SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : 23CST207 - DATABASE MANAGEMENT SYSTEMS

II YEAR / IV SEMESTER

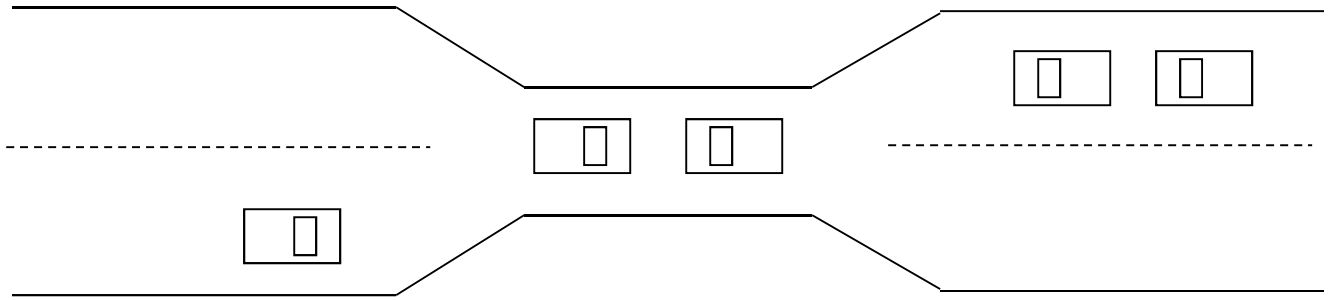
Unit 4- Transactions

Topic 7 : DEADLOCK

DEADLOCK

- Deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.
- Assume a set of transactions $\{T_0, T_1, T_2, ..., T_n\}$.
- T_0 needs a resource X to complete its task. Resource X is held by T_1, and T_1 is waiting for a resource Y, which is held by T_2. T_2 is waiting for resource Z, which is held by T_0.
- Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.

Bridge Crossing Example



- Traffic only in one direction.
- Each section of a bridge can be viewed as a resource.
- If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback).
- Several cars may have to be backed up if a deadlock occurs.
- Starvation is possible.

DEADLOCK CHARACTERISATION

NECESSARY CONDITIONS

ALL of these four **must** happen simultaneously for a deadlock to occur:

Mutual exclusion

One or more than one resource must be held by a process in a non-sharable (exclusive) mode.

Hold and Wait

A process holds a resource while waiting for another resource.

No Preemption

There is only voluntary release of a resource - nobody else can make a process give up a resource.

Circular Wait

Process A waits for Process B waits for Process C .... waits for Process A.

Find the Difference

- Two types of process

❖ Deadlock prevention

❖ Deadlock detection

➢ Deadlock Prevention

- The DBMS aggressively inspects all the operations, where transactions are about to execute.
- The DBMS inspects the operations and analyzes if they can create a deadlock situation.
- If it finds that a deadlock situation might occur, then that transaction is never allowed to be executed.
- There are deadlock prevention schemes that use timestamp ordering mechanism of transactions in order to predetermine a deadlock situation.

Two commonly used schemes

- **Wait-Die (WD): Non-preemptive**
  When *P* requests a resource currently held by *Q*, *P* is allowed to wait only if it is older than *Q*. Otherwise, *P* is rolled back (i.e., dies).

- **Wound-Wait (WW): Preemptive**
  When *P* requests a resource currently held by *Q*, *P* is allowed to wait only if *P* is younger than *Q*. Otherwise, *Q* is rolled back (releasing its resource). That is, *P* wounds *Q*.

- **Note:**
  - Both favor old jobs (1) to avoid starvation, and (2) since older jobs might have done more work, expensive to roll back.
  - Unnecessary rollbacks may occur.

