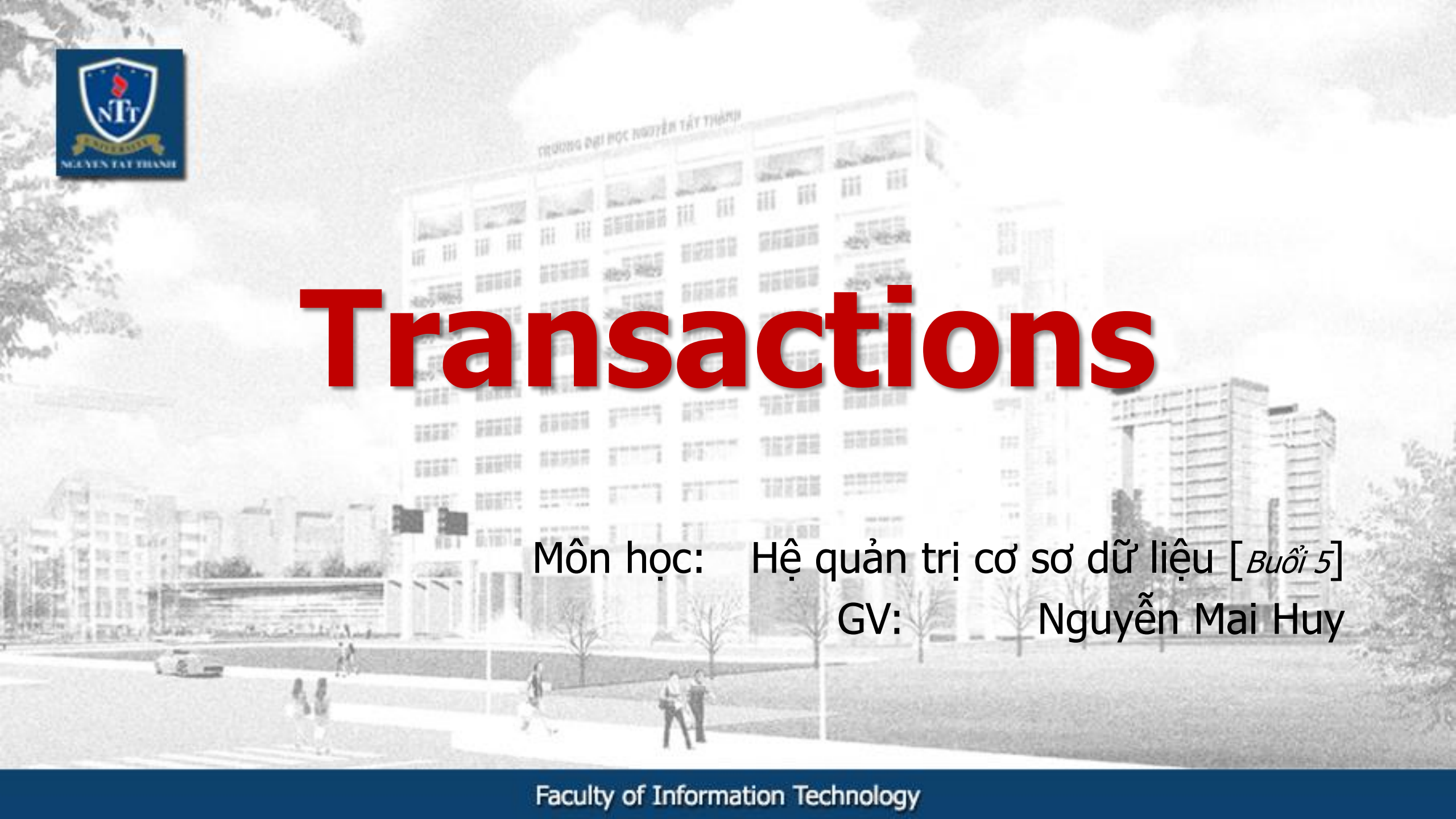


Transactions

Môn học: Hệ quản trị cơ sơ dữ liệu [*Buổi 5*]

