CÁC HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

CHƯƠNG 2

GIAO DỊCH (TRANSACTION)

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

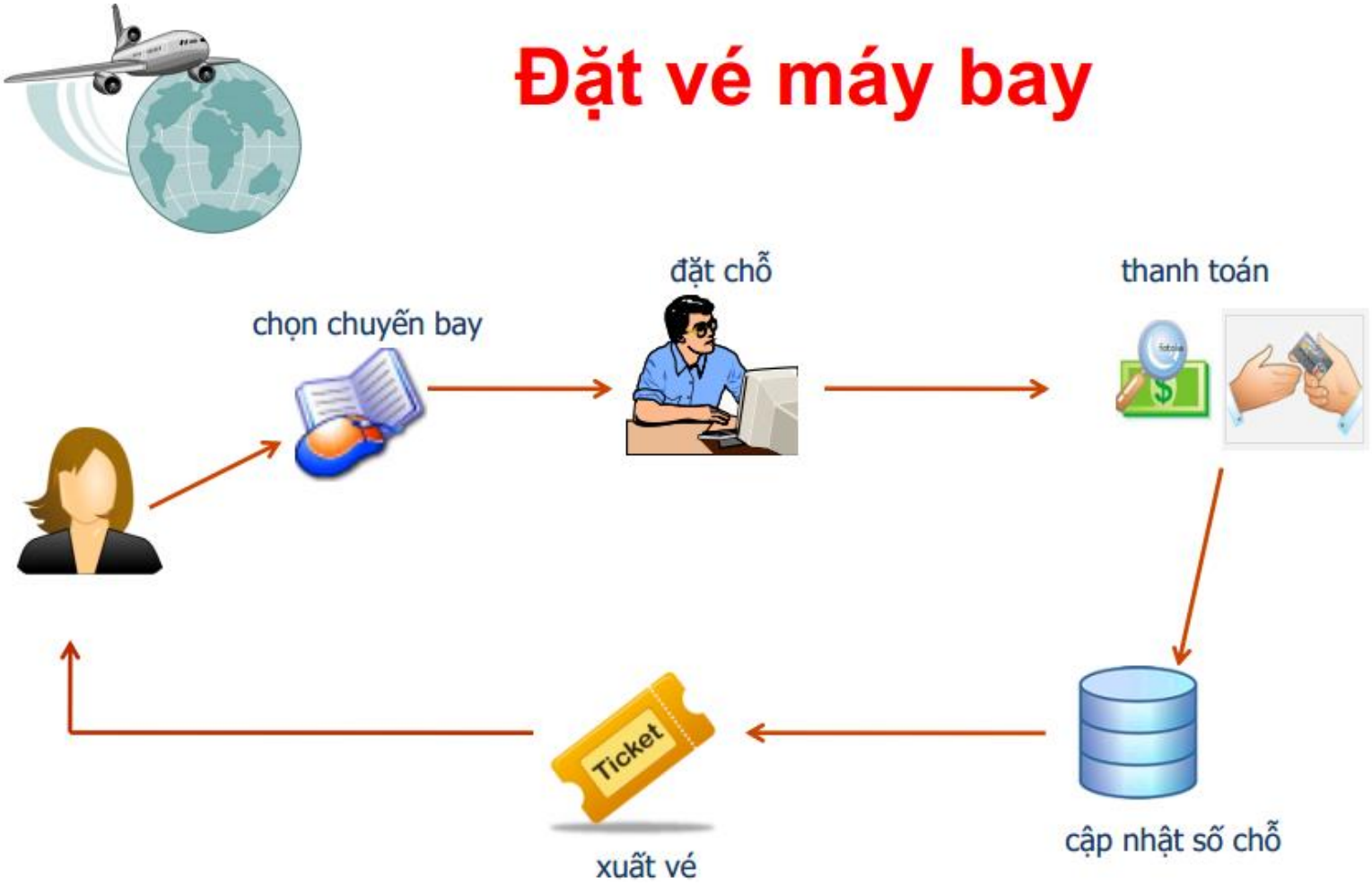
KHOA CÔNG NGHỆ THÔNG TIN

Nội dung

- Giới thiệu về Giao dịch (Transaction).
  - Các tính chất của giao dịch.
  - Các thành phần xử lý giao dịch trong HQT CSDL.
  - Các trạng thái của giao dịch.
- Lịch trình:
  - Khái niệm
  - Lịch trình tuần tự
  - Tính khả tuần tự
  - Khả tuần tự xung đột
  - Một số bài tập
  - Khả tuần tự view
- Giao dịch trong SQL

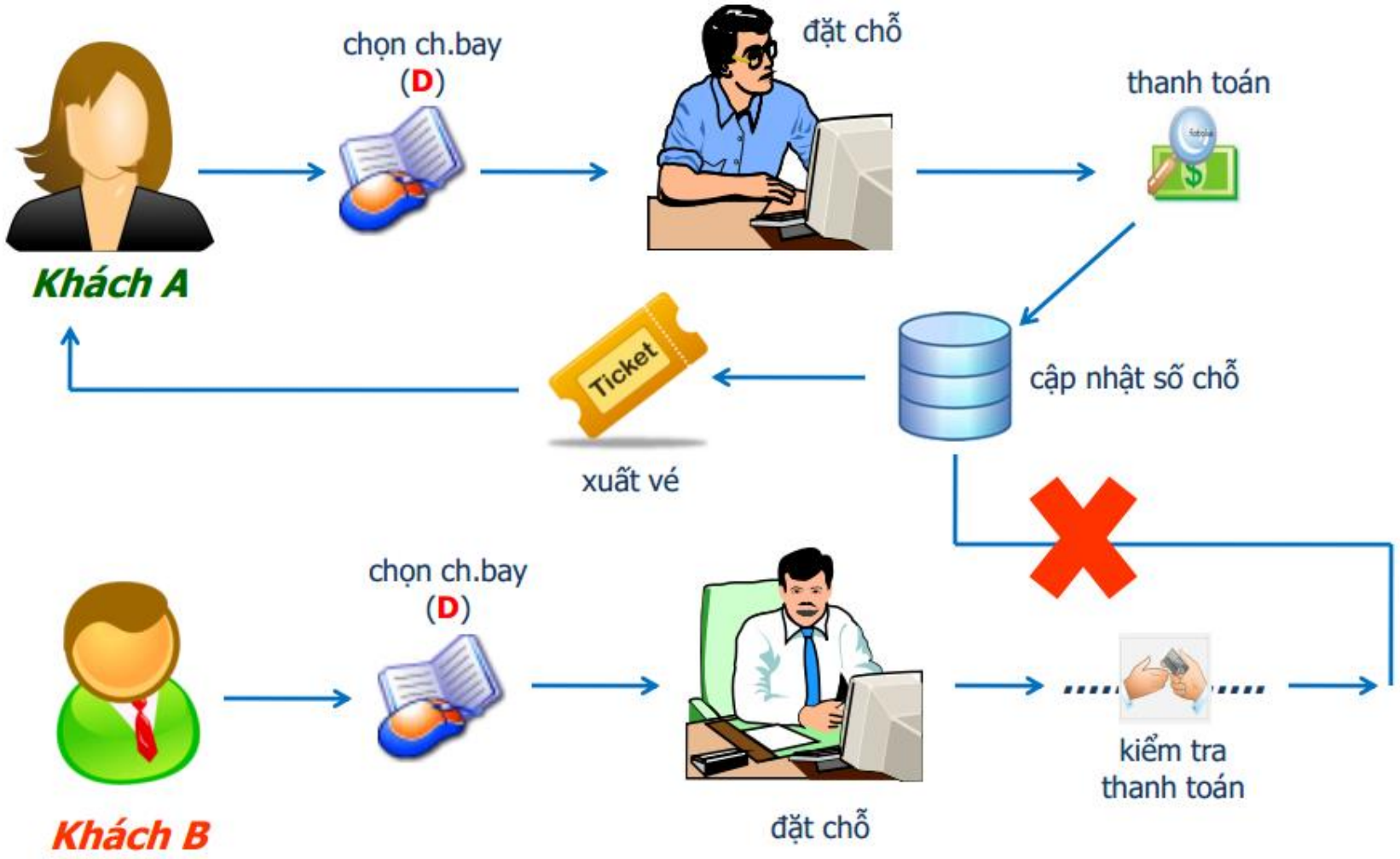
GIỚI THIỆU VỀ GIAO DỊCH

Ví dụ:



GIỚI THIỆU VỀ GIAO DỊCH

Ví dụ:

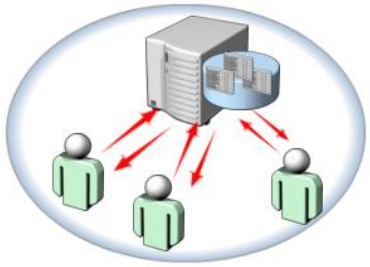


GIỚI THIỆU VỀ GIAO DỊCH

Nhiều người dùng truy cập CSDL cùng thời điểm

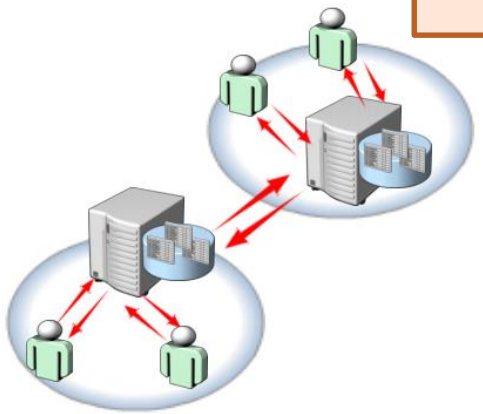
Môi trường multi-user

CSDL lớn



Tập trung

TRANH CHẤP TÀI NGUYÊN



Phân tán

GIỚI THIỆU VỀ GIAO DỊCH

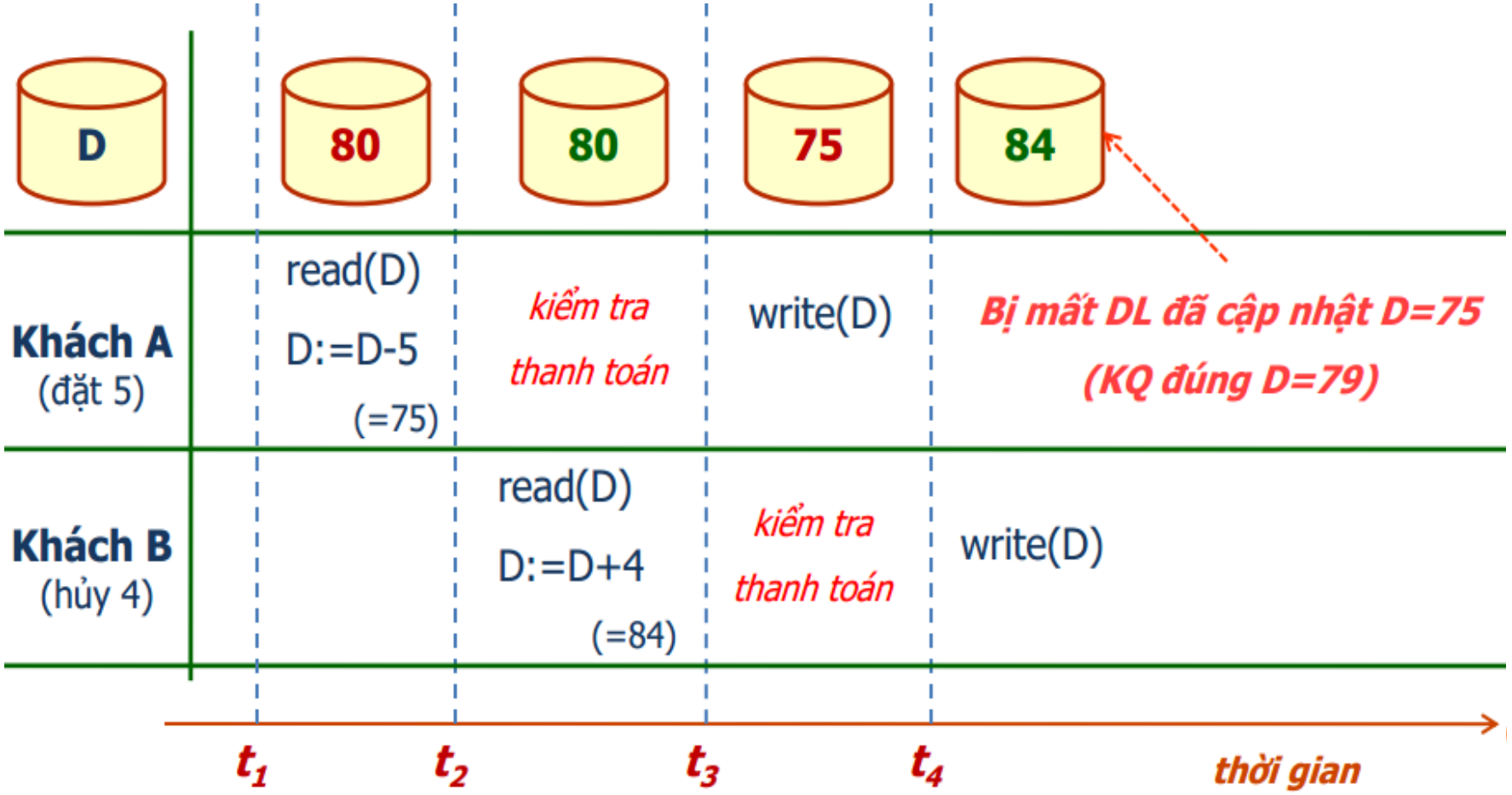
- 1 cách phân loại HQT CSDL khác là dựa trên số lượng người sử dụng đồng thời:
  - Single-user
  - Multi-user: hầu hết các HQT CSDL
- Xử lý xen kẽ (Interleaved processing)
- Xử lý song song (Parallel processing)

