Indian Institute of Information Technology Surat

Lab Report on
Advanced Database Management (CS 604) Practical

Submitted by

[RAHUL KUMAR SINGH] (UI21CS44)

Course Faculty

Mr. Rishi Sharma

Department of Computer Science and Engineering

Indian Institute of Information Technology Surat

Gujarat-394190, India

Jan-2024

Lab No: 5

Aim: To implement Deadlock Detection Algorithm for Distributed Database using Wait-for Graph to check for Deadlock.

Description:

- DetectDeadlock Procedure:
  - Creates a temporary table for wait-for graph.
  - Populates wait-for graph with data from deadlock_info.
- DepthFirstSearch Procedure:
  - Simulates stack for DFS using a temporary table.
  - Detects cycles by recursively traversing wait-for graph.
- DetectDeadlock Procedure Modification:
  - Inserts multiple rows into dfs_stack from wait_for_graph where requesting_node=start_node.

Source Code:

Table:

```
CREATE TABLE Deadlock_Info (
   transaction_id INT PRIMARY KEY AUTO_INCREMENT,
   requesting_node INT,
   holding_node INT
);
```

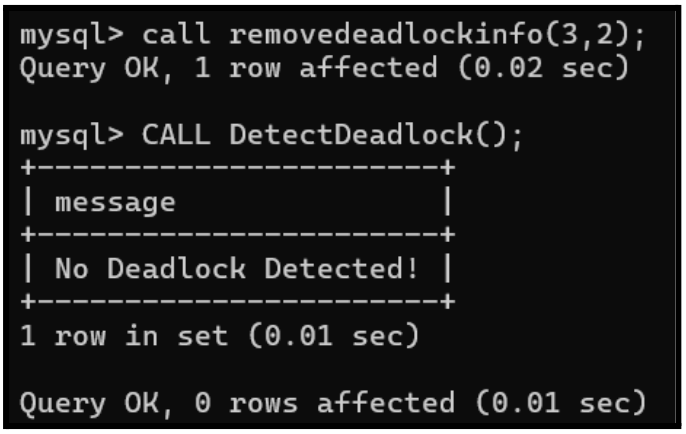
Detect Deadlock Procedure:

```
DELIMITER //
CREATE PROCEDURE DetectDeadlock()
BEGIN
   DECLARE result INT DEFAULT 0;
   DECLARE temp INT DEFAULT 0;
   DECLARE start_node INT;
   DECLARE current_node INT;
   DECLARE done INT DEFAULT 0;
   CREATE TEMPORARY TABLE IF NOT EXISTS wait_for_graph (
      requesting INT,
      holding INT
   );
   INSERT INTO wait_for_graph SELECT requesting_node, holding_node FROM deadlock_info;
   SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
   CREATE TEMPORARY TABLE IF NOT EXISTS distinct_nodes (
      node INT
   );
   INSERT INTO distinct_nodes
   SELECT DISTINCT requesting
   FROM wait_for_graph;
   WHILE (SELECT COUNT(*) FROM distinct_nodes) > 0 DO
      SELECT node INTO start_node FROM distinct_nodes ORDER BY node LIMIT 1;
      DELETE FROM distinct_nodes WHERE node = start_node;
      SELECT CONCAT("Start Node: ",start_node) as message;
      SELECT result;
      CALL DepthFirstSearch(start_node, start_node, temp);
      IF temp = 1 THEN
         SET result = 1;
      END IF;
      DELETE FROM wait_for_graph;
      INSERT INTO wait_for_graph SELECT requesting_node, holding_node FROM deadlock_info;
   END WHILE;
   SELECT result;
   IF result = 0  THEN
```

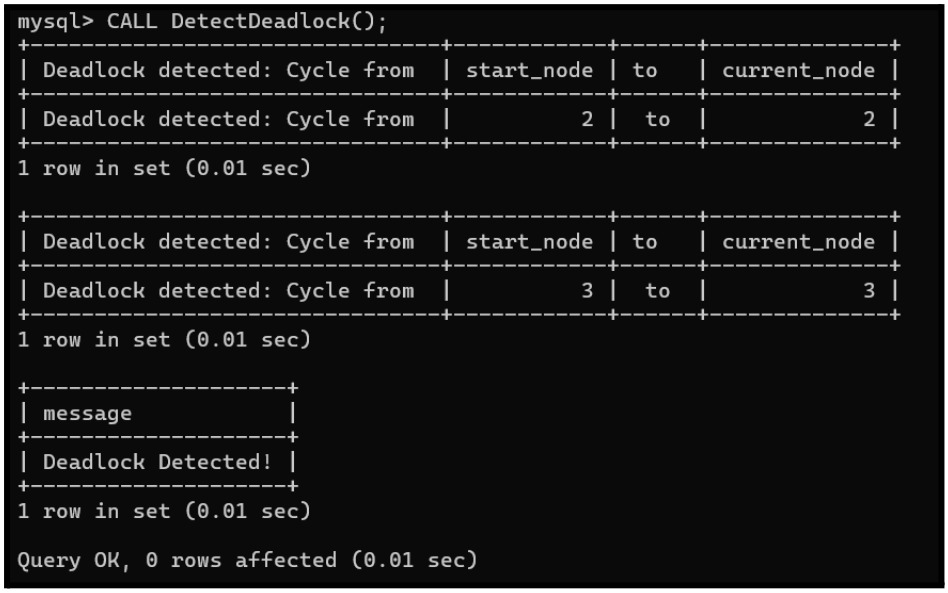
```
    SELECT "No Deadlock Detected!" as message;
  ELSE
    SELECT "Deadlock Detected!" as message;
  END IF;
  DROP TEMPORARY TABLE IF EXISTS wait_for_graph;
  DROP TEMPORARY TABLE IF EXISTS distinct_nodes;
END //
DELIMITER ;
```

Output:

In case of No Deadlock:



In case of Deadlock:



Conclusion:

- Detects deadlocks in a distributed database using a wait-for graph.
- Implemented depth-first search within MySQL stored procedures.
- Procedures manage deadlock information, simulate DFS, and execute deadlock detection.
- MySQL limitations for complex algorithms; use external languages for efficiency.
- Deadlock detection results in cycles which means careful consideration using it.

.