Quản lý giao tác

(Transaction Management)

Nội dung

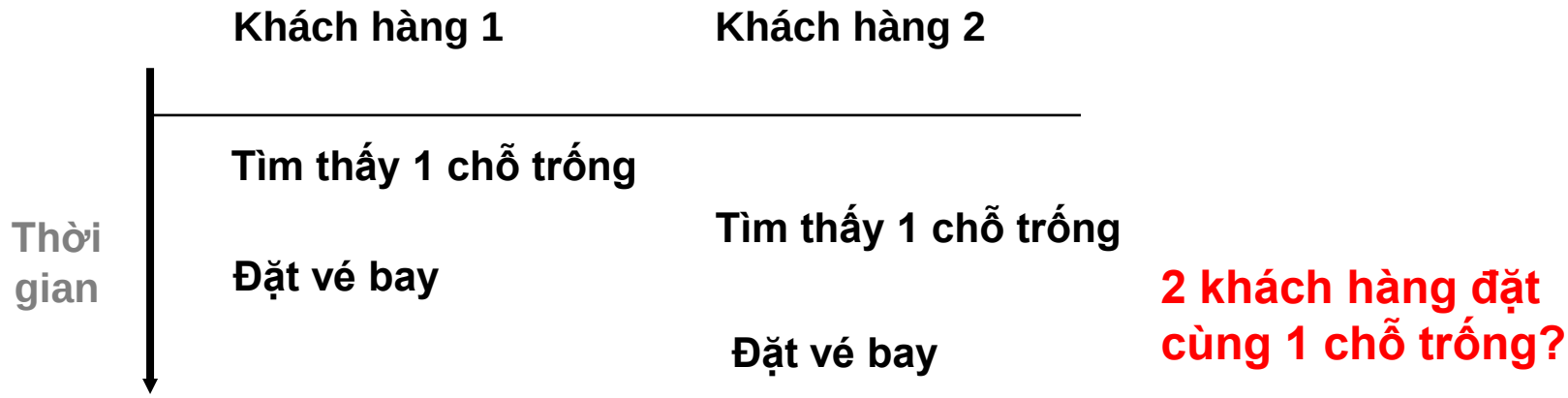
- Giới thiệu
- Giao tác
- Tính chất ACID của giao tác
- Các thao tác của giao tác
- Trạng thái của giao tác

Nội dung

- Giới thiệu
- Giao tác
- Tính chất ACID của giao tác
- Các thao tác của giao tác
- Trạng thái của giao tác

Giới thiệu

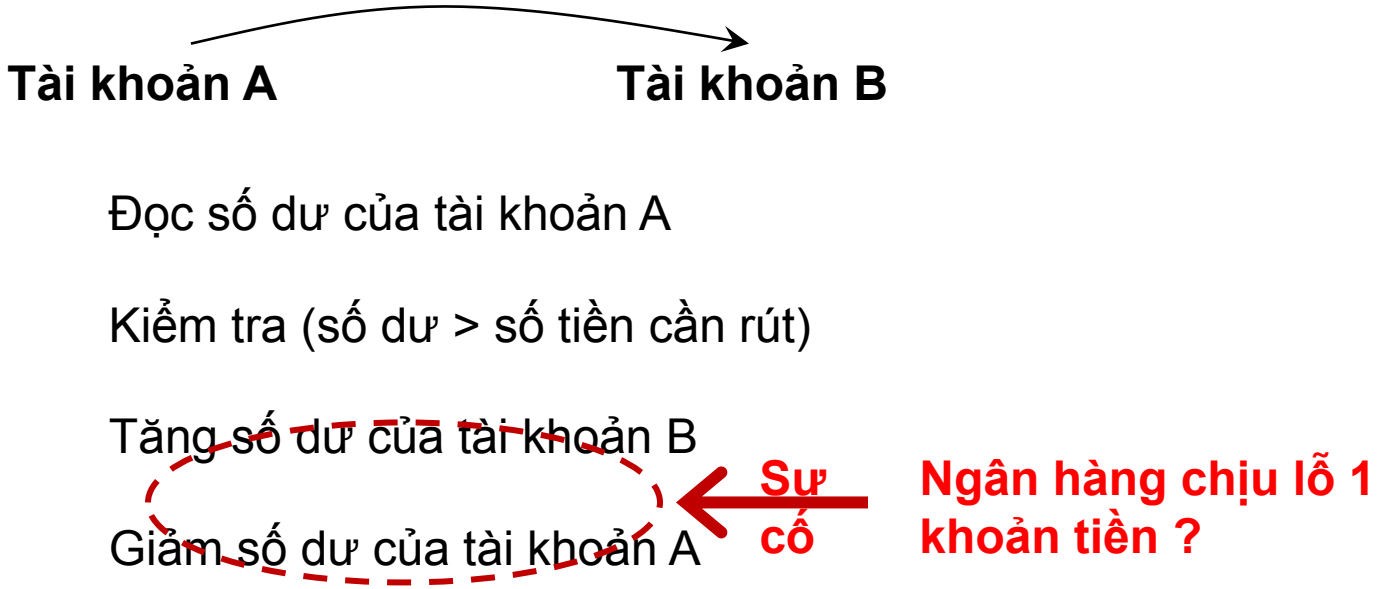
- DBMS là môi trường đa người dùng
  - Nhiều thao tác truy xuất lên cùng 1 đơn vị dữ liệu
  - Nhiều thao tác thi hành đồng thời
- Ví dụ: Hệ thống đặt vé bay



