IS

Hệ quản trị Cơ sở dữ liệu

Chương 5: Khôi phục sự cố

1

IS

Nội dung chi tiết

- Giới thiệu
- Phân loại sự cố
- Mục tiêu của khôi phục sự cố
- Nhật ký giao tác (transaction log)
- Điểm lưu trữ (checkpoint)
  - Checkpoint đơn giản
  - Checkpoint linh động (nonquiescent checkpoint)
- Phương pháp khôi phục
  - Undo-Logging (immediate modification)
  - Redo-Logging (deferred modification)
  - Undo/Redo Logging

2

IS

Nhắc lại

- ❑ Tính toàn vẹn - đúng đắn, chính xác của dữ liệu
- ❑ Tính toàn vẹn - nhất quán của ràng buộc
- ❑ Trạng thái nhất quán
  - Thỏa các ràng buộc toàn vẹn
- ❑ CSDL nhất quán
  - CSDL ở trạng thái nhất quán

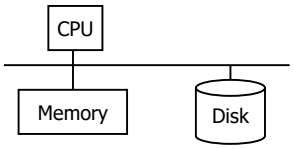
3

Giới thiệu

- Làm thế nào mà ràng buộc bị vi phạm?
  - Lỗi lập trình của các giao tác (transaction bug)
  - Lỗi lập trình của DBMS (DBMS bug)
  - Hư hỏng phần cứng (hardware failure)
  - Chia sẻ dữ liệu (data sharing)
- Làm thế nào để sửa lỗi và khôi phục?
  - Chia sẻ dữ liệu - chương 4
  - Sự cố - chương 5

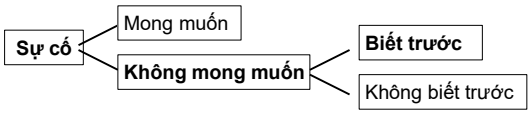
4

Sự cố



5

Phân loại sự cố



- Sự cố do nhập liệu sai
- Sự cố trên thiết bị lưu trữ (media failure)
- Sự cố của giao dịch (transaction failure)
- Sự cố liên quan đến hệ thống (system failure)

6

Sự cố do nhập liệu sai

- Dữ liệu sai hiển nhiên
  - Nhập thiếu 1 số trong dãy số điện thoại
- Dữ liệu sai không thể phát hiện
  - Nhập sai 1 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 giá trị
  - Trigger

7

Sự cố trên thiết bị lưu trữ

- ❑ Mất dữ liệu trên thiết bị lưu trữ
- ❑ Không thể truy cập lên thiết bị lưu trữ
- ❑ Ví dụ
  - Đầu đọc của đĩa cứng hư
  - Sector trên đĩa cứng hư
- ❑ DBMS áp dụng
  - Kỹ thuật RAID
  - Duy trì CSDL trên băng từ hoặc đĩa quang (archive)

8

Sự cố của giao tác

- ❑ Sự cố làm cho 1 giao tác kết thúc không bình thường
- ❑ Ví dụ
  - Chia cho không
  - Giao tác bị hủy
  - Dữ liệu nhập sai
  - Tràn số
- ❑ DBMS thực hiện lại giao tác

9

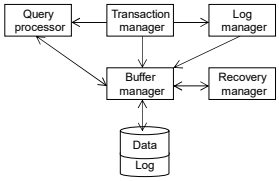
Sự cố hệ thống

- Mất dữ liệu của bộ nhớ trong
- Không thể truy cập bộ nhớ trong
- Ví dụ
  - Cúp điện
  - Lỗi phần mềm DBMS hoặc OS
  - Hư RAM
- DBMS cần cứu chữa và phục hồi dữ liệu
  - Nhật ký giao tác (transaction log)

10

Mục tiêu của khôi phục sự cố

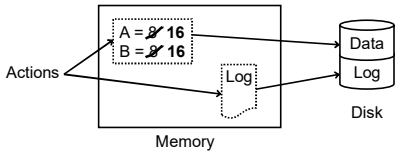
- Đưa CSDL về trạng thái nhất quán sau cùng nhất trước khi xảy ra sự cố
- Đảm bảo 2 tính chất của giao tác
  - Nguyên tố (atomic)
  - Bền vững (durability)



