CHAPTER 11

Deadlock Handling in Transaction Processing

Deadlock in Transaction Processing

It refers to a situation wherein Transactions Wait forever for accessing the Data Items locked by each other.

Necessary Conditions for Deadlock to Occur

There are four conditions, which must exist simultaneously, for a Deadlock to Occur:-

(1) **Mutual Exclusion** Some Data Items must be locked by some transactions in Exclusive Mode.

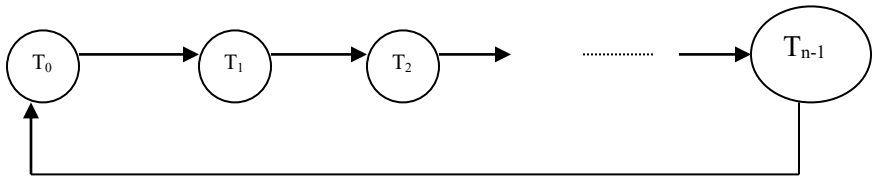
(2) **Hold & Wait** Some Transactions must be holding Exclusive Locks on some Data Items and at the same time must be requesting Exclusive Lock on some other data items currently locked by other transactions.

(3) **No Pre-emption** The data items locked exclusively by a Transaction can not be forcibly pre-empted. The Transaction will release the locks at its own will.

(4) **Cyclic Wait** The must exist a situation, wherein a set of n transactions say (T_0, T_1 , T_2, …… , T_{n-1}) are waiting in a cyclic manner for the data items locked by each other i.e. T_0 is waiting for some data item currently locked by T_1
T_1 is waiting for some data item currently locked by T_2
T_2 is waiting for some data item currently locked by T_3

|
|

T_{n-2} is waiting for some data item currently locked by T_{n-1}
T_{n-1} is waiting for some data item currently locked by T_0



Deadlock Prevention

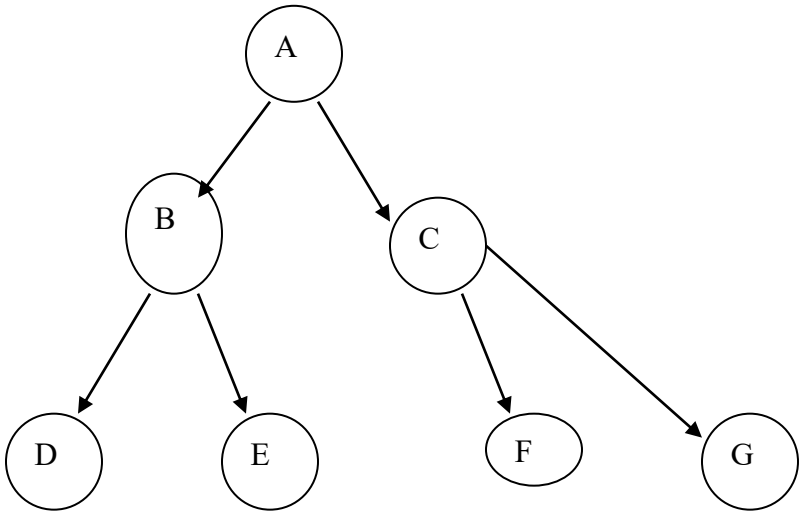
The Deadlocks can be prevented by imposing some restrictions on the sequence, in which a given set of data items, can be accessed by a Transaction. One such algorithm is explained below, which is graph based.

Graph Based Algorithm to Lock Resources

It works as follows:-

- A graph is drawn in which each node represents a Data Item
- A node will have only one parent.
- A transaction can Lock Data Items as follows:-
  - First Lock can be obtained on any Node
  - Any Subsequent lock can be obtained only on a Node whose parent is currently locked by the Transaction.
  - Lock on Root Node can be obtained only as a first lock; it cannot be locked subsequently.
  - If a Transaction violates the above protocol, it is forced to Roll-back.

Suppose, the Resource Graph for some environment is as follows:-



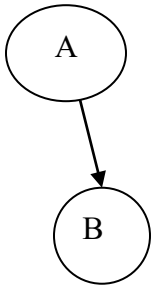
Suppose a Transaction T_i needs to access data items F, E, B in that order, it has to obtain locks in the following sequence

Lock-X (A);
Lock-X (C);
Lock-X (F);
Access (F);
Unlock (F);
Unlock (C);

Lock-X (B);
Lock-X (E);
Access(E);
Unlock (E);
Access (B);
Unlock (B);
Unlock (A);

How the above Algorithm helps to prevent Deadlocks is illustrated in the following Example:-

Let the protocol be that if both data items A and B are to be locked by a Transaction T_i then it must first lock data item A and then B, not the other way. If a transaction violates this protocol, then it must be rolled back.



