Sophie Dunfield

W0249605

Prog2700 Polling System

Database Design Document

Contents

[Use of this Document 3](#_Toc131060158)

[Purpose and Scope 4](#_Toc131060159)

[1.1 Document objective 4](#_Toc131060160)

[1.2 Points of Contact 4](#_Toc131060161)

[1.3 Glossary 4](#_Toc131060162)

[System Architecture 5](#_Toc131060163)

[2.1 System Design 5](#_Toc131060164)

[2.2 Database and Application Software 5](#_Toc131060165)

[2.3 Database management 5](#_Toc131060166)

[Database Specification 6](#_Toc131060167)

[3.1 Conceptual Model 6](#_Toc131060168)

[3.2 Logical Model 7](#_Toc131060169)

[Voters Table 7](#_Toc131060170)

[Addresses Table 7](#_Toc131060171)

[Contact Table 8](#_Toc131060172)

[Polling Officer Table 8](#_Toc131060173)

[Payment Table 9](#_Toc131060174)

[Bank Table 9](#_Toc131060175)

[Candidate Table 9](#_Toc131060176)

[District Table 10](#_Toc131060177)

[Voting Station Table 10](#_Toc131060178)

[Votes Table 10](#_Toc131060179)

[3.3 Physical Model 11](#_Toc131060180)

[Address Table 11](#_Toc131060181)

[Bank Table 11](#_Toc131060182)

[Candidate Table 11](#_Toc131060183)

[Contact Table 12](#_Toc131060184)

[District Table 12](#_Toc131060185)

[Payment Table 12](#_Toc131060186)

[Polling Officer Table 12](#_Toc131060187)

[Voter Table 12](#_Toc131060188)

[Votes Table 13](#_Toc131060189)

[Voting Station table 13](#_Toc131060190)

[Scripts 13](#_Toc131060191)

[4.1 TSQL Script for Database 13](#_Toc131060192)

[4.1.1 TSQL Script 13](#_Toc131060193)

[4.1.2 TSQL script use 16](#_Toc131060194)

[4.2 DDL Scripts for Database 16](#_Toc131060195)

[4.2.1 DDL Script 16](#_Toc131060196)

[4.2.2 DDL Script Use 19](#_Toc131060197)

[4.3 DML Scripts for Database 19](#_Toc131060198)

[4.3.1 DML Scripts 19](#_Toc131060199)

[4.3.2 DML Script Use 21](#_Toc131060200)

[Conclusion 21](#_Toc131060201)

[Bibliography 22](#_Toc131060202)

# Use of this Document

This document is meant as a guide in explaining various aspects of the ‘Prog2007 Polling machine’ system. Please refer to the quick reference table of contents to explore the document and figure there in by pressing the control key and select the section you wish to visit.

This project is being produced in concert with a System analyse project, and a programming in c language project for educational purposes and therefore is not meant to be used in anyway outside of the scope of the NSCC IT-Programming course. If you have received a copy of this program, or any involved documents, please be advised that the author of any of these items is not liable for any damages incurred from the use of these systems. And reproduction, or distribution of any of these documents requires express condition for the author and must be labelled properly as ‘for educational use only’.

This document will be referring several aspects not included in the document itself by citation to the proper document, or by images of documents of programs. Please see the bibliography at the end of the document for information on the referenced documents. In addition, the systems being used in this document, including that of the ‘IntelliJ CLion’ IDE (integrated development environment), the Microsoft MYSQL Server Management Studio 18, and Microsoft Visio will be added with their website will be linked in the bibliography.

The information in this document will serve to address a few specific areas of interest to the learning outcomes of the DBAS4002 course, including but not limited to database design, transaction control language, data manipulation language, data definition language, and data control language.

All the scripts used in this document were created using Microsoft SQL Server Management Studio 18. All information provided in these scripts pertaining to people, places and elections are a fabrication for test purposes and to display the use of the system. Any resemblance to events or people, fictional, living or dead is unintentional and purely coincidental.

# Purpose and Scope

## Document objective

In this document we will be discussing the models of this database, what it’s intended purposes are and how we go about using and exploring the schema of the information. This information will be filled with a test scenario for the purposes of exploring the educational purpose of database, but a TSQL script without added information can be provided upon request.

In addition to the documentation, you will find a glossary of terms at the end of the section explaining several terms and acronyms that are necessary to the understanding of this, and the other two documents being made for the purposes of this system. These glossaries are universal between documents to allow you to revisit and review words where needed.

