4/5/2023

Mark Deyarmond

DBAS4002: Final Project

PollSys United

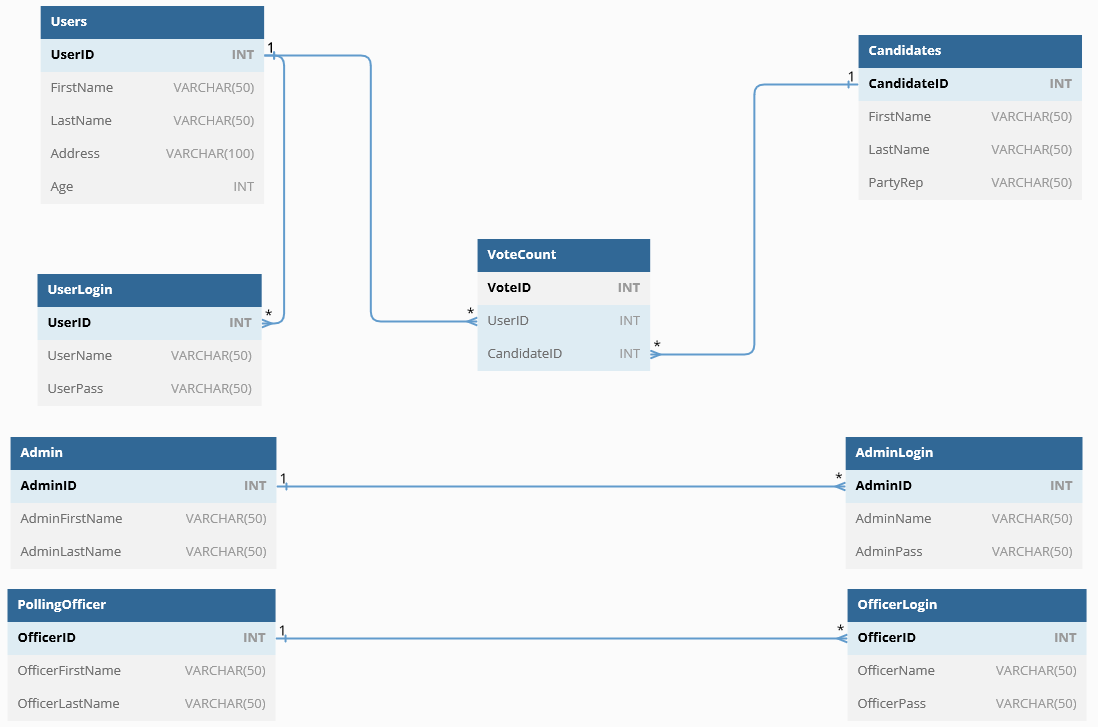
Database Design Document

### Introduction

For the final project in DBAS4002, I built a small database that was made up of voters, admins, polling officers and candidates. The purpose was to demonstrate the learned knowledge from term two DBAS course and apply that to a real-world situation. While the names used may have been derived from fiction, the methods and results are tied to the real world. **PollSys United** is the name I came up with to represent my work on tying together the projects for SAAD, PROG & DBAS by keeping familiar variable names and structures throughout. **NOTE:** All attached .sql files are numbered in accordance with the order that they should be executed.

### Summary

I kept my database relatively small in comparison to some of the databases we’ve worked with so far. It has a total of 8 tables, reasonable for the features and expectations of the software it ties in with.

