Introduction to Database Systems

Exercises: Transactions (Part 2)

Recap from last time

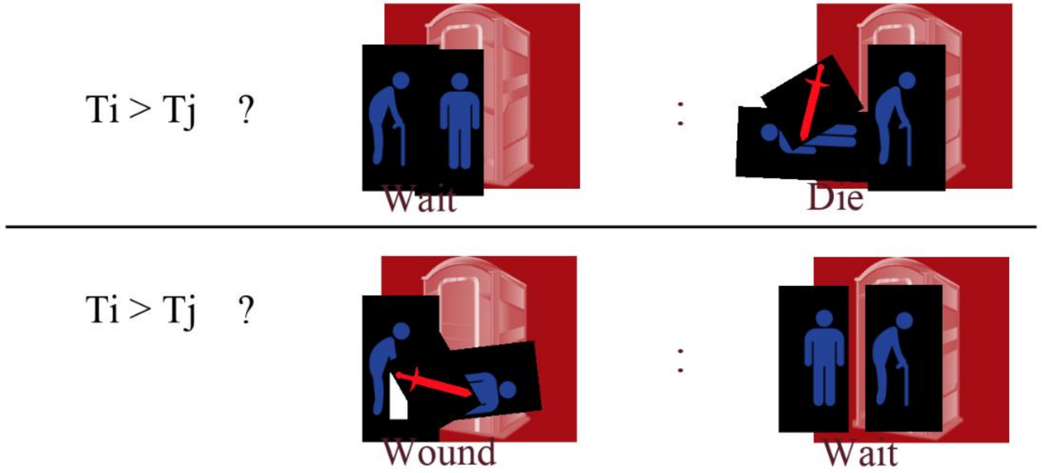
- ACID properties
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- Conflicts
  - Write-Read
  - Read-Write
  - Write-Write
- Two faze locking (2FL)
  - Shared lock
  - Exclusive lock
- Deadlocks

Conflict serializability

- Two schedules are conflict equivalent if:
  - Involve the same actions of the same transactions
  - Every pair of conflicting actions is ordered the same way
- Schedule S is conflict serializable if S is conflict equivalent to some serial schedule (dependency graph is acyclic)
- We can check for confllict serializability by bulding „**dependency“ graphs**
  - One node per each transaction
  - Edge from Ti to Tj if:
    - An operation Oi of Ti conflicts with an operation Oj of Tj
    - Oi appears earlier in the schedule than Oj

Deadlock - Avoidance

- One way to get around deadlocks – to avoid them
- We assign each transaction their priority
- Two main options:
    - Wait-Die: If Ti has higher priority, Ti waits for Tj; else Ti aborts
    - Wound-Wait: If Ti has higher priority, Tj aborts; else Ti waits



Deadlock - Detection

- Avoiding aborts many transactions
- With detecting deadlock in advance we can abort less transactions
- We detect deadlocks by creating **„waits-for“ graphs**
  - Graph will have one node per transaction and an edge from Ti to Tj if:
    - Tj holds a lock on resource A
    - Ti tries to acquire a lock on a resource A, but Tj must release its lock on resource A before Ti can acquire its desired lock
  - DBMS periodically builds the graph and checks for deadlocks
  - Deadlock is represented by a cycle in a graph
  - When a deadlock is found, one or more transactions need to be aborted

Exercises

Exercise 1

Transactions T1 and T2 have the following set of actions over objects A and B:

T1: R(A), W(A), R(B), W(B)

T2: W(B), R(A), R(B)

a) Show an example of concurrent implementation that will lead to a deadlock.

b) How would you handle a deadlock?

DBMS sets priorities **based on timestamps** (the smaller the timestamp, the higher the priority).

Ti wants a lock, which is held by Tj.

Two possible policies:

Wait-Die: If Ti has higher priority, Ti waits for Tj; otherwise Ti aborts.

Wound-Wait: If Ti has higher priority, Tj aborts; otherwise Ti waits.

Conservative 2FL: Ti gets all (necessary) locks at the beginning.

c) How would you resolve a deadlock with Wound-Wait policy?

d) How would you resolve a deadlock with Wait-Die policy?