GIỚI THIỆU VỀ GIAO DỊCH

- Từ đó đặt ra vấn đề:
  - Mất dữ liệu đã cập nhật (Lost Update Problem)
  - Số liệu tổng hợp không chính xác (Incorrect Summary Problem)
  - Khai thác dữ liệu ‘giả’ (Dirty Read Problem)

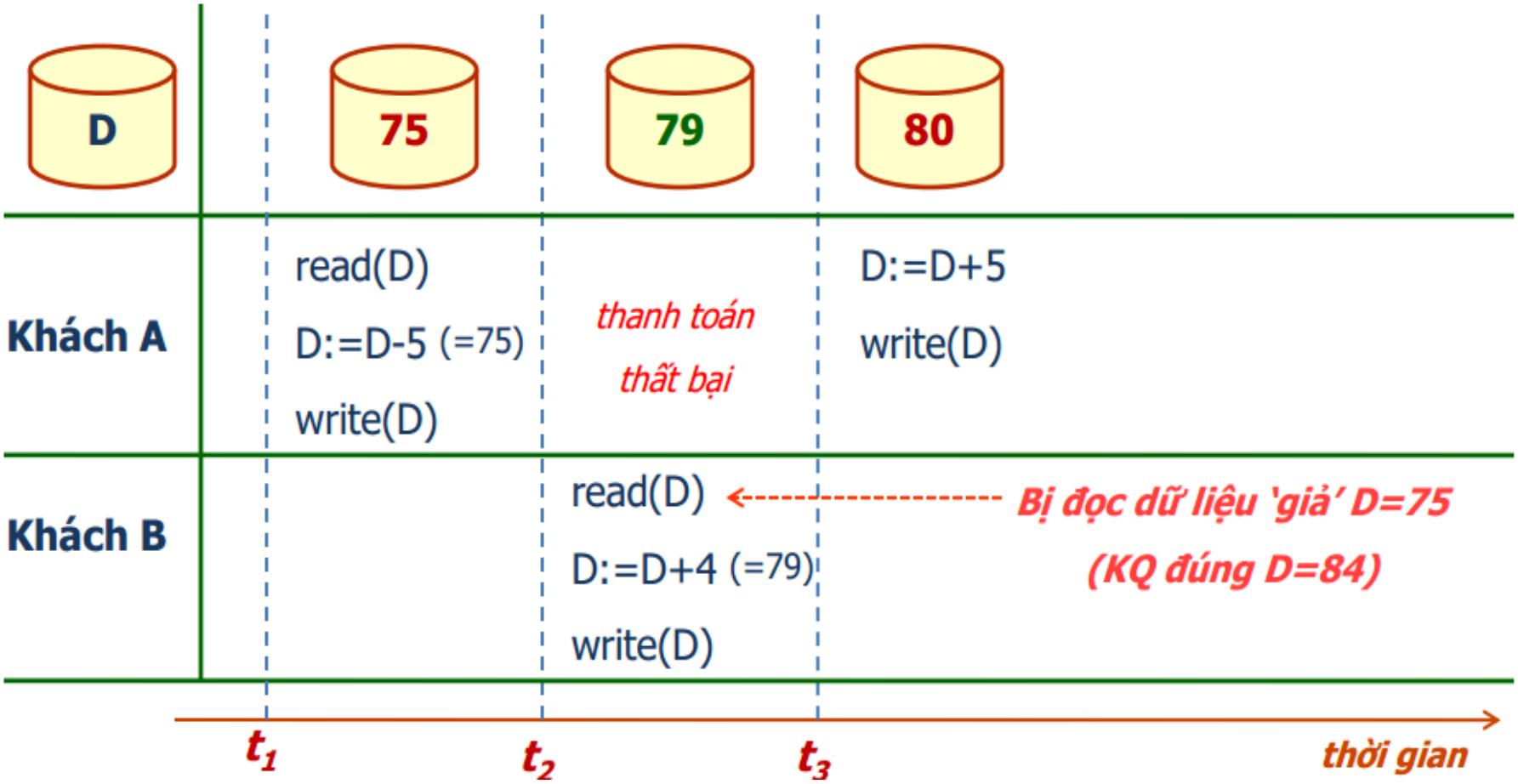
GIỚI THIỆU VỀ GIAO DỊCH

- Trường hợp 1: Mất dữ liệu đã cập nhật (Số chỗ còn D=80)



GIỚI THIỆU VỀ GIAO DỊCH

- Trường hợp 2: Khai thác dữ liệu giả (Số chỗ còn D=80)



GIỚI THIỆU VỀ GIAO DỊCH

- Trường hợp 3: Số liệu tổng hợp không chính xác
  - $W_1, W_2, \ldots, W_n$ là số chỗ đã đặt ở trạm 1, trạm 2, …, trạm n
  - Tính tổng số vé đã đặt của tất cả các trạm

$$Z_n = W_1 + W_2 + \ldots + W_n$$

  - Tính được tới trạm 4: $Z_4 = W_1 + W_2 + W_3 + W_4$

    Khi đó, trạm 1 đặt thêm 3 chỗ và trạm 2 trả lại 1 chỗ
  - Tại trạm n: $Z_n = Z_4 + W_5 + \ldots + W_n$

→ Z_n không còn chính xác

Trong cả ba trường hợp, đặt ra yêu cầu một DBMS hỗ trợ chức năng giải quyết tranh chấp có tính **đúng đắn, ổn định, an toàn, có thời gian đáp ứng ➔ GIAO DỊCH**

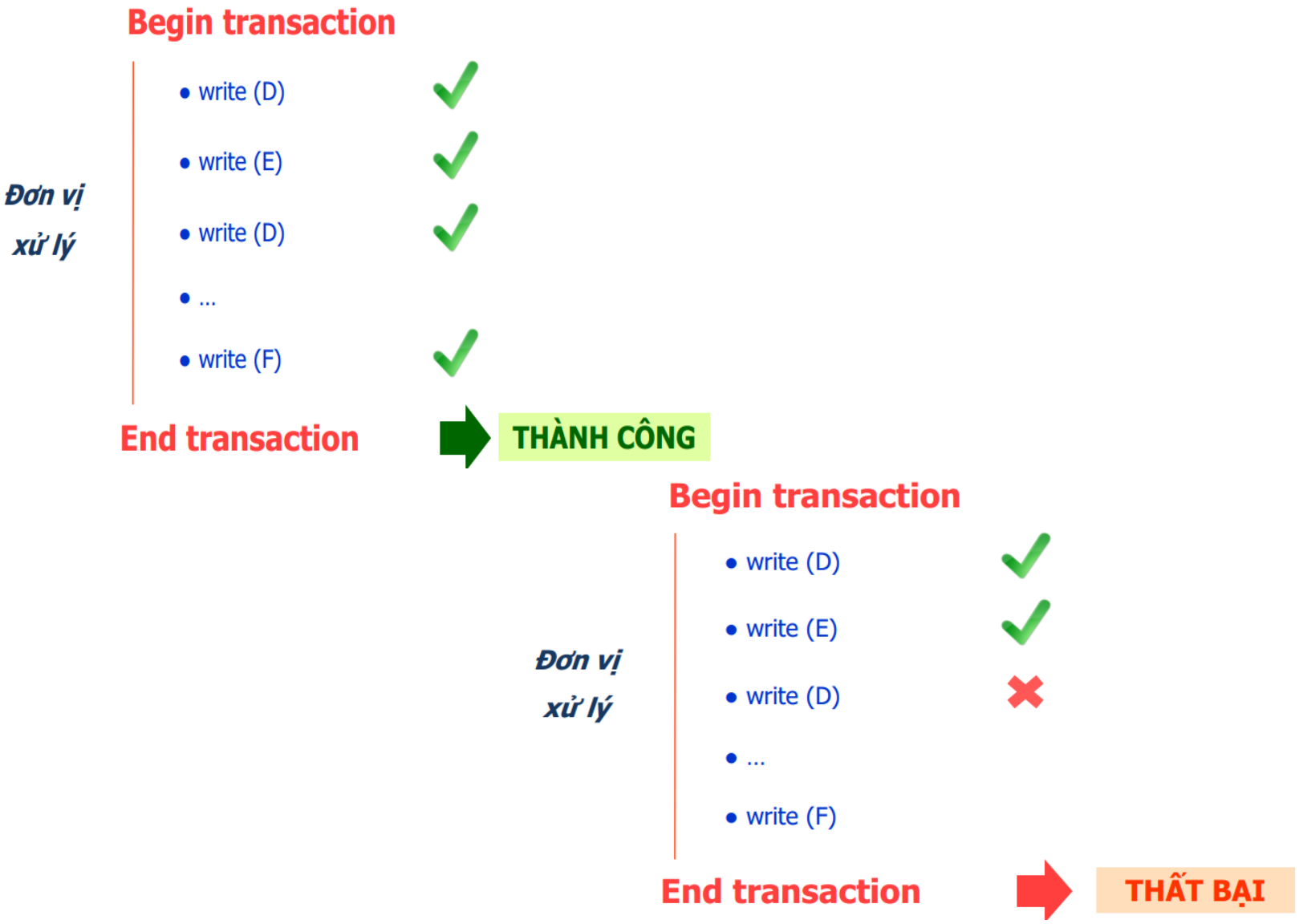
KHÁI NIỆM GIAO DỊCH (GIAO TÁC)

- Là một chương trình được thực thi đóng vai trò như một đơn vị xử lý truy xuất CSDL.
- Một giao dịch có thể gồm những xử lý trên 1 hoặc nhiều CSDL.
- Giao dịch có thể được viết trong ngôn ngữ SQL hoặc một ngôn ngữ lập trình nào đó.

