IS210 – Chương 5

Khôi phục sau sự cố

Trương Thu Thủy

Nội dung

- Giới thiệu
- Nhật ký giao tác (transaction log)
- Điểm kiểm tra (checkpoint)
- Phương pháp khôi phục

1 Giới thiệu

Nhắc lại

- Tính toàn vẹn dữ liệu
  - Tính đúng đắn của dữ liệu
  - Tính nhất quán của ràng buộc
- Trạng thái nhất quán
  - Thỏa các ràng buộc toàn vẹn
- Trạng thái nhất quán của cơ sở dữ liệu
  - Cơ sở dữ liệu ở trạng thái nhất quán

Giới thiệu

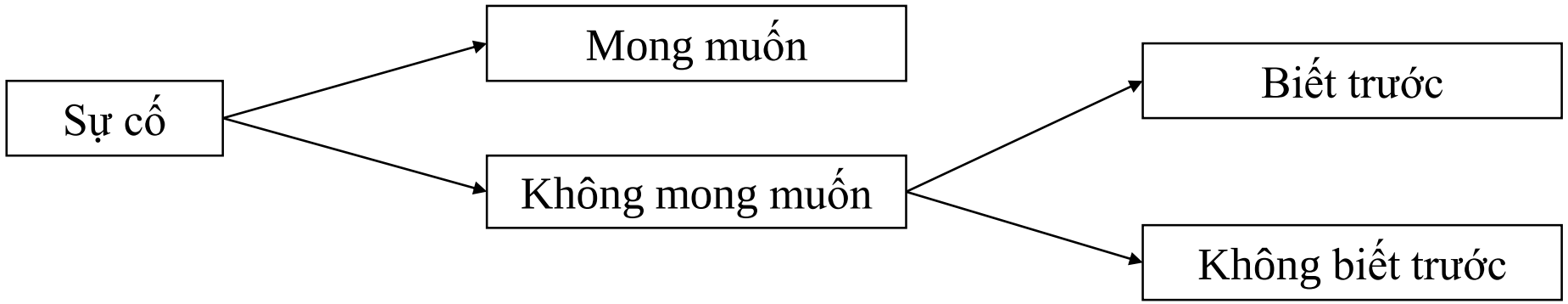
- Cơ sử dữ liệu phải luôn ở trạng thái nhất quán, đảm bảo tất cả các ràng buộc toàn vẹn (RBTV)
- Các nguyên nhân dẫn đến vi phạm RBTV
  - Lỗi trong các giao tác (transaction bug)
  - Lỗi lập trình của DBMS (DBMS bug)
  - Hư hỏng phần cứng (hardware failure)

  →Sử dụng các kỹ thuật khôi phục sự cố
  - Chia sẻ dữ liệu (data sharing)

  →Sử dụng các kỹ thuật quản lý giao tác và xử lý đồng thời

Các loại sự cố

- Phân loại theo nguyên nhân
  - Sự cố nhập liệu sai - Erroneous Data Entry
  - Sự cố trên thiết bị lưu trữ – Media failure
  - Sự cố giao tác – Transaction failure
  - Sự cố hệ thống – System failure
- Phân loại theo tính chất
-



Sự cố do nhập liệu sai

- Dữ liệu sai hiển nhiên
  - Thiếu một số trong dãy số điện thoại (không đúng định dạng số điện thoại)
- Dữ liệu sai không thể phát hiện
  - Nhập sai một số trong dãy số điện thoại
- DBMS cung cấp các cơ chế cho phép phát hiện lỗi
  - Ràng buộc khóa chính, khóa ngoại
  - Ràng buộc miền trị
  - Trigger

Sự cố trên thiết bị lưu trữ (Media failure)

- Là những sự cố gây nên việc mất hay không thể truy xuất dữ liệu ở bộ nhớ ngoài (ổ cứng, CD, băng từ …)
- Giải quyết
  - Backup thường xuyên
  - Chạy nhiều bản CSDL song hành

Sự cố giao tác (Transaction failure)

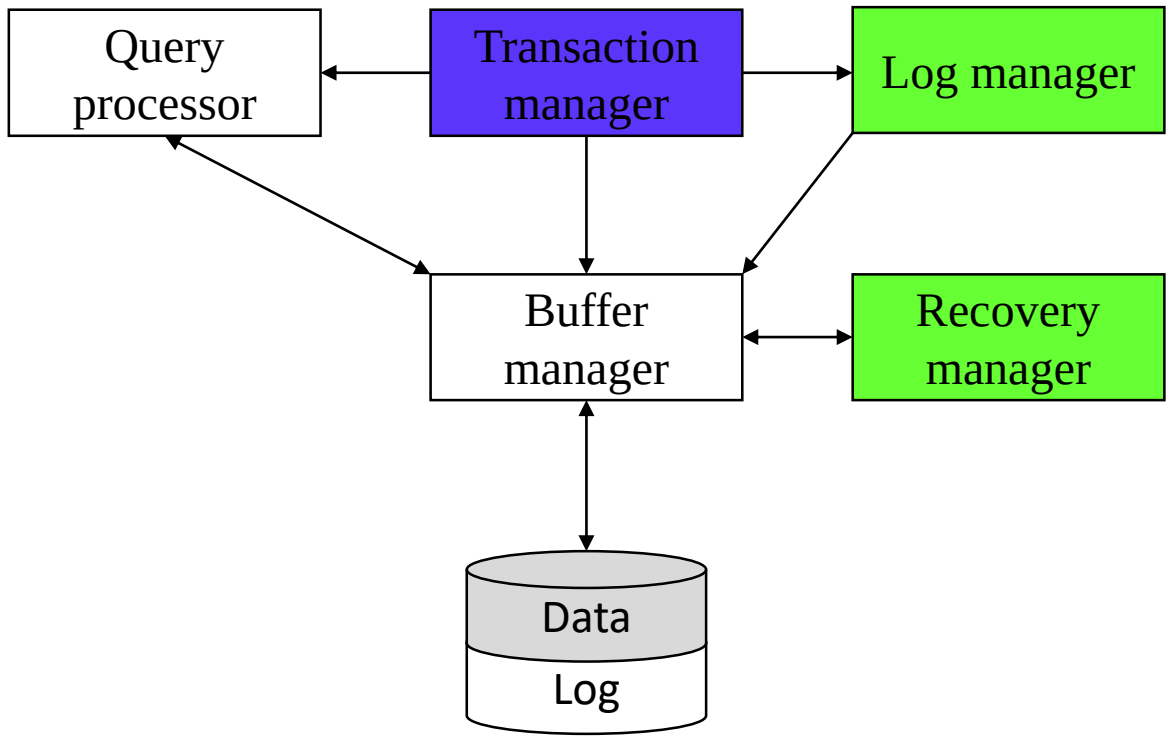
- Là sự cố làm cho một giao tác kết thúc không bình thường
  - Chia cho 0
  - Giao tác bị hủy bỏ
  - Dữ liệu nhập sai
  - Tràn số
- Giải quyết
  - Thực hiện lại giao tác

Sự cố hệ thống (System failure)

- Lỗi phần cứng
  - Cúp điện
  - Hư hỏng bộ nhớ trong, CPU
- Lỗi phần mềm
  - Lỗi hệ điều hành
  - Lỗi DBMS
- Khắc phục
  - DBMS sử dụng nhật ký giao tác (transaction log)

Mục tiêu của khôi phục dữ liệu

- Đưa dữ liệu về trạng thái sau cùng trước khi xảy ra sự cố
- Đảm bảo hai tính chất
  - Nguyên tố (atomic)
  - Bền vững (durability)

