

# ***Chương 6: Giới thiệu về giao tác***

30-11-2006



*Trường Đại học Khoa học Tự nhiên  
Khoa Công nghệ Thông tin  
Bộ môn Hệ thống Thông tin*

# ***Giao tác (transaction)***

- Ví dụ : chuyển khoản 100\$ từ tài khoản A sang tài khoản B (50\$). Các bước thực hiện gồm :
  - Trừ 100\$ khỏi tài khoản A
  - Nếu số dư trong A  $\geq$  100\$ thì cộng 50\$ vào tài khoản B
- Giả sử vừa trừ tiền khỏi tài khoản A thì sự cố kỹ thuật xảy ra, và các bước tiếp theo không được thực hiện
  - A mất 100\$ nhưng B không nhận được tiền (!?).
- Các bước xử lý nêu trên nếu đã làm thì phải làm cho hết, ngược lại thì không làm bước nào cả → chúng tạo thành một transaction, nói cách khác là một đơn vị công việc nguyên tố.

# Khái niệm

- Giao tác (transaction) là một **tập hợp có thứ tự các thao tác**. Tập hợp này được xem như một đơn vị công việc.
  - *i.e.* tất cả thao tác trong giao tác phải được thực hiện thành công, hoặc không thao tác nào được thực hiện.
  - nếu có một thao tác không hoàn thành được thì toàn bộ giao tác cũng không hoàn thành.
- Giao tác chuyển CSDL từ tình trạng nhất quán này sang tình trạng nhất quán khác. Ngoài ra còn có tính chất ACID sẽ bàn đến dưới đây.

# Khái niệm (tt)

- **Một số thuật ngữ liên quan đến giao tác**
  - Begin [transaction/tran] : bắt đầu một transaction
  - Commit [transaction/tran] : hoàn tất một transaction
  - Rollback [transaction/tran] : quay lui, hủy bỏ toàn bộ phần giao tác đã thực hiện trước đó

# ***Tính chất của giao tác: ACID***

- **Tính nguyên tử (Atomic)** : tất cả thao tác trong giao tác phải được thực hiện thành công, hoặc không thao tác nào được thực hiện.
- **Tính nhất quán (Consistency)** : Chuyển CSDL từ tình trạng nhất quán này sang tình trạng nhất quán khác.

# Tính chất của giao tác (tt)

- **Tính độc lập (Isolation)** : xử lý của một giao tác phải độc lập với những tác động (thấy or thay đổi dữ liệu) của các giao tác khác thực hiện đồng thời
- **Tính bền vững (Durability)** : Sau khi giao tác commit thành công, tất cả những thay đổi trên CSDL mà giao tác đã thực hiện phải được ghi nhận chắc chắn (vào ổ cứng).
  - HQT CSDL luôn phải có cơ chế phục hồi dữ liệu để đảm bảo điều này, thường dùng cơ chế ghi nhận bằng *transaction log*.

## Ghi chú : Đảm bảo tính Atomic

- Để đảm bảo tính chất A của giao tác  
Người sử dụng phải tường minh điều khiển sự rollback của giao tác.
- Cần kiểm tra lỗi sau khi thực hiện mỗi thao tác trong giao tác để có thể xử lý rollback kịp thời.
  - Trong SQL Server, dùng biến toàn cục @@error và @@rowcount

# Cài đặt thêm sinh viên

- **Dùng Stored:**

Create proc....

Begin

*/\*Goi lệnh insert thêm mới vào SinhVien\*/*

*/\* Đọc và kiểm tra SiSo lớp \*/*

*/\* Nếu SiSo >= Max , báo lỗi \*/*

rollback transaction

*/\*Cập nhật tăng sĩ số\*/*

*/\* Nếu có lỗi khi thực hiện thao tác :\*/*

rollback transaction

End



## Ví dụ - kiểm lỗi bằng @@error

```
Create proc sp_ThemSV    @MaSV int, @MaLop int
As
    begin transaction
        declare @SiSo int
        select @SiSo = SiSo
            from LOP
            where MaLop    = @MaLop
        if( @SiSo >= Max)
        begin
            rollback transaction
            return
        end
    end
```