GV: Nguyễn Mai Huy

Transactions

Transaction Overview

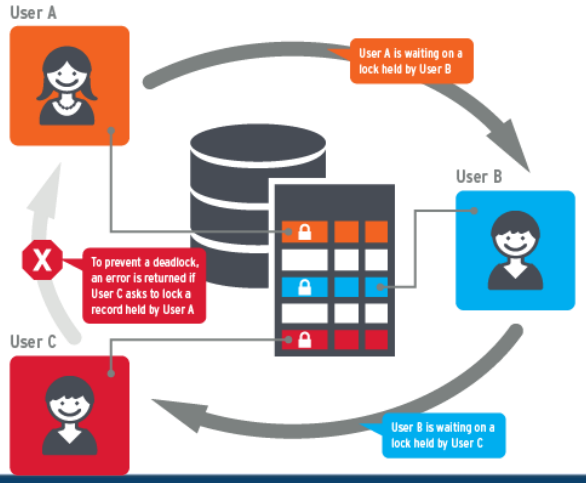
Nói chung, **Transaction** (*giao dịch*) là một mục tiêu cụ thể được thực hiện giữa hai chủ thể. Trong đó, một giao dịch có thể bao gồm nhiều hành động đơn lẻ trực thuộc, nhưng nếu chỉ một hành động (*thuộc giao dịch*) không thể thực hiện thì giao dịch tương ứng được xem là không thành công, ***giao dịch thất bại***.

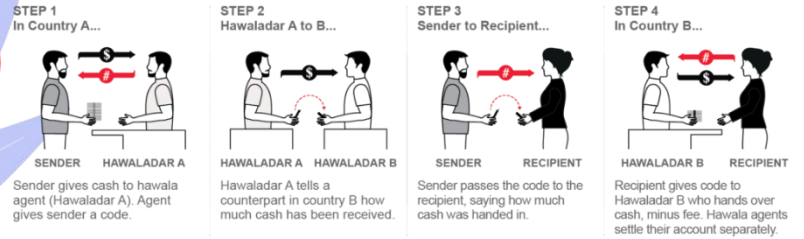
Các giao dịch quen thuộc: Mua hàng bao gồm (*Chọn hàng*, *Đóng gói*, *Thanh toán*), Ký kết hợp đồng (*Tìm hiểu*, *Đàm phán*, *thỏa thuận*, *Cam kết,* ...), … Hãy hình dung các bước mà bạn vẫn thực hiện "**Giao dịch tại cây ATM**"

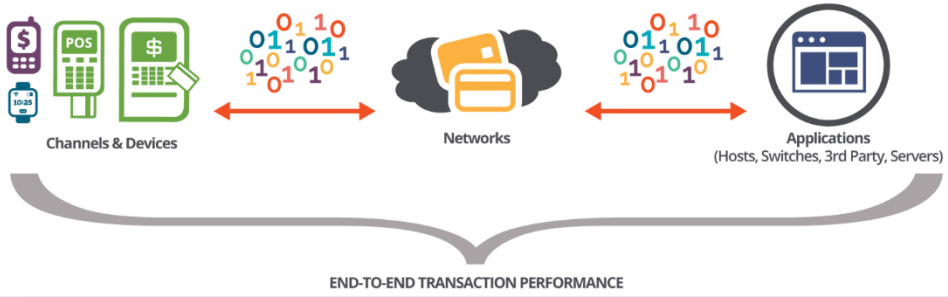
1. Insert your card
2. Select the language
3. Enter your PIN
4. Select the transaction type, which is withdraw in our example here
5. Enter the amount
6. Pull the card
7. Take your money

