CÁC HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

CHƯƠNG 4

CÁC KỸ THUẬT PHỤC HỒI CSDL

Giảng viên: ThS. Nguyễn Thị Uyên Nhi
Email: uyennhisgu@gmail.com

KHOA CÔNG NGHỆ THÔNG TIN

Nội dung

- Lịch trình khả phục hồi.
- Tổng quan về phục hồi.
- Kĩ thuật Write-Ahead Logging.
- Kĩ thuật phục hồi dựa trên Deferred Update.
- Kĩ thuật phục hồi dựa trên Immediate Update.
- Kĩ thuật phục hồi dựa trên Shadow Paging.

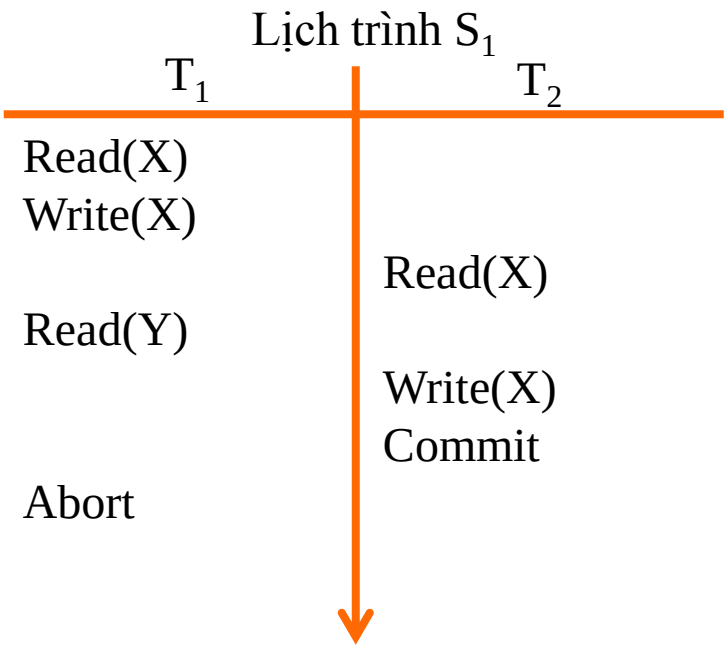
TÍNH KHẢ PHỤC HỒI CỦA LỊCH TRÌNH

- Trong việc tìm hiểu về điều khiển song hành, ta chưa xét nhiều đến sự thất bại của giao dịch.
- Nếu giao dịch Ti thất bại vì lý do nào đó (thường là các sự cố - failures), ta cần hủy bỏ giao dịch này để đảm bảo tính nguyên tử của giao dịch.
- Và để đảm bảo tính nhất quán, ta cần phải hủy bỏ tất cả các hiệu quả liên quan của giao dịch T.

TÍNH KHẢ PHỤC HỒI CỦA LỊCH TRÌNH

- Một số lịch trình dễ dàng phục hồi trong khi 1 số khác không thể phục hồi.
- Lịch trình mà có các giao dịch sau khi đã được bàn giao (Commit) không bao giờ phải rollback lại gọi là lịch trình khả phục hồi.
- Với mỗi cặp giao dịch Ti và Tj trong lịch trình khả phục hồi: nếu Ti đọc hạng mục dữ liệu được ghi bởi Tj thì lệnh commit của Tj phải diễn ra trước lệnh commit của Ti.

Ví dụ



- Giả sử trường hợp T1 gặp sự cố và phải rollback.
- T2?
- ➔ Lịch trình không thể phục hồi và không được phép.

Ví dụ

Lịch trình S_2

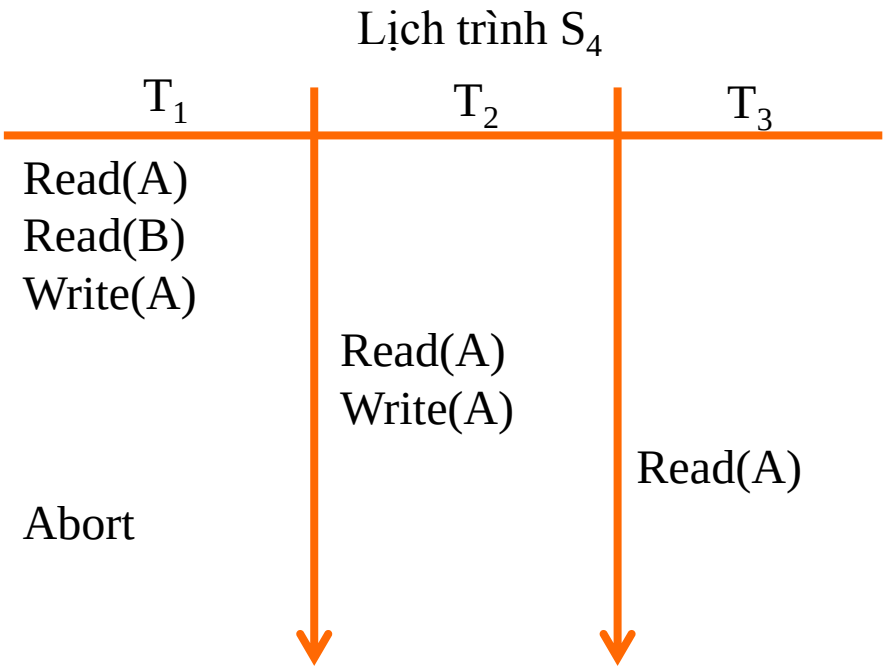
T_1	T_2
Read(X)	
	Read(X)
Write(X)	
Read(Y)	
	Write(X)
	Commit
Write(Y)	
Commit	

Lịch trình S_3

T_1	T_2
Read(X)	
Write(X)	
	Read(X)
Read(Y)	
	Write(X)
Write(Y)	
Commit	
	Commit

Khả phục hồi?