KHÁI NIỆM GIAO DỊCH

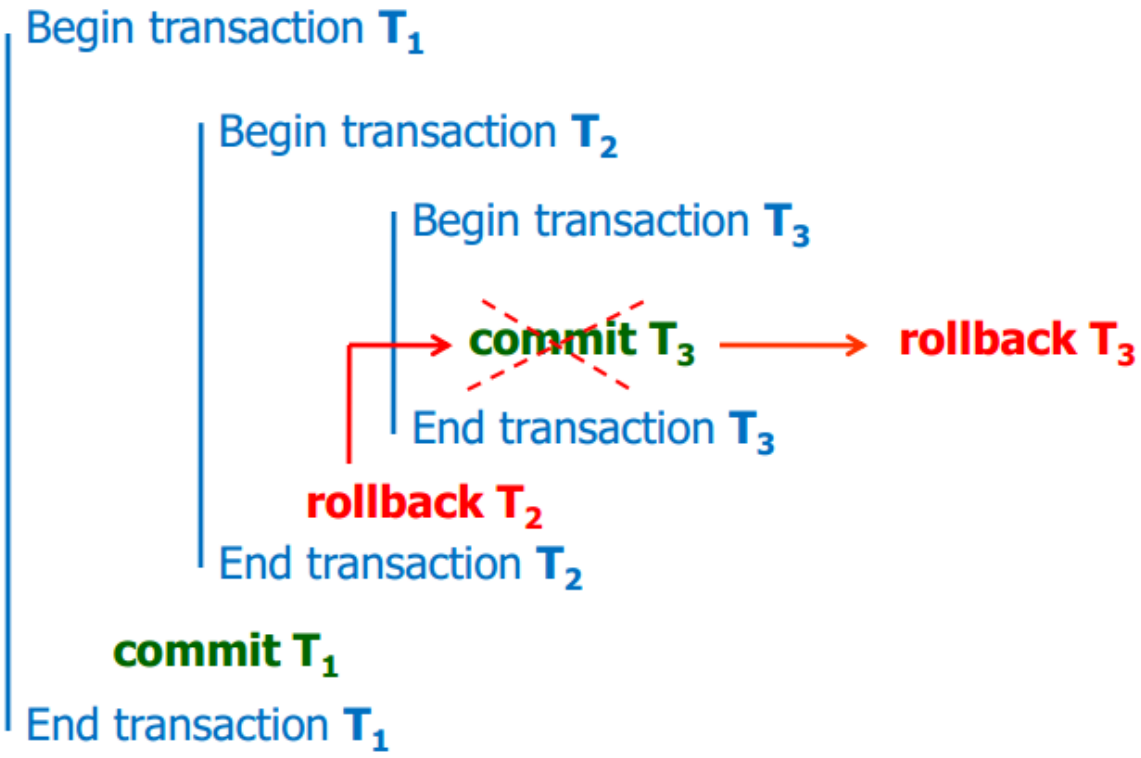
- **Thành phần của giao dịch:**
  - Begin transaction: bắt đầu giao tác
  - Read/Write: các thao tác đọc/ghi trong giao tác
  - End transaction: kết thúc giao tác
  - Commit: kết thúc thành công giao tác → Dữ liệu sẽ được ghi nhận trên CSDL
  - Rollback/Abort: giao tác bị thất bại → Khi có 1 thao tác trong giao tác không hoàn tất, trở về trạng thái trước khi giao tác bắt đầu

KHÁI NIỆM GIAO DỊCH



KHÁI NIỆM GIAO DỊCH

- Thực hiện các giao tác lồng nhau



KHÁI NIỆM GIAO DỊCH

- Chọn chuyến bay

Begin transaction

- Khóa đặt chỗ chuyến bay
- Đặt chỗ

If (thanh toán thành công)

commit

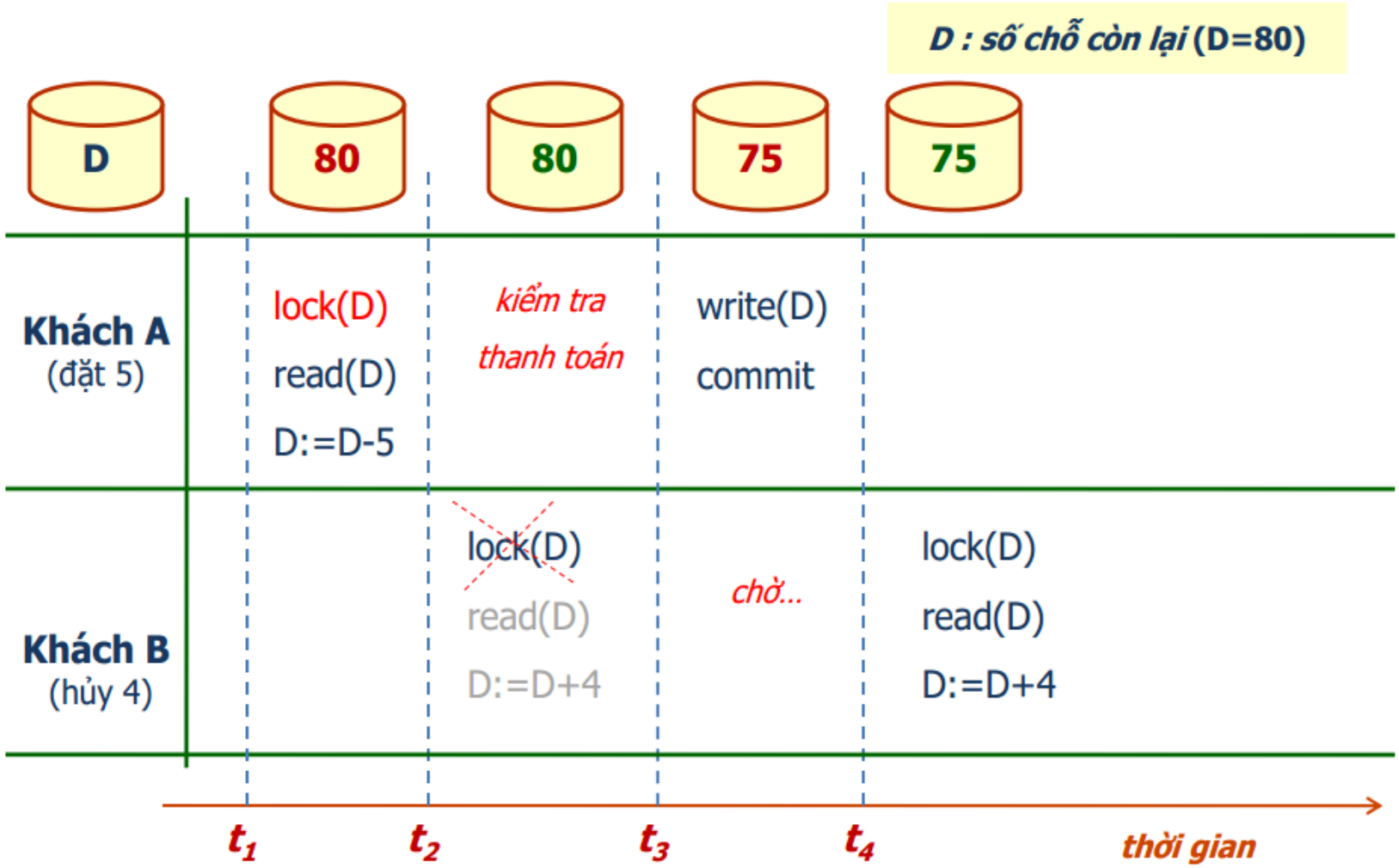
Else

rollback

- Mở khóa đặt chỗ chuyến bay

End transaction

KHÁI NIỆM GIAO DỊCH



KHÁI NIỆM GIAO DỊCH

- Để đảm bảo tính toàn vẹn của dữ liệu, phải đảm bảo các tính chất của giao dịch:
  - Tính nguyên tố (**A**tomicity): hoặc tất cả các xử lý trong giao tác được hoàn tất hoặc không có bất kỳ xử lý nào được thực hiện
  - Tính nhất quán (**C**onsistency): Khi một giao dịch thành công, phải đảm bảo sự nhất quán của CSDL trước và sau khi xảy ra giao dịch.
  - Tính cô lập (**I**solation): một giao tác khi thực hiện sẽ độc lập với các giao tác khác đang thực hiện đồng thời.
  - Tính bền vững (**D**urability): mọi thay đổi về dữ liệu khi giao tác hoàn tất phải được ghi nhận bền vững trên CSDL. Sau khi một giao dịch thành công, các thay đổi với CSDL phải còn nguyên cho dù có xảy ra sự cố hệ thống.

➔Công việc của Hệ quản trị Cơ sở dữ liệu khi xử lý giao dịch

Đọc/ghi dữ liệu

- CSDL nằm trên đĩa.
- Các truy xuất CSDL gồm:
  - Đọc nội dung X: lấy nội dung X từ CSDL trên đĩa ghi vào vùng nhớ đệm của giao dịch.
  - Ghi nội dung X: chép nội dung X từ vùng nhớ đệm của giao dịch vào lại CSDL trên đĩa.
- Thực tế: Write không nhất thiết là thực hiện việc ghi vào CSDL trên đĩa mà có thể ghi tạm trên bộ nhớ và ghi vào đĩa muộn hơn.

Ví dụ

- T là một giao dịch chuyển 50 từ Tài khoản A sang B.
  - Read(A);
  - A=A-50;
  - Write(A);
  - Read(B);
  - B=B+50;
  - Write(B);

ACID

- Tính nguyên tử (A): Sự cố xảy ra sau Write(A) và trước Read(B) → ?
- Tính nhất quán (C): Tổng A và B là không đổi khi thực hiện T (Tiền không tạo ra hoặc mất đi khi thực hiện giao dịch)
- Tồn tại thời điểm hệ thống ở trạng thái không nhất quán → cần phải có tính nguyên tử.

ACID

- Tính cô lập (I): ngay cả khi A và C được đảm bảo cho mỗi giao dịch, trạng thái không nhất quán vẫn có thể xảy ra khi một số giao dịch xảy ra đồng thời.
- VD: tại thời điểm không nhất quán tạm thời khi thực hiện T → 1 giao dịch khác xảy ra.
- Xử lý đơn giản: thực hiện tuần tự các giao dịch → giảm hiệu năng hệ thống.

ACID

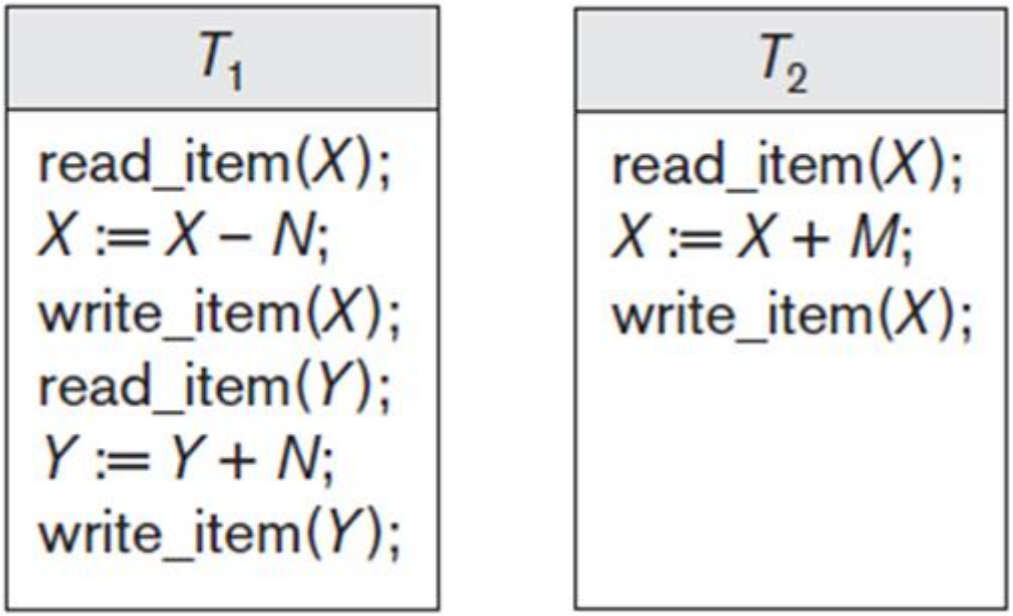
- Tính bền vững (D): giả sử xảy ra sự cố mất dữ liệu trên bộ nhớ, dữ liệu trên đĩa không mất.
- Đảm bảo tính bền vững:
  - Đảm bảo các thay đổi đã được viết trên đĩa trước khi giao dịch kết thúc.
  - Thông tin về những thay đổi do giao dịch được viết trên đĩa đủ để xây dựng lại CSDL khi hệ thống khởi động lại sau sự cố.

