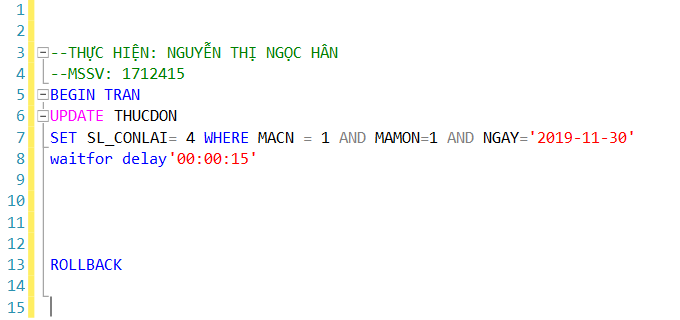
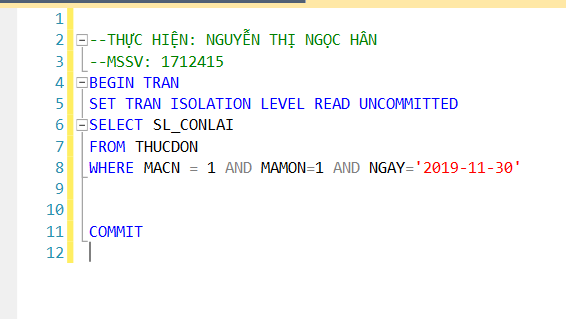
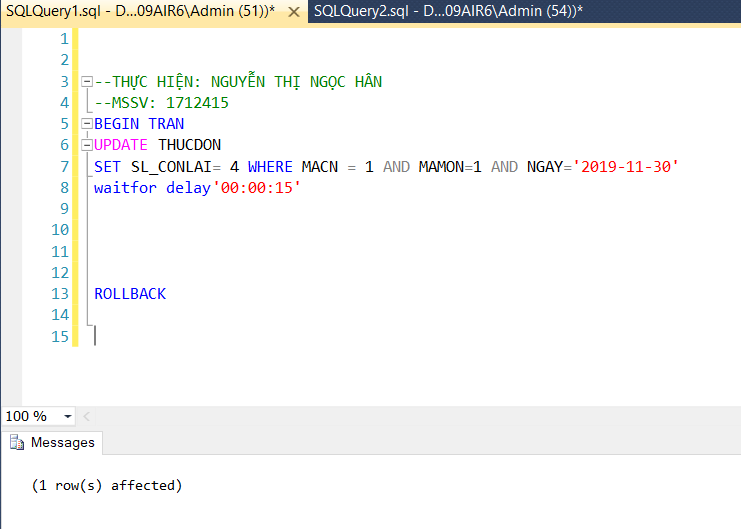
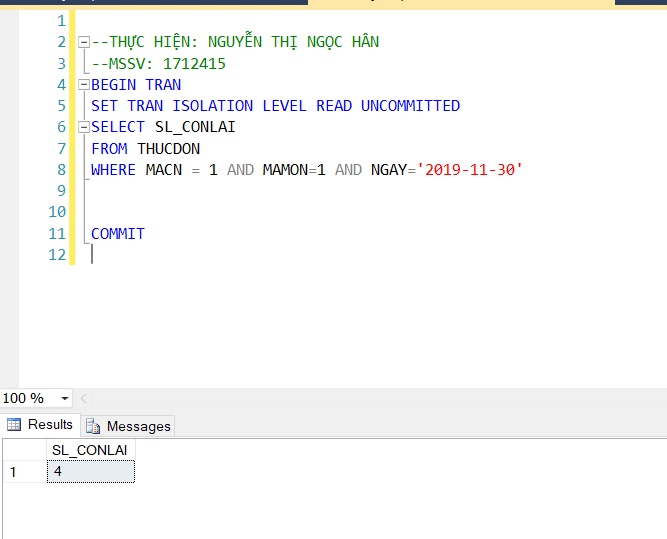
1. ***Dirty read:***
2. ***Lỗi :*** GIả sử món ăn số 1 tại chi nhánh 1 đã được đặt hết và có mộtkhách hàng A huỷ món ăn số 1 tại chi nhánh 1 nhưng chưa commit giao tác, tại thời điểm đó có 1 khách hàng B tới tìm món số 1 tại chi nhánh 1 và thấy số lượng còn lại là 4 ,sau đó khách hàng A đổi ý không huỷ món nữa. Kết quả là khách hàng B đặt được món trong khi nhà hàng không còn món đó nữa. . Ở ví dụ này, để xảy ra lỗi thì giả lập mức cô lập cho giao tác T2 là Read Uncommitted

|  |  |
| --- | --- |
| T1 | T2 |
| BEGIN TRAN  UPDATE THUCDON  SET SL\_CONLAI= 4 WHERE MACN = 1 AND MAMON=1 AND NGAY=’2019-30-11’  waitfor delay‘00:00:05’  ROLLBACK | BEGIN TRAN  SET TRAN ISOLATION LEVEL READ UNCOMMITTED  SELECT SOLUONGCONLAI FROM THUCDON WHERE MACN = 1 AND MAMON=1 AND NGAY=’2019-30-11’  COMMIT |

1. Chạy lỗi trên SQL:

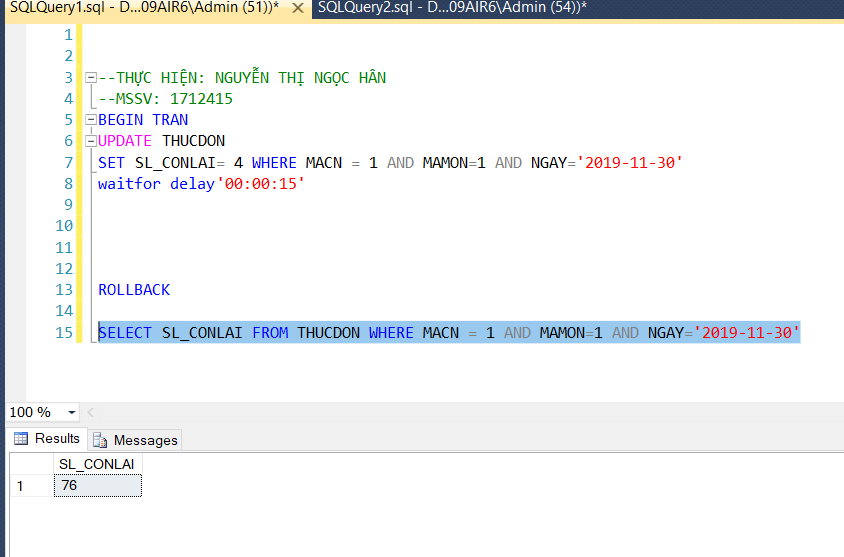
**Đoạn test case của trường hợp Dirty Read:**  

Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy hết giao tác 2 và ta có kết quả:

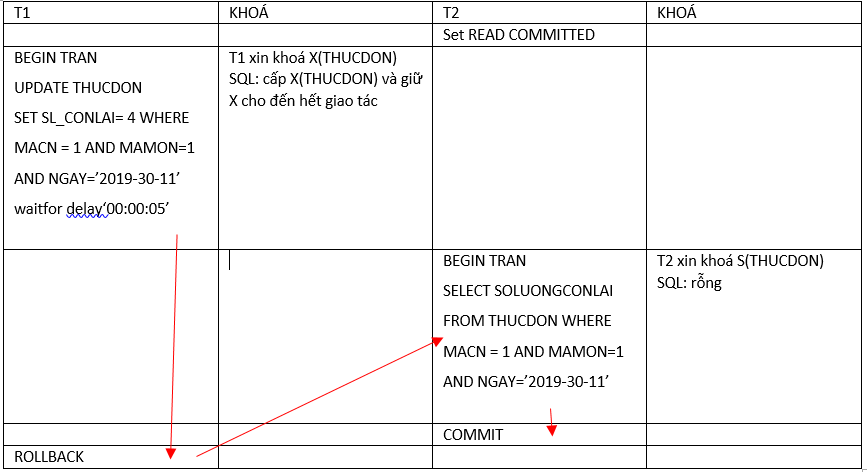
 

* Giao tác 2 đã đọc dữ liệu mà giao tác 1 đã cập nhật nhưng chưa rollback với nội dung Số lượng còn lại của món ăn có mã 1 là 4.

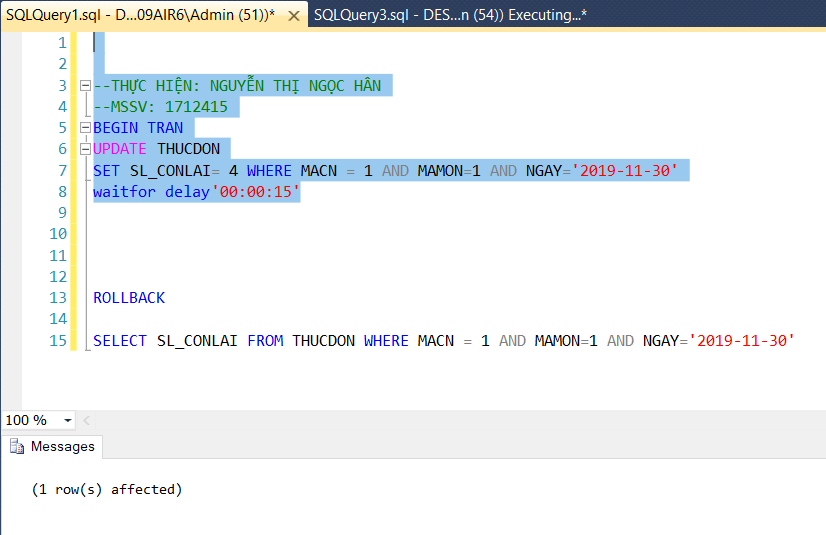
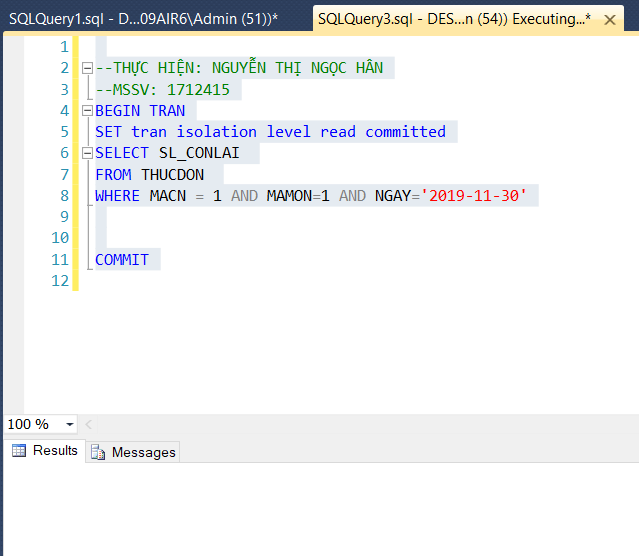
Sau khi Giao tác 1 rollback, thử truy vấn lại thì số lượng còn lại đã trở lại 76.