## 1.2 Points of Contact

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Email | Telephone(fake number) |
| System Designer | Sophie Dunfield | W0246905@campus.nsccc.ca | (902)-867-5309 |
| System Analyst | Sophie Dunfield | W0246905@campus.nsccc.ca | (902)-867-5309 |
| Application Developer | Sophie Dunfield | W0246905@campus.nsccc.ca | (902)-867-5309 |
| Data Manager | Sophie Dunfield | W0246905@campus.nsccc.ca | (902)-867-5309 |

## 1.3 Glossary

|  |  |  |  |
| --- | --- | --- | --- |
| Abbreviation | Term | Abbreviation | Term |
| DOB | Date of birth | ID | Identification |
| Admin | Administrator | IT | information technology |
| RO | returning officer | SQL | Structured Query Language |
| DO | Deputy Officer | TSQL | TSQL – Transaction SQL (see SQL) |
| SO | supervising officer | DML | Data Manipulation language |
| UML | Unified Modeling Language | DCL | Data control language |
| SWOT | Strength, Weakness, Opportunity Threat | SSMS | SQL server management studio |
| MS | Microsoft |
| SDLC | Software Development life cycle | SDD | System Design documentation |
| DDD | Database Design Document |
| SAAD | Software Analysis and Design | HRM | Halifax regional Municipality |
|  |  |

## System Architecture

## 2.1 System Design

DBAS4002-Polling\_Station is a local database used for the purposes of storing information about simulated elections using the Prog2007 Polling Station software developed for the Prog2007 course at NSCC. It is currently a base setup, so no client side exists. For the purposes of this exercise, we are not connecting the database to the software, but for the sake of this document it can be assumed that they are to work in concert in later editions.

The database is written and stored in the MS MySQL server management studio with the intention of linked systems and eventual JavaScript client-side systems.

## 2.2 Database and Application Software

|  |  |  |  |
| --- | --- | --- | --- |
| Vendor | Software | Version | Comments |
| Microsoft | MySQL server management studio | V 18 | Database managements studio |
| Microsoft | Local system software | Windows 10 | General setup is local, no existing web services |
| Clion Ide | Prog2007 Polling Software | V1.0 | Attached System helps generate data |

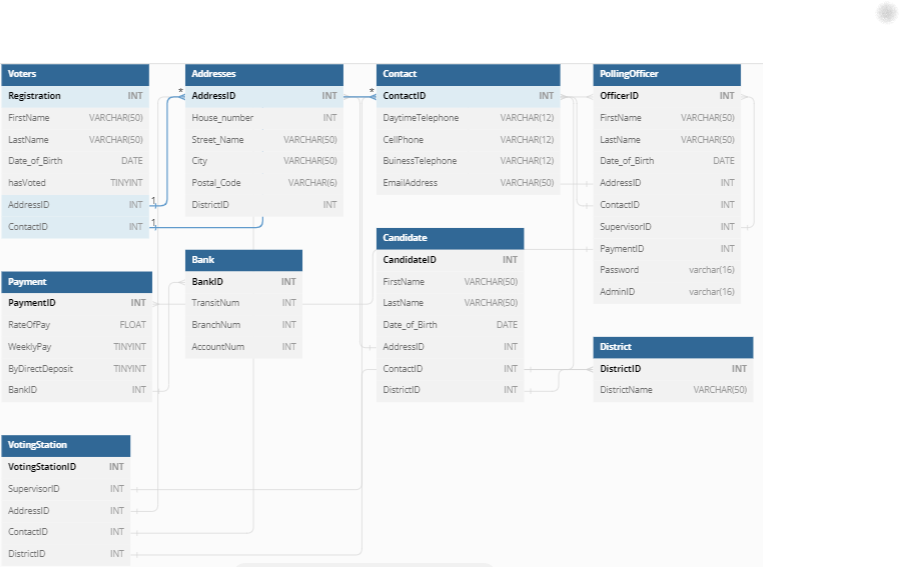
## 2.3 Database management

The purpose of this database is to store large amount of information about a specific election. As election information is a protected asset, and tampering, distributing or selling this information is illegal, there should be safe guards around who can access or view this information. It is suggested to have a database technician, or other IT expert to work on the database where needed, and it is necessary to enforce the use of an NDA as per protocol in all election work.

Certain information is intentionally omitted, such as a candidates home address or contact information, the votes of specific voters and other information considered protected. For more information, please see the link in the bibliography on election laws.

