**MINI BANKING APPLICATIONS IN JAVA**

Project Description:

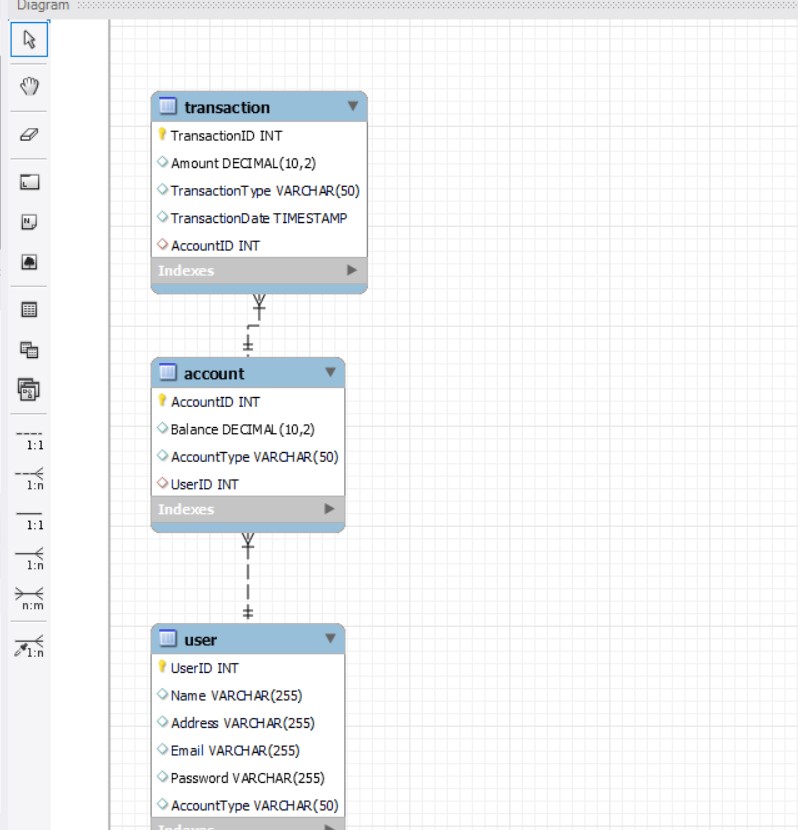
In any Bank Transaction, there are several parties involved to process transaction like a merchant, bank, receiver, etc. so there are several numbers reasons that transaction may get failed, declined, so to handle a transaction in Java, there is a JDBC (Java Database Connectivity) which provides us an API to connect, execute, fetch data from any databases. It provides the language Java database connectivity standards. It is used to write programs required to access databases.

Transactions in JDBC provide us a feature that considers a complete SQL statement as one unit, then executes once, and if any statement fails, the entire transaction fails. To use transaction, we have to set **set Auto Commit(false);** manually, and once all the statements are executed successfully, making changes in the database’ **commit()**method will be required.

In this Mini Banking Application, to handle a transaction, we are using JDBC Transaction to make transactions consistent.  This Application Provides Menu-Driven Console Interface to a User Using that User can perform functions like create Account, Login, View Balance And Transfer Money To The Other Customer

**ER DIAGRAM**

For a mini banking application in Java, we can extend the ER diagram to include a database design. Here's an example of how you can design the database schema based on the ER diagram provided earlier, assuming you're using a relational database like MySQL



**Software Prerequisite:**

* MySQL
* Eclipse

CREATE TABLE User (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255),

Address VARCHAR(255),

Email VARCHAR(255) UNIQUE,

Password VARCHAR(255),

AccountType VARCHAR(50)

);

CREATE TABLE Account (

AccountID INT PRIMARY KEY AUTO\_INCREMENT,

Balance DECIMAL(10, 2),

AccountType VARCHAR(50),

UserID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

CREATE TABLE Transaction (

TransactionID INT PRIMARY KEY AUTO\_INCREMENT,

Amount DECIMAL(10, 2),

TransactionType VARCHAR(50),

TransactionDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

AccountID INT,

FOREIGN KEY (AccountID) REFERENCES Account(AccountID)

);

**Relationships:**

* **User-Account (One-to-Many):** One user can have multiple accounts.
* **Account-Transaction (One-to-Many):** One account can have multiple transactions.

|  |
| --- |
| **USER** |
| User ID(PK)  Name(VARCHAR)  Address(VARCHAR)  Email(VARCHAR)  Password(VARCHAR)  Account Type(VARCHAR) |

|  |
| --- |
| **ACCOUNT** |
| Account ID(PK)  Balance(DECIMAL)  Account Type(VARCHAR)  User ID(FK) |

|  |
| --- |
| **TRANSACTION** |
| Transaction ID(PK)  Amount(DECIMAL)  Transaction Type(VARCHAR)  Transaction Date(DATE)  Account ID(FK) |

## 