Trường Đại Học Bách Khoa Tp.HCM
Hệ Đào Tạo Từ Xa
Khoa Khoa Học và Kỹ Thuật Máy Tính

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Chương 7. Các kỹ thuật khôi phục dữ liệu (data recovery)

Bùi Hoài Thắng

Nội dung

* Các khái niệm khôi phục dữ liệu
* Các kỹ thuật khôi phục dữ liệu dựa vào sự cập nhật trì hoãn
* Các kỹ thuật khôi phục dữ liệu dựa vào sự cập nhật tức thời
* Kỹ thuật phân trang đi kèm
* Giải thuật khôi phục ARIES
* Sao chép dự phòng và khôi phục dữ liệu từ những sự cố rủi ro
* Thể hiện của các cơ chế khôi phục dữ liệu trên SQL-Server

BÀI 1 – CƠ BẢN

Khái niệm

Mục tiêu của việc khôi phục CSDL

* Mục tiêu của khôi phục CSDL (Database Recovery)
  - Đưa CSDL về trạng thái nhất quán cuối cùng ngay trước khi xảy ra sự hỏng hóc (CSDL)
  - Đảm bảo các thuộc tính ACID của giao tác (Atomicity, Consistency, Isolation và Durability)

* Ví dụ:
  - Nếu hệ thống bị sụp trước khi một giao tác chuyển tiền hoàn tất việc thực thi, có thể các tài khoản liên quan đều có giá trị sai.
  - CSDL cần phải được phục hồi về trạng thái trước khi giao tác hiệu chỉnh các tài khoản (nghĩa là xem như việc chuyển tiền không thành công)

Tóm tắt các sự cố hỏng hóc giao tác (chương 5)

(A) Hỏng luận lý

* 1. Lỗi gây sụp hệ thống
* 2. Lỗi giao tác
* 3. Lỗi cục bộ hoặc các điều kiện ngoại lệ
* 4. Do điều khiển tương tranh

(B) Hỏng vật lý

* 5. Hỏng hóc đĩa
* 6. Tai hoạ: như nguồn điện, cháy nổ, trộm, phá hoại, mất tập tin, …

Chiến lược khôi phục

* Nếu hỏng vật lý:
  - Phục hồi dữ liệu từ các bản sao dự phòng (backup) tốt nhất
  - Tái tạo CSDL đến trạng thái nhất quán gần đây nhất có thể có bằng các tác vụ tái thực thi (redo) đối với các giao tác đã commit dựa vào sổ ghi sao dự phòng (backup log)
  - (Tự đọc và xem phần liên quan đến SQL-Server)

Chiến lược khôi phục (tt.)

* Nếu hỏng luận lý:
  - Đảo ngược các thay đổi gây mất nhất quán CSDL bằng các tác vụ tháo gỡ (undo)
  - Tái thực thi (redo) các tác vụ của các giao tác đã commit để khôi phục trạng thái nhất quán của CSDL nếu cần
  - Chỉ cần dùng sổ ghi hệ thống/giao tác (system/transaction log)
  - Tuỳ thuộc vào các chiến lược cập nhật dữ liệu và dùng vùng đệm
  - (Trong các phần kế, các kiến thức là dành cho trường hợp này)

Sổ ghi

- Lưu các giá trị trước (BFIM - BeFore Image) và sau khi cập nhật (AFIM – AFter Image) và các thông tin khác trong một tập tin tuần tự gọi là Sổ ghi giao tác (Transaction log)
- Ví dụ:
  - Back P và Next P chỉ đến mục sổ ghi trước hoặc kế của một giao tác

T ID	Back P	Next P	Operation	Data item	BFIM	AFIM
T1	0	1	Begin			
T1	1	4	Write	X	X = 100	X = 200
T2	0	8	Begin			
T1	2	5	W	Y	Y = 50	Y = 100
T1	4	7	R	M	M = 200	M = 200
T3	0	9	R	N	N = 400	N = 400
T1	5	nil	End			

Dùng vùng đệm (Data cache/buffer)