# Database Specification

## 3.1 Conceptual Model



The conceptual model above shows the relationship between tables, we will see more clearly what these relationships are in the physical models. You will find that this design revolves largely around the 3 types of people, all 3 and connected to the address and contact tables. These two tables are the center piece of the structure, connecting most of the tables.

In addition to this, you can see a few paths, such as the polling officer payments path that are designed for security and to limit access to certain pieces of information. Another aspect that is important to mention for security and legal reasons, is that the votes are in no way connected to the voters in any part of the system, allowing anonymity and security for the voters and realizing a more secure system by not releasing information that could be used for the purposes of bribes or illegal transfers.

## 3.2 Logical Model

### Voters Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| Registration# | Int | Increment | PK | Create table Voters(  Registration# INT INDENTITY (1,1) Primary KEY,  FirstName VARCHAR(50) NOT NULL,  LastName VARCHAR(50) NOT NULL,  Date\_of\_Birth DATE NOT NULL,  hasVoted TINYINT NOT NULL,  AddressID INT NOT NULL,  ContactID INT NOT NULL  ); |
| FirstName | Varchar(50) | Not null | / |
| LastName | Varchar(50) | Not null | / |
| Date-of-Birth | Date | Not null | / |
| hasVoted | Tinyint | Not null | FK |
| AddressID | Int | Not null | FK |
| ContactID | int | Not null | FK |

The Voter table is used to store information about the registered voters including their registration number, first name, last name, and date of birth. It also has a value to show if the voter has or has not voted in the election. In addition to this, the table references the Address and Contact tables to access the voter’s address and contact.

### Addresses Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| AddressID | Int | / | PK | Create table Addresses(  AddressID INT IDENTITY(1,1) PRIMARY KEY,  House\_number INT NOT NULL,  Street\_Name VARCHAR(50) NOT NULL,  City VARCHAR(50) NOT NULL,  Postal\_Code VARCHAR(6) NOT NULL,  DistrictID INT NOT NULL  ); |
| House\_number | Int | Not null | / |
| Street\_Name | Varchar(50) | Not null | / |
| City | Varchar(50) | Not null | / |
| Postal\_Code | Varchar(50) | Not null | / |
| DistrictID | int | Not null | FK |

The address table is a central table to the schema, it holds information about the voters, the polling officers, the candidates and the voting station. It contains data such as house number, street name, the city, and the postal code. It refences the district number, such as an address in Halifax would be part of the HRM.

### Contact Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| ContactID | Int | / | PK | CREATE TABLE Contact(  ContactID INT IDENTITY(1,1) PRIMARY KEY,  DaytimeTelephone VARCHAR(12) NOT NULL,  CellPhone VARCHAR(12),  BuinessTelephone VARCHAR(12),  EmailAddress VARCHAR(50)  ); |
| DaytimeTelephone | Varchar(12) | Not null | / |
| CellPhone | Varchar(12) | / | / |
| BuinessTelephone | Varchar(12) | / | / |
| EmailAddress | Varchar(50) | / | / |

Like The address table, the contact table is central to large amounts of other tables, including voters, candidates, polling officers, voting stations. It contains, daytime telephone numbers, cellphone, business telephone and email address. A lot of these columns are allowed to be null because most people use only a cellphone at this point, which would be entered into the daytime telephone column.

### Polling Officer Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| OfficerID | Int | / | PK | CREATE TABLE PollingOfficer(  OfficerID INT IDENTITY(1,1) PRIMARY KEY,  FirstName VARCHAR(50) NOT NULL,  LastName VARCHAR(50) NOT NULL,  Date\_of\_Birth DATE NOT NULL,  AddressID INT NOT NULL,  ContactID INT NOT NULL,  SupervisorID INT,  PaymentID INT NOT NULL,  Password varchar(16) UNIQUE NOT NULL,  AdminID varchar(16)  ); |
| FirstName | Varchar(50) | Not null | / |
| LastName | Varchar(50) | Not null | / |
| Date\_of\_Birth | Date | Not null | / |
| AddressID | Int | Not null | FK |
| ContactID | Int | Not null | FK |
| SupervisorID | Int | Not null | FK |
| PaymentID | Int | Not null | FK |
| Password | Varchar(16) | Not null, unique | / |
| AdminID | Varchar(16) | / | / |

One of the biggest tables in this system, the polling officer table deals with the employees who work on the election. This table includes, first name, last name, date of birth, as well as a password, and an optional adminID. The adminID is only used for the system admin assigned to operation and setting up the PROG2007 Polling station system. In addition to this, there are references to the address table, contact table, supervisor, which references the OfficerID in the officer table to denote the RO/SO, and a reference to the payment table.

