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%

CÀI ĐẶT 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.

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 UNCOMMITTED		SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED	
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	**R(KHACHHANG)** //Không cần xin khoá do mức cô lập		
B3: Thêm thông tin vào bảng KHACHHANG insert into KHACHHANG values(@MaKH,@HoTen,@DiaChi,@SDT,@Email)	**X(KHACHHANG)** //T1 xin khóa ghi trên bảng KHACHHANG và được cấp		

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	**R(KHACHHANG)** //Không cần xin khoá nên T2 có thể đọc bảng KHACHHANG
		B2: Xem thông tin khách hàng SELECT * FROM KHACHHANG WHERE MaKH = @MaKH	**R(KHACHHANG)** //Không cần xin khoá nên T2 có thể đọc bảng KHACHHANG

		COMMIT	

B4: Kiểm tra thông tin số điện thoại có bị trùng không
 	
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.
	R(KHACHHANG)
	
	//Không cần xin khoá		

ROLLBACK			
T2 đọc phải thông tin khách hàng đang thêm vào chưa commit và sau đó T1 rollback nên 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.

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 where MaDT=@MaDT) begin print N'Mã đối tác chưa tồn tại' rollback tran return 1 end	**S(DOITAC)** //T1 xin khoá S trên bảng DOITAC và được cấp khóa. T1 giữ khóa đến hết giao tác		
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 where MaDT=@MaDT) begin print N'Không thể xóa, đối tác không tồn tại' rollback tran return 1 end	**S(DOITAC)** //T2 xin khoá S trên bảng DOITAC và được cấp khóa. T2 giữ khóa đến hết giao tác
B3: Cập nhật thông tin đối tác update DOITAC set Email = @Email, NgDaiDien = @NgDaiDien, SLChiNhanh = @SLChiNhanh, TenQuan = @TenQuan,	//T1 xin khóa X trên bảng DOITAC nhưng không được do T2 đang giữ khoá đọc trên bảng. T1 chờ T2 trả khóa đọc trên DOITAC.	B3: Xóa đối tác delete from DOITAC where MaDT = @MaDT	//T2 xin khóa X trên bảng DOITAC nhưng không được do T1 đang giữ khoá đọc trên bảng. T2 chờ T1 trả khóa đọc trên DOITAC.

LoaiTP = @LoaiTP where MaDT = @MaDT			
COMMIT		COMMIT	
T1 chờ T2 trả khóa đọc trên bảng DOITAC và T2 chờ T1 trả khóa đọc trên bảng DOITAC nên xảy ra 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.

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 READ COMMITTED		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 X trên bảng DOITAC** **SQL : Cấp khóa X trên bảng DOITAC**		
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 S trên bảng DOITAC** **SQL : Không cấp khóa S do T1 đang giữa khóa X bảng DOITAC**
		B3: Thêm đối tác insert into DOITAC values (@MaDT, @Email, @NgDaiDien, @SLChiNhanh, @TenQuan, @LoaiTP)	**T2 : Xin khóa S trên bảng DOITAC** **SQL : Không cấp khóa S do T1 đang giữa khóa X bảng DOITAC**
SELECT * FROM DOITAC	**T1 : Trả khóa X**	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 => Xảy ra Phantom.

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.

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 ***Output***: Đơn hàng đó sẽ có thuộc tính TinhTrang là "Đã hủy đơn"		***Input***: Mã đơn hàng, tình trạng đơn ***Output***: Đơn hàng đó có thuộc sẽ có thuộc tính TinhTrang = input tình trạng đơn	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
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' rollback tran return 1 end	**T1 : Xin khóa X trên bảng DONDATHANG** **SQL : Cấp khóa X trên bảng 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 S trên bảng DONDATHANG** **SQL : Không cấp khóa S do T1 đang giữa khóa X 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 X**	B3: Update tình trạng hiện tại của đơn hàng update DONDATHANG SET TinhTrang = @TinhTrang where MaDH = @MaDonDH	
commit		commit	
Do user KHACHHANG thực hiện T1 hủy đơn hàng trước khi user DOITAC thực hiện T2 nên xảy ra tình trạng đơn hàng bị hủy không được update, dẫn đến đơn hàng bị DOITAC chuyển sang trạng thái mà DOITAC muốn chuyển và mất dữ liệu mà KHACHHANG đã update => Xảy ra Lost Update.			

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

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		SET TRANSACTION ISOLATION	

LEVEL **UNREPEATEABLE READ**		LEVEL **UNREPEATEABLE READ**	
BEGIN TRAN			
B1: Kiểm tra thông tin 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(ThucPham)** //Không cần xin khóa do mức cô lập		
B2: Update giá cho thực phẩm update ThucPham set Gia=@Gia where MaTP=@MaTP and MaDT=@MaDT	**U(ThucPham)** //T1 xin khoá U và được cho.		

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 print N'Thực phẩm này không tồn tại' rollback tran return 1 end	**R(ThucPham)** //Không cần xin khoá, T2 có thể đọc bảng ThucPham
		B2: Đọc thông tin thực phẩm select *from ThucPham where MaTP=@MaTP and MaDT=@MaDT	**R(ThucPham)** //Không cần xin khoá, T2 có thể đọc bảng ThucPham

		COMMIT	

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. Do đó dữ liệu T2 đọc được trước đó là dữ liệu rác.
ROLLBACK
	//Không cần xin khoá		

T2 đọc phải thông tin thực phẩm đang cập nhật chưa commit và sau đó T1 rollback nên 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.