## Ví dụ (tt)

```
Insert into SINH_VIEN(MaSV,MaLop)
      values(@MaSV,@MaLop)
if(@@error<>0)
begin
      rollback transaction
      return
end
```

## Ví dụ (tt)

update LOP

set SiSo = SiSo+1

where MaLop = @MaLop

if(@@error<>0)

begin

rollback transaction

return

end

commit transaction */\* hoàn tất giao tác\*/*

*/\*end stored proc\*/*

## *Các vấn đề liên quan đến xử lý truy xuất đồng thời*

- **Lost Update (mất dữ liệu cập nhật)**
- **Dirty read (đọc dữ liệu “rác”)**
- **Unrepeatable read (không thể đọc lặp lại)**
- **Phantom (“bóng ma”)**

- **Mất dữ liệu cập nhật (Lost update)**

- Tình trạng này xảy ra khi có **nhiều hơn một giao tác cùng thực hiện cập nhật trên 1 đơn vị dữ liệu**. Khi đó, tác dụng của giao tác cập nhật thực hiện sau sẽ đè lên tác dụng của thao tác cập nhật trước.

- **Đọc dữ liệu chưa commit (Uncommitted data, Dirty read)**

- Xảy ra khi **một giao tác thực hiện đọc** trên một đơn vị dữ liệu **mà đơn vị dữ liệu này đang bị cập nhật bởi một giao tác khác** nhưng việc cập nhật chưa được xác nhận đã hoàn tất.

- **Thao tác đọc không thể lặp lại (Unrepeatable data)**

- Tình trạng này xảy ra khi **một giao tác T1 vừa thực hiện xong thao tác đọc** trên một đơn vị dữ liệu (nhưng chưa commit) thì giao tác khác (T2) lại thay đổi (ghi) trên đơn vị dữ liệu này. Điều này làm cho lần đọc sau đó của T1 không còn nhìn thấy dữ liệu ban đầu nữa.

- **Bóng ma (Phantom)**

- Là tình trạng mà **một giao tác đang thao tác trên một tập dữ liệu** nhưng giao tác khác lại chèn thêm hoặc xóa đi các dòng dữ liệu vào tập dữ liệu mà giao tác kia quan tâm.

# Cách giải quyết các vấn đề

- **Dùng khái niệm giao tác**
- **và Dùng cơ chế khoá**
  - Write lock (exclusive lock)
  - Read lock (shared lock)
  - Các hệ quản trị cụ thể còn có những loại khoá mở rộng khác : updlock, holdlock,...
  - **Đặt khóa vào đơn vị dữ liệu**
    - ✓ Tự động (đặt mức cô lập cho giao tác)
    - ✓ Thủ công (đặt cấp độ khóa trong câu SELECT)

# ***Nội dung trình bày***

- **Giao tác (Transaction)**
- **Xử lý đồng thời (Concurrency)**
- **Chế độ khóa**
- **Khai báo tường minh giao tác**
- **Mức cô lập**
- **Các cấp độ khóa**
- **Dead-lock**












# Chế độ khóa

- **Các loại khóa**

- Khóa chia sẻ (shared lock) : Còn gọi là khóa đọc (read lock) . Gọi tắt : Khóa S
- Khóa dự định ghi (Intend to write lock) : Còn gọi là khóa cập nhật (update lock) . Gọi tắt : Khóa U
- Khóa độc quyền (exclusive lock) : Còn gọi là khóa ghi (write lock). Gọi tắt : Khóa X.
  - ✓ Khóa X luôn được phát ra khi ghi, bất kể thông số hệ thống đang thiết lập thế nào.
  - ✓ Sở dĩ như vậy vì HQT hỗ trợ xử lý đồng thời các thao tác đọc nhưng ghi phải ghi lần lượt.

# Chế độ khóa

- Bảng tương thích giữa các chế độ khóa

<i>gt T vs gt T'</i>	S	U	X
S			
U			
X			

Tương thích: T' không phải chờ T

Không tương thích: T' phải chờ T giải phóng khóa

# Chế độ khóa

- Khi một transaction  $T_i$  cần truy cập và thao tác trên một đơn vị dữ liệu  $X$ , nó sẽ đòi phát khóa  $A$  trên  $X$ .
- Nhưng khi ấy nếu transaction  $T_j$  đang giữ khóa  $B$  trên  $X$  (và khóa  $A$  với khóa  $B$  không tương thích) thì  $T_i$  phải đợi  $T_j$  giải phóng khóa  $B$  trên  $X$  nó mới phát được khóa  $A$  trên  $X$  → Hiện tượng chờ đợi lẫn nhau.

