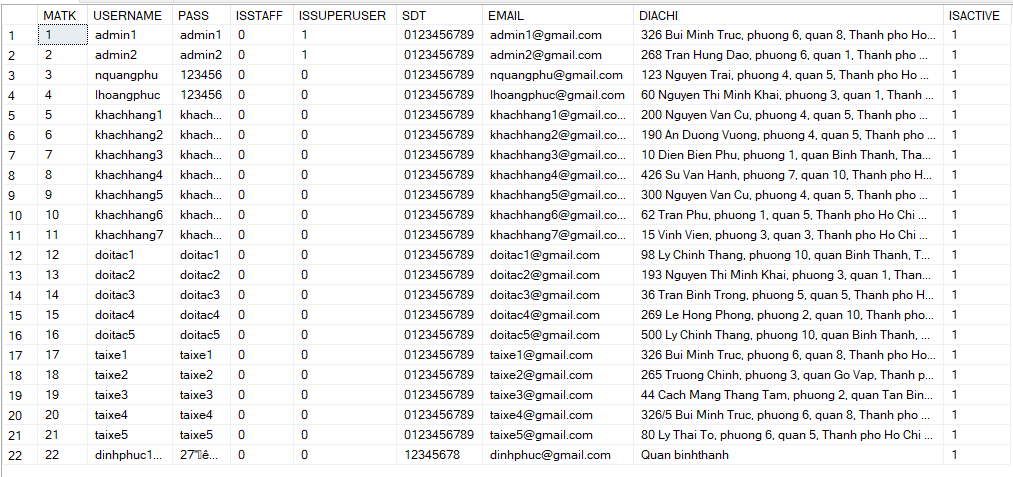
T1: set isolation level serializable



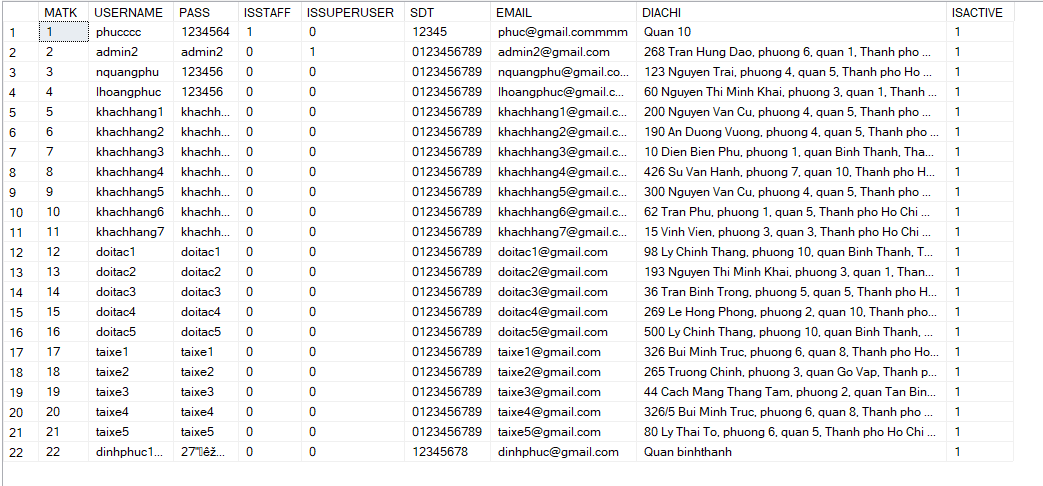
T2:



Kết quả lần đọc thứ nhất của T1:



Lần 2:

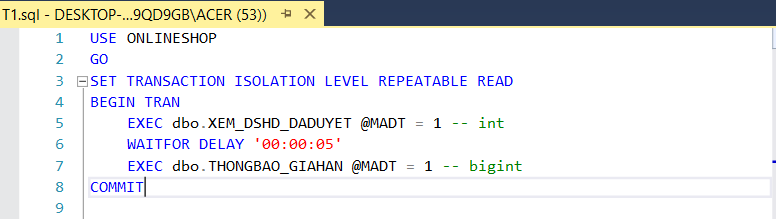


Ta thấy mặc dù, T2 có insert thêm tài khoản những vì ta đã set mức cô lập là serializable nên tranh được phantom.

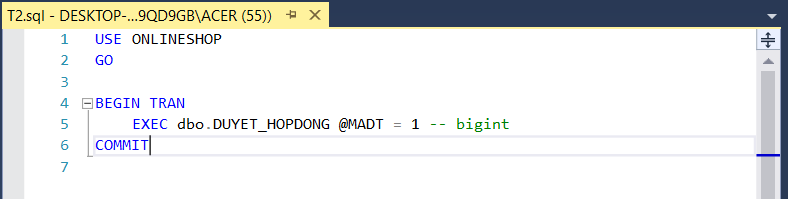
***Test case 10:***

***Test case 11:***

T1: SET TRANSACTION ISOLATION LEVEL REPEATABLE READ



T2:

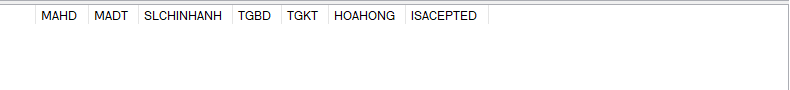


Danh sách hợp đồng đã duyệt của đối tác 1 (T1 đọc lần 1):



Ta thấy ngày hết hạn của hợp đồng 6 của đối tác 1 cách hiện tại (19/12/2021) hơn 10 ngày nên bảng thông báo gia hạn của chúng ta mong muốn sẽ rỗng.

Danh sách thông báo gia hạn:



Vậy ta đã giải quyết được unrepeatable read.