



Wolaita Sodo University

School of Informatics

Department of Computer Science

Course: Advanced Database System

PROJECT TITLE: *bank management system*

## GROUP - 6

NAME

ID NO

1. ABRAHAM BELAY ----- UGR/91385/16

2. REDIET ELIAS ----- UGR/92137/16

3. SEBLE ASHENAFI ----- UGR/92177/16

4. MEBRATU BELETE ----- UGR/92690/16

5. NETSANET ENDALE ----- UGR/92115/16

## INTRODUCTION

- ✓ **Transaction Management:** Used BEGIN, COMMIT, ROLLBACK, and SAVEPOINT to control execution flow and ensure atomicity.
- ✓ This ensures that either both updates happen, or none (atomicity). If an error occurs between updates, a rollback can be triggered.

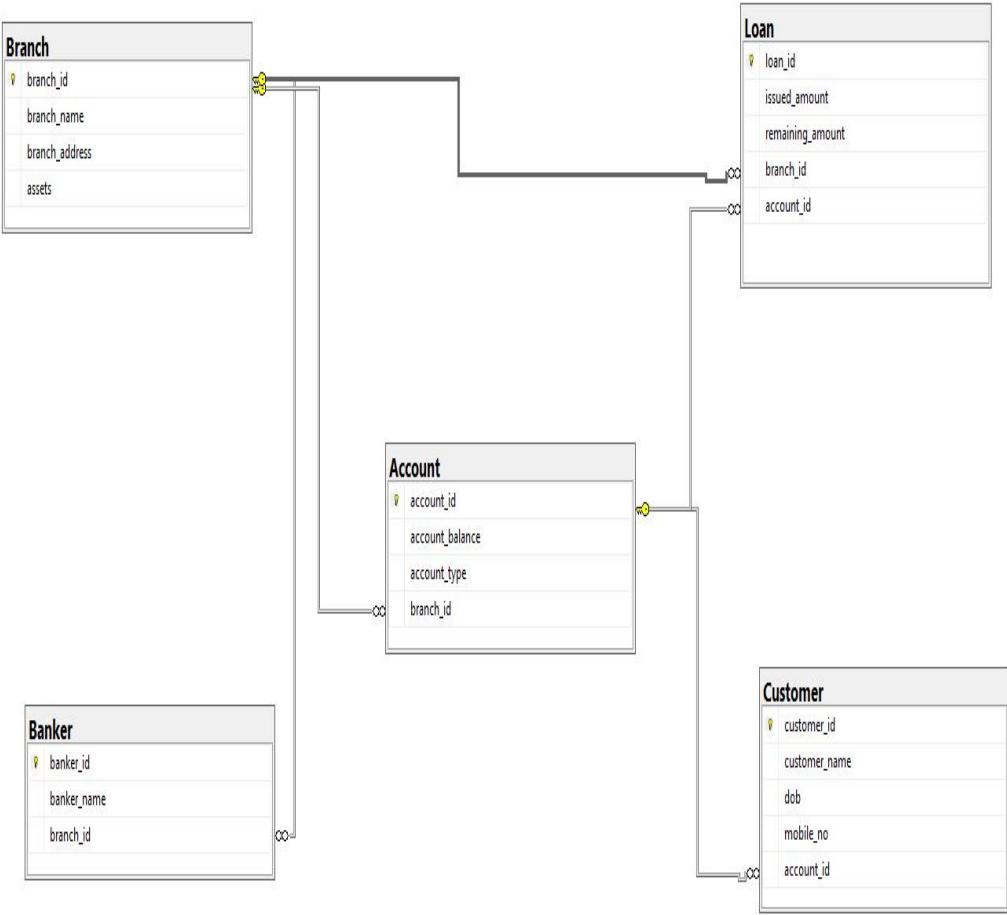
- ✓ **SAVEPOINT** allows partial rollbacks within a transaction. This is useful for complex operations.
- ✓ **Concurrency Control:** Applied FOR UPDATE locks and isolation levels (like SERIALIZABLE) to avoid race conditions.
- ✓ **Deadlock Handling:** Encouraged consistent access patterns and minimal locking duration.
- ✓ **Discretionary Access Control:** Granted/revoked permissions based on roles, ensuring principle of least privilege.

1. Design a relational database application based on your chosen project title.

## ❖ Relational Database Design for Bank Management System

### ➤ Key Entities:

- ✓ Branch(branch\_id,branch\_name,branch\_address,assets)
- ✓ Banker(banker\_id,banker\_name,branch\_id)
- ✓ Account(account\_id,account\_balance,account\_type,branch\_id)
- ✓ Customer(customer\_id,customer\_name,dob,mobile\_no,account\_id)
- ✓ Loan(loan\_id,issued\_amount,remaining\_amount,branch\_id,account\_id)



2. Implement transaction management for critical operations. For example, in a bank management system, this would include fund transfers.

