Quản trị dữ liệu

Chương 5: Hệ thống phục hồi (Recovery System)

Nội dung

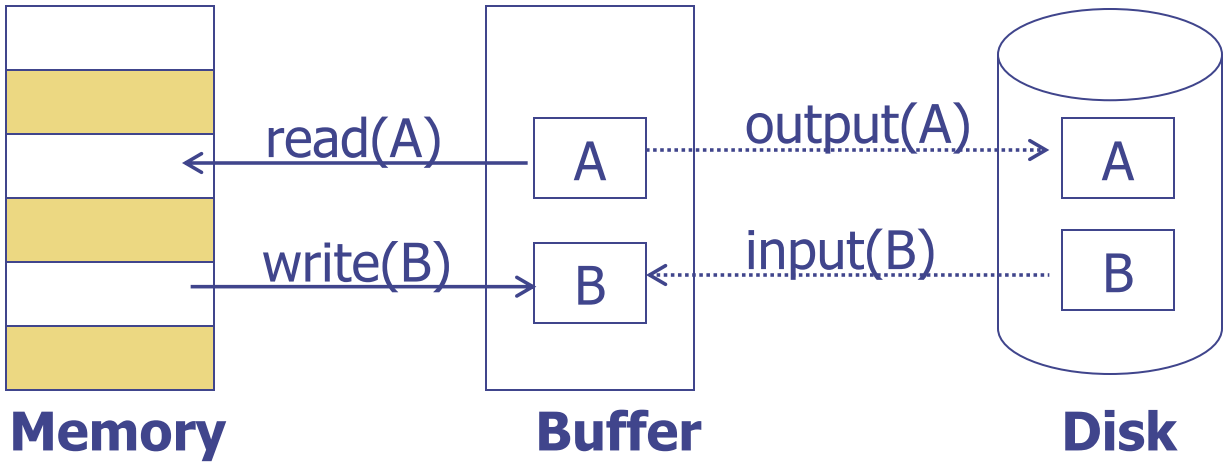
- Phân lớp hư hỏng
- Cấu trúc lưu trữ
- Phục hồi dựa trên sổ ghi lộ trình
  - Sự cập nhật có trì hoãn
  - Sự cập nhật tức thời
  - Điểm kiểm soát
- Phục hồi dành cho các giao dịch song song

Phân lớp hư hỏng

- Các loại hư hỏng:
  - Giao dịch:
    - Lỗi luận lý: dữ liệu đầu vào, tràn giá trị,…
    - Lỗi hệ thống: deadlock, cạnh tranh,..
  - Hệ thống: hệ điều hành, RAM,…
  - Đĩa
- Phục hồi: Xác định loại hư hỏng ⇒ đánh giá sự ảnh hưởng đến dữ liệu ⇒ đề xuất giải pháp đảm bảo tính bền vững và nguyên tử ⇒ **giải thuật phục hồi lỗi**
- Giải thuật phục hồi: bao gồm
  - Các hoạt động trong quá trình các giao dịch thực hiện
  - Các hoạt động sau khi lỗi phát sinh

Cấu trúc lưu trữ

- Các loại lưu trữ:
  - Lưu trữ bị phai (bay hơi)
  - Lưu trữ không phai (không bay hơi)
  - Lưu trữ bền: thông tin *không bao giờ* bị mất (!??)
- Thực hiện lưu trữ bền
- Phương pháp truy cập dữ liệu:



Phục hồi dựa trên Sổ ghi lộ trình (log)

- Dùng sổ ghi lộ trình ghi nhận lại các thay đổi trên CSDL
- Một thao tác cập nhật ⇒ một log record
- Các loại log record:

Loại Log Record	Ý nghĩa
$<T_i$, Start$>$	GD T_i đã khởi động.
$<T_i, X, V_1, V_2>$	GD T_i thay đổi giá trị của X từ V_1 thành V_2
$<T_i$, commit$>$	GD T_i đã bàn giao.
$<T_1$, abort$>$	GD T_i đã hủy bỏ.