THÀNH PHẦN CỦA HQT CSDL

- Đảm bảo tính nguyên tử: thành phần quản trị giao dịch (transaction management component).
- Đảm bảo tính bền vững: thành phần quản trị phục hồi (recovery management component).
- Đảm bảo tính cô lập: thành phần quản trị cạnh tranh (concurrency control component)

TÌNH HUỐNG XEM XÉT

- 2 giao dịch T1 và T2 như sau



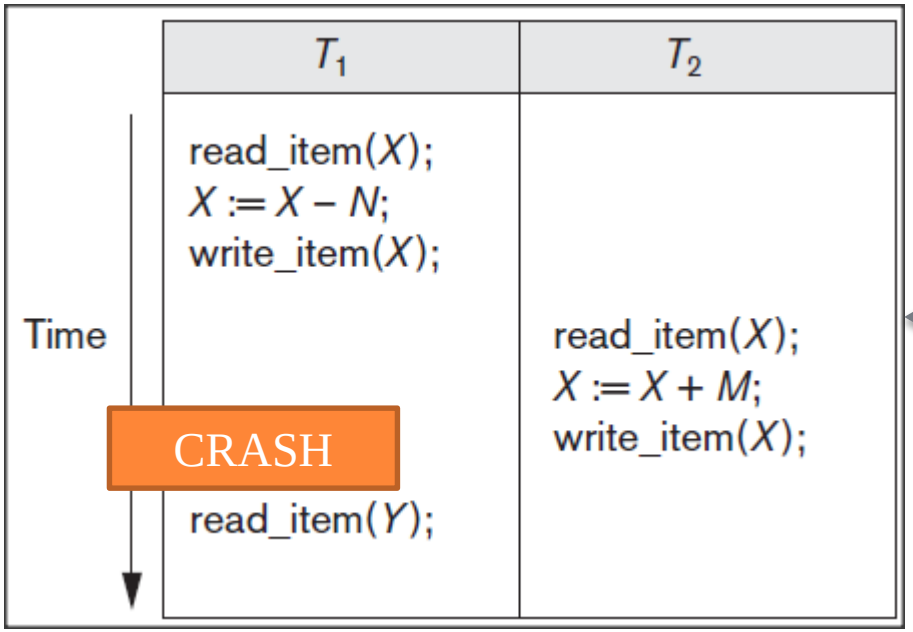
Sự cần thiết của Thành phần quản trị cạnh tranh

TIME

T_1	T_2
read_item(X); $X := X - N$;	
	read_item(X); $X := X + M$;
write_item(X); read_item(Y);	
	write_item(X);
$Y := Y + N$; write_item(Y);	

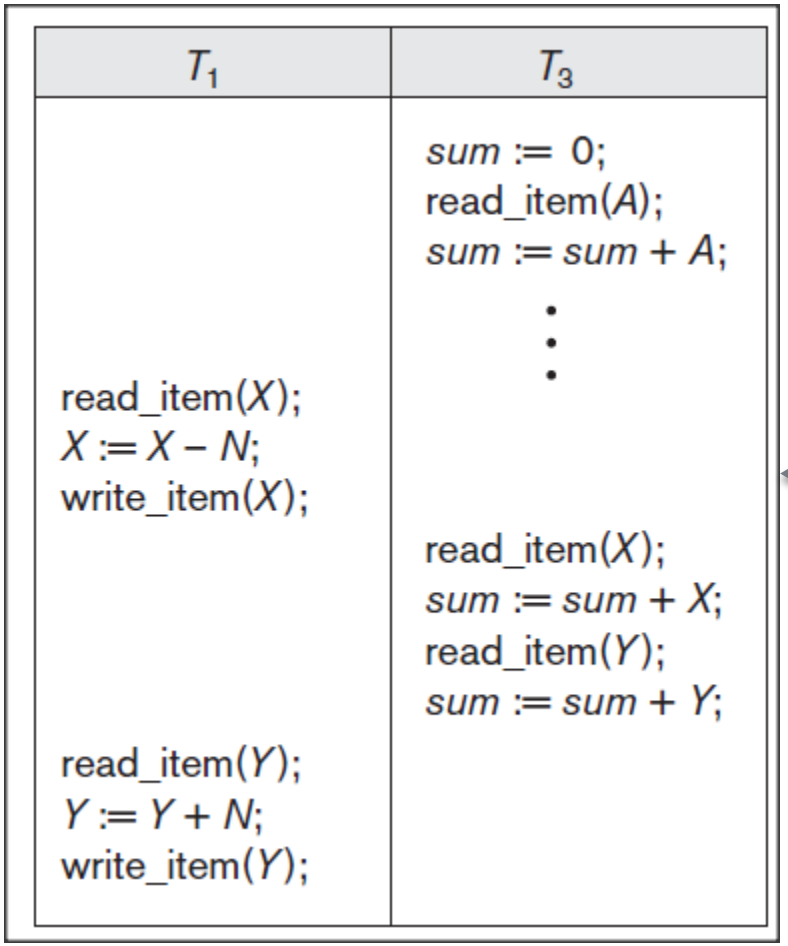
Item X có một giá trị không chính xác bởi vì nó được cập nhật bởi T1 bị mất (ghi đè)

Sự cần thiết của Thành phần quản trị cạnh tranh



Giao dịch T_1 thất bại và phải thay đổi giá trị của X trở lại giá trị cũ của mình, trong khi đó T_2 đã đọc các giá trị tạm thời không chính xác của X

Sự cần thiết của Thành phần quản trị cạnh tranh



T_1	T_3

	sum := 0;
	read_item(*A*);
	sum := *sum* + *A*;
	. . .

read_item(*X*);
X := *X* − *N*;
write_item(*X*);	

	read_item(*X*);
	sum := *sum* + *X*;
	read_item(*Y*);
	sum := *sum* + *Y*;

read_item(*Y*);
Y := *Y* + *N*;
write_item(*Y*);	

T3 đọc X sau khi N được trừ và đọc Y trước khi N được bổ sung; một bản tóm tắt sai là kết quả.

THÀNH PHẦN QUẢN TRỊ PHỤC HỒI ĐƠN GIẢN

- Điều kiện: 1 giao dịch hoạt động ở 1 thời điểm và thực hiện trên 1 bản sao của CSDL thật.
- CSDL thật là file trên đĩa. Một con trỏ db_pointer lưu trên đĩa và trỏ đến bản sao hiện hành của CSDL đó.
- Một giao dịch muốn cập nhật CSDL, đầu tiên tạo ra một bản sao đầy đủ của CSDL.Nếu giao dịch bị bỏ dở, bản sao mới bị xoá. Bản sao cũ của CSDL không bị ảnh hưởng.

THÀNH PHẦN QUẢN TRỊ PHỤC HỒI ĐƠN GIẢN

- Nếu giao dịch hoàn tất, nó được được bàn giao:
  - Đảm bảo rằng tất cả các trang của bản sao mới đã được viết lên đĩa.
  - Con trỏ được cập nhật để trỏ đến bản sao mới; bản sao mới trở thành bản sao hiện hành của CSDL.
  - Bản sao cũ bị xoá đi. Giao dịch được gọi là đã được được bàn giao (committed) tại thời điểm sự cập nhật con trỏ db_pointer được ghi lên đĩa.

THÀNH PHẦN QUẢN TRỊ PHỤC HỒI ĐƠN GIẢN

- Sự cố giao dịch:
  - Nếu giao dịch thất bại tại thời điểm bất kỳ trước khi con trỏ db_pointer được cập nhật, nội dung cũ của CSDL không bị ảnh hưởng. Ta có thể bỏ dở giao dịch bởi xoá bản sao mới.
- Mỗi khi giao dịch được được bàn giao (committed), tất cả các cập nhật mà nó đã thực hiện là ở trong CSDL được trỏ bởi db_pointer.

THÀNH PHẦN QUẢN TRỊ PHỤC HỒI ĐƠN GIẢN

- Sự cố hệ thống:
  - Sự cố hệ thống xảy ra tại thời điểm trước khi db_pointer đã được cập nhật được viết lên đĩa. Khi đó, khi hệ thống khởi động lại, nó sẽ đọc db_pointer và như vậy sẽ thấy nội dung gốc của CSDL – không hiệu quả nào của giao dịch được nhìn thấy trên CSDL.
  - Sự cố hệ thống xảy ra sau khi db_pointer đã được cập nhật lên đĩa: Khi hệ thống khởi động lại, nó sẽ đọc db_pointer và sẽ thấy nội dung của CSDL sau tất cả các cập nhật đã thực hiện bởi giao dịch.

THÀNH PHẦN QUẢN TRỊ PHỤC HỒI ĐƠN GIẢN

- Việc ghi db_pointer phải là nguyên tử.
- Sự thực thi này cực kỳ thiếu hiệu quả trong ngữ cảnh CSDL lớn, do sự thực hiện một giao dịch đòi hỏi phải sao toàn bộ CSDL.
- Sự thực thi này không cho phép các giao dịch thực hiện đồng thời với các giao dịch khác.

TRẠNG THÁI GIAO DỊCH

- Trên thực tế, một giao dịch có thể không hoàn tất công việc → Giao dịch không hoàn thành (Uncommited).

- Tính nguyên tử: Giao dịch bị bỏ dở này không thay đổi trạng thái CSDL → tất cả các thay đổi phải được rollback.

TRẠNG THÁI GIAO DỊCH

- Một giao dịch hoàn tất gọi là được bàn giao (committed) và không thể hủy bỏ những thay đổi được thực hiện bằng committed transaction.

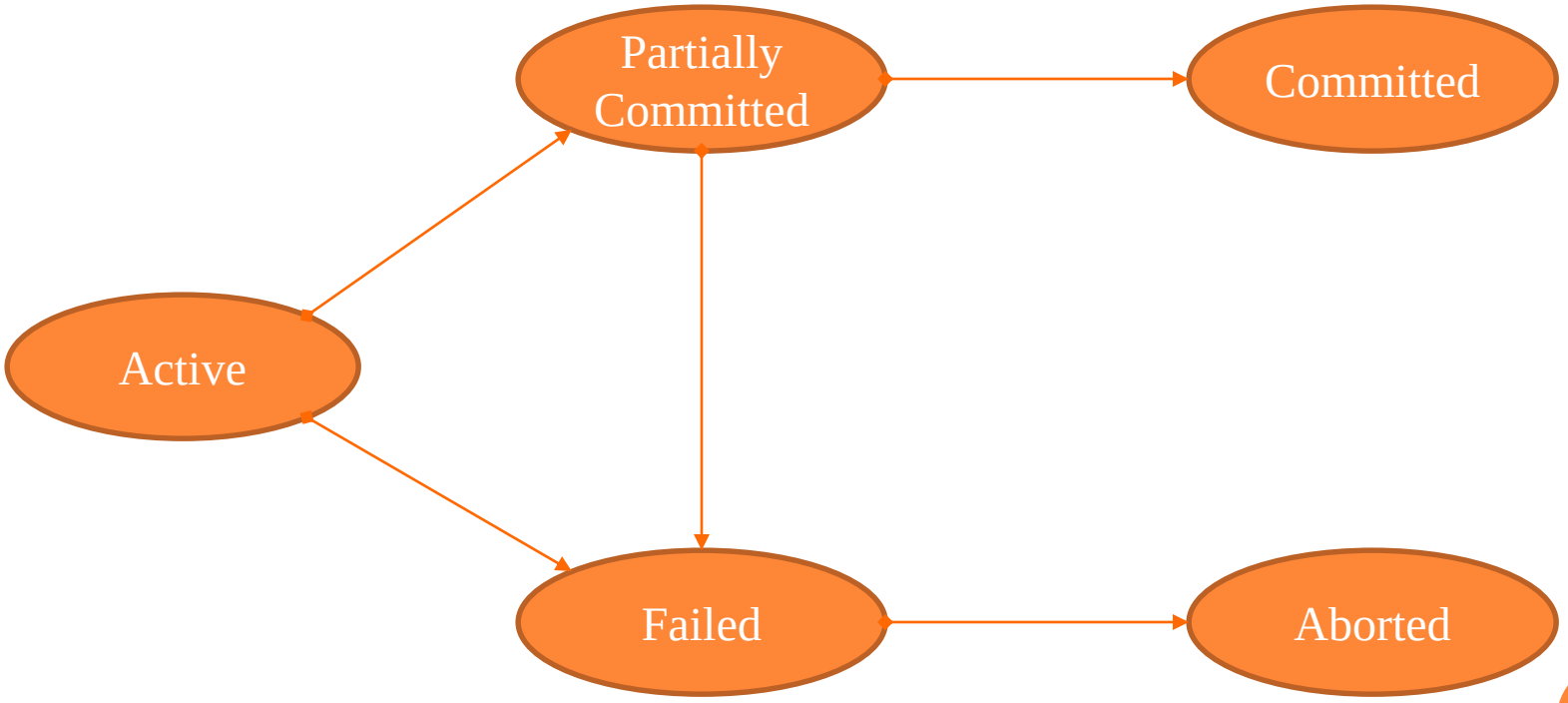
- Nếu muốn thay đổi: thực hiện giao dịch bù (compensating transaction) thuộc về người sử dụng.