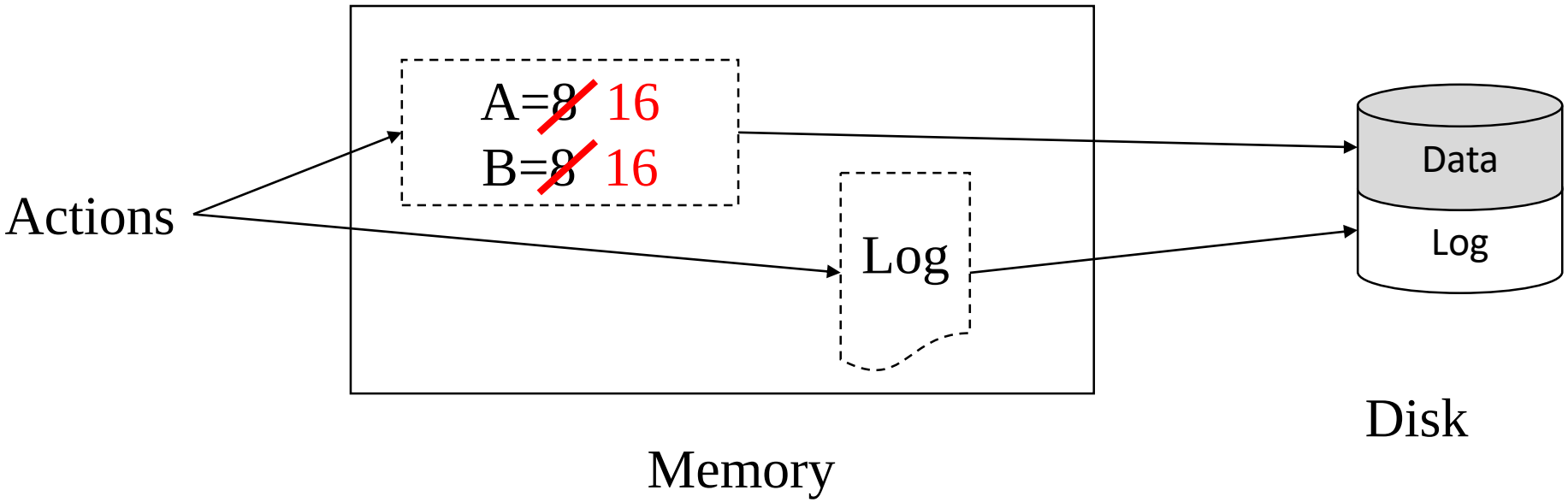


2

Nhật ký giao tác

Nhật ký giao tác – transaction log

- Nhật ký giao tác là một chuỗi các mẫu tin (log record) ghi lại các hành động của DBMS
- Nhật ký là một tập tin tuần tự được lưu trữ trên bộ nhớ chính, và sẽ ghi xuống **ổ đĩa** ngay khi có thể



Nhật ký giao tác

- Mẫu tin nhật ký (log record) gồm
  - <start T>
    - Giao tác T đã bắt đầu
  - <commit T>
    - Giao tác T đã kết thúc thành công
  - <abort T>
    - Giao tác T bị hủy
  - <T, X, v> hoặc <T, X, v, w>
    - Giao tác T đã thay đổi đơn vị dữ liệu X, giá trị cũ là v và giá trị mới là w

Nhật ký giao tác

- Khi sự cố xảy ra
  - DBMS sẽ tra cứu nhật ký giao tác để khôi phục lại trạng thái nhất quán của dữ liệu
- Để sửa chữa các sự cố
  - Một vài giao tác phải thực hiện lại (redo)
    - Giá trị mới đã ghi xuống CSDL sẽ được ghi lại
  - Một vài giao tác không cần thực hiện (undo)
    - Cơ sở dữ liệu phục hồi lại và những giao tác này như thể chưa bao giờ được thực thi

3 Điểm kiểm tra - checkpoint

Thảo luận

- Về nguyên tắc, việc khôi phục dữ liệu yêu cầu kiểm tra toàn bộ nhật ký
  - Quét nhật ký từ cuối lên đầu để xác định những giao tác chưa hoàn tất
  - Khi một giao tác đã có <Commit T> trong nhật ký thì không cần phải khôi phục giao tác này

  →Tốn thời gian
  - Không thể xóa những dòng nhật ký trước một <Commit T>
    - Nhiều giao tác cùng thực thi một lúc
- Dùng điểm kiểm tra (checkpoint): để xác định vị trí cần quét trong nhật ký

Điểm kiểm tra – checkpoint

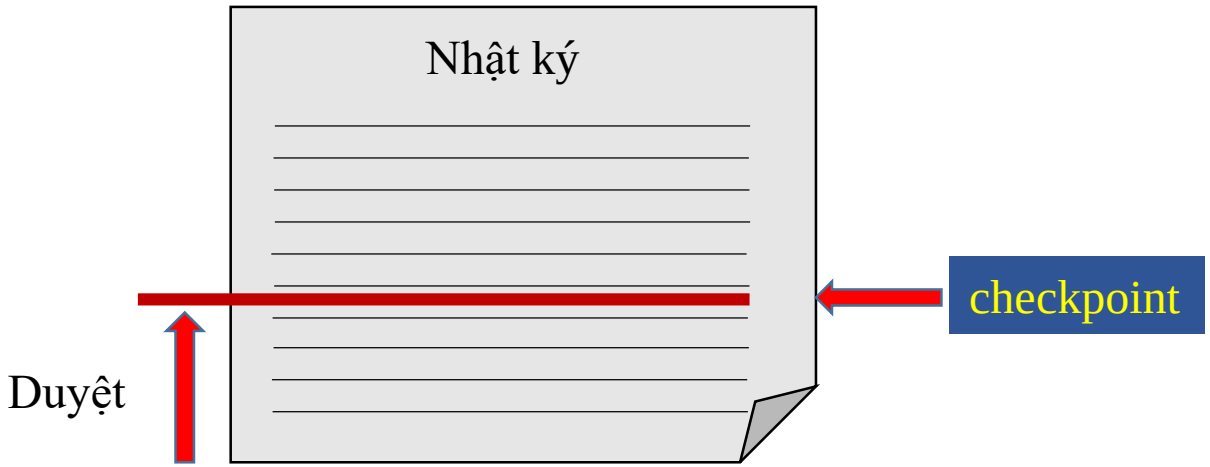
- Nhật ký giao tác có thêm mẫu tin <checkpoint> hay <ckpt>
- Mẫu tin <checkpoint>
  - Sẽ được ghi định kỳ vào thời điểm mà DBMS ghi tất cả những thay đổi của CSDL từ vùng đệm xuống đĩa
- Phân loại
  - Điểm kiểm tra đơn giản – simple checkpoint
  - Điểm kiểm tra linh động – nonquiescent checkpoint

Điểm kiểm tra đơn giản – simple checkpoint

- Tại thời điểm cần ghi checkpoint
  1. Tạm dừng tiếp nhận các giao tác mới
  2.
     - Đợi cho đến khi tất cả các ‘giao tác đang hoạt động’ commit hoặc abort
     - Ghi mẫu tin <Commit T> hoặc <Abort T> vào nhật ký
  3. Đưa dữ liệu nhật ký xuống đĩa (flush-log)
  4. Tạo mẫu tin <Checkpoint> và ghi xuống đĩa
  5. Tiếp tục nhận các giao tác mới

Điểm kiểm tra đơn giản – simple checkpoint

- Khi quét nhật ký để khôi phục dữ liệu:
  - Trước điểm kiểm tra (checkpoint) – là những giao tác đã kết thúc → không cần phải làm lại
  - Sau điểm kiểm tra (checkpoint) – là những giao tác chưa thực hiện xong→ cần phải khôi phục
- Không cần phải duyệt hết nhật ký
  - Chỉ cần duyệt từ cuối nhật ký đến điểm kiểm tra (checkpoint)

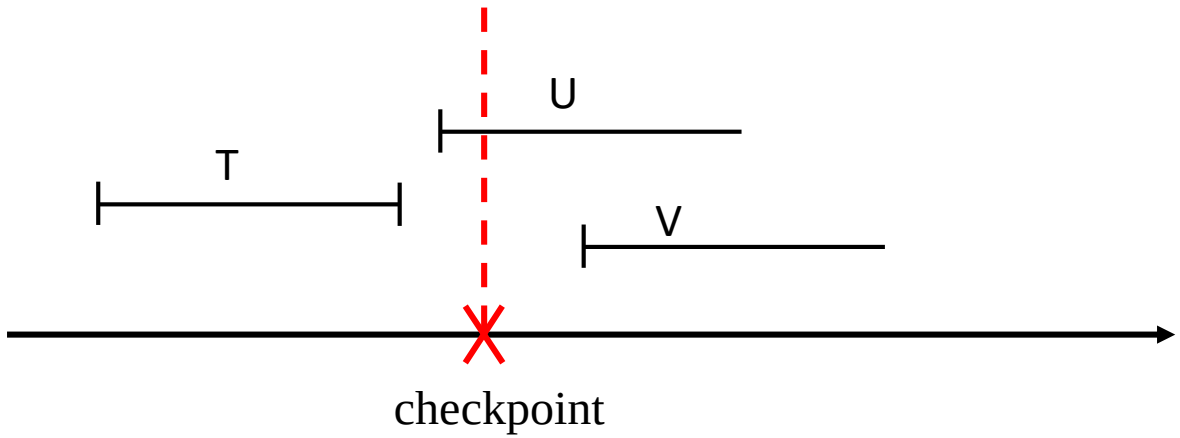


Điểm kiểm tra đơn giản – simple checkpoint

- Trong suốt quá trình ghi checkpoint
  - Hệ thống gần như tạm dừng hoạt động
  - Bởi vì những giao tác đang hoạt động có thể cần nhiều thời gian để commit hoặc abort
- Cần kỹ thuật phức tạp hơn
  - Cho phép tiếp nhận các giao tác mới trong quá trình checkpoint