### Payment Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| PaymentID | Int | / | PK | CREATE TABLE Payment(  PaymentID INT IDENTITY(1,1) PRIMARY KEY,  RateOfPay FLOAT NOT NULL,  WeeklyPay TINYINT NOT NULL,  ByDirectDeposit TINYINT NOT NULL,  BankID INT UNIQUE  ); |
| RateOfPay | FLOAT | Not null | / |
| WeeklyPay | TINYINT | Not null | / |
| ByDirectDeposit | TINYINT | Not null | / |
| BankID | INT | Unique | FK |

The payment table is only references by the polling officer table. It contains information about the employees pay such as the rate of pay, an integer denote whether they are paid weekly or by weekly, and whether they are paid by direct deposit. This them references the bank table for information on how to direct deposit checks. BankID is not null because if the employee doesn’t use direct deposit, it is not useful. It is unique, that is to protect from repeated entries causing a person to get paid more or less than allotted.

### Bank Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| BankID | Int | / | PK | CREATE TABLE Bank(  BankID INT IDENTITY(1,1) PRIMARY KEY,  TransitNum INT NOT NULL UNIQUE,  BranchNum INT NOT NULL,  AccountNum INT NOT NULL UNIQUE  ); |
| TransitNum | Int | Not null | / |
| BranchNum | Int | Not null | / |
| AccountNum | Int | Not null, Unique | / |

The Bank table is used to store the information necessary to direct deposit of polling officer checks. The three columns make up the transit number, branch number and account number as found on a check.

### Candidate Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| CandidateID | Int | / | PK | CREATE TABLE Candidate(  CandidateID INT IDENTITY(1,1) PRIMARY KEY,  FirstName VARCHAR(50) NOT NULL,  LastName VARCHAR(50) NOT NULL,  Date\_of\_Birth DATE NOT NULL,  AddressID INT NOT NULL,  ContactID INT NOT NULL,  DistrictID INT NOT NULL); |
| FirstName | Varchar(50) | Not null | / |
| LastName | Varchar(50) | Not null | / |
| Date\_of\_Birth | Date | Not null | / |
| AddressID | Int | Not null | FK |
| ContactID | Int | Not null | FK |
| DistrictID | Int | Not null | FK |

The candidate’s table is the information on the candidates being voted for. It stores their first name, last name, date of birth, and references the address, contact and district tables.

### District Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| DistrictID | Int | / | PK | CREATE TABLE District(  DistrictID INT IDENTITY(1,1) PRIMARY KEY,  DistrictName VARCHAR(50) not null  ); |
| DistrictName | Varchar(50) | Not null | / |

The district table simply stores the names of the district to be used a reference for where a person voted, where the candidate is running, and where the stations are held.

### Voting Station Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| VotingStationID | Int | / | PK | CREATE TABLE VotingStation(  VotingStationID INT IDENTITY(1,1) PRIMARY KEY,  SupervisorID INT NOT NULL,  AddressID INT NOT NULL,  ContactID INT NOT NULL,  DistrictID INT NOT NULL  ); |
| SupervisorID | Int | Not null | FK |
| AddressID | Int | Not null | FK |
| ContactID | Int | Not null | FK |
| DistrictID | Int | Not null | FK |

The voting station table is used to store a lot of references that make it easier to navigate and work with the system. It references information about the supervising officer to make it possible to figure out who is working that station, and the address, the contact, and the district of the station.

### Votes Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data Type | Constraints | Key | SQL |
| VoteID | Int | / | PK | CREATE TABLE Votes(  VoteID INT IDENTITY(1,1) PRIMARY KEY,  DateCast DATE NOT NULL,  AdvancedVoting TINYINT NOT NULL,  CandidateID INT NOT NULL,  VotingStationID INT  ); |
| DateCast | Date | Not null | / |
| AdvancedVoting | TinyInt | Not null | / |
| CandidateID | Int | Not null | FK |
| VotingStationID | Int | Not null | FK |

The votes table is used to store information about the votes. It stores the date case, whether or not the vote was done in advanced polling, the candidate that the vote was cast for and the station where the vote toke place. This also allows for a little bit of security to make sure that the vote is record in multiple systems and can be verified.

