Deadlock

Nội dung

- Một số điều cần nhắc lại
- Các sự cố và ảnh hưởng của chúng khi đang thực hiện 1 giao tác
- Quay lui dây chuyền và lịch chống quay lui dây chuyền
- Khả phục hồi và lịch khả phục hồi
- Deadlock
  - Định nghĩa
  - Phát hiện
  - Giải pháp khắc phục
  - Phòng chống
  - Ví dụ

- Một hệ quản trị CSDL phải đảm bảo các tính chất sau (ACID):
  - Atomicity
  - Consistency
  - Isolation
  - Durability

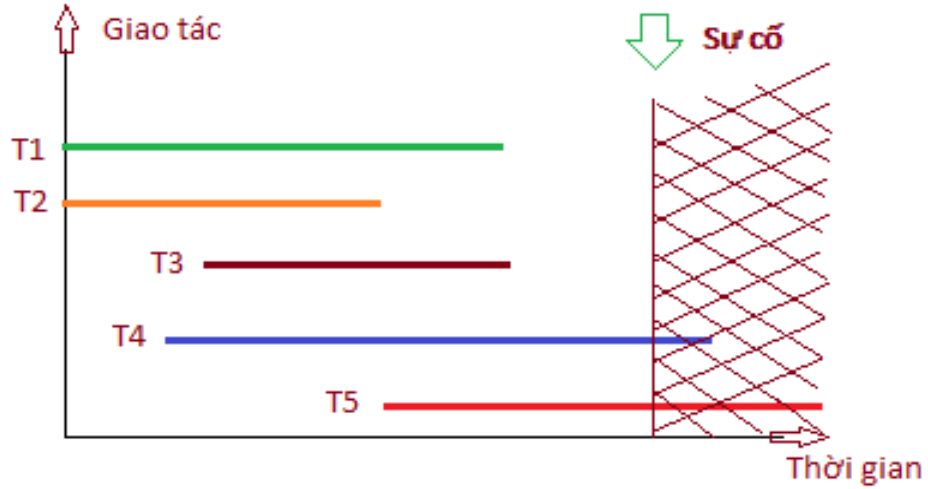
Ảnh hưởng khi có 1 sự cố xảy ra khi đang thực hiện 1 giao tác

- Các sự cố có thể xảy ra:
  - Giao tác bị hủy (abort hay rollback)
  - Hệ thống ngừng hoạt động bất chợt
- Các ảnh hưởng:
  - Giao tác bị hủy:
    Giả sử 2 thao tác thực hiện theo lịch S sau:

	T1	T2
1	R(A)	
2	W(A)	
3		R(A)
4		W(A)
5	Abort	

Xét trường hợp 2 giao tác T1 và T2 nhìn thấy nhau (giá trị của Isolation Level là Read Uncommitted) thì khi T1 bị hủy thì các thao tác của T2 xem như vô nghĩa ➔ T1 bị hủy thì T2 cũng bị hủy. Đó là hiện tượng *quay lui dây chuyền* sẽ được trình bày sau.

Ảnh hưởng khi có 1 sự cố xảy ra khi đang thực hiện 1 giao tác



- Hệ thống ngừng hoạt động bất chợt:
  Khi sự cố xảy ra, ví dụ như cúp điện, 2 giao tác T4 và T5 vẫn chưa thực hiện xong các thao tác của mình. Như vậy, các thay đổi của 2 giao tác này trước thời điểm xảy ra sự cố cần được phục hồi lại khi hệ thống khởi động lại.

- Hệ QTCSDL cần có 1 cơ chế quản lý các giao tác để phục hồi lại dữ liệu trong các trường hợp này.

Quay lui dây chuyền và lịch chống quay lui dây chuyền

- **Quay lui dây chuyền** (Cascading Abort) là trường hợp khi 1 giao tác Ti thực hiện đọc và ghi trên 1 đơn vị dữ liệu X đã được đọc và ghi bởi 1 giao tác Tj trước đó, Tj thực hiện hủy giao tác sau đó kéo theo Ti bị hủy (các thao tác ghi trên Ti là vô nghĩa).
- Để tránh trường hợp này, người ta đề ra **lịch chống quay lui dây chuyền** (Avoid Cascading Abort Schedule).
- **Nguyên lý** của lịch chống quay lui dây chuyền:
  Một giao tác Tj chỉ được đọc và ghi trên 1 đơn vị dữ liệu X, mà trước đó các giao tác thực hiện thao tác trên X đã hoàn tất (committed).

