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-IOT Including CS&BCT

COURSE NAME : 19SB504 DATABASE MANAGEMENT SYSTEMS

III YEAR / V SEMESTER

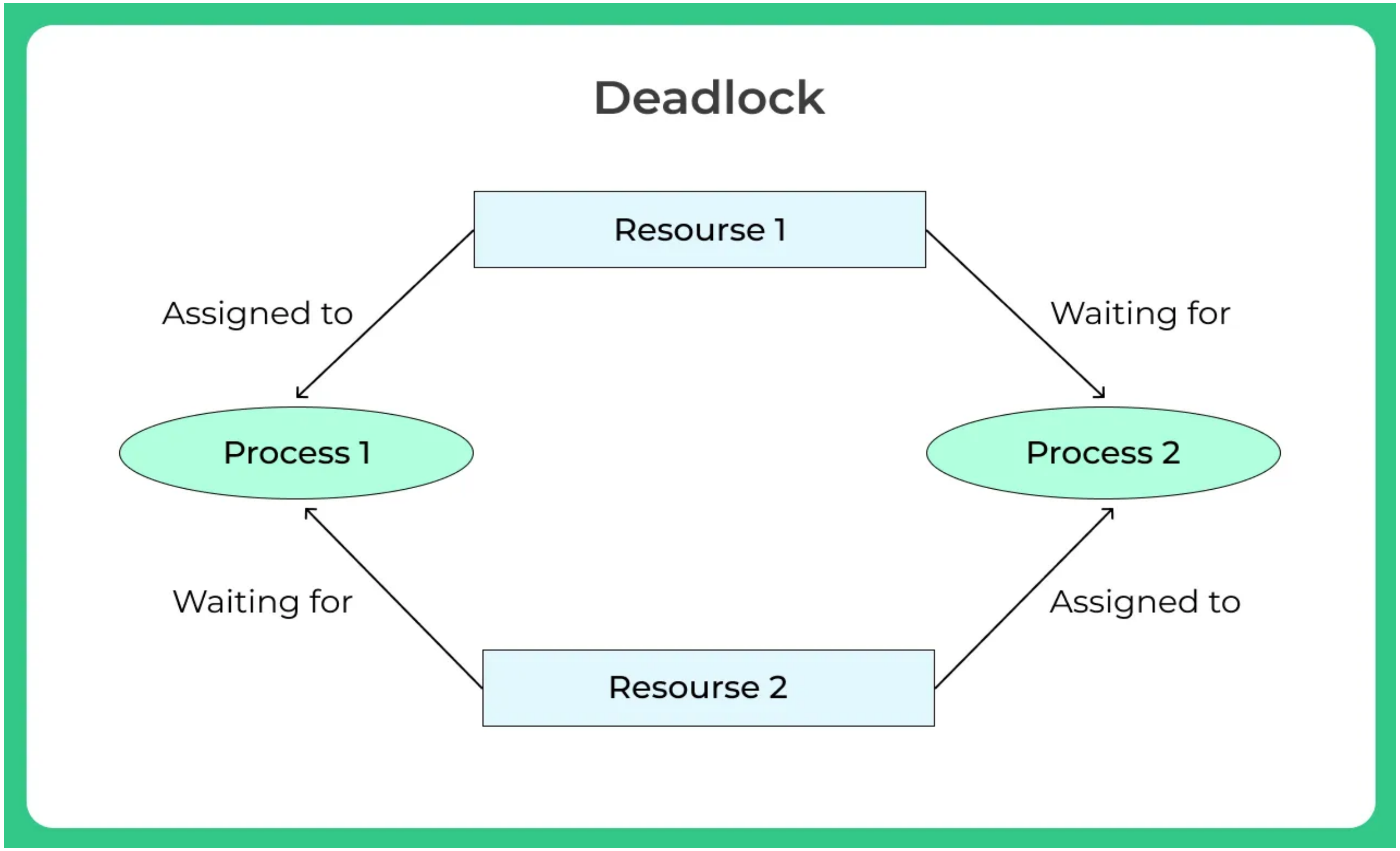
Unit IV- **TRANSACTIONS MANAGEMENT**

Topic :**DEADLOCK**

DEADLOCK

A deadlock in a Database Management System (DBMS) **is a situation** in which **two or more transactions or processes are unable to proceed** because they are **each waiting for a resource** that is held by another transaction within the set.

Deadlocks occur when there is a **circular chain of dependencies among transactions**, and each transaction in the cycle is waiting for a resource held by another transaction in the cycle.



Deadlock

Resourse 1

Assigned to

Waiting for

Process 1

Process 2

Waiting for

Assigned to

Resourse 2

Key characteristics of a deadlock in a DBMS

1. Mutual Exclusion
2. Hold and Wait
3. Circular Wait

Deadlock Detection:

When a transaction waits indefinitely to obtain a lock, The database management system should detect whether the transaction is involved in a deadlock or not.