Điểm kiểm tra linh động – nonquiescent checkpoint

- Đặc điểm:
  - Mẫu tin <checkpoint> trong 'Điểm điểm tra đơn giản' nay được chia thành hai mẫu tin:
    - Mẫu tin **<start ckpt (T_1, T_2, …, T_k)>**: Bắt đầu checkpoint
      - T_1, T_2, …, T_k là các giao tác đang thực thi (active transaction)
    - Mẫu tin **<end ckpt>**: Kết thúc checkpoint
  - Cho phép tiếp nhận các giao tác mới trong quá trình checkpoint



Điểm kiểm tra linh động – nonquiescent checkpoint

- Khi đến thời điểm ghi checkpoint, DBMS sẽ
  1.
     - Tạo mẫu tin **<start ckpt (T_1, T_2, …, T_k)>**
     - Ghi dữ liệu này vào ổ đĩa (flush-log)
  2.
     - Đợi đến khi tất cả các giao tác T_1, T_2, …, T_k commit hoặc abort
     - Không cấm các ‘giao tác khác’ bắt đầu
  3.
     - Khi tất cả T_1, T_2, …, T_k đều đã hoàn thành, tạo mẫu tin nhật ký <end ckpt>
     - Ghi dữ liệu này vào ổ đĩa (flush-log)

4

Phương pháp khôi phục – Recovery method

Undo-logging

Redo-logging

Undo/Redo logging

Nhắc lại

- Các hành động
  - INPUT (X): copy X vào bộ nhớ đệm
  - READ(X,t): copy giá trị của X vào t
  - WRITE(X,t): copy giá trị của t vào X
  - OUTPUT (X): copy X từ bộ nhớ đệm vào ổ đĩa
  - FLUSH LOG: nhật ký (log) được đẩy từ bộ đệm xuống ổ đĩa

Undo-logging

- Quy tắc
  1. Mẫu tin nhật ký
     - <T, X, v>: giao tác T đã thay đổi đơn vị dữ liệu X, và giá trị trước đó là v (không quan tâm đến giá trị hiện tại)
  2. Mẫu tin <T, X, v> phải được ghi xuống ổ đĩa trước khi giá trị mới của X được ghi vào ổ đĩa
  3. Mẫu tin <commit T> chỉ được ghi xuống log trên ổ đĩa sau khi tất cả các đơn vị dữ liệu bị thay đổi bởi T đã được ghi xuống ổ đĩa

Ví dụ

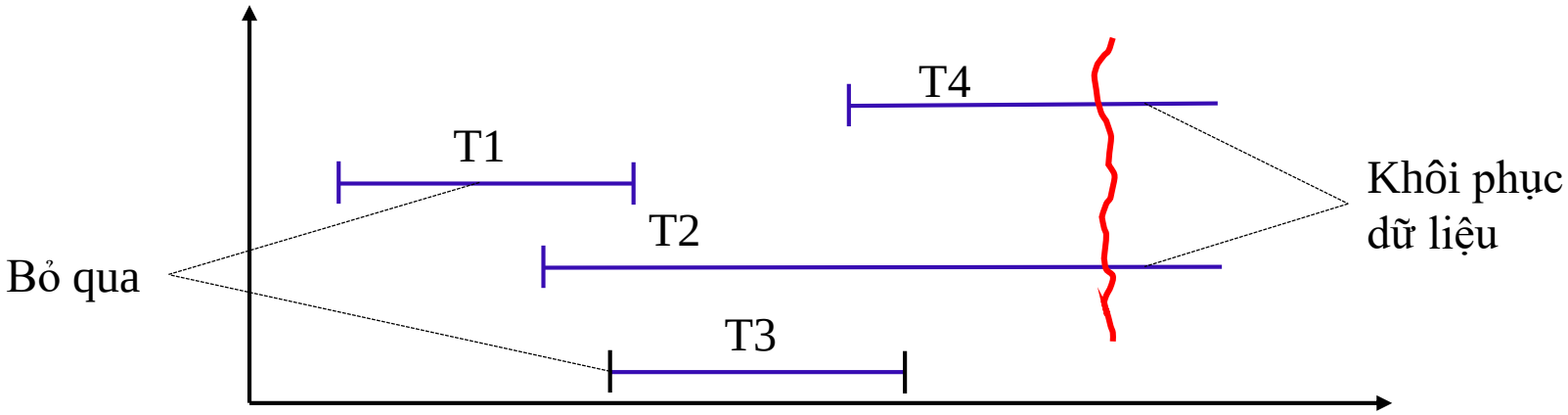
Step	Operation	t	Mem A	Mem B	Disk A	Disk B	Log
1							<START T>
2	Read(A,t)	8	8		8	8	
3	t=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8>
5	Read(B,t)	8	16	8	8	8	
6	t=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8>
8	FLUSH LOG						
9	Output(A)	16	16	16	16	8	
10	Output(B)	16	16	16	16	16	
11							<COMMIT T>
12	FLUSH LOG						

Undo-logging

- Quá trình khôi phục
  - Gọi S là tập các giao tác chưa kết thúc
    - Tìm thấy <start T> nhưng không có <commit T> hay <abort T> trong nhật ký
  - Với mỗi mẫu tin <T, X, v> trong nhật ký (theo thứ tự từ cuối → đầu tập tin) → thực hiện ghi lại giá trị cũ cho đơn vị dữ liệu D
    - Nếu T ∈ S thì: ghi giá trị v cho đơn vị dữ liệu X
  - Với mỗi giao tác T ∈ S
    - Ghi mẫu tin <abort T> vào nhật ký và đẩy mẫu tin này vào 'nhật ký trên ổ đĩa' (flush log)

Undo-logging

- Khi có sự cố
  - T1 và T3 đã hoàn tất
  - T2 và T4 chưa kết thúc



Ví dụ

Step	Operation	t	Mem A	Mem B	Disk A	Disk B	Log
1							<START T>
2	Read(A,t)	8	8		8	8	
3	t=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8>
5	Read(B,t)	8	16	8	8	8	
6	t=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8>
8	FLUSH LOG						
9	Output(A)	16	16	16	16	8	
10	Output(B)	16	16	16	16	16	
11							<COMMIT T>
12	FLUSH LOG						

Sự cố → Những thay đổi của A/B chưa ghi xuống ổ đĩa

Sự cố → Khôi phục A=8 và B=8

Sự cố → Không cần khôi phục T

Undo-logging

- Nhận xét
  - Về nguyên tắc, việc khôi phục dữ liệu phải quét qua toàn bộ nhật ký
  - Một khi giao tác đã hoàn tất và COMMIT log đã được ghi vào ổ đĩa thì dữ liệu log của giao tác này không còn cần thiết trong quá trình khôi phục nữa.
  - Mong muốn:
    - Xóa những dòng nhật ký trước COMMIT
    - Tuy nhiên, đôi khi không thể.
    - Lý do là nhiều giao tác cùng thực hiện một lúc. Nếu cắt bớt nhật ký sau khi một giao tác hoàn tất thì các bản ghi nhật ký liên quan đến các giao tác khác đang thực hiện có thể bị mất.
  - Cách đơn giản: dùng điểm kiểm tra (checkpoint)

Undo-logging và Checkpoint

- Cách ghi nhật ký

1. Ngừng tiếp nhận giao tác mới
2. Đợi cho đến khi tất cả giao tác hiện tại đều đã commit hoặc abort và đã ghi COMMIT hoặc ABORT lên nhật ký
3. Đẩy dữ liệu từ nhật ký vào ổ đĩa (flush-log)
4. Ghi mẫu tin <CKPT>, và đẩy dữ liệu từ nhật ký vào ổ đĩa (flush-log)
5. Tiếp nhận các giao tác mới

