Homework #8

Complete By: Part 1: Tuesday, Dec 3rd @ 5:00 pm
Part 2: Thursday, Dec 5th @ 5:00 pm

Submission: submitted via Blackboard

Part 1

Consider the scenario in which you are interviewing for a database administration position at a major banking company. They already have a system in place but ask you to design the system from scratch to assess your databases knowledge and familiarity with the banking domain. Alongside your design, you should state your assumptions and ask any questions you might have about how the domain must operate. In class we discussed a set of questions about the domain that your solution should be able to answer.
You are not required to use this exact set of entities, attributes, and relationships. This was a kicking-off point for the discussion, if you feel that there are changes that should be made, additions or removals, do so and state your reasoning for doing so. In particular, the design doesn't answer all the questions we asked, and there are many more questions that could be asked that we didn't get to in class.

Entities and attributes:

Customers
ID Addresses Name (First, Last, Middle) Phone number Email Date of Birth Credit Score

Accounts
Balance Account #s Interest rate

AccountType
ID Name

Deposits
DepositID Amount Timestamp Comment

Withdrawals
WithdrawalID Amount Timestamp Comment

Transfers
TransferID Amount Timestamp Comment

Relationships:
Customers have accounts
Accounts have types
Deposits are to Accounts
Withdrawals are from Accounts
Transfers are between Accounts (fromAccount + toAccount)

You must submit 4 files containing the following:

1. An image or pdf file with the name ER which contains an Entity Relationship diagram of the entities and relationships in your design. This diagram must include the specification of the cardinalities of each of the relationships.

2. A pdf of your relational schema with the name schema.pdf, generated from the ER diagram. Minimize the amount of data duplication using the rules for diagram decomposition from ER diagram to relational schema. Be sure to include the specification of both Primary and Foreign keys. Include in this file any comments you might have about changes you made to the Entities and Relationships listed above, the reasons why you think this design is better or enables you to do something you couldn't before.

3. A .sql file containing the commands to build the database from the relational schema with the name buildBank.sql. Take the tables you defined in the relational schema and actually write the DDL to construct those tables in MySQL. You can assume that this file will be run inside the database, so you don't need to delete and recreate the database. You should remove the tables if they already exist as part of this script. You must decide what are the most appropriate types for each of the fields. Tables should include primary and foreign keys as appropriate, and define cascading behavior (responses to violations of foreign key integrity constraints) where it makes sense. The columns should include any other integrity constraints such as unique and not null on attributes with those restrictions. You can use CHECK if you want to specify that an attribute has a more restrictive domain than the type indicates, even if the version of MySQL doesn't support the operation.

Part 2

Given the specific database schema discussed in class on 12/3/19, construct sql statements to build functions, procedures, triggers, and transactions as specified. Put the SQL query that generates the answer into #.sql where # is the number of the question.

1. Write a function ApplyInterest which takes an account number as a parameter and adds to the balance the current balance multiplied by the interest rate of that account.

2. Write a function PaySavingsInterest which takes no parameters and updates all the accounts with type Savings by paying interest to those accounts.

3. Write a procedure GetAccountInfo which takes a Customer ID as a parameter and during execution selects a table containing the account number, balance, and account type of each of the accounts owned by the customer.

4. Write one or more triggers to guarantee that the balance of the accounts table never goes below 0, throwing an exception if a change to the table would result in a balance containing a negative value.

5. Write a transaction which initiates a transfer of funds between the account with ID stored in variable @a and the account with the ID stored in variable @b, having @a pay @b the amount stored in the variable @amount.

6. Write a procedure or function of your choice to accomplish one of the tasks we discussed or an idea of your own that a bank would need to perform. Include a comment at the top containing a sentence or paragraph describing what the code is supposed to be doing.

7. Write a trigger or transaction of your choice to accomplish one of the tasks we discussed or an idea of your own that a bank would need to perform. Include a comment at the top containing a sentence or paragraph describing what the code is supposed to be doing.

For reference, here are some of the ideas we discussed. The rest can be found in the notes on Box under Lecture 23.

Deduct balance if account is idle
Make a deposit
Make a withdrawal
Produce a summary of account interactions for a particular account/user.
Update customer information (Address, phone number, email, name?)
What is the current amount of money physically in the bank?
Who are the top 10 people who have the most invested in the bank, and how much?
What is the number of transactions against an account in the past month?
Can you prevent a withdrawal if the withdrawal exceeds limit associated with account?