BÁO CÁO ĐỒ ÁN THỰC HÀNH

MÔN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

GVHD: HỒ THỊ HOÀNG VY

THÔNG TIN NHÓM

STT	MSSV	Họ tên	Công việc	% Hoàn thành
1	20120049	Nguyễn Hải Đăng	Lost Update + Phantom	100%
2	20120269	Võ Văn Minh Đoàn	Dirty Read + Conversation Deadlock	100%
3	20120592	Lê Minh Tiến	Dirty Read + Unrepeatable Read	100%
4	20120624	Mai Quyết Vang	Unrepeatable Read + Cycle Deadlock	100%

XỬ LÝ TÌNH HUỐNG TRANH CHẤP

I. Sinh viên thực hiện: 20120269 - Võ Văn Minh Đoàn

1. Tình huống 1: khách hàng (thêm khách hàng), quản trị (xem khách hàng) dẫn đến dirty read.

Giải pháp: đổi mức cô lập từ read uncommitted thành read committed ở 2 giao tác.

ERR01: ***Dirty read***

T1 (User = KhachHang): đăng ký thêm 1 khách hàng.

T2 (User = QuanTri): xem thông tin 1 khách hàng có mã khách hàng trùng với input của T1.

themKhachHang	**Khóa**	**timKiemKhachHang**	**Khóa**
Input: mã khách hàng, họ tên, địa chỉ, số điện thoại, email ***Output: thêm khách hàng trên***		***Input: mã khách hàng*** ***Output: thông tin khách hàng***	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRAN			
B1: Kiểm tra thông tin có rỗng hay không if (@MaKH='' or @HoTen='' or @DiaChi='' or @SDT='' or @Email='')			

begin print N'Thông tin trống' rollback tran return 1 end			
B2: Kiểm tra thông tin mã khách hàng có tồn tại hay chưa if exists(select* from KHACHHANG where MaKH=@MaKH) begin print N'Mã khách hàng đã tồn tại' rollback tran return 1 end	**S(KHACHHANG)** //T1 xin khóa S trên KHACHHANG và được phát khóa		
B3: Thêm thông tin vào bảng KHACHHANG insert into KHACHHANG values(@MaKH,@HoTen,@DiaChi,@SDT,@Email)	**X(KHACHHANG)** //T1 xin khóa X trên KHACHHANG		

	và được phát khóa. T1 giữ khóa đến hết giao tác		
WAITFOR DELAY '00:00:10'			
		BEGIN TRAN	
		B1: Kiểm tra thông tin mã khách hàng có tồn tại hay không if not exists(select* from KHACHHANG where MaKH=@MaKH) begin print @MaKH + N' không tồn tại!' rollback tran return 1 end	**S(KHACHHANG)** **SQL không cấp khóa** **T2 chờ** //T2 xin khóa S trên bảng KHACHHANG nhưng SQL Server không cấp khóa do T1 đang giữ khóa X trên KHACHHANG
B4: Kiểm tra thông tin số điện thoại có bị trùng không	**X(KHACHHANG)**		

if exists(select * from KHACHHANG where SDT = @SDT and MaKH != @MaKH)
 	
begin
 	
print N'Số điện thoại bị trùng!'
 	
ROLLBACK TRAN
 	
return 1
 	
end
 	
//Khi insert đến bảng KHACHHANG bị lỗi: trùng số điện thoại nên dữ liệu bị rollback. Vì vậy dữ liệu được T2 đọc trước đó là dữ liệu rác.
	//T1 còn giữ khóa X đến hết giao tác		

ROLLBACK			
		T2 tiếp tục thực hiện B1 do T1 đã rollback và nhả khóa X trên KHACHHANG	**S(KHACHHANG)** **SQL phát khóa**
		B2: Xem thông tin khách hàng SELECT * FROM KHACHHANG WHERE MaKH = @MaKH	**S(KHACHHANG)**

			SQL phát khóa
		COMMIT	
Do đổi mức cô lập từ read uncommitted sang read committed nên khi T2 yêu cầu khóa S trên KHACHHANG thì không được cấp và phải chờ T1 commit/rollback thì T2 mới được phát khóa. Vì vậy, T2 không thể đọc dữ liệu khi T1 chưa commit/rollback nên không xảy ra dirty read.			