Parent-Child :: Story

Many transactions in real life







ACID

Atomicity

Transactions are all or nothing

A

Consistency

Only valid data is save

C

Isolation

Transactions do not affect each other

I

Durability

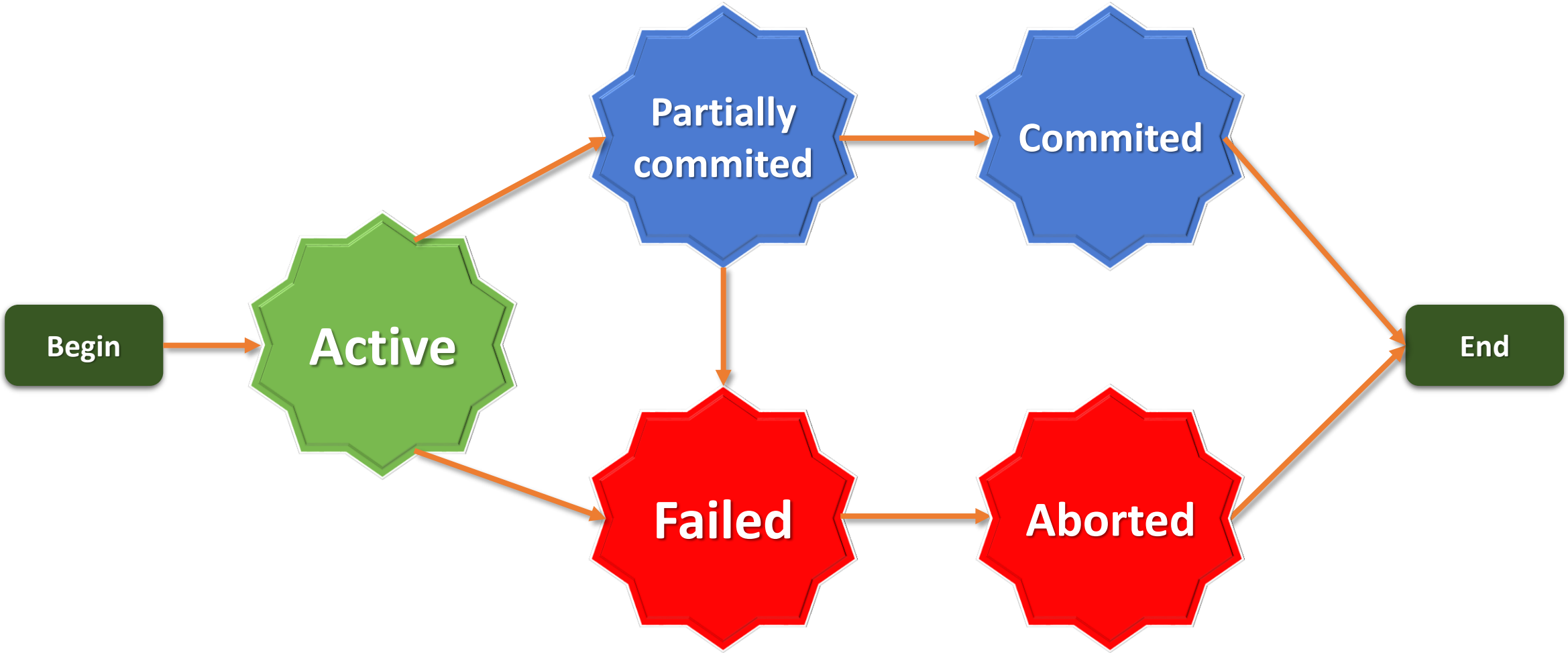
Written data will not be lost

D

ACID :: Decription

- **Atomicity:** Mang ý nghĩa là nguyên tử, tức là mỗi giao dịch được xem làm một đơn vị thống nhất (*nhỏ nhất trong các nhiệm vụ của hệ thống*); Nếu từng hành động đơn lẻ trong giao dịch đều thực hiện thành công thì giao dịch mới được xem là thành công. Ngược lại, bất cứ một hành động ở mức đơn vị thuộc giao dịch bị thất bại, thì toàn bộ giao dịch thất bại, bị hủy & phải bắt đầu lại từ đầu.
- **Consistency:** Tính nhất quán đảm bảo rằng giao dịch sẽ không ảnh hưởng đến tính nhất quán của cơ sở dữ liệu và có trạng thái hợp lệ khi tuân thủ tất cả các quy tắc đã được thiết lập trên cơ sở dữ liệu.
- **Isolation:** có nghĩa là, mỗi giao dịch trong hệ thống là độc lập với nhau, được tách biệt với các giao dịch khác, đang thực hiện đồng thời trong hệ thống. Điều này có nghĩa là các giao dịch sẽ không bị (*hoặc làm*) ảnh hưởng lẫn nhau trong quá trình thực thi.
