Case Study Report: Deadlock Detection Using Banker's Algorithm

Project Title: Deadlock Detection Using Banker's Algorithm

Language Used: Python

Author: Neelam

Date: JAN, 01, 2026

1. Introduction

Deadlock is a serious problem in operating systems where two or more processes are permanently blocked because each process is waiting for a resource held by another. When deadlock occurs, system resources are not utilized efficiently, leading to performance degradation, application failure, or complete system halt.

This case study focuses on understanding deadlock through a real-world database deadlock scenario and explains how Banker's Algorithm can be used to detect and avoid deadlock situations.

2. Objectives of the Case Study

- Understand the concept of deadlock
- Analyze a real-world database deadlock scenario
- Identify deadlock conditions
- Understand Banker's Algorithm
- Detect safe and unsafe states

3. Key Concepts and Definitions

Deadlock: A condition where processes wait indefinitely for resources.

Resource: Any hardware or software entity required by a process.

Safe State: A state where all processes can finish execution.

Unsafe State: A state that may lead to deadlock.

Banker's Algorithm: A deadlock avoidance algorithm.

4. Deadlock Scenario

Transaction T1 locks Table A and requests Table B.

Transaction T2 locks Table B and requests Table A.

Both transactions wait forever, causing deadlock.

5. Causes of Deadlock

- Mutual Exclusion

- Hold and Wait

- No Preemption

- Circular Wait

6. Deadlock Detection Using Banker's Algorithm

Banker's Algorithm checks system safety before allocating resources.

Need = Maximum - Allocation

7. Impact of Deadlock

- Application hangs

- Performance degradation

- Data inconsistency

8. Deadlock Prevention and Avoidance

- Resource ordering

- Avoid hold and wait

- Timeouts

- Deadlock detection and recovery

- Banker’s Algorithm

9. Conclusion

Deadlock can severely affect system performance. Banker’s Algorithm helps in detecting unsafe states and improving system stability.