- Two types of Scheme

1. Wait-Die Scheme
2. Wound-Wait Scheme

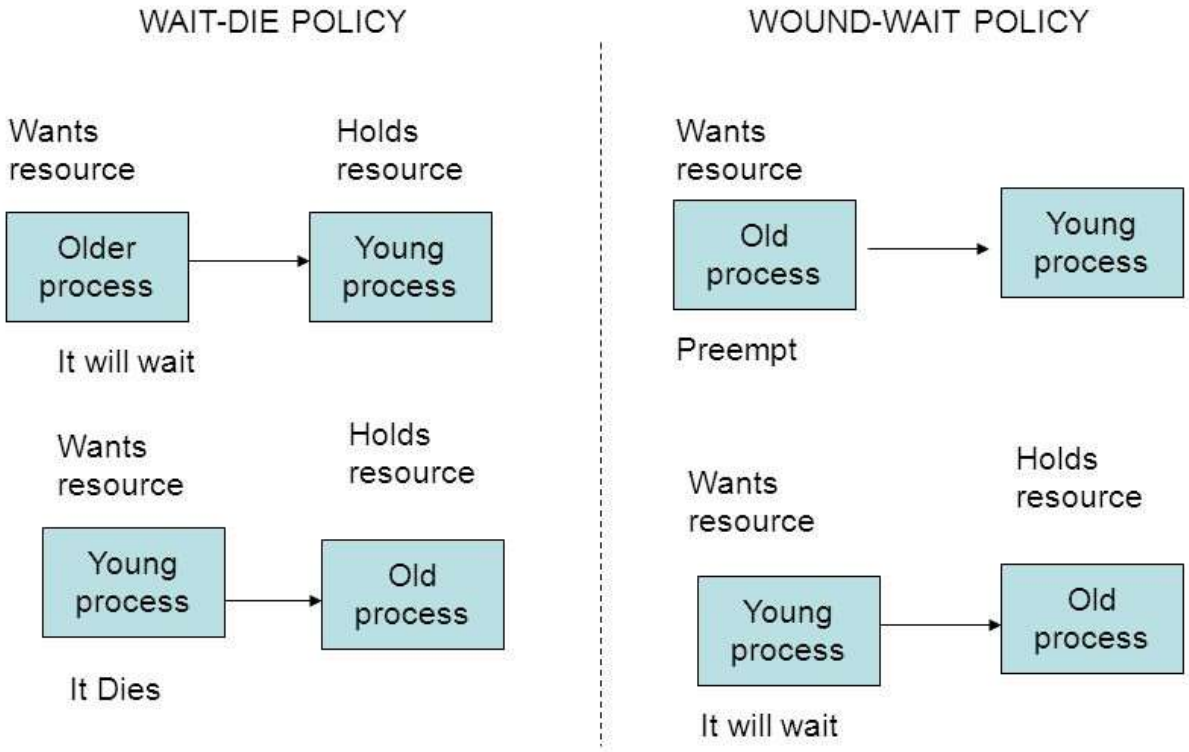
❑ Wait-Die Scheme

- In this scheme, if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one of the two possibilities may occur −
- If $TS(T_i) < TS(T_j)$ − that is T_i, which is requesting a conflicting lock, is older than T_j − then T_i is allowed to wait until the data-item is available.
- If $TS(T_i) > TS(t_j)$ − that is T_i is younger than T_j − then T_i dies. T_i is restarted later with a random delay but with the same timestamp.

❑ Wound-Wait Scheme

- In this scheme, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur −
- If $TS(T_i) < TS(T_j)$, then T_i forces T_j to be rolled back − that is $T_i$wounds T_j. T_j is restarted later with a random delay but with the same timestamp.
- If $TS(T_i) > TS(T_j)$, then T_i is forced to wait until the resource is available.
- This scheme, allows the younger transaction to wait; but when an older transaction requests an item held by a younger one, the older transaction forces the younger one to abort and release the item.

Wait die Vs Wound-wait



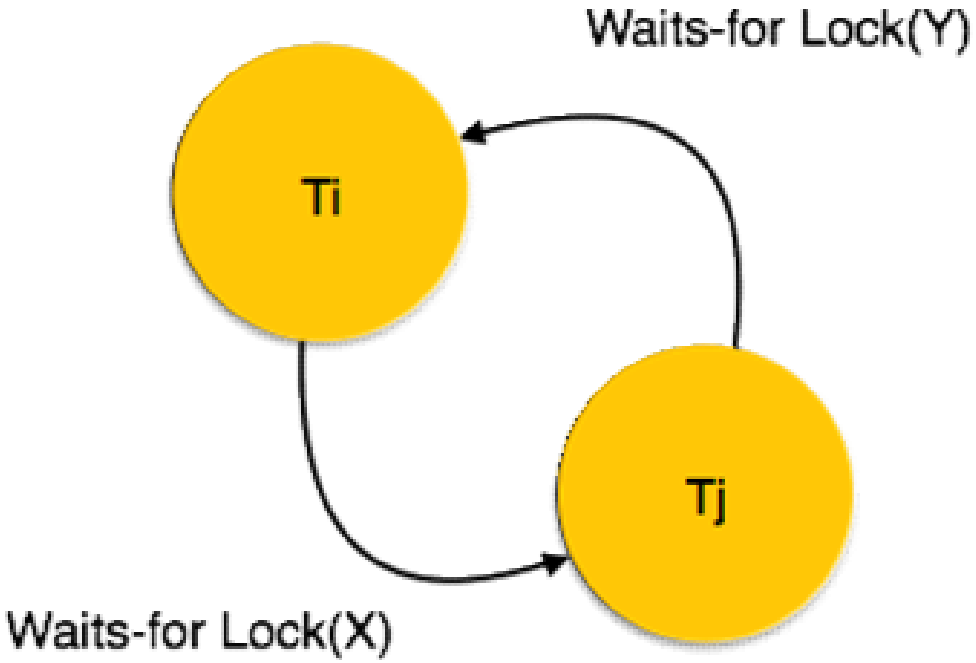
Deadlock Avoidance

- Instead, deadlock avoidance mechanisms can be used to detect any deadlock situation in advance. Methods like "wait-for graph" are available but they are suitable for only those systems where transactions are lightweight having fewer instances of resource. In a bulky system, deadlock prevention techniques may work well.