- Dùng vùng đệm dữ liệu để tăng hiệu năng sử dụng hệ thống
  - Các mục liệu cần được cập nhật sẽ được lưu vào trong vùng đệm trước
  - Sau khi cập nhật xong, dữ liệu trong vùng đệm sẽ được đưa xuống đĩa
- Mỗi khối/trang trong vùng đệm (block/page) có 2 bit:
  - Pin-Unpin: nếu là pinned (bằng 1) sẽ không cho phép hệ điều hành đưa dữ liệu xuống đĩa
  - Modified hay dirty: cho biết khối/trang vùng đệm có bị thay đổi hay chưa

Cập nhật dữ liệu

* Về mặt thời gian:
  - Cập nhật tức thời (**Immediate Update**): ngay khi mục liệu đã được thay đổi trong vùng đệm, dữ liệu sẽ được ghi nhận vào đĩa
  - Cập nhật trì hoãn (**Deferred Update**): tất cả các mục liệu trong vùng đệm sẽ được ghi nhận vào đĩa sau khi một giao tác kết thúc hoặc sau một số giao tác kết thúc
* Về mặt không gian:
  - Cập nhật bóng hình (**Shadow update**): Một phiên bản của mục liệu được tạo ra tại vị trí khác trên đĩa để lưu trữ giá trị mới
  - Cập nhật tại chỗ (**In-place update**): Mục liệu trên đĩa sẽ được ghi đè bằng dữ liệu trên vùng đệm

Kỹ thuật quay ngược (Rollback) và quay xuôi (Roll-forward)

- Để đảm bảo tính đơn thể (atomicity) một giao tác:
  - Các giao tác chưa hoàn thành phải bị huỷ bỏ và các tác vụ cập nhật dữ liệu của nó phải bị tháo gỡ (undo)
  - Các giao tác đã hoàn thành phải được đảm bảo là các thay đổi dữ liệu của nó phải được tái thực thi (redo) để lưu trữ vĩnh viễn trong CSDL
- Quay ngược giao tác (transaction rollback):
  - Undo: phục hồi tất cả các BFIM xuống đĩa (tức gỡ bỏ các AFIM)
- Quay xuôi giao tác (transaction roll-forward):
  - Redo: phục hồi tất cả các AFIM xuống đĩa

Kỹ thuật quay ngược và quay xuôi (tt.)

* Giả sử hệ thống có ba giao tác T_1, T_2 và T_3
* Giá trị ban đầu của A=30, B=15, C=40, D=20 là

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)

Kỹ thuật quay ngược và quay xuôi (tt.)

