**1. Compare Wait Die and Wound Wait Method with an example.**

Ans: Wait-die scheme: It is a non-pre-emptive technique for deadlock prevention. When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp smaller than that of Tj (That is Ti is older than Tj), otherwise Ti is rolled back (dies)

For example:

Suppose that transaction T22, T23, T24 have time-stamps 5, 10 and 15 respectively. If T22 requests a data item held by T23 then T22 will wait. If T24 requests a data item held by T23, then T24 will be rolled back.

Wound-wait scheme: It is a pre-emptive technique for deadlock prevention. It is a counterpart to the wait-die scheme. When Transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp larger than that of Tj, otherwise Tj is rolled back (Tj is wounded by Ti)

For example:

Suppose that Transactions T22, T23, T24 have time-stamps 5, 10 and 15 respectively. If T22 requests a data item held by T23, then data item will be pre-empted from T23 and T23 will be rolled back. If T24 requests a data item held by T23, then T24 will wait.

**2. Compare and Contrast Centralized and Hierarchical deadlock detection methods.**

Ans: Deadlock detection is the most popular and suitable technique for deadlock management in database environment. In deadlock detection and recovery method, first it is detected whether any deadlock has occurred or not in the system. After detection of a deadlock situation in the system, the victim transaction is chosen and aborted in order to resolve the deadlock situation. Deadlock situations are detected by explicitly constructing the wait-for graph and searching it for cycles. A cycle in the wait-for graph indicates that deadlock has occurred and one transaction in the cycle is chosen as victim, which is aborted and restarted. To minimize the cost of restarting, the victim selection is usually based on the amount of data items used by each transaction in the cycle. The principal difficulty to implement deadlock detection in a distributed database environment is constructing global wait-for graph (GWFG) efficiently. In a distributed DBMS, a local wait-for graph (LWFG) for each local DBMS can be drawn easily, however, these local wait-for graphs are not sufficient to characterize all deadlock situations in the distributed system.

The local wait-for graph in each site is constructed in the usual manner using local transactions and data items into that particular site. A cycle in local wait-for graph170 indicates that a deadlock has occurred locally. The local wait-for graphs in figure 9.1illustrated that no deadlock has occurred locally in three different sites since there are no cycle in local wait-for graphs but it does not guarantee that no deadlock has occurred globally. In order to detect deadlock situation in the distributed system, it is necessary to construct a global wait-for graph from these different local wait-for graphs and searching it for cycles.