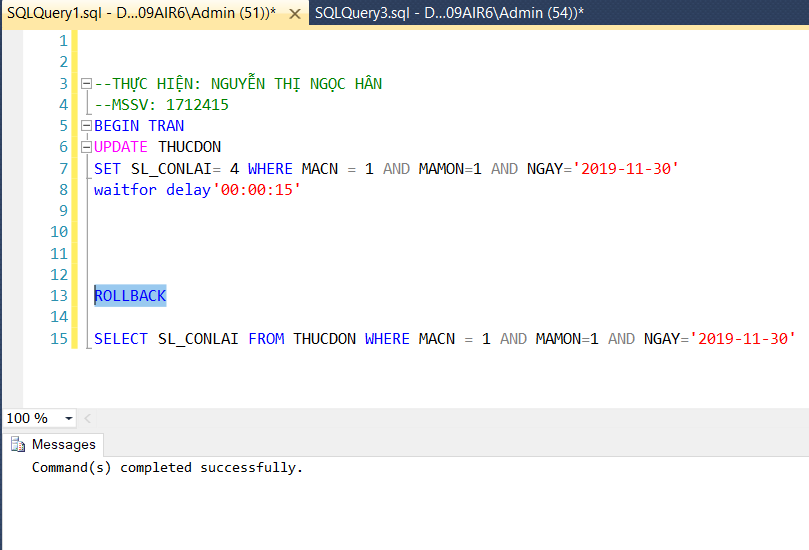
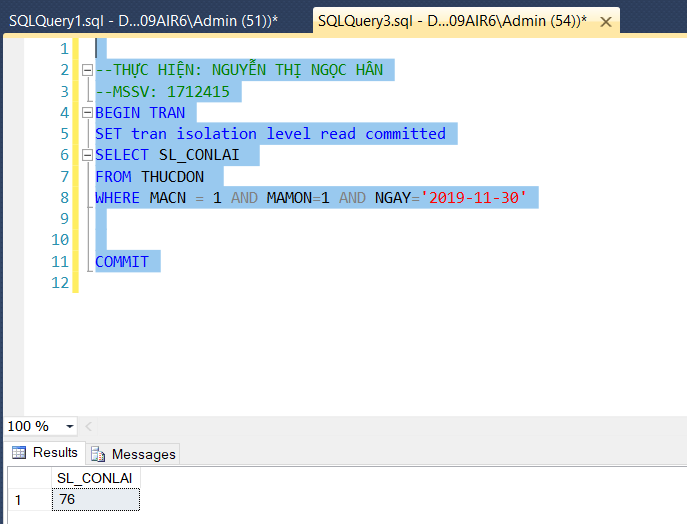
1. Giải pháp: Set cơ chế khóa Read Committed



Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy hết giao tác 2, lúc này T2 không đọc được dữ liệu mà T1 thay đổi do T1 vẫn chưa kết thúc giao tác.

Sau khi T1 rollback, T2 lập tức đọc được dữ liệu ban đầu.

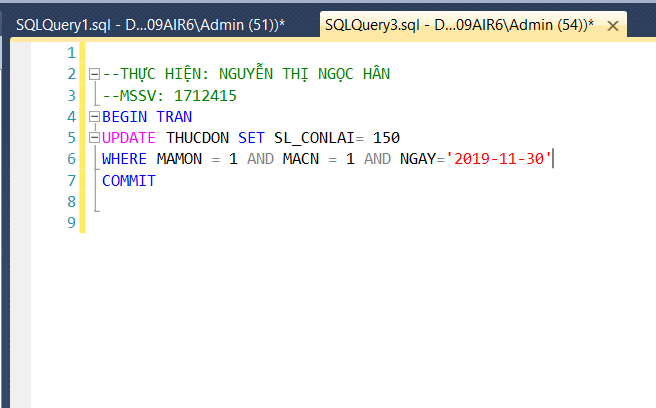
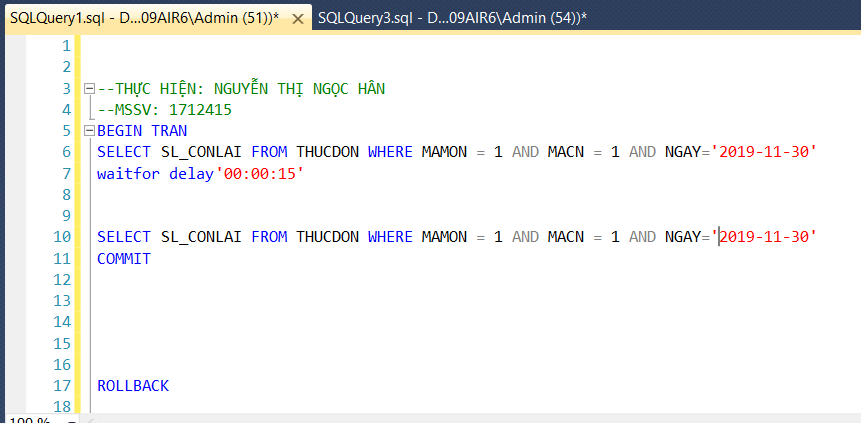
 

1. ***Unrepeatable read:***
2. ***Lỗi 4:*** Khách hàng đang xem giá số lượng món ăn số 1 ở chi nhánh 1, có khách hàng khác đặt món. Quản lí tiến hành cập nhật số lượng thì khách hàng đang xem sẽ thấy số lượng thay đổi

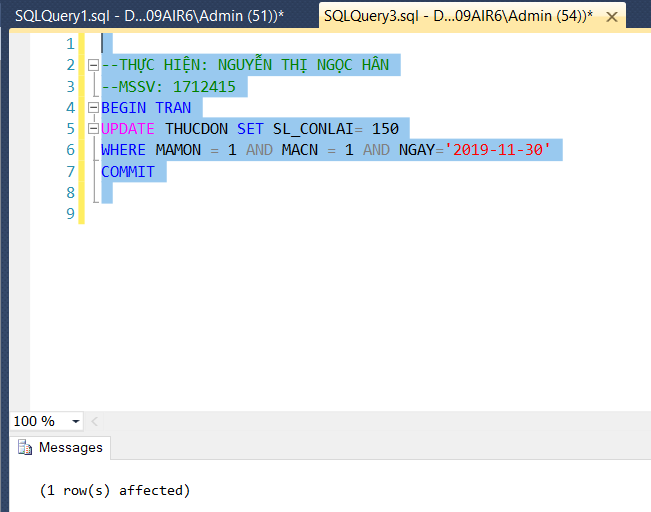
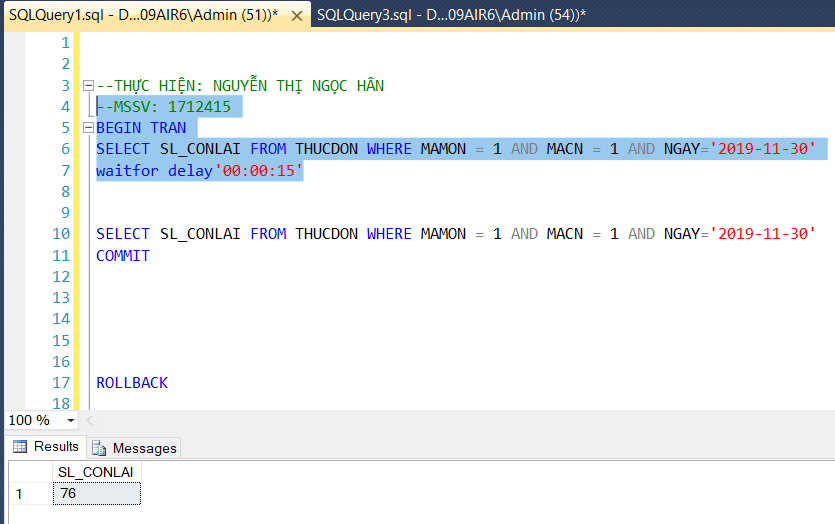
|  |  |
| --- | --- |
| T1 | T2 |
| BEGIN TRAN  SELECT SL\_CONLAI FROM THUCDON WHERE MAMON = 1 AND MACN = 1 AND NGAY=’2019-11-30’  waitfor delay‘00:00:15’  SELECT SL\_CONLAI FROM THUCDON WHERE MAMON = 1 AND MACN = 1 AND NGAY=’2019-11-30’  COMMIT | BEGIN TRAN  UPDATE THUCDON SET SL\_CONLAI= 150 WHERE MAMON = 1 AND MACN = 1 AND NGAY=’2019-11-30’  COMMIT |