- Sổ ghi lộ trình ghi vào các thiết bị "bền"
- Có 2 giải thuật phục hồi dựa trên Sổ ghi lộ trình

Sự cập nhật có trì hoãn

- Dùng khi các GD được thực hiện tuần tự
- Tất cả các thao tác cập nhật CSDL sẽ bị trì hoãn
- Sự thực thi của các giao dịch tiến hành như sau:
  - Trước khi T_i khởi động: **$<T_i$, Start$>$**
  - Trước khi T_i Write(X): **$<T_i$, X, $V_2>$**
  - Khi T_i bàn giao một phần: **$<T_i$, commit$>$**

- Phục hồi:

  $\forall T_i$: $<T_i$, start$>$ và $<T_i$, commit$>$ có trong sổ ghi lộ trình
  $\Rightarrow$ **redo(T_i)**

redo(T_i): cập nhật giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi T_i

Sự cập nhật có trì hoãn

- Ví dụ:

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

<T_1, start>
<T_1, A, 950>
<T_1, B, 2050>
<T_1, commit>
<T_2, start>
<T_2, C, 600>
<T_2, commit>

Sự cập nhật tức thời

- Các thao tác thay đổi gtrị các hạng mục CSDL sẽ thể hiện ngay lên CSDL
- Sự thực thi các GD được tiến hành như sau:
  - Trước khi T_i khởi động, **<T_i, Start>**
  - Trước khi T_i Write(X), **<T_i, X, V_1, V_2>**
  - Khi T_i hoàn thành, **<T_i, commit>**
- Phục hồi:
  - $\forall$ T_i: <T_i, start> và <T_i, commit> có trong sổ ghi lộ trình $\Rightarrow$ **redo(T_i)**
  - $\forall$ T_i: <T_i, start> có trong sổ ghi lộ trình nhưng không có <T_i, commit> $\Rightarrow$ **undo(T_i)**
  - Thực hiện sự phục hồi từ dưới lên

Sự cập nhật tức thời

- Ví dụ:

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

<T_1, start>
<T_1, A, 1000, 950>
<T_1, B, 2000, 2050>
<T_1, commit>
<T_2, start>
<T_2, C, 700, 600>
<T_2, commit>

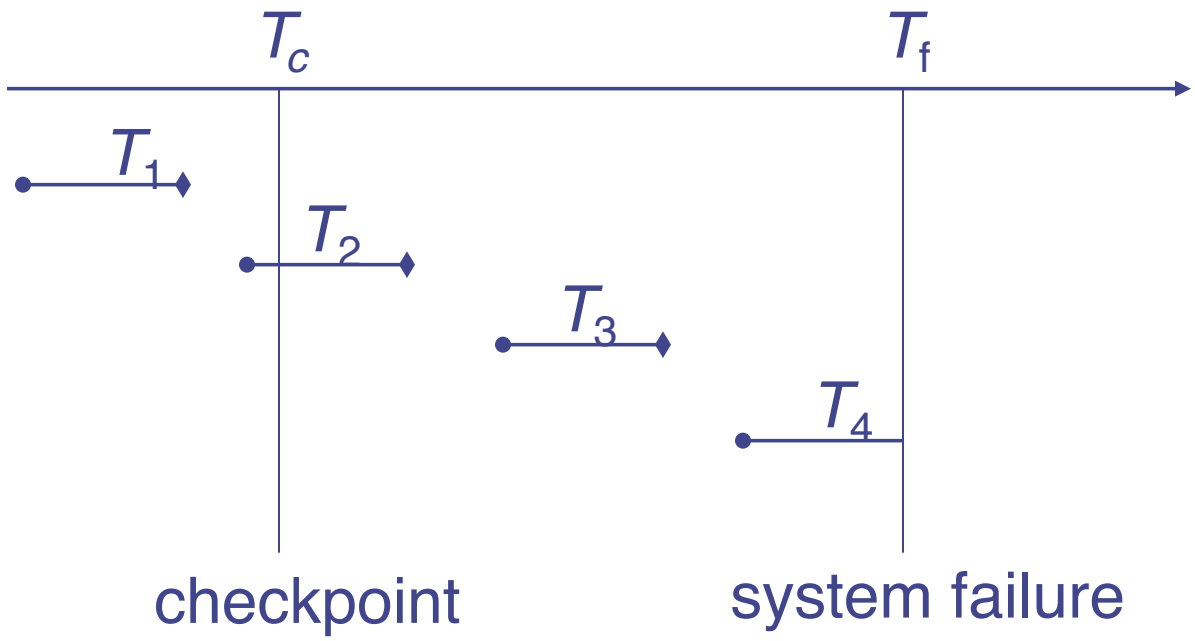
- **Redo(T_i):** đặt giá trị mới cho các hạng mục CSDL
- **Undo(T_i):** đặt giá trị cũ cho các hạng mục CSDL

Điểm kiểm soát (Checkpoint)

