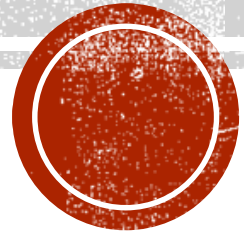
DEADLOCK MANAGEMENT



DEADLOCK MANAGEMENT

- A deadlock is a situation when transactions are endlessly waiting for one another.
- Any lock-based concurrency control algorithm may result in deadlocks, as these algorithms require transactions to wait for one another.
- In lock-based concurrency control algorithms, locks on data items are acquired in a mutually exclusive manner; thus, it may cause a deadlock situation.
- Whenever a deadlock situation arises in a database environment, outside interference is required to continue with the normal execution of the system.
- Therefore, the database systems require special procedures to resolve the deadlock situation.

DEADLOCK MANAGEMENT

- Deadlock situations can be characterized by wait-for graphs, directed graphs that indicate which transactions are waiting for which other transactions.
- In a wait-for graph, nodes of the graph represent transactions and edges of the graph represent the waiting-for relationships among transactions.
- An edge is drawn in the wait-for graph from transaction Ti to transaction T j, if the transaction Ti is waiting for a lock on a data item that is currently held by the transaction T j.
- Using wait-for graphs, it is very easy to detect whether a deadlock situation has occurred in a database environment or not.
- There is a deadlock in the system if and only if the corresponding wait-for graph contains a cycle.

DEADLOCK MANAGEMENT

- In a distributed DBMS it is not sufficient to draw a LWFG for each local DBMS only, but it is also necessary to draw a global wait-for graph (GWFG) for the entire system to detect a deadlock situation.
- In a distributed database, an LWFG is a portion of the GWFG, which consists of only those nodes and edges that are completely contained at a single site.
- Three general techniques are available for deadlock resolution in a distributed database system:
  - deadlock prevention,
  - distributed deadlock avoidance and
  - distributed deadlock detection and
  - recovery from deadlock.

DISTRIBUTED DEADLOCK PREVENTION

- Distributed Deadlock prevention is a cautious scheme in which a transaction is restarted when the system suspects that a deadlock might occur.
- Deadlock prevention is an alternative method to resolve deadlock situations in which a system is designed in such a way that deadlocks are impossible.
- In this scheme, the transaction manager checks a transaction when it is first initiated and does not permit to proceed if there is a risk that it may cause a deadlock.
- In the case of lock- based concurrency control, deadlock prevention in a distributed system is implemented in the following way.

DISTRIBUTED DEADLOCK PREVENTION

- Let us consider that a transaction Ti is initiated at a particular site in a distributed database system and that it requires a lock on a data item that is currently owned by another transaction Tj.
- Here, a deadlock prevention test is done to check whether there is any possibility of a dead-lock occurring in the system.
- The transaction Ti is not permitted to enter into a wait state for the transaction Tj, if there is the risk of a deadlock situation.
- In this case, one of the two transactions is aborted to prevent a deadlock.
- The deadlock prevention algorithm is called non-preemptive if the transaction Ti is aborted and restarted.
- On the other hand, if the transaction Tj is aborted and restarted, then the deadlock prevention algorithm is called preemptive.

DISTRIBUTED DEADLOCK AVOIDANCE

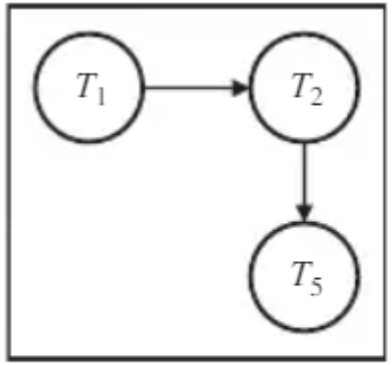
- Distributed deadlock avoidance is another technique to ensure that deadlock situations will not occur in a distributed system.
- Preordering of resources is a deadlock avoidance technique in which each data item in the database system is numbered, and each transaction requests locks on these data items in that numeric order.
- This technique requires that each transaction obtain all its locks before execution.
- Numbering of data items can be done either globally or locally.
- In distributed database systems, it is necessary that all sites are numbered, and transactions that require to access data items from multiple sites must request the corresponding locks by visiting the sites according to the predefined numbers.
- The priority of a transaction is the highest number among the locks that are already owned by the transaction.
- In this method, as a transaction can only wait for those transactions that have higher priorities, no deadlock situations can occur.