Cơ chế tuần tự

Giới thiệu

- Khi DBMS gặp sự cố, các thao tác có thể làm cho trạng thái CSDL không chính xác
- Ví dụ: Hệ thống giao dịch ngân hàng



Nguyên tố

Nội dung

- Giới thiệu
- Giao tác
- Tính chất ACID của giao tác
- Các thao tác của giao tác
- Trạng thái của giao tác

Giao tác (Transaction)

- Giải pháp cho vấn đề tuần tự (serial) và nguyên tố (atomic) là gom các nhóm thao tác phải thực hiện với nhau trong cùng 1 giao tác.
- **Định nghĩa**: Giao tác là một dãy các thao tác cần thực hiện trên cơ sở dữ liệu dưới một đơn vị duy nhất
  - hoặc tất cả các thao tác được thực hiện
  - hoặc không thực hiện thao tác nào cả

Giao tác

- Ví dụ: giao tác chuyển khoản từ A → B gồm 2 thao tác
  - Trừ tiền A
  - Cộng tiền B
- Chuyển khoản được thực hiện dưới dạng giao tác, nghĩa là
  - hoặc thực hiện cả 2 thao tác trừ tiền A và cộng tiền B (giao tác thành công)
  - hoặc nếu có sự cố thì không thực hiện thao tác nào cả (giao tác thất bại)

Nội dung

- Giới thiệu
- Giao tác
- Tính chất ACID của giao tác
- Các thao tác của giao tác
- Trạng thái của giao tác

Tính chất của giao tác

- Để đảm bảo tính toàn vẹn của dữ liệu, ta yêu cầu hệ CSDL duy trì các tính chất sau của giao tác:
  - Nguyên tố (**A**tomicity)
  - Nhất quán (**C**onsistency)
  - Cô lập (**I**solation)
  - Bền vững (**D**urability)

ACID

Tính chất ACID của giao tác

- Nguyên tố (Atomicity)
  - Hoặc là toàn bộ hoạt động của giao dịch được phản ánh đúng đắn trong CSDL hoặc không có hoạt động nào cả.
  - Đảm bảo bởi thành phần quản lý giao tác
- Nhất quán (Consistency)
  - Một giao tác được thực hiện độc lập với các giao tác khác xử lý đồng thời với nó để bảo đảm tính nhất quán cho CSDL.
  - Đảm bảo bởi người lập trình ứng dụng hay người viết ra giao tác

