


Hệ quản trị Cơ sở dữ liệu

Chương 4 (tt):

Những vấn đề khác trong

điều khiển đồng thời

1



Nội dung chi tiết

☐ Quay lui dây chuyền (cascading rollback)


☐ Lịch khả phục hồi (recoverable schedule)

☐ Deadlock

– Phát hiện (detection)

– Ngăn ngừa (prevention)


2



Ví dụ

	T ₁	T ₂	A	B
S	Lock(A); Read(A,t)		25	25
	t:=t+100; Write(A,t)		125	
	Lock(B); Unlock(A)			
		Lock(A); Read(A,s)		
		s:=s*2; Write(A,s)	250	
		Lock(B)		
		Chờ		
	Read(B,t); t:=t+100			125
	Write(B,t); Unlock(B)	Lock(B); Ulock(A)		
		Read(B,t); t:=t*2		
	Write(B,t); Unlock(B)		250	

3




Ví dụ (tt)

	T ₁	T ₂	A	B
S	Lock(A); Read(A,t)		25	25
	t:=t+100; Write(A,t)		125	
	Lock(B); Unlock(A)			
		Lock(A); Read(A,s)		
		s:=s*2; Write(A,s)	250	
		Lock(B)		
		Chờ		
	Read(B,t);	Lock(B); Ulock(A)		
	Abort; Unlock(B);	Read(B,t); t:=t*2		
		Write(B,t); Unlock(B)		50

Tính nhất quán bị vi phạm
→ T₂ cũng phải rollback

4




Ví dụ (tt)

	T ₁	T ₂	T ₃	A RT=0 WT=0	B RT=0 WT=0	C RT=0 WT=0
S	200	150	175			
		Write(B)			RT=0 WT=150	
	Read(B)				RT=200 WT=150	
		Read(A)		RT=150 WT=0		
			Read(C)			RT=175 WT=0
		Write(C) Abort			RT=0 WT=0	
			Write(A)	RT=150 WT=175		

→ Phục hồi giá trị của B


5



Quay lui dây chuyền

	T ₁	T ₂	T ₃	T ₄
	⋮	⋮	⋮	⋮
	w(A)	⋮	⋮	⋮
	⋮	r(A)	⋮	⋮
	⋮	⋮	r(A)	⋮
	⋮	⋮	⋮	⋮
	abort	⋮	⋮	⋮
		abort	⋮	⋮
			abort	⋮

6



Lịch khả phục hồi

T_i

⋮

w(A)

⋮

abort

T_j


⋮

r(A)

commit

- Xét mỗi cặp T_i và T_j sao cho
 - T_j đọc dữ liệu sau khi T_i ghi
 - T_i phải được hoàn tất (commit) trước khi T_j hoàn tất
 - Ký hiệu c_i: giao tác thứ i hoàn tất

7



Lịch khả phục hồi (tt)

S₁ : w₁(A); w₁(B); w₂(A); r₂(B); c₁; c₂;


↙ ↘

T₂ đọc B sau
T₁ ghi B

T₁ hoàn tất
trước T₂

S₁ tuần tự và khả phục hồi

8



Lịch khả phục hồi (tt)

S₂ : w₂(A); w₁(B); w₁(A); r₂(B); c₁; c₂;

↙ ↘


T₁ ghi B trước
T₂ đọc B sau

T₁ hoàn tất
trước T₂

S₂ không khả tuần tự nhưng khả phục hồi

9

3



Lịch khả phục hồi (tt)


$S_3 : w_1(A); w_1(B); w_2(A); r_2(B); c_2; c_1;$

T_1 ghi B trước
 T_2 đọc B sau

T_2 hoàn tất trước T_1

S_3 khả tuần tự nhưng không khả phục hồi

10



Lịch khả phục hồi (tt)

☐ Nhận xét


– Muốn khôi phục đôi khi cần quay lui dây chuyền

– Nhưng quay lui dây chuyền không thể xảy ra

- Tồn nhiều chi phí

→ Lịch không quay lui dây chuyền (cascadeless schedule)

11




Lịch không quay lui dây chuyền

T_i	T_j
⋮	⋮
$w(A)$	⋮
commit	⋮
	$r(A)$

Các giao tác chỉ đọc những giá trị đã được hoàn tất

12



is

Lịch không quay lui dây chuyền (tt)

