BANKING SYSTEM DATABASE

Authored By:

Mason Somerville 000370945

Marlon Mavinier 000372201

Nico Hammer 000377237

Course: CPRO 1301A Database Design and SQL

Table of Contents

Introduction

Project overview

For our database design project, we have chosen to use the domain of a banking system.

Objectives

The main objectives of this project are to:

1. have multiple types of users (customer, teller, administrator) with different abilities in interacting with the database.
2. Provide a useable and functional mock banking system with multiple types of users and accounts as well as transaction records.

Tools Used

To create our database, we will be using the following tools:

- MySQL as the query language used to interact with the database
- MySQL Workbench and DBeaver as the development environments used to create the database
- Draw.io to create the E-R diagram of the database

Requirement Analysis

Business Requirements

Customers may create either one or both of the following: a chequing account, a savings account. Each customer is assigned a unique customer id and must provide their first and last name as well as one or both of their email or phone number upon account creation.

Each account must have a unique account id as well as the corresponding customer id, the account balance, and the account type. A record of every account's transactions must be recorded and contain the transaction amount, transaction date, transaction type, and the sender and receiver IDs; accounts may have more than one transaction but a transaction can only be associated with one account.

Tellers must be able to view and update account balances as well as account emails/phone numbers, view the account and transaction information and ID's, and create additional accounts for new or existing customers.

Administrators must be able to view and update all information available within the database.

Data Requirements

The database will have at least the following entities: customers, accounts, transactions. Customers will be a strong entity, and the accounts and transactions will be weak entities as they are reliant on the customer existing. The relationship between them will be such that: each customer may have one or two accounts, each account can only have one customer_id associated with it, each account may have 0 or more transactions, and each transaction may only have one account_id associated with it. Each entities attributes are listed below.

Customers: customer_id, first_name, last_name, email_address, phone_number

Accounts: account_id, customer_id, account_type, account_balance

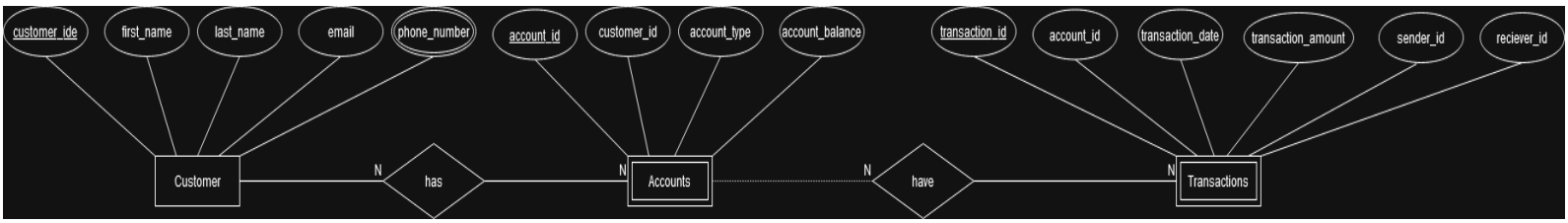
Transactions: transaction_id, account_id, transaction_date, transaction_amount, sender_id, receiver_id

User Roles

The database will have at least the following user roles:

- Customers who can create one of two account types, view their account balance and email/phone number, as well as being able to view their transactions.
- Tellers who can view and update account balances and id's, view the id's associated with both accounts and transactions, create additional accounts
- Administrators who will be able to view and modify all information in the database

Entity Relationship Diagram



Entities

The entities currently in the E-R diagram are the following: Customers, Accounts, Transactions.

- Customers are the strong entity and have the attributes: customer_id,first_name,last_name, email,phone_number
- Accounts is a weak entity and has the attributes: account_id,customer_id,account_type,account_balance
- Transactions is a weak entity and has the attributes: transaction_id,account_id,transaction_date, transaction_amount,sender_id,receiver_id

Relationships

The entities in the database will have at least the following relationships:

1. Customers will have a 1:1 total relationship with Accounts as a customer can have up to two account types but they must have at least one account, customer_id will be the primary key for the Customers table
2. Accounts will have a 1:1 total relationship with Customers as each account can only be associated with one customer using the customer_id as the foreign key and account_id as its primary key
3. Accounts will have a 0:M partial relationship with transactions as an account doesn't necessarily need to have any transactions
4. Transactions will have a M:1 total relationship with Accounts as each account can have multiple transactions using the account_id as the foreign key and transaction_id as its primary key