Visualizing Deadlock with Transaction Logs and Waits-For Graph

1 Introduction

A deadlock is a situation in a database or concurrent system where two or more transactions are unable to proceed because each is waiting for a resource that another holds, forming a cycle. This document illustrates a deadlock scenario using a transaction log and a waits-for graph.

2 Deadlock Scenario

Consider two transactions, T_1 and T_2, operating on two resources, R_1 and R_2 (e.g., database rows). The transactions execute as follows:

- T_1 locks R_1 and attempts to lock R_2.
- T_2 locks R_2 and attempts to lock R_1.

This creates a deadlock because T_1 waits for R_2 (held by T_2), and T_2 waits for R_1 (held by T_1).

3 Transaction Log

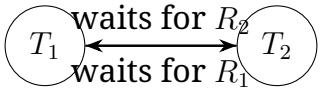
The transaction log below shows the sequence of operations leading to the deadlock:

Time	Operation
t_1	T_1: Lock R_1
t_2	T_2: Lock R_2
t_3	T_1: Request lock on R_2 (waits for T_2)
t_4	T_2: Request lock on R_1 (waits for T_1)

At t_4, a deadlock occurs because neither transaction can proceed.

4 Waits-For Graph

The waits-for graph visually represents the deadlock. Nodes are transactions (T_1, T_2), and directed edges show waiting relationships (e.g., $T_1 \rightarrow T_2$ means T_1 waits for T_2).



The cycle $T_1 \rightarrow T_2 \rightarrow T_1$ indicates a deadlock, as each transaction is waiting for the other to release a resource.

5 Conclusion

The transaction log and waits-for graph together illustrate how a deadlock arises when transactions form a cyclic dependency on resources. Database systems typically detect such cycles using algorithms like cycle detection in waits-for graphs and resolve them by aborting one or more transactions.