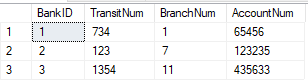
## 3.3 Physical Model

The data used in the physical model is invented for the purposes of this example and for the educational use of the system. And resemblance to anyone, living or dead is purely coincidental. Please be advised that this does not represent any real election, is not supposed to be used for the purpose of inventing election data and should be erased using the delete functions in the ‘Update’ script in the DML.

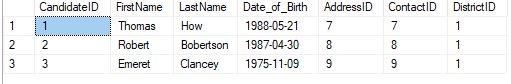
### Address Table



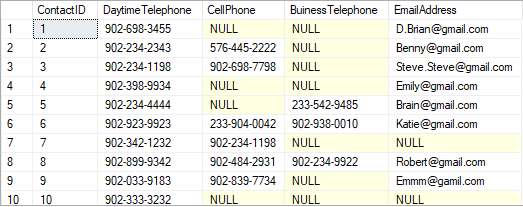
### Bank Table



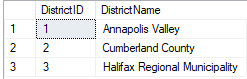
### Candidate Table



### Contact Table



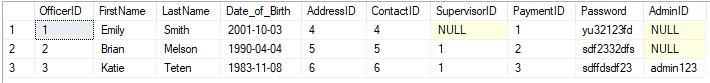
### District Table



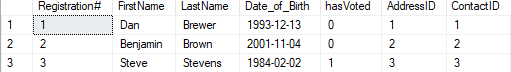
### Payment Table



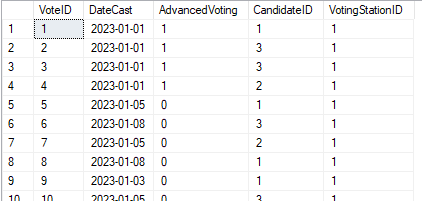
### Polling Officer Table



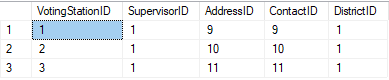
### Voter Table



### Votes Table



### Voting Station table



# Scripts

## 4.1 TSQL Script for Database

### 4.1.1 TSQL Script

DROP DATABASE if exists [DBAS4002-2023\_ELECTION]

GO

CREATE DATABASE [DBAS4002-2023\_ELECTION]

GO

USE [DBAS4002-2023\_ELECTION]

GO

BEGIN TRANSACTION

DROP TABLE if exists Voters;

DROP TABLE if exists Addresses;

DROP TABLE if exists Contact;

DROP TABLE if exists PollingOfficer;

DROP TABLE if exists Payment;

DROP TABLE if exists Bank;

DROP TABLE if exists Candidate;

DROP TABLE if exists Voting\_Station;

DROP TABLE if exists District;

DROP TABLE if exists Votes;

COMMIT

GO

THERE IS EXAMPLE DATA FOR EACH TABLE. IT'S USE IF ONLY FOR THE PURPOSE OF SHOWING THE SYSTEM. This information is purely invented for the purpose of this exercise and bears no resemblance to anyone, or any elections in the past. Any information corresponding any person living, fictional or dead is purely coincidental. If you wish to remove test data, simply add comment markers.

BEGIN TRANSACTION

CREATE TABLE Voters(

Registration# INT IDENTITY(1,1) PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Date\_of\_Birth DATE NOT NULL,

hasVoted TINYINT NOT NULL,

AddressID INT NOT NULL,

ContactID INT NOT NULL

);

CREATE TABLE Addresses(

AddressID INT IDENTITY(1,1) PRIMARY KEY,

House\_number INT NOT NULL,

Street\_Name VARCHAR(50) NOT NULL,

City VARCHAR(50) NOT NULL,

Postal\_Code VARCHAR(6) NOT NULL,

DistrictID INT NOT NULL

);

CREATE TABLE Contact(

ContactID INT IDENTITY(1,1) PRIMARY KEY,

DaytimeTelephone VARCHAR(12) NOT NULL,

CellPhone VARCHAR(12),

BuinessTelephone VARCHAR(12),

EmailAddress VARCHAR(50)

);

CREATE TABLE PollingOfficer(

OfficerID INT IDENTITY(1,1) PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Date\_of\_Birth DATE NOT NULL,

AddressID INT NOT NULL,

ContactID INT NOT NULL,

SupervisorID INT,

PaymentID INT NOT NULL,

Password varchar(16) UNIQUE NOT NULL,

--ADMIN ONLY, NULL OTHERWISE

AdminID varchar(16)

);

CREATE TABLE Payment(

PaymentID INT IDENTITY(1,1) PRIMARY KEY,

RateOfPay FLOAT NOT NULL,

WeeklyPay TINYINT NOT NULL,

ByDirectDeposit TINYINT NOT NULL,

BankID INT UNIQUE

);

