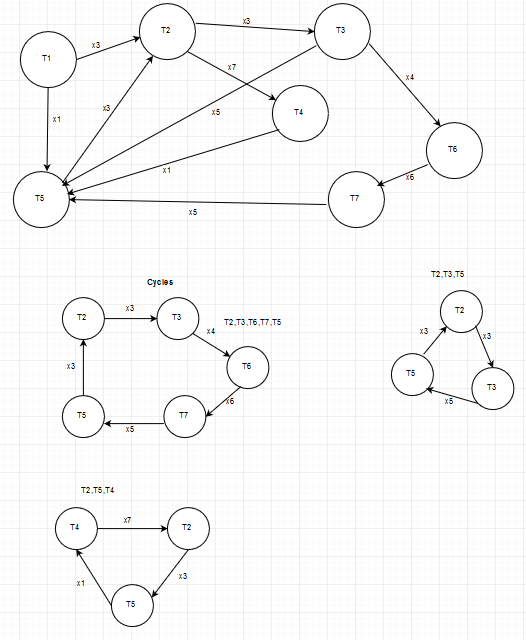
**Lab 9**

1. **[5] Produce a wait-for-graph for the following transaction scenario and determine whether deadlock exists.**

|  |  |  |
| --- | --- | --- |
| **Transaction** | **Data Items locked**  **by Transaction** | **Data items transaction is waiting for** |
| T1 | x2 | x1, x3 |
| T2 | x3, x10 | x7, x8 |
| T3 | x8 | x4, x5 |
| T4 | x7 | x1 |
| T5 | x1, x5 | x3 |
| T6 | x4, x9 | x6 |
| T7 | x6 | x5 |

Ans:-



**2. [5] Consider the following sequence of actions, listed in the order the actions are presented to the DBMS.**

**T1: R(X), T2: W(X), T2: W(Y), T3: W(Y), T1: W(Y), T3:R(Z), T3:W(Z), T1: Commit, T2: Commit, T3: Commit**

**Assume that the concurrency control mechanism is 2PL with “Wound-Wait” deadlock prevention strategy.   
Acquire locks as late as possible and release locks as early as possible. Waiting transactions continued and brought up to date as early as possible.  
Describe how the concurrency control mechanism handles the sequence of actions.**

**Ans:-**

* 2PL.
* Wound-Wait.
* Acquire locks as late as possible.
* Release locks as early as possible.
* waiting transactions continue and brought up to date as early as possible.

|  |  |
| --- | --- |
| t1 | T1 gets shared lock on X →T1: R(X) |
| t2 | T2 wants write lock on X, but since T2 is younger than T1, T2 **waits** for T1 |
| t3 | T3 gets exclusive lock on Y →T3:W(Y) |
| t4 | T1 wants write lock on Y, T3 has the lock and is younger than T1, T3 **abort** |
| t5 | T1 commits & releases lock X and Y |
| t6 | T2 acquires and gets the exclusive lock on X →T2:W(X) |
| t7 | T2 gets the exclusive lock on Y →T2:W(Y) |
| t8 | T2 commits & releases lock X and Y |
| t9 | T3 restarts and gets exclusive lock on Y →T3:W(Y) |
| t10 | T3 gets shared read lock on Z →T3:R(Z) |
| t11 | T3 gets exclusive write lock on Z →T3:W(Z) |
| t12 | T3 commits & releases lock Z |