CÁC TRẠNG THÁI CỦA GIAO DỊCH

- Hoạt động (Active): trạng thái khởi đầu
- Được bàn giao bộ phận (Partially committed): khi lệnh cuối cùng được thực hiện.
- Thất bai (Failed)
- Bỏ dở (aborted)
- Được bàn giao (Committed): giao dịch thành công.

Các trạng thái của giao dịch

- Giao dịch kết thúc: Committed hoặc Aborted.



Xử lý giao dịch đồng thời

- Khó khăn trong vấn đề đảm bảo sự nhất quán dữ liệu.
- Cách đơn giản nhất: thực hiện tuần tự.
- Một giao dịch gồm nhiều bước:
  - I/O
  - Xử lý trên CPU
- Tăng hiệu suất sử dụng CPU và đĩa.
- Tăng số lượng giao dịch có thể xử lý trong một đơn vị thời gian.

XỬ LÝ GIAO DỊCH ĐỒNG THỜI

- Một số giao dịch khác biệt về thời gian thực hiện.
- Thực hiện tuần tự: chờ đợi quá trình dài hoàn tất…
- Để đảm bảo tính nhất quán: hệ QT CSDL phải điều khiển sự trao đổi giữa các giao dịch đồng thời – sơ đồ điều khiển cạnh tranh (concurrency control scheme)

LỊCH TRÌNH

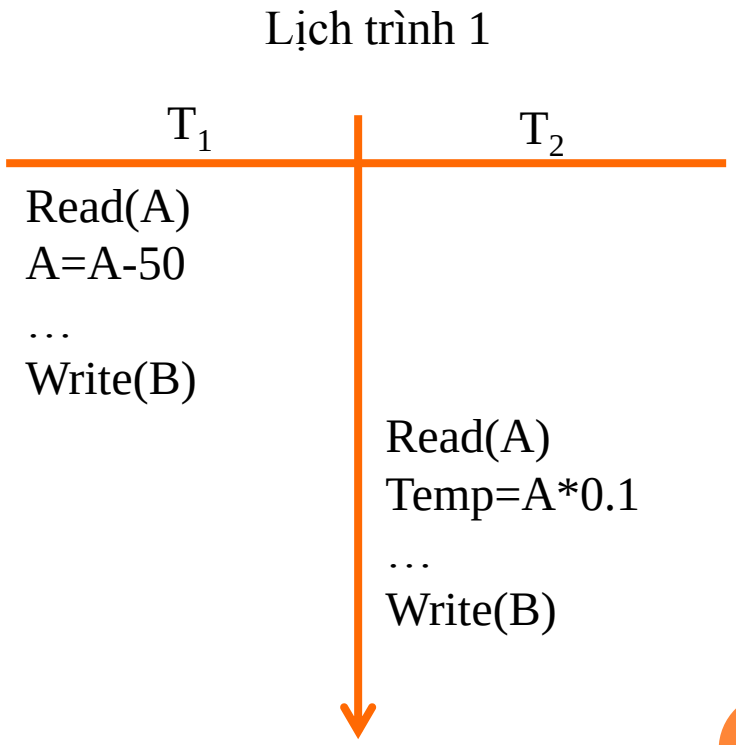
- Giá trị A, B ban đầu là 1000 và 2000
- T_1: Chuyển 50 từ tài khoản A sang tài khoản B.
  - Read(A)
  - A=A-50
  - Write(A)
  - Read(B)
  - B=B+50
  - Write(B)

LỊCH TRÌNH

- T_2: Chuyển 10% số dư từ tài khoản A sang tài khoản B.
  - Read(A)
  - Temp=A*0.1
  - A=A-Temp
  - Write(A)
  - Read(B)
  - B=B+Temp
  - Write(B)

LỊCH TRÌNH

- Giả sử 2 giao dịch này được thực hiện theo thứ tự T_1 rồi đến T_2.
- ➔ Lịch trình 1
- Kết quả: A, B?
- A+B trước và sau gd?



LỊCH TRÌNH

- Giả sử trường hợp giao dịch thực hiện theo thứ tự T_2 rồi đến T_1.
- Lịch trình?
- Kết quả?

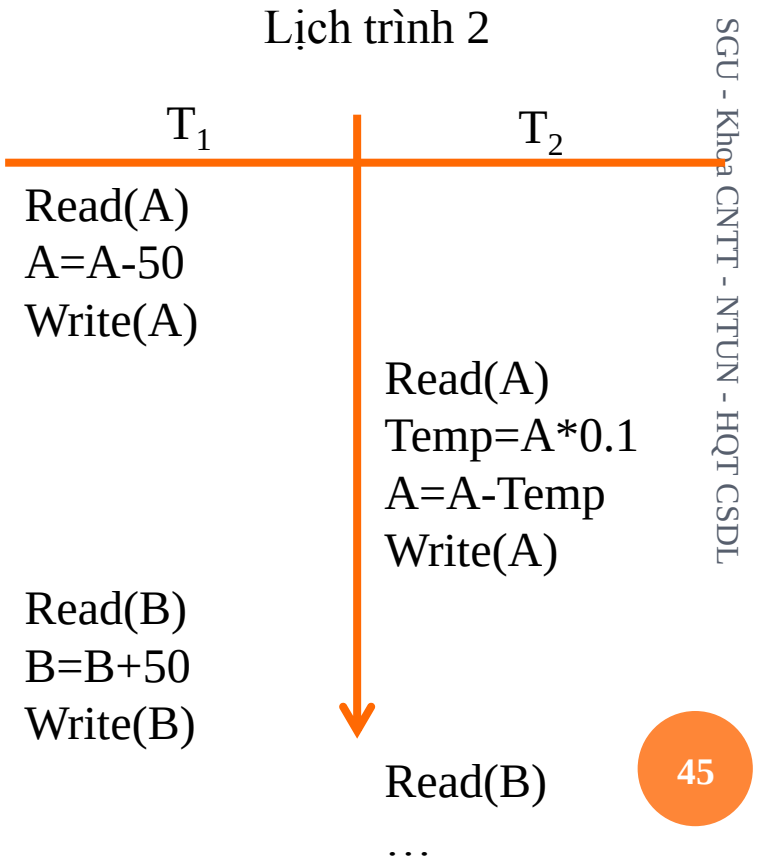
- → **Lịch trình**: biểu diễn trình tự thời gian các chỉ thị được thực hiện trong hệ thống.

LỊCH TRÌNH TUẦN TỰ

- Khái niệm: Là lịch trình mà các chỉ thị được thực hiện tuần tự đến khi hoàn tất một giao dịch thì mới chuyển sang thực hiện chỉ thị của giao dịch khác (Không có chỉ thị của giao dịch khác chen vào).
- N giao dịch → N! lịch trình tuần tự hợp lệ khác nhau.

LỊCH TRÌNH CÓ XỬ LÝ ĐỒNG THỜI

- Trên thực tế (chạy đồng thời): số lịch trình có thể như thế nào với N! ???
- Ví dụ:



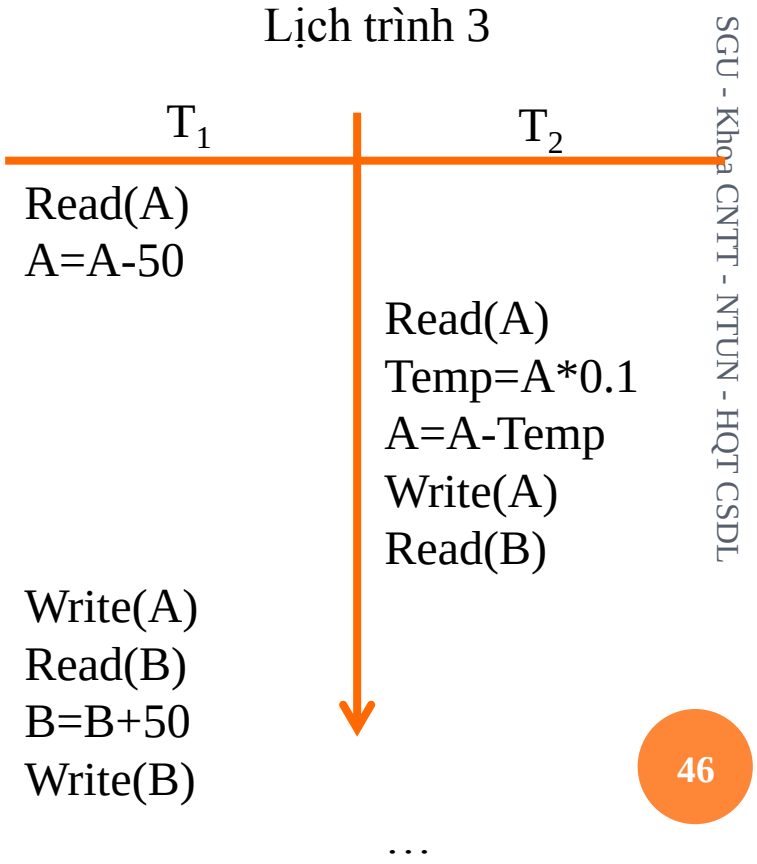
LỊCH TRÌNH CÓ XỬ LÝ ĐỒNG THỜI

- Tất cả lịch trình có xử lý đồng thời đều tương đương nhau???
- Kết quả: A, B?
- Trước và sau giao dịch

A+B?

- →Để HĐH xử lý cạnh

tranh hay không?

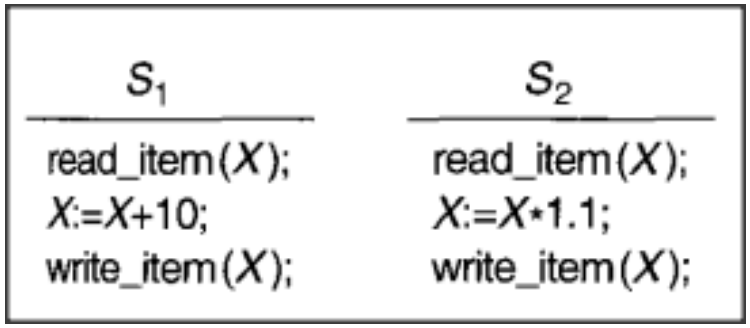


LỊCH TRÌNH CÓ XỬ LÝ ĐỒNG THỜI

- Nhiệm vụ của hệ quản trị CSDL là đảm bảo tính nhất quán của CSDL khi xử lý giao dịch đồng thời.
- Nhiệm vụ của thành phần điều khiển cạnh tranh (concurrency control component).
- Đảm bảo sự nhất quán của CSDL bằng cách **đảm bảo lịch trình được thực hiện có hiệu quả như một lịch trình tuần tự**.

