```
--Hồ Phúc Lâm
--Transaction

/*
I. SINGLE TRANSACTION
Autocommit mode là chế độ quản lý giao dịch mặc định của SQL Server
Database Engine. Mỗi lệnh Transact-SQL được commit hoặc Rollback khi
nó hoàn thành.
*/
Use Northwind

--1) Thêm vào bảng Categories một dòng dữ liệu tùy ý bằng câu lệnh INSERT..VALUES.
--a) Thực hiện lệnh chèn thêm vào bảng Categories một dòng dữ liệu tùy ý bằng cách thực hiện lệnh Begin
tran và Rollback, dùng câu lệnh Select * from Categories xem kết quả.
     begin tran
          insert into Categories(CategoryName, Description) values ('Computer', 'PC, Laptop')
          select * from Categories
     rollback
     Go
--xem kết quả: sau khi chèn thì thực hiện rollback để hủy giao dịch, khi kiểm tra lại bảng thì sẽ hủy
chèn dòng dữ liệu trên
     select * from Categories
     Go

--b) Thực hiện câu lệnh trên với lệnh commit và kiểm tra kết quả.
     begin tran
          insert into Categories(CategoryName, Description) values ('Computer', 'PC, Laptop')
          select * from Categories
     commit
     Go

--Xem kết quả: sau khi chèn thì thực hiện commit thì sẽ xác nhận giao dịch, lưu lại dữ liệu
     select * from Categories
```

```
    Go



--2) Tắt chế độ autocommit của SQL Server (SET IMPLICIT_TRANSACTIONS ON). Tạo đoạn batch gồm các thao
tác:
    set implicit_transactions on
    Go
--Thêm một dòng vào bảng Categories
    begin tran
        insert into Categories(CategoryName, Description) values ('Smart Phone', 'Iphone, Samsung,
Oppo, Xiaomi, Remind')
--Tạo một bảng Test (id int, name nvarchar(10))
    create table Test(
        id int,
        name nvarchar(10)
    )
--Thêm một dòng vào Test
    insert into Test(id, name) values (1, 'HoPhucLam')
--ROLLBACK;
    rollback
    Go
--Xem dữ liệu ở bảng Categories và Test để kiểm tra dữ liệu, giải thích kết quả.
    select * from Categories
    select * from Test
    Go
--Giải thích: Sau khi thực hiện chèn thì dữ liệu được chèn vào Categories và Test thì dữ liệu đc thêm
vào. Thực hiện Rollback thì sẽ hủy giao dịch và đưa dữ liệu ban đầu,
--sau đó kiểm tra lại thì trong bảng Categories và Test thì không có dòng dữ liệu nào do đã dùng
rollback
```

```
/*
3) Viết đoạn batch thực hiện các thao tác sau (lưu ý thực hiện lệnh SET
XACT_ABORT ON: nếu câu lệnh T-SQL làm phát sinh lỗi run-time, toàn
bộ giao dịch được chấm dứt và Rollback)
Thực thi đoạn batch, quan sát kết quả và các thông báo lỗi và giải
thích kết quả.
*/

set xact_abort on
Go

begin tran
     --Câu lệnh SELECT với phép chia 0 :SELECT 1/0 as Dummy
     select 1/0 as Dummy

     --Cập nhật một dòng trên bảng Categories với id=’9’ (id này không tồn tại)
     update Categories
     set CategoryName = 'New Name' where CategoryID = '9';

     --Xóa một dòng không tồn tại trên bảng Categories (id=’66’)
     delete from Categories where CategoryID = '66';

     --Thêm một dòng bất kỳ vào bảng Categories
     insert into Categories (CategoryName, Description) values ('New Category Apha', 'Test category
Apha');

commit
Go

--Xem lại bảng
select * from Categories
```

```
--Kết quả: với set xact_about on thì mọi lỗi run-time trong giao dịch sẽ làm cho toàn bộ giao dịch bị
rollback hủy đi.
--Nếu 1/0 đổi thành 1/1 thì giao dịch sẽ không lỗi và thực hiện được tới commit.