2. Tình huống 2: Quản trị 1 (cập nhật thông tin đối tác), quản trị 2 (xóa đối tác) dẫn đến conversion deadlock.

Giải pháp: thiết lập update lock thay vì shared lock khi thực hiện thao tác select ở 2 giao tác.

ERR05: ***Conversion deadlock***

T1 (User = QuanTri1): sửa thông tin của 1 đối tác.

T2 (User = QuanTri2): xóa 1 đối tác có mã đối tác là đối tác mà T1 đang cập nhật.

capNhatDoiTac	**Khóa**	**xoaDoiTac**	**Khóa**
***Input*: mã đối tác, email, người đại diện, số lượng chi nhánh, tên quán, loại thực phẩm** ***Output*: sửa thông tin đối tác như trên**		***Input*: mã đối tác** ***Output*: xóa đối tác như trên**	
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	
BEGIN TRAN			

B1: Kiểm tra thông tin có rỗng hay không
 	
if (@MaDT='' or @Email='' or @NgDaiDien='' or @TenQuan='' or @LoaiTP='')
 	
begin
 	
print N'Thông tin trống'
 	
rollback tran
 	
return 1
 	
end			

B2: Kiểm tra thông tin mã đối tác có tồn tại hay chưa if not exists(select* from DOITAC with (updlock) where MaDT=@MaDT) begin print N'Mã đối tác chưa tồn tại' rollback tran return 1 end	**U(DOITAC)** //T1 xin khoá Update trên bảng DOITAC và được cấp khóa.		

WAITFOR DELAY '00:00:10'			
		BEGIN TRAN	

		B1: Kiểm tra thông tin có rỗng hay không
		if (@MaDT='')
		begin
		print N'Thông tin nhập không được rỗng'
		rollback tran
		return 1
		end	

		B2: Kiểm tra thông tin mã đối tác có tồn tại hay không if not exists(select* from DOITAC with (updlock) where MaDT=@MaDT) begin print N'Không thể xóa, đối tác không tồn tại' rollback tran return 1	**U(DOITAC)** **SQL không phát khóa** **T2 chờ** //T2 xin khoá Update trên bảng DOITAC và không được phát khóa do T1 đang giữ

		end	Update lock trên DOITAC.
B3: Cập nhật thông tin đối tác update DOITAC set Email = @Email, NgDaiDien = @NgDaiDien, SLChiNhanh = @SLChiNhanh, TenQuan = @TenQuan, LoaiTP = @LoaiTP where MaDT = @MaDT	**X(DOITAC)** //T1 xin khóa X trên bảng DOITAC và được phát khóa.		
COMMIT			
		T2 tiếp tục thực hiện B2 do T1 đã kết thúc giao tác và nhả khóa trên DOITAC	**U(DOITAC)** //T2 xin khoá Update trên bảng DOITAC và được phát khóa.
		B3: Xóa đối tác	**X(DOITAC)**

		delete from DOITAC where MaDT = @MaDT	//T2 xin khóa X trên bảng DOITAC và được phát khóa.
		COMMIT	
Do thiết lập khóa update khi đọc trên bảng DOITAC ở cả 2 giao tác nên khi T2 yêu cầu đọc trên DOITAC ở B1 không được phát khóa và phải chờ T1 nhả khóa. Sau khi T1 đọc xong lại chuyển sang thao tác ghi nên khóa Update được chuyển sang khóa X và T2 phải chờ T1 commit/rollback mới tiếp tục thực hiện. Vì vậy, khóa Update đã giải quyết tình huống tranh chấp conversion deadlock.			