Undo-logging và Checkpoint

- Giả sử nội dung nhật ký như sau:

<start T_1>

< T_1, A, 5>

<start T_2>

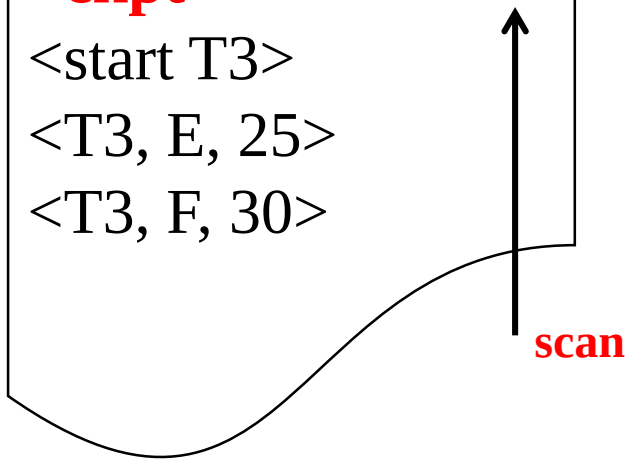
< T_2, B, 10>

Thời điểm DBMS cần thực hiện Checkpoint

- Cách ghi nhật ký
  - Vì T_1 và T_2 đang thực thi nên chờ
  - Sau khi T_1 và T_2 hoàn tất hoặc hủy bỏ
    - Ghi <checkpoint> vào nhật ký

Undo-logging và Checkpoint

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<T2, C, 15>
<T2, D, 20>
<commit T1>
<commit T2>
<ckpt>
<start T3>
<T3, E, 25>
<T3, F, 30>



- Ví dụ: khôi phục dữ liệu khi xảy ra sự cố
  - Giả sử sự cố xảy ra.
  - Phải quét nhật ký từ dưới lên, xác định được T3 là giao tác chưa hoàn tất và khôi phục lại giá trị F và E thành 30 và 25
  - Khi gặp <CKPT> thì ta biết được không cần phải xem xét những mẫu tin trước đó và việc khôi phục database được hoàn tất.

Undo-logging và Nonquiescent Checkpoint

- Giả sử nội dung nhật ký như sau:

<start T_1>

< T_1, A, 5>

<start T_2>

< T_2, B, 10>

Thời điểm DBMS cần thực hiện Nonquiescent Checkpoint

- Cách ghi nhật ký
  - Vì T_1 và T_2 đang thực thi nên tạo <start ckpt (T_1, T_2)>
  - Trong khi chờ T_1 và T_2 kết thúc, DBMS vẫn tiếp nhận các giao tác mới
  - Sau khi T_1 và T_2 kết thúc, ghi <end ckpt> lên nhật ký

Undo-logging và Nonquiescent Checkpoint

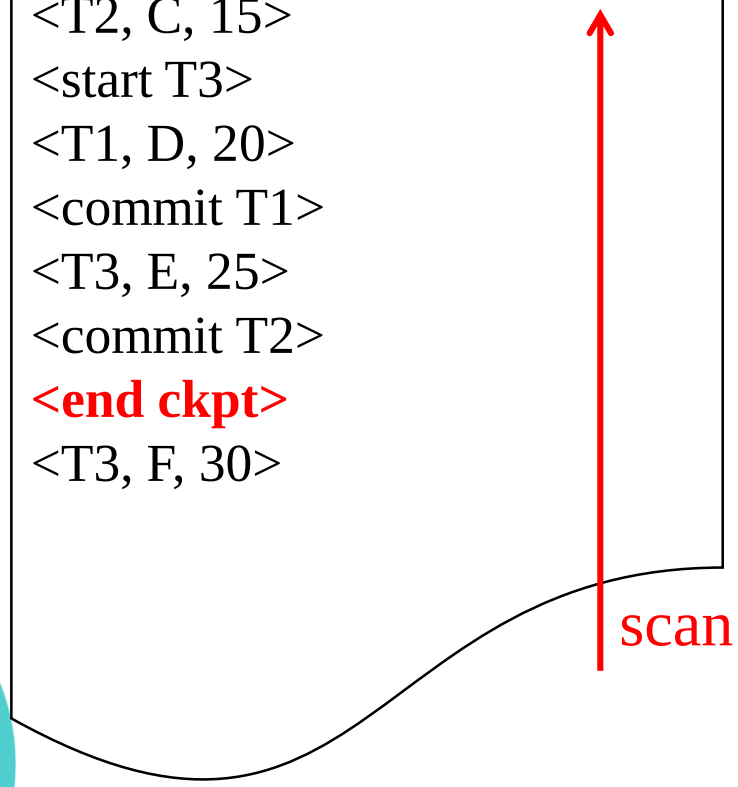
- Ví dụ:

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<start ckpt (T1, T2)>
<T2, C, 15>
<start T3>
<T1, D, 20>
<commit T1>
<T3, E, 25>

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<start ckpt (T1, T2)>
<T2, C, 15>
<start T3>
<T1, D, 20>
<commit T1>
<T3, E, 25>
<commit T2>
<end ckpt>
<T3, F, 30>

Undo-logging và Nonquiescent Checkpoint

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<start ckpt (T1, T2)>
<T2, C, 15>
<start T3>
<T1, D, 20>
<commit T1>
<T3, E, 25>
<commit T2>
<end ckpt>
<T3, F, 30>

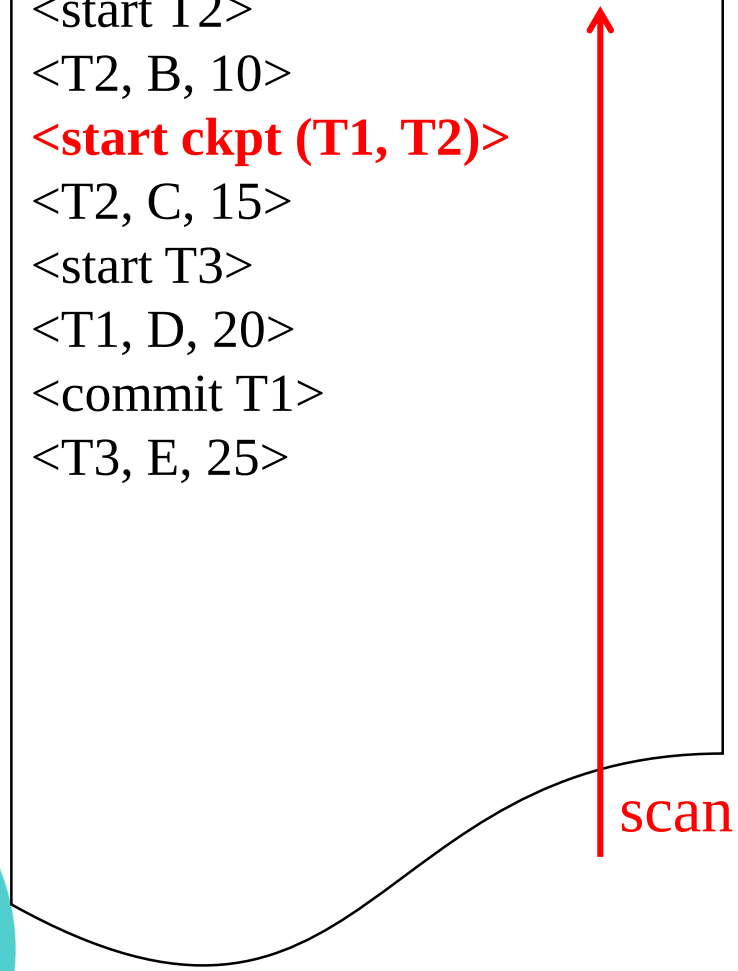


Trường hợp 1: khôi phục với đầy đủ <start ckpt> <end ckpt>

- <T3, F, 30>
  - T3 chưa kết thúc→ khôi phục F=30
- <end ckpt>
  - Những giao tác bắt đầu trước <start ckpt> đã hoàn tất
  - T1 và T2 đã hoàn tất
- <T3, E, 25>
  - Khôi phục E=25
- <start ckpt(T1, T2)>
  - Dừng
  - Ghi <abort T3>