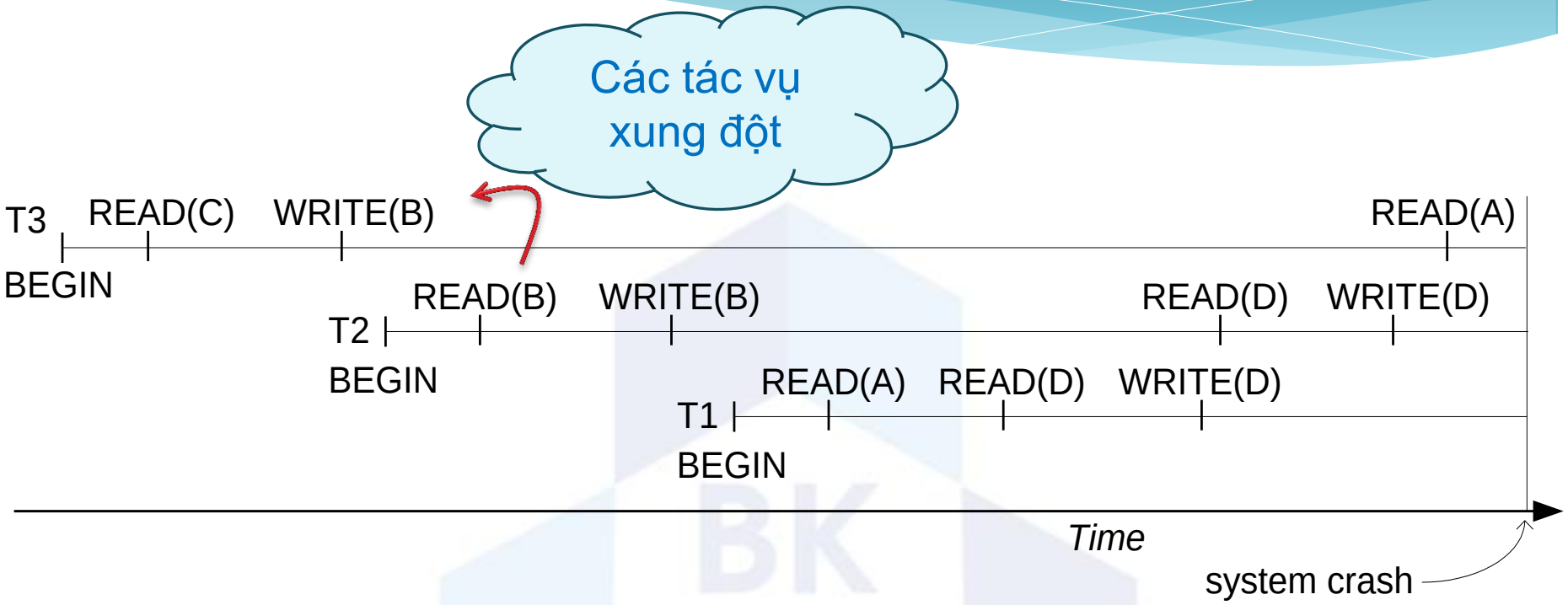
		A	B	C	D
		30	15	40	20
	[start_transaction, T_3]				
	[read_item, T_3, C]				
*	[write_item, T_3, B, 15, 12]		12		
	[start_transaction,T_2]				
	[read_item, T_2, B]				
**	[write_item, T_2, B, 12, 18]		18		
	[start_transaction,T_1]				
	[read_item, T_1, A]				
	[read_item, T_1, D]				
	[write_item, T_1,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

* T_3 sẽ bị quay ngược vì nó chưa đến điểm commit

** T_2 sẽ bị quay ngược theo vì nó đọc mục liệu B ghi bởi T_3.

Hình 7.1

Kỹ thuật quay ngược và quay xuôi (tt.)



Mô tả vấn đề quay ngược dắt dây

Hình 7.2

Giao thức ghi sổ trước (Write-Ahead Logging - WAL)

* Giao thức ghi sổ trước (write-ahead logging protocol):
  - Để tháo gỡ (Undo) được:
    - Trước khi dữ liệu được cập nhật vào trong CSDL (trong đĩa), giá trị hiện tại của dữ liệu (BFIM) cần phải được ghi vào sổ ghi và sổ ghi cần phải được lưu trữ vào một vùng lưu trữ ổn định (ví dụ như đĩa chứa sổ ghi)
  - Để tái thực thi (Redo) được:
    - Trước khi giao tác commit, tất cả các giá trị mới cập nhật (AFIM) của dữ liệu cần phải được ghi vào sổ ghi và sổ ghi cần phải được lưu trữ vào một vùng lưu trữ ổn định

Steal/No-Steal và Force/No-Force

* Các thuật ngữ dùng diễn đạt thời điểm vùng đệm (cache) sẽ được ghi vào đĩa:
  - Steal: có thể ghi vùng đệm (cache) của các mục liệu đã ghi bởi một giao tác vào đĩa trước khi giao tác này commit
  - No-Steal: không được ghi vùng đệm (của giao tác) vào đĩa trước khi giao tác commit
  - Force: buộc ghi vùng đệm vào đĩa ngay lập tức khi giao tác commit
  - No-Force: trì hoãn ghi vùng đệm cho đến khi một số giao tác commit

Steal/No-Steal và Force/No-Force (tt.)

* Ưu điểm:
  - Steal: tránh trường hợp phải duy trì vùng đệm quá lớn của một giao tác chưa commit
  - No-Steal: không cần undo
  - Force: không cần redo
  - No-Force: giảm chi phí ghi đĩa khi vùng đệm đã commit có thể được dùng bởi các giao tác khác và tăng hiệu năng hệ thống khi việc ghi có thể diễn ra vào thời gian thấp điểm của hệ thống
* Nhược điểm:
  - ???