1. *Chạy trên SQL*

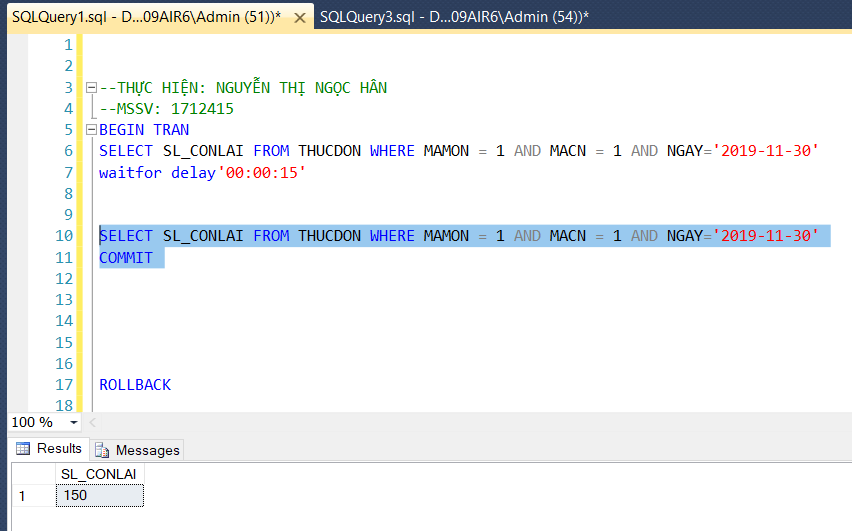
**Đoạn test case của trường hợp Unrepeatable Read: (Giả lập lấy cơ chế Read Committed)**



Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy hết giao tác 2 và ta có kết quả:

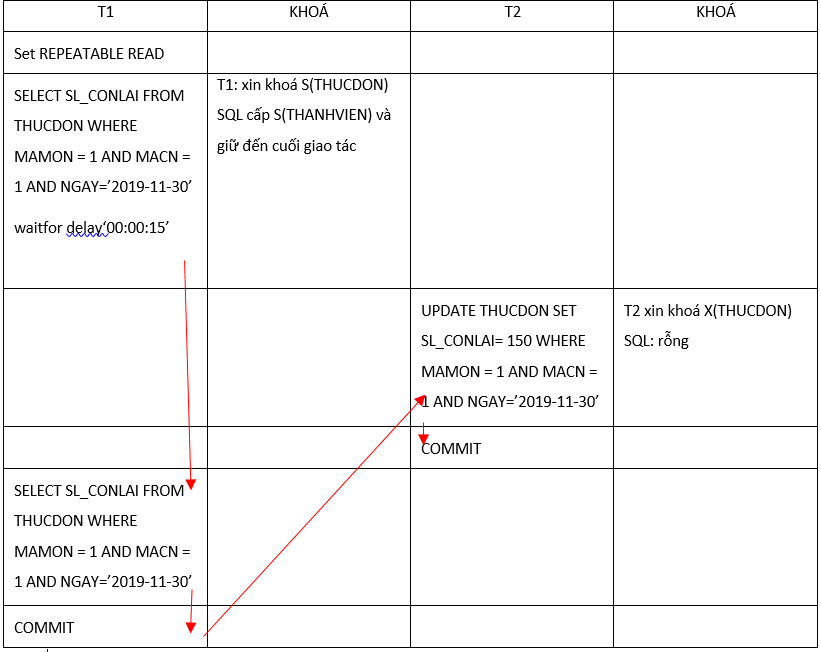


Giao tác 1 đọc dữ liệu của mã tv = 1 lần 2:

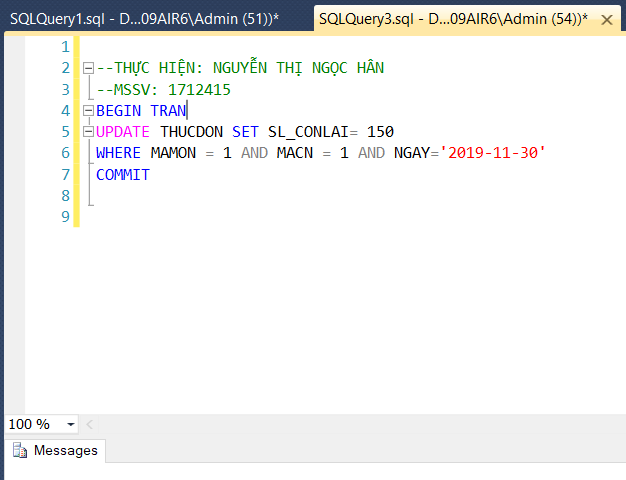
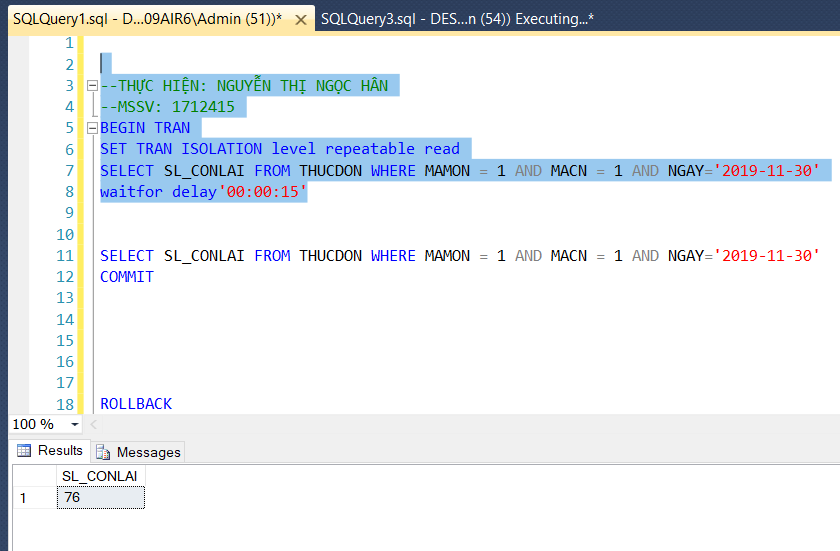


* Ở lần đọc thứ 2 này, Giao tác 1 đã thấy dữ liệu có sự thay đổi, cụ thể là số lượng sản phẩm đã bị thay đổi.

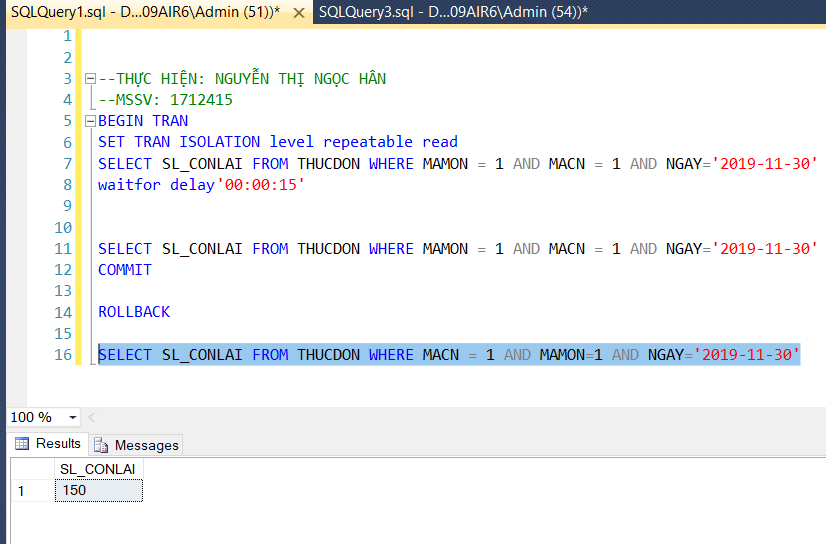
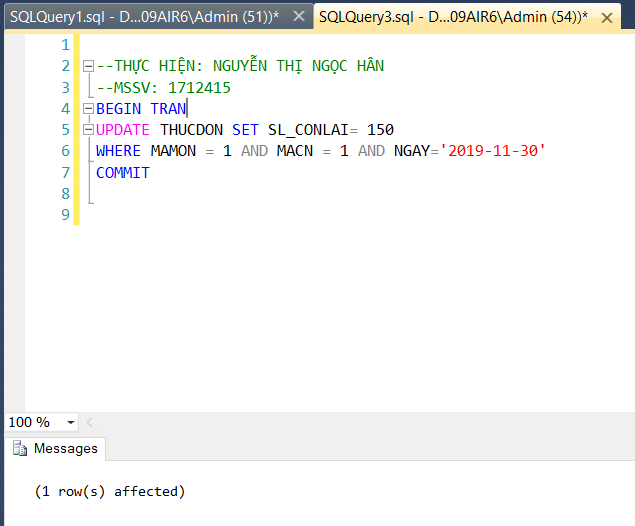
1. Giải pháp: Set cơ chế khóa Repeatable Read



Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy hết giao tác 2, lúc này T2 không cập nhật được dữ liệu do T1 set mức cô lập Repeatable Read. Khóa S đã được giữ cho đến hết giao tác, T2 muốn xin khóa X thì phải đợi T1 hoàn thành xong giao tác.