II. Sinh viên thực hiện: 20120049 - Nguyễn Hải Đăng

3. Tình huống 1: KHACHHANG tra cứu (select) bảng DOITAC trong khi DOITAC đang đăng ký thông tin (insert) dẫn đến tranh chấp Phantom.

Giải pháp: đổi mức cô lập từ read committed thành repeatable read ở giao tác T1.

ERR01: *Phantom*			
T1 (User = KHACHHANG): thực hiện tra cứu thông tin các đối tác. T2 (User = DOITAC): thực hiện đăng ký thông tin.			
sp_DSDoiTac	**Khóa**	**sp_ThemDoiTac**	**Khóa**
Input**: Không có tham số đầu vào* ***Output: Danh sách đối tác		***Input***: Mã đối tác, email, người đại diện, số lượng chi nhánh, tên quán, loại thành phố	

		Output:	
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRAN			
B1: KHACHHANG select bảng DOITAC để xem thông tin các đối tác SELECT * FROM DOITAC	**T1: Xin khóa S trên bảng DOITAC** **SQL : Cấp khóa S** **T1: giữ khóa S đến hết giao tác** **Ngăn chèn dữ liệu vào tập đang khóa**		
WAITFOR DELAY '00:00:10'			
		BEGIN TRAN	
		B1: Kiểm tra thông tin có trống hay không	

		if (@MaDT='' or @Email='' or @NgDaiDien='' or @SLChiNhanh='' or @TenQuan='')
		begin
		print N'Thông tin trống'
		rollback tran
		return 1
		end	

		B2: Kiểm tra mã đối tác đã tồn tại chư if exists(select* from DOITAC where MaDT = @MaDT) begin print N'Mã đối tác đã tồn tại' rollback tran return 1 end	**T2: Xin khóa X** **SQL : Không cấp khóa X do T1 ngăn không cho insert vào**
		B3: Thêm đối tác	**T2: Xin khóa X**

		insert into DOITAC values (@MaDT, @Email, @NgDaiDien, @SLChiNhanh, @TenQuan, @LoaiTP)	**SQL : Không cấp khóa X do T1 ngăn không cho insert vào**
SELECT * FROM DOITAC	**T1 : Trả khóa S**	COMMIT	
COMMIT			
Do T1 đang select dữ liệu bảng DOITAC thì T2 đang insert 1 đối tác vô bảng DOITAC nên xảy ra tình huống 2 lần select cho kết quả khác nhau. Để giải quyết, ta sửa mức cô lập từ read commited thành serializable ở T1, lúc đó T1 được cấp khóa X và giữ khóa cho đến hết giao tác, đồng thời ngăn chặn lệnh insert giữ liệu thỏa điều kiện ở giao tác T2 => phantom được giải quyết.			

4. Tình huống 2: KHACHHANG hủy đơn trong khi DOITAC chuyển tình trạng đơn hàng sang "Đã tiếp nhận" (không thể hủy), khi đó xảy ra tranh chấp Lost Update.

Giải quyết: thay đổi mức cô lập ở cả 2 giao tác từ read commited sang repeatable read

ERR02: *Lost Update*			
T1 (User = KHACHHANG): khách hàng hủy đơn đặt hàng. T2 (User = DOITAC): đối tác chuyển tình trạng đơn hàng sang "Đã tiếp nhận".			
sp_KhachHangHuyDon	**Khóa**	**sp_DoiTacCapNhatTinhTrangDon**	**Khóa**
Input: Mã đơn hàng		***Input***: Mã đơn hàng, tình trạng đơn	

Output: Đơn hàng đó sẽ có thuộc tính TinhTrang là "Đã hủy đơn"		***Output***: Đơn hàng đó có thuộc sẽ có thuộc tính TinhTrang = input tình trạng đơn	
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ		SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	
BEGIN TRAN			

B1: Kiểm tra thông tin có bị trống hay không
if (@MaDonDH='' or @TinhTrang='')
begin
print N'Thông tin trống'
rollback tran
return 1
end			

