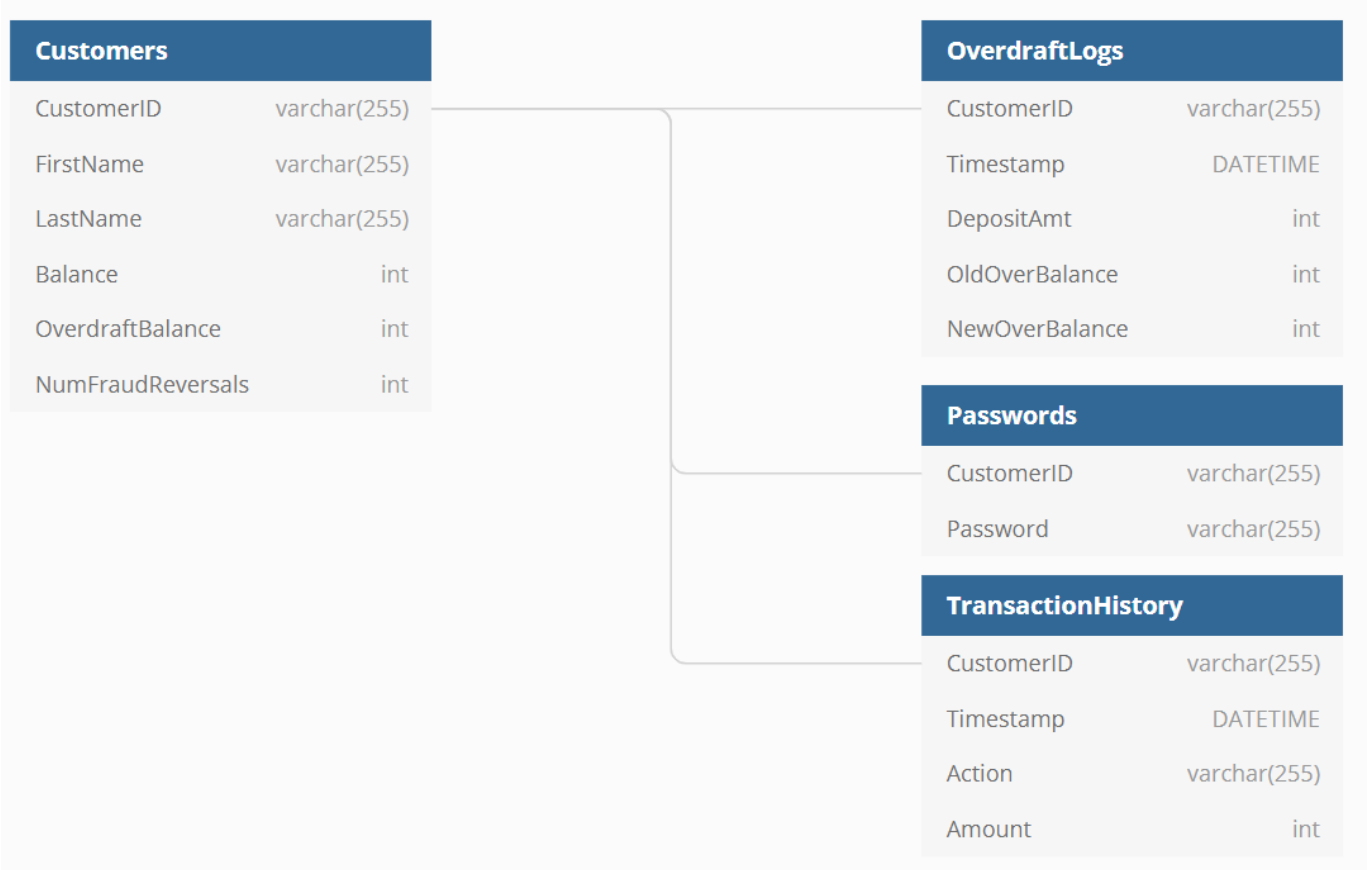
Testudo Bank Database Schema Documentation

Owner: Adithya Solai (adithyasolai7@gmail.com)

Schema Diagram

The Testudo Bank application uses a **MySQL** database with the following schema:



Schema Definition

Customers Table

- `CustomerID` is a primary key for the `Customers` table because it uniquely identifies each row in the table since each Customer is only given one row entry in this table.
- `CustomerID` can be used as a foreign key in the `OverdraftLogs`, `Passwords`, and `TransactionHistory` table.
- `OverdraftBalance` in the `Customers` table should only ever store values >= 0 and <= 1000. However, the customer actually owes `OverdraftBalance * 1.02` due to the **2% interest rate**.
    - Adding another column to this table to store the interest amount due is unnecessary because `OverdraftBalance` already exists. We can save on DB memory usage by keeping all interest rate logic in the Controller.
    - All user input will be in the normal $xx.xx format, but the `Balance` and `OverdraftBalance` are stored in pennies.

OverdraftLogs Table

- `CustomerID` is not a primary key in the `OverdraftLogs` table since each row in that table represents a single time a customer paid back some (or all) of their overdraft balance.
    - The same `CustomerID` can be in numerous rows in the `OverdraftLogs` table since customers can pay back their overdraft balance over multiple payments.
- The `DATETIME` data type used for `Timestamp` is in the form `YYYY-MM-DD hh:mm:ss`. Therefore, the `Timestamp` column can not capture instances where multiple repayments are made within the same second.
- Since `Timestamp` is granular only up to seconds and it is possible for a customer to make 2 repayments within the same second, the **candidate key** of this table is `CustomerID, Timestamp, NewOverBalance`.
    - A **candidate key** is the minimal group of columns needed to uniquely identify every row in a table.
    - The `DepositAmt` of two repayments occurring within the same second could be the same, so we don't gain anything from including this column in our candidate key.
    - The `NewOverBalance` must be different after each time a customer makes a re-payment, so this can be used to distinguish payments made within the same second.
    - **This forces us to enforce that any deposits made must be > 0 to avoid the same `NewOverBalance` in two re-payments made within the same second.**
    - There is an edge case where a customer can make a repayment, withdraw more on credit back to the previous `OverdraftBalance` amount, and then make another repayment for the same amount all within one second. We will assume this is not possible.

- `OverdraftBalance` starts at 0 for all customers, and is updated once a customer withdraws beyond their balance.
- All user input will be in the normal $xx.xx format, but the `OldOverBalance` and `NewOverBalance` are stored in pennies.

Passwords Table

- `CustomerID` is a primary key in the `Passwords` table since each customer can only have 1 password at a time.

TransactionHistory Table

- `CustomerID` is not a primary key in the `TransactionHistory` table since each row in that table represents a single time a customer made a transaction.
- - The same `CustomerID` can be in numerous rows in the `OverdraftLogs` table since customers can make multipe transactions.
- The `DATETIME` data type used for `Timestamp` is in the form `YYYY-MM-DD hh:mm:ss`. Therefore, the `Timestamp` column can not capture instances where multiple repayments are made within the same second.
- Since `Timestamp` is granular only up to seconds and it is possible for a customer to make 2 transactions within the same second, the **candidate key** of this table is `CustomerID, Timestamp, Action, Amount`.
  - A **candidate key** is the minimal group of columns needed to uniquely identify every row in a table.
  - We include all four columns because CustomerID and Timestamp are complusory for distinguishing each transaction.