Lịch trình Cascadeless



- Ngay cả khi lịch trình là khả phục hồi, việc phục hồi đúng sau thất bại của một giao dịch cũng xảy ra vấn đề.
- Việc rollback của S4 diễn ra như thế nào?

Lịch trình Cascadeless

- Hiện tượng 1 giao dịch thất bại kéo theo một loạt các giao dịch khác phải rollback gọi là sự cuộn lại hàng loạt (cascading rollback).
- Việc này dẫn đến việc hủy bỏ một khối lượng công việc đáng kể.

- Các lịch trình không xảy ra cascading rollback được gọi là lịch trình cascadeless.

Lịch trình Cascadeless

- Một lịch trình cascadeless là một lịch trình trong đó mỗi cặp giao dịch T_i và T_j: Nếu T_i đọc một hạng mục dữ liệu được ghi trước đó bởi T_j thì lệnh commit của T_j phải diễn ra trước lệnh đọc của T_i.
- Nghĩa là: hành động commit của giao dịch ghi phải diễn ra trước hành động đọc của giao dịch đọc.

NGUYÊN TẮC CHUNG

- Khôi phục/phục hồi sau sự cố có nghĩa là CSDL khôi phục lại trạng thái nhất quán gần nhất trước thời điểm xảy ra sự cố.
- Để thực hiện được điều này, hệ thống cần lưu giữ các thông tin về sự thay đổi của dữ liệu bởi các giao dịch.

Cơ chế phục hồi cơ bản

- Nếu xảy ra sự cố nặng, hệ thống khôi phục lại dữ liệu được sao lưu trước đó và thực hiện lại (redo) các thao tác của các giao dịch đã được bàn giao.
- Nếu các sự cố nhẹ (1-4), hệ thống sẽ hủy bỏ các thao tác gây ra tính không nhất quán (undo). Trong một số tình huống, cần phải thực hiện lại một số thao tác (redo).

PHÂN LOẠI CƠ BẢN

- Về lý thuyết, ta có thể phân biệt 2 loại giải thuật phục hồi dựa trên:
  - Trì hoãn cập nhật (Deferred update): không cập nhật vật lý dữ liệu cho đến khi giao dịch hoàn tất.
  - Cập nhật ngay (Immediate update)
- Một giải thuật phục hồi thường gồm 2 phần:
  - Các hành động được thực hiện trong suốt quá trình hoạt động bình thường của hệ thống.
  - Các hành động thực hiện sau khi lỗi phát sinh.

Deferred Update Techniques

- Không cập nhật CSDL trên đĩa cho đến khi giao dịch được bàn giao.
- Trước đó, những thay đổi được lưu trong vùng làm việc cục bộ của giao dịch – local transaction workspace (buffers).
- Trong quá trình bàn giao, những thay đổi trước tiên được lưu trên log và sau đó ghi vào CSDL.

Deferred Update Techniques

- Nếu giao dịch lỗi trước khi bàn giao, nó sẽ không thay đổi CSDL do đó UNDO không cần thực hiện.
- Có thể cần thực hiện lại (REDO) một số tác động của giao dịch hoàn tất dựa trên log khi xảy ra sự cố khi chưa kịp ghi vào CSDL.

- ➔ **Giải thuật NO-UNDO/REDO.**

Immediate Update Techniques

- CSDL có thể được cập nhật bởi một số thao tác của giao dịch chưa được bàn giao.
- Việc cập nhật này thường được lưu vào log trên đĩa trước khi được cập nhật vào CSDL.
- Nếu sự cố xảy ra sau khi thay đổi CSDL nhưng trước thời điểm bàn giao, giao dịch phải được rollback.
- Thông thường cả UNDO và REDO đều cần.
- ➔ **Giải thuật UNDO/REDO và biến thể UNDO/NO-REDO.**

Caching of Disk Blocks

- Cơ chế phục hồi liên quan chặt chẽ đến hệ điều hành cụ thể là việc đọc dữ liệu vào bộ nhớ (caching/buffering).
- Một hay nhiều disk page có chứa dữ liệu cần được thay đổi sẽ được đọc vào bộ nhớ chính và sau khi thay đổi sẽ được ghi ngược lại CSDL.
- Đây là công việc của hệ điều hành, tuy nhiên do quan trọng với quy trình phục hồi → được điều khiển bởi HQT CSDL.

Caching of Disk Blocks

- Thông thường 1 số vùng đệm của bộ nhớ được điều khiển bởi HQT CSDL (DBMS) gọi là DBMS cache.
- Một directory cho cache để quản lý disk page nào nằm ở vùng đệm nào (tương tự khái niệm page tables của HĐH)
  - <Disk page address, buffer location>

Caching of Disk Blocks

- Khi DBMS thực hiện 1 hành động lên dữ liệu X nào đó:
  - Đầu tiên kiểm tra cache directory xem disk page có chứa X có nằm trong cache không.
  - Nếu không thấy sẽ chép vùng nhớ cần thiết vào cache, có thể sẽ cần replace (flush) 1 số vùng trong cache.
    - Least recently used (LRU)
    - First-in-first-out (FIFO)
    - 1 số cơ chế riêng của mỗi DBMS
- Với mỗi vùng đệm kèm theo 1 bit gọi là dirty bit để đánh dấu vùng đệm đó có được cập nhật hay không.

Caching of Disk Blocks

- 2 cơ chế được thực thi khi ghi ngược từ buffer vào đĩa:
  - In-place updating: ghi buffer ngược lại vị trí cũ trên đĩa.
  - Shadowing: ghi buffer vào địa chỉ khác trên đĩa, do đó tồn tại nhiều phiên bản của dữ liệu → không cần thiết log tuy nhiên không khả thi trên thực tế
- Cách gọi thông thường:
  - Dữ liệu cũ trước khi cập nhật: before image – BFIM.
  - After image – AFIM.
- Trong kĩ thuật Shadowing, cả BFIM và AFIM đều được ghi trên đĩa.