- **Durability:** Bền bỉ có nghĩa là kết quả của giao dịch phải được cam kết lưu trữ vĩnh viễn vào cơ sở dữ liệu, không thể bị mất ngay cả trong trường hợp có sự cố hoặc hệ thống bị ngắt một cách bất thường, đồng thời phải có cơ chế khôi phục dữ liệu ngay cả khi xảy ra những tình huống này.

Transaction states

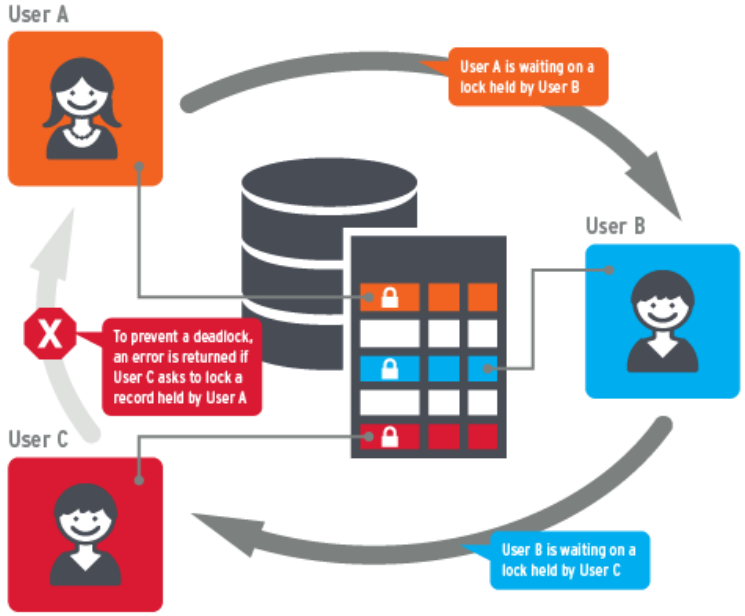


Transaction states

- Khi giao dịch bắt đầu khởi chạy, trạng thái **Active** sẽ được thiết lập
- Từng hành động nhỏ trong giao dịch được thực hiện thành công, được gọi là: **Partially Committed**, cam kết từng phần
- Khi tất cả các hành động đơn lẻ thuộc giao dịch đều được thực hiện thành công, trạng thái này được gọi là: **Committed**
- Nếu trong trường hợp, giao dịch không thể thực hiện thành công, trạng thái này được gọi là: **Aborted**, giao dịch tương ứng bị hủy bỏ
- Nếu giao dịch gặp bất cứ thất bại nào trong quá trình thực thi, trạng thái này được gọi là **Failed**, và điều này sẽ khiến giao dịch bị hủy bỏ **Aborted**, không thể thực hiện .

