Visit: Whatisdbms.com for more Articles

What is Deadlock in DBMS, Prevention and Detection

What is Deadlock in DBMS, Prevention and Detection: If you are running a multi-process Database management system, one of the most feared complications is the Deadlock. It often arises if you're working in an environment where you share information and resources as one department's function depends on the integrity of results provided by the other department.

In this article, we are going to learn about deadlock in DBMS, Deadlock prevention and detection, Wait and die scheme, Wound-Wait Scheme.

The classic Deadlock situation

To understand this better, take the example of a set of transactions T1, T2, T3 and so on till Tn. For T1to complete a task, it requires a resource X. On the course of completion, the resource X is held by T1 while another transaction T2 is waiting for a resource Y which is held by T3. T3 waits for a resource Z to accomplish the task, which is held by T1.

This is a complicated situation where all the processes of a transaction are interrupted as they are waiting for the release of resources by fellow transactions. This is the classical deadlock model. Often, all transactions stuck in a deadlock are rebooted.

How to Detect a Deadlock || Deadlock Detection

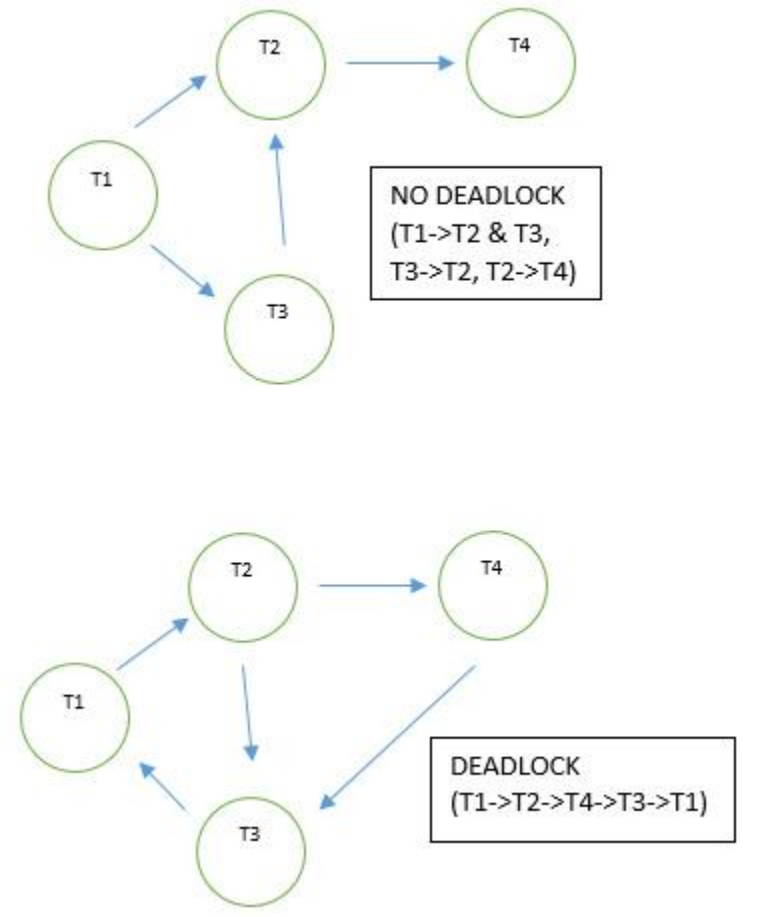
Early detection is very important for any organization transactional fluency. One of the simplest methods of deadlock detection is through a wait-for graph. The graph, generated by the system itself, represents each transaction that is in execution with a node.

For example, a transaction named T1 which is represented in the graphs below is trying to complete the task of completing an item X. However, the item is not available as it is locked by another transaction T2 which is a direct branch as shown below.

Now as T2 releases the item X which T1 was anticipating, the direct branch is removed from the wait-for graph. A state of deadlock on the wait-for graph would resemble a cycle of nodes and their branches. In such a situation, each transaction that is in execution is said to be in a deadlock.

For their detection, the source system needs to generate a wait-for graph and monitor it periodically through an algorithm such that it results in an error every time there is a deadlock situation.

Two wait-for graphs are shown below, one where there is no deadlock and another where there is a deadlock between all transactions.



Deadlock in DBMS

Factors to consider for the Deadlock Detection Algorithm

Before you invoke a deadlock detector algorithm, you need to estimate the frequency of a deadlock occurrence on your source systems occur and the number of transactions that are interrupted in the complication.

If your source systems face deadlocks often, ensure that the algorithm checks your systems frequently. As resources which are trapped in a deadlock are not available to other transactions which aren't, continuous monitoring by the algorithm is recommended.

How to Prevent a Deadlock || Deadlock Prevention

Preventing a Deadlock situation from arising isn't as simple as detecting it. It involves the inspection of all executable transactions of the system by the DBMS. Based on the complexity of

operations, an analysis is made and it is estimated whether certain transactions could lead to a deadlock situation.

If the DBMS detects that a possible deadlock situation might occur for specific transactions, then those transactions are terminated and rectifications are made. Deadlock prevention schemes predict future deadlock situations and use a mechanism of postponing certain transactions such that there is proper fluency of operations.

Wait-Die Scheme

This Deadlock prevention scheme as indicated by its name halts a transaction and terminates any further transactions occurring after such that there is no deadlock.

For example, if a transaction requires a resource that is already in use by another transaction,

- If a transaction is requesting a lock on the resource and is found to be of an older time stamp than the transaction which has the resource locked, it is not terminated.
- If a transaction is requesting a lock on the resource and is found to be of a younger time stamp than the transaction which has the resource locked, it is terminated.

Wound-Wait Scheme

This Deadlock prevention scheme as indicated by its name halts any further transaction that requests the resource. However, if a transaction of older time stamp places a request, it can terminate the transactions of younger time stamp status and reward the resource to the older one. This way there is no deadlock.

For example, if a transaction requires a resource that is already in use by another transaction,

- If a transaction is requesting a lock on the resource, it is not terminated and made to wait for the resource to be available.
- If a transaction is requesting a lock on the resource and is found to be of an older time stamp than the transaction which is in line for the resource, it can terminate the younger transaction and take over the resource. The time stamp with a younger time stamp is then rebooted.

The one thing that’s common in both the deadlock prevention schemes is that the transaction that enters the system late and has a younger time stamp is aborted.

So it was all about **What is Deadlock in DBMS, Prevention and Detection.** If you have any doubt then please comment below.