Final Project

DBAS 4002

Database Design Document

Ian Christian

W0480449

Due Date: April 6th, 2023

Contents

[1. Introduction 2](#_Toc131449587)

[2. Overview 2](#_Toc131449588)

[3. Conceptual Model 2](#_Toc131449589)

[4. Logical Model 4](#_Toc131449590)

[5. Submission files 5](#_Toc131449591)

[5.1. Create-Tables-&-Relationships.sql 5](#_Toc131449592)

[5.2. Register-New-Records-Processes.sql 5](#_Toc131449593)

[5.3. Update-Records-Processes.sql 5](#_Toc131449594)

[5.4. Delete-User.sql 6](#_Toc131449595)

[5.5. Clear-All-Tables.sql 6](#_Toc131449596)

[5.6. Record-Retrieval-Statements.sql 6](#_Toc131449597)

# Introduction

This document serves to provide explanations for the design decisions, implementation, and general methodology regarding Ian Christian’s SQL-script submissions for the Transactional Database course.

# Overview

The objective of the Election database (hereafter referred to as the “database”) is to provide a storage structure, and data definitions for the various record items deemed useful in a voter polling software scenario. The primary actors involved in this scenario are voters, candidates, and officers. It was interpreted that the term “user” may be used interchangeably with “officer.” This is the hypothetical person who would be responsible for inputting all data into the system for processing and storage.

# Conceptual Model

Diagram

Description automatically generated

In this conceptual model I have decided that each person-type can be provided a main table for their personal information (names, email addresses, etc.) and a child table where necessary. To prevent unauthorized access, officers need to login to the system to validate their identity. Their account information is stored in a separate table which is associated by the use of foreign keys.

The relationship between candidates and officers is that the officer must input the candidate’s information into the system. This means an officer ID number will be associated with the candidate they entered, to act as an author signature. Candidates also belong to the party they represent. The party table is where information about voting statistics will be tracked.

Voters will have the option to select a candidate they choose to vote for. As such, there will be a column for each voter record to note the candidate of their choice. The voter also requires a Boolean style data point to make note of whether they have voted yet. SQL Server does not provide a Boolean data type so a separate table exists simply to associate a key-value pair between an integer and its “true/false” state.

# Diagram Description automatically generatedLogical Model

In this logical model the precise data-types for each item have been established. The password information of officers will be stored as encrypted, reversible hash values. For simplicity this will be stored as a varchar and is assumed that encryption will occur outside of the database layer. This way, the actual password will never be stored in its true form in the database.

To add validation sophistication, the social insurance numbers of voters will be recorded, as well as each person’s unique voter keycode. This is done to emulate real-world security measures as data like a person’s SIN is often used to verify a voter’s identity.

# Submission files

To properly execute the following sql scripts a blank database called **Election** must first be created in SQL Server.

It is recommended that scripts be run in the following order. Some scripts contain procedures which overwrite, or some cases invalidate the procedures in others (for example, Clear-All-Tables.sql only serves a purpose if the database has records to begin with.)

## Create-Tables-&-Relationships.sql

Here is where all the tables are defined and their foreign-key relationships are established

## Register-New-Records-Processes.sql

To simplify the inserting of personal data, each person type (voter, candidate, and officer) has a customized stored procedure: insertNewVoter, insertNewCandidate, insertNewOfficer. Each procedure takes execution parameters for the specific person and stores them in the appropriate tables. As some data like foreign-keys belong in multiple tables, this significantly reduces the amount of repetitive INSERT statements. In a transactional statement, separated by save points, each procedure is demonstrated using hypothetical data. This hypothetical data will be important for demonstrating following procedures.

This script also contains a trigger called vote\_update\_tr. The purpose of this is so that whenever a voter’s ballot is cast (more to follow) the appropriate candidate’s party vote-count increases by 1. This means the Party table will automatically track the votes being cast. This further adds simplicity for officers.

## Update-Records-Processes.sql

Here is where procedures are defined and demonstrated for the purpose of editing pre-existing voter, candidate, and officer records. This sort of functionality allows for correcting possible human errors with data entry.

Similar to the procedures for registering new people, the update procedures accept parameters and store them where appropriate. The primary difference is that here the UPDATE keyword is used to replace data which has already been set.

There is also procedure included to allow updating a voter’s vote information. This accepts the voter ID number (used to identify their specific record and its associates), the voter’s unique keycode (used to further validate the record to be updated), and the voter’s choice of candidate. Executing this procedure will also fire the vote\_update\_tr trigger defined previously. In this way only one command must be entered in order for someone’s vote to be recorded, and for the change to be made in the overall vote statistics.

## Delete-User.sql

This contains a single procedure and its demonstration. For the purposes of this project, the terms users and officers were interpreted to be interchangeable. The procedure herein accepts an officer ID number (to identify their record) as well as their username and password. This is a security measure to safeguard against unauthorized deletion. If the provided credentials match the targeted record, it, and its foreign-key related records in other tables are deleted.

## Clear-All-Tables.sql

Here contains a procedure to remove all data from all tables simultaneously. To achieve this without also removing the table schema, first all the foreign-key relationships are dropped, then the TRUNCATE keyword was used for each table. The previously removed foreign-key relationships are then reinstated so as to leave the structure in its original state. This procedure is then demonstrated. If this script is run and all records are deleted, it is recommended to rerun the contents of Create-Tables-&-Relationships.sql, followed by Register-New-Records-Processes.sql. This will restore the contents of the database.

## Record-Retrieval-Statements.sql

The contents of this script serve to fulfill the requirements for fetching reports as outlined in the project instructions document. For each report a view was created then executed.