Tính chất ACID của giao tác

- Cô lập (Isolation)
  - Một giao tác không cần quan tâm đến các giao tác khác đang thực hiện đồng thời trong hệ thống.
  - Đảm bảo bởi thành phần quản lý truy xuất đồng thời
- Tính bền vững (Durability)
  - Mọi thay đổi mà giao tác thực hiện trên CSDL phải được ghi nhận bền vững.
  - Đảm bảo bởi thành phần quản lý phục hồi

Tính chất ACID của giao tác

- Ví dụ: T là một giao dịch chuyển 50$ từ tài khoản A sang tài khoản B.
  - Giao dịch này có thể được xác định như sau:

```
T: Read(A,t);
   t:=t-50;
   Write(A,t);
   Read(B,t);
   t:=t+50;
   Write(B,t);
```

Nội dung

- Giới thiệu
- Giao tác
- Tính chất ACID của giao tác
- Các thao tác của giao tác
- Trạng thái của giao tác

Các thao tác của giao tác

- Giả sử CSDL gồm nhiều đơn vị dữ liệu
- Một đơn vị dữ liệu:
  - Có một giá trị
  - Được truy xuất và sửa đổi bởi các giao tác

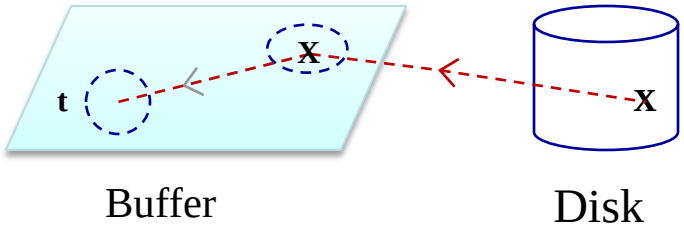
Các thao tác của giao tác

Các truy xuất CSDL được thực hiện bởi hai hoạt động sau:

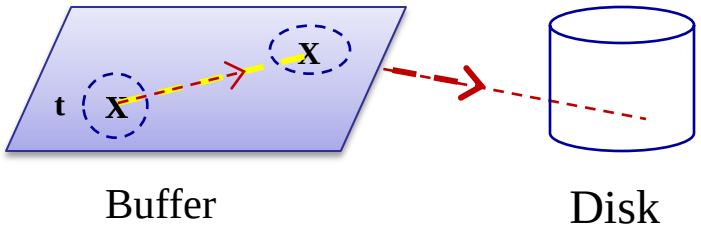
- **READ(X)**
  - chuyển hạng mục dữ liệu X từ CSDL đến buffer của giao dịch thực hiện hoạt động READ này
- **WRITE(X)**
  - chuyển hạng mục dữ liệu X từ buffer của giao dịch thực hiện WRITE đến CSDL

Các thao tác của giao tác

- Input(X)
- Read(X, t)



- Write(X, t)
- Output(X)



- Bufffer manager
  - Input
  - Output
- Transaction
  - Read
  - Write

Ví dụ

- Giả sử CSDL có 2 đơn vị dữ liệu A và B với ràng buộc A=B trong mọi trạng thái nhất quán
- Giao tác T thực hiện 2 bước
  - A:=A*2
  - B:=B*2
- Biểu diễn T
  - Read(A,t) ; t=t*2; Write(A,t);
  - Read(B,t) ; t=t*2; Write(B,t);

Ví dụ

Hành động	t	Mem A	Mem B	Disk A	Disk B
Read(A,t)	8	8		8	8
t:=t*2	16	8		8	8
Write(A,t)	16	16		8	8
Read(B,t)	8	16	8	8	8
t:=t*2	16	16	8	8	8
Write(B,t)	16	16	16	8	8
Output(A)	16	16	16	16	8
Output(B)	16	16	16	16	16

Nội dung

- Giới thiệu
- Giao tác
- Tính chất ACID của giao tác
- Các thao tác của giao tác
- Trạng thái của giao tác

