Distributed Database

Munawar, PhD

DISTRIBUTED CONCURRENCY AND DBMS RELIABILITY

Concurrency Control

- The problem of synchronizing concurrent transactions such that the consistency of the database is maintained while, at the same time, maximum degree of concurrency is achieved.
- Anomalies:
  - Lost updates
    - The effects of some transactions are not reflected on the database.
  - Inconsistent retrievals
    - A transaction, if it reads the same data item more than once, should always read the same value.

Concurrency Control and Recovery

- Distributed Databases encounter a number of concurrency control and recovery problems which are not present in centralized databases. Some of them are listed below.
  - Dealing with multiple copies of data items
  - Failure of individual sites
  - Communication link failure
  - Distributed commit
  - Distributed deadlock

Concurrency Control and Recovery

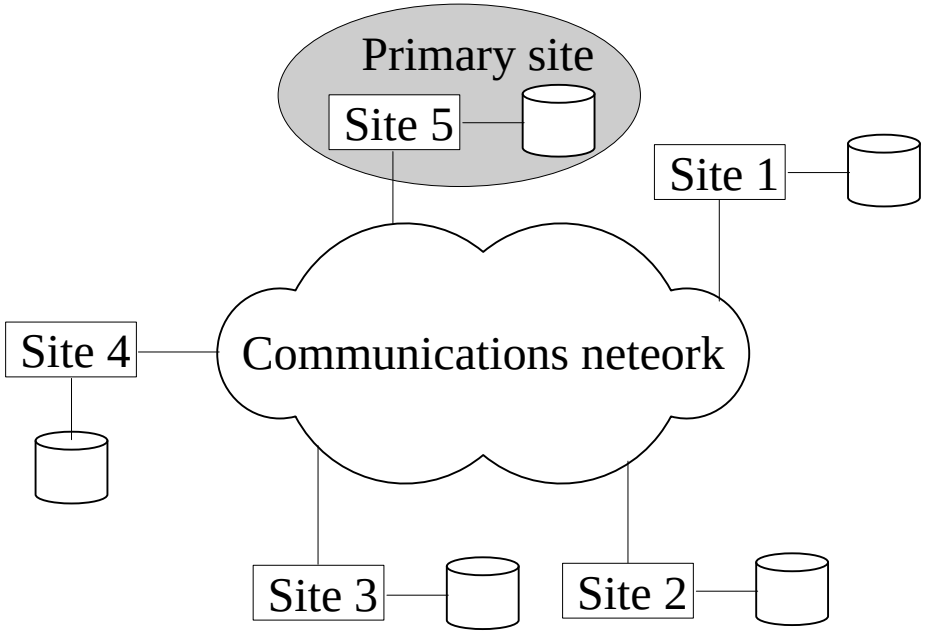
- Details
  - Dealing with multiple copies of data items:
    - The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.
  - Failure of individual sites:
    - Database availability must not be affected due to the failure of one or two sites and the recovery scheme must recover them before they are available for use.

Concurrency Control and Recovery

- Details (contd.)
  - Communication link failure:
    - This failure may create network partition which would affect database availability even though all database sites may be running.
  - Distributed commit:
    - A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit.
  - Distributed deadlock:
    - Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.

Concurrency Control and Recovery

- Distributed Concurrency control based on a distributed copy of a data item
  - Primary site technique: A single site is designated as a primary site which serves as a coordinator for transaction management.



Concurrency Control and Recovery

- Transaction management:
  - Concurrency control and commit are managed by this site.
  - In two phase locking, this site manages locking and releasing data items. If all transactions follow two-phase policy at all sites, then serializability is guaranteed.

Concurrency Control and Recovery

- Transaction Management
  - Advantages:
    - An extension to the centralized two phase locking so implementation and management is simple.
    - Data items are locked only at one site but they can be accessed at any site.
  - Disadvantages:
    - All transaction management activities go to primary site which is likely to overload the site.
    - If the primary site fails, the entire system is inaccessible.
  - To aid recovery a backup site is designated which behaves as a shadow of primary site. In case of primary site failure, backup site can act as primary site.

Concurrency Control and Recovery

- Primary Copy Technique:
  - In this approach, instead of a site, a data item partition is designated as primary copy. To lock a data item just the primary copy of the data item is locked.
- Advantages:
  - Since primary copies are distributed at various sites, a single site is not overloaded with locking and unlocking requests.
- Disadvantages:
  - Identification of a primary copy is complex. A distributed directory must be maintained, possibly at all sites.