Sau khi kết thúc giao tác T1, T2 mới vào thực hiện cập nhật dữ liệu của MATV = 1, và đây là kết quả sau khi cập nhật:

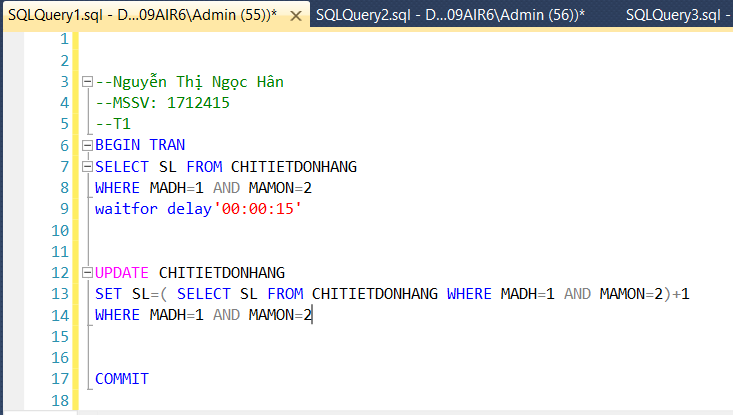
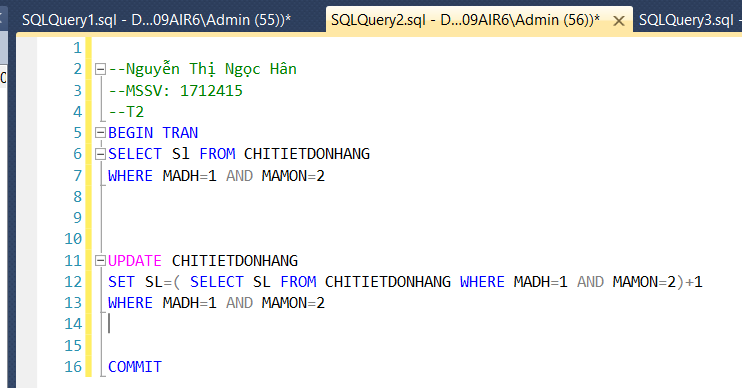


1. ***Lost Update:***
2. **Lỗi 4:** Gỉa sử khách hàng A (ban đầu đã mua món số 2) đặt thêm hàng món số 2 tại chi nhánh 1 với số lượng là 1, Lúc này có 2 nhân viên quản lí khách hàng truy cập vào tài khoản để cập nhật lại chi tiết đơn hàng. Nhân viên 1 tăng số lượng món lên 1, nhân viên 2 cũng tăng số lượng lên 1. Nhân viên 1 nếu lần sau vào xem sẽ thấy không giống thông tin mình đã sữa

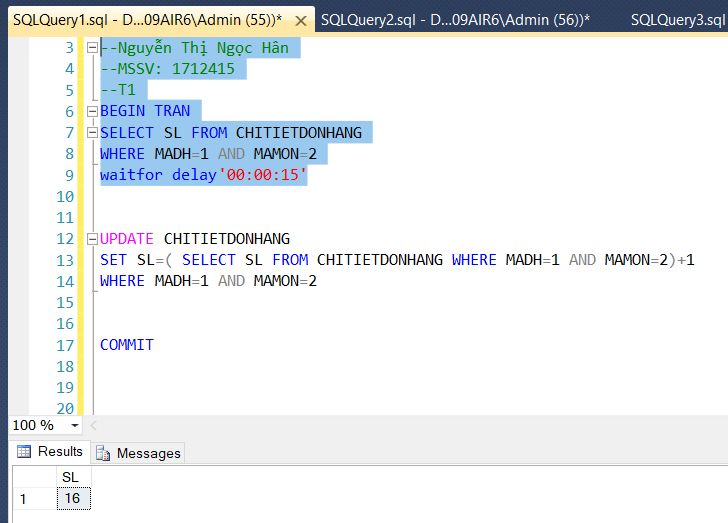
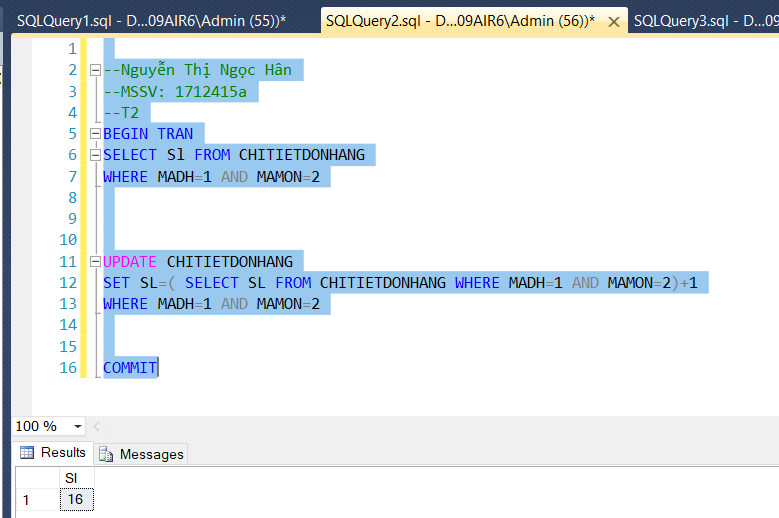
|  |  |
| --- | --- |
| T1 | T2 |
| BEGIN TRAN  SELECT SOLUONG FROM CHITIETDONHANG  WHERE MAHD=1 AND MAMON=2  waitfor delay‘00:00:05’  UPDATE DONHANGCHITIET SET SOLUONG=( SELECT SOLUONG FROM DOHANGCHITIET WHERE MADH=1 AND MAMON=2)+1  WHERE MADH=1 AND MAMON=2  COMMIT | BEGIN TRAN  SELECT SOLUONG FROM CHITIETDONHANG  WHERE MAHD=1 AND MAMON=2  UPDATE DONHANGCHITIET SET SOLUONG=( SELECT SOLUONG FROM DOHANGCHITIET WHERE MADH=1 AND MAMON=2)+1  WHERE MADH=1 AND MAMON=2  COMMIT |

1. Chạy lỗi trên SQL:

**Đoạn test case của trường hợp Lost Update: (Giả lập lấy cơ chế Read Committed)**

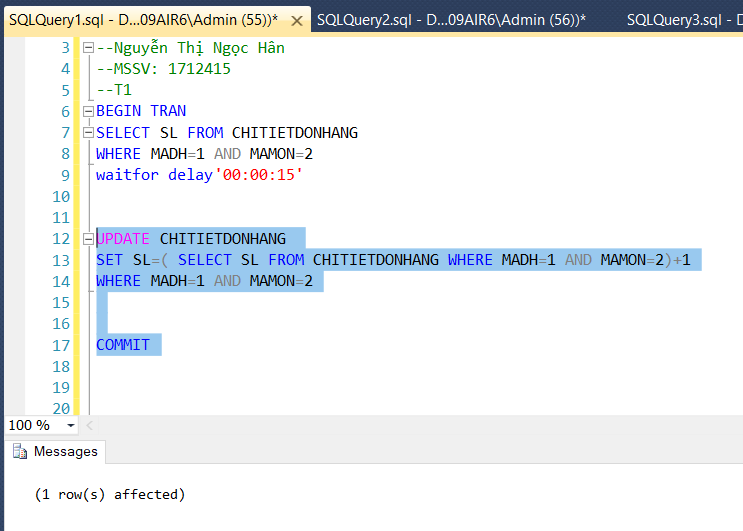
 

Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15 ’, sau đó chạy hết tác 2 và ta có kết quả:

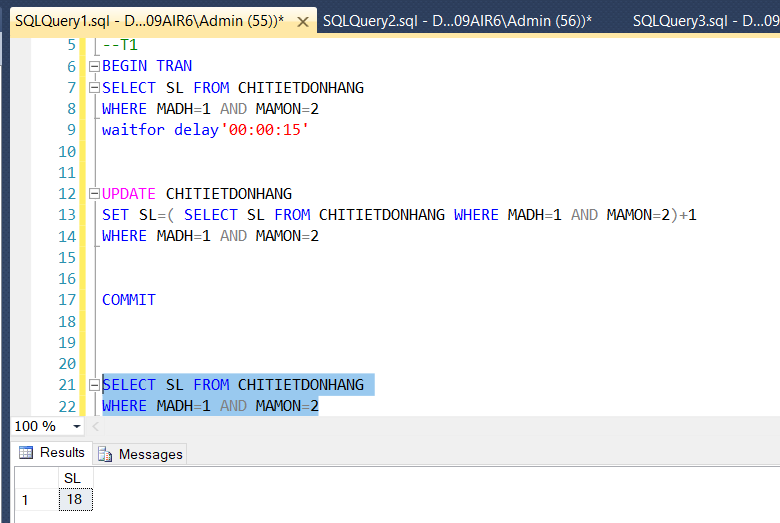
 

* 2 giao tác đều đọc được dữ liệu giống nhau, cũng là dữ liệu ban đầu.

