Group Name: LinKDattaT

Kaydence Lin | lin.kay@northeastern.edu

Tanishi Datta | datta.ta@northeastern.edu

CS3200: Database Design

Project Final Report: ATM

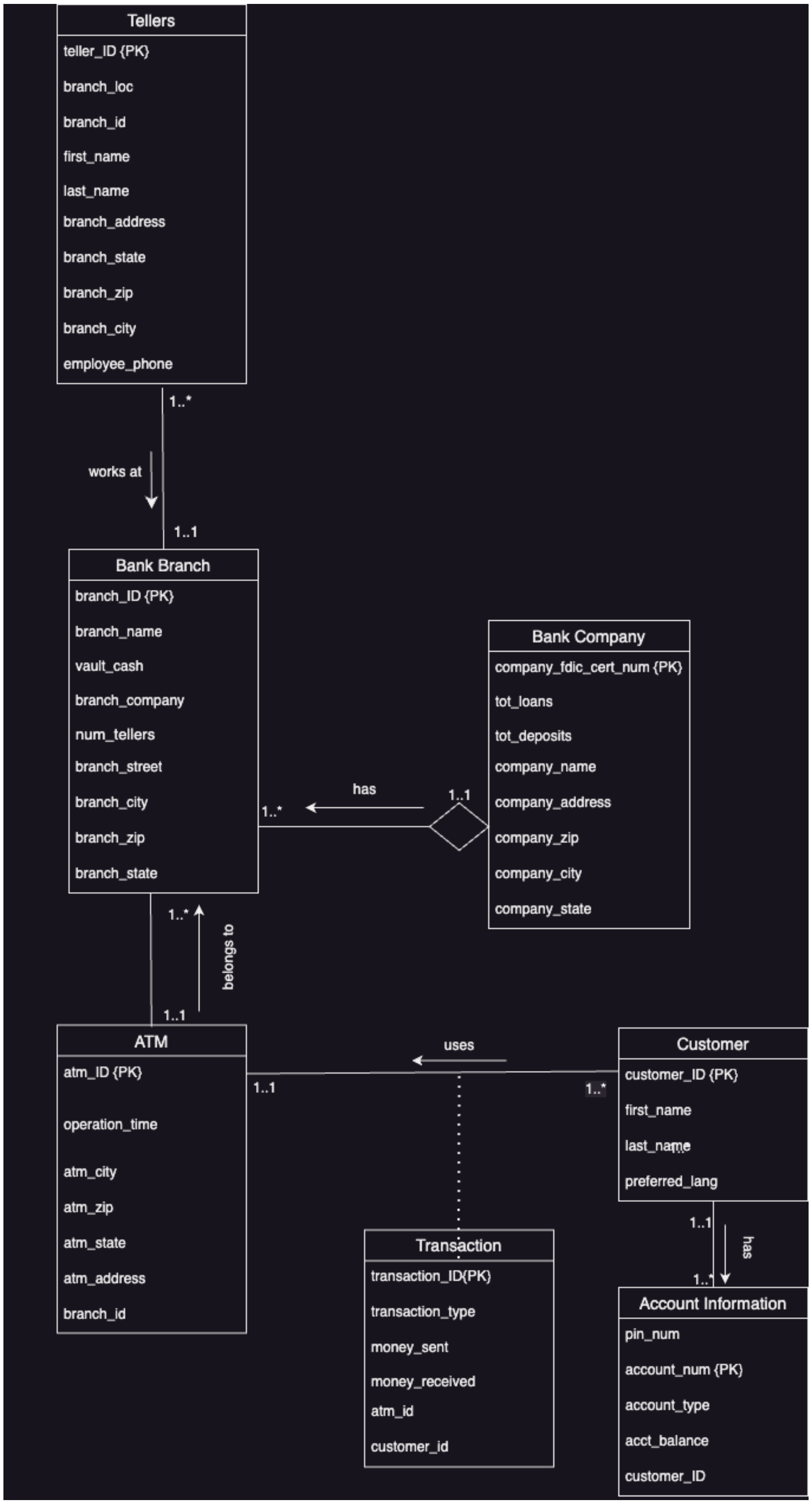
README

For the application, you should install software that uses SQL, such as MySQLWorkbench, and software that uses Python, such as PyCharm or Spyder in Anaconda. The Python libraries you need to install are random and pymysql. In the terminal of your computer, use the command ‘pip install random’ to install the random library, and ‘pip install pymysql’ to install the pymysql library.

