



TRƯỜNG ĐẠI HỌC CẦN THƠ  
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  
KHOA CÔNG NGHỆ THÔNG TIN

QUẢN TRỊ DỮ LIỆU - CT467

# Chương 5: **HỆ THỐNG PHỤC HỒI**

Biên soạn:



Ths. Nguyễn Thị Kim Yến



Ntkyen@ctu.edu.vn



CANTHO UNIVERSITY

# NỘI DUNG

1

**Phân lớp hồng học**

2

**Cấu trúc lưu trữ**

3

**Phục hồi dựa trên sổ ghi lộ trình**

4

**Phục hồi với các giao dịch song song**

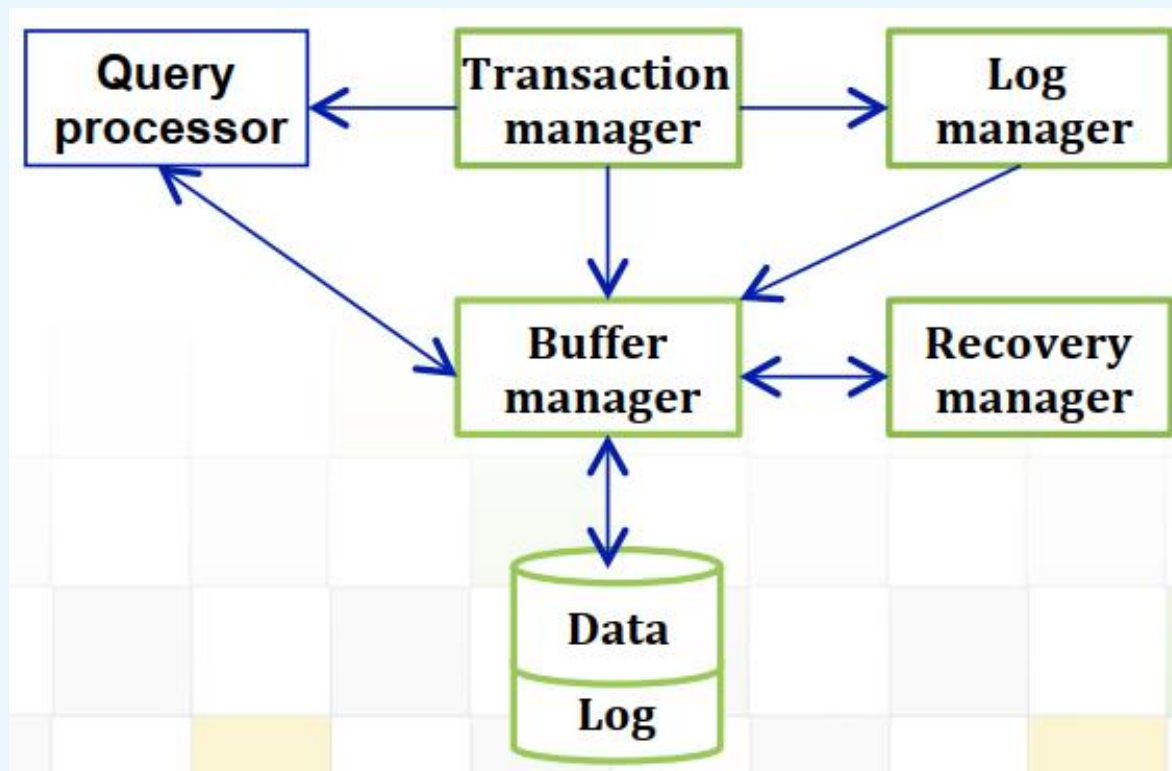
5

**Phân trang bóng**



# Mục tiêu của khôi phục sự cố

- ❖ Đưa DL về trạng thái sau cùng nhất trước khi xảy ra sự cố
- ❖ Đảm bảo 2 tính chất của GD: Nguyên tử và Bền vững





# 1. Phân lớp hỏng hóc

❖ Một số loại hỏng hóc sau:

## Hỏng hóc trong GD

- **Lỗi luận lý:** DL đầu vào, tràn,...
- **Lỗi hệ thống:** deadlock, cạnh tranh

## Hệ thống bị hư hỏng

- Có 1 phần cứng sai chức năng
- Hoặc có 1 sai sót trong phần mềm CSDL hay HĐH

## Đĩa bị hư hỏng

- Một khối đĩa bị mất nội dung.



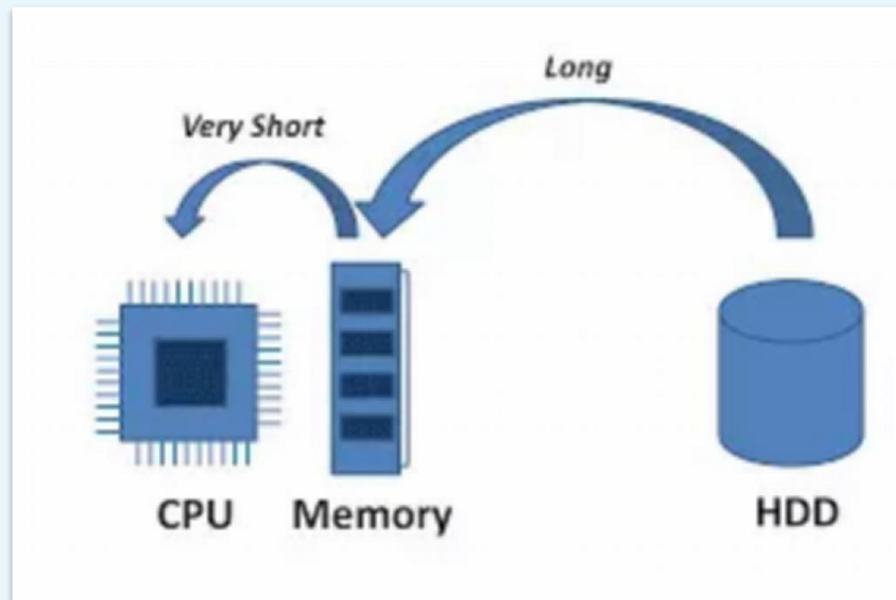
# 1. Phân lớp hỏng hóc (tt)