B2: Kiểm tra đơn hàng có tồn tại không if not exists (select* from DONDATHANG where MaDH = @MaDonDH) begin print N'Mã đơn đặt hàng không tồn tại'	**T1: Xin khóa S trên bảng DONDATHANG** **SQL : Cấp khóa S**		

rollback tran return 1 end	**DONDATHANG** **T1 : giữ khóa S đến hết giao tác T1** **DONDATHANG**		
waitfor delay '0:0:5'			
		BEGIN TRAN	

		B1: Kiểm tra thông tin có bị trống hay không
		if (@MaDonDH='')
		begin
		print N'Thông tin trống'
		rollback tran
		return 1
		end	

		B2: Kiểm tra mã đơn hàng có tồn tại không if not exists (select* from DONDATHANG where MaDH = @MaDonDH) begin print N'Mã đơn đặt hàng không tồn tại' rollback tran return 1 end	**T2 : Xin khóa X trên bảng DONDATHANG** **SQL : Không cấp khóa X do T1 đang giữ khóa S bảng DONDATHANG**
		waitfor delay '0:0:5'	
B3: Update tình trạng "Đã hủy đơn" update DONDATHANG SET TinhTrang = N'Đã hủy đơn' where MaDH = @MaDonDH	**T1 : Trả khóa S**	B3: Update tình trạng hiện tại của đơn hàng update DONDATHANG SET TinhTrang = @TinhTrang where MaDH = @MaDonDH	
commit		commit	
Vì 2 transaction đều được cài đặt mức cô lập repeatable read, khi T1 đang được cấp khóa S trên bảng DONDATHANG nên không thể thiết lập khóa X trên đơn vị dữ liệu đang có khóa S, do đó T2 không được cấp khóa X, vì vậy T2 không thể update dữ liệu tình trạng cho bảng DONDATHANG. Do đó dẫn tới deadlock vì tranh chấp khóa => Lost Update được giải quyết.			

III. Sinh viên thực hiện: 20120592- Lê Minh Tiến

5. Tình huống 1: Đối tác đang cập nhật giá cho 1 thực phẩm, thì khách hàng tìm kiếm thông tin thực phẩm đó, nhưng do giá đối tác nhập không hợp lệ (vô tình nhập số âm) dẫn đến Dirty read

Giải pháp: đổi mức cô lập từ read uncommitted thành read committed ở 2 giao tác.

ERR01: *Dirty read*

T1 (User = Đối tác): Cập nhật giá cho một loại thực phẩm

T2 (User = Khách hàng): Tìm kiếm thông tin thực phẩm T1 đang cập nhật.

sp_CapNhatGiaTP	**Khóa**	**sp_TimKiemThucPham**	**Khóa**
Input**: MaTP, MaDT, Gia (1 số âm)* ***Output: Cập nhật thành công		***Input***: MaTP, MaDT ***Output***: Thông tin thực phẩm	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRAN			
B1: Kiểm tra thông tin thực phẩm if not exists (select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin	**S(ThucPham)** //T1 xin khóa S trên bảng ThucPham và được cấp		

print N'Thực phẩm này không tồn tại' rollback tran return 1 end			
B2: Update giá cho thực phẩm update ThucPham set Gia=@Gia where MaTP=@MaTP and MaDT=@MaDT	**X(ThucPham)** //T1 xin khóa X trên bảng ThucPham và được cấp. T1 giữ khóa đến hết giao tác.		
WAITFOR DELAY '00:00:10'			
		BEGIN TRAN	
		B1: Kiểm tra thông tin thực phẩm có tồn tại if not exists (select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin	**S(ThucPham)** **SQL không cấp khóa** **T2 chờ**

		print N'Thực phẩm này không tồn tại' rollback tran return 1 end	//T2 xin khóa S trên bảng ThucPham nhưng SQL Server không cấp khóa do T1 đang giữ khóa X trên bảng ThucPham