Khái niệm Kĩ thuật Write-Ahead Logging

- Trong quá trình ghi dữ liệu ngược từ cache vào đĩa, cần thiết phải sử dụng log để phục hồi trong những trường hợp xảy ra sự cố trong quá trình ghi này.
- Cơ chế phục hồi phải đảm bảo BFIM của dữ liệu được lưu trong log thích hợp và log được ghi vào đĩa trước khi BFIM được ghi đè bởi AFIM.

Khái niệm Kĩ thuật Write-Ahead Logging

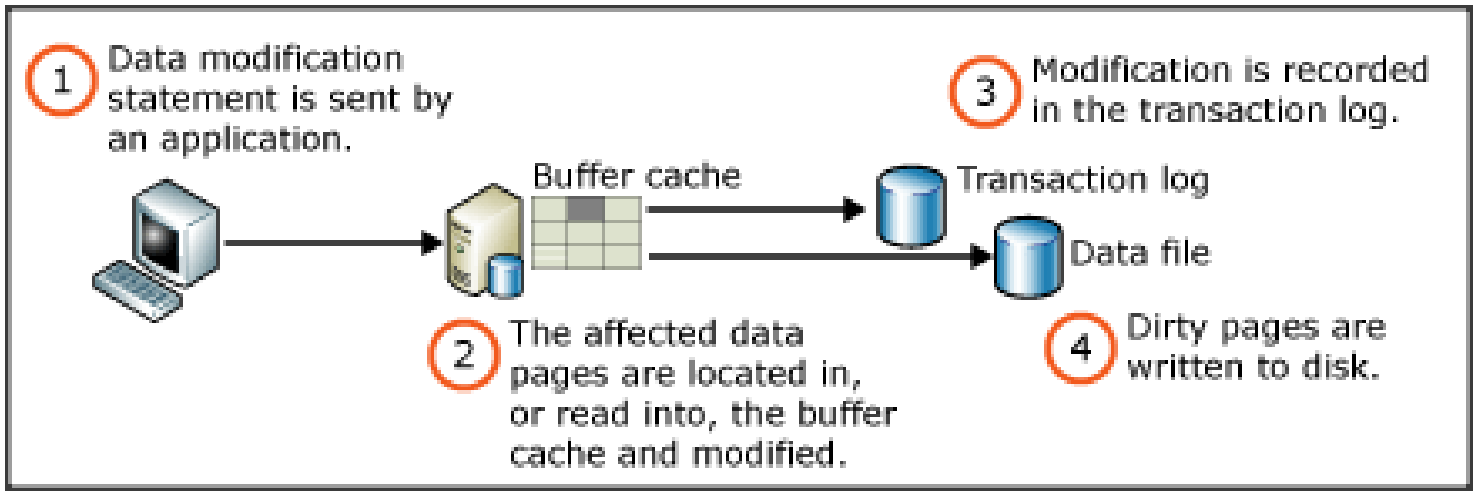
- 2 loại thông tin trong log cần thiết cho việc ghi dữ liệu:
  - Thông tin cần thiết cho việc UNDO: chứa giá trị cũ (BFIM) của dữ liệu, cần thiết cho việc phục hồi giá trị của dữ liệu về giá trị trước đó (bằng cách thay đổi giá trị trong database = giá trị BFIM).
  - Thông tin cần thiết cho việc REDO: giá trị mới (AFIM) của thao tác ghi, nó cần thiết cho việc thực hiện lại các thay đổi do thao tác ghi đó từ log (bằng cách thay đổi giá trị trong database = giá trị AFIM).

Khái niệm Kĩ thuật Write-Ahead Logging

- DBMS cache lưu database disk blocks gồm:
  - Data blocks
  - Index blocks
  - Log blocks
- Khi một dòng trong log được cập nhật, nó được ghi vào log blocks hiện tại trong DBMS cache.
- Khi có sự cập nhật dữ liệu trong data blocks, records log tương ứng được ghi tiếp vào cuối log blocks trong DBMS cache.

KHÁI NIỆM KĨ THUẬT WRITE-AHEAD LOGGING

- Trong cơ chế Write-Ahead Logging (WAL), log blocks tương ứng với data blocks (có sự thay đổi dữ liệu, cần được cập nhật lên đĩa) phải được ghi lên đĩa trước khi ghi chính data block đó ngược lại đĩa.



THUẬT NGỮ: STEAL/NO-STEAL

- No-steal: Nếu những cache page cập nhật bởi giao dịch không được phép ghi lên đĩa trước khi giao dịch commit. Thông thường dùng thêm 1 bit pin-unpin.
- Steal: Ngược lại, cho phép ghi cache ngay cả trước khi giao dịch commit.
  - Không cần buffer quá lớn để lưu các page có thay đổi.

THUẬT NGỮ: FORCE/NO-FORCE

- Force: Tất cả các page có cập nhật phải được ghi vào đĩa khi giao dịch commit. Ngược lại là No-force.
- Ưu điểm No-force:
  - Updated pages của những giao dịch đã commit có thể vẫn được giữ trong buffer để dùng cho các giao dịch khác do đó hạn chế chi phí I/O.
  - Rất lợi trong trường hợp một số vùng dữ liệu được thay đổi liên tục bởi nhiều giao dịch.
- Deferred Update: Không cập nhật CSDL trên đĩa cho đến khi giao dịch được bàn giao.

→ No-steal.

- Thông thường, DBMS thực hiện steal/no-force.

GIẢI THUẬT CHO WAL

- BFIM của dữ liệu trên đĩa không được phép ghi đè cho đến khi tất cả các record log thuộc loại UNDO của các giao dịch cập nhật được ghi lên đĩa.
- Thao tác commit của một giao dịch không được phép hoàn tất cho đến khi tất cả log thuộc loại UNDO và REDO cho giao dịch đó được ghi lên đĩa.