☐ Khả phục hồi


$S_1 : w_1(A); w_1(B); w_2(A); r_2(B); c_1; c_2;$

☐ Ngăn ngừa quay lui dây chuyền

$S_1 : w_1(A); w_1(B); w_2(A); c_1; r_2(B); c_2;$

☐ → Các lịch ngăn ngừa quay lui dây chuyền đều khả phục hồi

13



is

Nội dung chi tiết

☐ Quay lui dây chuyền (cascading rollback)


☐ Lịch khả phục hồi (recoverable schedule)

☐ Deadlock

– Phát hiện (detection)

– Ngăn ngừa (prevention)

14



is

Deadlock

☐ Nhắc lại 2 tình huống

S

T₁

T₂

Lock(A)

Read(A)

Write(A)

Lock(B)

↓

Chờ

Lock(B)

Read(B)

Write(B)

Lock(A)

↓

Chờ

Quí tắc khóa 2PL

S

T₁

T₂

RLock(A)

Read(A)

WLock(A)

↓

Chờ

RLock(A)

Read(A)

WLock(A)


↓

Chờ

Nâng cấp khóa

15

5




is

Deadlock (tt)

❑ Hệ thống rơi vào trạng thái deadlock khi

- Các giao tác phải chờ đợi lẫn nhau để được thao tác lên các đơn vị dữ liệu bị khóa bởi chúng
- Và không một giao tác nào có thể thực hiện tiếp công việc của mình

16



is

Giải quyết Deadlock


❑ Phát hiện

- Cho phép trạng thái deadlock xảy ra và sau đó cố gắng khôi phục lại hệ thống
 - Chọn 1 giao tác để rollback
- Phương pháp
 - Đồ thị chờ (wait-for graph)

❑ Ngăn ngừa

- Quản lý các giao tác sao cho không bao giờ có deadlock
- Phương pháp
 - Sắp thứ tự tài nguyên (resource ordering)
 - Timeout
 - Wait-die
 - Wound-wait

17



is

Đồ thị chờ

❑ Đồ thị gồm

- Đỉnh là các giao tác đang giữ khóa hoặc đang chờ khóa
- Cung đi từ đỉnh T sang U khi
 - U đang giữ khóa trên đơn vị dữ liệu A
 - T đang chờ khóa trên A
 - T không thể khóa đơn vị dữ liệu A nếu U không giải phóng khóa


❑ Nếu đồ thị chờ không có chu trình

- Các giao tác có thể hoàn tất

❑ Ngược lại

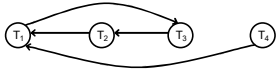
- Không một giao tác nào trong chu trình có thể tiếp tục thực hiện → deadlock

18




Ví dụ

	T ₁	T ₂	T ₃	T ₄
1	L(A); R(A)			
2		L(C); R(C)		
3			L(B); R(B)	
4				L(D); R(D)
5		L(A)		
6		↓ Chờ	L(C)	
7			↓ Chờ	L(A)
8	L(B) ↓ Chờ			↓ Chờ

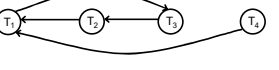


19




Ví dụ (tt)

	T ₁	T ₂	T ₃	T ₄
1	L(A); R(A)			
2		L(C); R(C)		
3			L(B); R(B)	
4				L(D); R(D)
5		L(A)		
6			L(C)	
7				L(A)
8	L(B)			



20




Sắp thứ tự tài nguyên

☐ Áp đặt một thứ tự nào đó lên các đơn vị dữ liệu
☐ Nếu các giao tác thực hiện khóa những đơn vị dữ liệu theo thứ tự này
☐ Thì không có deadlock xảy ra trong khi chờ đợi

☐ Chứng minh

- Bài tập về nhà

21



Ví dụ

Giả sử các đơn vị dữ liệu được sắp thứ tự theo alphabet


$T_1:$
 $l(A); r(A); l(B); w(B); u(A); u(B);$

$T_2:$
 $l(C); r(C); l(A); w(A); u(C); u(A);$

$T_3:$
 $l(B); r(B); l(C); w(C); u(B); u(C);$

$T_4:$
 $l(D); r(D); l(A); w(A); u(D); u(A);$

22



Ví dụ (tt)


$T_1:$
 $l(A); r(A); l(B); w(B); u(A); u(B);$

$T_2:$
 $l(A); l(C); r(C); w(A); u(C); u(A);$

$T_3:$
 $l(B); r(B); l(C); w(C); u(B); u(C);$

$T_4:$
 $l(A); l(D); r(D); w(A); u(D); u(A);$

23




Ví dụ (tt)