B3: Kiểm tra giá có hợp lệ
if @Gia<0
begin
print N'Giá không hợp lệ'
rollback tran
return 1
end
//Khi update do giá không hợp lệ nên dữ liệu rollback.
	X(ThucPham)
	//T1 còn giữ khóa X đến hết giao tác		

ROLLBACK			
		B1: Kiểm tra thông tin thực phẩm có tồn tại if not exists (select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin print N'Thực phẩm này không tồn tại' rollback tran return 1 end	**S(ThucPham)** **SQL cấp khóa** //T2 xin khóa S trên bảng ThucPham và được cấp do lúc này T1 đã trả khóa.
		B2: Đọc thông tin thực phẩm select *from ThucPham where MaTP=@MaTP and MaDT=@MaDT	**S(ThucPham)** **SQL cấp khóa** //T2 xin khóa S trên bảng ThucPham và được cấp.
		COMMIT	

Do đổi mức cô lập từ read uncommitted sang read committed nên khi T2 yêu cầu khóa S trên ThucPham thì không được cấp và phải chờ T1 commit/rollback thì T2 mới được phát khóa. Vì vậy, T2 không thể đọc dữ liệu khi T1 chưa commit/rollback nên không xảy ra dirty read.

6. Tình huống 2: khách hàng đang tìm kiếm thực phẩm với một tình trạng cụ thể thì đối tác cập nhật tình trạng thực phẩm dẫn đến Unrepeatable Read.

Giải quyết: thay đổi mức cô lập ở cả 2 giao tác từ read uncommited sang repeatable read.

ERR02: Unrepeatable Read

T1 (User = khách hàng): tìm kiếm (đọc) thông tin 1 loại thực phẩm với trạng thái cụ thể.

T2 (User =đối tác): cập nhật tình trạng thực phẩm mà T1 đang tìm kiếm.

sp_TimKiemThucPhamVoiTinhTrang	Khóa	sp_CapNhatTinhTrangTP	Khóa
***Input**: mã thực phẩm, mã đối tác, tình trạng* ***Output**: thông tin thực phẩm*		***Input**: mã thực phẩm, mã đối tác, tình trạng* ***Output**: cập nhật tình trạng thực phẩm thành công*	
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ		SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	
BEGIN TRAN			
B1: Kiểm tra thực phẩm có tồn tại không	S(ThucPham)		

if not exists(select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin print N'Thực phẩm này không tồn tại' rollback tran return 1 end	//T1 xin khóa S và được cấp. T1 giữ khóa S đến hết giao tác.		
B2: Kiểm tra thực phẩm có tình trạng cần tìm không if not exists(select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT and TinhTrang=@TinhTrang) begin print N'Thực phẩm không có tình trạng này' rollback tran	S(ThucPham) //T1 vẫn giữ khóa S trên bảng ThucPham		

return 1 end			
WAITFOR DELAY '00:00:10'			
		BEGIN TRAN	
		B1: Kiểm tra thực phẩm có tồn tại không if not exists(select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin print N'Thực phẩm này không tồn tại' rollback tran return 1 end	**S(ThucPham)** //T2 xin khóa S và đc cấp

		B2: Kiểm tra tình trạng mới có trùng tình trạng cũ if @TinhTrang=(select TinhTrang from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin print N'Tình trạng mới giống tình trạng cũ' rollback tran return 1 end	**S(ThucPham)** //T2 xin khóa S và đc cấp
		B3: Cập nhật tình trạng thực phẩm update ThucPham set TinhTrang=@TinhTrang where MaTP=@MaTP and MaDT=@MaDT	**X(ThucPham)** //T2 xin khóa X trên bảng ThucPham nhưng không được cấp do