Steal/No-Steal và Force/No-Force (tt.)

* Như vậy, có 4 cách khôi phục tuỳ theo cách lưu vùng đệm:
  - Steal/No-Force (Undo/Redo)
  - Steal/Force (Undo/No-redo)
  - No-Steal/No-Force (Redo/No-undo)
  - No-Steal/Force (No-undo/No-redo)

Kỹ thuật dùng điểm kiểm tra (Checkpointing)

* Tại thời điểm này (checkpoint), hệ thống sẽ lưu toàn bộ vùng đệm xuống đĩa để giảm thiểu tác vụ khôi phục
* Các giao tác commit trước điểm kiểm tra sẽ không cần phải tái thực thi nếu hệ thống bị hỏng sau điểm kiểm tra
* Qui trình:
  1. Tạm ngưng toàn bộ các giao tác
  2. Buộc ghi các vùng đệm đã hiệu chỉnh xuống đĩa
  3. Ghi một mục [checkpoint] vào trong sổ ghi, lưu sổ ghi vào đĩa
  4. Hồi phục lại các giao tác (đang bị tạm ngưng)

Kỹ thuật dùng điểm kiểm tra mờ (Fuzzy checkpointing)

* Do việc ghi vùng đệm xuống đĩa tốn nhiều thời gian, nên các giao tác sẽ bị tạm ngưng một thời gian trong quá trình checkpoint
* Kỹ thuật dùng điểm kiểm tra mờ (fuzzy checkpointing) sẽ giảm thiểu nhược điểm này:
  - Hệ thống có thể hồi phục các giao tác sau khi mục [checkpoint] được ghi vào sổ ghi mà không cần đợi bước 2 kết thúc
  - Cho đến trước khi bước 2 kết thúc, điểm [checkpoint] cũ vẫn phải còn đúng. Chỉ sau khi bước 2 kết thúc thì điểm [checkpoint] này mới được xem là điểm checkpoint đúng cuối cùng

Kỹ thuật dùng điểm kiểm tra mờ (tt.)

* Như vậy từ lúc bắt đầu checkpoint đến lúc kết thúc checkpoint một số vùng đệm đã ghi xuống đĩa vẫn tiếp tục được cập nhật
* Nếu tiến hành REDO trên sổ ghi từ đầu checkpoint đến cuối checkpoint thì sẽ tương đương với tác vụ checkpoint bình thường (còn gọi là full checkpoint)

Kỹ thuật dùng điểm kiểm tra mờ (tt.)

* Có nhiều phiên bản của kỹ thuật này:
  - Định nghĩa các tác vụ mờ (fuzzy operation):
    - DB2
  - Chia việc ghi đĩa thành từng mẻ nhỏ (small batch):
    - InnoDB MySQL

Đọc thêm:

https://www-304.ibm.com/support/docview.wss?uid=swg21140289

http://dev.mysql.com/doc/mysql-backup-excerpt/5.0/en/innodb-checkpoints.html

http://www.xaprb.com/blog/2011/01/29/how-innodb-performs-a-checkpoint/

Khôi phục dựa trên chiến lược cập nhật trì hoãn

Khôi phục dựa trên cập nhật trì hoãn (Deferred Update)

* Deferred Update = No-Undo/Redo:
  - Các giao tác cập nhật dữ liệu vào vùng đệm và ghi nhận các cập nhật vào sổ ghi
  - Tại thời điểm commit, dùng WAL, các cập nhật trong vùng đệm sẽ được ghi xuống đĩa
    - Không cần ghi liền mà có thể dùng checkpoint
* Nếu hệ thống trục trặc, sổ ghi sẽ được dùng để redo tất cả các giao tác bị ảnh hưởng (đã commit nhưng chưa kịp ghi vào đĩa)
* Không cần undo vì không có cập nhật nào được ghi vào đĩa nếu giao tác chưa commit
* Mục sổ ghi chỉ cần lưu các AFIM