GIẢI THUẬT CHO WAL

- Để thuận lợi hơn cho quá trình phục hồi, hệ thống quản trị phục hồi có thể cần lưu giữ thông tin danh sách các giao dịch liên quan đang được xử lý:
  - Committed & aborted transactions.
  - Active transactions: các giao dịch đã bắt đầu nhưng chưa hoàn tất (commit).
- Quá trình phục hồi hiệu quả hơn.

CHECKPOINT

- Một loại dữ liệu khác trong log: [checkpoint]
- Thông thường được ghi vào log định kì tại thời điểm hệ thống ghi vào CSDL trên đĩa tất cả các page có chỉnh sửa trong buffer.
- Do đó tất cả giao dịch T đã được commit bởi dòng [commit, T] trong log trước thời điểm [checkpoint] không cần thiết REDO các thao tác write vì các thay đổi đã được ghi vào đĩa tại thời điểm [checkpoint].

CHECKPOINT

- Hệ thống quản trị phục hồi quy định khoảng thời gian giữa 2 lần checkpoint.
- Có thể dựa trên thời gian, ví dụ mỗi m phút.
- Hoặc có thể dựa trên số giao dịch t được commit tính từ thời điểm checkpoint trước đó.
- Giá trị m hay t đó là các tham số hệ thống.

CHECKPOINT

- Quy trình thực hiện một checkpoint:
  1. Tạm dừng hoạt động xử lý các giao dịch.
  2. Force-write tất cả buffer có thay đổi lên đĩa.
  3. Ghi [checkpoint] record vào log, force-write log lên đĩa.
  4. Tiếp tục lại các giao dịch đang thực hiện.

FUZZY CHECKPOINT

- Quá trình ghi buffer lên đĩa (do yêu cầu của bước 1) có thể làm chậm quá trình xử lý giao dịch.

- Để hạn chế điều này, hệ thống có thể tiếp tục xử lý giao dịch sau khi [checkpoint] record được ghi vào log mà không cần đợi bước 2 hoàn tất.

FUZZY CHECKPOINT

- Tuy nhiên trước khi bước 2 hoàn tất, [checkpoint] hợp lệ vẫn phải là [checkpoint] cũ.
- Để làm điều này, hệ thống duy trì một con trỏ chỉ đến [checkpoint] hợp lệ.
- Trước khi bước 2 hoàn tất, con trỏ vẫn chỉ đến [checkpoint] cũ trong log.
- Cho đến khi bước 2 hoàn tất, con trỏ được cập nhật đến [checkpoint] mới trong log.

FUZZY CHECKPOINT

- Nếu một giao dịch bị hủy bỏ sau khi đã có sự cập nhật dữ liệu, giao dịch đó cần được quay lui (rollback).
- Nếu có giá trị nào được thay đổi bởi giao dịch và ghi vào CSDL, nó cần được phục hồi lại giá trị trước đó (BFIM).
- Undo-type log entries được dùng cho việc phục hồi lại giá trị cũ của dữ liệu cần rollback.

FUZZY CHECKPOINT

- Nếu một giao dịch T bị rollback, tất cả giao dịch T' đã đọc giá trị nào đó được ghi bởi T cũng phải được rollback…(Cascading rollback)
- Cascading rollback rất phức tạp và tốn thời gian do đó hầu hết cơ chế phục hồi được thiết kế đảm bảo việc cascading rollback không xảy ra.

Ví dụ

(a)

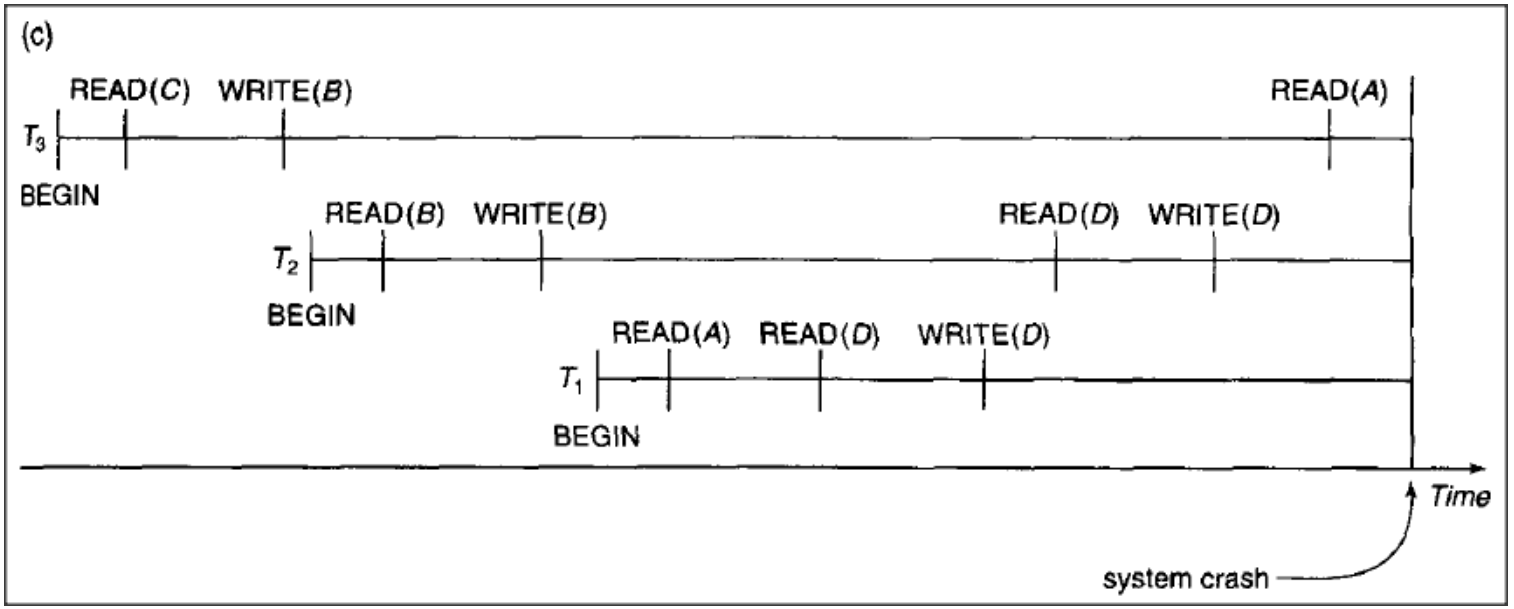
T_1	T_2	T_3
read_item(A)	read_item(B)	read_item(C)
read_item(D)	write_item(B)	write_item(B)
write_item(D)	read_item(D)	read_item(A)
	write_item(D)	write_item(A)

