Concurrency Control

Introduction

- Fundamental Property of a transaction is isolation
- To ensure that isolation property is preserved when several transactions are running concurrently, the system controls the interaction among the concurrent transactions
- These schemes are called concurrency control

2

Lock-Based Protocols

- One way to ensure serializability
  - Require data items be accessed in a mutually exclusive manner
  - That is when one transaction is accessing the data item, no other transaction can modify that data item
  - Common method to implement is that transaction must hold a lock on an item

3

Lock-Based Protocols

- Modes in which data item may be locked
  - Shared
    - If a transaction T1 has obtained a shared-mode lock on item Q, then T1 can read, but cannot write, Q
  - Exclusive
    - If a transaction T1 has obtained an exclusive-mode lock on item Q, then T1 can both read or write Q

4

Lock-Based Protocols

- Every transaction is required to request a lock in an appropriate mode on each data item
- Request is made to the concurrency-control manager
- Concurrency-control manager must grant the lock to the transaction before it can proceed.

5

Compatibility Function

- Given set of lock modes can define compatibility function
  - A & B represent arbitrary lock modes
  - Transaction T1 request a lock of mode A on item Q
  - Transaction T2 currently holds a lock of mode B
  - If T1 can be granted a lock on Q immediately in spite of the presence of the mode B lock, then mode A is compatible with mode B

6

Compatibility Function

- Can represent with a Lock-compatibility matrix

	Shared	Exclusive
Shared	true	false
Exclusive	false	false

7

Compatibility Function

T1	T2	Concurrency Manager
Lock-X(B)		
		Grant-X(B, T1)
Read(B)		
B:=B-50		
Write(B)		
Unlock(B)		
	Lock-S(A)	
		Grant-S(A, T2)
	Read(A)	
	Unlock(A)	
	Lock-S(B)	
		Grant-S(B, T2)
	Read(B)	
	Unlock(B)	
Lock-X(A)	Display (A+B)	
		Grant-X(A, T2)
Read(A)		
A:=A + 50		
Write(A)		
Unlock(A)		

Lock-X = exclusive lock
Lock-S = shared lock

8

Compatibility Function

- When a transaction requests a lock that is incompatible, it enters a wait state until all incompatible locks have been released
- Transactions cannot execute until concurrency-control manager grants the requested locks

9

Deadlock

- Locking can lead to an undesirable situation where no transaction can proceed with normal execution
- This situation is called deadlock
- When this occurs
  - The system must roll back one of the transactions
  - Unlocking transactions until execution can be continued

10

Deadlock

T1	T2
Lock-X(B)	
Read(B)	
B := B-50	
Write(B)	
	Lock-S(A)
	Read(A)
	Lock-S(B)
Lock-X(A)	

11

Deadlock

- If locking is not used, or if a data item is unlocked as soon as possible after reading or writing, inconsistent states may occur
- On the other hand, if a data item is not unlocked before requesting a lock on some other data item deadlocks may occur

12

Deadlock

- In general, deadlocks are a necessary evil associated with locking which is necessary to avoid inconsistent states
- Deadlocks are preferable to inconsistent states
  - since they can be handled via rolling back transactions
  - inconsistent states cannot be handled by database

13

Locking Protocol

- Locking Protocol
  - a set of rules that each transaction must follow
  - indicates when a transaction may lock or unlock each data item

14

Locking Protocol

- A schedule is legal under a given locking protocol if it follows the rules
- A locking protocol ensures conflict serializability if and only if all legal schedules are conflict serializable

15

Granting of Locks

- Grant can take place if
  - a transaction requests a lock on a data item in some mode
  - and no other transaction has a lock on the same data item in a conflicting mode
- Care must be take to avoid certain situations - see next slide

16

Granting of Locks

- Transaction T2 has a shared-mode lock on a data item
- Transaction T1 requests an exclusive-mode lock on the data item
- Clearly, T1 has to wait for T2 to release the shared-mode lock
- Meanwhile, transaction T3 may request a shared-mode lock on the same data item ….