CREATE TABLE Bank(

BankID INT IDENTITY(1,1) PRIMARY KEY,

TransitNum INT NOT NULL UNIQUE,

BranchNum INT NOT NULL,

AccountNum INT NOT NULL UNIQUE

);

CREATE TABLE Candidate(

CandidateID INT IDENTITY(1,1) PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Date\_of\_Birth DATE NOT NULL,

AddressID INT NOT NULL,

ContactID INT NOT NULL,

DistrictID INT NOT NULL

);

CREATE TABLE District(

DistrictID INT IDENTITY(1,1) PRIMARY KEY,

DistrictName VARCHAR(50)

);

CREATE TABLE VotingStation(

VotingStationID INT IDENTITY(1,1) PRIMARY KEY,

SupervisorID INT NOT NULL,

AddressID INT NOT NULL,

ContactID INT NOT NULL,

DistrictID INT NOT NULL

);

CREATE TABLE Votes(

VoteID INT IDENTITY(1,1) PRIMARY KEY,

DateCast DATE NOT NULL,

AdvancedVoting TINYINT NOT NULL,

CandidateID INT NOT NULL,

VotingStationID INT

);

BEGIN TRANSACTION

ALTER TABLE Voters

ADD FOREIGN KEY (AddressID) REFERENCES Addresses(AddressID);

ALTER TABLE Voters

ADD FOREIGN KEY (ContactID) REFERENCES Contact(ContactID);

ALTER TABLE Addresses

ADD FOREIGN KEY (DistrictID) REFERENCES District(DistrictID);

ALTER TABLE PollingOfficer

ADD FOREIGN KEY (AddressID) REFERENCES Addresses(AddressID);

ALTER TABLE PollingOfficer

ADD FOREIGN KEY (ContactID) REFERENCES Contact(ContactID);

ALTER TABLE PollingOfficer

ADD FOREIGN KEY (SupervisorID) REFERENCES PollingOfficer(OfficerID);

ALTER TABLE PollingOfficer

ADD FOREIGN KEY (PaymentID) REFERENCES Payment(PaymentID);

ALTER TABLE Payment

ADD FOREIGN KEY (BankID) REFERENCES Bank(BankID);

ALTER TABLE Candidate

ADD FOREIGN KEY (AddressID) REFERENCES Addresses(AddressID);

ALTER TABLE Candidate

ADD FOREIGN KEY (ContactID) REFERENCES Contact(ContactID);

ALTER TABLE Candidate

ADD FOREIGN KEY (DistrictID) REFERENCES District(DistrictID);

ALTER TABLE VotingStation

ADD FOREIGN KEY (SupervisorID) REFERENCES PollingOfficer(OfficerID);

ALTER TABLE VotingStation

ADD FOREIGN KEY (AddressID) REFERENCES Addresses(AddressID);

ALTER TABLE VotingStation

ADD FOREIGN KEY (ContactID) REFERENCES Contact(ContactID);

ALTER TABLE VotingStation

ADD FOREIGN KEY (DistrictID) REFERENCES District(DistrictID);

ALTER TABLE Votes

ADD FOREIGN KEY (CandidateID) REFERENCES Candidate(CandidateID);

ALTER TABLE Votes

ADD FOREIGN KEY (VotingStationID) REFERENCES VotingStation(VotingStationID);

COMMIT

### 4.1.2 TSQL script use

This is the script to create the database, it has a few specific sections to ensure the success of running. This script has removed the test data that you can find in the original script.

The first thing this script does is to ensure that the database does not exist in the system. If it does, it will delete the database and replace it. The second aspect ensures that the tables don’t exists. These do not have to be run the first time the database is added but can be a boon if you are trying to work with the system as an exercise.

The next section sets up all the table structures, and creates the database schema as represented in the conceptual model above. After this, all the keys are added to ensure that the relations between tables are created properly

## 4.2 DDL Scripts for Database

### 4.2.1 DDL Script

USE [DBAS4002-2023\_ELECTION]

GO

--Register Candidate

INSERT INTO Candidate

VALUES(

-- Candidate First Name --example 'Sophie'

''

,-- Candidate Last Name --example 'Dunfield'

''

,-- Date of Birth (yyyy-mm-dd) --example 1999-12-13

''

,--AddressID (See addresses to find proper office address \*THIS IS NOT A HOME ADDRESS\*) -- Number existing on Addresses table.