Giao tác 1 cập nhật số lượng và comit:



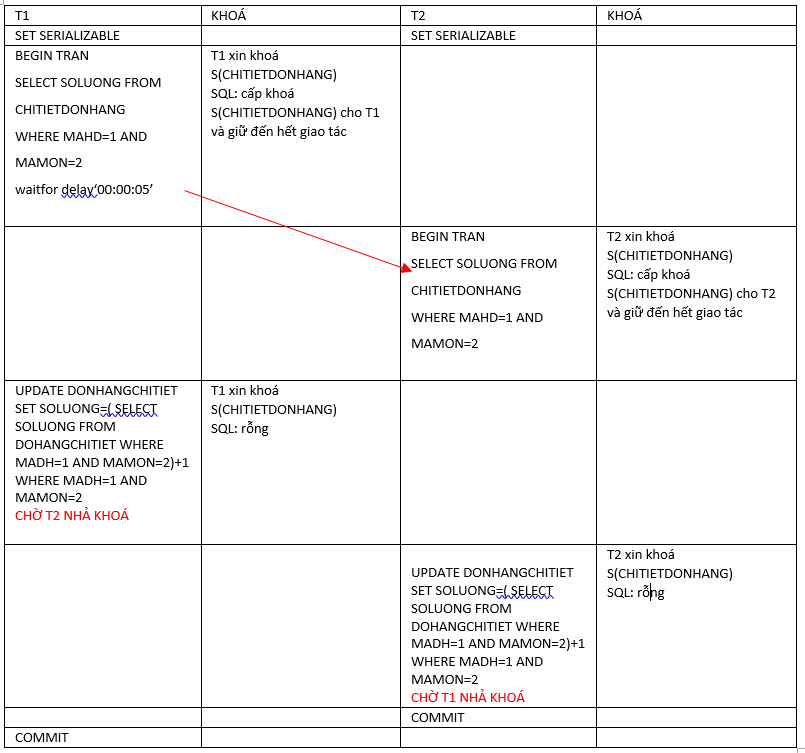
Select lại ta được kết quả được kết quả:



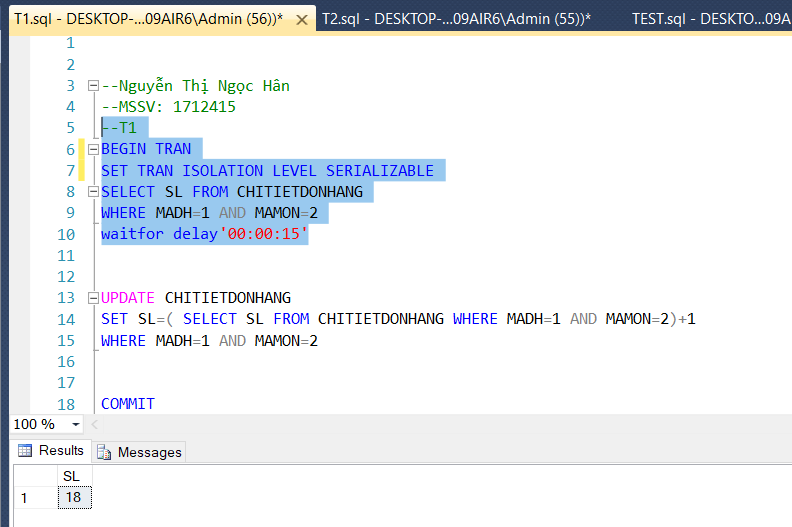
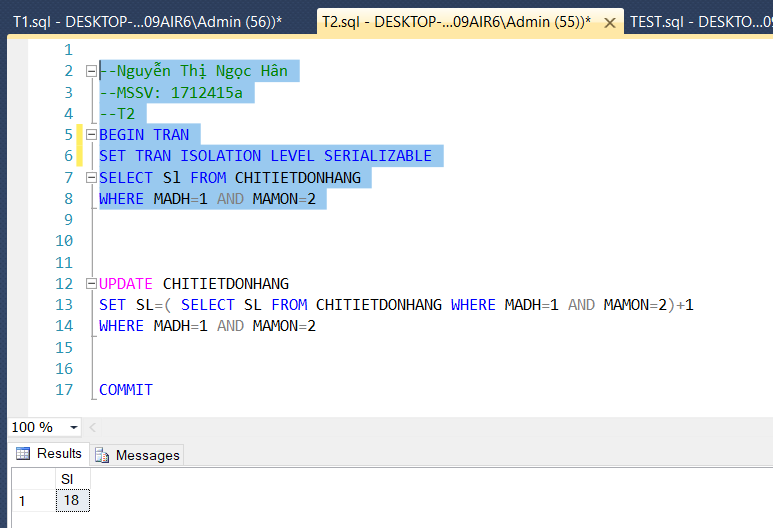
* Giao tác 2 đã cập nhật dữ liệu đè lên giao tác 1, nếu người dùng của giao tác 1 vào xem lại dữ liệu sẽ thấy có sự thay đổi.

1. Giải pháp: xét cơ chế khoá SERIALIZABLE nhưng tạo thành deadlock

Đồ thị ưu tiên:

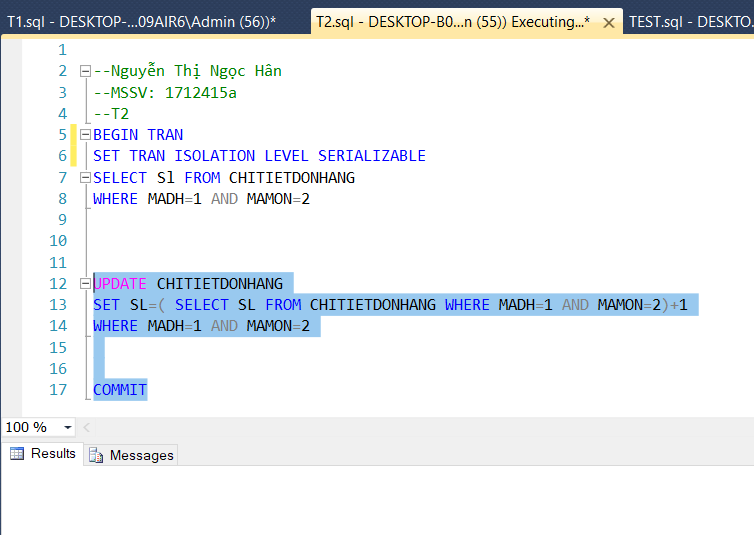


Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy giao tác 2 đến hết phần select ta có kết quả:

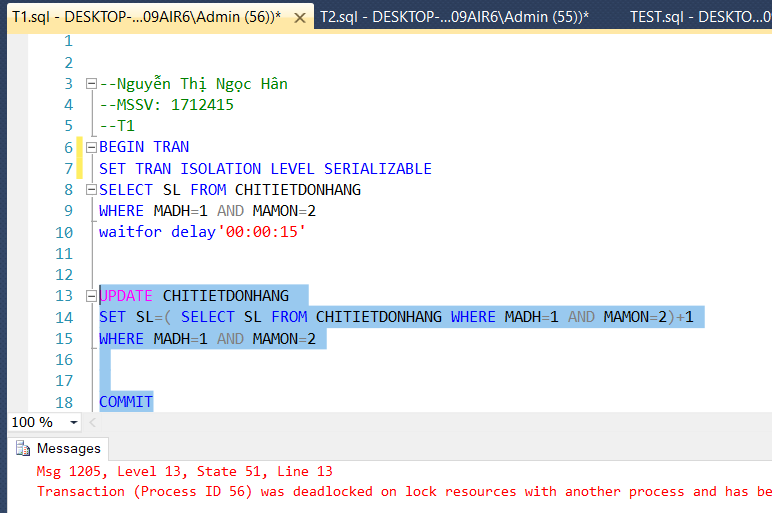
* T1 và T2 đều đọc được dữ liệu do

Chạy tiếp T2 cho đến hết ta có kết quả sau:



* T2 không cập nhật được dữ liệu do T1 đang SET mức cô lập SERIALIZABLE nên khoá S được giữ đến cuối giao tác, T2 muốn xin khoá X thì phải đợi T1 hoàn thành xong

Chạy hết T1 ta có kết quả sau:



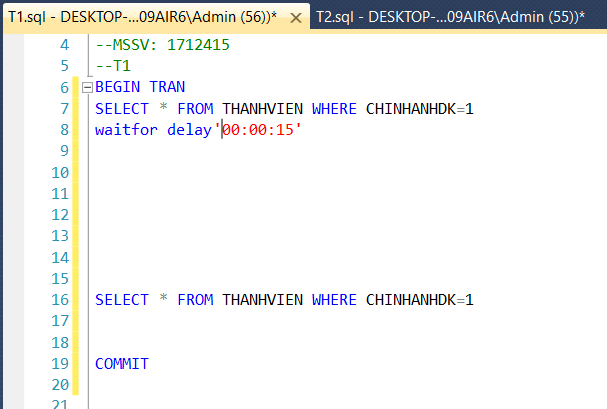
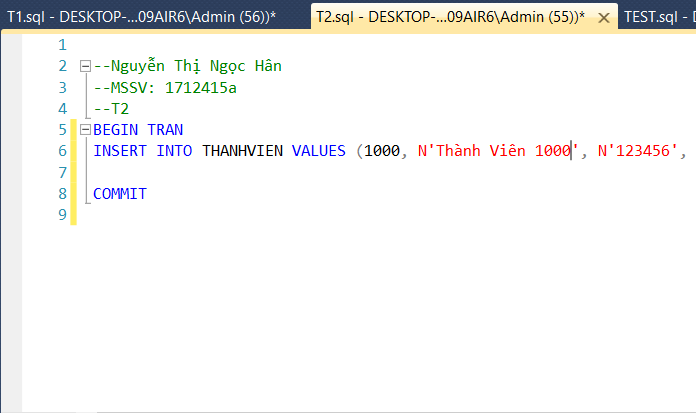
* Xảy ra deadlock do T1 không cập nhật được dữ liệu do T2 đang SET mức cô lập SERIALIZABLE nên khoá S được giữ đến cuối giao tác, T1 muốn xin khoá X thì phải đợi T2 hoàn thành xong
* T1 và T2 chờ nhau nên xảy ra deadlock

1. ***Phantom:***
2. **Lỗi 4:** quản lí chi nhánh thống kê những thành viên của mình, tại thời điểm đó thì nhân viên lễ tân lại chèn thêm 1 thành viên mới

