Concurrency Control Techniques

Outline

Databases Concurrency Control

1. Purpose of Concurrency Control
2. Two-Phase locking
3. Deadlock and Starvation

1.1 Purpose of Concurrency Control

- To enforce Isolation (through mutual exclusion) among conflicting transactions.
- To preserve database consistency through consistency preserving execution of transactions.
- To resolve read-write and write-write conflicts.

Example: In concurrent execution environment if T1 conflicts with T2 over a data item A, then the existing concurrency control decides if T1 or T2 should get the A and if the other transaction is rolled-back or waits.

2.1 Two-Phase Locking Techniques

Locking is an operation which secures (a) permission to Read or (b) permission to Write a data item for a transaction. Example: Lock (X). Data item X is locked on behalf of the requesting transaction.

Unlocking is an operation which removes these permissions from the data item.

Example: Unlock (X). Data item X is made available to all other transactions.

Lock and Unlock are Atomic operations.

2.2 Two-Phase Locking Techniques

Two locks modes (a) shared (read) and (b) exclusive (write).

Shared mode: shared lock (X). More than one transaction can apply share lock on X for reading its value. (In other words, many can read)

Exclusive mode: Write lock (X). Only one write lock on X can exist at any time and no shared lock can be applied by any other transaction on X. (In other words, only one can write)

Conflict matrix

	Read	Write
Read	Y	N
Write	N	N

2.3 Two-Phase Locking Techniques

Lock Manager: Managing locks on data items.

Lock table: Lock manager uses it to store the identity of transaction locking a data item, the data item, lock mode and pointer to the next data item locked.

One simple way to implement a lock table is through linked list.

Transaction ID	Data item id	lock mode	Ptr to next data item
T1	X1	Read	Next

2.4 Two-Phase Locking Techniques

Database requires that all transactions should be well-formed. A transaction is well-formed if:

- It must lock the data item before it reads or writes to it.
- It must not lock an already locked data items and it must not try to unlock a free data item.

2.5 Two-Phase Locking Techniques

A transaction is said to follow Two Phase Locking protocol if all locking operations (read_lock, write_lock) precede the first unlock operation.

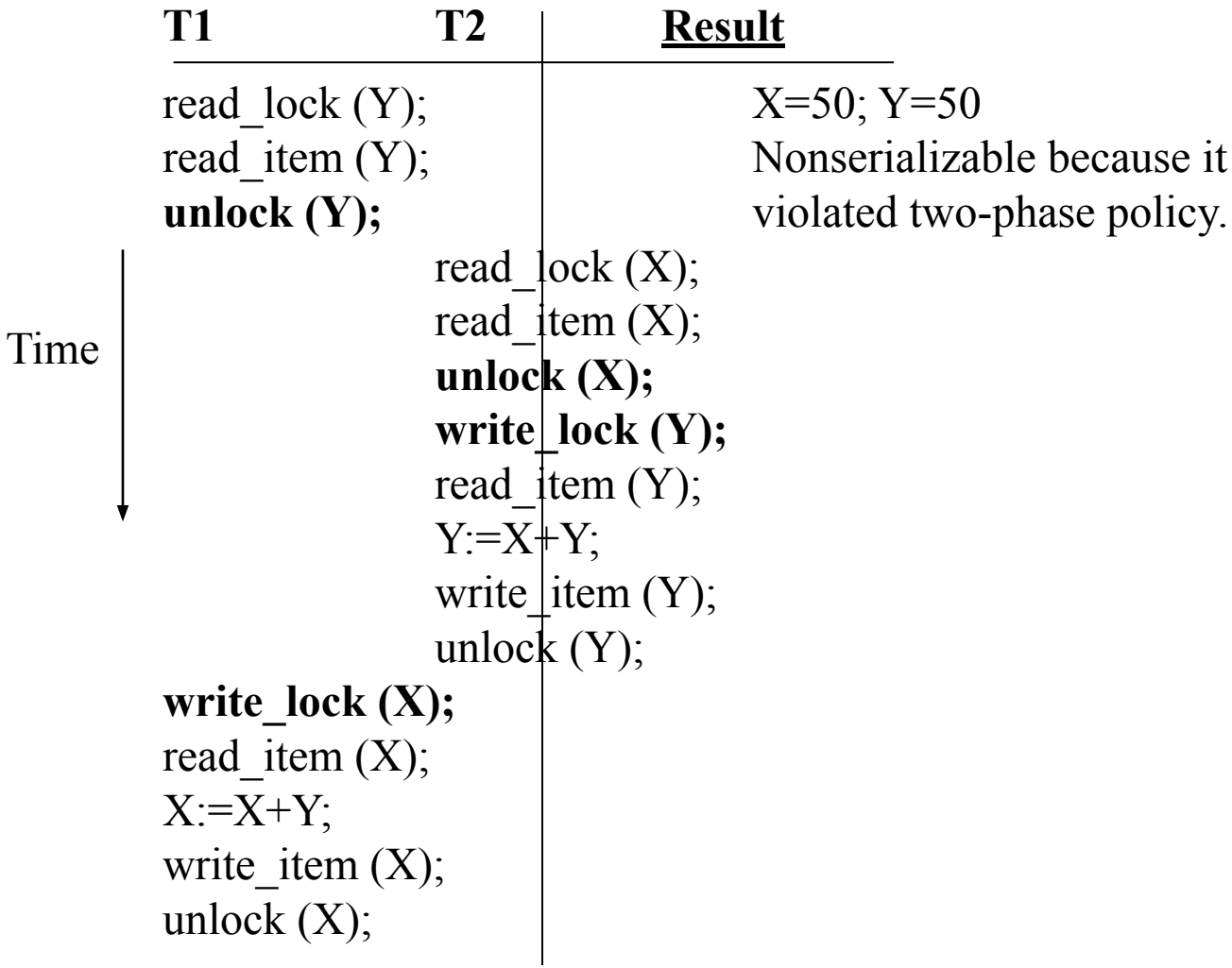
If a schedule follows the Two Phase Locking protocol, the schedule is serializable.