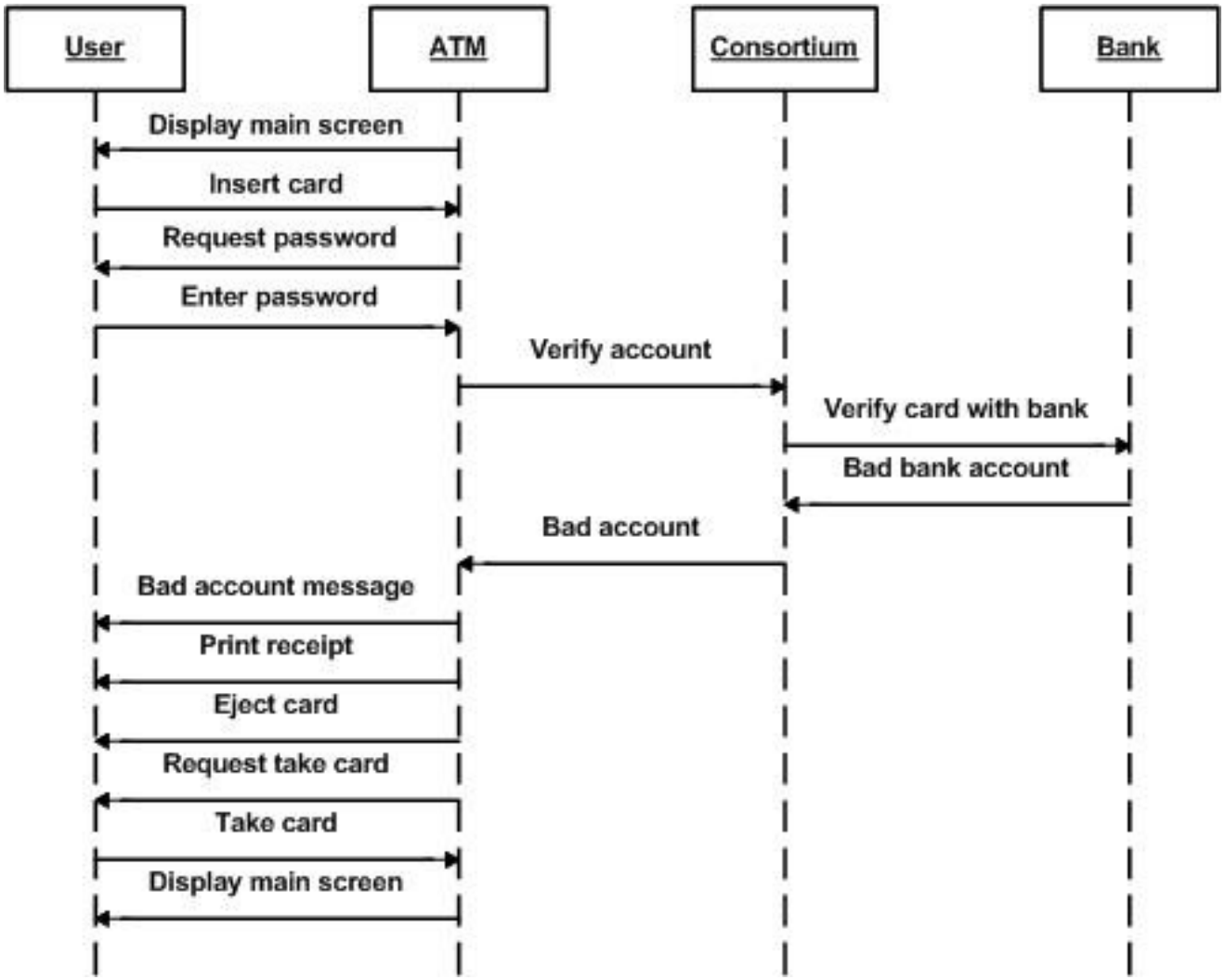
❖ Wait-for Graph

- This is a simple method available to track if any deadlock situation may arise.
- For each transaction entering into the system, a node is created.
- When a transaction T_i requests for a lock on an item, say X, which is held by some other transaction T_j, a directed edge is created from T_i to T_j. If T_j releases item X, the edge between them is dropped and T_i locks the data item.

- The system maintains this wait-for graph for every transaction waiting for some data items held by others. The system keeps checking if there's any cycle in the graph.



EXAMPLE ATM CARD WITHBANK



Evaluation

Filling in the blanks

- List out the two types of scheme

a)_____

b)_____

Answer

a) Wait-Die Scheme

b) Wound-Wait Scheme

THANK YOU