- **Phục hồi:** Xác định loại hư hỏng  $\Rightarrow$  đánh giá sự ảnh hưởng đến dữ liệu  $\Rightarrow$  đề xuất giải pháp **đảm bảo** tính bền vững và nguyên tử  $\Rightarrow$  **giải thuật phục hồi.**
- **Giải thuật phục hồi:** bao gồm 2 phần
  - Các hoạt động **trong quá trình** các GD thực hiện  $\Rightarrow$  thu thập thông tin cần thiết
  - Các hoạt động **sau khi lỗi phát sinh**  $\Rightarrow$  đảm bảo tính nguyên tử và bền vững



## 2. Cấu trúc lưu trữ

- Lưu trữ không ổn định (bay hơi): DL sẽ bị mất
- Lưu trữ ổn định (không bay hơi): DL không bị mất
- Lưu trữ bền: *không bao giờ bị mất* (tăng cường độ bền)





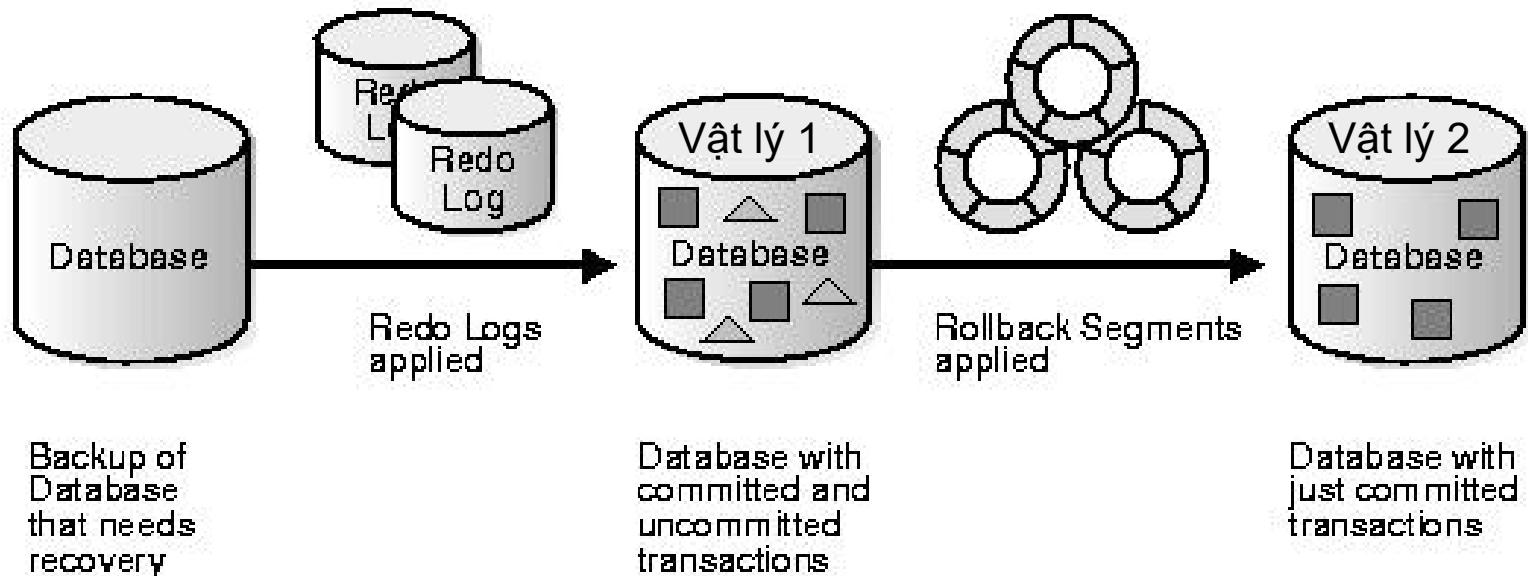
## 2.1 Thực thi lưu trữ bền

- ❖ Việc chuyển khối DL giữa bộ nhớ và đĩa có thể dẫn đến kết quả:
  - **Thành công hoàn toàn:** DL chuyển đến đích an toàn
  - **Bị lỗi 1 phần:** Có lỗi trong quá trình chuyển DL và khối đích chưa có thông tin đúng
  - **Bị lỗi hoàn toàn:** Lỗi xuất hiện ngay ở giai đoạn đầu của quá trình truyền DL. Khối đích giữ nguyên như ban đầu



## 2.1 Thực thi lưu trữ bền (tt)

- Một thao tác ghi được thực thi như sau:



So sánh từng khối đĩa vật lý => **mất nhiều chi phí**

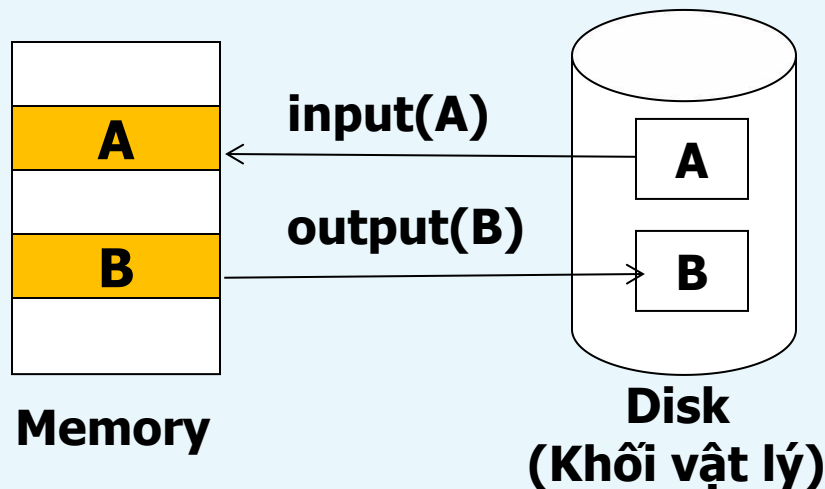
Lưu vết các thao tác ghi khối đĩa





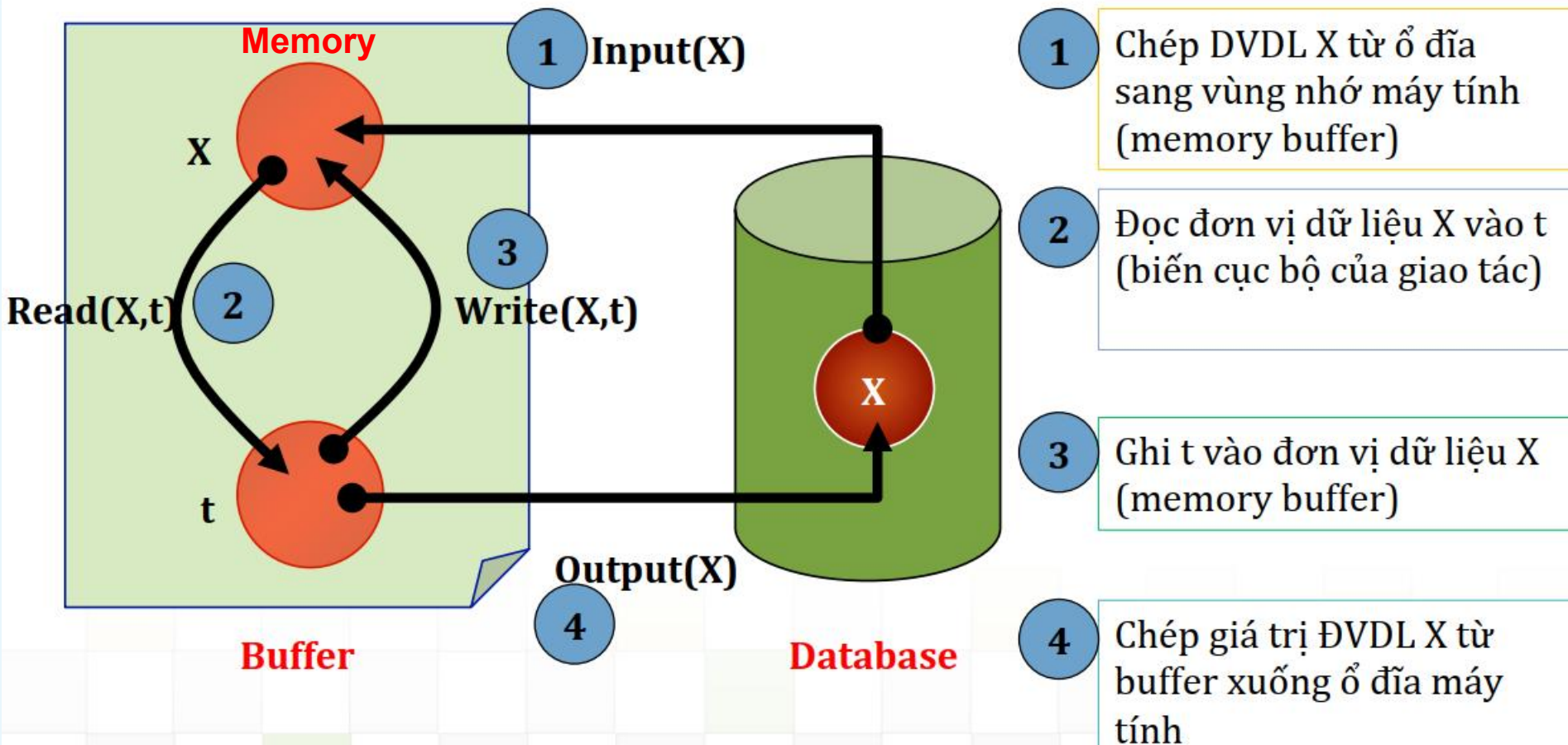
## 2.2 Truy cập dữ liệu

- ❖ Di chuyển giữa Disk và Memory thực hiện qua 2 thao tác:
- **Input(A)**: chuyển khối vật lý A vào bộ nhớ đệm
  - **Output(B)**: chuyển khối đệm B từ bộ nhớ đệm ra đĩa và thay thế nội dung của khối đĩa vật lý tương đương