- **BEGIN TRANSACTION OR START TRANSACTION:** is used to mark the beginning of a sequence of operations that should be executed as a single unit. If all operations succeed, the transaction is committed. If any operation fails, the transaction can be rolled back to maintain database integrity.

-- TASK 1: IMPLEMENT TRANSACTION MANAGEMENT FOR FUND TRANSFERS

```
--- TRANSFER FROM ACCOUNT_ID 1001 TO 1002
BEGIN TRANSACTION;
UPDATE Account SET account_balance=account_balance-5000 WHERE account_id=1001;
UPDATE Account SET account_balance=account_balance+5000 WHERE account_id=1002;
ROLLBACK;
COMMIT;

---update customer name and mobile number
BEGIN TRANSACTION;
UPDATE Customer SET customer_name='Rediet Elias' WHERE customer_id=201;
UPDATE Customer SET mobile_no='0945152686' WHERE customer_id=202;
ROLLBACK;
COMMIT;
```

### **3.Demonstrate the use of commit, rollback and savepoints**

#### **► COMMIT:**

Finalizes all changes made during the current transaction, making them permanent in the database.

#### **► ROLLBACK:**

Undoes all changes made during the current transaction, reverting the database to its previous state.

#### **► SAVEPOINT:**

Creates a temporary point within a transaction to which you can later roll back, without affecting the entire transaction.

```
--> TASK 2: DEMONSTRATE COMMIT, ROLLBACK AND SAVEPOINT
SELECT * FROM Account;

BEGIN TRANSACTION;
UPDATE Account SET account_balance=account_balance-1000 WHERE account_id=1003;
SAVE TRANSACTION Before_Credirt;
UPDATE Account SET account_balance=account_balance+1000 WHERE account_id=1004;
ROLLBACK TRANSACTION Before_Credirt;
COMMIT;

BEGIN TRANSACTION;
INSERT INTO Account VALUES(1006, 5000.00, 'Savings', 2);
SAVE TRANSACTION Irst1;
INSERT INTO Account VALUES(1007, 15000.00, 'Current', 3);
ROLLBACK TRANSACTION Irst1;
COMMIT;
```

### 3. Simulate concurrent transactions and analyze the impact of concurrency control techniques

- Locking concurrency control is a method used in databases to ensure that multiple transactions can safely execute at

the same time without interfering with each other. It works by placing locks on data items to prevent other transactions from accessing them in ways that could cause inconsistencies or errors.

- Types of Locks:
- **Shared Lock (S):** Allows multiple transactions to read a data item but not modify it.
- **Exclusive Lock (X):** Allows one transaction to read and modify a data item. No other transaction can access it until the lock is released.

```
-- TASK 3:SIMULATE CONCURRENCY TRANSACTION AND CONCURRENCY CONTROL BY USING LOCKING PROTOCOL
USE BANK_MANAGMENTSYSTEM
SELECT * FROM Account;
---TRANSACTION-1: HOLD LOACKS AND WAIT
BEGIN TRANSACTION
UPDATE Account SET account_balance=account_balance-5000 WHERE account_id=1003;
ROLLBACK;
COMMIT;

---TRANSACTION-2:THIS SESSION WILL WAIT UNTILL TRANSACTION-1 ROLLBACK/UNLOCK
SELECT * FROM Account;
UPDATE Account SET account_balance=account_balance-5000 WHERE account_id=1003;
```

**5.Implement discretionary access control by granting and revoking privileges to users., Demonstrate how to handle statistical database security.**

- **Discretionary Access Control (DAC)** allows users (especially data owners) to grant or revoke access to resources like tables or files. It's implemented using SQL commands like **GRANT** and **REVOKE** in databases.

---TASK-4; DISCRETIONARY ACCESS CONTROL AND STATISTICAL DATABASE SECURITY

```
-- TO CREATE SQL SERVER SECURITY  
CREATE LOGIN GROUP_6 WITH PASSWORD='050505';  
GO  
-- TO CREATE USER FOR LOGIN  
CREATE USER BANKER FOR LOGIN GROUP_6;  
GO  
--TO CREATING ROE  
CREATE ROLE G6 ;  
GO  
--TO ADDING USER TO ROLE  
SP_ADDROLEMEMBER 'G6', 'BANKER';  
GO
```

```
--GRANT SELECT AND INSERT ON CUSTOMER TO BANKER  
GRANT SELECT,INSERT ON Account TO BANKER WITH GRANT OPTION;  
GO  
--TO REVOKE GRANT OPTION FROM BANKER  
REVOKE GRANT OPTION FOR INSERT,SELECT ON Account FROM BANKER;  
GO
```