# ***Nội dung trình bày***

- **Giao tác (Transaction)**
- **Xử lý đồng thời (Concurrency)**
- **Chế độ khóa**
- **Khai báo tường minh giao tác**
- **Mức cô lập**
- **Các cấp độ khóa**
- **Dead-lock**

## ***Khai báo tường minh giao tác***

- Trong SQL Server, ta có thể khai báo tường minh các giao tác, có thể là trên một khối lệnh độc lập hay trong thân một thủ tục thường trú.
- Ngoài ra SQL Server còn có thể phát sinh các giao tác ngầm định (Ví dụ : Trigger)

# Khai báo tường minh giao tác

- **Các chỉ thị :**

- Begin tran : Đặt trước dòng lệnh đầu tiên của giao tác (1 chỉ thị duy nhất cho 1 giao tác)
- Commit tran : Đặt sau dòng lệnh cuối cùng hoàn tất giao tác (1 chỉ thị duy nhất cho 1 giao tác)
- Rollback tran : Đặt tại các vị trí kiểm lỗi hay các nhánh rẽ logic ứng với trường hợp nghiệp vụ thất bại. (nhiều chỉ thị cho 1 giao tác)

?

# ***Khai báo tường minh giao tác***

- **Kiểm lỗi trong giao tác**

- Lỗi có thể xảy ra sau các thao tác :
  - ✓ Insert, Update (trùng khóa chính, sai kiểu dữ liệu, sai định dạng ngày tháng,...)
  - ✓ Delete (ràng buộc tồn tại,...)
  - ✓ Select (login không có quyền trên object...)
- Sau mỗi thao tác trên phải kiểm lỗi bằng biến hệ thống @@error (=0 → không có lỗi, ≠0 → có lỗi)

# Khai báo tường minh giao tác

- **Kiểm lỗi trong giao tác**

- Khi lỗi xảy ra ( $@@error \neq 0$ ), cần thực hiện các công việc :
  - ✓ Báo lỗi (nếu cần) bằng các lệnh print hay raise error
  - ✓ Rollback tran (bắt buộc)
  - ✓ Nếu Tran khai báo trong SP thì thông thường gọi lệnh *return* để kết thúc SP (tính automic)



# Khai báo tường minh giao tác

- Ví dụ đoạn lệnh kiểm lỗi trong giao tác

```
if @@error <> 0
```

```
Begin
```

```
    Print '...'
```

```
    Rollback tran
```

```
    Return
```

```
End
```

**Khởi lệnh kiểm  
tra lỗi**

## *Nội dung trình bày*

- **Giao tác (Transaction)**
- **Xử lý đồng thời (Concurrency)**
- **Chế độ khóa**
- **Khai báo tường minh giao tác**
- **Mức cô lập**
- **Các cấp độ khóa**
- **Dead-lock**

## ***Mức cô lập cho giao tác***

- **Mục đích: tự động đặt khóa cho các thao tác (đọc) trong kết nối dữ liệu hiện hành.**

# Mức cô lập cho giao tác

- Ghi chú

- Tầm vực của Isolation level là ở mức connection chứ không phải mức transaction.
  - ✓ Khi 1 connection N được đặt mức cô lập X thì X sẽ phát huy hiệu lực trên tất cả các transaction  $T_i$  chạy trên N.
- Mức cô lập chỉ quyết định cách phát và giữ khóa S của transaction (vì khóa X luôn được phát ra khi ghi). Mức cô lập không quan tâm khóa U

## ***Mức cô lập cho giao tác***

- **Bốn mức cô lập mà SQL Server cung cấp**
  - Read Uncommitted : Không phát S khi đọc. Có nhu cầu đọc tức thời, e.g. khi cần đánh giá tổng quát toàn hệ thống. Tuy nhiên, không giải quyết được bất cứ vấn đề xử lý đồng thời nào.
  - Read Committed : Phát S khi đọc, giải phóng S ngay sau khi đọc. Chỉ giải quyết được Dirty Read