(b)

		A	B	C	D
		30	15	40	20
	[start_transaction, T_3]				
	[read_item, T_3, C]				
*	[write_item, T_3, B, 15, 12]		12		
	[start_transaction, T_2]				
	[read_item, T_2, B]				
**	[write_item, T_2, B, 12, 18]		18		
	[start_transaction, T_1]				
	[read_item, T_1, A]				
	[read_item, T_1, D]				
	[write_item, T_1, D, 20, 25]				25
	[read_item, T_2, D]				
**	[write_item, T_2, D, 25, 26]				26
	[read_item, T_3, A]				
	← system crash				

Ví dụ

- Chỉ các thao tác Write cần UNDO.
- Thao tác Read trong log chỉ cần cho việc xác định cascading rollback.



Ví dụ

- Trên thực tế, cascading rollback không bao giờ xảy ra vì các phương pháp phục hồi thực tế luôn phải đảm bảo cascadeless hoặc strict schedule.
- Do đó, không cần thiết lưu các thao tác Read trong log.

Kĩ thuật phục hồi dựa trên Deferred Update

- Ý tưởng chính của kĩ thuật Deferred Update là trì hoãn tất cả các cập nhật lên CSDL cho đến khi giao dịch hoàn tất (no-steal).
- Trong quá trình xử lý, các thay đổi được lưu trong log và trong cache.
- Sau khi giao dịch commit và log được force-write vào đĩa, các thay đổi mới được ghi vào CSDL trong đĩa.

Kĩ thuật phục hồi dựa trên Deferred Update

- Nếu một giao dịch bị lỗi trước khi commit, không cần phải UNDO bất cứ thao tác nào cả (do giao dịch không ảnh hưởng đến dữ liệu trên đĩa).

- Có thể sử dụng cho hệ thống có giao dịch ngắn và mỗi giao dịch thay đổi ít dữ liệu.

Kĩ thuật phục hồi dựa trên Deferred Update

- Mặc dù cách này làm đơn giản hóa qui trình phục hồi, tuy nhiên nó ít được sử dụng trong thực tế.
- Nguyên nhân: tiềm tàng khả năng hết bộ nhớ đệm do các thay đổi của giao dịch phải được lưu trong cache cho đến khi hoàn tất.

Typical deferred update protocol

- Một giao dịch không thể thay đổi CSDL trên đĩa cho đến khi nó hoàn tất.
- Một giao dịch không hoàn tất được trừ khi tất cả thay đổi được lưu vào log và log được force-write lên đĩa (Write Ahead Logging).
- Còn được gọi là Giải thuật phục hồi NO-UNDO/REDO.

TYPICAL DEFERRED UPDATE PROTOCOL

- REDO cần thiết trong trường hợp hệ thống lỗi sau khi giao dịch commit nhưng trước khi thay đổi được ghi lên đĩa hoàn tất.
- Phương pháp phục hồi liên quan chặt chẽ với phương pháp xử lý song hành:
  - Single-user environment: không có xử lý song hành.
  - Multi-user environment: thông thường, cấp độ xử lý song hành càng cao thì cơ chế phục hồi càng phức tạp, mất thời gian.

GIẢI THUẬT RDU_S

- Recovery using Deferred Update in a Single-user environment (RDU_S) algorithm.
- Sử dụng 2 danh sách giao dịch: giao dịch đã commit từ thời điểm checkpoint trước và giao dịch đang thực hiện (tối đa 1 giao dịch trong danh sách này).
- Thực hiện thao tác REDO cho tất cả thao tác Write của các giao dịch đã commit dựa trên log theo **thứ tự thực thi trong log**.
- Khởi động lại giao dịch đang thực hiện.

Thủ tục REDO

- Thủ tục REDO: Thực hiện lại các thao tác Write bằng cách đọc các dòng trong log [write, T, X, new_value] và gán giá trị X trong CSDL bằng giá trị new_value trong log (AFIM).
- Thao tác REDO phải có tính lũy đẳng – idempotent (việc thực hiện nhiều lần cũng như 1 lần).

GIẢI THUẬT RDU_S

- Trên thực tế, cả quy trình phục hồi cũng phải idempotent: Hệ thống lỗi trong khi đang phục hồi, lần phục hồi tiếp có thể REDO một số thao tác Write mà đã được REDO bởi lần phục hồi lỗi trước đó.
- → Kết quả phục hồi phải giống nhau.

Ghi chú

- Giao dịch duy nhất trong danh sách giao dịch đang thực hiện không ảnh hưởng đến CSDL (deferred update).
- Do đó nó được bỏ qua trong quá trình phục hồi.
- Tuy nhiên nó cần được khởi động lại tự động hoặc do người dùng.

GIẢI THUẬT RDU_M

- Xét hệ thống sử dụng strict two-phase locking và [checkpoint] trong log.
- Sử dụng 2 danh sách giao dịch:
  - Các giao dịch đã commit từ thời điểm checkpoint trước.
  - Các giao dịch đang thực hiện.
- REDO tất cả thao tác Write của các giao dịch đã commit theo thứ tự được ghi trong log.
- Hủy bỏ các giao dịch đang thực hiện và khởi tạo lại.

