

---

## Chương 5

# Hệ thống phục hồi

## Phân lớp hỏng hóc

---

### □ Hỏng hóc trong giao dịch

- Lỗi luân lý: xảy ra khi giao dịch không thể tiếp tục thực hiện bình thường được nữa do một số điều kiện bên trong không được thỏa.
- Ví dụ: dữ liệu đầu vào không đúng, không tìm thấy dữ liệu, trao dữ liệu hoặc do việc sử dụng tài nguyên vượt hạn định
- Lỗi hệ thống: xảy ra khi hệ thống rơi vào trạng thái không mong muốn. Ví dụ như trạng thái deadlock (trạng thái bế tắc, chờ xoay vòng)

### □ Hệ thống bị hư hỏng

### □ Đĩa bị hư hỏng

---

## Cấu trúc lưu trữ

---

### □ Các loại lưu trữ

- Lưu trữ không ổn định ( volatile storage ): Thông tin lưu trong thiết bị lưu trữ không ổn định sẽ bị mất khi hệ thống bị hỏng hóc. Ví dụ của thiết bị lưu trữ không ổn định là : bộ nhớ chính, bộ nhớ cache.
  - Lưu trữ ổn định ( nonvolatile storage ): Thông tin lưu trữ trong thiết bị lưu trữ ổn định thường không bị mất khi hệ thống bị sự cố.
  - Lưu trữ bền ( stable storage ): Theo lý thuyết thì thông tin chứa trong thiết bị lưu trữ bền không bao giờ bị mất khi hệ thống bị hư hỏng.
- 

## Cấu trúc lưu trữ

---

### □ Thực thi lưu trữ bền

- Nhân bản thông tin cần thiết trong các phương tiện lưu trữ ổn định khác nhau với các phương thức hỏng hóc độc lập.
  - Cập nhật các phiên bản thông tin này một cách có tổ chức, sao cho dù có lỗi xuất hiện trong quá trình chuyển dữ liệu thì thông tin vẫn không bị hư hại.
  - Các hệ thống RAID đảm bảo rằng việc hỏng hóc của một đĩa không gây sự mất dữ liệu. Dạng thức đơn giản và nhanh nhất của RAID là dùng đĩa gương (mirrored disk)
-

## Cấu trúc lưu trữ

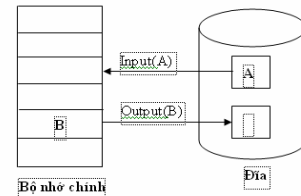
### □ Chuyển khối dữ liệu giữa bộ nhớ và đĩa

- Thành công hoàn toàn
- Bị lỗi một phần
- Bị lỗi hoàn toàn

## Truy cập dữ liệu

### □ Chuyển nhận khối vật lý trên đĩa vào bộ nhớ và khối đệm trong bộ nhớ ra đĩa:

- Input(B): chuyển khối vật lý B vào bộ nhớ chính
- Output(B): chuyển khối đệm B ra đĩa và thay thế cho khối vật lý tương ứng



## Truy cập dữ liệu

### □ Chuyển dữ liệu từ vùng làm việc của hệ thống đến vùng làm việc của giao dịch và ngược lại:

- Read (X)
- Write (X)

### □ Read (X) & Write (X) không yêu cầu chuyển một khối đệm từ bộ nhớ ra đĩa

- Xuất bắt buộc khối đệm ra đĩa: Output(B)

## Phục hồi dựa trên sổ ghi lộ trình

### □ Ví dụ: Giao dịch Ti chuyển \$50 từ tài khoản A sang tài khoản B.

- Giá trị ban đầu của các tài khoản A và B lần lượt là \$1000 và \$2000.
- Giả sử hệ thống bị hư hỏng trong khi Ti đang thực thi.
- Ta có:
  - Nếu thực hiện lại Ti: giá trị của A là .... thay vì phải là .... và giá trị của B là ....
  - Nếu không thực hiện lại Ti: giá trị của A và B tương ứng sẽ là ..... và .....

### □ Ý tưởng để đạt được tính nguyên tử là: trước khi thực hiện các thao tác sửa đổi cơ sở dữ liệu, cần ghi ra các thiết bị lưu trữ bền những thông tin mô tả các sửa đổi này.

## Phục hồi dựa trên sổ ghi lộ trình

- Sổ ghi lộ trình (log): gồm các mẫu tin lộ trình (log record)
- Mẫu tin lộ trình:
  - Định danh giao dịch
  - Định danh hạng mục dữ liệu
  - Giá trị cũ
  - Giá trị mới
- Ví dụ mẫu tin lộ trình:
  - < T<sub>i</sub> start >
  - < T<sub>i</sub>, X<sub>j</sub>, V<sub>2</sub> >
  - < T<sub>i</sub>, X<sub>j</sub>, V<sub>1</sub>, V<sub>2</sub> >
  - < T<sub>i</sub> commit >
  - < T<sub>i</sub> abort >