,-- ContactID (See contact table to find office contact information \* THIS IS NOT A HOME CONTACT\*) -- Number existing on Contact table

,--DistrictID (See candidate form and district table for district ID) -- Number existing on District table

);

--Register Voter

INSERT INTO Voter

VALUES(

-- Voter First Name --example 'Sophie'

''

,-- Voter Last Name --example 'Dunfield'

''

,-- Date of Birth (yyyy-mm-dd) --example 1999-12-13

''

,-- hasVoted (0 means no, 1 means yes) -- 0 or 1

,-- AddressID (See addresses to find proper home address) -- Number existing on Addresses table

,-- ContactID (See contact table to find home contact information) -- Number existing on Contact table

);

--Update Candidate Information

UPDATE Candidate

SET

-- Candidate First Name --example 'Doug'

FirstName=''

,-- Candidate Last Name --example 'Bennet'

LastName=''

,-- Date of Birth (yyyy-mm-dd) --example 1984-01-09

Date\_of\_Birth=''

,--AddressID (See addresses to find proper office address \*THIS IS NOT A HOME ADDRESS\*) -- Number existing on Addresses table.

AddressId=

,-- ContactID (See contact table to find office contact information \* THIS IS NOT A HOME CONTACT\*) -- Number existing on Contact table

ContactId=

,-- DistrictID (See candidate form and district table for district ID) -- Number existing on District table

DistrictId=

-- Select which candidate you are altering by selecting their candidateID as seen in the candidate table

WHERE CandidateId= ;

--Update Voter Infomation Information

UPDATE Voters

SET

-- Voter First Name --example 'Doug'

FirstName=''

,-- Voter Last Name --example 'Bennet'

LastName=''

,-- Date of Birth (yyyy-mm-dd) --example 1984-01-09

Date\_of\_Birth=''

,-- hasVoted (0 means no, 1 means yes) -- 0 or 1

hasVoted=

,--AddressID (See addresses to find proper office address \*THIS IS NOT A HOME ADDRESS\*) -- Number existing on Addresses table

AddressId=

,-- ContactID (See contact table to find home contact information) -- Number existing on Contact table

ContactId=

-- Select which Voter you are altering by selecting their candidateID as seen in the candidate table

WHERE Registration#= ;

--Update Login credetials for Admin users

UPDATE PollingOfficer

SET

-- First Name --example 'Doug'

FirstName=''

,-- Last Name --example 'Bennet'

LastName=''

,-- Password --example 'ert231$sd'

Password=''

,-- AdminID should only be entered if the user is the system admin, otherwise leave blank

AdminID=' '

-- Select with Officer's login you wish to update by selecting their ID as seen in PollingOfficer table

WHERE OfficerID= ;

-- Delete a specific User from database based on VoterID

-- as seen in the Voters Table

DELETE FROM Voters WHERE Registration#= ;

-- Delete a specific Polling Officer from database based on OfficerID

-- as seen in the PollingOfficer Table

DELETE FROM PollingOfficer WHERE OfficerID= ;

-- Delete a specific Candidate from database based on CandidateID

-- as seen in the Candidate Table

Delete From Votes WHERE CandidateID= ;

DELETE FROM Candidate WHERE CandidateID= ;

-- Delete all records from database

BEGIN TRANSACTION

Delete from Voters;

Delete from Votes;

Delete from VotingStation;

Delete from PollingOfficer;

Delete from Payment;

Delete from Bank;

Delete from Candidate;

Delete from Addresses;

Delete from District;

Delete from Contact;

COMMIT

### 4.2.2 DDL Script Use

There are 5 separate sections within these scripts. By selecting the section desired you can run just what you want to use. Below, you will find an explanation on what these scripts are used for.

1. Enter the information in each given section to add a candidate. The CandidateId is incremental so doesn’t need to be added.
2. Enter the information in each given section to add a voter. The VoterID is incremental so doesn’t need to be added.
3. This can used to update Candidate information; it selects a candidate using the CandidateID and allows you to add new information to overwrite the old.
4. This can used to update Voter information, it selects a candidate using the VoterID and allows you to add new information to overwrite the old.
5. This section is used to remove all information from all tables in the system. There is no was of recovering this information when deleted, please use caution.