			T1 đang giữ khóa S
select *from ThucPham where MaTP=@MaTP and MaDT=@MaDT and TinhTrang=@TinhTrang	**S(ThucPham)** //T1 vẫn giữ khóa S trên bảng ThucPham đến hết giao tác		
COMMIT			
		Thực hiện B3. B3: Cập nhật tình trạng thực phẩm update ThucPham set TinhTrang=@TinhTrang where MaTP=@MaTP and MaDT=@MaDT	**X(ThucPham)** //T2 được cấp khóa do T1 đã thực hiện hết giao tác.
		COMMIT	

Do đổi mức cô lập từ read uncommitted sang repeatable read nên khi T2 yêu cầu khóa S trên ThucPham thì không được cấp và phải chờ T1 commit thì T2 mới được phát khóa. Sau khi T1 đọc xong 2 lần T2 mới được cấp khóa S nên dữ liệu ở 2 lần đọc của T1 là không khác nhau. Vì vậy, không xảy ra repeatable read.

IV. Sinh viên thực hiện: 20120624 - Mai Quyết Vang

7. Tình huống 1: khách hàng đang kiểm tra đơn đặt hàng thì đối tác cập nhật thông tin trạng thái giao hàng Unrepeatable Read

Giải quyết: thay đổi mức cô lập ở cả 2 giao tác từ read uncommited sang repeatable read.

ERR01: *Unrepeatable Read*

T1 (User = khách hàng): thực hiện kiểm tra trạng thái đơn hàng đã đặt.

T2 (User = tài xế): thực hiện cập nhật trạng thái đặt hàng

sp_KiemTraDonHang	**Khóa**	**sp_CapNhatTinhTrangGiao**	**Khóa**
Input: *MaDH.* ***Output***: Tình trạng đơn hàng		***Input***: *MaDH, TinhTrang* ***Output***: Cập nhật tình trạng đơn hàng thành công	
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ		SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	
BEGIN TRAN			

B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONDATHANG A WHERE A.MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END	**R(DONDATHANG)** //T1 xin khóa S và được cấp. T1 giữ khóa S đến hết giao tác.		
IF N'Thành công' = (SELECT TinhTrang FROM DONDATHANG A WHERE MaDH=@MaDH) BEGIN print N'Đơn hàng giao thành công' return 0 END	**R(DONDATHANG)** //T1 vẫn giữ khóa S trên bảng ThucPham		
WAITFOR DELAY '00:00:20'			
		BEGIN TRAN	

		B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONDATHANG WHERE MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END IF @TinhTrang <> N'Đang xử lý' or @TinhTrang <>N'Thành công' BEGIN print N'Tình trạng không hợp lệ' rollback tran return 1 END	**R(DONDATHANG)** //T1 xin khóa S và không được cấp do T1 đang giữ
print N'Đơn hàng đang được giao' return 1	**R(ThucPham ThucPham)**		

	//Không cần xin khoá		
COMMIT			
		B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONDATHANG WHERE MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END IF @TinhTrang <> N'Đang xử lý' or @TinhTrang <>N'Thành công' BEGIN print N'Tình trạng không hợp lệ' rollback tran return 1	**R(DONDATHANG)** //T1 xin khóa S và không được cấp do T1 đã nhả khoá

		END	
		B2: Cập nhật tình trạng đơn hàng UPDATE DONDATHANG SET TinhTrang= @TinhTrang WHERE MaDH = @MaDH	**U(DONDATHANG)** //T2 xin khóa U và được cho
		COMMIT	
Vì 2 transaction đều được cài đặt mức cô lập repeatable read, khi T1 đang được cấp khóa S trên bảng DONDATHANG nên không thể thiết lập khóa X trên đơn vị dữ liệu đang có khóa S, do đó T2 không được cấp khóa S, vì vậy T2 phải chờ để update dữ liệu tình trạng cho bảng DONDATHANG. Unrepeatable read được giải quyết.			