	T_1	T_2	T_3	T_4
1	$l(A); r(A)$			
2		$l(A)$		
3		↓	$l(B); r(B)$	
4		Chờ		$l(A)$
5			$l(C); w(C)$	↓
6			$u(B); u(C)$	Chờ
7	$l(B); r(B)$			
8	$u(A); u(B)$			
9		$l(A); l(C)$		
10		$r(A); w(C)$		
11		$u(C); u(A)$		
12				$l(A); l(D)$
13				$r(D); w(A)$
14				$u(D); u(A)$

24

8




IS
INFORMATION SYSTEMS

Timeout

- ☐ Giới hạn các giao tác chỉ được thực hiện trong 1 khoảng thời gian nào đó
- ☐ Nếu giao tác vượt quá thời gian này
- ☐ Thì giao tác phải bị rollback

25



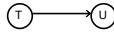
UIT
UNIVERSITY OF INFORMATION TECHNOLOGY

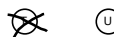
Wait-die

❑ Mỗi giao tác sẽ được gán một nhãn ghi nhận thứ tự xuất hiện, kí hiệu: $ts(T)$


❑ Xét 2 giao tác T và U

- U đang giữ khóa trên đơn vị dữ liệu A
- T muốn khóa đơn vị dữ liệu A
- T sẽ chờ-wait U khi $ts(T) < ts(U)$
- Ngược lại T sẽ bị hủy-die và bắt đầu làm lại ở 1 thời điểm khác





26




UTM
UNIVERSITI TEKNIK MALAYSIA

Ví dụ

	T_1	T_2	T_3	T_4
1	L(A); R(A)			
2		L(A) ↓		
3		Dies	L(B); R(B)	
4				L(A) ↓
5			L(C); W(C)	Dies
6			U(B); U(C)	
7	L(B); R(B)			
8	U(A); U(B)			

27



MIT
Massachusetts Institute of Technology

Ví dụ (tt)

□ T_2 bắt đầu trước T_4

	T_1	T_2	T_3	T_4
1	L(A); R(A)			
2		L(A)		
3		↓		
4		Dies		
5			L(B); R(B)	
6				
7	L(B); R(B)		L(C); W(C)	
8	U(A); U(B)		U(B); U(C)	
9		L(A); L(C)		
10				
11		R(A); W(C)		
12		U(C); U(A)		
13				
14				
15				


IS

INTEGRATED SYSTEMS

Ví dụ (tt)

□ T_4 bắt đầu trước T_2

	T_1	T_2	T_3	T_4
1	L(A); R(A)			
2		L(A)		
3		↓ Dies	L(B); R(B)	
4				L(A)
5			L(C); W(C)	↓ Dies
6			U(B); U(C)	
7	L(B); R(B)			
8	U(A); U(B)			
9				L(A); L(D)
10		L(A)		
11		↓ Waits		R(D); W(A)
12				U(D); U(A)
13		L(A); L(C)		
14		R(A); W(C)		
15		U(C); U(A)		



UIT
UNIVERSITY OF INFORMATION TECHNOLOGY

Wound-wait

☐ Mỗi giao tác sẽ được gán một nhãn ghi nhận thứ tự xuất hiện, kí hiệu: $ts(T)$


☐ Xét 2 giao tác T và U

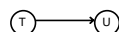
- U đang giữ khóa trên đơn vị dữ liệu A
- T muốn khóa đơn vị dữ liệu A
- T buộc U rollback và trao khóa

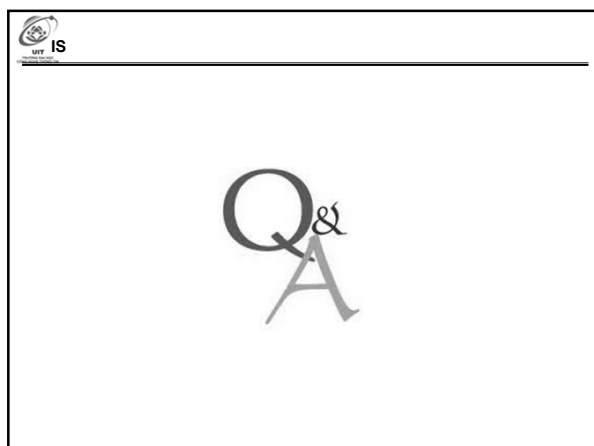
lấy cho T-wound khi $ts(T) < ts(U)$

- Ngoại lệ: nếu U đã kết thúc và giải phóng khóa, U sẽ không rollback

- Ngược lại T sẽ chờ-wait U







34