Ví dụ

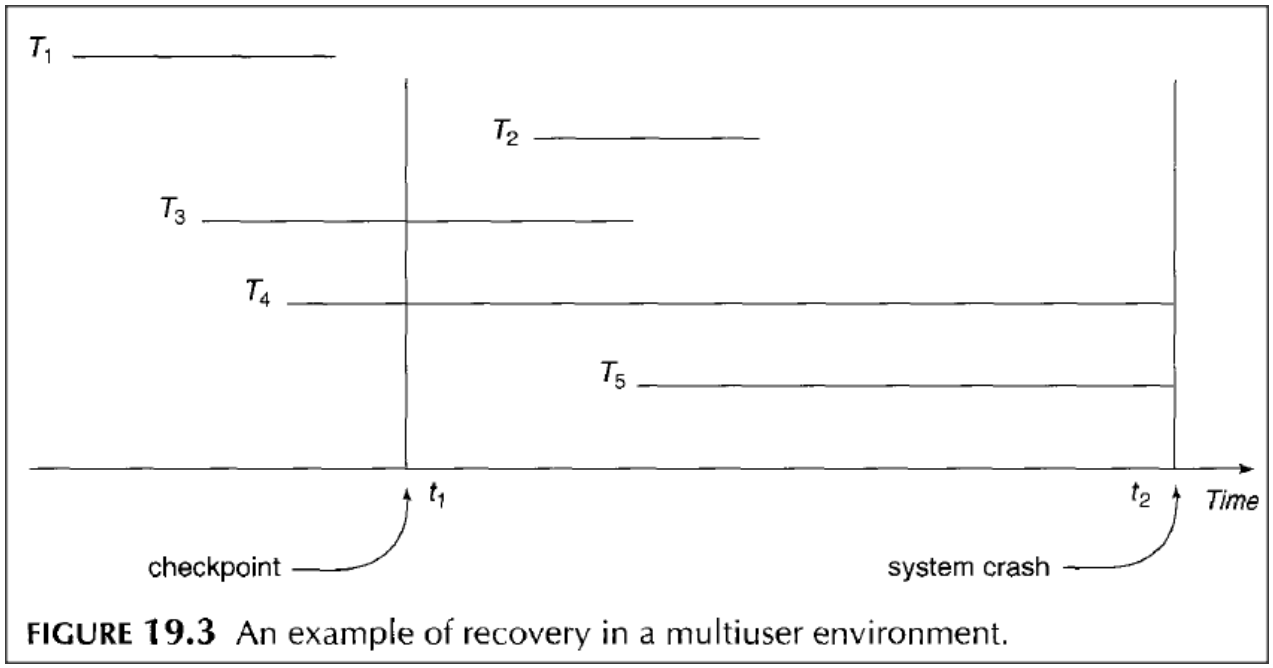


FIGURE 19.3 An example of recovery in a multiuser environment.

- Không cần REDO T1.
- REDO các thao tác Write của T2 và T3.
- Hủy bỏ và gọi lại T4 và T5.