Transaction

in Transact – SQL



Auto-committed transaction

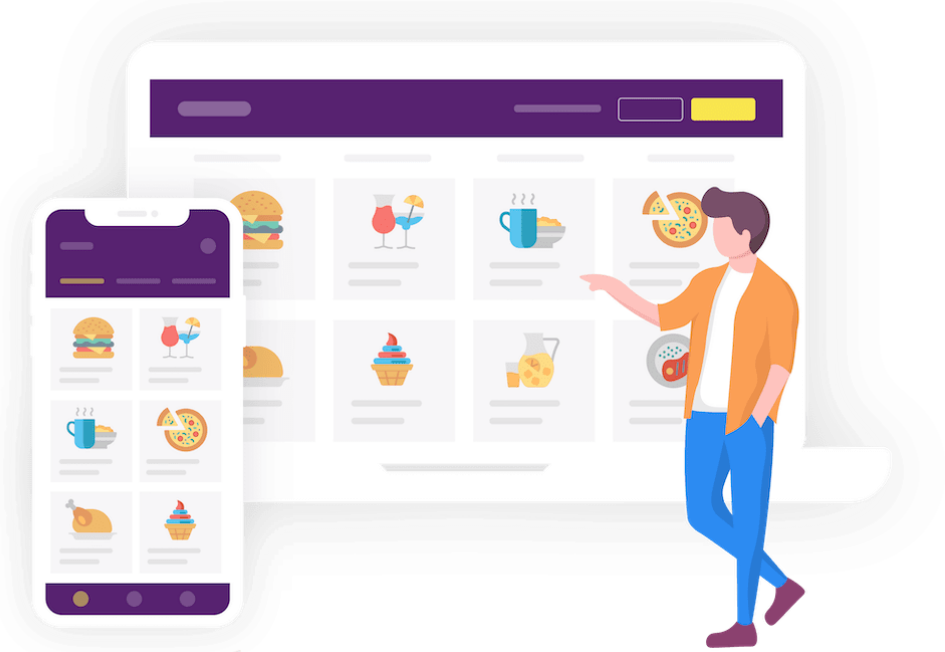
Trong SQL Server, mỗi câu lệnh T-SQL duy nhất được coi là một **Auto-committed transaction**, tức là từng lệnh sẽ tự động được **committed** (*cam kết*) khi thi hành thành công và được **rolled-back** (*khôi phục*) nếu câu lệnh bị lỗi, không thể thi hành.

Như vậy, để kiểm soát giao dịch trong T-SQL, ta sử dụng câu lệnh **BEGIN TRANSACTION** để bắt đầu một giao dịch, và gọi lệnh **Commnit** khi giao dịch hoàn tất thành công, và chúng ta sử dụng lệnh **ROLLBACK TRANSACTION** để hủy giao dịch. Hãy nhớ, chế độ **Auto-committed transaction** sẽ bị ghi đè để trở thành một Giao dịch được chỉ định một cách tường minh "**Explicit Transaction**"

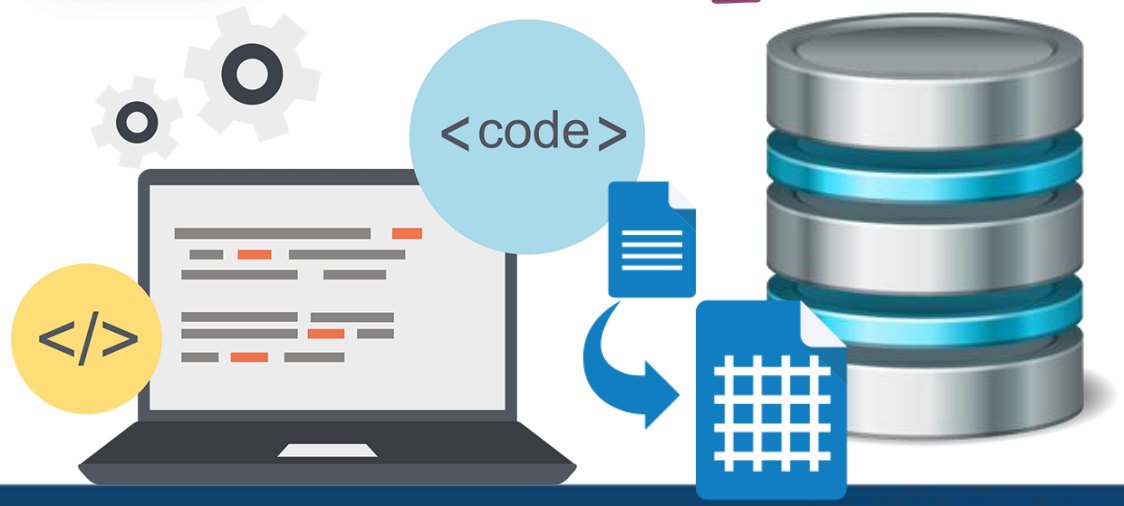
How to Code Transactions

- **BEGIN TRANSACTION**: This marks the beginning of a transaction.
- **COMMIT TRANSACTION**: This marks the successful end of a transaction. It signals the database to save the work.
- **ROLLBACK TRANSACTION**: This denotes that a transaction hasn't been successful and signals the database to roll back to the state it was in prior to the transaction.