11

Nhật ký giao tác

- Nhật ký giao tác là một chuỗi các mẫu tin (log record) ghi nhận lại các hành động của DBMS
  - Một mẫu tin cho biết một giao tác nào đó đã làm những gì
- Nhật ký là một tập tin tuần tự được lưu trữ trên bộ nhớ chính, và sẽ được ghi xuống đĩa ngay khi có thể



12

Nhật ký giao tác (tt)

- Mẫu tin nhật ký gồm có
  - <start T>
    - Ghi nhận giao tác T bắt đầu hoạt động
  - <commit T>
    - Ghi nhận giao tác T đã hoàn tất
  - <abort T>
    - Ghi nhận giao tác T bị hủy
  - <T, X, v, w>
    - Ghi nhận giao tác T cập nhật lên đơn vị dữ liệu X
    - X có giá trị trước khi cập nhật là v và sau khi cập nhật là w

13

Nhật ký giao tác (tt)

- Khi sự cố hệ thống xảy ra
  - DBMS sẽ tra cứu nhật ký giao tác để khôi phục những gì mà các giao tác đã làm
- Để sửa chữa các sự cố
  - Một vài giao tác sẽ phải thực hiện lại (redo)
    - Những giá trị đã cập nhật xuống CSDL sẽ phải cập nhật lần nữa
  - Một vài giao tác không cần phải thực hiện lại (undo)
    - CSDL sẽ được khôi phục về lại trạng thái trước khi thực hiện

14

Nội dung chi tiết

- Giới thiệu
- Phân loại sự cố
- Mục tiêu của khôi phục sự cố
- Nhật ký giao tác (transaction log)
- Điểm lưu trữ (checkpoint)
  - Checkpoint đơn giản
  - Checkpoint linh động (nonquiescent checkpoint)
- Phương pháp khôi phục

15

Điểm lưu trữ

- Quá trình tra cứu nhật ký mất nhiều thời gian
  - Do phải quét hết tập tin nhật ký
- Thực hiện lại các giao tác đã được ghi xuống đĩa làm cho việc phục hồi diễn ra lâu hơn

→ Checkpoint

- Nhật ký giao tác có thêm mẫu tin <checkpoint> hay <ckpt>
- Mẫu tin <checkpoint> sẽ được ghi xuống nhật ký định kỳ
  - Vào thời điểm mà DBMS ghi tất cả những gì thay đổi của CSDL từ vùng đệm xuống đĩa

16

Điểm lưu trữ đơn giản

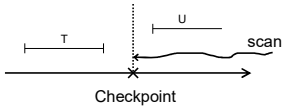
- Khi đến điểm lưu trữ, DBMS
  - (1) Tạm dừng tiếp nhận các giao tác mới
  - (2) Đợi các giao tác đang thực hiện
    - Hoặc là hoàn tất (commit)
    - Hoặc là hủy bỏ (abort)

    và ghi mẫu tin <commit T> hay <abort T> vào nhật ký
  - (3) Tiến hành ghi nhật ký từ vùng đệm xuống đĩa
  - (4) Tạo 1 mẫu tin <checkpoint> và ghi xuống đĩa
  - (5) Tiếp tục nhận các giao tác mới

17

Điểm lưu trữ đơn giản (tt)

- ❑ Các giao tác ở phía trước điểm lưu trữ là những giao tác đã kết thúc → không cần làm lại
- ❑ Và sau điểm lưu trữ là những giao tác chưa thực hiện xong → cần khôi phục
- ❑ Không phải duyệt hết nhật ký
  - Duyệt từ cuối nhật ký đến điểm lưu trữ

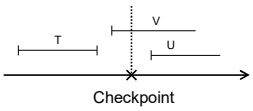


18

Điểm lưu trữ linh động

- Trong thời gian checkpoint hệ thống gần như tạm ngưng hoạt động
  - Chờ các giao tác hoàn tất hoặc hủy bỏ