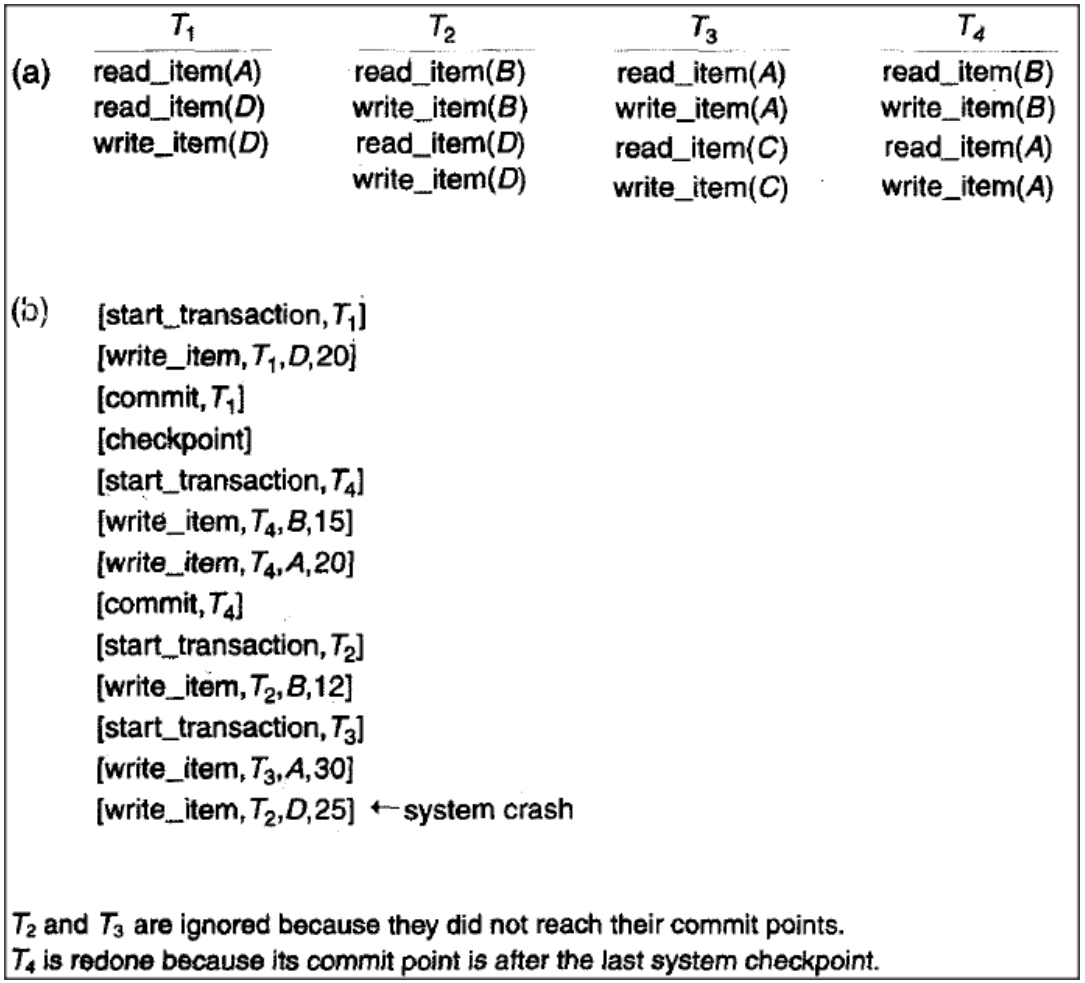
CẢI TIẾN RDU

- Nếu dữ liệu X được cập nhật (thể hiện trong log) nhiều hơn 1 lần bởi các giao dịch đã commit, ta chỉ cần REDO giá trị cuối cùng của X trong quá trình phục hồi.
- Khi đó ta có thể bắt đầu từ cuối log, nếu dữ liệu được REDO hoàn tất thì thêm vào danh sách đã REDO.
- Trước khi thực hiện REDO, kiểm tra danh sách trước.

CẢI TIẾN RDU

- Nếu 1 giao dịch bị hủy bỏ, chỉ cần thực hiện lại nó mà không ảnh hưởng gì đến CSDL.
- Hạn chế:
  - Giới hạn các giao dịch đồng thời vì khóa trên các hạng mục dữ liệu chỉ được mở khi giao dịch hoàn tất.
  - Cần không gian buffer lớn để lưu các dữ liệu có chỉnh sửa cho đến khi giao dịch được commit.
- Ưu điểm: giao dịch không bao giờ cần UNDO:
  - Không thay đổi dữ liệu cho đến khi giao dịch commit.
  - Không bao giờ đọc dữ liệu của giao dịch chưa commit bởi khóa.

Ví dụ



(a)

T_1	T_2	T_3	T_4
read_item(A)	read_item(B)	read_item(A)	read_item(B)
read_item(D)	write_item(B)	write_item(A)	write_item(B)
write_item(D)	read_item(D)	read_item(C)	read_item(A)
	write_item(D)	write_item(C)	write_item(A)

(b)

[start_transaction, T_1]
[write_item, T_1, D, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[write_item, T_4, A, 20]
[commit, T_4]
[start_transaction, T_2]
[write_item, T_2, B, 12]
[start_transaction, T_3]
[write_item, T_3, A, 30]
[write_item, T_2, D, 25] ← system crash

T_2 and T_3 are ignored because they did not reach their commit points.
T_4 is redone because its commit point is after the last system checkpoint.

GIAO DỊCH KHÔNG ẢNH HƯỞNG CSDL

- Một số giao dịch không ảnh hưởng đến CSDL như xuất message, báo cáo.
- Nếu giao dịch lỗi trước khi hoàn tất → report lỗi → cảnh báo user.
- Do đó các report này cũng phải được tạo sau khi các giao dịch hoàn tất.

Kĩ thuật phục hồi dựa trên Immediate Update

- Khi giao dịch thay đổi giá trị, CSDL có thể được cập nhật mà không cần đợi cho giao dịch đó hoàn tất.

- Việc cập nhật thay đổi lên đĩa trước hết vẫn cần được ghi lại trong log trên đĩa – cơ chế WAL để có thể phục hồi trong trường hợp xảy ra sự cố.

Kĩ thuật phục hồi dựa trên Immediate Update

- Khi giao dịch thất bại, ta cần phải khôi phục (undo) tác động của các thao tác cập nhật ảnh hưởng đến CSDL của giao dịch đó.
- Thực hiện: rollback giao dịch, UNDO tác động của các thao tác Write.
- Về lý thuyết, chia làm 2 loại:
  - Tất cả cập nhật được ghi trên đĩa trước khi giao dịch commit: UNDO/NO-REDO.
  - Tổng quát: Cho phép giao dịch commit trước khi tất cả thay đổi được ghi vào đĩa: UNDO/REDO.

GIẢI THUẬT RIU_S

- Recovery using Immediate Update in a Single-user environment (RIU_S).
- Khi sự cố xảy ra, giao dịch đang thực hiện có thể đã cập nhật một số dữ liệu trên CSDL, những thay đổi đó cần phải được UNDO.
- Giải thuật RIU_S sử dụng thủ tục REDO ở phần trước và thủ tục UNDO.

THỦ TỤC UNDO

- Phục hồi (UNDO) một thao tác Write dựa trên việc xem xét các entry trong log [write, T, X, old_value, new_value] và đổi giá trị X trong CSDL thành old_value (BFIM).
- Việc phục hồi phải thực hiện theo thứ tự ***ngược lại*** với thứ tự của thao tác được ghi trong log.

GIẢI THUẬT RIU_S

- Sử dụng 2 danh sách giao dịch:
  - Danh sách các giao dịch đã được commit từ checkpoint trước.
  - Danh sách giao dịch đang thực hiện (max 1)
- UNDO tất cả thao tác Write của giao dịch đang thực hiện theo thủ tục UNDO.
- REDO các thao tác Write của các giao dịch đã commit dựa trên log theo thứ tự được ghi trong log theo thủ tục REDO.

GIẢI THUẬT RIU_M

- Xử lý song hành: Sử dụng strict 2PL → xảy ra deadlock → hủy bỏ và UNDO.
- Sử dụng 2 danh sách giao dịch:
  - Danh sách các giao dịch đã được commit từ checkpoint trước.
  - Danh sách các giao dịch đang thực hiện.
- UNDO tất cả thao tác Write của các giao dịch đang thực hiện theo thủ tục UNDO theo thứ tự ngược với thứ tự ghi trong log.
- REDO các thao tác Write của các giao dịch đã commit dựa trên log theo thứ tự được ghi trong log theo thủ tục REDO.