Cập nhật trì hoãn trong môi trường đơn người dùng

- (Deferred Update in a Single-user environment)
  - Không có các chia sẻ dữ liệu tương tranh
- Thủ tục RDU_S:
  - Dùng hai danh sách các giao tác:
    - Các giao tác đã commit (committed transaction) kể từ điểm kiểm tra (checkpoint) cuối
    - Các giao tác đang hoạt động (active transaction):
      - Chỉ có tối đa một giao tác đang hoạt động
  - Áp dụng tác vụ REDO cho tất cả các tác vụ write_item của các giao tác đã commit từ sổ ghi theo thứ tự xuất hiện (thứ tự ghi) của chúng trong sổ ghi
  - Tái khởi tạo các giao tác đang hoạt động

Cập nhật trì hoãn trong môi trường đơn người dùng (tt.)

* Thủ tục REDO(WRITE_OP):
  - Tái thực thi tác vụ write_item bằng cách khám xét mục sổ ghi [write_item, T, X, new_value] và gán giá trị của X trong CSDL (trên đĩa) cho giá trị new_value (tức là AFIM)
* Tác vụ REDO cần phải "idempotent":
  - Áp dụng tác vụ redo trên một mục liệu nhiều lần có kết quả giống như áp dụng một lần
* Do quá trình khôi phục có thể bị trục trặc và cần phải khởi động (quá trình khôi phục) lại. Việc áp dụng tác vụ redo có thể bị tiến hành nhiều lần và có kết quả như một lần. Nghĩa là quá trình khôi phục diễn ra một lần thành công hay lặp lại (do bị sự cố) có kết quả như nhau.
  - Xem như quá trình khôi phục chỉ diễn ra một lần !

Cập nhật trì hoãn trong môi trường đơn người dùng (tt.)

Xét hai giao tác T_1 và T_2 và một sổ ghi sau:

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

--- log ---

[start_transaction, T_1]
[write_item, T_1, D, 20]
[commit T_1]
[start_transaction, T_2]
[write_item, T_2, B, 10]
[write_item, T_2, D, 25] ← system crash

Tác vụ [write_item, T_1, D, 20] của T_1 sẽ được tái thực thi
Các mục sổ ghi của T_2 sẽ bị bỏ qua vì T_2 chưa commit

Hình 7.3

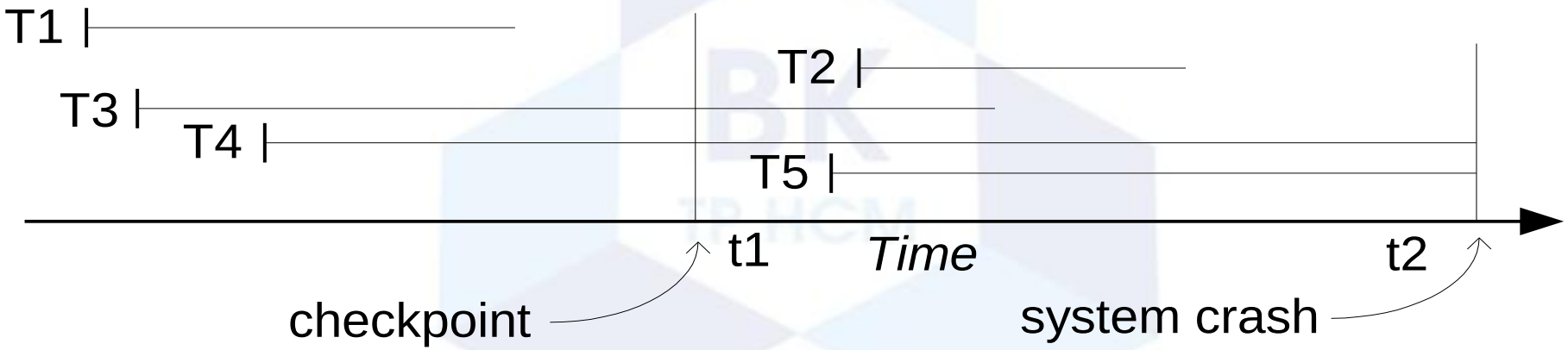
Cập nhật trì hoãn trong môi trường tương tranh

* (Deferred Update with concurrent users)
* Cần phải có cơ chế điều khiển tương tranh để đảm bảo tính đơn lập của các giao tác
* Khi khôi phục, cần phải quét thêm một số mục sổ ghi trước điểm kiểm tra cuối để tái thực thi

Recovery in a concurrent users environment.