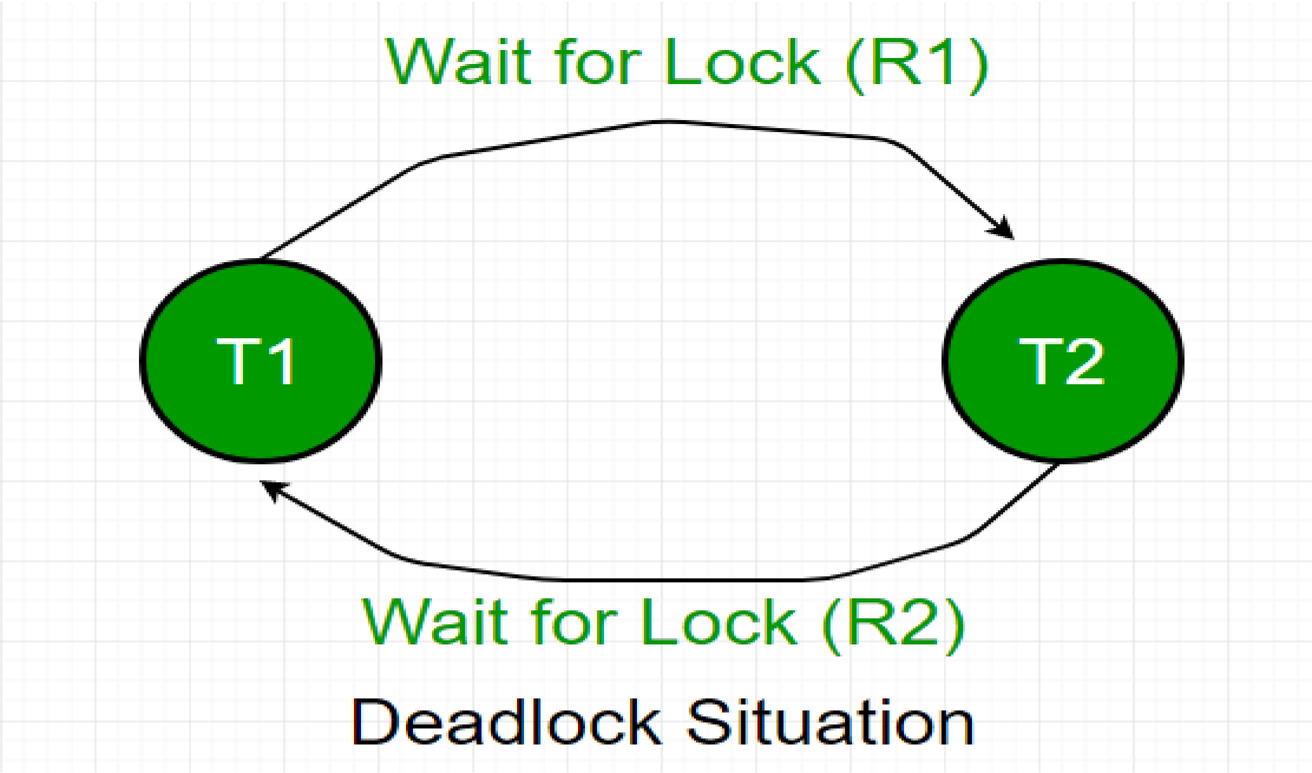
Wait-for-graph

It is one of the methods for **detecting the deadlock situation**.

This method is suitable for smaller databases.

In this method, a graph is drawn based on the transaction and its lock on the resource.

If the graph created has a closed loop or a cycle, then there is a deadlock.



Deadlock Situation

Example:

Imagine a simple **online hotel reservation system**. The system uses a database to keep track of available rooms and bookings. Two transactions, T1 and T2, are attempting to **book two rooms simultaneously**.

Transaction T1:

- ✓ T1 begins by trying to book Room A.
- ✓ It acquires a lock on Room A to ensure no one else can book it.
- ✓ Now, T1 needs Room B for a complete reservation, so it requests a lock on Room B.

Transaction T2:

- ✓ T2 begins by trying to book Room B.
- ✓ It acquires a lock on Room B to ensure no one else can book it.
- ✓ Now, T2 needs Room A for a complete reservation, so it requests a lock on Room A.

Now, here's the deadlock situation:

- ✓ Transaction T1 has a lock on **Room A** and is waiting for a lock on **Room B**.
- ✓ Transaction T2 has a lock on **Room B** and is waiting for a lock on **Room A**.

Deadlock prevention:

For a large database, the deadlock prevention method is suitable.

A deadlock can be prevented if the resources are allocated in such a way that a deadlock never occurs.

1. Timeout
2. Kill a Transaction
3. Wait-Die Scheme
4. Wound Wait Scheme

In a DBMS, when a deadlock occurs, the system needs to take **action to resolve it**, typically by choosing one of the following methods:

Timeout:

The DBMS sets a timeout for transactions. If a transaction doesn't get the necessary locks within the timeout, it's **aborted and rolled back**.

Kill a Transaction:

The DBMS may choose to **terminate one of the conflicting transactions** to **break the deadlock** and allow the others to proceed.

Wait-Die or Wound-Wait Schemes:
These are two strategies that determine which transaction should be aborted based on their **priorities or timestamps**.
The **younger transaction** (with less progress) may be aborted in the "Wait-Die" scheme, while the **older one is aborted** in the "Wound-Wait" scheme.

Prevention: DBMS can use various techniques, like strict **two-phase locking**, to prevent deadlocks from happening in the first place.

Wait – Die	**Wound -Wait**
It is based on a **non-preemptive technique**.	It is based on a **preemptive technique.**
In this, **older transactions must wait** for the younger one to release its data items.	In this, **older transactions never wait** for younger transactions.
The number of **aborts and rollbacks is higher** in these techniques.	In this, the **number of aborts and rollback is lesser.**

Thank You…….