TÍNH KHẢ TUẦN TỰ

- Tính khả tuần tự (Serializability): Một lịch trình có tính khả tuần tự là một lịch trình **tương đương** với một lịch trình tuần tự nào đó.
- **Kết quả tương đương** (Result Equivalent): phát sinh cùng trạng thái cuối của CSDL.



S_1	S_2
read_item(X);	read_item(X);
X:=X+10;	X:=X*1.1;
write_item(X);	write_item(X);

- → KQ tương đương là chưa đủ để thể hiện sự tương đương (equivalent) của 2 lịch trình.

Lịch trình tuần tự

???

Lịch trình khả tuần tự

Quy ước

- Chỉ quan tâm thao tác Read và Write trên các dữ liệu.
- Giữa Read(X) và Write(X) sẽ có 1 dãy thao tác tùy ý trên bản sao của hạng mục dữ liệu X trong bộ nhớ đệm.

BIỂU DIỄN LỊCH TRÌNH

Lịch trình 3

T_1	T_2
Read(A)	
A=A-50	
Write(A)	
	Read(A)
	Temp=A*0.1
	A=A-Temp
	Write(A)
Read(B)	
B=B+50	
Write(B)	
	Read(B)
	B=B+Temp
	Write(B)

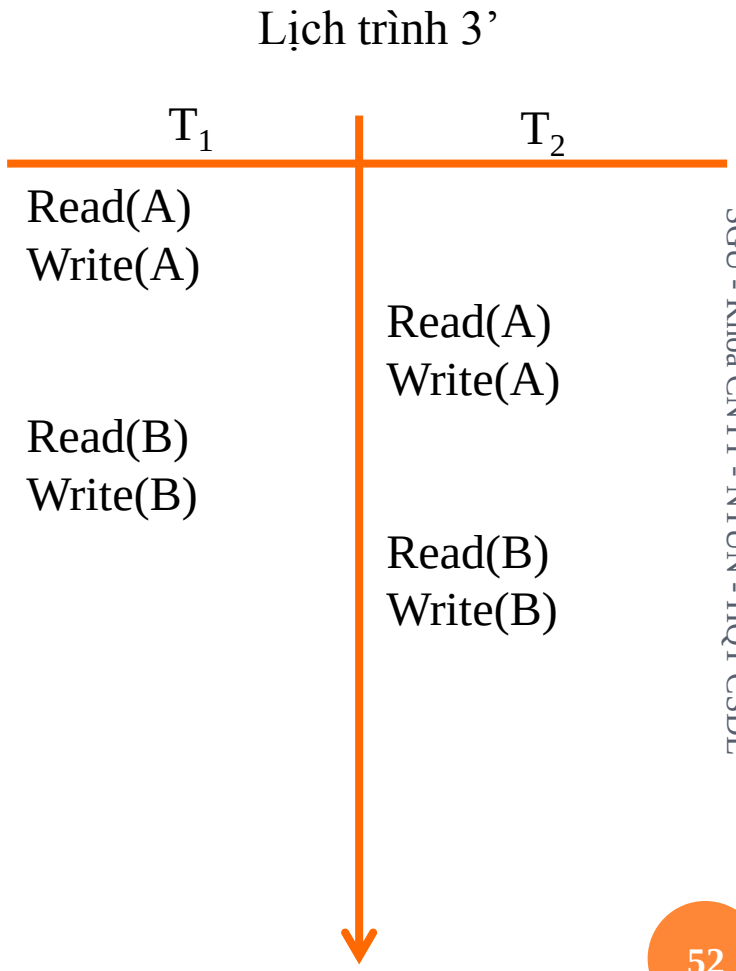
Lịch trình 3’

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

Biểu diễn lịch trình

- S: $R_1(A)$ $W_1(A)$ $R_2(A)$ $W_2(A)$ $R_1(B)$ $W_1(B)$ $R_2(B)$ $W_2(B)$

Lịch trình 3’



Xung đột

- Xung đột (Conflict): 2 chỉ thị xung đột khi ta không thể thay đổi **thứ tự thực hiện** của chúng.
- Xét 2 chỉ thị Ii và Ij của 2 giao dịch Ti và Tj tương ứng.

- Ii(**X**) và Ij(**Y**): không xung đột.
- Ii=Read(X) và Ij=Read(X): đọc cùng giá trị X bất kể thứ tự Ii và Ij.

Xung đột

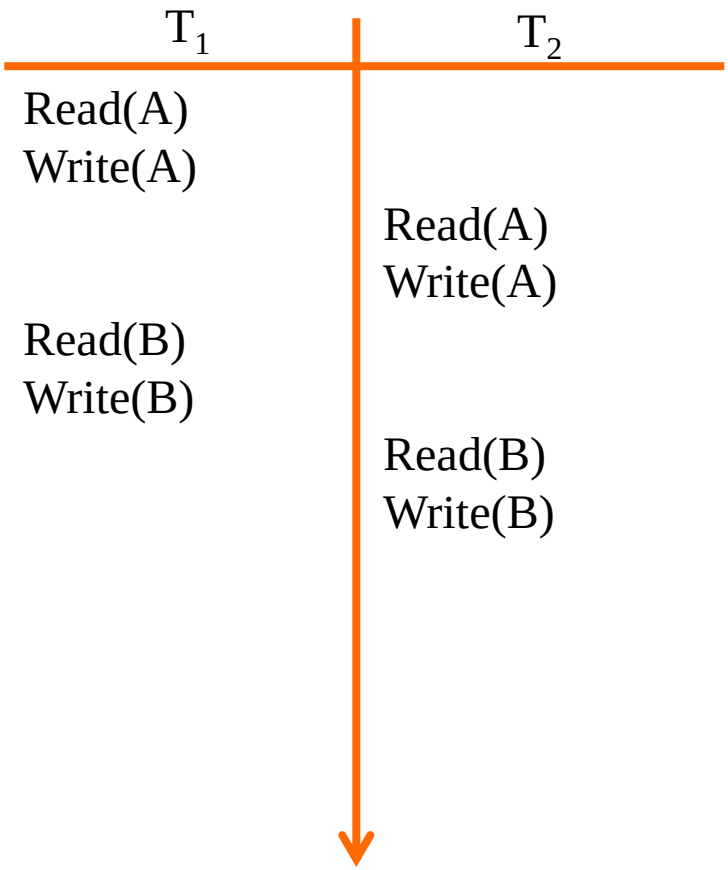
- I_i=Read(X) và I_j=Write(X): xung đột.
- I_i=Write(X) và I_j=Read(X): tương tự.
- I_i=Write(X) và I_j=Write(X): thứ tự thực hiện sẽ liên quan đến
  - Read(X) tiếp theo sau bị ảnh hưởng
  - Hoặc Trạng thái CSDL bị ảnh hưởng.

XUNG ĐỘT

- Vậy có thể tổng hợp lại như sau:
- 2 chỉ thị trong một lịch trình gọi là xung đột khi nó thỏa 3 điều kiện sau:
  1. Thuộc về 2 giao dịch khác nhau
  2. Thực hiện trên cùng dữ liệu X
  3. Ít nhất một chỉ thị là Write(X)

Ví dụ

Lịch trình 3’



- 1 – 2: Không xét
- 2 – 3: Yes
- 4 – 5: No
- 6 – 7: Yes

TƯƠNG ĐƯƠNG XUNG ĐỘT

- 2 lịch trình gọi là tương đương về mặt xung đột (conflict equivalent) nếu thứ tự của 2 chỉ thị xung đột giống nhau trong cả 2 lịch trình.
- VD: S1: R1(X), W2(X) và S2: W2(X), R1(X): không tương đương xung đột.

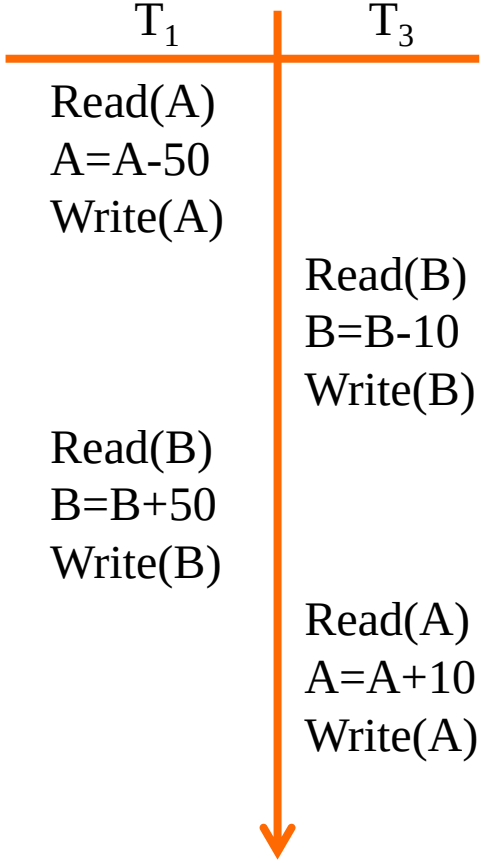
KHẢ TUẦN TỰ XUNG ĐỘT

- Một lịch trình S gọi là khả tuần tự xung đột (conflict serializable) nếu nó tương đương xung đột với 1 lịch trình tuần tự S’ nào đó.

- Khi đó ta có thể thay đổi thứ tự các chỉ thị không xung đột trong S cho đến khi ta được một lịch trình tuần tự S’ tương đương S.

CHÚ Ý

Lịch trình 5



- A=1000, B=2000
- Xét T2 là giao dịch chuyển 10 từ B sang A.
- Kết quả A, B?
- Lịch trình 5 có khả tuần tự xung đột hay không?

KIỂM TRA KHẢ TUẦN TỰ XUNG ĐỘT

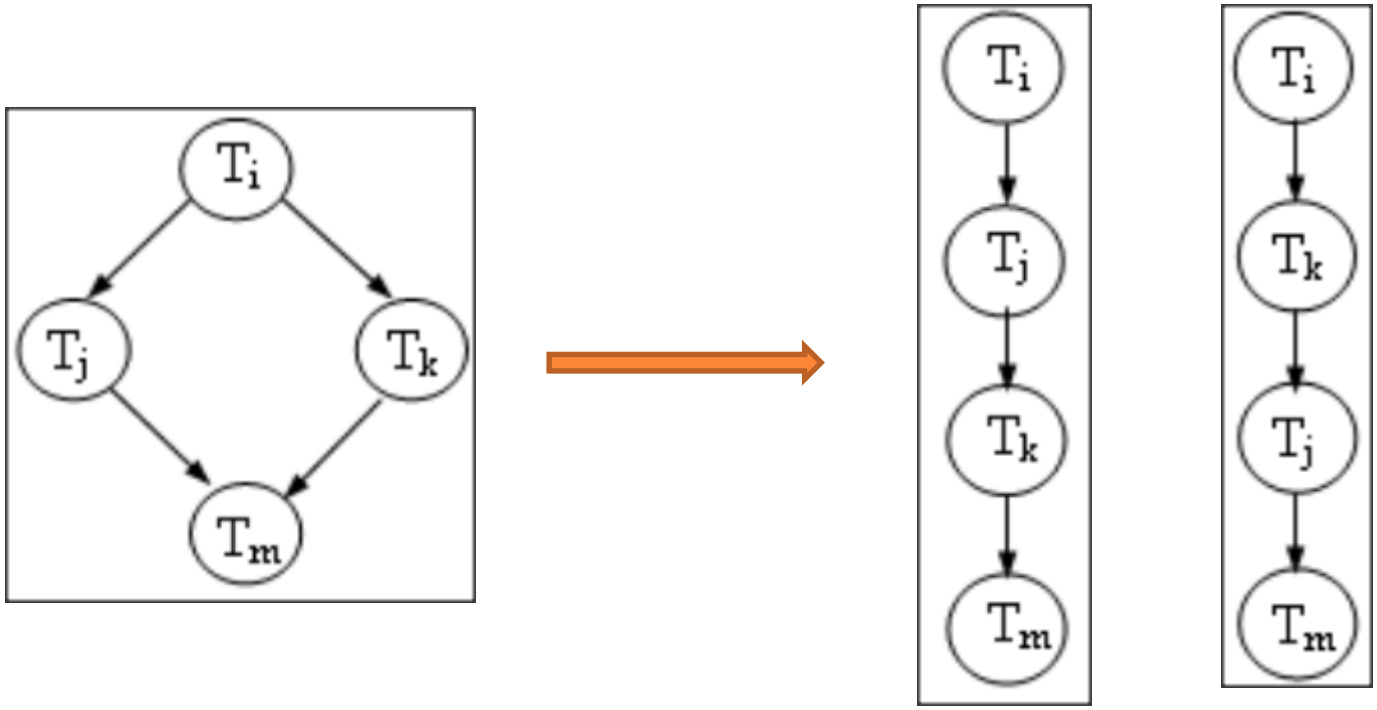
- Xây dựng một đồ thị có hướng, gọi là đồ thị trình tự (precedence graph) từ lịch trình S cần kiểm tra.
- G=(N, E):
  - N={T1, T2, … , Tn}
  - E là tập hợp các cung.