/*
4) Thực hiện lệnh SET XACT_ABORT OFF (những câu lệnh lỗi sẽ rollback,
transaction vẫn tiếp tục) sau đó thực thi lại các thao tác của đoạn batch
ở câu 3. Quan sát kết quả và giải thích kết quả?
*/

set xact_abort off
Go
begin tran
     --Câu lệnh SELECT với phép chia 0 để gây lỗi
     select 1/0 as Dummy

     --Cập nhật một dòng trên bảng Categories với id=’9’ (id này không tồn tại)
     update Categories
     set CategoryName = 'New Name' where CategoryID = '9';

     --Xóa một dòng không tồn tại trên bảng Categories (id=’66’)
     delete from Categories where CategoryID = '66';

     --Thêm một dòng bất kỳ vào bảng Categories
     insert into Categories (CategoryName, Description) values ('NewApha', 'Test category Apha');

commit
Go

--Xem lại bảng
select * from Categories
```

```
--Kết quả: lỗi khi khi chia cho 0 sẽ không khiến toàn bộ giao dịch bị rollback chỉ hủy câu lệnh đó.
--Câu lệnh update không gây lỗi, câu lệnh insert sẽ được thêm vào bảng Categories. Commit sẽ thực hiện
xác nhận giao dịch.
--Set xact_abort off thì các lỗi run-time sẽ không hủy bỏ toàn bộ giao dịch chỉ ảnh hướng đến câu lệnh
gây lỗi.
--còn các câu lệnh khác trong giao dịch vẫn tiếp tục thực thi


-- II.CONCURRENT TRANSACTIONS

/*
1)Tạo bảng Accounts (AccountID int NOT NULL PRIMARY KEY,balance int NOT NULL
CONSTRAINT unloanable_account CHECK
(balance >= 0)
Chèn dữ liệu:
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
*/
--Tạo bảng
create table Accounts(
     AccountID int NOT NULL PRIMARY KEY,
     balance int NOT NULL CONSTRAINT unloanable_account CHECK(balance >= 0)
)
Go

--Chèn dữ liệu
insert into Accounts (AccountID,balance) values (101,1000)
insert into Accounts (AccountID,balance) values (202,2000)

--xem lại bảng
select * from Accounts
Go
--Xóa dữ liệu chèn
```

```
delete from Accounts
Go



/*2)SET TRANSACTION ISOLATION LEVEL*/

/*-- Thiết lập mức độ cô lập là READ UNCOMMITTED:
có thể đọc những dòng đang đượchiệu chỉnh bởi
các transaction khác nhưng chưa commit */
set transaction isolation level read uncommitted
begin tran
     select * from Employees
commit
Go

/*-- Thiết lập mức độ cô lập là READ COMMITTED
không thể đọc những dòng đang hiệu chỉnh bởi
những transaction khác mà chưa commit*/
set transaction isolation level read committed
begin tran
     select * from Employees
commit
Go

/*-- Thiết lập mức độ cô lập là REPEATABLE READ
Transaction đọc các dòng dữ liệu và đảm bảo rằng các dòng
này sẽ không thay đổi trong suốt quá trình thực thi của transaction.*/
set transaction isolation level repeatable read
begin tran
     select * from Employees
commit
Go
```

```
/*-- Thiết lập mức độ cô lập là SERIALIZABLE:Mức độ cô lập cao nhất,
đảm bảo tính nhất quán tuyệt đối bằng cách chặn các transaction
khác có thể ảnh hưởng đến dữ liệu mà transaction hiện tại đang đọc.*/
set transaction isolation level serializable
begin tran
     select * from Employees
commit
Go

/*3)Mở 2 cửa sổ Query của SQL server, thiết lập SET TRANSACTION
ISOLATION LEVEL READ COMMITTED ở cả 2 cửa sổ (tạm gọi là
clients A bên trái, và client B bên phải).