## Sự hiệu chỉnh CSDL bị trì hoãn

- Ghi lại tất cả những sửa đổi cơ sở dữ liệu vào sổ ghi lộ trình, nhưng trì hoãn sự thực hiện tất cả các thao tác write dữ liệu ra đĩa của giao dịch cho đến khi giao dịch bàn giao một phần (partially commits) .
- Sự thực thi một giao dịch:
  - Trước khi giao dịch T<sub>i</sub> bắt đầu thực hiện, một mẫu tin <T<sub>i</sub> start> được ghi ra sổ ghi lộ trình.
  - Trước khi T<sub>i</sub> thực hiện thao tác **write(X)**, một mẫu tin <T<sub>i</sub>, X, V<sub>2</sub>> được ghi ra sổ ghi lộ trình.
  - Khi giao dịch T<sub>i</sub> bàn giao một phần, mẫu tin <T<sub>i</sub> commit> được ghi ra sổ ghi lộ trình.

## Sự hiệu chỉnh CSDL bị trì hoãn

- Ví dụ:
  - Số tiền ban đầu tài khoản A, B, C là 1000, 2000, 700
  - Giao dịch T<sub>0</sub>: chuyển \$50 từ tài khoản A sang B
  - Giao dịch T<sub>1</sub>: rút \$100 từ tài khoản C
- Giao dịch T<sub>0</sub> thực hiện xong rồi tới T<sub>1</sub>

T <sub>0</sub>	T <sub>1</sub>
read(A)	Read(C)
A = A - 50	C = C - 100
write(A)	write(C)
read(B)	
B = B + 50	
write(B)	

## Sự hiệu chỉnh CSDL bị trì hoãn

- Sổ ghi lộ trình

<T <sub>0</sub> start>	T <sub>0</sub> bắt đầu hoạt động
<T <sub>0</sub> , A, 950>	
<T <sub>0</sub> , B, 2050>	
<T <sub>0</sub> commit>	T <sub>0</sub> hoàn tất
<T <sub>1</sub> start>	T <sub>1</sub> bắt đầu hoạt động
<T <sub>1</sub> , C, 600>	
<T <sub>1</sub> commit>	T <sub>1</sub> hoàn tất

- Giao dịch T<sub>i</sub> cần được làm lại khi và chỉ khi sổ ghi lộ trình chứa cả hai mẫu tin <T<sub>i</sub> start> và <T<sub>i</sub> commit>
- Thủ tục Redo(T<sub>i</sub>): đặt giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T<sub>i</sub>.

## Sự hiệu chỉnh CSDL tức thời

- Các thao tác sửa đổi CSDL có quyền xuất dữ liệu ra đĩa tức thời trong khi giao dịch vẫn còn ở trong trạng thái hoạt động (active state).
- Sự thực thi một giao dịch:
  - Trước khi giao dịch Ti bắt đầu sự thực hiện, một mẫu tin <Ti start> được ghi ra sổ ghi log trình.
  - Trước khi Ti thực hiện thao tác write(X), một mẫu tin <Ti, X, V1, V2> được ghi ra sổ ghi log trình.
  - Cuối cùng, khi giao dịch Ti bàn giao một phần, mẫu tin <Ti commit> được ghi ra sổ ghi log trình.

## Sự hiệu chỉnh CSDL tức thời

- Ví dụ ở trên - Sổ ghi log trình

<T0 start>
<T0, A, 1000, 950>
<T0, B, 2000, 2050>
<T0 commit>
<T1 start>
<T1, C, 700, 600>
<T1 commit>

- Khôi phục lỗi:
  - Undo(Ti): đặt lại giá trị cũ của tất cả các hạng mục dữ liệu của giao dịch Ti.
  - Redo(Ti): đặt lại giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch Ti,

## Sự hiệu chỉnh CSDL tức thời

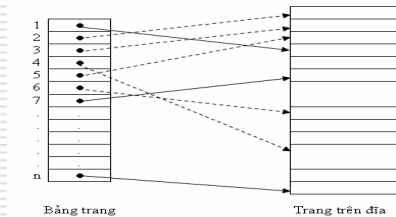
- Điều kiện để giao dịch Ti làm lại (redo) và không làm lại (undo):
  - Không làm lại: sổ ghi log trình chỉ chứa mẫu tin <Ti start> nhưng không chứa <Ti commit>
  - Làm lại: sổ ghi log trình chứa cả mẫu tin log trình <Ti start> và <Ti commit>