DISTRIBUTED DEADLOCK DETECTION AND RECOVERY

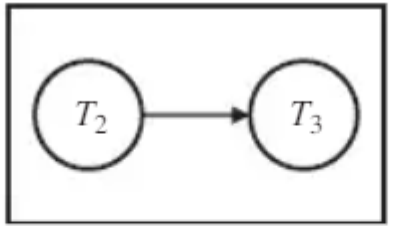
- Deadlock detection and recovery is the most popular and most suitable technique for deadlock management in a database environment.
- In deadlock detection and recovery method, first it is checked whether any deadlock has occurred in the system.
- After detection of a deadlock situation in the system, one of the involved transactions is chosen as the victim transaction and is aborted to resolve the deadlock situation.
- Deadlock situations are detected by explicitly constructing a wait-for graph and searching it for cycles.
- A cycle in the wait-for graph indicates that a deadlock has occurred, and one transaction in the cycle is chosen as the victim, which is aborted and restarted.
- To minimize the cost of restarting, the victim selection is usually based on the number of data items used by each transaction in the cycle.

DISTRIBUTED DEADLOCK DETECTION AND RECOVERY

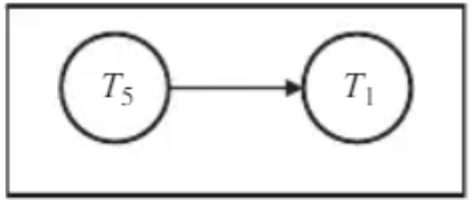
- The principal difficulty in implementing deadlock detection in a distributed database environment is constructing the GWFG efficiently.
- In a distributed DBMS, a LWFG for each local DBMS can be drawn easily; however, these LWFGs are not sufficient to represent all deadlock situations in the distributed system.



Site 1



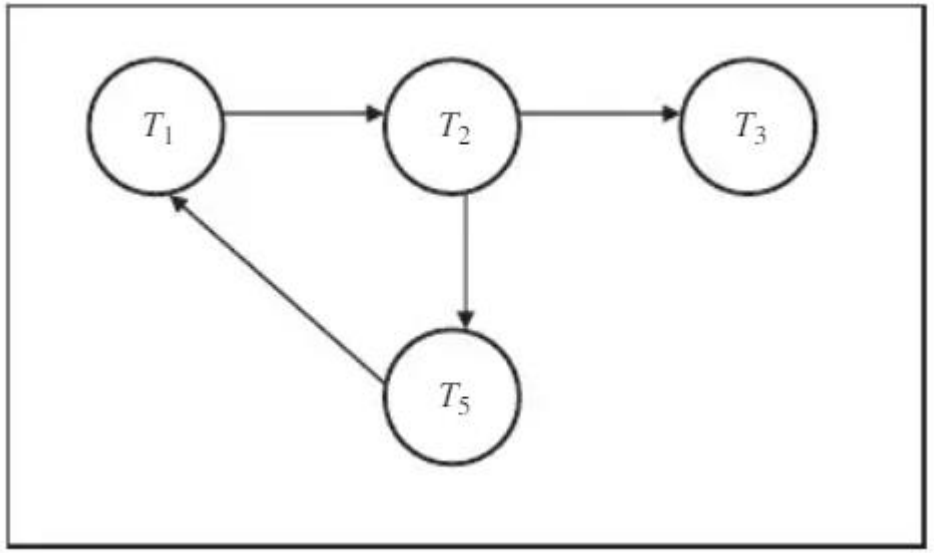
Site 2



Site 3

- The corresponding GWFG illustrate that a deadlock has occurred in the distributed system, although no deadlock has occurred locally.