- Dùng để cải thiện hiệu năng của quá trình khôi phục
- Muốn đặt điểm kiểm soát:
  1. Ghi log record vào thiết bị lưu trữ
  2. Cập nhật các khối đệm đã cập nhật lên CSDL
  3. Thêm **<checkpoint>** vào sổ ghi lộ trình
- Phục hồi:
  1. Từ điểm checkpoint cuối cùng, dò ngược lên tìm **<T_i, start>** gần nhất.
  2. Gọi **T** là tập các GD gồm $\mathbf{T_i}$ và các GD diễn ra sau $\mathbf{T_i}$
     - $\forall T_K \in T$: <T_K, commit> không có trong SGLT
       - Nếu Cập nhật tức thời: ⇒ **undo(T_K)**
       - Nếu Cập nhật trì hoãn: ⇒ bỏ qua T_K (không undo)
     - $\forall T_K \in T$: <T_K, commit> có trong SGLT ⇒ **redo(T_K)**

Điểm kiểm soát

- Ví dụ:



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 tập các GD đang hoạt động tại thời điểm đặt checkpoint
- Phục hồi:
  1. **Tạo hai danh sách redo-list và undo-list:**
     a. Redo-list = ∅; undo-list = ∅
     b. Dò ngược SGLT đến khi gặp mẫu tin **<checkpoint L>** đầu tiên:
        - Nếu thấy $<T_i$, commit$>$:
          ⇒ thêm T_i vào redo-list
        - Nếu thấy $<T_i$, start$>$ và $T_i \notin$ redo-list:
          ⇒ thêm T_i vào undo-list
     c. $\forall T_i \in L$ nhưng $T_i \notin$ redo-list và undo-list ⇒ thêm T_i vào undo-list

Phục hồi cho các giao dịch song song

2. **Tiến trình phục hồi:**
   a. Dò ngược sổ ghi đến khi tìm thấy **<T_i, start>** cho tất cả các T_i trong undo-list, tiến hành undo đối với mỗi record-log thuộc undo-list
   b. Định vị **<checkpoint L>** cuối cùng
   c. Dò xuôi sổ ghi lịch trình cho đến cuối sổ ghi, thực hiện redo với mỗi record-log thuộc giao dịch T_i nằm trong redo-list

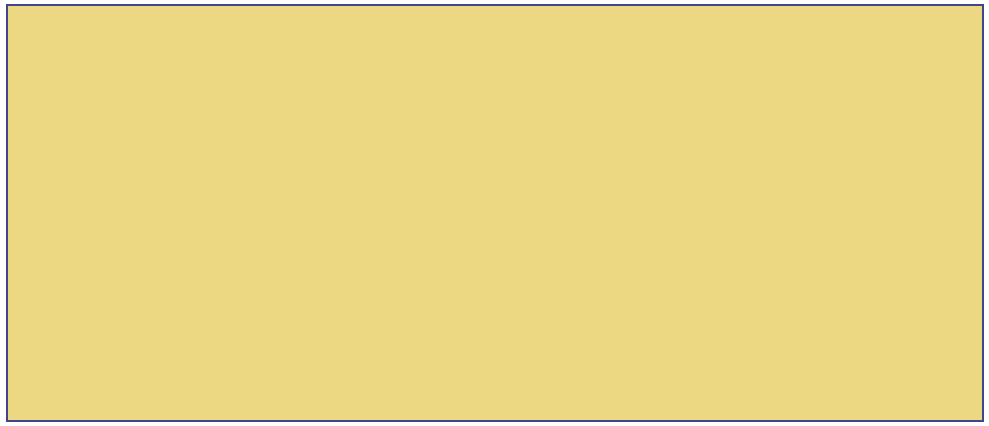
Phục hồi cho các giao dịch song song

Cho một sổ ghi lịch trình sau:

a. Hãy xác định L1, L2.
b. Xác định giá trị của A – E trên đĩa sau khi sự cố và trước khi phục hồi
c. Xác định giá trị của A – E trên đĩa sau khi phục hồi.

<T1, start>
<T2, start>
<T1, A, 10, 30>
<T1, B, 15, 35>
<T2, C, 110, 160>
<checkpoint **L1**>
<T1, commit>
<T2, D, 210, 260>
<T3, start>
<T3, E, 1010, 1510>
<checkpoint **L2**>
<T2, commit>
~~~ crash ~~~

Giải:


~~~

Example

Given the following log file:

a. Identify L1, L2.
b. Identify value of A – E on the DB after the crash and before the recovery
c. Identify value of A – E on the DB after the recovery.

Sol.:

<T1, start>
<T2, start>
<T1, A, 10, 30>
<T1, B, 15, 35>
<T2, C, 110, 160>
<checkpoint **L1**>
<T1, commit>
<T2, D, 210, 260>
<T3, start>
<T3, E, 1010, 1510>
<checkpoint **L2**>
<T2, commit>
~~~ crash ~~~
~~~