KIỂM TRA KHẢ TUẦN TỰ XUNG ĐỘT

- Tạo 1 cung $T_i \rightarrow T_j$ khi:
  - T_i thực hiện Write(X) sau đó T_j thực hiện Read(X).
  - T_i thực hiện Read(X) sau đó T_j thực hiện Write(X).
  - T_i thực hiện Write(X) sau đó T_j thực hiện Write(X).
- Lịch trình S là khả tuần tự xung đột nếu G không có chu trình.
- Chu trình?

THỨ TỰ KHẢ TUẦN TỰ

- Thứ tự có thể có được thông qua Topological sorting.



Ví dụ

TIME

T_1	T_2
read_item(X);	
X:=X-N;	
write_item(X);	
read_item(Y);	
Y:=Y+N;	
write_item(Y);	
	read_item(X);
	X:=X+M;
	write_item(X);

LỊCH TRÌNH A

TIME

T_1	T_2
	read_item(X);
	X:=X+M;
	write_item(X);
read_item(X);	
X:=X-N;	
write_item(X);	
read_item(Y);	
Y:=Y+N;	
write_item(Y);	

LỊCH TRÌNH B

TIME

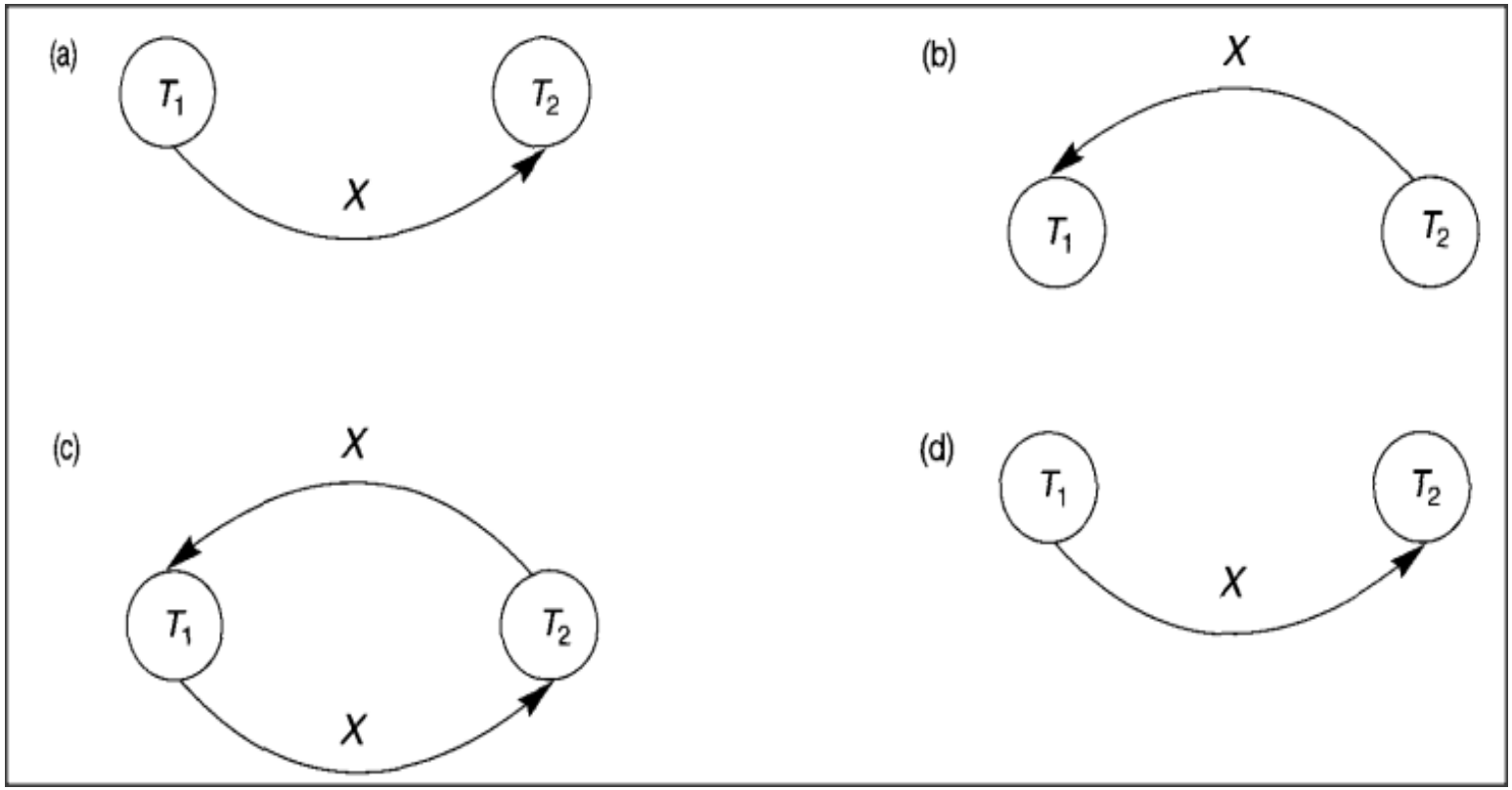
T_1	T_2
read_item(X);	
X:=X-N;	
	read_item(X);
	X:=X+M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y:=Y+N;	
write_item(Y);	

LỊCH TRÌNH C

T_1	T_2
read_item(X);	
X:=X-N;	
write_item(X);	
	read_item(X);
	X:=X+M;
	write_item(X);
read_item(Y);	
Y:=Y+N;	
write_item(Y);	

LỊCH TRÌNH D

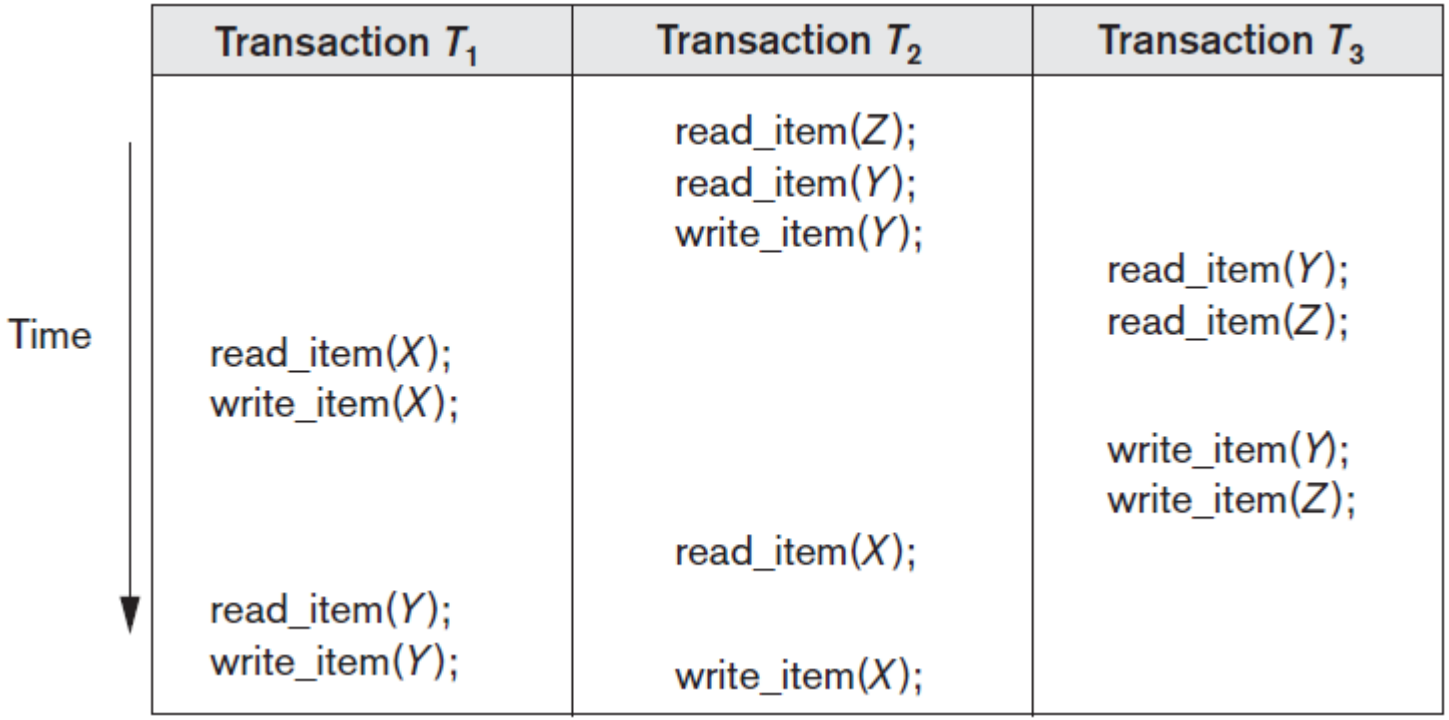
Ví dụ



Ứng dụng của khả tuần tự

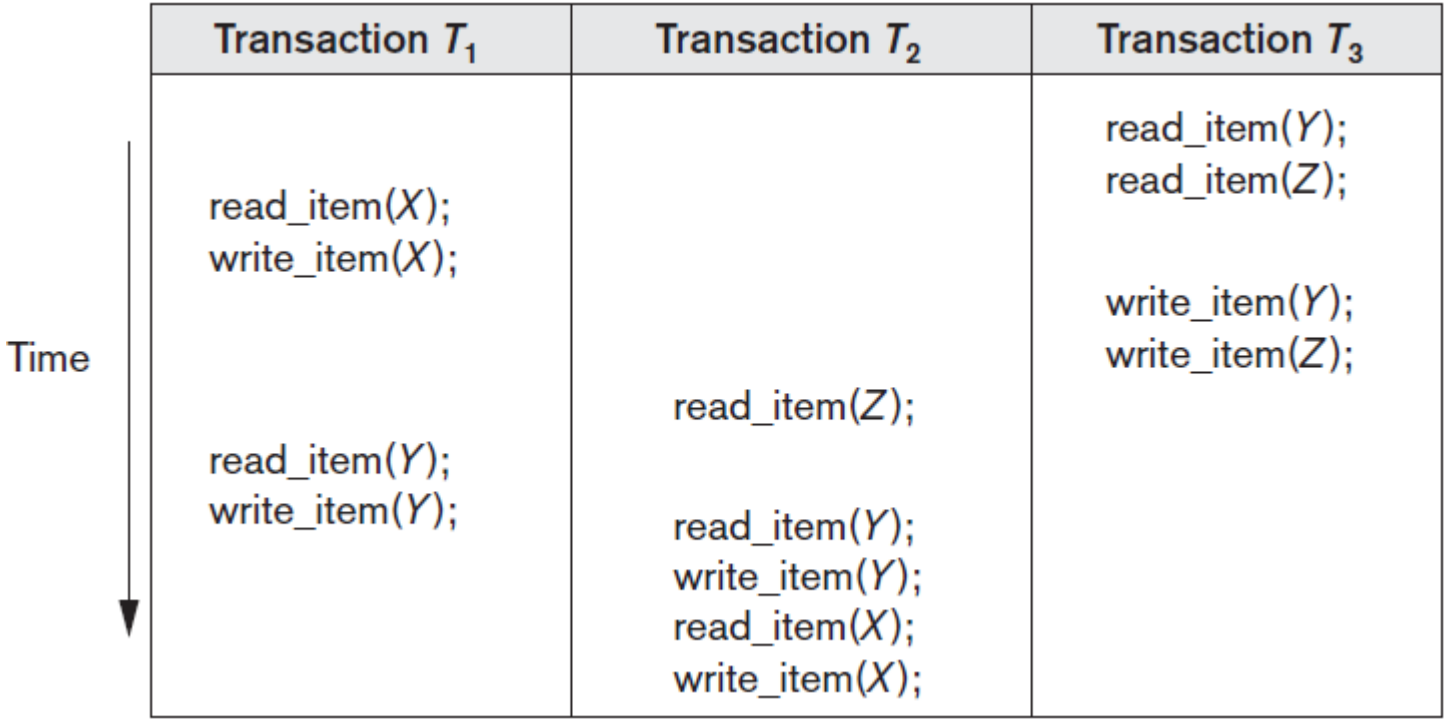
- Trên thực tế, khó kiểm tra tính khả tuần tự của lịch trình.
- Nếu giao dịch được thực hiện sau đó kết quả của lịch trình được kiểm tra tính khả tuần tự, khi đó ta phải hủy bỏ lịch trình nếu nó không thỏa → Không thực tế.
- Khi giao dịch được gửi liên tục, rất khó kiểm soát khi nào lịch trình bắt đầu và kết thúc.