DISTRIBUTED DEADLOCK DETECTION AND RECOVERY



- There are three different techniques for detecting deadlock situations in a distributed system:
  - Centralized deadlock detection,
  - Hierarchical deadlock detection and
  - Distributed deadlock detection

CENTRALIZED DEADLOCK DETECTION

- In Centralized Deadlock Detection method, a single site is chosen as Deadlock Detection Coordinator (DDC) for the entire distributed system.
- The DDC is responsible for constructing the GWFG for the system.
- Each lock manager in the distributed database system transmits its LWFG to the DDC periodically.
- The DDC constructs the GWFG from these LWFGs and checks for cycles in it.
- The occurrence of a global deadlock situation is detected if there are one or more cycles in the GWFG.
- The DDC must break each cycle in the GWFG by selecting the transactions to be rolled back and restarted to recover from a deadlock situation.
- The information regarding the transactions that are to be rolled back and restarted must be transmitted to the corresponding lock managers by the deadlock detection coordinator.

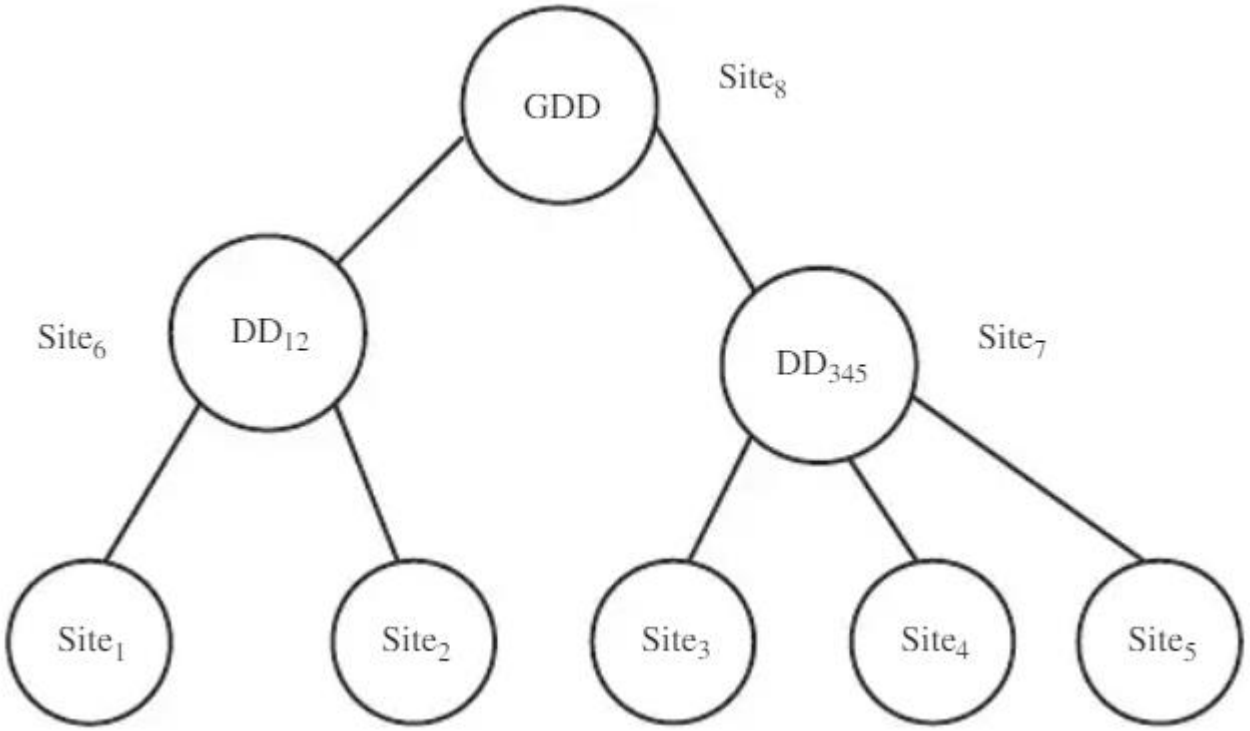
CENTRALIZED DEADLOCK DETECTION

- The centralized deadlock detection approach is very simple, but it has several drawbacks.
- This method is less reliable, as the failure of the central site makes the deadlock detection impossible.
- The communication cost is very high in this case, as other sites in the distributed system send their LWFGs to the central site.
- Another disadvantage of centralized deadlock detection technique is that false detection of deadlocks can occur, for which the deadlock recovery procedure may be initiated, although no deadlock has occurred.
- In this method, unnecessary rollbacks and restarts of transactions may also result owing to phantom deadlocks.

HIERARCHICAL DEADLOCK DETECTION