Undo-logging và Nonquiescent Checkpoint

<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
<start ckpt (T1, T2)>
<T2, C, 15>
<start T3>
<T1, D, 20>
<commit T1>
<T3, E, 25>



Trường hợp 2: khôi phục khi thiếu <end ckpt>

- <T3, E, 25>
  - T3 chưa kết thúc→ khôi phục E=25
- <commit T1>
  - T1 bắt đầu trước <start ckpt> và đã hoàn tất
- <T2, C, 15>
  - T2 bắt đầu trước <start ckpt> và chưa kết thúc
  - Khôi phục C=15
- <start ckpt(T1, T2)>
  - Chỉ có T1 và T2 bắt đầu trước đó
- <T2, B, 10>
  - Khôi phục B=10
- <start T2>
  - Dừng, ghi <abort T3> và ghi <abort T2>

Undo-logging – Bài tập

- Bên cạnh là file undo-log ghi lại quá trình thực thi hai giao tác T và U
- Mô tả hành động khôi phục dữ liệu nếu ‘có sự cố xảy ra và mẫu tin nhật ký cuối cùng trên ổ đĩa’ là:
  1. <START U>
  2. <COMMIT U>
  3. <T, E, 50>
  4. <COMMIT T>

<START T>
<T, A, 10>
<START U>
<U, B, 20>
<T, C, 30>
<U, D, 40>
<COMMIT U>
<T, E, 50>
<COMMIT T>

Undo-logging – Bài tập

- Bên cạnh là file undo-log
- Giả sử một nonquiescent checkpoint bắt đầu ngay sau khi một trong những record được ghi (in memory)
  a. <S, A, 60>
  b. <T, A, 10>
  c. <U, B, 20>
  d. <U, D, 40>
  e. <T, E, 50>
- Đối với mỗi trường hợp
  - Khi nào <END CKPT> được ghi
  - Và nếu có sự cố, chúng ta phải đi bao xa để tìm những giao tác chưa hoàn tất

<START S>
<S, A, 60>
<COMMIT S>
<START T>
<T, A, 10>
<START U>
<U, B, 20>
<T, C, 30>
<START V>
<U, D, 40>
<V, F, 70>
<COMMIT U>
<T, E, 50>
<COMMIT T>
<V, B, 80>
<COMMIT V>

4

Phương pháp khôi phục – Recovery method

Undo-logging

Redo-logging

Undo/Redo logging

Redo – logging

- Quy tắc
  1. Một giao tác T muốn cập nhật đơn vị dữ liệu sẽ phát sinh ra một mẫu tin nhật ký
     - Ghi nhận lại giá trị mới trong nhật ký <T, X, w>
  2. Trước khi X được cập nhật xuống đĩa
     - Tất cả các mẫu tin **nhật ký** của giao tác cập nhật X đã phải có mặt trên đĩa
     - Bao gồm mẫu tinh cập nhật <T, X, w> và <commit T>
  3. Khi T hoàn tất, tiến hành ghi nhật ký xuống đĩa: flush-log
     - Các hành động <T, X, w>, <commit> và flush-log phải trước OUTPUT (X)

Ví dụ

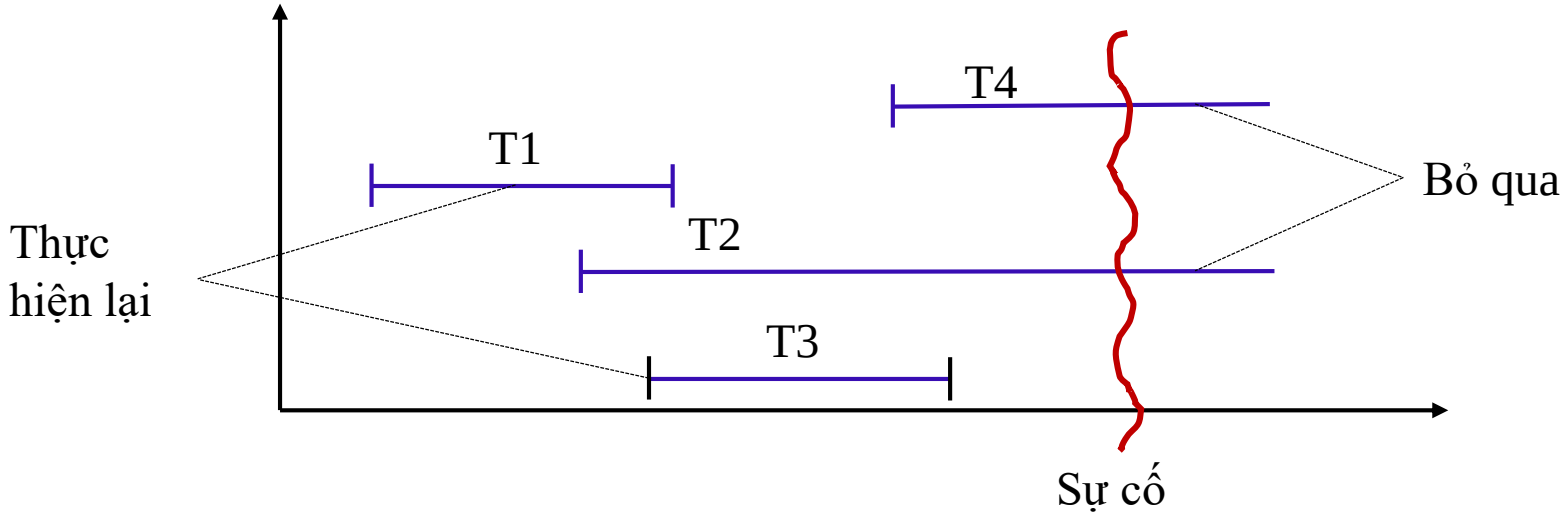
Step	Operation	t	Mem A	Mem B	Disk A	Disk B	Log
1							<START T>
2	Read(A,t)	8	8		8	8	
3	t=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 16>
5	Read(B,t)	8	16	8	8	8	
6	t=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 16>
8							<commit T>
9	**FLUSH LOG**						
10	Output(A)	16	16	16	16	8	
11	Output(B)	16	16	16	16	16	

Redo – logging

- Quá trình khôi phục
  - Xác định các giao tác đã hoàn tất
  - Đối với tập tin nhật ký <T, X, v>
    - Nếu T chưa commit: không làm gì (do nothing)
    - Nếu T đã commit, ghi giá trị v vào đơn vị dữ liệu X
  - Đối với những giao tác T chưa hoàn tất, ghi <abort T> vào log và flush-log

Redo – logging

- Khi có sự cố
  - T1 và T3 đã hoàn tất
  - T2 và T4 chưa kết thúc



Ví dụ

Step	Operation	t	Mem A	Mem B	Disk A	Disk B	Log
1							<START T>
2	Read(A,t)	8	8		8	8	
3	t=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 16>
5	Read(B,t)	8	16	8	8	8	
6	t=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 16>
8							<commit T>
9	**FLUSH LOG**						
10	Output(A)	16	16	16	16	8	
11	Output(B)	16	16	16	16	16	

Xem như T chưa hoàn tất, A và B không có thay đổi

Thực hiện ghi lại T, ghi A=16 và B=16

Thực hiện ghi lại T, ghi A=16 và B=16

Redo – logging và checkpoint

- Nhận xét
  - Phương pháp Redo thực hiện ghi dữ liệu trễ so với thời điểm hoàn tất của các giao tác

<start ckpt>
Thực hiện ghi những đơn vị dữ liệu (chưa được ghi xuống ổ đĩa) đang ở trên buffer xuống đĩa của những giao tác đã commit
<end ckpt>

  - Mẫu tin <end ckpt> được ghi vào nhật ký mà không phải đợi các giao tác khác hoàn tất (commit) hoặc hủy bỏ (abort)

Redo – logging và checkpoint

- Khi DBMS muốn ghi checkpoint
  - Ghi mẫu tin <start ckpt (Ti, …, Tk)> với Ti, …, Tk là những giao tác **chưa hoàn tất** và flush-log
  - Thực hiện ghi vào ổ đĩa **hết tất cả** những đơn vị dữ liệu đang ở trên buffer (mà chưa được ghi xuống ổ đĩa) của những giao tác đã commit
  - Ghi mẫu tin <end ckpt> vào nhật ký và flush-log
  - Lưu ý: không cần đợi các giao tác Ti, …, Tk hoàn tất hoặc hủy bỏ

