BIT2008A/ITEC2000A: Multimedia Data Management
Fall 2022

Final Project

Imagine that we are responsible for an e-banking application. We have decided to handle its database using PostgreSQL. This application logs user data, account information, and transactions. The explanation of the requirements for your implementation comes in the following paragraphs.

Functionalities

- **Banking**
  Users can control their accounts using our application i.e. their incoming and outgoing transactions, total balance, etc.
- **Clients**
  First, each client enters data fields such as first name, last name, user_id, at least one phone number, and at least one address. Also, the client chooses a password that we should store.
- **Accounts**
  Each account has an ID, total balance, account type (checking, saving, etc.) and the number of cosigners. The number of co-signers can be fewer than the number of account owners. For example, two people can have a joint account with one required signer. That means that each individual can make changes to the account on behalf of both of them. Each client can have access to different accounts under certain roles (sign, view, pay). Remember that each account can have multiple roles with respect to an account. Also, each client can have multiple accounts. To log the information about transactions, we need to store the amount, type (withdrawal or deposit), time, and a note.
- **Statements** Each statement has a unique ID, a note, initiator (who is a client), "From" account (source), total amount, and a couple of transactions. Each transaction has a "To" account and amount of money involved. To calculate the total amount for a statement, we have to add the amount of each transaction. Each client with sign permission on an account can sign the statements of that account as the source. When a statement reaches the minimum number of signatures, a user with "pay" role (who is not necessarily the initiator) will confirm the statement. Before then, the signers can decline their signatures or the initiator might edit the statement as well.
- ** Remember that it is up to you to insert a sufficient number of rows in each table to make the logging phase work properly.

Design

1) List all the entities in your application and determine if they are weak or strong entities. For weak entities, mention their corresponding strong entity.
2) For each entity, list all attributes, their type, and their options for the value, default value, or other explanations (if applicable). Also, the primary keys and foreign keys should be determined. For the foreign keys you should mention the reference, too.
3) Draw the final version of your ERD containing all the entities and the cardinalities of the relationships.
4) Explain how you maintained the normal form in your design and the trade off you might have made to enhance the performance.

Logging

You can create stored procedures or functions to parameterize these queries.

1) Show the name of clients who have a "sign" role on at least one account.
2) Show the list of accounts with fewer required signatures than the signers.

3) Show the list of every statement that is confirmed.
4) Show the list of all transactions to a certain account that is not paid.
5) Show the list of all declined signatures of a certain client.
6) Show the list of all of the accounts that two certain clients have in common.
7) Show the list of all statements that a certain client can sign.
8) Show the list of every transaction that is initiated by a certain user, but the person does not have the "sign" permission on it.
9) Show the list of every deposit in a certain account that is more than a certain amount.

Operation

1) Adding a user.
2) Adding an account.
3) Add, edit, or remove the access level of a certain user to a certain account.
4) Create a statement.
5) Edit or remove a statement.
6) Sign or unsign a statement.
7) Pay a statement.
8) Add incoming and outgoing transactions to an account.

Notes

- After each money transfer, there are two accounts to be updated, the sender and the receiver.
- Before executing an operation (Sign, unsign, pay, create, and edit) on an account, it should be evaluated to make sure that is valid. Statements with a sufficient number of signatures can be paid. Once a statement is paid, it cannot be unsigned. When a statement is signed by at least one person, it will not be edited but can be removed. Also, when the transactions inside a statement change, total amount should be updated. You can use *triggers* and *functions*.
- Store the history of all operations on an account including sign, unsign, pay and initiation.
- Store the log of all changes to the roles of each client.
- Store the last time the tables are edited.
- You should create backup tables if applicable to make accidental deletions reversible.
- You should avoid redundancy in your implementation unless you explain the necessity.
- You should automate the loggings using triggers.
- Keep in mind to use Views, Functions, and triggers when applicable since using the right tools and commands has a share in your grade.

Bonus

- If you manage to create a Django project to connect and work with this database as an interface, you can get up to 10% bonus on your final score for this course.

Description

1) The due date of this assignment is on December 9th, 11:55 PM. Late submission policy can be found on the course outline.
2) You are expected to submit a report that explains every little detail about your implementation. The report itself has 15 marks whereas the code has 10. As a result, you should not rely on the examiners to discover the features of your project unless you elaborate on them in your report.
3) Please try to sort your scripts into multiple files for creating tables, inserting instances, handling the functions and triggers, etc. and name the files accordingly.
4) Please put your code implementation in a folder, without the database, cached, or migration files. There should be (.py) and/or (.sql) files in your submission.
5) Please upload your submission as Lastname_Firstname_StudentID.zip on Brightspace.