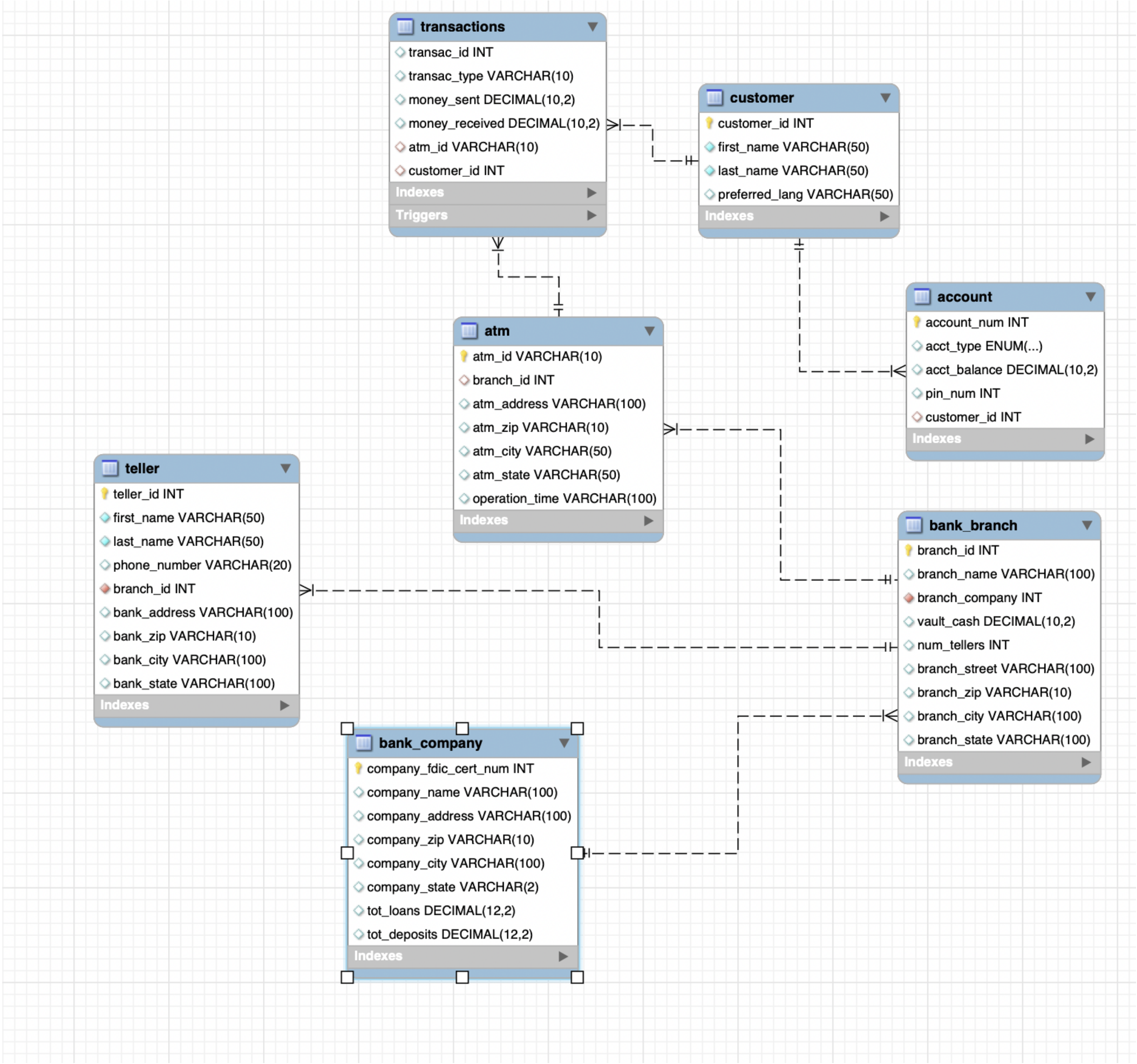
Technical Specifications

The application is written in Python and utilizes the command line interface (CLI). It connects to a SQL database called ‘banking_system’. When you run the Python code, it will ask you to input your SQL username and password. The username is normally ‘root’ and your password should be the same one you use to connect to SQL. If the username is not root, then it should be whatever username is linked to your SQL.