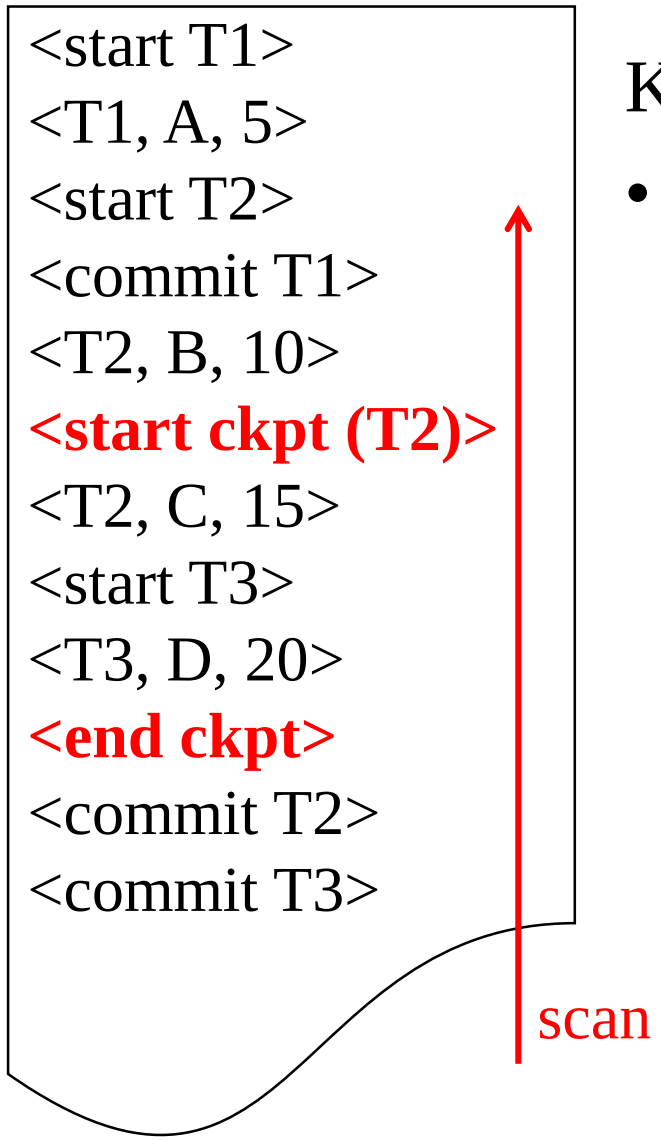
Redo – logging và checkpoint

<start T1>
<T1, A, 5>
<start T2>
<commit T1>
<T2, B, 10>
<start ckpt (T2)>
<T2, C, 15>
<start T3>
<T3, D, 20>
<end ckpt>
<commit T2>
<commit T3>

- T1 đã hoàn tất trước <start ckpt>
  - Có thể A đã được ghi xuống đĩa
  - Nếu chưa thì trước khi <end ckpt> cũng được ghi xuống đĩa
- Sau <start ckpt>
  - T2 đang thực thi
  - T3 bắt đầu
- Sau <end ckpt>
  - T2 và T3 hoàn tất (nhưng những giá trị được thay đổi bởi T2, và T3 có thể chưa được ghi vào ổ đĩa)

Redo – logging và checkpoint

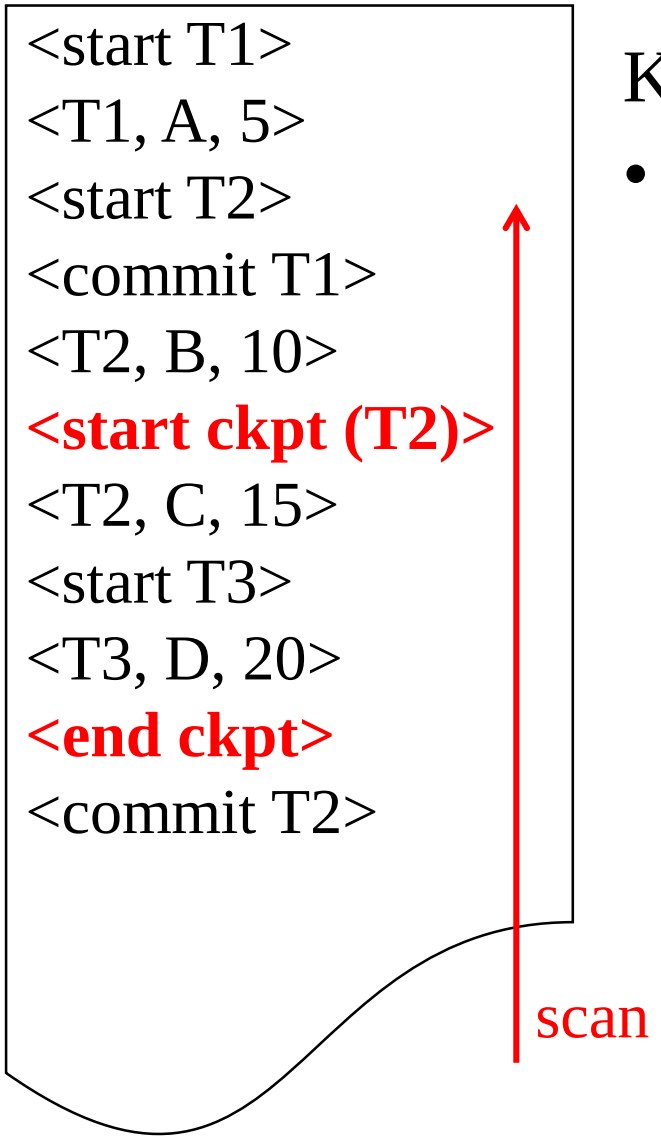
<start T1>
<T1, A, 5>
<start T2>
<commit T1>
<T2, B, 10>
<start ckpt (T2)>
<T2, C, 15>
<start T3>
<T3, D, 20>
<end ckpt>
<commit T2>
<commit T3>



Khôi phục dữ liệu:

- Quét nhật ký từ dưới lên
  - Tìm thấy <end ckpt>
    - Các giá trị của giao tác kết thúc trước <start ckpt(…)> đã được ghi đầy đủ lên ổ đĩa - ở đây là T1
    - Các giao tác bắt đầu từ sau <start ckpt (T2)> và các giao tác trong danh sách <start ckpt (T2)> phải được xem xét. Ta có được hai giao tác (T2, T3). Mà T2 và T3 đã commit, nên thực hiện lại (redo) T2 và T3
  - Thực hiện redo T2, T3
    - Tiếp tục quét nhật ký đến <start T2> rồi redo lại T2, T3 theo chiều từ trên xuống: B=10, C=15, D=20