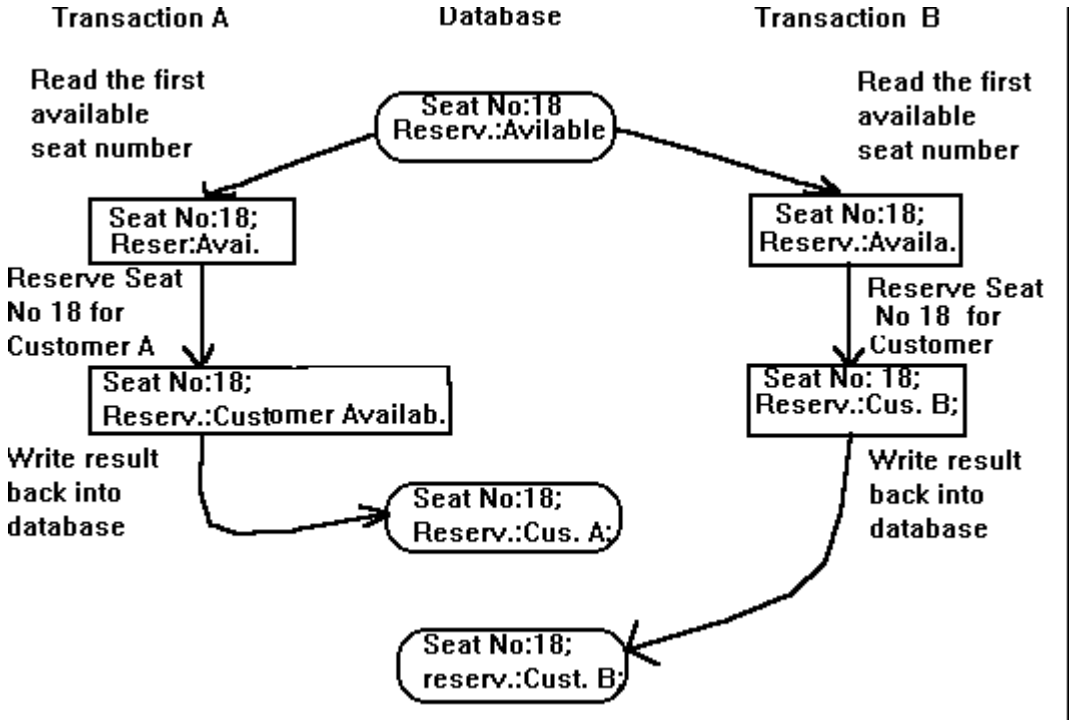
Concurrency Control and Recovery

- Recovery from a coordinator failure
  - In both approaches a coordinator site or copy may become unavailable. This will require the selection of a new coordinator.
- Primary site approach with no backup site:
  - Aborts and restarts all active transactions at all sites. Elects a new coordinator and initiates transaction processing.
- Primary site approach with backup site:
  - Suspends all active transactions, designates the backup site as the primary site and identifies a new back up site. Primary site receives all transaction management information to resume processing.
- Primary and backup sites fail or no backup site:
  - Use election process to select a new coordinator site.

Concurrency Control and Recovery

- Concurrency control based on voting:
  - There is no primary copy of coordinator.
  - Send lock request to sites that have data item.
  - If majority of sites grant lock then the requesting transaction gets the data item.
  - Locking information (grant or denied) is sent to all these sites.
  - To avoid unacceptably long wait, a time-out period is defined. If the requesting transaction does not get any vote information then the transaction is aborted.

Anomaly in DB in Absence of Concurrency Control



Execution History (or Schedule)

- An order in which the operations of a set of transactions are executed.
- A history (schedule) can be defined as a partial order over the operations of a set of transactions.