Quay lui dây chuyền và lịch chống quay lui dây chuyền

- Ví dụ: Cho lịch S như sau

TT	T1	T2
1	R(A)	
2	W(A)	
3		R(A)
4		W(A)
5	Abort	

Nhận xét, lịch S trên không phải là lịch chống quay lui dây chuyền vì T2 đọc dữ liệu A trong khi T1 chưa hoàn tất giao tác của mình trên ĐVDL đó.

Quay lui dây chuyền và lịch chống quay lui dây chuyền

- Để lịch S trở thành lịch chống quay lui dây chuyền thì S có thể phải thay đổi như sau:

TT	T1	T2
1	R(A)	
2	W(A)	
3	Commit	
4		R(A)
5		W(A)

Khả phục hồi và lịch khả phục hồi

- Trở lại ví dụ trên, nhưng với lịch S có 1 chút thay đổi:

TT	T1	T2
1	R(A)	
2	W(A)	
3		R(A)
4		W(A)
5		Commit
6	Abort	

- Rõ ràng khi thực hiện hủy bỏ T1. T2 phụ thuộc vào T1 (T2 đọc và ghi đè giá trị lên A). Tuy nhiên T2 đã thực hiện commit, việc hủy bỏ T2 là không thể thực hiện được

  ➔ Lịch trên không khả phục hồi.
- Vì vậy, người ta đưa ra yêu cầu cho 1 lịch để đảm bảo tính khả phục hồi của nó: Một giao tác Tj chỉ được phép kết thúc giao tác (committed) khi tất cả các giao tác khác mà nó phụ thuộc đã kết thúc.

Nhận xét

- Lịch chống quay lui dây chuyền $\rightarrow$ khả phục hồi?
- Lịch khả phục hồi $\not\rightarrow$ chống quay lui dây chuyền?

Deadlock

- Định nghĩa: Là tình trạng 2 hay nhiều giao tác đang tranh chấp tài nguyên và phải chờ các giao tác còn lại hoàn tất, kết quả là không 1 giao tác nào thực hiện được.
- Phát hiện (Detection):
  - Dùng đồ thị chờ:

    Phương pháp biểu diễn đồ thị chờ:

    - Với T, U là 2 transaction trong lịch.
    - Vẽ cung kéo từ T U khi:
    - T đang chờ U nhả khóa trên DVDL X.
    - U đang giữ khóa.
    - T không thể khóa X khi U chưa nhả khóa.

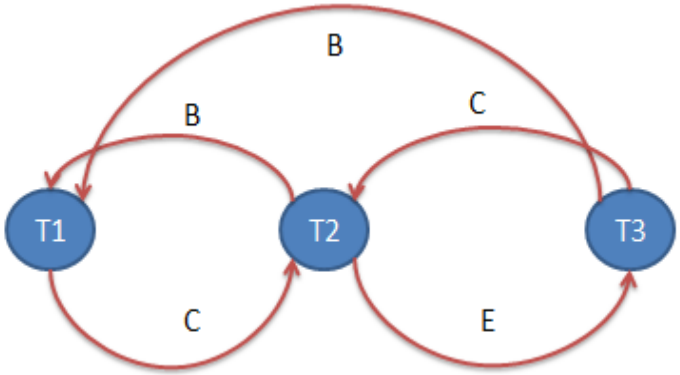
Ví dụ

STT	T1	T2	T3
1	Rlock (A)		
2	S1=A		
3		Rlock(C)	
4		S2=C+1	
5			WLock(E)
6			E=E-1
7	Wlock(B)		
8	B=S1+B		
9		RLock(B)	
10		S2=S2-B	
11			Rlock(B)
12			S3=B+1
13	Wlock(C)		
14	C=C+1		
15		Wlock(E)	
16		E=S2	
17		Rlock(D)	
18		Print(D)	
19			Wlock(C)
20			C=S3
	Unlock	Unlock	Unlock

Cho A=1, B=2, C=1, D=2, E=3.

Dùng đồ thị chờ để đánh giá có

Dead lock hay ko?



Đồ thị có chu trình nên ➔ xảy ra hiện tượng Deadlock.