## 2.2 Truy cập dữ liệu (tt)

- Thao tác đọc/ghi khối đĩa:





### 3. Phục hồi dựa trên Sổ ghi lộ trình (log)

- Dùng **sổ ghi lộ trình** ghi nhận lại các thay đổi CSDL
- **Log** là 1 dãy các mẫu tin lộ trình (log record)
- Một thao tác **cập nhật** trên CSDL sẽ ghi nhận bằng 1 **log record**
- Các loại log record:

Loại Log Record	Ý nghĩa
$\langle T_i, \text{Start} \rangle$	GD $T_i$ đã khởi động.
$\langle T_i, X, V_1, V_2 \rangle$	GD $T_i$ thay đổi giá trị của $X$ từ $V_1$ thành $V_2$
$\langle T_i, \text{commit} \rangle$	GD $T_i$ đã bàn giao.
$\langle T_i, \text{abort} \rangle$	GD $T_i$ đã hủy bỏ.



## 3.1 Sự cập nhật bị trì hoãn

- Dừng khi các GD được thực hiện **tuần tự**
- Đảm bảo tính nguyên tử và tính bền vững của GD
- Tất cả các thao tác cập nhật CSDL sẽ **bị trì hoãn**
- **Hoạt động ghi nhật ký** của giải thuật thực hiện như sau:
  - Trước khi  $T_i$  **khởi động**:  $\langle T_i, \text{Start} \rangle$
  - Trước khi  $T_i$  **Write(X)**:  $\langle T_i, X, V_2 \rangle$
  - Khi  $T_i$  **bàn giao một phần**:  $\langle T_i, \text{commit} \rangle$



## 3.1 Sự cập nhật bị trì hoãn (tt)

- **Hoạt động phục hồi:**
  - $\forall T_i: \langle T_i, \text{start} \rangle$  và  $\langle T_i, \text{commit} \rangle$  có trong sổ ghi lộ trình  
 $\Rightarrow$  **redo( $T_i$ )**
  - $\forall T_i: \langle T_i, \text{start} \rangle$  có trong sổ ghi lộ trình nhưng **không có**  $\langle T_i, \text{commit} \rangle \Rightarrow$  **bỏ qua**
  - **Thao tác redo( $T_i$ ):** **Dò xuôi** sổ ghi lộ trình từ trên xuống, khi gặp mẫu tin có dạng  $\langle T_i, X, V_2 \rangle$  thì thực hiện gán giá trị  $V_2$  cho hạng mục dữ liệu  $X$ .



## 3.1 Sự cập nhật bị trì hoãn (tt)

- Ví dụ: Bỏ qua  $T_1$

$T_1$	$T_2$
R(A)	R(C)
$A=A-50$	$C=C-100$
W(A)	W(C)
R(B)	
$B=B+50$	
W(B)	



$T_1$  thực hiện  
xong W(B)

**$A = 1000, B = 2000, C = 700$**

$\langle T_1, \text{start} \rangle$   
 $\langle T_1, A, 950 \rangle$   
 $\langle T_1, B, 2050 \rangle$   
 $\langle T_1, \text{commit} \rangle$   
 $\langle T_2, \text{start} \rangle$   
 $\langle T_2, C, 600 \rangle$   
 $\langle T_2, \text{commit} \rangle$

Sổ ghi lộ trình


**$A = 1000, B = 2000, C = 700$**

**Bỏ qua GD  $T_1$**  (do GD  $T_1$  chỉ có  $\langle T_1, \text{start} \rangle$  nhưng không có  $\langle T_1, \text{commit} \rangle$  trong SGLT)



## 3.1 Sự cập nhật bị trì hoãn (tt)

- Ví dụ: Thao tác phục hồi  $T_1$  và bỏ qua  $T_2$

$T_1$	$T_2$
R(A)	R(C)
$A=A-50$	$C=C-100$
W(A)	W(C)
R(B)	
$B=B+50$	
W(B)	

Xong  $T_1$  và  $T_2$  thực hiện xong W(C)

$A = 1000, B = 2000, C = 700$

$\langle T_1, \text{start} \rangle$   
 $\langle T_1, A, 950 \rangle$   
 $\langle T_1, B, 2050 \rangle$   
 $\langle T_1, \text{commit} \rangle$   
 $\langle T_2, \text{start} \rangle$   
 $\langle T_2, C, 600 \rangle$   
 $\langle T_2, \text{commit} \rangle$

Sổ ghi lộ trình

$A = 950, B = 2050, C = 700$

**Redo( $T_1$ ) và bỏ qua  $T_2$**  (do  $T_1$  có cả  $\langle T_1, \text{start} \rangle$  và  $\langle T_1, \text{commit} \rangle$  trong SGLT, còn  $T_2$  thì có  $\langle T_2, \text{start} \rangle$  không có  $\langle T_2, \text{commit} \rangle$ )