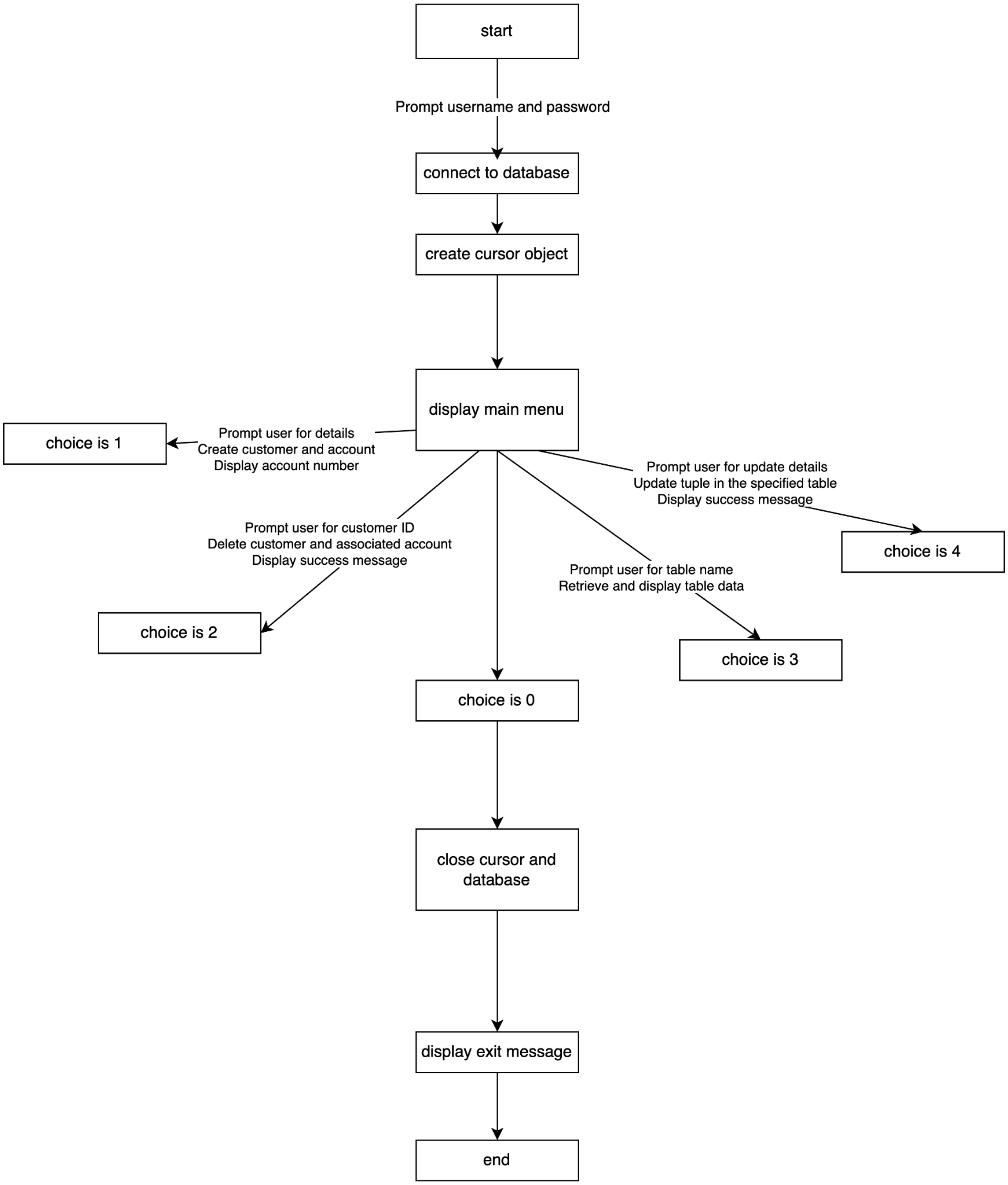
UML


Tellers
teller_ID {PK}
branch_loc
branch_id
first_name
last_name
branch_address
branch_state
branch_zip
branch_city
employee_phone
1..*
works at
1..1
Bank Branch
branch_ID {PK}
branch_name
vault_cash
branch_company
num_tellers
branch_street
branch_city
branch_zip
branch_state
has
1..*
1..1
Bank Company
company_fdic_cert_num {PK}
tot_loans
tot_deposits
company_name
company_address
company_zip
company_city
company_state
1..*
belongs to
1..1
ATM
atm_ID {PK}
operation_time
atm_city
atm_zip
atm_state
atm_address
branch_id
uses
1..1
1..*
Customer
customer_ID {PK}
first_name
last_name
preferred_lang
Transaction
transaction_ID{PK}
transaction_type
money_sent
money_received
atm_id
customer_id
1..1
has
1..*
Account Information
pin_num
account_num {PK}
account_type
acct_balance
customer_ID