Phương pháp giải quyết Deadlock (Prevention)

- **Phương pháp**: Khi hệ thống xảy ra Deadlock, để giải quyết Deadlock, trên đồ thị chờ, hủy đi đỉnh (giao tác) có nhiều cung đi vào đi ra nhất (bậc cao nhất).
- Ví dụ: Trở lại ví dụ trên, hãy đưa ra phương án giải quyết hiện tượng deadlock?
  - Hướng dẫn: Hủy đỉnh (hay giao tác có nhiều cung vào hay ra). T2

Phương pháp ngăn ngừa Deadlock (Avoidance)

- **Nguyên lý**: Trước khi hệ thống chấp nhận 1 yêu cầu(thao tác) từ 1 giao tác lên CSDL, hệ thống sẽ kiểm tra xem CSDL này có đang bị tranh chấp không đồng thời dự báo xem nếu chấp nhận yêu cầu này có thể đưa hệ thống vào tình trạng Deadlock ko. Nếu có thì hệ thống tạm ngưng hoặc hủy bỏ yêu cầu.
- **Yêu cầu**: Hệ thống cần biết các tài nguyên đang được sử dụng, đang tranh chấp,…

Phương pháp

- Thuật toán 1: WAIT or DIE

Ti, Tj có time-stamp tsTi và tsTj

Ti yêu cầu lock trên ĐVDL đang bị lock bởi Tj (Ti → Tj)

Nếu tsTi < tsTj thì:

Ti phải chờ.

Ngược lại:

Ti rollback.

- Thuật toán 2: WOUND or WAIT

Ti, Tj có time-stamp tsTi và tsTj

Ti yêu cầu lock trên ĐVDL đang bị lock bởi Tj (Ti → Tj)

Nếu tsTi < tsTj thì:

Rollback Tj

Ngược lại:

Ti phải chờ

Phương pháp

- Giải thích: Xét 2 giao tác trong lịch giao tác.
  - Một giao tác được gọi là Younger nếu ts của nó lớn hơn. Ngược lại nó được gọi là Older.
  - Như ở trên đồ thị chờ, cung Ti → Tj khi Ti yêu cầu lock X khi Tj đang nắm lock trên X.
- 2 thuật toán trên được tóm gọn bởi bảng sau:

	Wait/Die	**Wound/Wait**
Older → Younger	Older waits	Younger dies
Younger → Older	Younger dies	Younger waits

Chú ý

- Với thuật toán Wait/Die thì chỉ có các transaction Older là phải thực hiện chờ các transaction khác. Nên nếu ta có chu trình của 1 lịch rơi vào trạng thái Deadlock như sau:

  T1 → T2 → T3 → … → Tn-1 → Tn → T1

- Giả sử với thuật toán trên vẫn xảy ra Deadlock, có nghĩa là:

  tsT1 > tsT2 > tsT3 > … >tsTn-1 > tsTn > tsT1.

  Suy ra tsT1 > tsT1

  Rõ ràng vô lý.

- Tương tự đối với thuật toán Wound or Wait với các transaction phải thực hiện chờ là Younger.
- Đó là cách thức mà 2 thuật toán trên phòng chống DeadLock.

Ví dụ

- Ví dụ: Trở lại ví dụ trên nhưng thêm các dữ kiện sau:
  - TS(T1) = 100
  - TS(T2) = 200
  - TS(T3) = 300
  - A= 10, B=20, C=15, D=25, E=30.
- Yêu cầu:
  - Đưa ra giải pháp để tránh.
  - Cho biết các giá trị của A, B, C, D, E ứng với giải pháp này sau khi kết thúc.
- Hướng dẫn: sử dụng thuật toán Wound-Wait

Xét cung đầu tiên (đến bước chạy thứ 9):

STT	T1=100	T2=200	T3=300	A=10	B=20	C=15	D=25	E=30
1	Rlock (A)							
2	S1=A			S1=10				
3		Rlock(C)						
4		S2=C+1						
5			WLock(E)					
6			E=E-1					E=29
7	Wlock(B)							
8	B=S1+B				B=30			
9		RLock(B)		A=10	B=30	C=15	D=25	E=29
10		S2=S2-B						
11			Rlock(B)					
12			S3=B+1					
13	Wlock(C)							
14	C=C+1							
15		Wlock(E)						
16		E=S2						
17		Rlock(D)						
18		Print(D)						
19			Wlock(C)					
20			C=S3					
	Unlock	Unlock	Unlock					

Nhận xét

- T2 đợi T1 trên ĐVDL(B) → Ta có cung T2 → T1.
- Ta có TsT2= 200 và TsT1=100 → TsT2 > TsT1.
- Suy ra T2 thực hiện chờ T1 nhả lock trên B.