Cập nhật trì hoãn trong môi trường tương tranh (tt.)

* Trong ví dụ này, giao tác T2 và T3 commit sau checkpoint và trước khi hệ thống bị sụp
  - T2 và T3 cần phải được tái thực thi
  - Do T3 bắt đầu trước checkpoint nên cần phải quét sổ ghi trước checkpoint (từ lúc bắt đầu T3) để tái thực thi T3



Hình 7.4

Cập nhật trì hoãn trong môi trường tương tranh (tt.)

* Cần duy trì hai bảng các giao tác:
  - 1. Active table: Theo dõi các giao tác đang hoạt động
  - 2. Commit table: Theo dõi các giao tác đã commit
* Trong quá trì khôi phục,
  - Tất cả các giao tác trong commit table được tái thực thi
  - Các giao tác trong active table bị bỏ qua
* Chú ý:
  - Tác vụ REDO cần phải là “idempotent”

Cập nhật trì hoãn trong môi trường tương tranh (tt.)

* Thủ tục RDU_M:
  - Áp dụng tác vụ REDO cho tất cả các tác vụ write_item của các giao tác đã commit từ sổ ghi theo thứ tự xuất hiện (thứ tự ghi) của chúng trong sổ ghi
    - Thứ tự ghi trong sổ ghi chứ không phải thứ tự theo từng giao tác
  - Tái khởi tạo các giao tác đang hoạt động

Cập nhật trì hoãn trong môi trường tương tranh (tt.)

--- log ---
[start_transaction, T_1]
[write_item, T_1, D, 20]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[start_transaction T_2]
[commit, T_1]
[write_item, T_4, A, 20]
[commit, T_4]
[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 và T_3 bị bỏ qua vì chưa commit
- T_1 và T_4 được tái thực thi vì nó commit sau checkpoint.

D ← 20
B ← 15
A ← 20

Hình 7.5

Cập nhật trì hoãn trong môi trường tương tranh (tt.)

- Nhận xét:
  - Do một mục liệu có thể được cập nhật nhiều lần bởi các giao tác đã commit sau điểm kiểm tra (checkpoint), trong quá trình khôi phục, chỉ cần tái thực thi lần cập nhật cuối cùng của mục liệu trong sổ ghi là đủ
- Giải thuật khác:
  - Có thể bắt đầu từ cuối sổ ghi
  - Dùng một danh sách các mục liệu đã được tái thực thi (redo)
  - Khi tái thực thi (redo) một mục liệu, kiểm tra trong danh sách, nếu chưa có thì tái thực thi và ghi nhớ lại
  - Hiệu quả hơn quét sổ ghi từ đầu

Khôi phục dựa trên chiến lược cập nhật tức thời

Khôi phục dựa trên cập nhật tức thời (Immediate Update)

* Cập nhật tức thời (Immediate update)
  - Mỗi khi giao tác cập nhật mục liệu, dữ liệu sẽ được cập nhật xuống đĩa ngay mà không cần đợi giao tác commit
  - WAL: sổ ghi phải được cập nhật và ghi vào đĩa trước
* Trong khi khôi phục:
  - Các thay đổi của các giao tác chưa commit phải được tháo gỡ
* Nhận xét:
  - Không cần phải giữ các mục liệu được cập nhật bởi các giao tác chưa commit trên vùng đệm
  - Cần phải tháo gỡ (undo) các giao tác hỏng

Khôi phục dựa trên cập nhật tức thời (tt.)

