COMP SCI 470 – Introduction to Database Management Systems

Group 8 – Drop Tables

Group Project

Nick King, John Goza, Max Schoettger, Sasha Niehorster-Cook

OVERVIEW

Our group was tasked with designing, optimizing, and implementing a relational database using the concepts and milestones provided by our class. We chose to create a database for a bank chain, with data involving bank patrons and their financials. Our database was designed to have tables for bank patrons and their respective accounts, budgets, and transactions. We implemented this database using PostgreSQL for the convenience with regards to local presentation. Our implementation placed foreign keys to tie account numbers to users, and to constrain transactions and budgets to specified accounts. The database was then normalized to ensure efficient queries. Our database was designed upon a two tier architecture, with PostgreSQL queries as our first layer, and a web frontend as our second layer.

IMPLIMENTATION

The PostgreSQL layer of our project was created by a static query which would first check for an existing infrastructure. If one was found, the query was written to drop the existing schema, cascading drops to all tables and data contained within it. The cascading drop was intended to ensure the database was initialized consistently on each contributor’s local machine, as local data would have to be re-inspected upon each schema change. Upon creation, each contributor was to run a script to import data held in a local csv file into the database. The query created the overall schema for the database, as well as the tables contained within. Our SQL queries were crafted with the intent to test the most relevant functions, such as common budget retrievals or summing transactions based upon dates or transaction types. Our group provided many common transactions, as well as the algebra for them. The transactions were refactored to reflect more efficient queries, but database design proved to be more important. The front end of the database architecture was designed to receive data from a given query, and display the data in a user friendly and inherently understandable fashion. The user interface was implemented with Bootstrap and Javascript, and reflected changes based upon what account was returned. The implementation of a third tier was desired for dynamic queries directly from the front end of the project, but due to time constraints was abandoned in favor of a more concrete implementation of the two existing layers.

IMPROVEMENTS / FUTURE SCOPE

Our group’s database was efficient, stable, and featured an inviting user interface. Our database could be improved in the future by the construction of a third, middle tier that would facilitate the transfer of data between the database and the user interface. Further improvements would include an automated script to recreate database schema and then import relevant data, as well as more queries for database administrators. Documentation of the design analysis of this database would further aid the possible efficiency impacts of future changes, and would help with optimization of aforementioned future queries. Overall, our database was designed for existing systems, and would profit from design for full implementation. In the future, this database could be implemented in a variety of banking applications, and would require relatively modest changes for each version. Database design would be heavily dependent upon client infrastructure/goals, and would inherently require be tailored to each client.

MEMBER CONTRIBUTIONS

Nick King served as the project’s main architect, and contributed the main database design, the common queries for the database, the ER diagram, and our secondary user interface implementation, which was important in determining the gap between the two levels of our implementation. Max Schoettger was the project’s user interface designer, and created the primary user interface which the database interacted with. John Goza was the project support developer, who provided the database schema and query algebra. Sasha Niehorster-Cook served as the project’s test developer, as well as the IO developer. Overall, contribution was well spread, but the project’s main architect was essential in achieving the team’s goals. All contributions were logged on Github, with the notable exception of Max Schoettger, who due to flawed repository settings, was not allowed write access and therefor contributed via other members of the group.

REFERENCE

https://github.com/spexican924/databasesProject