## Mức cô lập cho giao tác

- **Bốn mức cô lập mà SQL Server cung cấp**
  - Repeatable Read : Phát S khi đọc và giữ S đến khi transaction kết thúc. Không ngăn chặn lệnh insert/delete dữ liệu thoả điều kiện thiết lập S. Giải quyết được Dirty Read và Unrepeatable Read
  - Serializable : Giống Repeatable Read nhưng có ngăn chặn lệnh insert/delete dữ liệu thoả điều kiện thiết lập S. Giải quyết được Dirty Read và Unrepeatable Read và Phantom

# Mức cô lập cho giao tác

- **Thiết lập :**

- Đặt lệnh thiết lập cần thiết trước khi bắt đầu giao tác.
- Thiết lập mặc định là Read Committed.
- Lệnh thiết lập :
  - ✓ Set transaction Isolation level Tên\_MứcCôLập
- Ví dụ :
  - ✓ Set transaction Isolation level Read Committed
  - ✓ Set transaction Isolation level Serializable

## *Nội dung trình bày*

- **Giao tác (Transaction)**
- **Xử lý đồng thời (Concurrency)**
- **Chế độ khóa**
- **Khai báo tường minh giao tác**
- **Mức cô lập**
- **Các cấp độ khóa**
- **Dead-lock**



# Các cấp độ khóa

- **Đặt vấn đề : Mức cô lập là chưa đủ**
  - Mức cô lập quyết định cách phát và giữ khóa S trong một transaction và có hiệu lực trên tất cả các thao tác đọc trong transaction đó.
  - Thực tế, ta cần phát và giữ khóa S theo các cách khác nhau cho các thao tác đọc khác nhau trong cùng một transaction
  - Ngoài ra, ta cũng cần dùng nhiều dạng khóa linh động hơn là chỉ 1 loại khóa S đơn giản

# Các cấp độ khóa

- **Khái niệm**

- Cấp độ khóa là các loại khóa khác nhau (không chỉ khóa S) được gắn vào từng table trong mệnh đề from của từng thao tác select
- Ngoài lệnh select, cấp độ khóa còn có thể gắn vào các câu lệnh cập nhật, tuy nhiên ở đây ta chỉ quan tâm câu select

# ***Các cấp độ khóa***

- **Các cấp độ khóa**
  - Read Uncommitted / No lock
  - Read Committed (mặc định)
  - Repeatable
  - Serializable / Hold lock
  - Updlock
  - Tablock
  - TablockX
  - ReadPast.....

# Các cấp độ khóa

- **Cách thiết lập**

Select ...

From {Tab1 Alias1 with Lock\_mode [...n]} [...n]

Where ...

- **Ví dụ :**

Select SV.HoVaTen, K.TenKhoa

From SinhVien SV with ReadCommite,

Khoa K with Updlock

Where SV.Khoa = K.Ma And Year(SV.NgaySinh) >= 1983

- **Phối hợp với Isolation Level**

- Trong transaction luôn có các thao tác yêu cầu bảo vệ nghiêm ngặt và các thao tác ít yêu cầu bảo vệ nghiêm ngặt
- Dùng Isolation level ứng với yêu cầu bảo vệ ít nghiêm ngặt nhất
- Bổ sung lock mode vào các thao tác yêu cầu bảo vệ nghiêm ngặt hơn mức mà Isolation level đó cung cấp.

# Các cấp độ khóa

- **Khóa với dữ liệu trong cursor**

- Nếu cursor là loại tĩnh (static) thì các đơn vị dữ liệu đọc ra sẽ được lock ngay khi vừa Open cursor
- Nếu cursor là loại động (dynamic) thì fetch đến đâu sẽ khóa đến đó
- Cách phát khóa và giữ khóa là do mức cô lập của connection và các lock mode trong câu select định nghĩa cursor quyết định

## ***Nội dung trình bày***

- **Giao tác (Transaction)**
- **Xử lý đồng thời (Concurrency)**
- **Chế độ khóa**
- **Khai báo tường minh giao tác**
- **Mức cô lập**
- **Các cấp độ khóa**
- **Dead-lock**