T_1:	Read(x)	T_2:	Write(x)	T_3:	Read(x)
	Write(x)		Write(y)		Read(y)
	Commit		Read(z)		Read(z)
			Commit		Commit

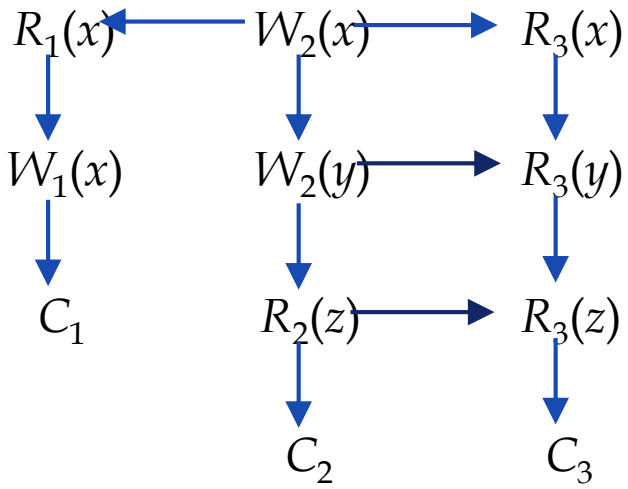
$$H_1=\{W_2(x),R_1(x), R_3(x),W_1(x),C_1,W_2(y),R_3(y),R_2(z),C_2,R_3(z),C_3\}$$

Complete Schedule – Example

Given three transactions

T_1:	Read(x)	T_2:	Write(x)	T_3:	Read(x)
	Write(x)		Write(y)		Read(y)
	Commit		Read(z)		Read(z)
			Commit		Commit

A possible complete schedule is given as the DAG



Serializability in Distributed DBMS

- Somewhat more involved. Two histories have to be considered:
  - local histories
  - global history
- For global transactions (i.e., global history) to be serializable, two conditions are necessary:
  - Each local history should be serializable.
  - Two conflicting operations should be in the same relative order in all of the local histories where they appear together.

Concurrency Control Algorithms

- Pessimistic
  - Two-Phase Locking-based (2PL)
    - Centralized (primary site) 2PL
    - Primary copy 2PL
    - Distributed 2PL
  - Timestamp Ordering (TO)
    - Basic TO
    - Multiversion TO
    - Conservative TO
  - Hybrid
- Optimistic
  - Locking-based
  - Timestamp ordering-based

Locking-Based Algorithms

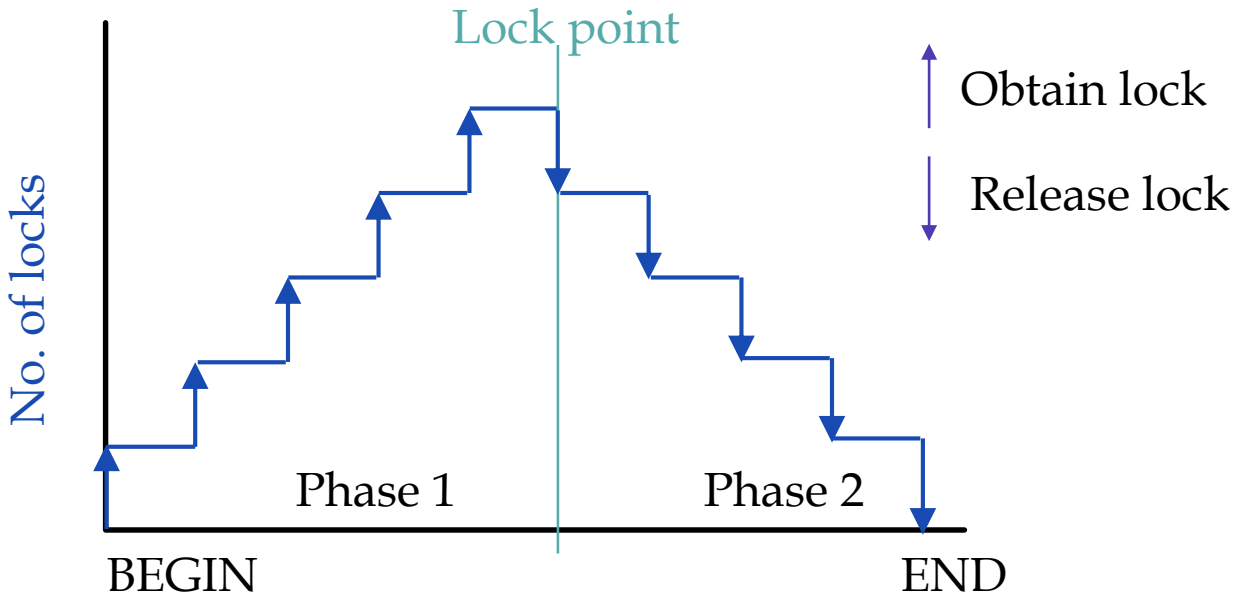
- Transactions indicate their intentions by requesting locks from the scheduler (called lock manager).
- Locks are either read lock (*rl*) [also called shared lock] or write lock (*wl*) [also called exclusive lock]
- Read locks and write locks conflict (because Read and Write operations are incompatible

	rl	*wl*
rl	yes	no
wl	no	no

- Locking works nicely to allow concurrent processing of transactions.