## Điểm kiểm soát - Checkpoint

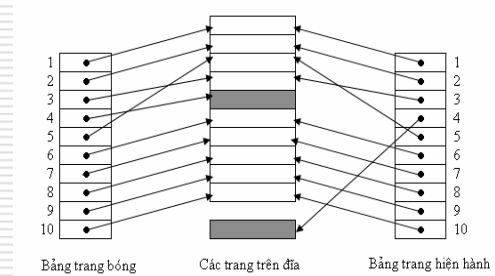
- Cải tiến hiệu năng của quá trình khôi phục lỗi
- Đặt điểm kiểm soát định kỳ
- Phục hồi sau lỗi giao dịch Ti:
  - Giao dịch gần đây nhất được khởi động trước điểm kiểm soát gần đây nhất.
  - Thủ tục **undo** và **redo** chỉ được áp dụng cho giao dịch Ti và các giao dịch diễn ra sau Ti. Tập các giao dịch này gọi là T.
  - Kỹ thuật hiệu chỉnh CSDL tức thời:
    - Với Tk trong T, không có mẫu tin log trình <Tk commit> thì thực hiện undo(Tk)
    - Với Tk trong T, có mẫu tin log trình <Tk commit> thì thực hiện redo(Tk)
  - Kỹ thuật hiệu chỉnh CSDL tri hoãn: không cần thực hiện undo(Tk)

## Phân trang bóng - Shadow Paging

- 2 bảng trang trong suốt chu trình sống của giao dịch:
  - Trang hiện hành
  - Trang bóng



## Phân trang bóng - Shadow Paging



## Phân trang bóng - Shadow Paging

- Ưu điểm:
  - Không mất thời gian ghi ra các mẫu tin lộ trình.
  - Khôi phục sau sự cố nhanh hơn do không cần các thao tác **undo** hoặc **redo**.
- Nhược điểm:
  - Tổng phí bản giao lớn
  - Phân mảnh dữ liệu
  - Phải thu nhặt rác
  - Khó thực hiện hơn so với kỹ thuật sổ ghi lộ trình

## Phục hồi các giao dịch cạnh tranh

- Ý tưởng: dùng một vùng đệm đĩa và sổ ghi lộ trình.
- Phương pháp cuộn lại giao dịch  $T_i$ :
  - Dò ngược sổ ghi lộ trình tìm các mẫu tin  $\langle T_i, X_j, V1, V2 \rangle$
  - Hạng mục dữ liệu  $X_j$  được trả lại giá trị cũ  $V1$
  - Việc dò tìm kết thúc khi thấy mẫu tin  $\langle T_i \text{ start} \rangle$
- Sử dụng điểm kiểm soát (checkpoint):
  - Mỗi mẫu tin lộ trình ghi điểm kiểm soát có dạng:
    - <checkpoint L>
 L: danh sách các giao dịch đang hoạt động tại thời điểm đặt điểm kiểm soát.
  - Khi hành động đặt điểm kiểm soát đang diễn ra, các giao dịch không được phép thực hiện bất kỳ thao tác cập nhật dữ liệu nào trên các khối đệm lẫn trên sổ ghi lộ trình.
    - Điểm kiểm soát mờ (fuzzy checkpoint).

## Phục hồi các giao dịch cạnh tranh

- 2 danh sách undo-list và redo-list.
- Tạo 2 danh sách:
  - Đầu tiên, chúng là rỗng.
  - Dò ngược sổ ghi lộ trình, tìm mẫu tin lộ trình <checkpoint> đầu tiên:
    - Với mỗi mẫu tin được tìm thấy có dạng <Ti commit>, ta thêm Ti vào trong *redo-list*.
    - Với mỗi mẫu tin được tìm thấy theo có dạng <Ti start> và Ti không thuộc *redo-list* thì thêm Ti vào trong *undo-list*.
  - Khi tất cả các mẫu tin lộ trình đã được xem xét, tiếp tục kiểm tra danh sách L trong mẫu tin lộ trình <checkpoint L> và thực hiện : với mỗi giao dịch Ti trong L mà Ti không thuộc *redo-list* thì thêm Ti vào *undo-list*.

## Phục hồi các giao dịch cạnh tranh

- Tiến trình phục hồi
  - Dò ngược sổ ghi lộ trình và thực hiện thủ tục undo(Ti) đối với mỗi mẫu tin lộ trình có chứa giao dịch Ti trong *undo-list*.
    - Dừng lại khi tìm thấy mẫu tin <Ti start> với mọi Ti thuộc danh sách *undo-list*.
  - Tìm mẫu tin <checkpoint L> gần đây nhất trong sổ ghi lộ trình.
  - Dò sổ ghi lộ trình theo chiều xuôi bắt đầu từ mẫu tin lộ trình <checkpoint L> gần đây nhất và thực hiện thủ tục redo(Ti) đối với mỗi mẫu tin lộ trình có chứa giao dịch Ti nằm trong danh sách *redo-list*.

## Phục hồi các giao dịch cạnh tranh

- Điểm kiểm soát mờ
  - Cho phép các giao dịch được cập nhật dữ liệu trên các khối đệm khi mẫu tin lộ trình <checkpoint> đã được viết xong nhưng trước thời điểm các khối đệm đã sửa đổi được ghi ra đĩa .
  - Ý tưởng:
    - Lưu vị trí của mẫu tin lộ trình checkpoint trong đĩa (*last\_checkpoint*).
    - *last\_checkpoint* chỉ cập nhật sau khi tất cả khối đệm bị sửa đổi.
    - *last\_checkpoint* chỉ được dùng cho hủy bỏ (undo)