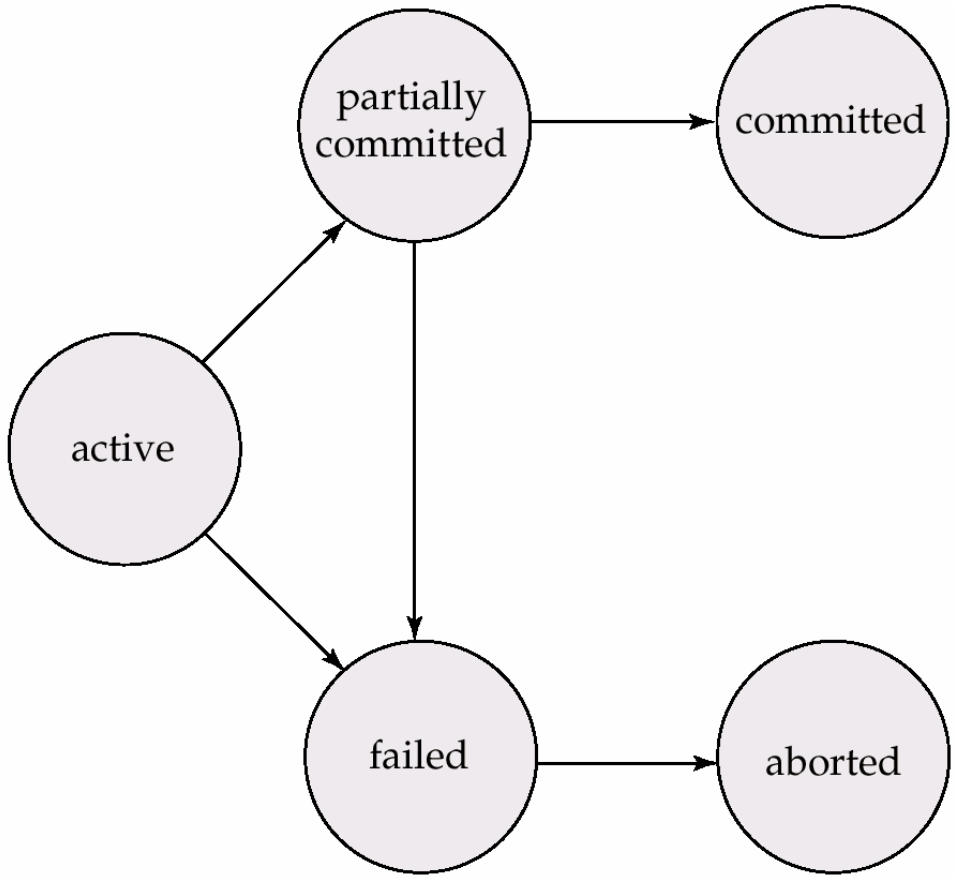
Các trạng thái của giao tác

Một giao tác phải ở trong một trong các trạng thái sau:

- **Hoạt động (Active)**
  - Ngay khi bắt đầu thực hiện thao tác đọc/ghi
- **Được bàn giao bộ phận (Partially committed)**
  - Sau khi lệnh thi hành cuối cùng được thực hiện
- **Thất bại (Failed)**
  - Sau khi phát hiện ra sự thực hiện không thể tiếp tục được nữa
- **Bỏ dở (Aborted)**
  - Sau khi giao tác được quay lui và CSDL được phục hồi về trạng thái trước trạng thái bắt đầu giao dịch
    - Bắt đầu lại giao tác (nếu có thể)
    - Hủy giao tác
- **Được bàn giao (Committed)**
  - Sau khi mọi hành động hoàn tất thành công

Các trạng thái của giao tác

- Sơ đồ trạng thái



Quản lý truy xuất đồng thời

Nội dung

- Giới thiệu
- Lịch thao tác (schedule)
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)
    - Conflict-Serializable
    - View-Serializable

Nội dung

- Giới thiệu
- Lịch thao tác (schedule)
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)

Giới thiệu

- Thực hiện tuần tự
  - Tại một thời điểm, một giao tác chỉ có thể bắt đầu khi giao tác trước nó hoàn tất
- Thực hiện đồng thời
  - Cho phép nhiều giao tác cùng truy xuất dữ liệu
  - Gây ra nhiều phức tạp về nhất quán dữ liệu