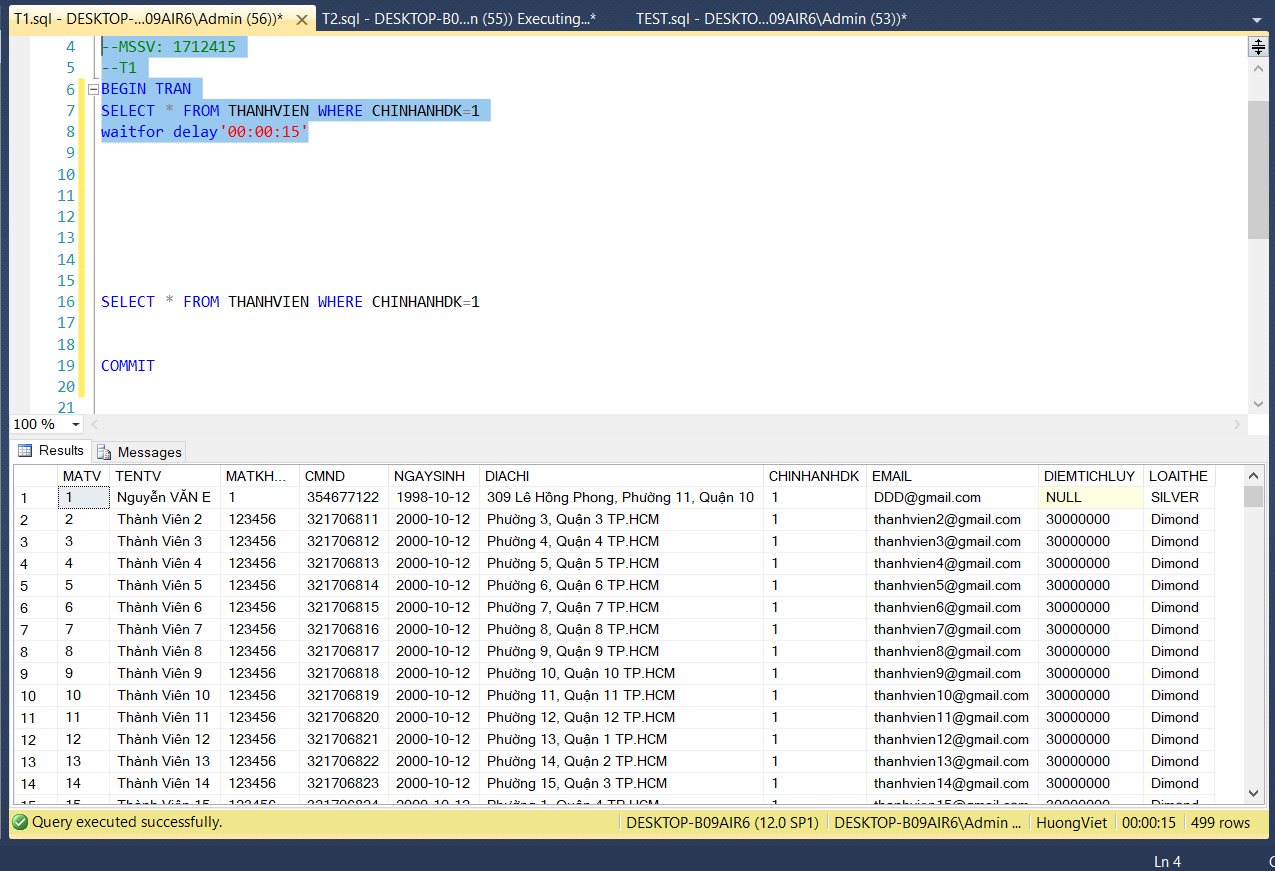
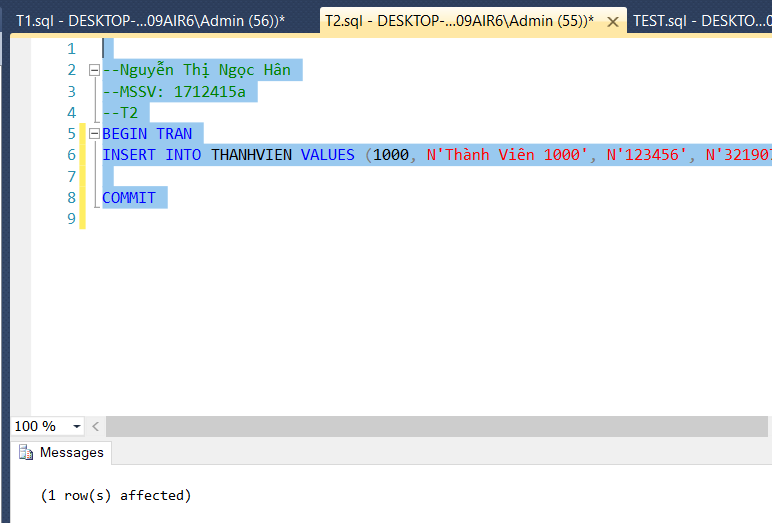
|  |  |
| --- | --- |
| T1 | T2 |
| BEGIN TRAN  SELECT \* FROM THANHVIEN WHERE CHINHANHDK=1  waitfor delay‘00:00:15’  SELECT \* FROM THANHVIEN WHERE CHINHANHDK=1  COMMIT | BEGIN TRAN  INSERT INTO THANHVIEN VALUES (…)  COMMIT |

1. Chạy lỗi trên SQL:

**Đoạn test case của trường hợp Phantom: (Giả lập lấy cơ chế Read Committed)**

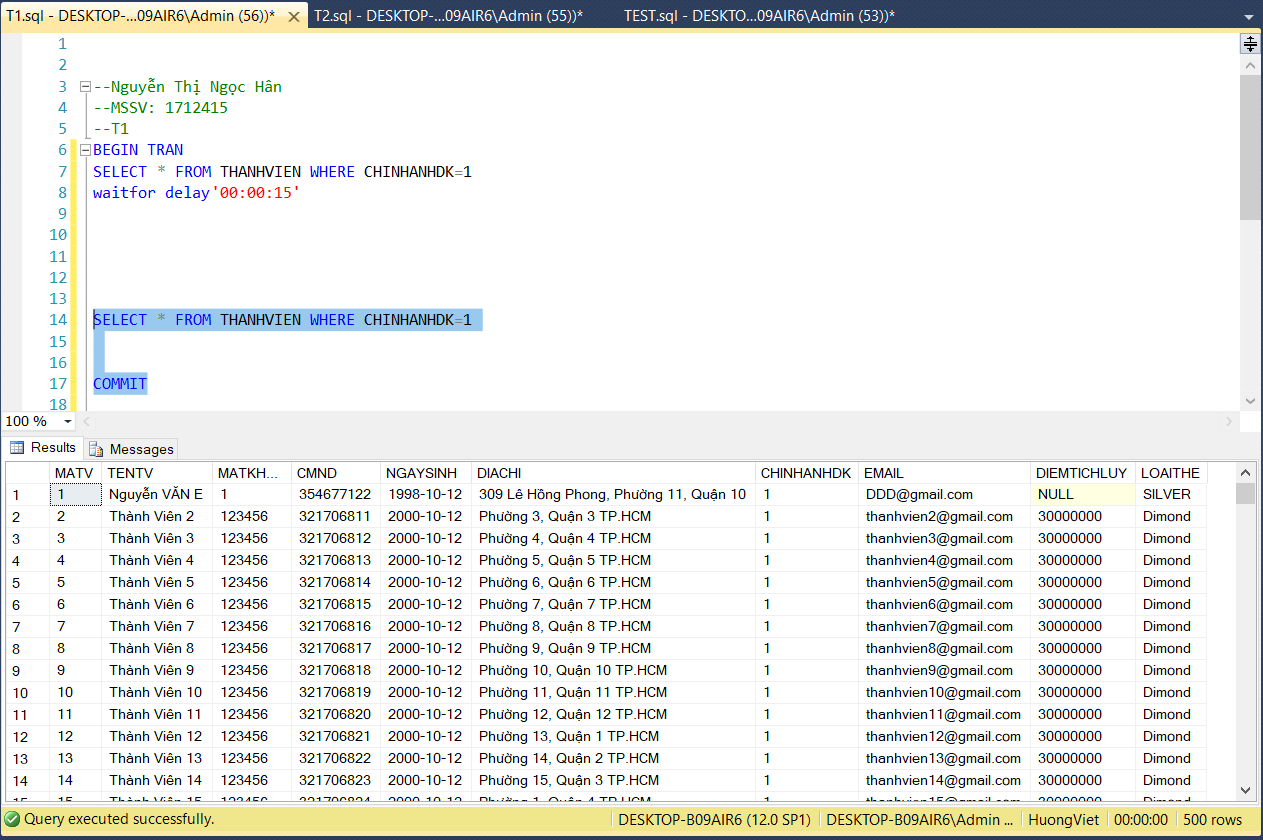
 

Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy hết tác 2 và ta có kết quả:

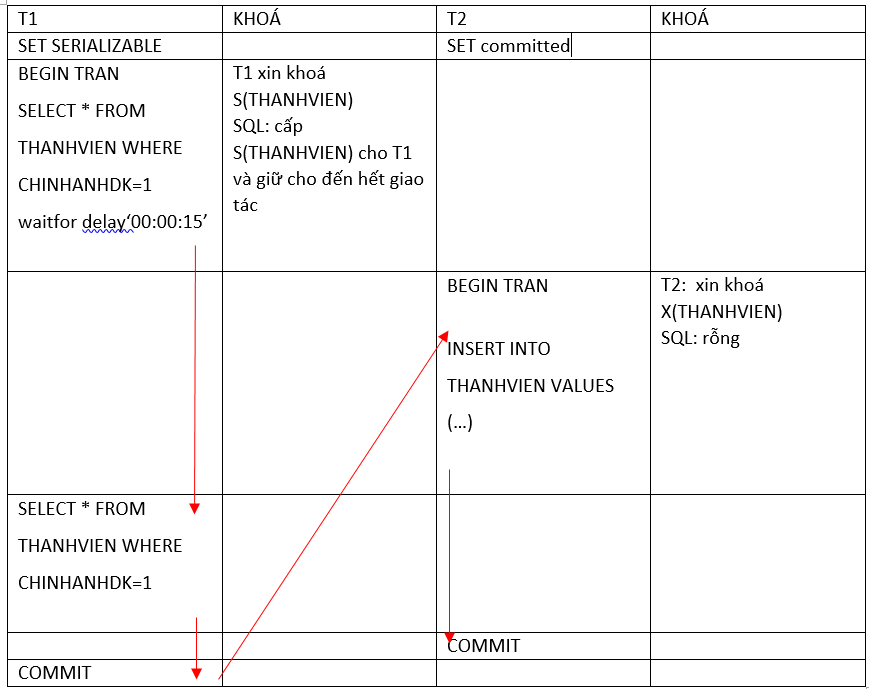
* Giao tác 1 xem Thành viên của chi nhánh 1 và nhận được dữ liệu ban đầu (không cập nhật/ thêm). Giao tác 2 đã thêm dữ liệu thành công.