Xét bước chạy thứ 11

STT	T1=100	T2=200	T3=300	A=10	B=20	C=15	D=25	E=30
1	Rlock (A)							
2	S1=A			S1=10				
3		Rlock(C)						
4		S2=C+1						
5			WLock(E)					
6			E=E-1					E=29
7	Wlock(B)							
8	B=S1+B				B=30			
9		RLock(B)		A=10	B=30	C=15	D=25	E=29
10		S2=S2-B						
11			Rlock(B)	A=10	B=30	C=15	D=25	E=29
12			S3=B+1					
13	Wlock(C)							
14	C=C+1							
15		Wlock(E)						
16		E=S2						
17		Rlock(D)						
18		Print(D)						
19			Wlock(C)					
20			C=S3					
	Unlock	Unlock	Unlock					

Nhận xét

- Ta có T3 → T1
- Và tsT3= 300, tsT1=100, suy ra tsT3>tsT1
  - Nên T3 thực hiện chờ T1 nhả lock

Xét bước chạy thứ 13

STT	T1=100	T2=200	T3=300	A=10	B=20	C=15	D=25	E=30
1	Rlock (A)							
2	S1=A			S1=10				
3		Rlock(C)						
4		S2=C+1				S2=16		
5			WLock(E)					
6			E=E-1					E=29
7	Wlock(B)							
8	B=S1+B				B=30			
9		RLock(B)		A=10	B=30	C=15	D=25	E=29
10		S2=S2-B				S2=-14		
11			Rlock(B)	A=10	B=30	C=15	D=25	E=29
12			S3=B+1					
13	Wlock(C)							
14	C=C+1							
15		Wlock(E)						
16		E=S2						
17		Rlock(D)						
18		Print(D)						
19			Wlock(C)					
20			C=S3					
	Unlock	Unlock	Unlock					

Nhận xét

- Ta có T1 → T2
- Và tsT1= 100, tsT2=200, suy ra tsT2>tsT1
- Thực hiện rollback T2

Xét bước chạy thứ 15:

STT	T1=100	T2=200	T3=300	A=10	B=20	C=15	D=25	E=30
1	Rlock (A)							
2	S1=A			S1=10				
3		Rlock(C)						
4		S2=C+1				S2=16		
5			WLock(E)					
6			E=E-1					E=29
7	Wlock(B)							
8	B=S1+B				B=30			
9		RLock(B)		A=10	B=30	C=15	D=25	E=29
10		S2=S2-B				S2=-14		
11			Rlock(B)	A=10	B=30	C=15	D=25	E=29
12			S3=B+1		S3=31			
13	Wlock(C)							
14	C=C+1							
15		Wlock(E)		A=10	B=30	C=15	D=25	E=30
16		E=S2						E=-14
17		Rlock(D)						
18		Print(D)						
19			Wlock(C)					
20			C=S3					
	Unlock	Unlock	Unlock					

Nhận xét

- Ta có T2 → T3.
- Và tsT3= 300, tsT2=200, suy ra tsT3>tsT2
- Thực hiện rollback T3

Xét bước chạy thứ 19

STT	T1=100	T2=200	T3=300	A=10	B=20	C=15	D=25	E=30
1	Rlock (A)							
2	S1=A			S1=10				
3		Rlock(C)						
4		S2=C+1				S2=16		
5			WLock(E)					
6			E=E-1					E=29
7	Wlock(B)							
8	B=S1+B				B=30			
9		RLock(B)		A=10	B=30	C=15	D=25	E=29
10		S2=S2-B				S2=-14		
11			Rlock(B)	A=10	B=30	C=15	D=25	E=29
12			S3=B+1		S3=31			
13	Wlock(C)							
14	C=C+1							
15		Wlock(E)		A=10	B=30	C=15	D=25	E=30
16		E=S2						E=-14
17		Rlock(D)					D=25	
18		Print(D)						
19			Wlock(C)					
20			C=S3					
	Unlock	Unlock	Unlock					
				A=10	B=30	C=31	D=25	E=-14

Nhận xét

- Ta có T3 → T1.
- Và tsT3= 300, tsT1=100, suy ra tsT3>tsT1
- T3 đợi T1 hoàn tất và nhả lock.

- ***Kết quả cuối cùng:***
  - A=10
  - B=30
  - C=31
  - D=25
  - E=-14