8. Tình huống 2: một khách hàng đang cập nhật đơn hàng của họ thì có một khác hàng khác thêm món ăn vào đơn hàng của họ dẫn đến Cycle Deadlock

ERR01: *Cycle Deadlock* T1 (User = khách hàng 1): thực hiện cập nhật số lượng của món ăn trong đơn hàng. T2 (User = khách hàng 2): thực hiện thêm món ăn vào đơn hàng			
sp_capNhatDonHang		**sp_themChiTietDonHang**	

***Input**: MaDH, MaTP, MaDT, SoLuong* ***Output**: cập nhật đơn hàng thành công*	**Khóa**	***Input**: MaDH, MaTP, MaDT, SoLuong* ***Output**: thêm chi tiết thành công*	**Khóa**
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ		SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	
BEGIN TRAN			
B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONDATHANG A WHERE A.MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END	**S(DONDATHANG)** giữ khóa S đến hết giao tác trên bảng DONDATHANG		
B2: Kiểm tra thông tin (2) thực phẩm IF NOT EXISTS (SELECT * FROM THUCPHAM A WHERE A.MaTP=@MaTP and A.MaDT=@MaDT) BEGIN	**S(THUCPHAM)** giữ khóa S đến hết giao tác		

print N'Thực phẩm này không tồn tại' rollback tran return 1 END	trên bảng THUCPHAM		
B3: Cập nhật tổng giá của đơn đặt hàng UPDATE DONDATHANG SET GiaTriDH = GiaTriDH + (@SoLuong - SLCu)*DonGia WHERE MaDH=@MaDH	**X(DONDATHANG)** giữ khóa U đến hết giao tác trên bảng DONDATHANG		
If exixts (SELECT * FROM CHITIETDONDATHANG WHERE MaDH = @MaDH and MaTP = @MaTP and MaDT = @MaDT print('')	**X(CHITIETDONDATHANG)**		
WAITFOR DELAY '00:00:05'			

		BEGIN TRAN	
		B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONDATHANG A WHERE A.MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END	**R(DONDATHANG)** //Không cấp khoá S được do T1 đang giữ
B4: Cập nhật số lượng của chi tiết đơn hàng UPDATE CHITIETDONDATHANG SET SoLuong = @SoLuong WHERE MaDH = @MaDH and MaTP = @MaTP and MaDT = @MaDT	**X(CHITIETDONDATHANG)**		
COMMIT			

		B2: Kiểm tra thông tin (2) thực phẩm IF NOT EXISTS (SELECT * FROM THUCPHAM WHERE MaTP=@MaTP and MaDT=@MaDT) BEGIN print N'Thực phẩm này không tồn tại' rollback tran return 1 END	**R(THUC PHAM)** //Không cần xin khoá
		B3: Cập nhật số lượng món ăn của đơn đặt hàng IF EXISTS (SELECT * FROM CHITIETDONDATHANG WHERE MaTP=@MaTP and MaDT=@MaDT) BEGIN UPDATE CHITIETDONDATHANG	**X(CHITIET DONHANG)**

		SET SoLuong = @SoLuong
		WHERE MaDH = @MaDH and MaTP = @MaTP and MaDT = @MaDT
		END
		ELSE
		BEGIN
		INSERT INTO CHITIETDONDATHANG
		VALUES (@MaDH, @MaTP, @MaDT, @SoLuong, NULL)
		END	

		WAITFOR DELAY '00:00:05'	
		B4: Cập nhật tổng giá của đơn đặt hàng UPDATE DONDATHANG SET GiaTriDH = GiaTriDH + (@SoLuong - SLCu)*DonGia WHERE MaDH=@MaDH	**X(DONDATHANG)**
		COMMIT	

Vì 2 transaction đều được cài đặt mức cô lập repeatable read, khi T1 đang được cấp khóa S trên bảng DONDATHANG và bảng CHITIETDONDATHANG, bảng 2 không được cấp khóa X cho cả 2 bảng. Vì vậy T2 phải chờ T1 hoàn thành thì mới có thể cập nhật dữ liệu. Hướng giải quyết là không cho Tran chạy lồng vào nhau.