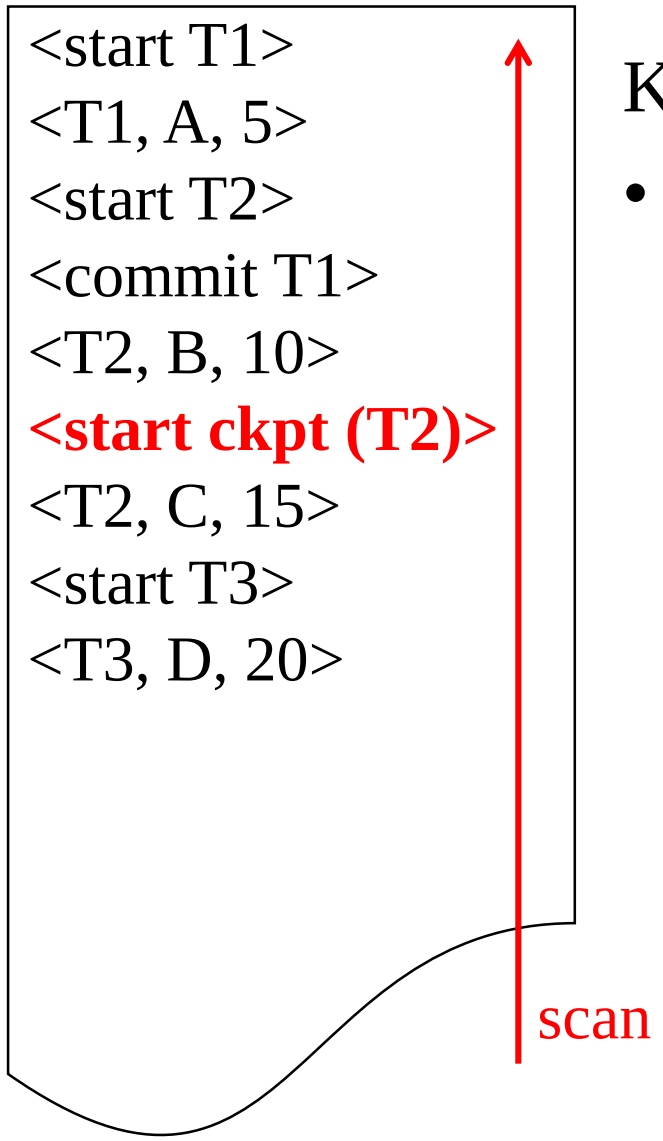
Redo – logging và checkpoint



Khôi phục dữ liệu

- Quét nhật ký từ dưới lên
  - Tìm thấy <end ckpt>
    - Các giá trị của giao tác kết thúc trước <start ckpt(…)> đã được ghi đầy đủ lên ổ đĩa - ở đây là T1
    - Các giao tác bắt đầu từ sau <start ckpt (T2)> và các giao tác trong danh sách <start ckpt (T2)> phải được xem xét. Ta có được hai giao tác (T2, T3). Mà T2 đã commit, nên thực hiện lại (redo) T2.
  - Thực hiện redo T2
    - Tiếp tục quét nhật ký đến <start T2> rồi redo lại T2 theo chiều từ trên xuống: B=10, C=15
  - T3 chưa commit nên sau khi hoàn tất việc khôi phục, ghi<Abort T3> vào nhật ký

Redo – logging và checkpoint



Khôi phục dữ liệu

- Quét nhật ký từ dưới lên
  - Không thấy <end ckpt> mà thấy <start ckpt(T2)>
    - Từ <start ckpt(T2) tiếp tục quét ngược lên trên để tìm <start ckpt …> liền trước đó để tìm ra danh sách các giao tác đang hoạt động (active)
    - Trong ví dụ này không có checkpoint phía trước đó thì phải đi đến đầu của nhật ký
    - Xác định được T1 đã commit, redo lại T1. Ghi lại các giá trị A=5.
  - Ghi <abort T2> và <abort T3> vào nhật ký

Nhận xét

- Trong quá trình khôi phục
  - Undo
    - Hủy bỏ các giao tác chưa commit (undo)
    - Bỏ qua giao tác đã commit
  - Redo
    - Bỏ qua giao tác chưa commit
    - Thực hiện lại những thay đổi được tạo ra bởi giao tác đã commit (redo)

Redo – logging – Bài tập

- Bên cạnh là file redo-log ghi nhận hai giao tác T và U
- Mô tả hành động khôi phục dữ liệu nếu ‘có sự cố xảy ra và mẫu tin nhật ký cuối cùng trên ổ đĩa’ là
  1. <START U>
  2. <COMMIT U>
  3. <T, E, 50>
  4. <COMMIT T>

<START T>
<T, A, 10>
<START U>
<U, B, 20>
<T, C, 30>
<U, D, 40>
<COMMIT U>
<T, E, 50>
<COMMIT T>

Redo – logging – Bài tập

- Bên cạnh là file redo-log
- Giả sử một nonquiescent checkpoint bắt đầu ngay sau khi một trong những record được ghi (in memory)
  a. <S, A, 60>
  b. <T, A, 10>
  c. <U, B, 20>
  d. <U, D, 40>
  e. <T, E, 50>
- Đối với mỗi trường hợp
  - Khi nào <END CKPT> được ghi
  - Và nếu có sự cố, phải đi bao xa để tìm những giao tác chưa hoàn tất. Cần xem xét trường hợp <END CKPT> được ghi trước và sau khi sự cố xảy ra

<START S>
<S, A, 60>
<COMMIT S>
<START T>
<T, A, 10>
<START U>
<U, B, 20>
<T, C, 30>
<START V>
<U, D, 40>
<V, F, 70>
<COMMIT U>
<T, E, 50>
<COMMIT T>
<V, B, 80>
<COMMIT V>

4

Phương pháp khôi phục – Recovery method

Undo-logging

Redo-logging

Undo/Redo logging

Undo/redo logging

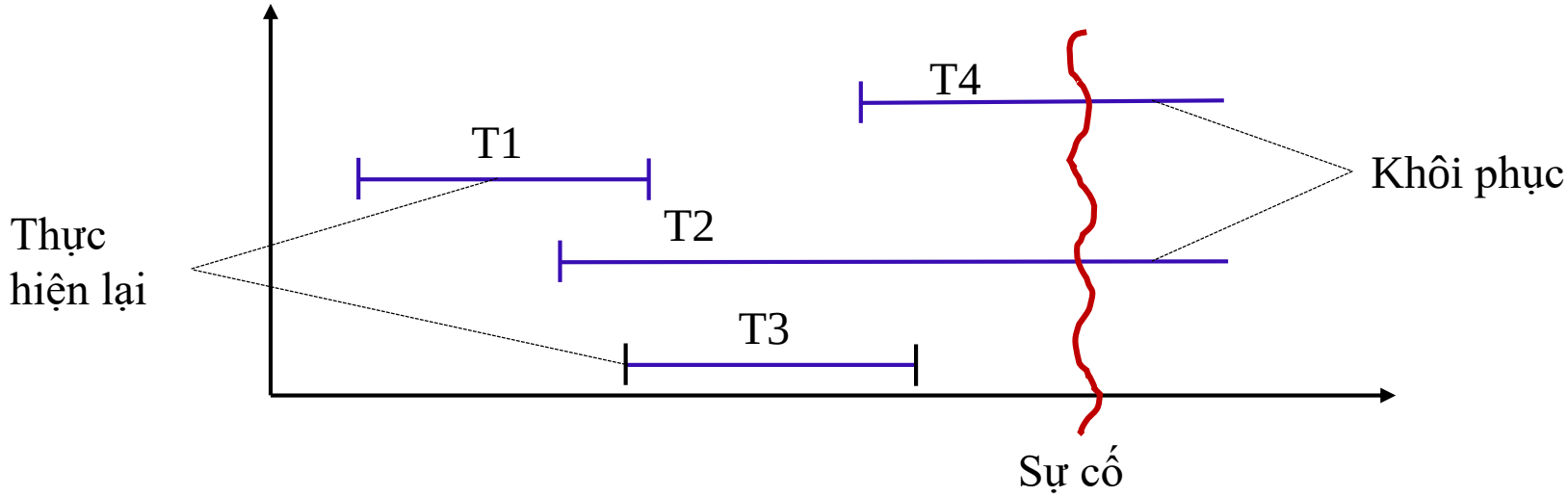
- Quy tắc
  - Log record
    - <T, X, v, w>
    - v là giá trị cũ
    - w là giá trị mới
  - Trước khi X được ghi vào ổ đĩa thì <T, X, v, w> có trong nhật ký trên ổ đĩa
  - <commit> có thể được ghi trước hoặc sau khi có sự thay đổi đơn vị dữ liệu trên ổ đĩa

Undo/redo logging

Step	Operation	t	Mem A	Mem B	Disk A	Disk B	Log
1							<START T>
2	Read(A,t)	8	8		8	8	
3	t=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8, 16>
5	Read(B,t)	8	16	8	8	8	
6	t=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8,16>
8	**FLUSH LOG**						
9	Output(A)	16	16	16	16	8	
10							<COMMIT T>
11	Output(B)	16	16	16	16	16	

Undo/redo logging

- Khôi phục dữ liệu
  - Thực hiện lại (redo) tất cả các giao tác đã commit
    - Theo thứ thự từ đầu đến cuối nhật ký
  - Khôi phục lại (undo) những giao tác chưa kết thúc
    - Theo thứ tự tứ cuối đến đầu nhật ký



Undo/redo logging

Step	Operation	t	Mem A	Mem B	Disk A	Disk B	Log
1							<START T>
2	Read(A,t)	8	8		8	8	
3	t=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8, 16>
5	Read(B,t)	8	16	8	8	8	
6	t=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8,16>
8	**FLUSH LOG**						
9	Output(A)	16	16	16	16	8	
10							<COMMIT T>
11	Output(B)	16	16	16	16	16	