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 READ UNCOMMITTED		SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED	
BEGIN TRAN			
B1: Kiểm tra thực phẩm có tồn tại không if not exists(select* from ThucPham where MaTP=@MaTP and MaDT=@MaDT) begin	R(ThucPham) //Không cần xin khóa do mức cô lập		

print N'Thực phẩm này không tồn tại' rollback tran return 1 end			
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 return 1 end	R(ThucPham) //Không cần xin khóa do mức cô lập		

		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	**R(ThucPham)** //Không cần xin khoá nên T2 có thể đọc bảng ThucPham
		B2: Kiểm tra tình trạng mới có trùng tình trạng cũ if @TinhTrang=(select TinhTrang from ThucPham where MaTP=@MaTP and MaDT=@MaDT)	**R(ThucPham)** //Không cần xin khoá nên T2 có thể đọc

		begin print N'Tình trạng mới giống tình trạng cũ' rollback tran return 1 end	bảng ThucPham
		B3: Cập nhật tình trạng thực phẩm update ThucPham set TinhTrang=@TinhTrang where MaTP=@MaTP and MaDT=@MaDT	**U(ThucPham)** //T2 xin khóa U và được cho
select *from ThucPham where MaTP=@MaTP and MaDT=@MaDT and TinhTrang=@TinhTrang	**R(ThucPham ThucPham)** //Không cần xin khoá		

COMMIT		COMMIT	
T1 lần 1 tìm (đọc) được thực phẩm với trạng thái cần tìm nhưng lần 2 không tìm được (lần 2 xuất được dòng nào) do T2 đã thay đổi tình trạng của thực phẩm mà T1 tìm kiếm. Dữ liệu 2 lần đọc trong giao tác là khác nhau.			

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

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		SET TRANSACTION ISOLATION	

LEVEL READ UNCOMMITTED		LEVEL READ UNCOMMITTED	
BEGIN TRAN			
B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONHANG A WHERE A.MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END	**R(DONDATHANG)** //Không cần xin khóa do mức cô lập		
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)**		
WAITFOR DELAY '00:00:20'			

		BEGIN TRAN	
		B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONHANG 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(DONHANG)** //Không cần xin khóa do mức cô lập

		B2: Cập nhật tình trạng đơn hàng UPDATE DONHANG SET TinhTrang= @TinhTrang WHERE MaDH = @MaDH	**U(DONHANG)** //T2 xin khóa U và được cho
		COMMIT	
print N'Đơn hàng đang được giao' return 1	**R(ThucPham ThucPham)** //Không cần xin khoá		
COMMIT			
T1 lần đầu tìm kiếm thì đơn hàng đang được xử lý nhưng lần thứ 2 tình trạng đã được sửa thành đã giao thành công			

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	Khóa	sp_themChiTietDonHang	Khóa
***Input**: MaDH, MaTP, MaDT, SoLuong*		***Input**: MaDH, MaTP, MaDT, SoLuong*	

Output: *cập nhật đơn hàng thành công*		***Output***: *thêm chi tiết thành công*	
SET TRANSACTION ISOLATION LEVEL CYCLE DEADLOCK		SET TRANSACTION ISOLATION LEVEL **CYCLE DEADLOCK**	
BEGIN TRAN			
B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONHANG A WHERE A.MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1 END	**R(DONHANG)**		
B2: Kiểm tra thông tin (2) thực phẩm IF NOT EXISTS (SELECT * FROM THUCPHAM A WHERE A.MaTP=@MaTP and A.MaDT=@MaDT) BEGIN print N'Thực phẩm này không tồn tại'	**R(THUC PHAM)**		

rollback tran return 1 END			
B3: Cập nhật tổng giá của đơn đặt hàng UPDATE DONDATHANG SET GiaTriDH = GiaTriDH + (@SoLuong - SLCu)*DonGia WHERE MaDH=@MaDH	**U(DONDATHANG)**		
WAITFOR DELAY '00:00:20'			
		BEGIN TRAN	
		B1: Kiểm tra thông tin (1) đơn hàng IF NOT EXISTS (SELECT * FROM DONHANG A WHERE A.MaDH=@MaDH) BEGIN print N'Đơn hàng này không tồn tại' rollback tran return 1	**R(DONHANG)** //Không cần xin khoá

		END	
		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 hoặc thêm món ăn vào đơn đặt hàng IF EXISTS (SELECT * FROM THUCPHAM WHERE MaTP=@MaTP and MaDT=@MaDT) BEGIN UPDATE CHITIETDONHANG SET SoLuong = @SoLuong WHERE MaDH = @MaDH and MaTP = @MaTP and MaDT = @MaDT	**X(CHITIET DONHANG)**

		END
		ELSE
		BEGIN
		INSERT INTO CHITIETDONHANG
		VALUES (@MaDH, @MaTP, @MaDT, @SoLuong, NULL)
		END	

		WAITFOR DELAY '00:00:20'	
B4: Cập nhật số lượng của chi tiết đơn hàng UPDATE CHITIETDONHANG SET SoLuong = @SoLuong WHERE MaDH = @MaDH and MaTP = @MaTP and MaDT = @MaDT	**U(CHITIETDONHANG)**		
COMMIT			
		B4: Cập nhật tổng giá của đơn đặt hàng UPDATE DONDATHANG SET GiaTriDH = GiaTriDH + (@SoLuong - SLCu)*DonGia	**U(DONDATHANG)**

		WHERE MaDH=@MaDH	
		COMMIT	

Khi T1 và T2 chạy lượt đầu, T1 xin khoá **U(DONDATHANG)** còn T2 xin khoá **X(CHITIETDONHANG)** và cả 2 đều chưa trả khoá vì chưa thực hiện xong sp.

Đến lượt thứ 2, T1 cần xin khoá **U(CHITIETDONHANG)** nhưng T2 đang giữ, và T2 không thể trả khoá vì đang chờ xin khoá **U(DONDATHANG)** của T1. Cả 2 bên đều đang chờ nnhau dẫn đến tình trạng Cycle Deadlock