* Có hai nhóm giải thuật:
  - UNDO/NO-REDO: Nếu hệ thống đảm bảo là tất cả các thay đổi của một giao tác đã được ghi nhận vào CSDL (trên đĩa) trước khi giao tác commit, không cần phải tái thực thi các giao tác này (NO-REDO)
  - UNDO/REDO: Nếu hệ thống cho phép một giao tác được commit trước khi các thay đổi của nó được ghi nhận vào đĩa, cần phải tái thực thi các giao tác này (REDO)

UNDO/NO-REDO

* Chỉ cần ngăn cản một giao tác commit cho đến khi tất cả các cập nhật của nó được ghi xong vào đĩa
* Có khả năng một giao tác đã hoàn tất toàn bộ các tác vụ của mình nhưng bị tháo gỡ (vì chưa commit được)

UNDO/REDO trong môi trường đơn người dùng

* Không có tương tranh
* Chỉ có một giao tác cùng lúc tại cùng một thời điểm
  - Hoặc là đã commit, hoặc là đang hoạt động
* Ghi sổ theo giao thức WAL
* Giải thuật:
  - Tháo gỡ (Undo) một giao tác nếu nó đang trong active table
  - Tái thực thi (Redo) một giao tác nếu nó đang trong commit table

UNDO/REDO trong môi trường tương tranh

* Dùng hai bảng: active table và commit table để lưu các giao tác đang hoạt động và các giao tác đã commit
* Có thể có nhiều giao tác cùng lúc trong các bảng này
* Ghi sổ theo giao thức WAL
* Để tăng hiệu quả của việc khôi phục, checkpoint sẽ được dùng
* Giải thuật tổng quát:
  - Tháo gỡ (Undo) các giao tác trong bảng active table
  - Tái thực thi (Redo) các giao tác trong bảng commit table

UNDO/REDO trong môi trường tương tranh (tt.)

* Thủ tục RIU_M:
  - Áp dụng tác vụ UNDO cho tất cả các tác vụ write_item của các giao tác trong active table từ sổ ghi ngược với thứ tự xuất hiện (thứ tự ghi) của chúng trong sổ ghi
    - Quét sổ ghi từ cuối lên đầu
  - Áp dụng tác vụ REDO cho tất cả các tác vụ write_item của các giao tác trong commit table từ sổ ghi theo thứ tự xuất hiện của chúng trong sổ ghi

UNDO/REDO trong môi trường tương tranh (tt.)

* Để tăng hiệu quả:
  - REDO: có thể quét sổ ghi từ dưới lên và mỗi khi redo một mục liệu, đưa vào danh sách các mục liệu đã redo để không thực hiện tiếp nếu gặp phải yêu cầu redo lần nữa (về phía trên sổ ghi)
  - UNDO: có thể quét sổ ghi từ trên xuống, ngược với cách trên

UNDO/REDO trong môi trường tương tranh (tt.)

--- log ---
[start_transaction, T_1]
[write_item, T_1, D, 12, 20]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 23, 15]
[start_transaction T_2]
[commit, T_1]
[write_item, T_2, B, 15, 12]
[start_transaction, T_3]
[write_item, T_4, A, 30, 20]
[commit, T_4]
[write_item, T_3, A, 20, 30]
[write_item, T_2, D, 20, 25]
[write_item, T_2, B, 12, 17]
← system crash

T_2 và T_3 bị tháo gỡ vì chưa commit

B ← 12
D ← 20
A ← 20
B ← 15

T_1 và T_4 được tái thực thi vì nó commit sau checkpoint.

D ← 20
B ← 15
A ← 20

Hình 7.6

Tóm tắt bài 1

- Các khái niệm khôi phục dữ liệu
  - Khái niệm
  - Sổ ghi – WAL
  - Deferred/Immediate update
  - Shadow/In-place update
  - UNDO và REDO
  - Steal/No-Steal và Force/No-Force
  - Checkpoint
  - Fuzzy checkpoint
- Các kỹ thuật khôi phục dữ liệu dựa vào sự cập nhật trì hoãn
  - Deferred update
  - RDU_S / RDU_M
- Các kỹ thuật khôi phục dữ liệu dựa vào sự cập nhật tức thời
  - Immediate update
  - UNDO/NO-REDO
  - UNDO/REDO