Giao tác 1 tiến hành xem thành viên của chi nhánh 1 một lần nữa:

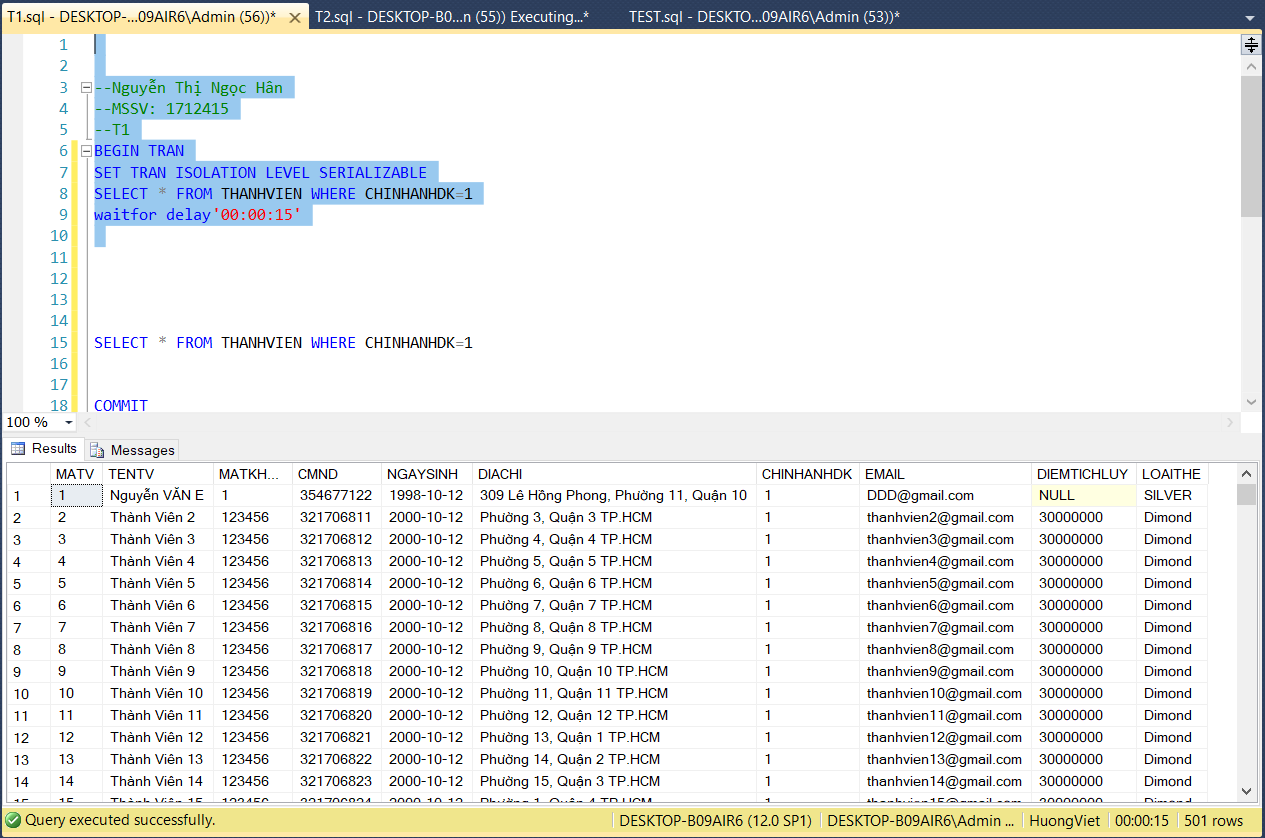
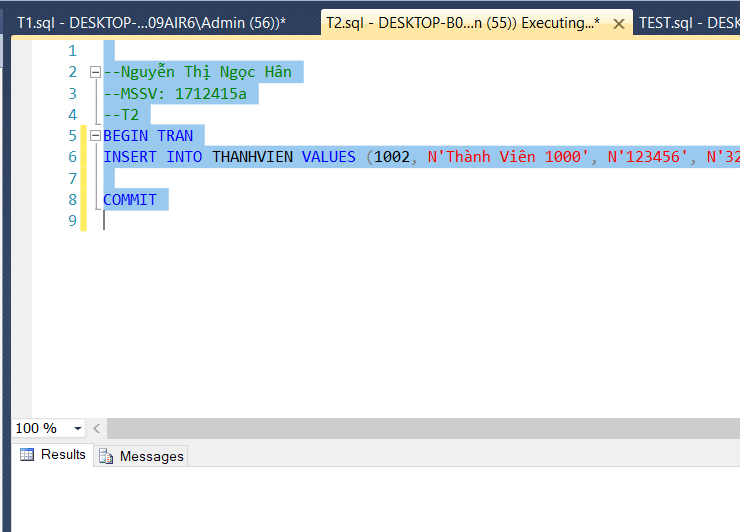


* Giao tác 1 đọc dữ liệu và thấy kết quả bị thêm 1 dòng.

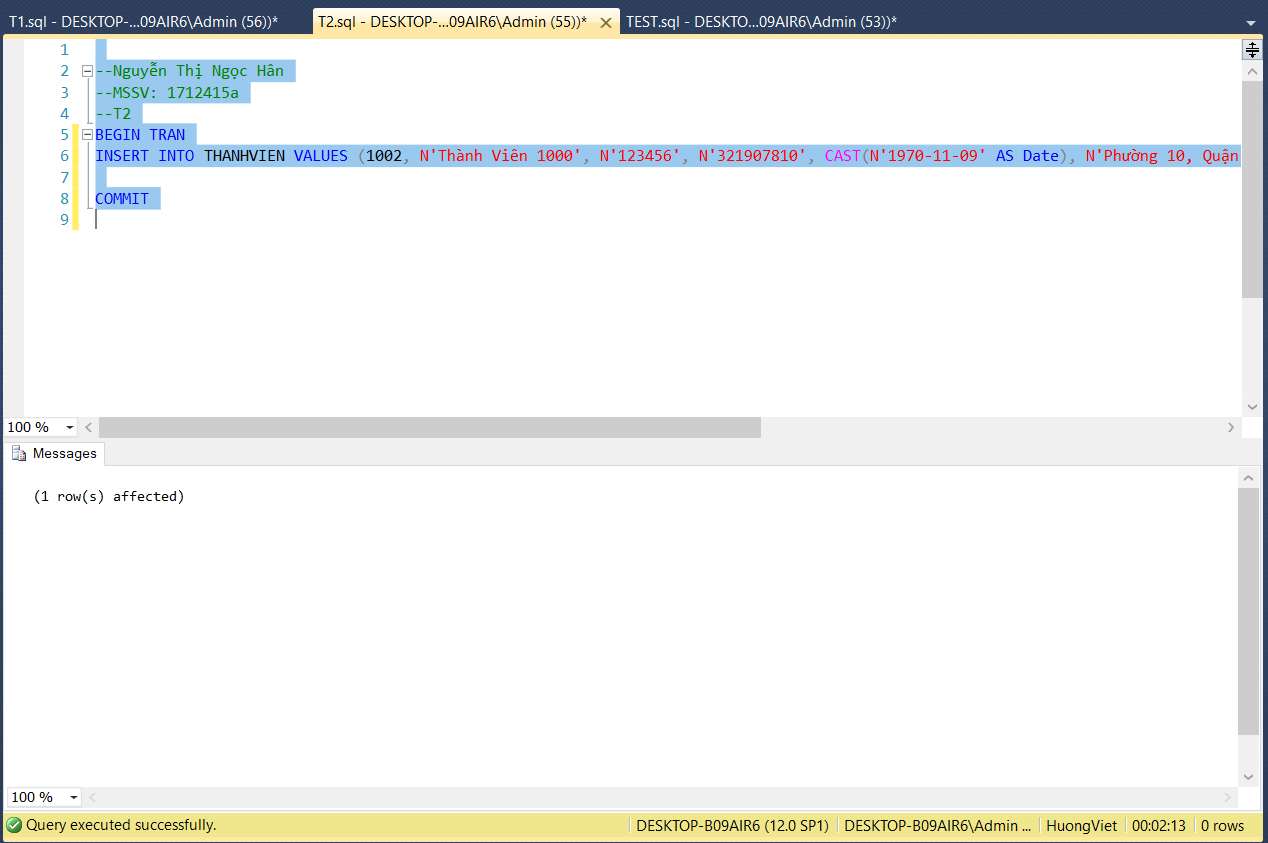
1. ***Giải pháp:*** Set cơ chế khóa Serializable