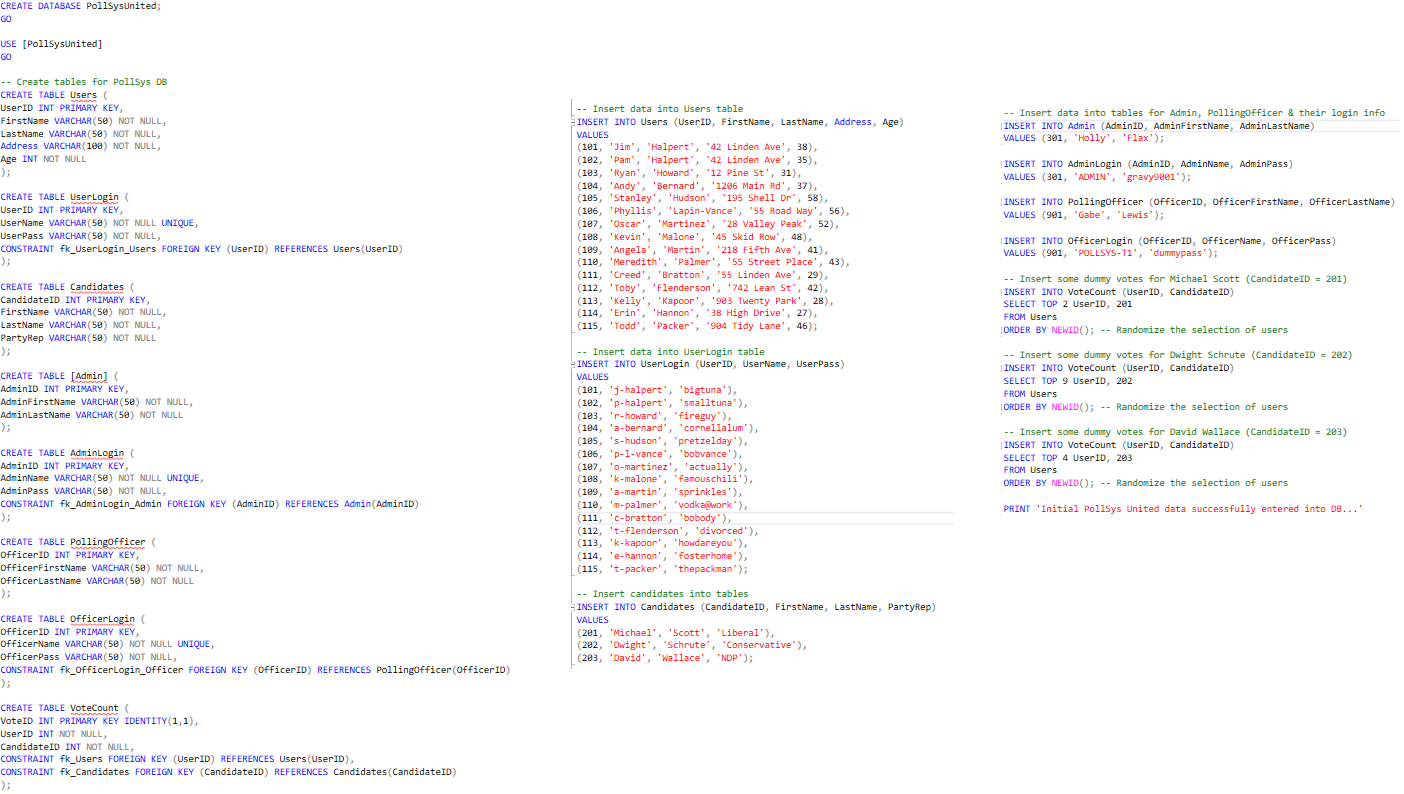


*Figure 1*

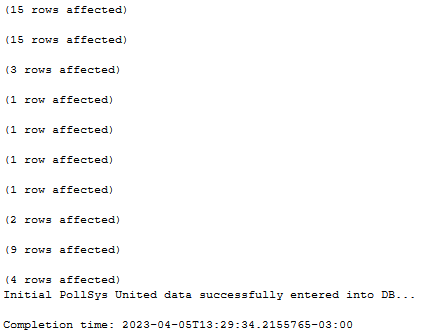
Seen above in *Figure 1* is the relationships between all the tables. For PROG final project, I kept most information within the own classes but I opted to separate it a little bit with this DB. Log-in information has been separated from each class so as to keep things a little bit tidier and less complicated. A table called *VoteCount* was created as well which helped to keep track of which UserID voted for which CandidateID. Users table was able to connect with UserLogin and VoteCount for tracking purposes. Candidates were not assigned log-ins as they represented an entity just to vote for, with no functions associated. Admin and PollingOfficer also linked with their respective log-in tables.

### Scripts

I made a total of 3 separate scripts for this database. The first one creates the database and then proceeds to insert some *dummy data*. The second script is just the insertion pieces of the first script, designed to be used and a quick method to insert the data back in after a truncation or if the DB get’s an error. Lastly, I had my DML scripts that ran the requested queries for the assignment. These scripts can all be found attached with this assignment’s submission files.



*Figure 2*



*Figure 3*

Above in *Figure 2* (if you zoom in) in a screenshot of all commands used to create database, tables, constraints and then insert that data into the tables. The aforementioned script to re-insert data is just the second half of the first one that creates the DB.



*Figure 4*

The other script included for this assignment was to satisfy the requirements to return reports about candidates and functions such as password resets. For the purposes of the assignment, I opted to hardcode some values into the commands such as delete user or update user information. *Figure 4* zoomed in shows all the commands with comments as documentation.

### Error checking & extra

For error checking, I was able to use @@ROWCOUNT to determine if for example, a row was successfully deleted on user deletion. If the @@ROWCOUNT was greater than 0 then I knew an error had happened. If an error was detected, the transaction would roll back. Likewise, if no errors were detected then transaction would commit. I also made sure to make use of print statements for user friendliness purposes. Any time a transaction was successful or errored out, a message prints for the user to see and get more information.