Logical Design



Final User Flow of the System


start
Prompt username and password
connect to database
create cursor object
display main menu
Prompt user for details
Create customer and account
Display account number
choice is 1
Prompt user for update details
Update tuple in the specified table
Display success message
choice is 4
Prompt user for customer ID
Delete customer and associated account
Display success message
choice is 2
Prompt user for table name
Retrieve and display table data
choice is 3
choice is 0
close cursor and database
display exit message
end

Data Collection

Some of our data was collected using BankFind Suite on the FDIC website, the banks' website, and the data values that we made up. For the bank_co and bank_branch data values, most of them were gathered from the FDIC. We primarily used banks close to Northeastern's Boston campus. In bank_co, we made up the data values for tot_loans and tot_deposits. In bank_brach, we made up the data for vault_cash. For the tellers' values, we used employees that we found on the bank branch's website and we made up some if there were not any on the website. For the ATM values, we looked up the website of each bank branch to see if they had ATM machines and if they had more than one. We made up the values for atm_id so that it would be easier to work with for our project. For the initial customers, account_info, and transaction data values, we created randomly generated entries to fill the database.

Lessons Learned

Firstly, significant technical expertise was gained through the process. The creation of tables, the establishment of relationships using foreign keys, and the implementation of triggers and stored procedures provided hands-on experience in database design and management. This project helped in developing a deeper understanding of SQL syntax and the importance of proper database structuring.

Working on this project also provided valuable insights into the data domain. Understanding the relationships between entities, such as customers, accounts, and transactions, highlighted the intricacies of a banking system. It emphasized the need for accurate data representation and efficient querying to ensure reliable and meaningful results. For example, we kept getting errors while inserting our data and could not understand why. The error said unable to update to a child table, but after retracing our steps and looking back at every piece of code, we realized that the issue was rooted in our foreign key constraints. We created foreign keys to show relationships between entities, but while inserting the data we forgot to make sure the values were the same for both attributes involved in the constraint. Once we fixed the values for any errored foreign keys, our SQL was executing properly again.

We learned to back up all our files and code to a cloud drive whenever we are done editing and writing each time to ensure we do not lose any work. We initially met over Zoom and wrote the code together on Tanishi's computer. On June 18, Tanishi had to update her computer in order to successfully use a newly updated version of SQL, which resulted in her computer crashing. On June 19, Tanishi was able to get an appointment to get her laptop fixed, however, her hard drive had to be wiped in order for her computer to work again, losing all our code. This meant that we had to start our code, both our SQL database and application code in Python, completely from scratch again within a short period before the project was due.

The technical expertise that we gained was creating a more advanced application in Python that can be connected to a SQL database using pymysql. We learned how to call in procedures, functions, events, and triggers from SQL into Python. We were able to better familiarize ourselves and gain experience with creating and changing data in databases utilizing a front-end application.

Future Work

In the future, this project could be applied to a stock exchange transaction or online payment transactions. We could also include other transaction types, such as transfers to transfer money to another account, or payments to pay a credit card or bill. One limitation of our project is that our front-end application was focused on a customer perspective in mind. In the future, we could create other front-end applications with our database from a business perspective. From the business perspective, the bank company could manage the bank branches and ATMs that are opening or closing, tellers that are leaving or joining, and keep track of transactions from customers and their accounts. Another limitation is that customers can put in any amount of money without verifying that they have enough cash on hand or in other accounts since this is an online user CLI. In the future, it could be implemented into an actual ATM machine or we could create another table or function to verify that they have the specified amount of money they want to put in.

There are potential areas for added functionality that could enhance the database's capabilities. For instance, implementing additional security measures, such as encryption of sensitive data, would enhance data protection. Moreover, incorporating data analytics features could provide valuable insights into customer behavior, allowing banks to optimize their services and offerings. Finally, more planned uses of the database can be identified. One potential use is to support ATM operations, enabling customers to withdraw cash, check account balances, and perform other transactions seamlessly. The database can also facilitate customer management, allowing for the creation and maintenance of customer profiles, account information, and transaction histories.

Citations

BankFind Suite, banks.data.fdic.gov/bankfind-suite/bankfind?activeStatus=1. Accessed 21 June 2023.