Lý do thực hiện đồng thời

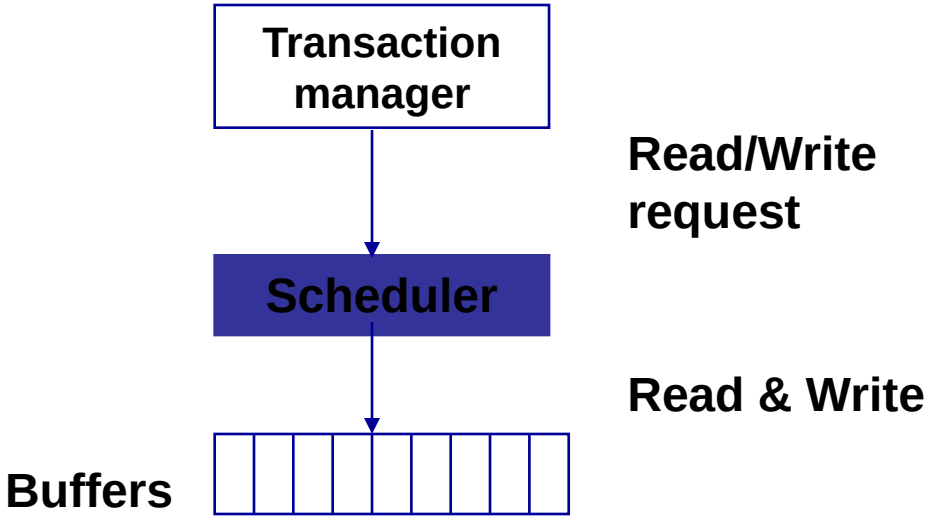
- Tận dụng tài nguyên và thông lượng
  - Trong khi 1 giao tác đang thực hiện đọc/ghi trên đĩa, 1 giao tác khác đang xử lý tính toán trên CPU
- Giảm thời gian chờ
  - Các giao tác ngắn phải chờ đợi các giao tác dài
  - Chia sẻ chu kỳ CPU và truy cập đĩa để làm giảm sự trì hoãn trong khi các giao tác thực thi

Giới thiệu

- Khi có nhiều giao tác thực hiện đồng thời, tính nhất quán CSDL có thể bị phá vỡ mặc dù cá nhân mỗi giao tác vẫn thực hiện đúng đắn.
- Vì vậy, cần có **lịch thao tác** (schedule) để xác định chuỗi các thao tác của nhiều giao tác cạnh tranh mà vẫn đảm bảo tính nhất quán.
- Bộ phận quản lý các lịch thao tác này gọi là **Bộ lập lịch** (scheduler).

Bộ lập lịch (scheduler)

- Là một thành phần của DBMS, có nhiệm vụ lập lịch để thực hiện nhiều giao tác xử lý đồng thời



Nội dung

- Giới thiệu
- Lịch thao tác (schedule)
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)

Lịch thao tác (schedule)

- Lịch **S** của **n** giao tác $\mathbf{T_1}$, $\mathbf{T_2}$, …, $\mathbf{T_n}$ là *dãy có thứ tự các thao tác* trong n giao tác này
- Thứ tự xuất hiện của các thao tác trong lịch phải giống với thứ tự xuất hiện trong giao tác
- Gồm có:
  - Lịch tuần tự (Serial)
  - Lịch khả tuần tự (Serializable)
    - Conflict-Serializability
    - View-Serializability

Ví dụ

- Cho lịch S của 2 giao tác T1 và T2 như sau:

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

- Lịch S có thể được viết lại:

  S: R1(X), R2(X), W1(X), R1(Y),W2(X), W1(Y)

- Trong S chỉ quan tâm 2 hoạt động cơ sở là R và W

Ví dụ

- Giả sử T1 và T2 là hai giao dịch chuyển khoản từ một tài khoản sang một tài khoản khác.
  - Giao dịch T1 chuyển 50$ từ tài khoản A sang tài khoản B.
  - Giao dịch T2 chuyển 10% số dư từ tài khoản A sang tài khoản B

