DBMS LAB EXAM – P Batch

Date: 05.03.2026

1. To implement and demonstrate the **ACID properties** (Atomicity, Consistency, Isolation, Durability) using a Banking Transaction Database. A bank maintains customer accounts in a database. When a customer transfers money from one account to another, the system must ensure:

- The amount is deducted from the sender.
- The amount is credited to the receiver.
- The transaction is fully completed or fully rolled back.
- Data integrity is preserved even in case of system failure.

   I. Simulate a transaction where ₹2000 is transferred from Account 101 to Account 102.

   II. Attempt to deduct ₹20000 from Account 101 (insufficient balance).
   Ensure database constraints prevent invalid state.

   III. Open two sessions. In Session 1: Start transaction and update balance but do not commit. In Session 2: Try to read updated balance.

2. An e-commerce company stores product details in a database. The system frequently searches products by:

- Product Name
- Category
- Price

   Implement single and composite indexes.

3. A college stores student course details in a single table as shown below.

Student_ID	Student_Name	Courses	Instructor	Instructor_Phone
101	Arun	DBMS, OS	Kumar, Ravi	9876, 9123
102	Bala	DBMS	Kumar	9876

- Create UNF table and identify anomalies.
- Convert to 1NF and insert data.
- Decompose into 2NF tables.
- Convert into 3NF.
- Justify whether schema satisfies BCNF.
- Identify insertion, deletion, update anomalies at each stage.

4. Perform Deadlock detection

```
CREATE TABLE Accounts (
   Acc_No NUMBER PRIMARY KEY,
```

```
    Name VARCHAR2(50),
    Balance NUMBER
);

INSERT INTO Accounts VALUES (101, 'Arun', 10000);
INSERT INTO Accounts VALUES (102, 'Bala', 8000);

COMMIT;
```

Open **two SQL sessions** in Oracle SQL Developer.

Session 1

```
SET AUTOCOMMIT OFF;

UPDATE Accounts
SET Balance = Balance - 1000
WHERE Acc_No = 101;
```

Session 2

```
SET AUTOCOMMIT OFF;

UPDATE Accounts
SET Balance = Balance - 500
WHERE Acc_No = 102;
```

⚠ Do NOT commit.

Now Create Circular Wait

Back to Session 1

```
UPDATE Accounts
SET Balance = Balance + 1000
WHERE Acc_No = 102;
```

(Session 1 waits because Session 2 locked 102)

Back to Session 2

```
UPDATE Accounts
SET Balance = Balance + 500
WHERE Acc_No = 101;
```

Session	Operation	Result
Session 1	Update Acc 101	Success
Session 2	Update Acc 102	Success
Session 1	Update Acc 102	Waiting
Session 2	Update Acc 101	ORA-00060 Error

5. Demonstrate the different join operations – Inner join, Left outer, Right Outer, Full outer, Self join for the following tables.

- Students
- Departments
- Courses
- Enrollments

Spot Questions

1. A university maintains a database to manage **students, courses, and enrollments**. When a student enrolls in a course, the system should automatically record the enrollment and update the total number of students enrolled in that course.

Database Tables

1. **STUDENT**
    - Student_ID (Primary Key)
    - Student_Name
    - Department
2. **COURSE**
    - Course_ID (Primary Key)
    - Course_Name
    - Credits
    - Total_Enrolled
3. **ENROLLMENT**
    - Enrollment_ID (Primary Key)
    - Student_ID (Foreign Key)
    - Course_ID (Foreign Key)
    - Semester

Write SQL queries using JOIN to perform the following:

a) Display Student Name and Course Name for all enrollments.

b) Display Student Name, Department, Course Name, and Credits using appropriate joins.

c) List all students who enrolled in the DBMS course.

d) Display courses with the number of students enrolled.

e) Show students who have not enrolled in any course using LEFT JOIN.

f) Create a trigger that automatically updates the Total_Enrolled column in the COURSE table whenever a new enrollment is inserted.

• When a new record is inserted into ENROLLMENT, The Total_Enrolled value in COURSE should increase by 1.

g) Insert a new enrollment record and verify that the trigger updates the COURSE table automatically.