Order online context



- ✓**Customer information**(name, gender, phoneNumber, customerAddres, …)
- ✓**Order information**(orderId, custName, orderDate, deliveryDate, deliveryAdd, paymentMethod, …)
- ✓**Order details**(productId, quantity, price, discount, …)



Coding with Transaction

```
-- 1/- Start the Order transaction
Begin Transaction
    -- Set the safety code by using Try ... Catch Block
    Begin Try
        -- 1.1/- Add new customer information -----------------------------------
        insert into khachHang(maKH, tenKH, gioiTinh, ngaySinh, soDT, email, diaChi)
        values ('KH002', N'Nguyễn Quang Hưng',1,'2000/10/28','0705101028','nqhung@bodua.com',
               N'113 Tên Lửa, P.An Lạc, Q.Bình Tân, TP.Hồ Chí Minh');
        print 'The customer information was inserted !.'
        -- 1.2/- Add order information ------------------------------------------
        insert into donHang(soDH, maKH, taiKhoan, ngayDat, ngayGH, diaChiGH)
        values('DH010','KH002','NVGH_01','2020/08/01','2020/08/03',
              N'472/4 Cách Mạng Tháng 8, P.11, Q.3, TP.Hồ Chí Minh');
        print 'The customer information was inserted !.'
        -- 1.3/- Add order details ----------------------------------------------
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '01',1, (select giaBan from sanPham where maSP='01'),
                                 (select giamGia from sanPham where maSP='01'));
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '04',1, (select giaBan from sanPham where maSP='04'),
                                 (select giamGia from sanPham where maSP='04'));
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '05',3, (select giaBan from sanPham where maSP='05'),
                                 (select giamGia from sanPham where maSP='05'));
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '08',1, (select giaBan from sanPham where maSP='08'),
                                 (select giamGia from sanPham where maSP='08'));
    End Try
    Begin Catch
        -- 2/- Rolled-Back when the transaction get the failed action
        print 'Transaction Abort'
        Rollback transaction;
    End Catch
    -- 3/- Commite the transaction whether completted succesfully
    If (@@ERROR = 0 and @@TRANCOUNT > 0)
    Begin
        print 'The transaction Committed !.'
        Commit transaction;
    End
```

1

2

3

- **Try block** to process actions to insert into data to 3 table: khachHang, donHang, ctDonHang
- **Catch block** to rollback the transaction when it get failed
- **If block** to commit transaction if get no error and trancount>0

Begin transaction and Try block

```
-- 1/- Start the Order transaction
Begin Transaction
    -- Set the safety code by using Try ... Catch Block
    Begin Try
        -- 1.1/- Add new customer information -----------------------------------
        insert into khachHang(maKH, tenKH, gioiTinh, ngaySinh, soDT, email, diaChi)
        values ('KH002', N'Nguyễn Quang Hưng',1,'2000/10/28','0705101028','nqhung@bodua.com',
               N'113 Tên Lửa, P.An Lạc, Q.Bình Tân, TP.Hồ Chí Minh');
        print 'The customer information was inserted !.'
        -- 1.2/- Add order information ----------------------------------------
        insert into donHang(soDH, maKH, taiKhoan, ngayDat, ngayGH, diaChiGH)
        values('DH010','KH002','NVGH_01','2020/08/01','2020/08/03',
              N'472/4 Cách Mạng Tháng 8, P.11, Q.3, TP.Hồ Chí Minh');
        print 'The customer information was inserted !.'
        -- 1.3/- Add order details --------------------------------------------
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '01',1, (select giaBan from sanPham where maSP='01'),
                                 (select giamGia from sanPham where maSP='01'));
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '04',1, (select giaBan from sanPham where maSP='04'),
                                 (select giamGia from sanPham where maSP='04'));
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '05',3, (select giaBan from sanPham where maSP='05'),
                                 (select giamGia from sanPham where maSP='05'));
        insert into ctDonHang(soDH, maSP, soLuong, giaBan, giamGia)
        values ('DH010', '08',1, (select giaBan from sanPham where maSP='08'),
                                 (select giamGia from sanPham where maSP='08'));
    End Try
```