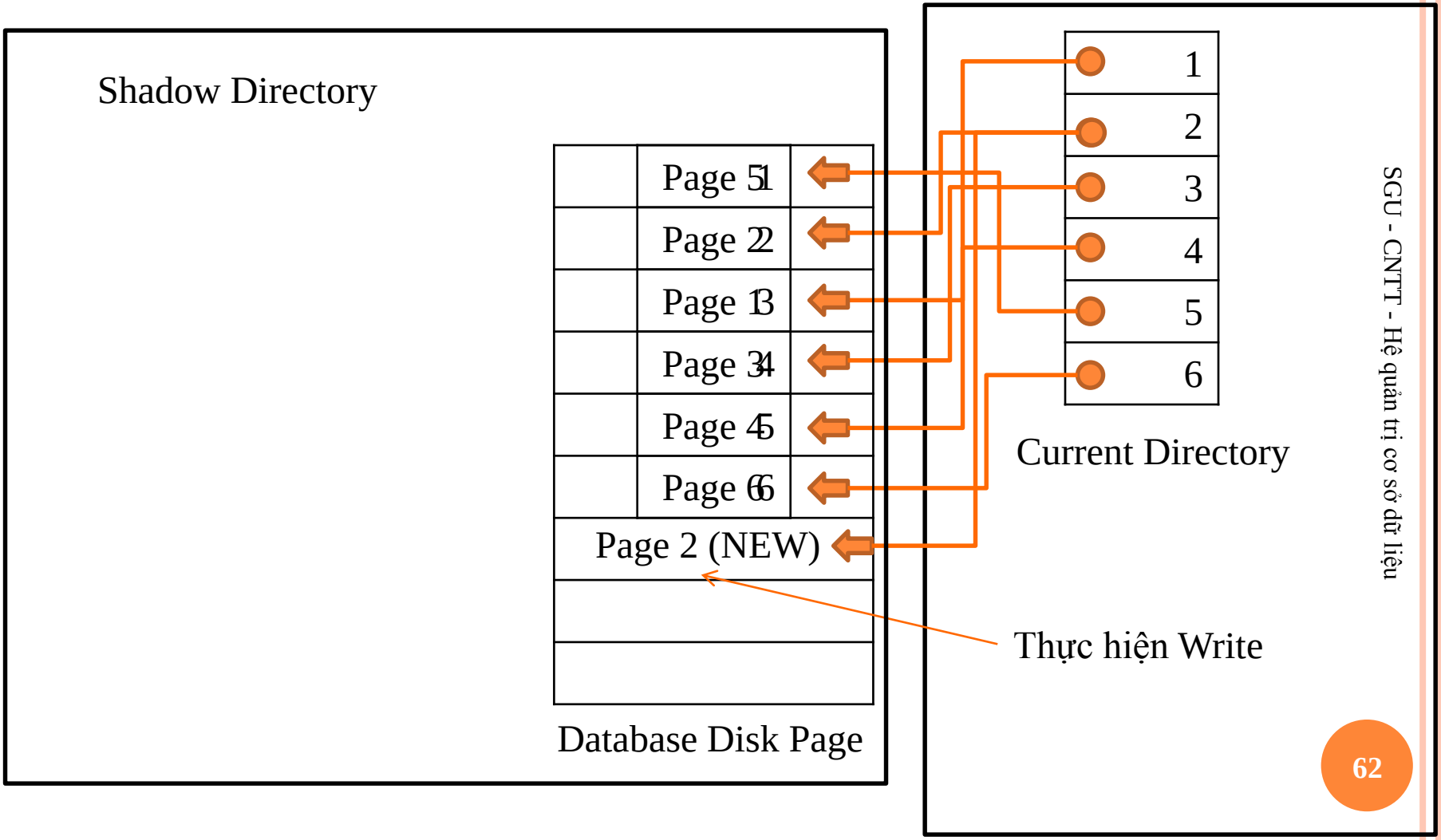
Kĩ thuật phục hồi dựa trên Shadow Paging

- Không cần đến log trong Single-user environment, trong multi-user environment, cần đến log để xử lý song hành.
- Xem CSDL được tạo thành bởi tập hợp disk page kích thước cố định.
- Dùng Directory lưu địa chỉ các page của Database trên đĩa và được quản lý trong bộ nhớ chính nếu nó không quá lớn.
- Khi bắt đầu 1 giao dịch, directory hiện tại được lưu lại 1 bản (shadow directory) và lưu vào đĩa.

Kĩ thuật phục hồi dựa trên Shadow Paging

- Trong suốt quá trình thực hiện các giao dịch, chỉ xử lý trên current directory.
- Khi thực hiện write, một page mới được tạo chứ không ghi đè lên page cũ.
- Địa chỉ tương ứng của page trong current directory được thay đổi chỉ đến page mới được tạo.

MINH HỌA SHADOW PAGE



KĨ THUẬT PHỤC HỒI DỰA TRÊN SHADOW PAGING

- Phục hồi đơn giản là thực hiện:
  - Xóa các page được chỉnh sửa trong disk (Page 2 New).
  - Hủy bỏ current directory.
- Trạng thái dữ liệu trước khi xảy ra sự cố được tái lập thông qua shadow directory.
- Khi commit giao dịch: hủy bỏ các shadow directory.
- Không cần undo hay redo dữ liệu ➔ **NO-UNDO/NO-REDO**

Kĩ thuật phục hồi dựa trên Shadow Paging

- Với multi-user environment, log và checkpoint được sử dụng.
- Hạn chế:
  - Khó kiểm soát vị trí các page trong đĩa gần nhau.
  - Vấn đề khi ghi directory xuống đĩa trong trường hợp directory quá lớn.
  - Vấn đề về garbage collection khi giao dịch commit.
  - Phải đảm bảo tính nguyên tử trong quá trình xử lý current và shadow directory.

MỘT SỐ VẤN ĐỀ KHÁC

- Giải thuật phục hồi ARIES.
- Vấn đề phục hồi trong hệ thống multiDB.
- Sao lưu và phục hồi trong các sự cố lớn.

END

Tham khảo chương 23

Fundamentals of Database System, 6th Edtion