Exercise 2 (1/2)

Given are the following schedules:

S1 = W2(A), W1(A), R3(A), R1(A), W2(B), R3(B), R3(C), R2(A)

S2 = R3(C), R3(B), W2(B), R2(C), W1(A), R3(A), W2(A), R1(A)

S3 = R3(C), W2(A), W2(B), R1(A), R3(A), R2(C), R3(B), W1(A)

S4 = R2(C), W2(A), R3(C), W1(A), W2(B), R1(A), R3(A), R3(B)

S1	T1		W1(A)		R1(A)				
	T2	W2(A)				W2(B)			R2(A)
	T3			R3(A)			R3(B)	R3(C)	

S2	T1					W1(A)			R1(A)
	T2			W2(B)	R2(C)			W2(A)	
	T3	R3(C)	R3(B)				R3(A)		

S3	T1				R1(A)				W1(A)
	T2		W2(A)	W2(B)			R2(C)		
	T3	R3(C)				R3(A)		R3(B)	

S4	T1				W1(A)		R1(A)		
	T2	R2(C)	W2(A)			W2(B)			
	T3			R3(C)				R3(A)	R3(B)

Exercise 2

S1	T1		W1(A)		R1(A)				
	T2	W2(A)				W2(B)			R2(A)
	T3			R3(A)			R3(B)	R3(C)	

S2	T1					W1(A)			R1(A)
	T2			W2(B)	R2(C)			W2(A)	
	T3	R3(C)	R3(B)				R3(A)		

S3	T1				R1(A)				W1(A)
	T2		W2(A)	W2(B)			R2(C)		
	T3	R3(C)				R3(A)		R3(B)	

S4	T1				W1(A)		R1(A)		
	T2	R2(C)	W2(A)			W2(B)			
	T3			R3(C)				R3(A)	R3(B)

a) Which of the above schedules is conflict equivalent?

Two schedules are conflict equivalent if:
- Involve the same actions of the same transactions
- Every pair of conflicting actions is ordered the same way

b) Which of the above schedules is conflict serializable?
Schedule S is conflict serializable if S is conflict equivalent to some serial schedule

Exercise 3

The following transactions are given:

T1: R(A); R(B); if A = 0 then B=B+1; W(B).

T2: R(B); R(A); if B = 0 then A=A+1; W(A).

a) Can these two transactions lead to a deadlock?

b) How would you resolve a deadlock with a conservative 2PL (C2PL)?

c) How would you resolve a deadlock with of Wait-Die policy?

Exercise 4 (1/2)

The following sequences of actions are given:

S1: T1:R(A), T2:W(A), T2:W(B), T3:W(B), T1:W(B), T1:Commit, T2:Commit, T3:Commit

S2: T1:R(A), T2:W(B), T2:W(A), T3:W(B), T1:W(B), T1:Commit, T2:Commit, T3:Commit

	T1	R(A)				W(B)	C		
S1	T2		W(A)	W(B)				C	
	T3				W(B)				C
	T1	R(A)				W(B)	C		
S2	T2		W(B)	W(A)				C	
	T3				W(B)				C

S1

T1	R(A)				W(B)	C		
T2		W(A)	W(B)				C	
T3				W(B)				C

S2

T1	R(A)				W(B)	C		
T2		W(B)	W(A)				C	
T3				W(B)				C

Exercise 4 (2/2)

For each schedule describe how the concurrency control mechanisms prevent a deadlock.

a) Strict 2PL with timestamps (Wait-Die) used for deadlock prevention.

S1: T1 / T2 / T3

S2: T1 / T2 / T3

b) Strict 2PL with deadlock detection (in case of deadlock display Wait-for graph).

S1: T1 / T2 / T3

S2: T1 / T2 / T3

c) Conservative 2PL.

S1: T1 / T2 / T3

S2: T1 / T2 / T3

Exercise 5

Deadlock detection (with Wait-for graph)

T1:	S(A)	S(D)		S(B)					
T2:			X(B)				X(C)		
T3:					S(D)	S(C)			X(A)
T4:								X(B)	