→ Nonquiescent checkpoint
  - Cho phép tiếp nhận các giao tác mới trong quá trình checkpoint
  - Mẫu tin <start ckpt (T_1, T_2, …, T_k)>
  - Mẫu tin <end ckpt>



19

Điểm lưu trữ linh động (tt)

- Khi đến điểm lưu trữ, DBMS
  - (1) Tạo mẫu tin <start ckpt (T_1, T_2,…, T_k)> và ghi xuống đĩa
    - T_1, T_2, …, T_k là những giao tác đang thực thi
  - (2) Chờ cho đến khi T_1, T_2, …, T_k hoàn tất hay hủy bỏ, nhưng không ngăn các giao tác mới bắt đầu
  - (3) Khi T_1, T_2, …, T_k thực hiện xong, tạo mẫu tin <end ckpt> và ghi xuống đĩa

20

Nội dung chi tiết

- Giới thiệu
- Phân loại sự cố
- Mục tiêu của khôi phục sự cố
- Nhật ký giao tác (transaction log)
- Điểm lưu trữ (checkpoint)
- Phương pháp khôi phục
  - Undo-Logging (immediate modification)
  - Redo-Logging (deferred modification)
  - Undo/Redo Logging

21

Phương pháp Undo-Logging

- Qui tắc
  - (1) Một thao tác phát sinh ra 1 mẫu tin nhật ký
    - Mẫu tin của thao tác cập nhật chỉ ghi nhận lại giá trị cũ
    - <T, X, v>
  - (2) Trước khi X được cập nhật xuống đĩa, mẫu tin <T, X, v> đã phải có trên đĩa
  - (3) Trước khi mẫu tin <commit, T> được ghi xuống đĩa, tất cả các cập nhật của T đã được phản ánh lên đĩa
    - Flush-log: chỉ chép những block mẫu tin nhật ký mới chưa được chép trước đó

22

Ví dụ

Bước	Hành động	t	Mem A	Mem B	Disk A	Disk B	Mem 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**						

23

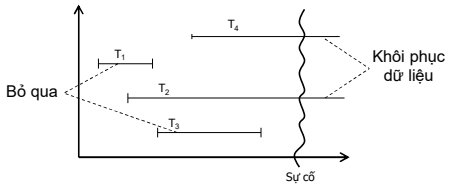
Phương pháp Undo-Logging (tt)

- Khôi phục
  - (1) Gọi S là tập các giao tác chưa kết thúc
    - Có <start T_i> trong nhật ký nhưng
    - Không có <commit T_i> hay <abort T_i> trong nhật ký
  - (2) Với mỗi mẫu tin <T_i, X, v> trong nhật ký (theo thứ tự cuối tập tin đến đầu tập tin)
    - Nếu $T_i \in S$ thì
      - Write(X, v)
      - Output(X)
  - (3) Với mỗi $T_i \in S$
    - Ghi mẫu tin <abort T_i> lên nhật ký

24

Phương pháp Undo-Logging (tt)

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



25

Ví dụ

Bước	Hành động	t	Mem A	Mem B	Disk A	Disk B	Mem 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>	**A và B không thay đổi nên không cần khôi phục**
8	**Flush log**							
9	Output(A)	16	16	16	16	8		
10	Output(B)	16	16	16	16	16		
11							<commit T>	**Khôi phục A=8 và B=8**
12	**Flush log**							**Không cần khôi phục A và B**

26

Undo-Logging & Checkpoint

```
<start T1>
<T1, A, 5>
<start T2>
<T2, B, 10>
```

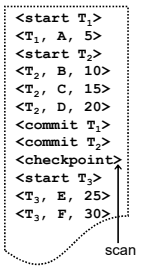
Checkpoint

- Vì T_1 và T_2 đang thực thi nên chờ
- Sau khi T_1 và T_2 hoàn tất hoặc hủy bỏ
- Ghi mẫu tin <checkpoint> lên nhật ký

27

Undo-Logging & Checkpoint (tt)