Catch block

```
    Begin Catch
        -- 2/- Rolled-Back when the transaction get the failed action
        print 'Transaction Abort'
        Rollback transaction;
    End Catch
    -- 3/- Commite the transaction whether completted succesfully
```

If block

```
-- 3/- Commite the transaction whether completted succesfully
If (@@ERROR = 0 and @@TRANCOUNT > 0)
Begin
    print 'The transaction Committed !.'
    Commit transaction;
End
```

- **@@ERROR**: Returns the error number for the last Transact-SQL statement executed.
- **@@TRANCOUNT**: Returns the number of **BEGIN TRANSACTION** statements that have occurred on the current connection. **ROLLBACK TRANSACTION** decrements **@@TRANCOUNT** to 0
- **print** '***Message !***': Returns a user-defined message to the client..

Nhớ gì ?!!!

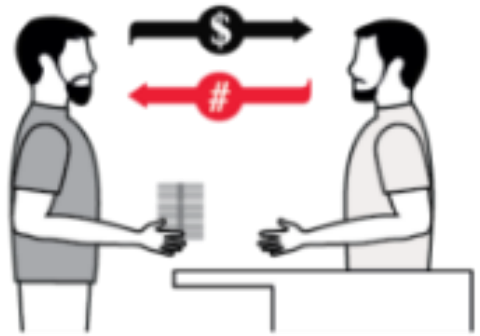
- What is transaction ?
- Understanding the ACID
- How to coding transaction in T-SQL
  - **Begin** transaction
  - **Rollback** transaction
  - **Commit** transaction

Tài liệu tham khảo

- Itzik Ben-Gan, "**Microsoft® SQL Server ® 2012 T-SQL Fundamentals**", O'Reilly Media Inc, 2012
- Itzik Ben-Gan, Dejan Sarka, Ed Katibah, Greg Low, Roger Wolter, and Isaac Kunen, "**Inside Microsoft SQL Server 2008: T-SQL Programming**", Microsoft Press, 2010

Transfer cash via HAWALADAR service

STEP 1
In Country A...



SENDER HAWALADAR A

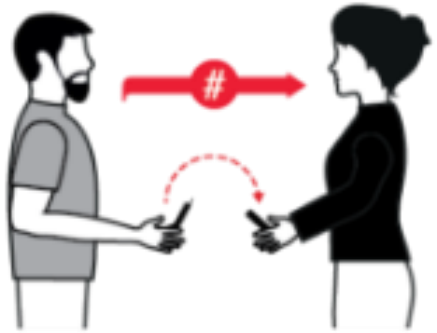
Sender gives cash to hawala agent (Hawaladar A). Agent gives sender a code.

STEP 2
Hawaladar A to B...

HAWALADAR A HAWALADAR B

Hawaladar A tells a counterpart in country B how much cash has been received.

STEP 3
Sender to Recipient...



SENDER RECIPIENT

Sender passes the code to the recipient, saying how much cash was handed in.

STEP 4
In Country B...

HAWALADAR B RECIPIENT

Recipient gives code to Hawaladar B who hands over cash, minus fee. Hawala agents settle their account separately.

Online shopping

1. Customer **places an Order**
2. Order **received by Warehouse**
3. **Fulfillment** Pic, Pack, Ship
4. Order is **Delivered** to customer
5. **Happy** customer

Client – Server in transaction