2.6 Two-Phase Locking Techniques

T1	**T2**	**Result**
read_lock (Y);	read_lock (X);	Initial values: X=20; Y=30
read_item (Y);	read_item (X);	Result of serial execution
unlock (Y);	unlock (X);	T1 followed by T2
write_lock (X);	Write_lock (Y);	X=50, Y=80.
read_item (X);	read_item (Y);	Result of serial execution
X:=X+Y;	Y:=X+Y;	T2 followed by T1
write_item (X);	write_item (Y);	X=70, Y=50
unlock (X);	unlock (Y);	

T1 and T2 are not serial since they violated Two Phase Locking protocol.

2.7 Two-Phase Locking Techniques



2.8 Two-Phase Locking Techniques

T'1	**T'2**	
read_lock (Y);	read_lock (X);	T1 and T2 follow two-phase
read_item (Y);	read_item (X);	policy but they are subject to
write_lock (X);	Write_lock (Y);	deadlock, which must be
unlock (Y);	unlock (X);	dealt with.
read_item (X);	read_item (Y);	
X:=X+Y;	Y:=X+Y;	
write_item (X);	write_item (Y);	
unlock (X);	unlock (Y);	

3.1 Deadlock & Starvation

Dealing with Deadlock and Starvation

Deadlock

T'1	T'2	
read_lock (Y);		T1 and T2 did follow two-phase
read_item (Y);		policy but they are deadlock
	read_lock (X);	
	read_item (Y);	
write_lock (X);		
(waits for X)	write_lock (Y);	
	(waits for Y)	

Deadlock (T'1 and T'2)

3.2 Deadlock & Starvation

Dealing with Deadlock and Starvation

Deadlock prevention

A transaction locks all data items it refers to before it begins execution. This way of locking prevents deadlock since a transaction never waits for a data item. The conservative two-phase locking uses this approach.

3.3 Deadlock & Starvation

Dealing with Deadlock and Starvation

Deadlock detection and resolution

In this approach, deadlocks are allowed to happen. The scheduler maintains a wait-for-graph for detecting cycle. If a cycle exists, then one transaction involved in the cycle is selected (victim) and rolled-back.

A wait-for-graph is created using the lock table. As soon as a transaction is blocked, it is added to the graph. When a chain like: Ti waits for Tj waits for Tk waits for Ti or Tj occurs, then this creates a cycle. One of the transaction of the cycle is selected and rolled back.

3.4 Deadlock & Starvation

Dealing with Deadlock and Starvation

Deadlock avoidance

There are many variations of two-phase locking algorithm. Some avoid deadlock by not letting the cycle to complete. That is as soon as the algorithm discovers that blocking a transaction is likely to create a cycle, it rolls back the transaction.

3.5 Deadlock & Starvation

Dealing with Deadlock and Starvation

Starvation

Starvation occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further. In a deadlock resolution it is possible that the same transaction may consistently be selected as victim and rolled-back. This limitation is inherent in all priority based scheduling mechanisms.