## 3.2 Sự cập nhật tức thời

- Áp dụng cho cả LT tuần tự và LT cạnh tranh
- Các thao tác thay đổi gtrị các hạng mục CSDL sẽ thể hiện ngay lên CSDL
- **Hoạt động ghi nhật ký** của giải thuật thực hiện như sau:
  - Trước khi  $T_i$  khởi động:  $\langle T_i, \text{start} \rangle$
  - Trước khi  $T_i$  Write(X):  $\langle T_i, X, V_1, V_2 \rangle$
  - Khi  $T_i$  hoàn thành:  $\langle T_i, \text{commit} \rangle$





## 3.2 Sự cập nhật tức thời

- **Hoạt động phục hồi:**
  - $\forall T_i: \langle T_i, \text{start} \rangle$  và  $\langle T_i, \text{commit} \rangle$  có trong sổ ghi lộ trình  
 $\Rightarrow \text{redo}(T_i)$
  - $\forall T_i: \langle T_i, \text{start} \rangle$  có trong sổ ghi lộ trình nhưng **không có**  $\langle T_i, \text{commit} \rangle \Rightarrow \text{undo}(T_i)$
  - **Thao tác  $\text{undo}(T_i)$ :** **Dò ngược** sổ ghi từ dưới lên, tìm các mẫu tin có dạng  $\langle T_i, X, V_1, V_2 \rangle$  và thực hiện gán giá trị  $V_1$  cho  $X$ .



## 3.2 Sự cập nhật tức thời (tt)

◆ **Ví dụ:** Sổ ghi lộ trình - Giải thuật cập nhật tức thời  $T_1 \rightarrow T_2$

$T_1$	$T_2$
R(A)	R(C)
A=A-50	C=C-100
W(A)	W(C)
R(B)	
B=B+50	
W(B)	

**A = 1000, B = 2000, C = 700**

```
<T1, start>
<T1, A, 1000, 950>
<T1, B, 2000, 2050>
<T1, commit>
<T2, start>
<T2, C, 700, 600>
<T2, commit>
```

- ◆ **Redo( $T_i$ ):** đặt **giá trị mới** cho các hạng mục CSDL
- ◆ **Undo( $T_i$ ):** đặt **giá trị cũ** cho các hạng mục CSDL



## 3.2 Sự cập nhật tức thời (tt)

### Trạng thái CSDL

Sổ ghi lộ trình	CSDL
$\langle T_1, \text{start} \rangle$	
$\langle T_1, A, 1000, 950 \rangle$	
$\langle T_1, B, 2000, 2050 \rangle$	
<del>✗</del>	$A = 950$ $B = 2050$
$\langle T_1, \text{commit} \rangle$	
$\langle T_2, \text{start} \rangle$	
$\langle T_2, C, 700, 600 \rangle$	
<del>✗</del>	$C = 600$
$\langle T_2, \text{commit} \rangle$	

◆ **Undo( $T_1$ )** (do  $T_1$  có  $\langle T_1, \text{start} \rangle$  nhưng không có  $\langle T_1, \text{commit} \rangle$ )

$A=1000, B=2000$

◆ **Redo( $T_1$ ) và Undo ( $T_2$ )**

$A= 950, B=2050, C=700$



## 3.3 Điểm kiểm soát (checkpoint)

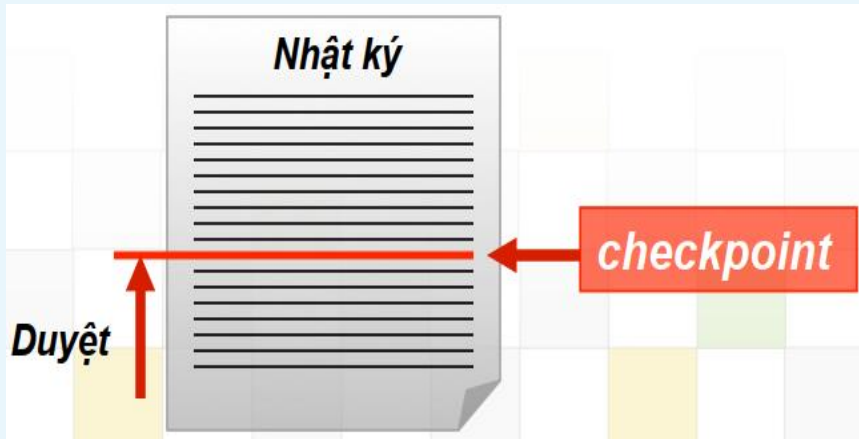
- ◆ Khi bị sự cố, DBMS không thể tra cứu toàn bộ nhật ký vì:
  - Nhật ký tích lũy thông tin về tất cả các hành động của 1 giai đoạn **rất dài**
  - Quá trình tra cứu nhật ký → Phải **quét hết** tập tin nhật ký → **mất nhiều thời gian**
  - Thực hiện lại các GD đã được ghi xuống đĩa làm cho việc khôi phục lặp lại → **tốn thời gian vô ích**

➔ **Giải pháp: Dùng điểm kiểm soát**



## 3.3 Điểm kiểm soát (checkpoint)

- ◆ ĐKS để **cải thiện hiệu năng** của quá trình khôi phục
- ◆ Muốn **đặt điểm kiểm soát** thực hiện như sau:
  1. Ghi log record vào thiết bị lưu trữ bền
  2. Xuất ra đĩa tất cả các khối đệm đã được cập nhật
  3. Thêm **<checkpoint>** vào sổ ghi lộ trình



**Nhược điểm:** Khi hệ thống phục hồi đang đặt ĐKS, thì các GD không được thực hiện bất kỳ thao tác nào.

**Khắc phục:** Điểm kiểm soát mờ



## 3.3 Điểm kiểm soát (checkpoint)

### ❖ Thao tác phục hồi:

1. Quy trình tìm tập  $T$  được thực hiện như sau:

- Dò ngược từ cuối SGLT đến khi gặp **checkpoint** đầu tiên.
- Từ checkpoint này, dò ngược khi gặp  $\langle T_i, \text{start} \rangle$  gần nhất.
- $T$  sẽ bao gồm các GD  $T_i$  và các GD diễn ra sau  $T_i$ .