- Hierarchical deadlock detection method reduces the communication overhead of centralized deadlock detection method.
- With this approach, all sites in a distributed database system are organized into a hierarchy, and a complete tree of deadlock detectors is constructed instead of a single centralized deadlock detector.
- Each site in the distributed system sends its LWFG to the deadlock detection site above it (adjacent parent node) in the hierarchy.
- Thus, local deadlock detection is performed in the leaf nodes of the tree, whereas the non-leaf nodes are responsible for detecting any deadlock situation involving all its child nodes.

HIERARCHICAL DEADLOCK DETECTION



HIERARCHICAL DEADLOCK DETECTION

- The deadlock detector at $site_6$, namely DD_{12}, is responsible for detecting any deadlock situation involving its child nodes $site_1$ and $site_2$.
- Similarly, $site_3$, $site_4$ and $site_5$ send their LWFGs to $site_7$ and the deadlock detector at $site_7$ searches for any deadlocks involving its adjacent child nodes.
- A global deadlock detector exists at the root of the tree that would detect the occurrence of global deadlock situations in the entire distributed system.
- Here, the global deadlock detector resides at $site_8$ and would detect the deadlocks involving $site_6$ and $site_7$.

HIERARCHICAL DEADLOCK DETECTION

- The performance of the hierarchical deadlock detection approach depends on the hierarchical organization of nodes in the system.
- This organization should reflect the network topology and the pattern of access requests to different sites of the network.
- This approach is more reliable than centralized deadlock detection as it reduces the dependence on a central site; also it reduces the communication cost.
- However, its implementation is considerably more complicated, particularly with the possibility of site and communication link failures.
- In this approach, detection of false deadlocks can also occur.