T1:	T2:
Read(A); A:=A-50; Write(A); Read(B); B:=B+50; Write(B);	Read(A); Temp:=A*0.1; A:=A-temp; Write(A); Read(B); B:=B+temp; Write(B);

Giả sử giá trị hiện tại của A và B tương ứng là 1000$ và 2000$

Ví dụ

- Trường hợp 1: thực hiện xong giao dịch T1 rồi đến giao dịch T2

T1	T2
Read(A); A:=A-50; Write(A); Read(B); B:=B+50; Write(B);	
	Read(A); Temp:=A*0.1; A:=A-temp; Write(A); Read(B); B:=B+temp; Write(B);

- Giá trị sau cùng của
  - A là 855
  - B là 2145
- Tổng 2 tài khoản (A+B) là không đổi

Ví dụ

- Trường hợp 2: thực hiện xong giao dịch T2 rồi đến giao dịch T1

T1	T2
	Read(A); Temp:=A*0.1; A:=A-temp; Write(A); Read(B); B:=B+temp; Write(B);
Read(A); A:=A-50; Write(A); Read(B); B:=B+50; Write(B);	

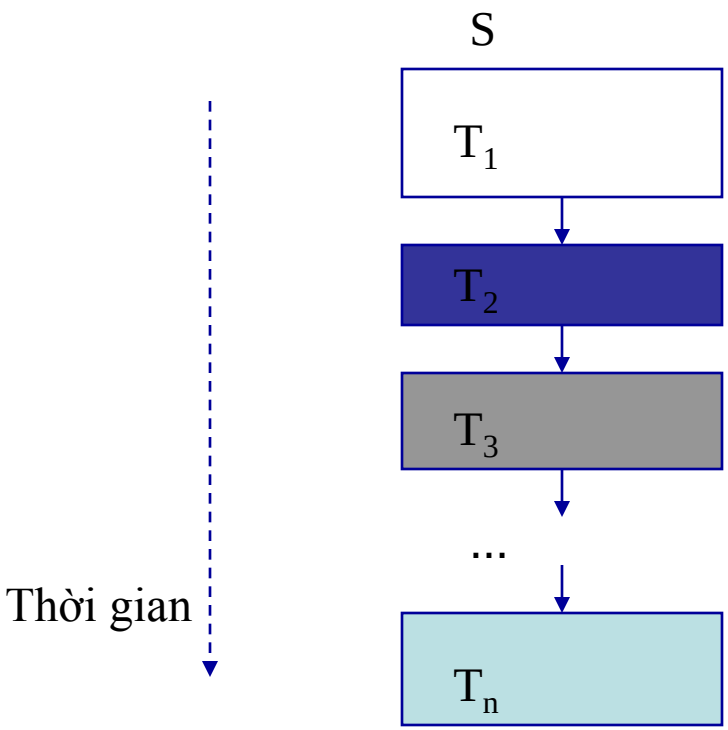
- Giá trị sau cùng của
  - A là 850
  - B là 2150
- Tổng 2 tài khoản (A+B) là không đổi

Nội dung

- Giới thiệu
- Lịch thao tác (schedule)
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)

Lịch tuần tự (Serial schedule)

- Một lịch S được gọi là tuần tự nếu các hành động của các giao tác Ti (i=1..n) được thực hiện liên tiếp nhau



Lịch tuần tự

- Ví dụ

T_1	T_2
Read(A,t)	Read(A,s)
t:=t+100	s:=s*2
Write(A,t)	Write(A,s)
Read(B,t)	Read(B,s)
t:=t+100	s:=s*2
Write(B,t)	Write(B,s)

- Giả sử ràng buộc nhất quán trên CSDL là **A=B**
- Từng giao tác thực hiện riêng lẻ thì tính nhất quán sẽ được bảo toàn