2. Dựa trên sự cập nhật **tức thời**:

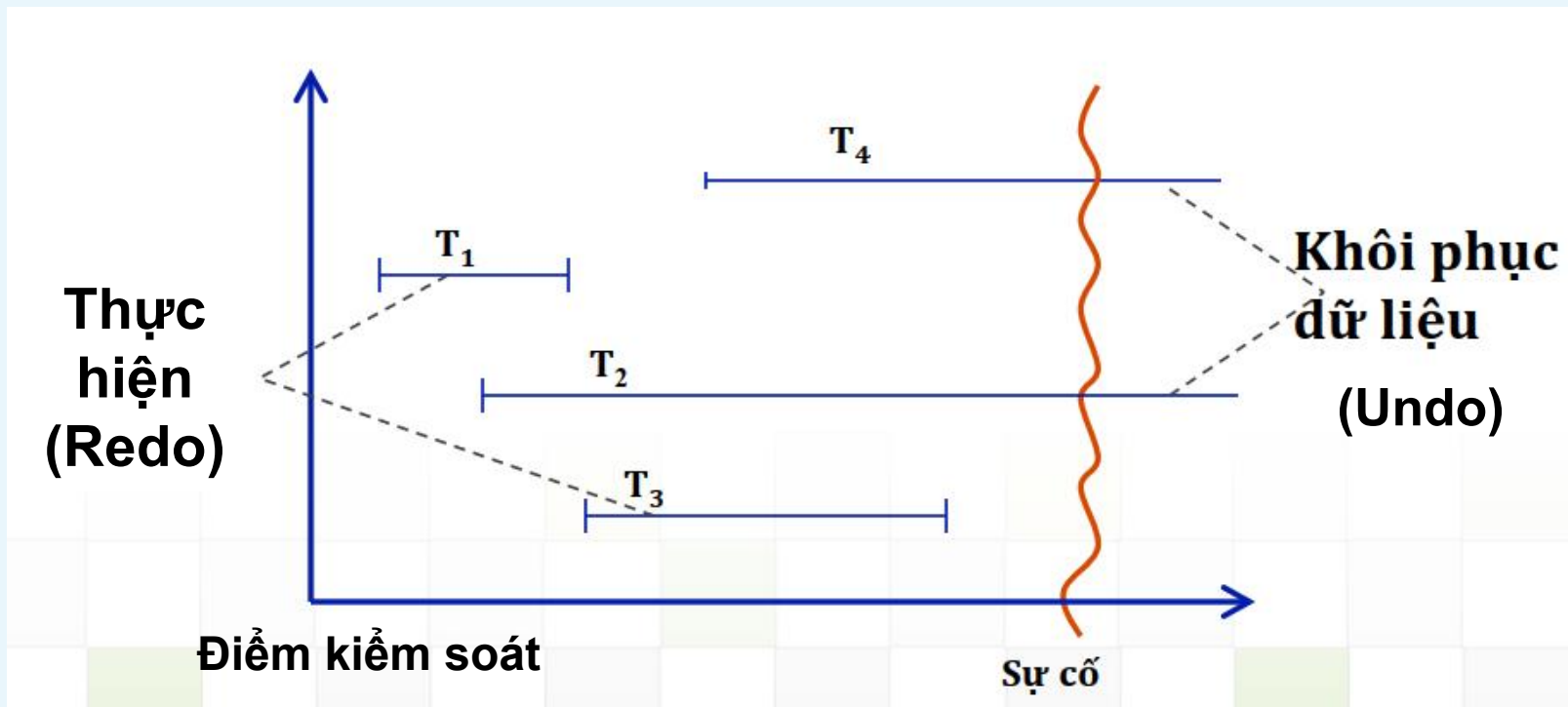
- $\forall T_K \in T$ , nếu  $\langle T_K, \text{commit} \rangle$  **có** trong SGLT  $\Rightarrow \text{redo}(T_K)$
- $\forall T_K \in T$ , nếu  $\langle T_K, \text{commit} \rangle$  **không có** trong SGLT  $\Rightarrow \text{undo}(T_K)$