17

Granting of Locks

- The lock request is compatible with the lock granted to T2 so T3 may be granted the shared-mode lock
- At this point, T2 releases the lock but T1 has to still wait for T3 to finish
- But again, there are other transactions Ti, that requests a shared-mode lock
- ….

18

Granting of Locks

- In fact it is possible that there is a sequence of transactions that each requests a shared-mode lock on the data item and T1 NEVER gets the exclusive-mode lock
- T1 is then said to be starved

19

Granting of Locks

- Avoiding starvation of transactions
  - T1 requests a lock on a data item Q in some mode M
  - the lock is granted provided that
    - there is no other transaction holding a lock on Q in a mode that conflicts with M
    - there is no other transaction that is waiting for a lock on Q and that made its lock request before T1

20

Two-Phase Locking Protocol

- A protocol that ensures serializability is the two-phase locking protocol
- Each transaction issues lock and unlock requests in two phases
  - Growing phase: a transaction may obtain locks but may not release any lock
  - Shrinking phase: a transaction may release locks but may not obtain any new locks

21

Two-Phase Locking Protocol

- Initially a transaction is in the growing phase
- Once the transaction releases a lock, it enters shrinking phase and cannot request more locks

22

Two-Phase Locking Protocol

- Two-phase locking protocol
  - Ensures conflict serializability
  - Does not ensure freedom from deadlock

23

Two-Phase Locking Protocol

- Variation of two-phase locking protocol
  - Strict two-phase locking protocol
    - requires not only that locking be two phase, but that all exclusive-mode locks taken by a transaction be held until that transaction commits
  - Rigorous two-phase locking protocol
    - requires that all locks be held until the transaction commits

24

Two-Phase Locking Protocol

- Refinement of basic two-phase locking protocol
  - lock conversions are allowed
    - a mechanism is allowed for upgrading a shared lock to an exclusive lock
    - a mechanism is allowed for downgrading an exclusive lock to a shared lock
- Strict two-phase and rigorous two-phase locking (with lock conversions) are extensively used in DBMSs

25

Two-Phase Locking Protocol

- A simple but widely used scheme automatically generates the appropriate lick and unlock instructions for a transaction on the basis of read and write requests
- ….

26

Two-Phase Locking Protocol

- When a transaction T1 issues a read(Q), the system issues a lock-S(Q) instruction followed by the read(Q) instruction
- When T1 issues a write(Q) operation, the system checks to see whether T1 already holds a shared lock on Q.
  - If yes, system issues an upgrade(Q) followed by a write(Q)
  - If no, system issues a lock-X(Q) followed by a write(Q)
- ….

27

Two-Phase Locking Protocol

- All locks obtained by a transaction are unlocked after that transaction commits or aborts

28

Other Locking Protocols

- Graph-based protocols
- Timestamp-based protocols
- Validation-based protocols
  - majority of transactions are read-only
- Multiversion schemes
  - each write(Q) creates a new version of Q

29

Deadlock Handling

- Deadlock state
  - there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set
- Two principal methods for dealing with deadlocks
  - deadlock prevention
  - deadlock detection and deadlock recovery

30

Deadlock Handling

- Deadlock prevention - two approaches
  - one approach: ensure that no cyclic waits can occur by
    - ordering the requests for locks
    - or requiring all locks be acquired together
  - second approach: impose an ordering of all data items and require that a transaction lock data items only in a sequence consistent with the ordering

31

Deadlock Handling

- Deadlock detection and recovery
  - an algorithm that examines the state of the system is invoked periodically to see if a deadlock has occurred
  - if one has, then the system must recover

32

Deadlock Handling

- To recover from a deadlock the system must
  - maintain information about the current allocation of data items to transactions as well as any outstanding data item requests
  - provide an algorithm that uses this information to determine whether the system has entered a deadlock state
  - recover from the deadlock when the detection algorithm determines that a deadlock exists.

33