Client A, client B: cùng thực hiện lệnh SELECT trên bảng
Accounts với AccountID =101
Clients A cập nhật account trên AccountID =101, balance
=1000-200
Client B cập nhật account trên AccountID =101, balance =1000-
500
Client A: SELECT trên Accounts với AccountID =101; COMMIT;
Client B: SELECT trên Accounts với AccountID =101; COMMIT;
Quan sát kết quả hiển thị và giải thích.
*/
--Phiên A bên trái:
--thiết lập read committed
set transaction isolation level read committed
begin tran

     --xem Accounts
     select * from Accounts where AccountID = '101'

     --Clients A cập nhật account trên AccountID =101, balance=1000-200
     update Accounts set balance = 1000-200 where AccountID = '101'
```

```
    --Client A: SELECT trên Accounts với AccountID =101; COMMIT;
    select * from Accounts where AccountID = '101';
Commit;


--Câu 3
--Phiên B bên phải:
use Northwind
select * from Accounts
Go

--thiết lập read committed
set transaction isolation level read committed
begin tran

    --xem Accounts
    select * from Accounts where AccountID = '101'

    --Clients B cập nhật account trên AccountID =101, balance=1000-500
    update Accounts set balance = balance-500 where AccountID = '101'

    --Client B: SELECT trên Accounts với AccountID =101; COMMIT;
    select * from Accounts where AccountID = '101'
Commit

--=> kết quả:
/*sau khi set và thiết lập read committed thì: không thể đọc những dòng đang hiệu
chỉnh bởi những transaction khác mà chưa commit. Ở trường hợp này xảy ra
Tình huống Phiên A cập nhật accoutID ='101' nhưng chưa commit nên Phiên B không thể chạy được lệnh
xem dữ liệu của accountID='101' và Phiên B sẽ trong trạng thái Executing(đang thực thi) điều nãy sẽ
diễn ra cho đến khi Phiên A được Commit thì Phiên B sẽ hoàn thành lệnh xem dữ liệu đó.
```

```
=> kết luận: phải để Phiên A hoàn thành thao tác và commit thì mới tới lượt Phiên B cập nhật tiếp giá
trị,
từ đó đảm bảo tính nhất quán và cô lập dữ liệu
*/



/*
4)Thiết lập ISOLATION LEVEL REPEATABLE READ (không thể đọc được
dữ liệu đã được hiệu chỉnh nhưng chưa commit bởi các transaction khác
và không có transaction khác có thể hiệu chỉnh dữ liệu đã được đọc bởi các
giao dịch hiện tại cho đến transaction hiện tại hoàn thành) ở 2 client. Thực
hiện yêu cầu sau:
Client A, client B: cùng thực hiện lệnh SELECT trên bảng
Accounts với AccountID =101
Clients A cập nhật accounts trên AccountID =101, balance
=1000-200
Client B cập nhật accounts trên AccountID =101, balance =1000-
500.
Client A: SELECT trên Accounts với AccountID =101; COMMIT;
Quan sát kết quả hiển thị và giải thích.
*/
--Phiên Client A bên trái
set transaction isolation level repeatable read
begin tran

     --xem Accounts
     select * from Accounts where AccountID = '101'

     --Clients A cập nhật account trên AccountID =101, balance=1000-200
     update Accounts set balance = 1000-200 where AccountID = '101'

     --Client A: SELECT trên Accounts với AccountID =101; COMMIT;
```

```
     select * from Accounts where AccountID = '101';
Commit
Go

--Câu 4
--Phiên B bên phải:
--thiết lập repeatable read
set transaction isolation level repeatable read
begin tran

     --xem Accounts
     select * from Accounts where AccountID = '101'

     --Clients B cập nhật account trên AccountID =101, balance=1000-500
     update Accounts set balance = 1000-500 where AccountID = '101'

commit
Go
--Kết quả và giải thích:
/*
4e: giải thích sau khi commit bên phiên A thì phiên B sẽ thực hiện.