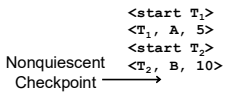
❑ Ví dụ



- <T_3, F, 30>
  - T_3 chưa kết thúc
  - Khôi phục F=30
- < T_3, E, 25 >
  - Khôi phục E=25
- <checkpoint>
  - Dừng

28

Undo-Logging & Checkpoint (tt)

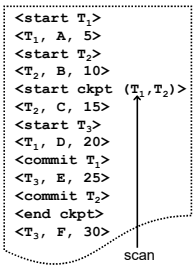


- Vì T_1 và T_2 đang thực thi nên tạo <start ckpt (T_1,T_2)>
- Trong khi chờ T_1 và T_2 kết thúc, DBMS vẫn tiếp nhận các giao tác mới
- Sau khi T_1 và T_2 kết thúc, ghi <end ckpt> lên nhật ký

29

Undo-Logging & Checkpoint (tt)

❑ Ví dụ 1 >

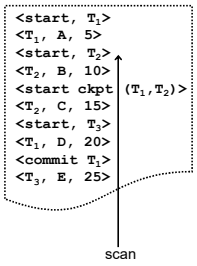


- <T_3, F, 30>
  - T_3 chưa kết thúc
  - Khôi phục F=30
- <end ckpt>
  - Những giao tác bắt đầu trước <start ckpt> đã hoàn tất
  - T_1 và T_2 đã hoàn tất
- < T_3, E, 25 >
  - Khôi phục E=25
- <start ckpt (T_1, T_2)>
  - Dừng

30

Undo-Logging & Checkpoint (tt)

❑ Ví dụ 2



- <T_3, E, 25>
  - T_3 chưa kết thúc
  - Khôi phục E=25
- <commit T_1>
  - T_1 bắt đầu trước <start ckpt> và đã hoàn tất
- <T_2, C, 15>
  - T_2 bắt đầu trước <start ckpt> và chưa kết thúc
  - Khôi phục C=15
- <start ckpt (T_1, T_2)>
  - Chỉ có T_1 và T_2 bắt đầu trước đó
- < T_2, B, 10 >
  - Khôi phục B=10
- <start T_2>
  - Dừng

31

Phương pháp Redo-Logging

- Qui tắc
  - (1) Một thao tác phát sinh ra 1 mẫu tin nhật ký
    - Mẫu tin của thao tác cập nhật chỉ ghi nhận lại giá trị mới
    - <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 trên đĩa
    - Bao gồm mẫu tin cập nhật <T, X, w> và <commit T>
  - (3) Khi T hoàn tất, tiến hành ghi nhật ký xuống đĩa
    - Flush-log: chỉ chép những block mẫu tin nhật ký mới chưa được chép trước đó

32

Ví dụ

Bước	Hành động	t	Mem A	Mem B	Disk A	Disk B	Mem 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	

33

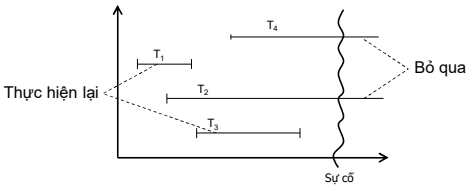
Phương pháp Redo-Logging (tt)

- Khôi phục
  - (1) Gọi S là tập các giao tác hoàn tất
    - Có mẫu tin <commit T_i> trong nhật ký
  - (2) Với mỗi mẫu tin <T_i, X, w> trong nhật ký (theo thứ tự cuối tập tin đến đầu tập tin)
    - Nếu $T_i \in S$ thì { - Write(X, w); - Output(X) }
  - (3) Với mỗi $T_j \notin S$
    - Ghi mẫu tin <abort T_j> lên nhật ký

34

Phương pháp Redo-Logging (tt)

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



35

Ví dụ

Bước	Hành động	t	Mem A	Mem B	Disk A	Disk B	Mem 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>	**Xem như T chưa hoàn tất, A và B không có thay đổi**
9	**Flush log**							
10	Output(A)	16	16	16	16	8		**Thực hiện lại T, ghi A=16 và B=16**
11	Output(B)	16	16	16	16	16		**Thực hiện lại T, ghi A=16 và B=16**