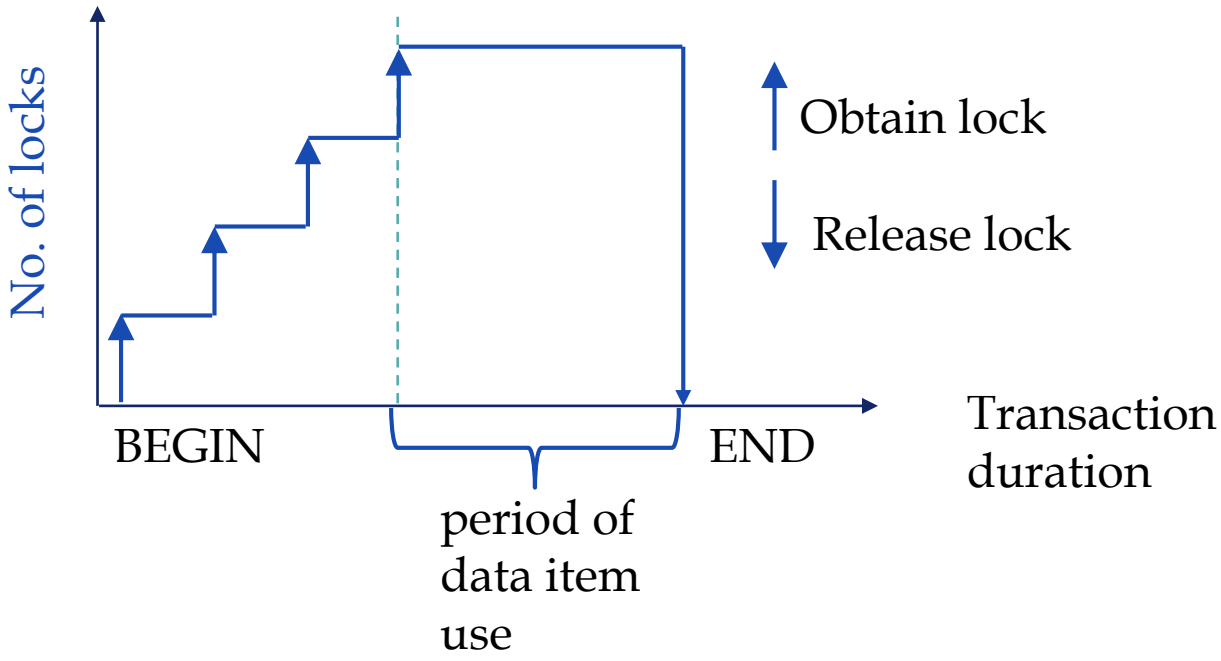
Two-Phase Locking (2PL)

- A Transaction locks an object before using it.
- When an object is locked by another transaction, the requesting transaction must wait.
- When a transaction releases a lock, it may not request another lock.