# GT phục hồi dựa trên sự cập nhật tức thời

- **Khi có sự cố**

- $T_1$  và  $T_3$  đã hoàn tất
- $T_2$  và  $T_4$  chưa kết thúc





## 3.3 Điểm kiểm soát (checkpoint)

### ❖ Thao tác phục hồi (tt):

#### 3. Dựa trên sự cập nhật **trì hoãn**:

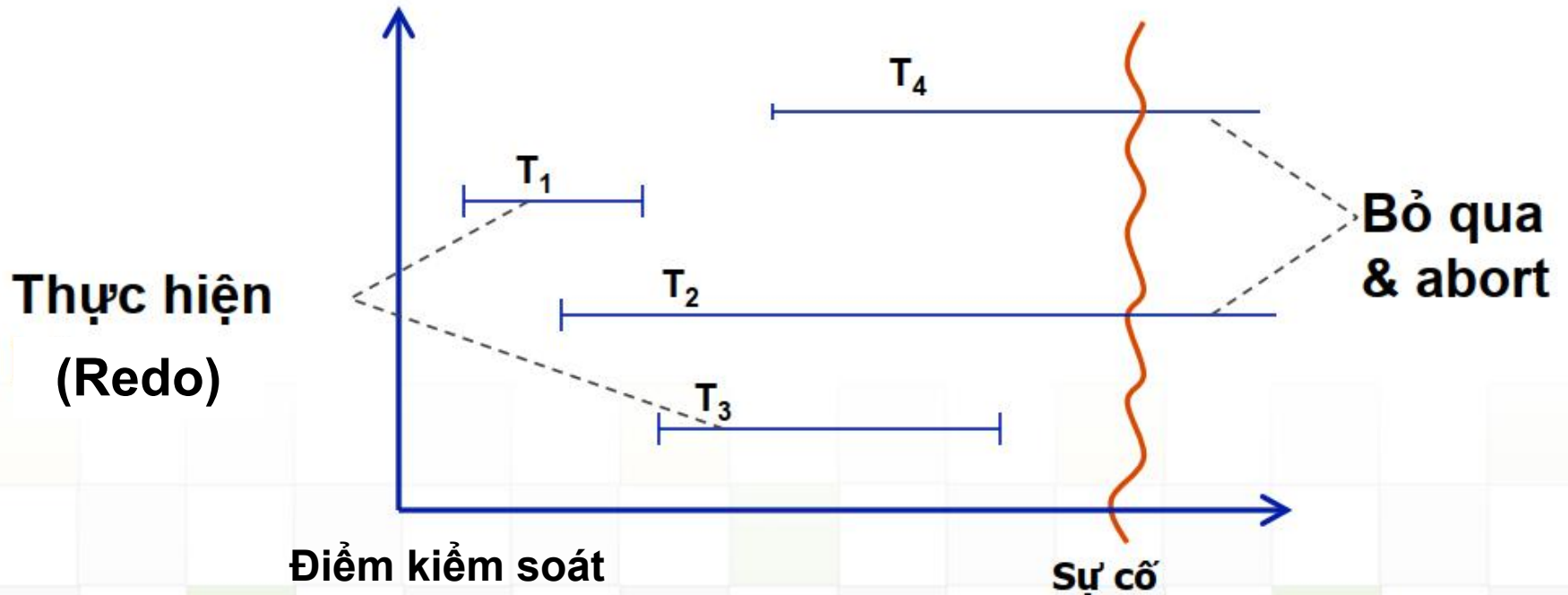
- $\forall T_K \in T$ , nếu  $\langle T_K, \text{commit} \rangle$  **có** trong SGLT  $\Rightarrow$  redo( $T_K$ )
- $\forall T_K \in T$ , nếu  $\langle T_K, \text{commit} \rangle$  **không có** trong SGLT  $\Rightarrow$  bỏ qua ( $T_K$ )





# GT phục hồi dựa trên sự cập nhật trì hoãn

- **Khi có sự cố**
  - $T_1$  và  $T_3$  đã hoàn tất
  - $T_2$  và  $T_4$  chưa kết thúc





## 4. Phục hồi cho các giao dịch song song

- ◆ Dùng sổ ghi lộ trình với sự **cập nhật tức thời**
- ◆ Checkpoint: **<checkpoint, L>** với **L** là danh sách các GD đang thực thi tại thời điểm đặt checkpoint
- ◆ **Hoạt động ghi sổ ghi lộ trình**
  - Trước khi  $T_i$  **khởi động**: **< $T_i$ , start>**
  - Trước khi  $T_i$  **Write(X)**: **< $T_i$ , X,  $V_1$ ,  $V_2$ >**
  - Trước khi  $T_i$  **bàn giao**: **< $T_i$ , commit>**



## 4. Phục hồi cho các GD song song (tt)

### Hoạt động phục hồi:

#### 1. Tạo hai danh sách **redo-list** và **undo-list**:

- a. Khởi tạo **Redo-list** =  $\emptyset$ ; **undo-list** =  $\emptyset$
- b. **Dò ngược** SGLT đến khi gặp **<checkpoint, L>** đầu tiên:
  - Nếu thấy **<T<sub>i</sub>, commit>**:  $\Rightarrow$  thêm T<sub>i</sub> vào **redo-list**
  - Nếu thấy **<T<sub>i</sub>, start>** và T<sub>i</sub>  $\notin$  **redo-list**:  
 $\Rightarrow$  thêm T<sub>i</sub> vào **undo-list**
  - Khi thấy **<checkpoint, L>**:  $\forall T_i \in L$  nếu T<sub>i</sub>  $\notin$  **redo-list** thêm T<sub>i</sub> vào **undo-list**