## 4.3 DML Scripts for Database

### 4.3.1 DML Scripts

USE [DBAS4002-2023\_ELECTION]

GO

--Generate the 2 candidates that have the most number of votes--

-----------------------------------------------------------------

SELECT TOP 2 v.CandidateId, c.FirstName, c.LastName, COUNT(v.CandidateID) as NumberOfVotes

FROM Votes v

join (Select FirstName, LastName, CandidateID From Candidate) c

on c.CandidateID=v.CandidateID

GROUP BY c.FirstName,c.LastName,v.CandidateID

ORDER BY NumberOfVotes desc;

--Generate the Candidate with the least votes

-----------------------------------------------------------------

SELECT TOP 1 v.CandidateId, c.FirstName, c.LastName, COUNT(v.CandidateID) as NumberOfVotes

FROM Votes v

join (Select FirstName, LastName, CandidateID From Candidate) c

on c.CandidateID=v.CandidateID

GROUP BY c.FirstName,c.LastName,v.CandidateID

ORDER BY NumberOfVotes asc;

--Generate Candidates who got between 5 to 15 votes

-----------------------------------------------------------------

SELECT v.CandidateId, c.FirstName, c.LastName, COUNT(v.CandidateID) as NumberOfVotes

FROM Votes v

join (Select FirstName, LastName, CandidateID From Candidate) c

on c.CandidateID=v.CandidateID

GROUP BY c.FirstName,c.LastName,v.CandidateID

having COUNT(v.CandidateID) BETWEEN 5 AND 15

ORDER BY NumberOfVotes desc

--Generate all Candidates

-----------------------------------------------------------------

SELECT v.CandidateId, c.FirstName, c.LastName, COUNT(v.CandidateID) as NumberOfVotes

FROM Votes v

join (Select FirstName, LastName, CandidateID From Candidate) c

on c.CandidateID=v.CandidateID

GROUP BY c.FirstName,c.LastName,v.CandidateID

ORDER BY CandidateID asc;

--Generate all Candidates

-----------------------------------------------------------------

SELECT TOP 1 v.CandidateId, c.FirstName, c.LastName, COUNT(v.CandidateID) as NumberOfVotes

FROM Votes v

join (Select FirstName, LastName, CandidateID From Candidate) c

on c.CandidateID=v.CandidateID

GROUP BY c.FirstName,c.LastName,v.CandidateID

ORDER BY NumberOfVotes desc;

### 4.3.2 DML Script Use

Once again, this script is set up in 5 distinct sections, each one with a specific purpose. . By selecting the section desired you can run just what you want to use. Below, you will find an explanation on what these scripts are used for.

1. The first script will select the 2 candidates that have the most votes.
2. The second script will generate the candidate with the least votes.
3. The third script will generate all candidates who have votes between 5 and 15.
4. The fourth script will generate all the available candidates.
5. The last script will show only the candidate with the most votes.

# Conclusion

I hope that this document has helped you understand and use the PROG2007 Polling System database. For further information on the database please contact the Database designer as shown in the points of contact section or view the video explaining the use of the database attached to these files.

For further reading on the systems and the design of the software and data for the PROG2007 Polling station please refer to the User Guide and the System Design Document available on request.

# Bibliography

Microsoft (n.d.). *Microsoft Visio flowchart Maker*. Retrieved November 4, 2022, from <https://www.microsoft.com/en-ca/microsoft-365/visio/flowchart-software>

Microsoft (n.d.). *Download SQL server management studio*. Retrieved February 11, 2023, from <https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>

JetBrains (n.d.). *Clion: A Cross platform IDE for C and C++*. Retrieved January 11, 2023, from <https://www.jetbrains.com/clion/promo/?source=google&medium=cpc&campaign=11960744855&term=clion&content=489240779231&gclid=Cj0KCQjw8e-gBhD0ARIsAJiDsaUYQSzG-T7VDEiNuA-GPM0xvtkfSiO3_M9-CLECwLkwXXkJ9xmuviIaAmL2EALw_wcB>

(n.d.). *DB diagram. Io*. DB Diagram.io. Retrieved February 20, 2023, from <https://dbdiagram.io/home>

Grammarly (n.d.). *Free citation generator*. Retrieved January 15, 2023, from <https://www.grammarly.com/citations>

(n.d.). *GitHub*. GitHub. Retrieved March 20, 2023, from <https://github.com/>

(n.d.). *UML diagram type*. Creately. Retrieved February 21, 2023, from <https://creately.com/blog/diagrams/uml-diagram-types-examples/#ClassDiagram>

Government of Canada (n.d.). Canadian Elections Act. Justice Laws Website. Retrieved January 30, 2023, from <https://laws-lois.justice.gc.ca/eng/acts/e-2.01/>