REPEATABLE READ đảm bảo rằng các giao dịch không thể đọc hoặc ghi vào
các bản ghi đã được đọc bởi giao dịch hiện tại cho đến khi
giao dịch hiện tại hoàn tất. Điều này đảm bảo rằng dữ liệu được đọc
bởi một giao dịch sẽ không thay đổi bởi các giao dịch khác cho
đến khi giao dịch hoàn tất.=> Đảm bảo tính nhất quán và cô lập dữ liệu
*/

select * from Accounts
Go

/*
5)Giả sử có 2 giao dịch chuyển tiền từ tài khoản 101 và 202 như sau:
```

```
Client A chuyển 100$ từ tài khoản 101 sang 202Client B chuyển 200$ từ tài khoản 202 sang 101.
Viết các lệnh tương ứng ở 2 client để kiểm soát các giao dịch xảy ra đúng
*/
--Lệnh ở Client A
set transaction isolation level serializable
begin tran

     --khai bao bien
     declare @num101 money
     select @num101 = balance from Accounts where AccountID = '101'

     --kiem tra ngan sach tien co trong tai khoan 101
     if @num101 >= 100
          begin
               update Accounts set balance = balance - 100 where AccountID = '101'
               update Accounts set balance = balance + 100 where AccountID = '202'
               commit
               print N'Chuyển khoản thành công từ 101 sang 202';
          end
     else
          begin
               rollback
               print N'Không đủ tiền trong tài khoản 101';
          end
Go

--Lệnh ở Client B
set transaction isolation level serializable
begin tran

     --khai bao bien
     declare @num202 money
     select @num202 = balance from Accounts where AccountID = '202'
```

```
     --kiem tra ngan sach tien co trong tai khoan 202
     if @num202 >= 200
          begin
               update Accounts set balance = balance - 200 where AccountID = '202'
               update Accounts set balance = balance + 200 where AccountID = '101'
               commit
               print N'Chuyển khoản thành công từ 202 sang 101';
          end
     else
          begin
               rollback
               print N'Không đủ tiền trong tài khoản 202';
          end
Go

--xem lại
select * from Accounts

/*
6)Xóa tất cả dữ liệu của bảng Accounts. Thêm lại các dòng mới
INSERT INTO Accounts (AccountID ,balance) VALUES (101,1000);
INSERT INTO Accounts (AccountID ,balance) VALUES (202,2000);
Quan sát kết quả và giải thích.
*/
--Xóa dữ liệu chèn
delete from Accounts
Go

--Chèn dữ liệu
insert into Accounts (AccountID,balance) values (101,1000)
insert into Accounts (AccountID,balance) values (202,2000)
```

```
--xem lại bảng
select * from Accounts
Go

--Clien A bên trái
/*Client A: cập nhật balance của account giảm đi 100 cho
AccountID =101, cập nhật balance của account tăng lên 100 cho
AccountID =202*/
begin tran
     update Accounts set balance = balance - 100 where AccountID = '101'
     update Accounts set balance = balance + 100 where AccountID = '202'

--client A thực hiện lệnh
rollback;
select * from Accounts
commit;

--Câu 6:
--Client B: thiết lập ISOLATION LEVEL READ UNCOMMITTED
set transaction isolation level read uncommitted
begin tran
     select * from Accounts
commit


/*7)Xóa tất cả dữ liệu của bảng Account, thêm lại các dòng mới
INSERT INTO Accounts (AccountID ,balance) VALUES (101,1000); INSERT INTO Accounts (AccountID ,balance)
VALUES (202,2000);*/
--Xóa dữ liệu chèn
delete from Accounts
Go

--Chèn dữ liệu
```

```
insert into Accounts (AccountID,balance) values (101,1000)
insert into Accounts (AccountID,balance) values (202,2000)

--xem lại bảng
select * from Accounts
Go

--Phía Phiên Client A
--Client A: thiết lập ISOLATION LEVEL REPEATABLE READ;
--Lấy ra các Accounts có Balance>1000
set transaction isolation level repeatable read
begin tran
     select * from Accounts where balance > '1000'

     --client a: select * FROM Accounts WHERE balance > 1000; COMMIT;
     select * from Accounts where balance > '1000'
commit


--Quan sát kết quả và giải thích.
```