Chưa kết thúc, khôi phục lại A=8, B=8

Nếu <commit T> đã được ghi xuống ổ đĩa, thực hiện lại T, A=16 và B=16

Undo/redo logging và checkpoint

- Tại thời điểm cần ghi checkpoint
  - Ghi <start ckpt (T1, …, Tn)> vào nhật ký, và flush-log vào ổ đĩa
    - T1, …, Tn là những giao tác đang thực thi
  - Thực hiện ghi vào ổ đĩa **hết tất cả** những đơn vị dữ liệu đang ở trên buffer
    - Những đơn vị dữ liệu được cập nhật bởi các giao tác. Kể cả những dữ liệu đã commit hay chưa commit.
  - Tạo mẫu tin <end ckpt> trong nhật ký và flush-log vào ổ đĩa

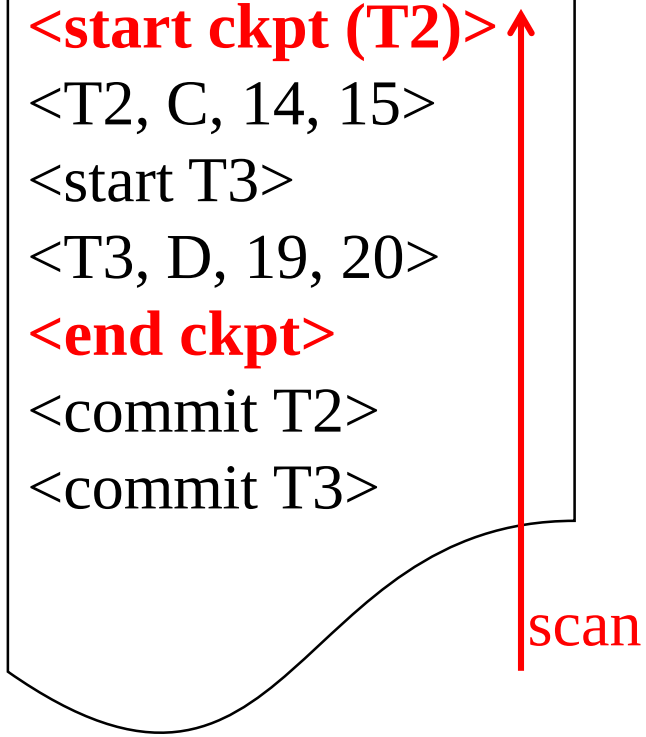
Undo/redo logging và checkpoint

<start T1>
<T1, A, 4, 5>
<start T2>
<commit T1>
<T2, B, 9, 10>
<start ckpt (T2)>
<T2, C, 14, 15>
<start T3>
<T3, D, 19, 20>
<end ckpt>
<commit T2>
<commit T3>

- T1 hoàn thành trước <start ckpt>
  - Giá trị của A có thể được ghi vào ổ đĩa
  - Nếu không, thì giá trị của A cũng được ghi xuống ổ đĩa trước <end ckpt>
- Giá trị B=10 đã được ghi xuống ổ đĩa

Undo/redo logging và checkpoint

<start T1>
<T1, A, 4, 5>
<start T2>
<commit T1>
<T2, B, 9, 10>
<start ckpt (T2)>
<T2, C, 14, 15>
<start T3>
<T3, D, 19, 20>
<end ckpt>
<commit T2>
<commit T3>

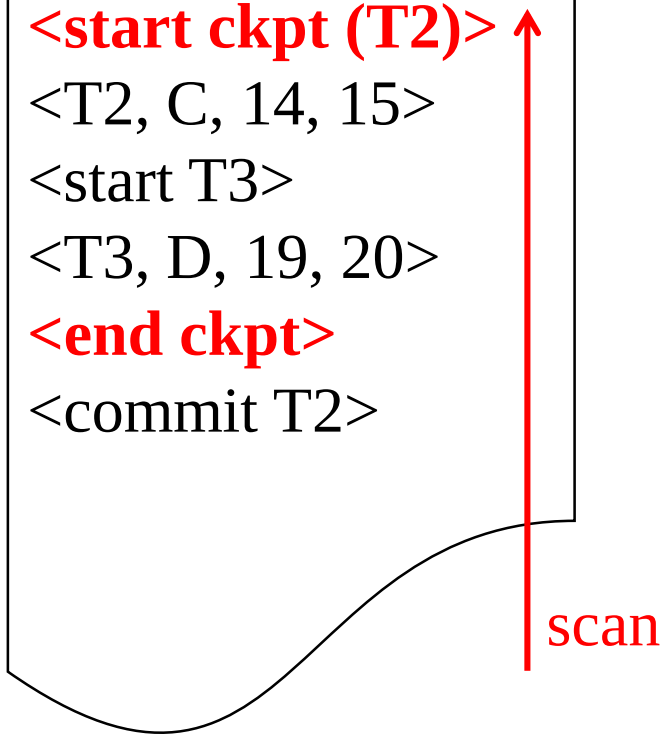


Khôi phục dữ liệu (recovery)

- Tìm thấy <end ckpt>
  - Không làm gì với T1
- Xem xét T2 và T3
  - <commit T2>
    - Redo T2
    - Ghi C=15
    - Không cần ghi B
  - <commit T3>
    - Redo T3
    - Ghi D=20

Undo/redo logging và checkpoint

<start T1>
<T1, A, 4, 5>
<start T2>
<commit T1>
<T2, B, 9, 10>
<start ckpt (T2)>
<T2, C, 14, 15>
<start T3>
<T3, D, 19, 20>
<end ckpt>
<commit T2>

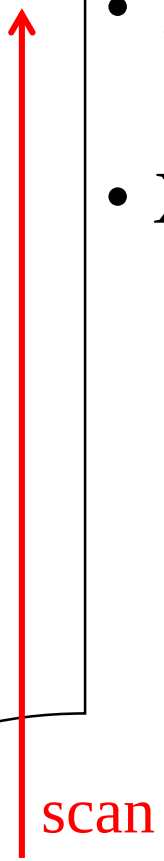


Khôi phục dữ liệu (recovery)

- Tìm thấy <end ckpt>
  - Không làm gì với T1
- Xem xét T2 và T3
  - <commit T2>
    - Redo T2
    - Ghi C=15
    - Không cần ghi B
  - T3 chưa commit
    - Khôi phục lại D=19

Undo/redo logging và checkpoint

<start T1>
<T1, A, 4, 5>
<start T2>
<commit T1>
<start T3>
<T2, B, 9, 10>
<T3, E, 6, 7>
<start ckpt (T2, T3)>
<T2, C, 14, 15>
<T3, D, 19, 20>
<end ckpt>
<commit T2>



Khôi phục dữ liệu (recovery)

- Tìm thấy <end ckpt>
  - Không làm gì với T1
- Xem xét T2 và T3
  - <commit T2>
    - Redo T2
    - Ghi C=15
    - Không cần ghi B
  - T3 chưa commit
    - Khôi phục lại D=19 và E=6

Undo/redo logging – bài tập

- Bên cạnh là file undo/redo-log
- Mô tả hành động khôi phục dữ liệu nếu ‘có sự cố xảy ra và mẫu tin nhật ký cuối cùng trên ổ đĩa’ là
  1. <START U>
  2. <COMMIT U>
  3. <T, E, 50>
  4. <COMMIT T>

<START T>
<T, A, 10, 11>
<START U>
<U, B, 20, 21>
<T, C, 30, 31>
<U, D, 40, 41>
<COMMIT U>
<T, E, 50, 51>
<COMMIT T>

Undo/redo logging – bài tập

- Bên cạnh là file redo-log
- Giả sử một nonquiescent checkpoint bắt đầu ngay sau khi một trong những record được ghi (in memory)
  a. <S, A, 60>
  b. <T, A, 61, 62>
  c. <U, B, 20, 21>
  d. <U, D, 40, 41>
  e. <T, E, 50, 51>
- Đối với mỗi trường hợp
  - Khi nào <END CKPT> được ghi
  - Và nếu có sự cố, phải đi bao xa để tìm những giao tác chưa hoàn tất. Cần xem xét cả hai trường hợp <END CKPT> được ghi trước và sau khi sự cố xảy ra

<START S>
<S, A, 60, 61>
<COMMIT S>
<START T>
<T, A, 61, 62>
<START U>
<U, B, 20, 21>
<T, C, 30, 31>
<START V>
<U, D, 40, 41>
<V, F, 70, 71>
<COMMIT U>
<T, E, 50, 51>
<COMMIT T>
<V, B, 21, 22>
<COMMIT V>