Week 2 LAB 1 - Practice: 1

1. **Question**
   i. Create a Database (Assume the database name as per your choice)
   ii. Create a table for a user account with attribute ID, Name, and Balance in a Database.
   iii. Insert the data in the table.
   iv. Create a banking transaction that transfers 100,000VND from one account_id to another account_id. (such as account_id 001, 002)
      a. Deduct amount from Account such as (A).
      b. Add amount to Account such as (B).
   v. Commit if both operations succeed; rollback if any error occurs
2. **Question**
   i. Create a table user Accounts if not created with columns AccountID, Name, and Balance.
   ii. Insert sample data into the table.
   iii. Using SQL transactions, demonstrate the effects of the following isolation levels:
      a. READ UNCOMMITTED
      b. READ COMMITTED
      c. REPEATABLE READ
      d. SERIALIZABLE
3. **Question**
   - Write a SQL transaction to transfer 50,000 VND from Account (001) to Account (002), subject to the following conditions:
      a. Both Account 001 and Account 002 must exist
      b. Account 001 must have a sufficient balance (at least 500,000 VND)
      c. If all conditions are satisfied, commit the transaction
      d. If any condition fails, roll back the transaction
4. **Question**
   - Create an Orders table. Then create a stored procedure that processes multiple orders within a single transaction. If any order fails, roll back the entire transaction.
      - Create the Orders table
      - Create the stored procedure
      - Execute (call) the procedure
5. **Question**
   i. Using two concurrent transactions (Session A and Session B), demonstrate the following concurrency problems:
      a. Lost Update
      b. Dirty Read
      c. Non-Repeatable Read
      d. Phantom Read

ii. Using appropriate transaction isolation levels or locking mechanisms, write SQL statements to prevent the following concurrency problems:
    a. Lost Update
    b. Dirty Read
    c. Non-Repeatable Read
    d. Phantom Read

Remember For each case:

- Use appropriate SQL statements
- Clearly show the actions in Session A and Session B
- Explain the result observed

6. Question

- Assume the table Accounts (Account ID, Name, Balance) already exists and contains data. Multiple transactions access the table concurrently.

i. Shared Lock (Read Lock): Write SQL statements for Transaction T1 to:
    a. Read the balance of AccountID = 101
    b. Prevent other transactions from updating the row while it is being read

ii. Exclusive Lock (Write Lock): Write SQL statements for Transaction T2 to:
    a. Update the balance of AccountID = 101
    b. Ensure no other transaction can read or write the row during the update

iii. Write SQL statements to demonstrate Two-Phase Locking while transferring 100.000VND from AccountID = 101 to AccountID = 102.

Conditions:

a. All required locks must be acquired before any lock is released
b. Commit the transaction after both updates

iv. Write SQL statements for two concurrent transactions that prevent the lost update problem using explicit locking.

7. Question

Assume the table Accounts(AccountID, Balance) already exists and contains data. Two transactions run concurrently on the database.

a. Write SQL statements for Transaction T1 and Transaction T2 that will result in a deadlock situation.
b. Write SQL statements to identify a deadlock using database system commands or queries.
c. Rewrite the SQL statements from part (a) to prevent the deadlock by ensuring a consistent locking order.
d. Write SQL statements that avoid deadlock by using a lock wait timeout.

8. Question: Assume the table Accounts(AccountID, Balance) already exists and contains data.

Write SQL statements to:

a. Start a transaction
b. Deduct 5,000VND from AccountID = 101
c. Add 5,000VND to AccountID = 102
d. Commit the transaction

Transaction with ROLLBACK (Simulated Failure): Write SQL statements to-

a. Start a transaction
b. Deduct 5,000 units from AccountID = 101
c. Simulate a failure before crediting AccountID = 102
d. Roll back the transaction

Recovery After System Failure

Write SQL statements to:

a. Perform a transaction that is not committed
b. Show that after a rollback (simulating recovery), the database returns to a consistent state.

SAVEPOINT-Based Recovery: Write SQL statements to-

a. Start a transaction
b. Update two accounts
c. Create a SAVEPOINT after the first update
d. Roll back only the second update using SAVEPOINT
e. Commit the transaction