Now, using the above Graph, the above Schedule (involving Deadlock) gets modified as follws:-

T_1	T_2	Lock Manager
Lock-X (A);		Grant-X (A, T_1)
	Lock-X (B);	Grant-X (B, T_2)
Read (A); Write (A); Lock-X (B);		T_1 has to wait, since B is currently Locked by T_2
	Read (B); Write (B); Lock-X (A);	At this point, it is Deadlock; But T_2 is violating the algorithm, since it is attempting to lock A After locking B. So, T_2 is rolled back and it is forced To release lock on data item B.

So, T_1 will proceed and Deadlock is Prevented.

Deadlock Detection & Recovery

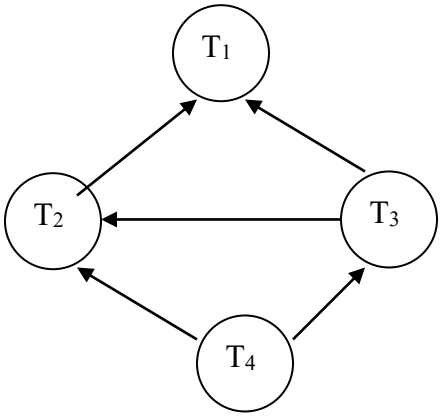
Deadlock Recovery will involve Roll-back of some of the transactions involved in the Deadlock; but before that a deadlock needs to be detected.

Deadlock Detection

Transaction Wait For Graph Method

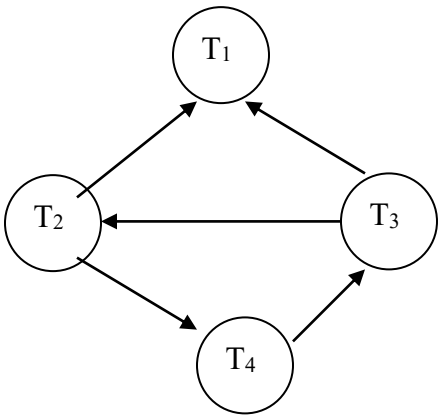
- A Directed Edge Graph is drawn, wherein each node represents a Transaction.
- A directed edge from T_i to T_j indicates that T_i is waiting for a data item currently locked by T_j.
- If there exists a cycle in the Graph, it indicates existence of a Deadlock; else there is no Deadlock.

Example 1



The above Graph has no cycle; thus there is no Deadlock.

Example 2



The above Graph has a cycle $T_2 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2$; thus there exists a Deadlock.

Deadlock Recovery

When a deadlock is detected, the system initiates recovery action, which involves roll-back of some of the transactions involved in the Deadlock. The rolled-back transactions would release the exclusive locks currently held by these transactions. So, the other transactions, which may be waiting for such locks, would get the awaited resources and would proceed; thus breaking the Deadlock.

The issues involved are:-

1. **Selection of Victims for Roll-Back**: The Criteria for selection of victim would take into consideration:-

   - **The amount of work already completed by the transaction**. Since this work will be undone during the roll-back, so a transaction which has completed lesser work should be a preferred as a victim.

   - **The amount of work yet to be completed**. This information is difficult to get. However, if this information is available, then the transaction, which is still farther from the end, should be preferred as a victim. A Transaction, which is near its end, should not be rolled-back.

   - **The Number of resources locked by the Transaction .** If a transaction with large number of resources locked by it is rolled back, then large number of resources will get free, which may meet the need of large number of waiting transaction.

2. **Roll Back the selected Transaction**. With the help of log file entries, the data items modified by this transaction will be reverted back to their old values. Also, the data items, currently locked by this transaction, will be unlocked. So, the other transactions, awaiting lock on such data items, will be able to proceed and the deadlock will be broken.

3. **Restart the Rolled-Back Transaction.** Once the deadlock is broken, the rolled-back transaction is restarted.

Implications of Deadlock Recovery

1. The work already performed by the rolled-back transaction is undone. In fact, undoing it also needs more work to be performed. So, system throughput gets affected adversely.

2. There is always a possibility that a Transaction may face roll-back again and again and may never get completed; thus facing ***starvation.***

In fact the Deadlock Detection also needs a lot of book-keeping and processing. This can be avoided by the following approach:-

Time-Out based approach for Deadlock Detection & Recovery

When a Transaction T_i requests lock on a data item Q, which may be currently locked by another transaction T_j then T_i is put to Wait State. Based upon history of Transaction processing on a system, in a given environment, it can always quantified time period δt within which the data item Q is likely to be free. This time period δt is explicitly specified in a system. If data item Q does not become available in time δt (Time-Out Condition) then it presumes existence of a Deadlock and Transaction T_i is rolled back automatically.

Exercises

Ex.11.1 (a) What are the necessary conditions for a deadlock to occur?
(b) Explain mechanisms to prevent deadlocks.

Ex.11.1 (a) Explain how lock-based protocols can lead to deadlocks.

(b) Will the following schedule end up having a deadlock?

T_1	T_2
Lock-X (A); Read (A)	
	Lock-X(B); Read (B)
Write(A); Lock-X(B);	
	Write(B); Lock-X(A);

(c) Explain various techniques to resolve the deadlock in the above schedule.