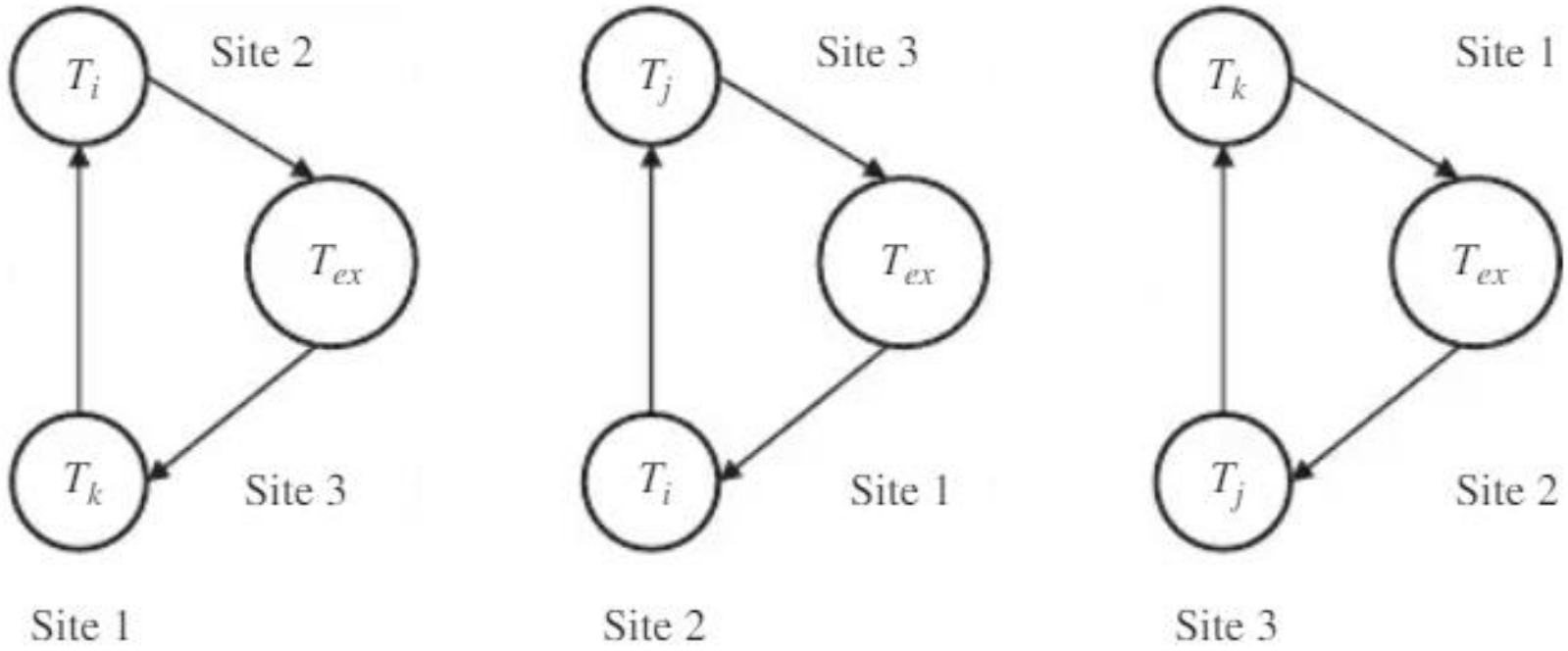
DISTRIBUTED DEADLOCK DETECTION

- In distributed deadlock detection method, a deadlock detector exists at each site of the distributed system.
- In this method, each site has the same amount of responsibility, and there is no such distinction as local or global deadlock detector.
- A LWFG is constructed for each site by the respective local deadlock detectors.
- An additional external node is added to the LWFGs, as each site in the distributed system receives the potential deadlock cycles from other sites.
- In the distributed deadlock detection algorithm, the external node Tex is added to the LWFGs to indicate whether any transaction from any remote site is waiting for a data item that is being held by a transaction at the local site or whether any transaction from the local site is waiting for a data item that is currently being used by any transaction at any remote site.

DISTRIBUTED DEADLOCK DETECTION

- For instance, an edge from the node Ti to Tex exists in the LWFG, if the transaction Ti is waiting for a data item that is already held by any transaction at any remote site.
- Similarly, an edge from the external node Tex to Ti exists in the graph, if a transaction from a remote site is waiting to acquire a data item that is currently being held by the transaction Ti at the local site.
- Thus, the local detector checks for two things to determine a deadlock situation.
- If a LWFG contains a cycle that does not involve the external node Tex, then it indicates that a deadlock has occurred locally and it can be handled locally.
- On the other hand, a global deadlock potentially exists if the LWFG contains a cycle involving the external node Tex.
- However, the existence of such a cycle does not necessarily imply that there is a global deadlock, as the external node Tex represents different agents.

DISTRIBUTED DEADLOCK DETECTION



DISTRIBUTED DEADLOCK DETECTION

- After detecting a deadlock in the system, an appropriate recovery protocol is invoked to resolve the deadlock situation.
- One or more transactions are chosen as victims, and these transactions, together with all their sub transactions, are rolled back and restarted.
- The major benefit of distributed deadlock detection algorithm is that it is potentially more robust than the centralized or hierarchical methods.
- Deadlock detection is more complicated in this method, because no single site contains all the information that is necessary to detect a global dead-lock situation in the system, and therefore substantial inter-site communication is required, which increases the communication overhead.
- Another disadvantage of distributed deadlock detection approach is that it is more vulnerable to the occurrence of false deadlocks than centralized or hierarchical methods.