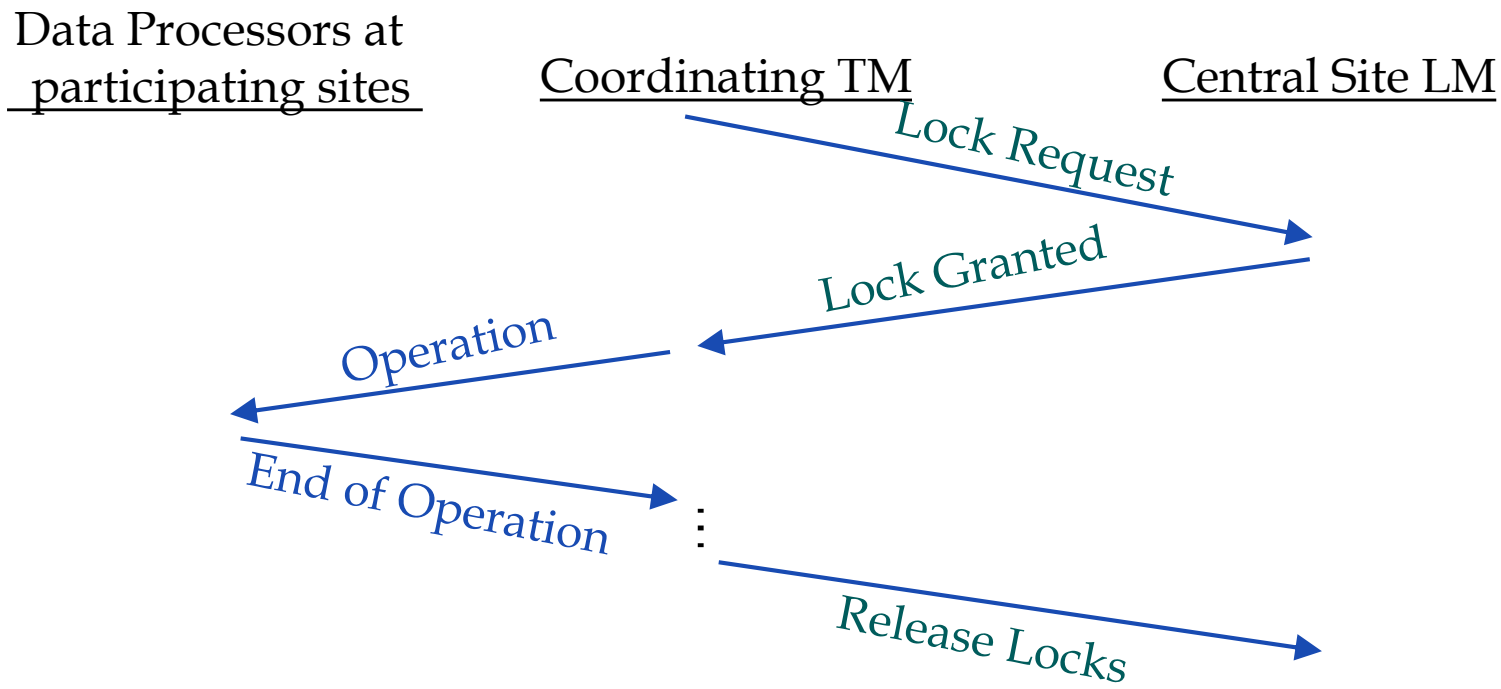
Strict 2PL

Hold locks until the end.