Lịch tuần tự

- Ví dụ

S_1	T_1	T_2	A	B
			25	25
	Read(A,t)			
	t:=t+100			
	Write(A,t)		125	
	Read(B,t)			
	t:=t+100			
	Write(B,t)			125
		Read(A,s)		
		s:=s*2		
		Write(A,s)	**250**	
		Read(B,s)		
		s:=s*2		
		Write(B,s)		**250**

Lịch tuần tự

- Ví dụ

S_2

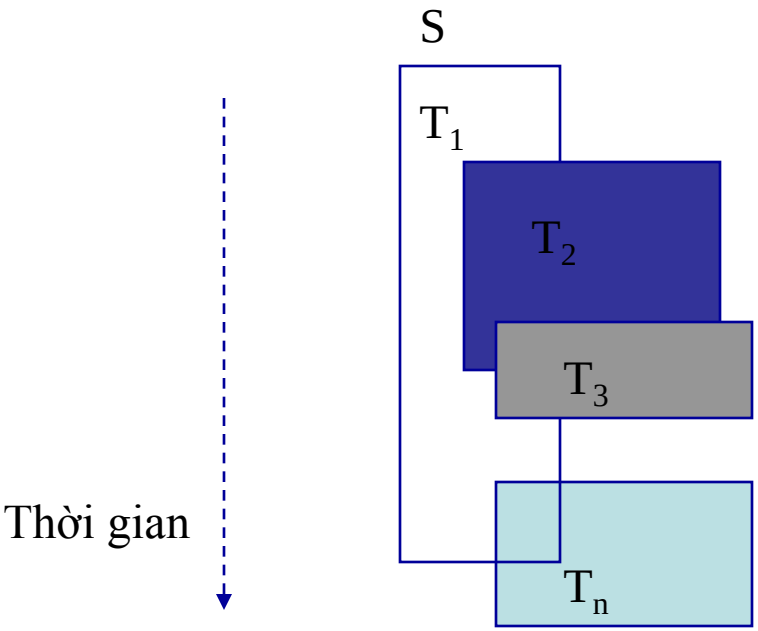
T_1	T_2	A	B
		25	25
	Read(A,s)		
	s:=s*2		
	Write(A,s)	50	
	Read(B,s)		
	s:=s*2		
	Write(B,s)		50
Read(A,t)			
t:=t+100			
Write(A,t)		**150**	
Read(B,t)			
t:=t+100			
Write(B,t)			**150**

Nội dung

- Giới thiệu
- Lịch thao tác (schedule)
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)

Lịch khả tuần tự (Serializable schedule)

- Một lịch S được lập từ n giao tác T_1, T_2, …, T_n xử lý đồng thời được gọi là khả tuần tự nếu nó cho cùng kết quả với 1 lịch tuần tự nào đó được lập từ n giao tác này



Lịch khả tuần tự

S_3

T_1	T_2	A	B
		25	25
Read(A,t)			
t:=t+100			
Write(A,t)		125	
	Read(A,s)		
	s:=s*2		
	Write(A,s)	**250**	
Read(B,t)			
t:=t+100			
Write(B,t)			125
	Read(B,s)		
	s:=s*2		
	Write(B,s)		**250**

- Trước S3 khi thực hiện
  - A=B=c (c là hằng số)
- Sau khi S3 kết thúc
  - A=2*(c+100)
  - B=2*(c+100)
- Trạng thái CSDL nhất quán

→ S3 khả tuần tự

Lịch khả tuần tự

S_4	T_1	T_2	A	B
			25	25
	Read(A,t)			
	t:=t+100			
	Write(A,t)		125	
		Read(A,s)		
		s:=s*2		
		Write(A,s)	**250**	
		Read(B,s)		
		s:=s*2		
		Write(B,s)		50
	Read(B,t)			
	t:=t+100			
	Write(B,t)			**150**

- Trước S4 khi thực hiện
  - A=B=c (c là hằng số)
- Sau khi S4 kết thúc
  - A = 2*(c+100)
  - B = 2*c + 100
- Trạng thái CSDL không nhất quán