## 4. Phục hồi cho các GD song song (tt)

### 2. Thực hiện thủ tục **undo**:

- Dò ngược SGLT thực hiện **undo** các GD  $T_i$  trong **undo-list** (với dạng  $\langle T_i, X, V_1, V_2 \rangle$  đặt giá trị  **$V_1$**  cho  $X$ )
- $T_i \notin \text{undo-list} \Rightarrow$  Bỏ qua bước này

### 3. Thực hiện thủ tục **redo**:

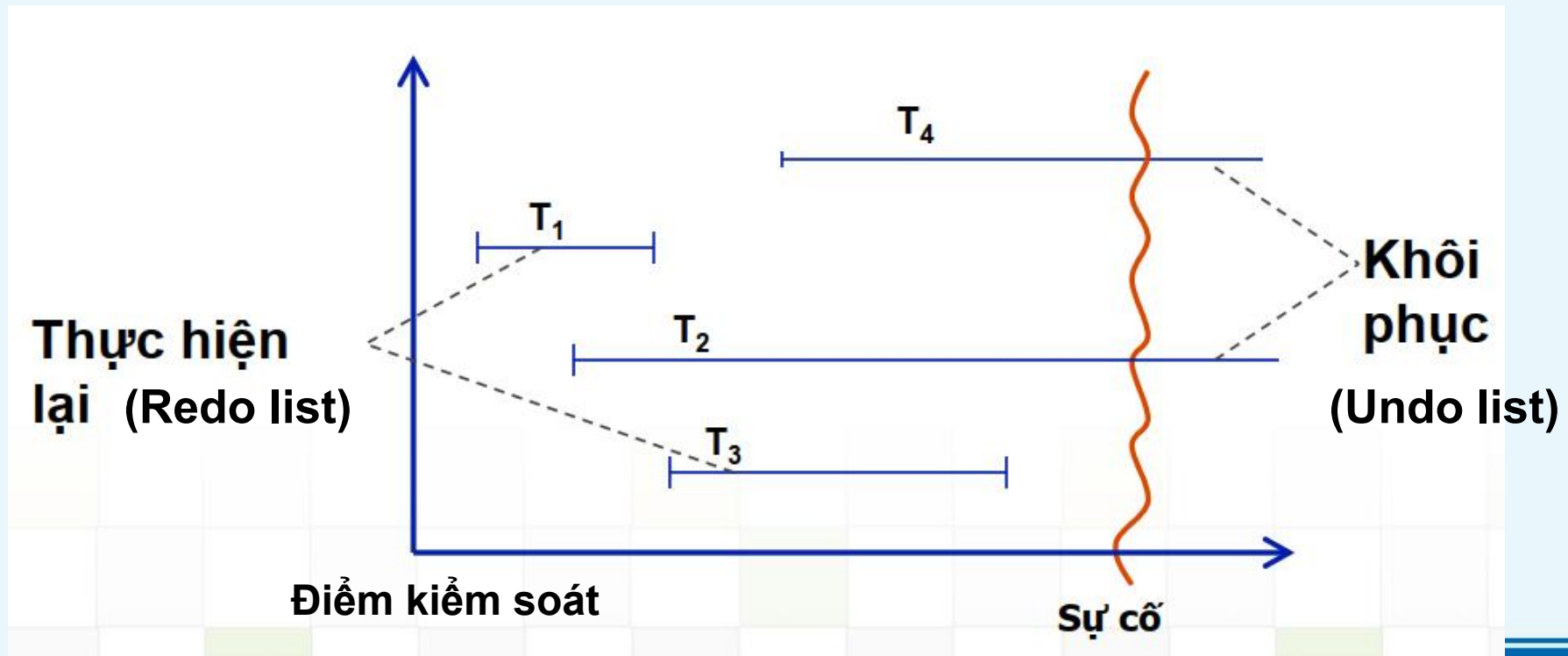
- Định vị  **$\langle \text{checkpoint}, L \rangle$**  cuối cùng trong SGLT
- Dò xuôi SGLT và thực hiện **redo** các GD  $T_i$  trong **redo-list** (với dạng  $\langle T_i, X, V_1, V_2 \rangle$  đặt giá trị  **$V_2$**  cho  $X$ )

**Lưu ý:** Thao tác **Undo** phải được thực hiện **trước** **Redo**



## Ví dụ: Redo list - Undo list

- **Khi có sự cố**
  - $T_1$  và  $T_3$  đã hoàn tất
  - $T_2$  và  $T_4$  chưa kết thúc





## Ví dụ: Phục hồi cho các GD song song

**Cho một sổ ghi lịch trình sau:**

- Hãy mô tả quá trình phục hồi
- Xác định giá trị của A, B, C, D trên đĩa **trước** khi phục hồi
- Xác định giá trị của A, B, C, D trên đĩa **sau** khi phục hồi.

**Giải:**

- ◆ Redo-list =  $\{T_1\}$
- ◆ Undo-list =  $\{T_2\}$
- ◆ Trước PH: A=30, B=35, C=160, D=260
- ◆ Sau PH: A=**10**, B=**15**, C=160, D=260

```
<T1, start>
<T1, A, 30, 10>
<T1, B, 35, 15>
<T1, commit>
<T2, start>
<T2, C, 160, 110>
<T2, D, 260, 210>
<checkpoint {T1, T2}>
<T3, start>
<T2, commit>
<T3, E, 1510, 1010>
.....
```



CANTHO UNIVERSITY

# *HẾT CHƯƠNG 5*