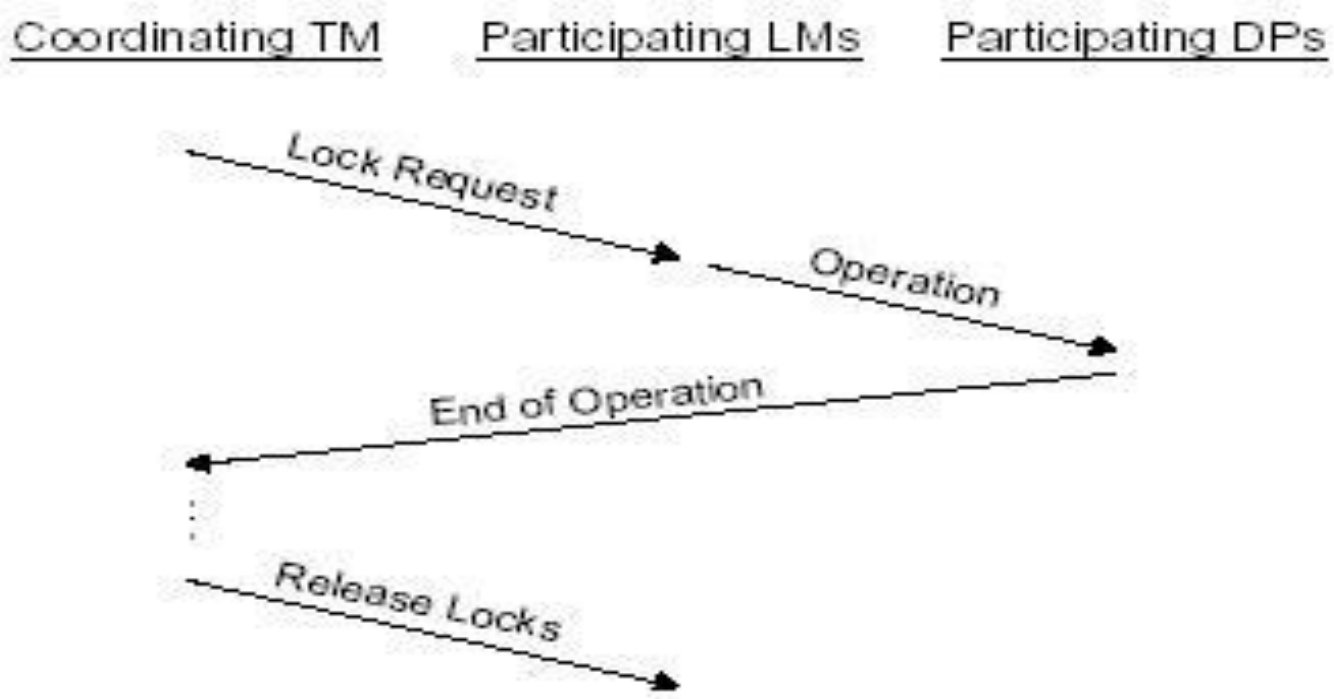
Centralized 2PL

- There is only one 2PL scheduler in the distributed system.
- Lock requests are issued to the central scheduler.



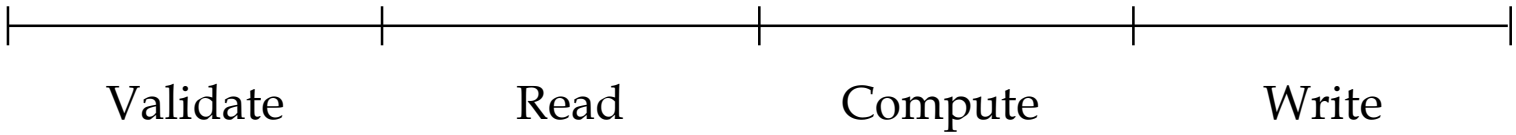
- 2PL schedulers are placed at each site. Each scheduler handles lock requests for data at that site.
- A transaction may read any of the replicated copies of item *x*, by obtaining a read lock on one of the copies of *x*. Writing into *x* requires obtaining write locks for all copies of *x*.

Distributed 2PL

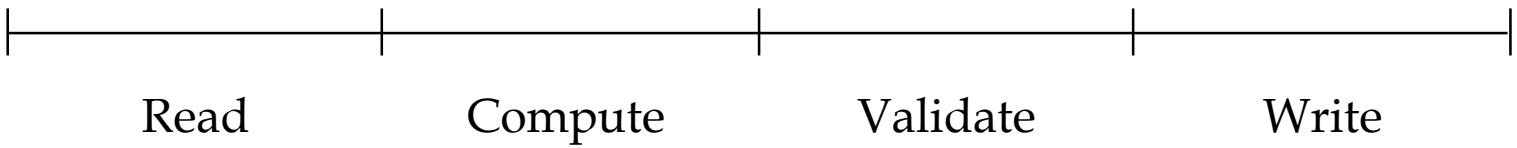


Optimistic Concurrency Control Algorithms

Pessimistic execution



Optimistic execution



Deadlock Management

- Ignore
  - Let the application programmer deal with it, or restart the system
- Prevention
  - Guaranteeing that deadlocks can never occur in the first place. Check transaction when it is initiated. Requires no run time support.
- Avoidance
  - Detecting potential deadlocks in advance and taking action to insure that deadlock will not occur. Requires run time support.
- Detection and Recovery
  - Allowing deadlocks to form and then finding and breaking them. As in the avoidance scheme, this requires run time support.

Deadlock Prevention

- All resources which may be needed by a transaction must be predeclared.
  - The system must guarantee that none of the resources will be needed by an ongoing transaction.
  - Resources must only be reserved, but not necessarily allocated a priori
  - Unsuitability of the scheme in database environment
  - Suitable for systems that have no provisions for undoing processes.
- Evaluation:
  - – Reduced concurrency due to preallocation
  - – Evaluating whether an allocation is safe leads to added overhead.
  - – Difficult to determine (partial order)
  - \+ No transaction rollback or restart is involved.

Deadlock Avoidance

- Transactions are not required to request resources a priori.
- Transactions are allowed to proceed unless a requested resource is unavailable.
- In case of conflict, transactions may be allowed to wait for a fixed time interval.
- Order either the data items or the sites and always request locks in that order.
- More attractive than prevention in a database environment.

Thank You !

Munawar, PhD