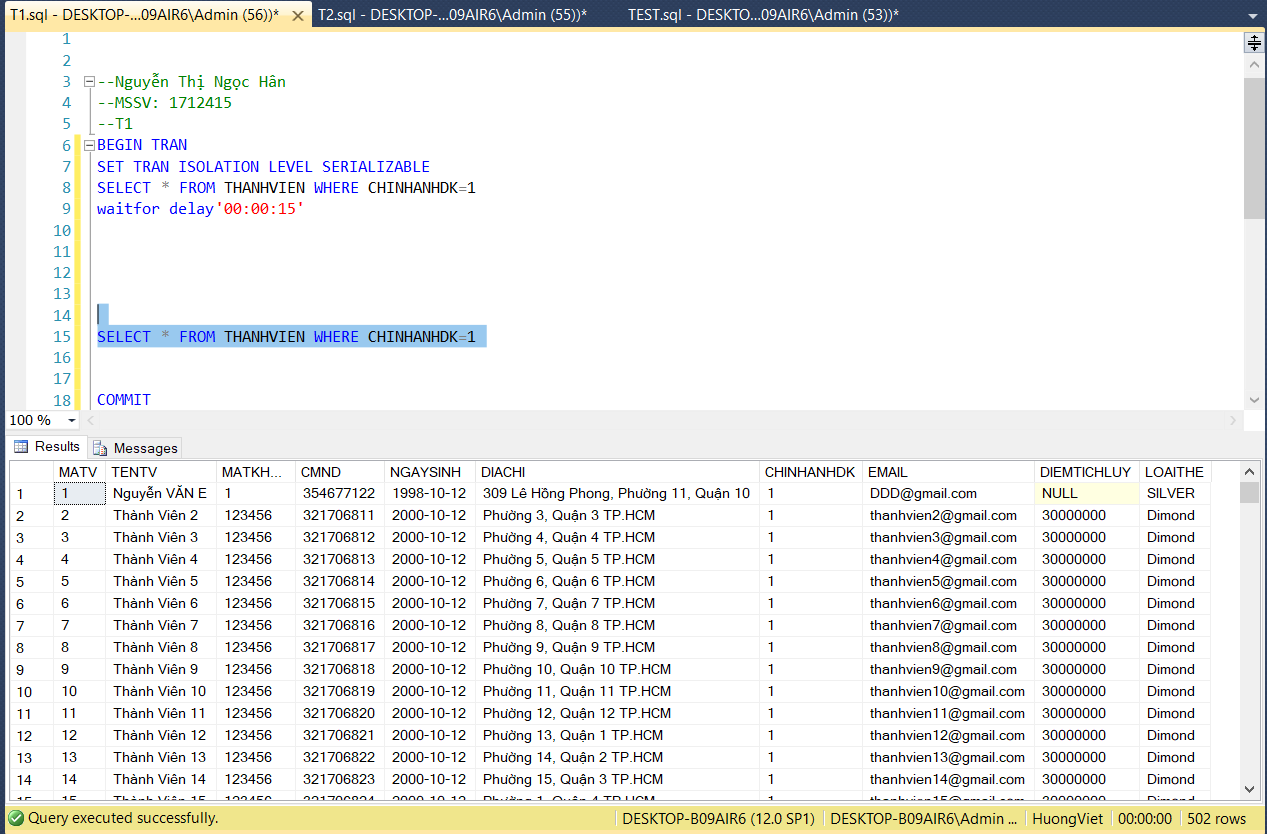
Chạy giao tác 1 đến lệnh waitfor delay ’00:00:15’, sau đó chạy hết giao tác 2, lúc này T2 không thêm được dữ liệu do T1 set mức cô lập Serializable. Khóa S đã được giữ cho đến hết giao tác, không cho insert dòng thỏa điều kiện thiết lập khóa, nên T2 muốn thêm dữ liệu thì phải đợi T1 commit.

Đây là kết quả sau khi T1 commit:



SELECT lại bảng thành viên ta được kết quả:



* T2 thêm dữ liệu thành công

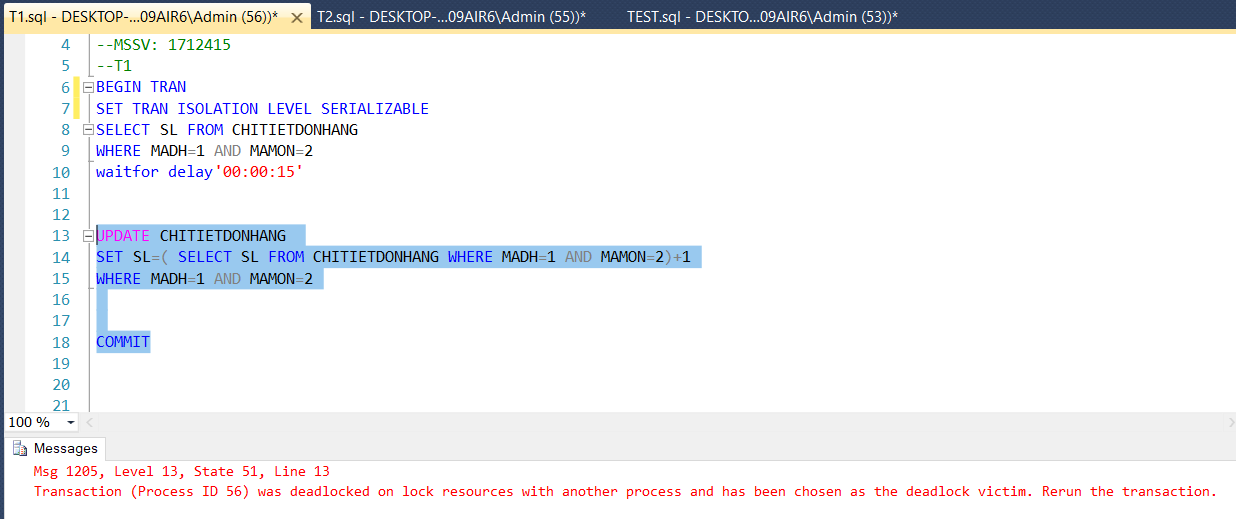
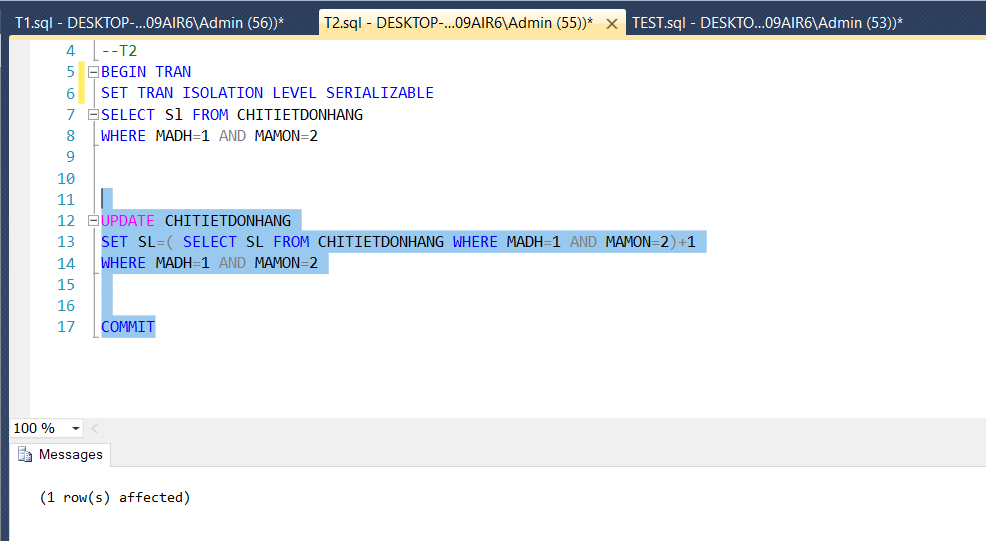
***5.Deadlock:***

1. **Lỗi 4:** Gỉa sử khách hàng A (ban đầu đã mua món số 2) đặt thêm hàng món số 2 tại chi nhánh 1 với số lượng là 1, Lúc này có 2 nhân viên quản lí khách hàng truy cập vào tài khoản để cập nhật lại chi tiết đơn hàng. Nhân viên 1 tăng số lượng món lên 1, nhân viên 2 cũng tăng số lượng lên 1. Nhân viên 1 nếu lần sau vào xem sẽ thấy không giống thông tin mình đã sữa. để xảy ra deadlock ta xét mức cô lập SERIALAZABLE

|  |  |
| --- | --- |
| T1 | T2 |
| BEGIN TRAN  SET TRAN ISOLATION LEVEL SERIALIZABLE  SELECT SOLUONG FROM CHITIETDONHANG  WHERE MAHD=1 AND MAMON=2  waitfor delay‘00:00:05’  UPDATE DONHANGCHITIET SET SOLUONG=( SELECT SOLUONG FROM DOHANGCHITIET WHERE MADH=1 AND MAMON=2)+1  WHERE MADH=1 AND MAMON=2  COMMIT | BEGIN TRAN  SET TRAN ISOLATION LEVEL SERIALIZABLE  SELECT SOLUONG FROM CHITIETDONHANG  WHERE MAHD=1 AND MAMON=2  UPDATE DONHANGCHITIET SET SOLUONG=( SELECT SOLUONG FROM DOHANGCHITIET WHERE MADH=1 AND MAMON=2)+1  WHERE MADH=1 AND MAMON=2  COMMIT |

1. Chạy trên SQL

Thực hiện tương tự câu lỗi lost update ta dược kết quả sau:

* Xảy ra lỗi deadlock

1. Giải pháp để giải quyết deadlock ta huỷ 1 giao tác chạy 1 giao t, sau đó thực hiện lại giao tác bị huỷ. Ở đây thì SQL đã tự động huỷ T1 nên T2 update dữ liệu thành công, T1 bị huỷ.