36

Redo-Logging & 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 xuống đĩa những dữ liệu đã hoàn tất mà trước đó chưa được ghi
  - <end ckpt>
  - Mẫu tin <end ckpt> được ghi vào nhật ký mà không phải đợi các giao tác hoàn tất (commit) hoặc hủy bỏ (abort)

37

Redo-Logging & Checkpoint (tt)

❑ Đến điểm lưu trữ, DBMS
  - (1) Tạo mẫu tin <start ckpt (T_1, T_2,…, T_k)> và ghi xuống đĩa
    - T_1, T_2, …, T_k là những giao tác đang thực thi
  - (2) Ghi xuống đĩa những dữ liệu của các giao tác đã hoàn tất trên vùng đệm
  - (3) Tạo mẫu tin <end ckpt> và ghi xuống đĩa

38

Redo-Logging & Checkpoint (tt)

❑ Ví dụ 1

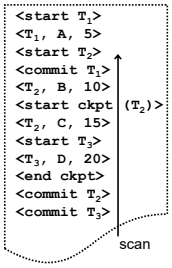
```
<start T1>
<T1, A, 5>
<start T2>
<commit T1>
<T2, B, 10>
<start ckpt (T2)>
<T2, C, 15>
<start T3>
<T3, D, 20>
<end ckpt>
<commit T2>
<commit T3>
```

- T_1 đã hoàn tất trước <start ckpt>
  - Có thể đã được ghi xuống đĩa
  - Nếu chưa thì trước khi <end ckpt> cũng được ghi xuống đĩa
- Sau <start ckpt>
  - T_2 đang thực thi
  - T_3 bắt đầu
- Sau <end ckpt>
  - T_2 và T_3 hoàn tất

39

Redo-Logging & Checkpoint (tt)

❑ Ví dụ 1

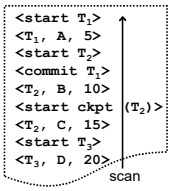


- Tìm thấy <end ckpt>
- Chỉ xét T_2 và T_3
- <commit T_2>
  - Thực hiện lại T_2
  - Ghi C=15 và B=10
- <commit T_3>
  - Thực hiện lại T_3
  - Ghi D=20

40

Redo-Logging & Checkpoint (tt)

❑ Ví dụ 2



- T_2 và T_3 chưa hoàn tất
  - Không thực hiện lại
- T_1 đã hoàn tất
  - Thực hiện lại T_1
  - Ghi A=5

41

Nhận xét

❑ Undo-logging (immediate modification)
- Khi giao tác kết thúc, dữ liệu được ghi xuống đĩa ngay lập tức
- Truy xuất đĩa nhiều

❑ Redo-logging (deferred modification)
- Giữ lại các cập nhật trên vùng đệm cho đến khi giao tác hoàn tất và mẫu tin nhật ký được ghi xuống đĩa
- Tốn nhiều bộ nhớ

42

IS Phương pháp Undo/Redo-Logging

- Qui tắc
  - (1) Một thao tác phát sinh ra 1 mẫu tin nhật ký
    - Mẫu tin của thao tác cập nhật ghi nhận giá trị cũ và mới của một đơn vị dữ liệu
    - <T, X, v, w>
  - (2) Trước khi X được cập nhật xuống đĩa, các mẫu tin cập nhật <T, X, v, w> đã phải có trên đĩa
  - (3) Khi T hoàn tất, tạo mẫu tin <commit T> trên nhật ký và ghi xuống đĩa

43

IS Ví dụ

Bước	Hành động	t	Mem A	Mem B	Disk A	Disk B	Mem 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	

44

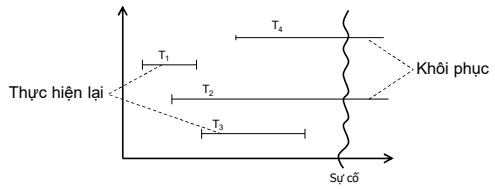
IS Phương pháp Undo/Redo-Logging (tt)