→ S4 không khả tuần tự

Nhận xét

- Các lịch trình **tuần tự** kém hiệu quả do không cho phép giao tác thi hành xen kẻ
- Các lịch trình **không khả tuần tự** tiềm tàng khả năng gây các vấn đề bất thường
- Các lịch trình **khả tuần tự** cho phép giao tác thi hành xen kẻ và cho kết quả đúng

Nhận xét

Tuy nhiên, việc kiểm tra tính khả tuần tự của một lịch trình thực tế rất khó khăn, do các nguyên nhân:

- Các giao tác, khi thi hành thường thể hiện dưới dạng những tiến trình của HĐH, thường được bản thân HĐH điều phối. HQTCSDL thực tế không điều phối được các thao tác trong giao tác.
- Các giao tác được gửi tới HQTCSDL liên tục, khó xác định điểm bắt đầu và kết thúc giao tác.

→ Các HQTCSDL thường sử dụng những quy tắc để đảm bảo tính khả tuần tự của lịch trình.

Câu hỏi

- Xét 2 lịch sau có tương đương về mặt kết quả không?
- Giả sử X = 20, Y = 30

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

T1	T2
R1(X)	
X = X-10	
W1(X)	
	R2(X)
	X = X+5
	W2(X)
R1(Y)	
Y=Y-15	
W1(Y)	

HƯỚNG DẪN THỰC HÀNH

Giao tác

- Một giao tác thường là kết quả của việc thực hiện một chương trình người dùng được viết trong một ngôn ngữ cấp cao hay ngôn ngữ SQL và được phân cách bởi các câu lệnh có dạng:

begin transaction

…

end transaction

Phân loại

- Giao tác ngầm định (Implicit transaction)
- Giao tác tường minh (Explicit transaction)
- Giao tác xác nhận (Commit transaction)

Giao tác tường minh (Explicit)

- Giao tác

  Begin tran [tên_giao_tác]

  lệnh | khối_lệnh

  { Commit tran | Rollback tran } [tên_giao_tác]

- Tạo điểm lưu

  save tran tên_điểm_lưu

  Hủy những gì sau điểm lưu nếu rollback.

Điểm lưu (save point)

```
begin tran t1
        lệnh | khối_lệnh
        save tran s1
                lệnh | khối_lệnh
        rollback tran s1                => chưa chấm dứt t1
        lệnh | khối_lệnh
commit tran t1
```

=> Rollback tran s1 chỉ hủy bỏ kết quả sau lệnh save tran s1 và tiếp tục làm tiếp (giao tác t1 vẫn còn).

GT không tường minh (implicit)

- Bắt đầu giao tác với các lệnh

  ALTER TABLE, DROP, TRUNCATE TABLE,

  CREATE, OPEN, FETCH, REVOKE, GRANT

  DELETE, INSERT, SELECT, UPDATE

- Kết thúc bằng lệnh : commit | rollback tran
- Khi kết thúc cũng là lúc bắt đầu một giao tác mới.
- Thiết lập thông số chấp nhận

  SET IMPLICIT_TRANSACTIONS ON|OFF

Giao tác tự động (autocommit)

- Cơ chế tự động xác nhận được thực thi khi trong giao tác xuất hiện lỗi lúc chạy hay lỗi cú pháp.
  - Lỗi cú pháp == giao tác bị hủy (rollback)
  - Lỗi lúc chạy (khóa chính, sai dữ liệu,…)

    == giao tác được chấp nhận đến thời điểm đó.

Giao tác lồng nhau

- Cho phép các giao tác lồng với nhau.
- Lệnh commit chỉ có tác dụng cho giao tác cấp ‘con’ gần nhất.
- Lệnh rollback tran có tác dụng hủy tất cả và trở về điểm ban đầu của giao tác cấp ‘cha’ nhất.
- Biến @@trancount chỉ xem vào thời điểm hiện tại có bao nhiêu giao tác đang tồn tại.

Giao tác lồng nhau

Begin tran t1

……

begin tran t2

…….

print @@trancount => kết quả là 2

commit tran t2

……..

print @@trancount => kết quả là 1

commit tran t1