BÀI TẬP



LỊCH TRÌNH E

BÀI TẬP



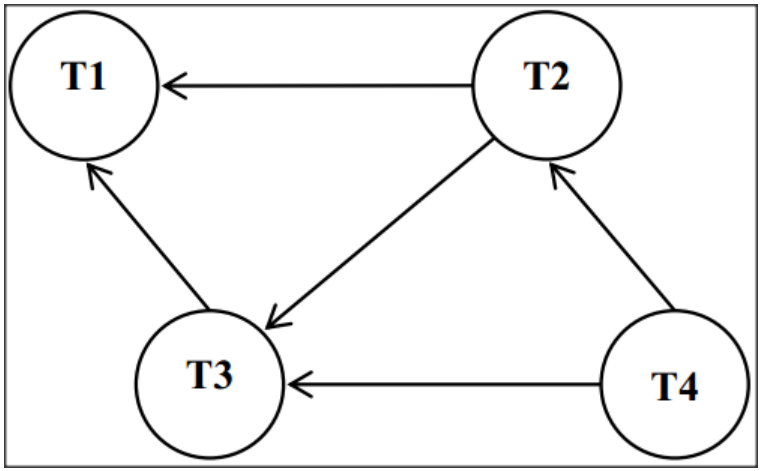
LỊCH TRÌNH F

BÀI TẬP

- Lịch biểu nào dưới đây là khả tuần tự?
  - R2(Y), R1(X), R1(Y), R3(X), W3(X), W2(Y), W1(X)
  - R2(Y), R1(X), R1(Y), R3(Z), W3(Z), W2(Y), W1(X)
  - R2(Y), R1(X), R1(Y), R3(X), W2(Y), W1(X), W3(X)
  - R1(X), R1(Y), R3(X), R2(Y), W2(Y), W1(X), W3(X)

BÀI TẬP

- Cho đồ thị trình tự của của lịch trình S:



- Lịch trình nào tương đương với S:
  - T1 → T2 → T3 → T4
  - T2 → T1 → T3 → T4
  - T4 → T2 → T3 → T1
  - T4 → T3 → T1 → T2

TƯƠNG ĐƯƠNG VIEW

- 2 lịch trình S và S' được gọi là tương đương view (view equivalent) nếu thỏa:
  - Có cùng tập hợp các giao dịch.
  - Với mỗi dữ liệu X, nếu Ti thực hiện Read(X) trong S và giá trị X được sinh ra bởi giao dịch Tj thì Ti cũng phải đọc giá trị X được sinh ra bởi Tj trong S'.
  - Với mỗi dữ liệu X, giao dịch thực hiện Write(X) sau cùng trong S cũng phải thực hiện Write(X) cuối cùng trong S'.

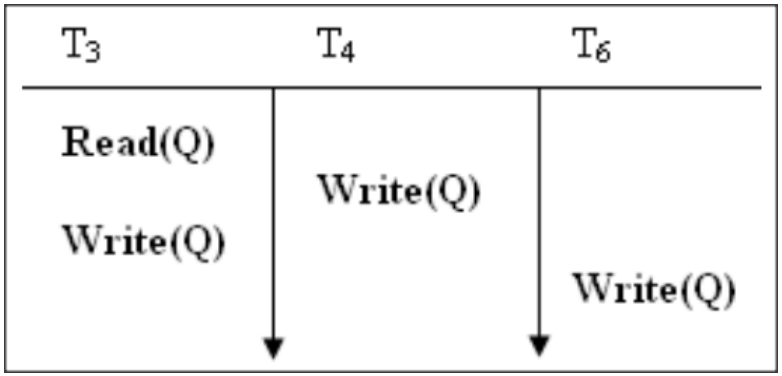
TƯƠNG ĐƯƠNG VIEW

- Các điều kiện đảm bảo:
  - Mỗi giao dịch đọc cùng một giá trị trong cả 2 lịch trình.
  - Cả 2 lịch trình cho ra kết quả trạng thái hệ thống như nhau.

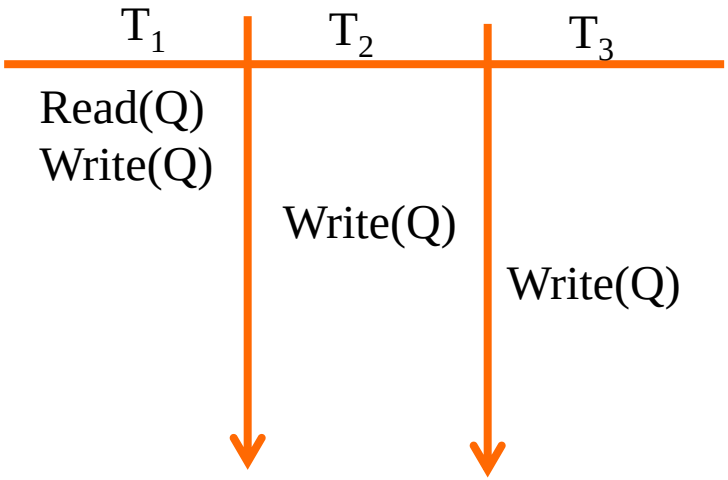
- Ngoài ra còn 1 số loại tương đương khác do các ứng dụng khác nhau đưa ra.

KHẢ TUẦN TỰ VIEW

- Khả tuần tự view (view serializable): 1 lịch trình gọi là khả tuần tự view nếu nó tương đương view với 1 lịch trình tuần tự.



Lịch trình S



KHẢ TUẦN TỰ VIEW

- Lịch trình khả tuần tự xung đột là khả tuần tự view.
- Lịch trình khả tuần tự view có thể **không** khả tuần tự xung đột → Do các write mù (blind write).
- Write mù: thực hiện hoạt động write mà không thực hiện read.

GIAO DỊCH TRONG SQL

- Chuẩn SQL đặc tả sự bắt đầu của giao dịch một cách không tường minh.
- Giao dịch được kết thúc bởi một trong 2 lệnh:
  - Commit tran
  - Rollback tran
- Cho phép định nghĩa giao dịch được thực hiện không khả tuần tự.

CÁC DẠNG READ TRONG SQL

- **Dirty read**: Giao dịch T1 được phép đọc những thay đổi của T2 chưa được committed. Nếu T2 bị hủy bỏ, giá trị đọc được của T1 không tồn tại và không chính xác.

- **Non-repeatable read**: Giao dịch T1 đọc 1 giá trị từ bảng, nếu có giao dịch T2 thay đổi giá trị đó và T1 đọc lại, T1 sẽ thấy giá trị được cập nhật.

CÁC DẠNG READ TRONG SQL

- **Phantoms**: Giao dịch T1 đọc 1 số dòng trong bảng dựa trên 1 số điều kiện Where nào đó. Khi đó 1 giao dịch T2 khác thêm 1 dòng mới thỏa mãn điều kiện Where của T1 vào bảng đó. Nếu T1 đọc lại giá trị của bảng sẽ thấy 1 bóng ma (phantom) – 1 dòng mới được thêm vào, trước đó chưa xuất hiện.

CÁC MỨC NHẤT QUÁN TRONG SQL

- Serializable: mặc định, với nghĩa không cho phép:
  - Dirty read
  - Non-repeatable read
  - Phantoms
- Repeatable read: cho phép Phantoms
- Read committed: chỉ không cho phép Dirty read.
- Read uncommitted: cho phép tất cả.

CÁC MỨC NHẤT QUÁN TRONG SQL

Isolation level	Dirty Reads	Non-repeatable reads	Phantom reads	Concurrency control
READ UNCOMMITTED	Yes	Yes	Yes	Pessimistic
READ COMMITTED (with locking)	No	Yes	Yes	Pessimistic
READ COMMITTED (with snapshot)	No	Yes	Yes	Optimistic
REPEATABLE READ	No	No	Yes	Pessimistic
SNAPSHOT	No	No	No	Optimistic
SERIALIZABLE	No	No	No	Pessimistic

CÚ PHÁP SQL

- BEGIN { TRAN | TRANSACTION }
- [{ *transaction_name* | *@tran_name_variable* }
- [WITH MARK [*'description'*]]
-]
- [;]

- WITH MARK: chỉ định giao tác sẽ được đánh dấu trong log

- VD:
  - ***BEGIN TRAN T1***

Cú pháp SQL

- BEGIN TRAN T2
- SELECT * FROM NHANVIEN
- COMMIT TRAN

- Lệnh xem thông tin về thiết lập hiện tại trên SQL:

DBCC USEROPTIONS

READ UNCOMMITED

- Cho phép đọc cả dữ liệu đang trong xử lý ở một giao tác chưa hoàn tất khác (uncommited)
- Đây là mức thấp nhất, cho phép cả 3 kiểu đọc dữ liệu.
- Cú pháp SQL:
- SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

READ UNCOMMITED

- BEGIN TRAN T1
- INSERT INTO NHANVIEN VALUES ('NV10','A','B','C','1999-1-22','SU VAN HANH','NAM',88888,'P1')
- WAITFOR DELAY '00:00:10'
- ROLLBACK TRAN T1

- BEGIN TRAN T2
- SELECT * FROM NHANVIEN
- COMMIT TRAN

READ COMMITED

- Mặc định của SQL Server.
- Chỉ đọc các dữ liệu đã xử lý hoàn tất (commited)
- Do cơ chế lock của SQL Server.
- Cú pháp SQL:
- SET TRANSACTION ISOLATION LEVEL READ COMMITTED

REPEATABLE READ

- Tình huống trong khi đang truy xuất dữ liệu lại có sự cập nhật dữ liệu đó ở một giao tác khác dẫn đến sự không nhất quán về dữ liệu.
- Do đó cần đến REPEATABLE READ
- Cú pháp:
- SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

REPEATABLE READ

- SET TRANSACTION ISOLATION LEVEL READ COMMITTED
- SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
- GO
- DBCC USEROPTIONS
- BEGIN TRAN REPEATREAD
- SELECT * FROM NHANVIEN
- WAITFOR DELAY '00:00:05'
- SELECT * FROM NHANVIEN
- COMMIT TRAN REPEATREAD

- BEGIN TRAN SHORTUPDATE
- UPDATE NHANVIEN SET HONV='A' WHERE MANV='NV10'
- COMMIT TRAN SHORTUPDATE
- INSERT INTO NHANVIEN VALUES ('NV10','A','B','C','1999-1-22','SU VAN HANH','NAM',88888,'P1')
- DELETE FROM NHANVIEN WHERE MANV='NV10'

- SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
- SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
- SET TRANSACTION ISOLATION LEVEL READ COMMITTED
- SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

HẾT!

Tham khảo: Chương 21

Fundamentals of Database Systems