# ***Dead lock***

- **Khái niệm**

- Khi xử lý đồng thời, không tránh khỏi việc transaction này phải chờ đợi transaction khác
- Nếu vì lý do gì đó mà hai transaction lại chờ lẫn nhau vĩnh viễn, không cái nào trong hai có thể hoàn thành được thì ta gọi đó là hiện tượng Dead Lock



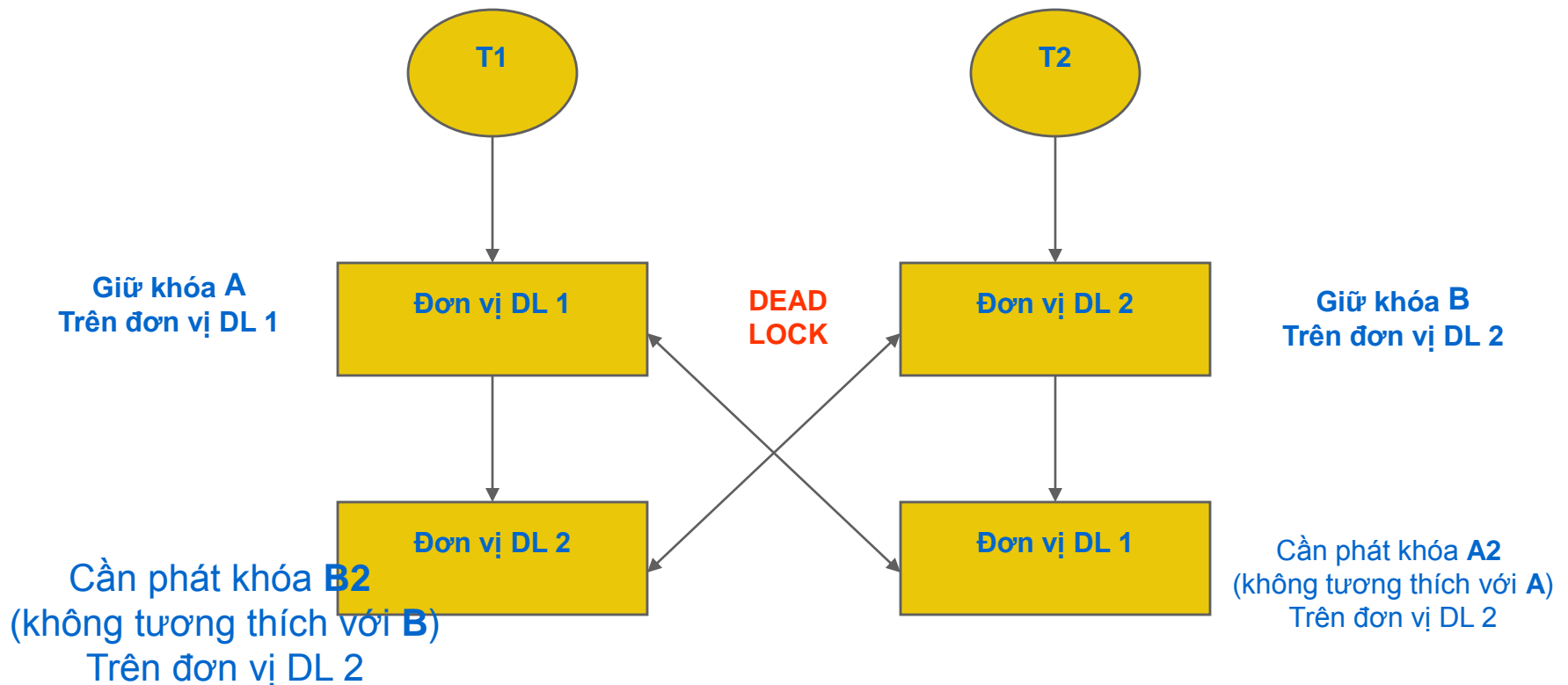
# ***Dead lock***

- **Phân loại**
  - Cyclic Deadlock
  - Conversion Deadlock

# Dead lock

- Phân loại

