LAB MANUAL 05

Course: Advanced Database Systems
Duration: 2 Hours

Course Instructor: Khairum Islam

Lab Manual and Plan

Lab Title: Transactions and Concurrency Control: Deadlock Detection and Resolution in Oracle 10g Express Edition

Objective: To understand and experiment with database transactions, concurrency control, and deadlock detection mechanisms. This lab will use Oracle 10g Express Edition to demonstrate how deadlocks occur and how they can be resolved programmatically.

Lab Plan

Session 1: Introduction to Transactions and Concurrency

1. **Objective:**
    - Understand the concept of database transactions and their properties (ACID).
    - Explore concurrency control mechanisms in relational databases.
2. **Activities:**
    - Brief theory session on transactions, isolation levels, and locking mechanisms.
    - Hands-on with simple transaction examples using SQL commands: BEGIN, COMMIT, and ROLLBACK.
    - Explore concurrency control issues such as dirty reads, non-repeatable reads, and phantom reads.
3. **Outcome:**
    - Familiarity with transactions and basic concurrency control issues.

Session 2: Deadlocks in Transactions

1. **Objective:**
    - Understand what a deadlock is and how it occurs in database systems.
    - Experiment with transactions that intentionally create a deadlock.
2. **Activities:**
    - Set up two tables (e.g., Table_A and Table_B) and populate them with sample data.
    - Create two transactions that try to lock resources in opposite order to simulate a deadlock.
    - Use Oracle's V$LOCK and V$SESSION views to detect deadlocks.
3. **Outcome:**
    - Ability to identify and replicate deadlocks in a controlled environment.

Session 3: Resolving Deadlocks

1. **Objective:**
    - Learn strategies to resolve and prevent deadlocks in database systems.
    - Understand Oracle's internal deadlock detection mechanism.
2. **Activities:**
    - Write and execute a PL/SQL script that detects deadlocks.
    - Modify transaction order to resolve the deadlock.
    - Discuss database design and coding strategies to minimize the chances of deadlocks (e.g., consistent resource ordering, proper indexing, and partitioning).
3. **Outcome:**
    - Practical knowledge of resolving deadlocks and improving database application design.

Lab Manual

Prerequisites:

- Oracle 10g Express Edition installed.
- Basic understanding of SQL and PL/SQL.

Step-by-Step Procedure:

- **Setup:**
    - Create two tables for the experiment:

```
CREATE TABLE Table_A (
   id NUMBER PRIMARY KEY,
   data VARCHAR2(100)
);

CREATE TABLE Table_B (
   id NUMBER PRIMARY KEY,
   data VARCHAR2(100)
);

INSERT INTO Table_A VALUES (1, 'Sample A');
INSERT INTO Table_B VALUES (1, 'Sample B');

COMMIT;
```

- **Simulate a Deadlock:**
    - Open two separate SQL sessions.
    - In Session 1, execute the following commands:

```
BEGIN;
UPDATE Table_A SET data = 'Updated A' WHERE id = 1;
```

- In Session 2, execute the following commands:

BEGIN;

UPDATE Table_B SET data = 'Updated B' WHERE id = 1;

- Go back to Session 1 and execute:

UPDATE Table_B SET data = 'New Value B' WHERE id = 1;

- Finally, in Session 2, execute:

UPDATE Table_A SET data = 'New Value A' WHERE id = 1;

- **Detect Deadlock:**
  - Use Oracle's V$LOCK and V$SESSION views:

SELECT * FROM V$LOCK;

SELECT * FROM V$SESSION WHERE SID IN (SELECT SID FROM V$LOCK);

- **Resolve Deadlock:**
  - Terminate one of the sessions using:

ALTER SYSTEM KILL SESSION '<sid>,<serial#>';

- Adjust transaction order to prevent deadlock.

- **Query the V$SESSION view to find the SID and Serial# for the session causing the issue:**

```
SELECT SID, SERIAL#, USERNAME, STATUS, MACHINE
FROM V$SESSION
WHERE BLOCKING_SESSION IS NOT NULL;
```

Alternatively, to identify all active sessions:

SELECT SID, SERIAL#, USERNAME, STATUS, MACHINE

FROM V$SESSION;

Otherwise,

SELECT SID, SERIAL#, STATUS, USERNAME, MACHINE

FROM V$SESSION

WHERE STATUS = 'ACTIVE';

Use the SID and Serial# values from the query in the ALTER SYSTEM KILL SESSION command. For example:

ALTER SYSTEM KILL SESSION '123,456';

If the session does not terminate immediately, you can use the IMMEDIATE keyword to force termination:

ALTER SYSTEM KILL SESSION '123,456' IMMEDIATE;

Discussion Questions:

1. Why did the deadlock occur in the given scenario?
2. How does Oracle detect and resolve deadlocks internally?
3. What are some design strategies to avoid deadlocks?