- Khôi phục
  - (1) Khôi phục lại (undo) những giao tác chưa kết thúc
    - Theo thứ tự từ cuối nhật ký đến đầu nhật ký
  - (2) Thực hiện lại (redo) những giao tác đã hoàn tất
    - Theo thứ tự từ đầu nhật ký đến cuối nhật ký

45

IS Phương pháp Undo/Redo-Logging (tt)

- Khi gặp sự cố
  - T_1 và T_3 đã hoàn tất
  - T_2 và T_4 chưa kết thúc



46

IS Ví dụ

Bước	Hành động	t	Mem A	Mem B	Disk A	Disk B	Mem 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		**T chưa kết thúc, khôi phục A=8**
10							<commit T>	**<commit T> đã được ghi xuống đĩa, thực hiện lại T, A=16 và B=16**
11	Output(B)	16	16	16	16	16		

47

IS Undo/Redo-Logging & Checkpoint

- Khi đến điểm lưu trữ, DBMS
  - (1) Tạo mẫu tin <start ckpt (T_1, T_2,…, T_k)> và ghi xuống đĩa
    - T_1, T_2, …, T_k là những giao tác đang thực thi
  - (2) Ghi xuống đĩa những dữ liệu đang nằm trên vùng đệm
    - Những đơn vị dữ liệu được cập nhật bởi các giao tác
  - (3) Tạo mẫu tin <end ckpt> trong nhật ký và ghi xuống đĩa

48

IS Undo/Redo-Logging & Checkpoint (tt)

❑ Ví dụ 1

<start T_1>
<T_1, A, 4, 5>
<start T_2>
<commit T_1>
<T_2, B, 9, 10>
<start ckpt (T_2)>
<T_2, C, 14, 15>
<start T_3>
<T_3, D, 19, 20>
<end ckpt>
<commit T_2>
<commit T_3>

- T_1 đã hoàn tất trước <start ckpt>
  - Có thể đã được ghi xuống đĩa
  - Nếu chưa thì trước khi <end ckpt> cũng được ghi xuống đĩa
- Giá trị B=10 đã được ghi xuống đĩa

49

IS Undo/Redo-Logging & Checkpoint (tt)

❑ Ví dụ 1

<start T_1>
<T_1, A, 4, 5>
<start T_2>
<commit T_1>
<T_2, B, 9, 10>
<start ckpt (T_2)>
<T_2, C, 14, 15>
<start T_3>
<T_3, D, 19, 20>
<end ckpt>
<commit T_2>
<commit T_3>
scan

- Tìm thấy <end ckpt>
  - T_1 không cần thực hiện lại
- Xét T_2 và T_3
- <commit T_2>
  - Thực hiện lại T_2 và ghi C=15
  - Không cần ghi B
- <commit T_3>
  - Thực hiện lại T_3 và ghi D=20

50

IS Undo/Redo-Logging & Checkpoint (tt)

❑ Ví dụ 2

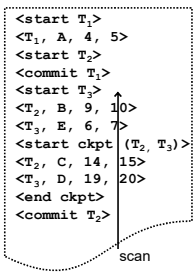
<start T_1>
<T_1, A, 4, 5>
<start T_2>
<commit T_1>
<T_2, B, 9, 10>
<start ckpt (T_2)>
<T_2, C, 14, 15>
<start T_3>
<T_3, D, 19, 20>
<end ckpt>
<commit T_2>
scan

- Tìm thấy <end ckpt>
  - T_1 không cần thực hiện lại
- Xét T_2 và T_3
- <commit T_2>
  - Thực hiện lại T_2 và ghi C=15
  - Không cần ghi B
- T_3 chưa kết thúc
  - Khôi phục D=19

51

IS Undo/Redo-Logging & Checkpoint (tt)

❑ Ví dụ 3



- Tìm thấy <end ckpt>
  - T_1 không cần thực hiện lại
- Xét T_2 và T_3
- <commit T_2>
  - Thực hiện lại T_2 và ghi C=15
  - Không cần ghi B
- T_3 chưa kết thúc
  - Khôi phục D=19 và E=6

52

IS

53