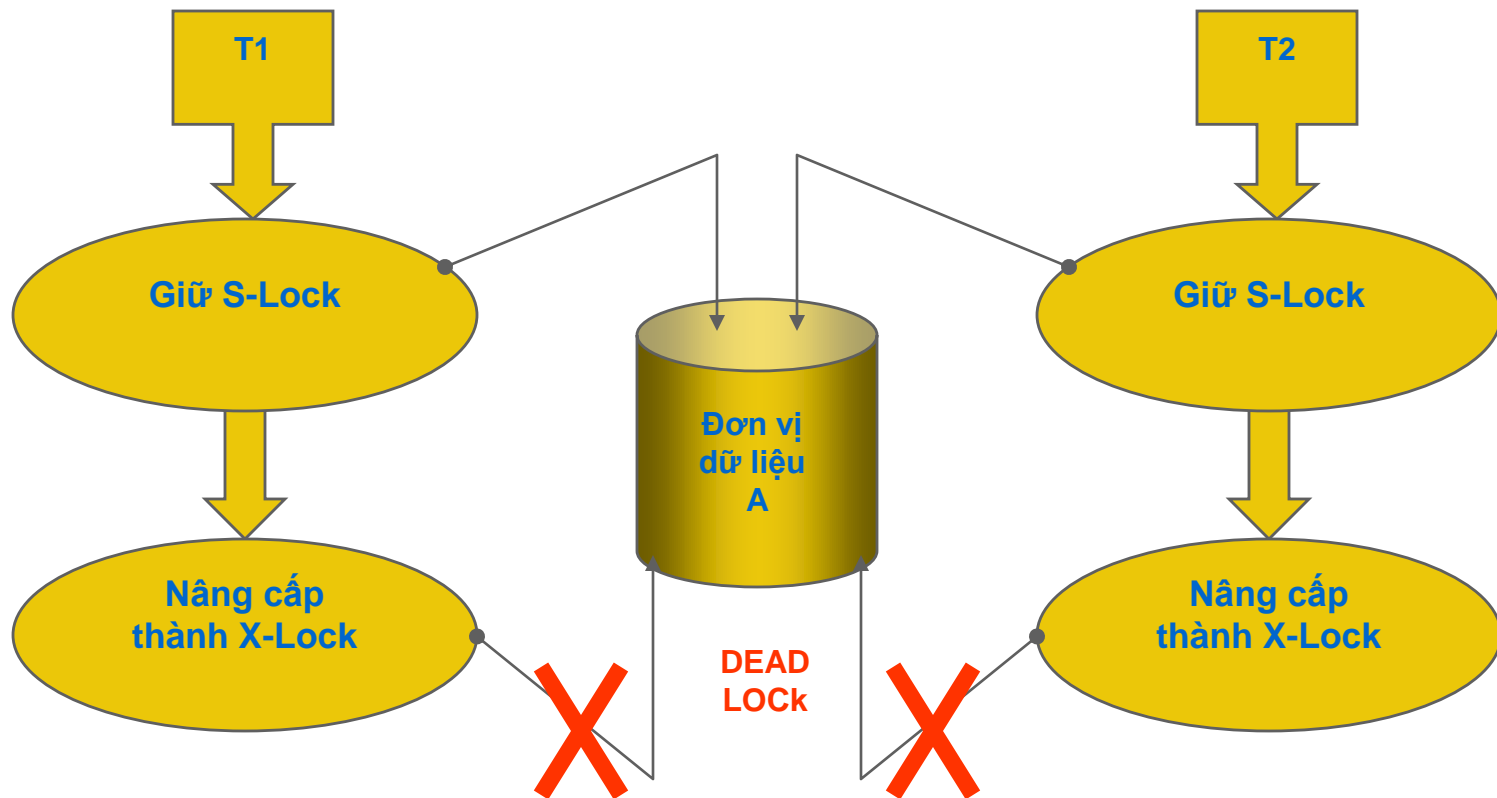
- Cyclic Deadlock



# Dead lock

- Phân loại

- Conversion Deadlock



# Dead lock

- **Khi dead lock xảy ra**
  - SQL Server sẽ chọn 1 trong 2 transaction gây dead lock để hủy bỏ, khi đó transaction còn lại sẽ được tiếp tục thực hiện cho đến khi hoàn tất
  - Transaction bị chọn hủy bỏ là transaction mà SQL ước tính chi phí cho phần việc đã làm được ít hơn transaction còn lại.