Information Management Project

Cian Walsh - 17322794

11 December 2020

1 Application Description

The application implemented in this project is a database designed for election management in a simplified voting environment. The election system has a number of different elections for different levels - Town Council, Senate and Parliament elections. These elections all use the first past the post system, where the candidate who obtains the most votes in an election is allocated a seat. Candidates can stand in one election. Elections are participated in by constituencies of voters. Constituencies then have polling stations, usually in public buildings like community centres and schools.
The concept consists of six entities: *Election*, *Constituency*, *Polling Station*, *Voter*, *Institution* and *Candidate*. Further information about these entities can be found in the Entity Relationship Diagram below (Figure 1).

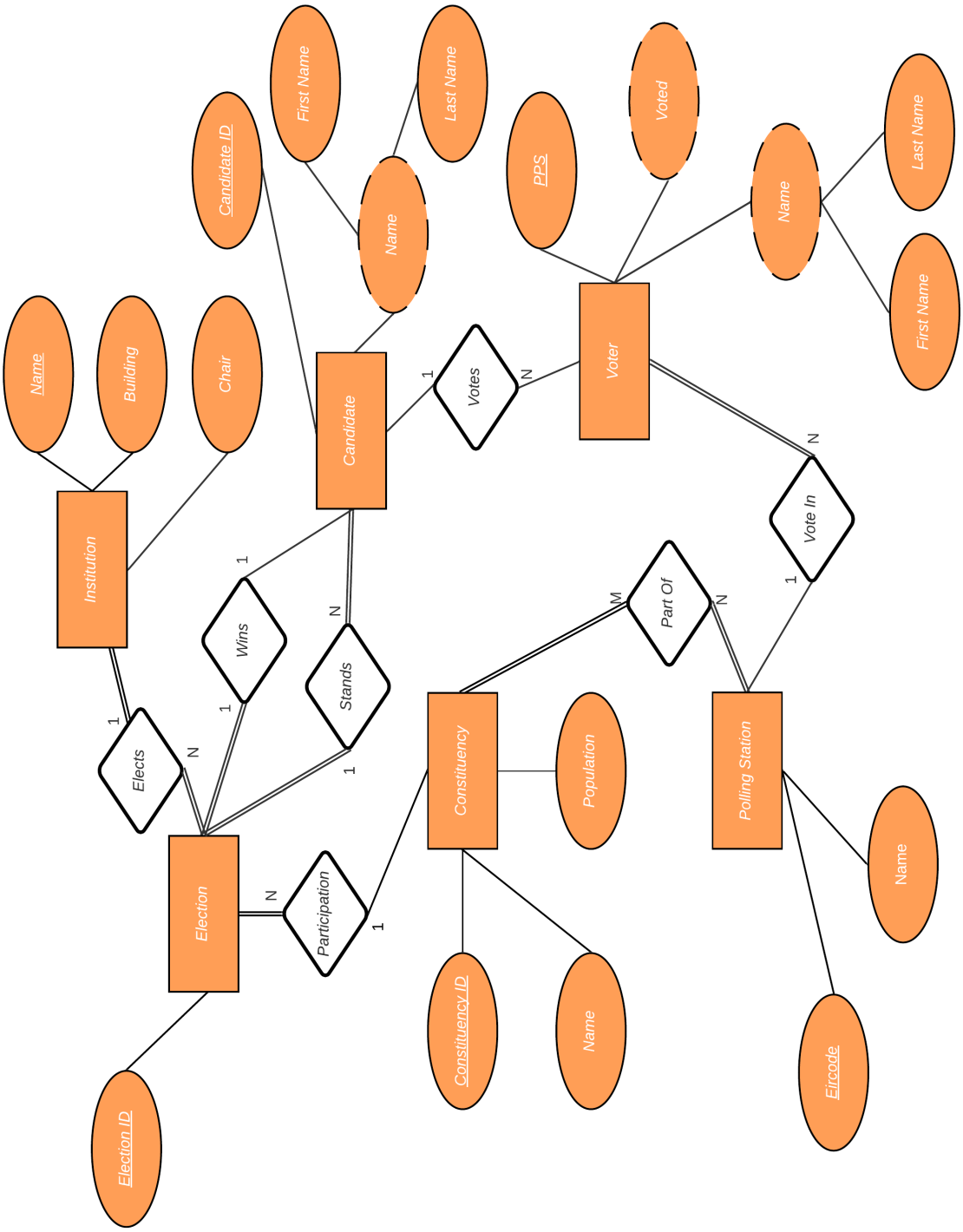

Figure 1: Entity Relationship Diagram

Next, this ERD is converted to a Relationship Schema, as seen below:

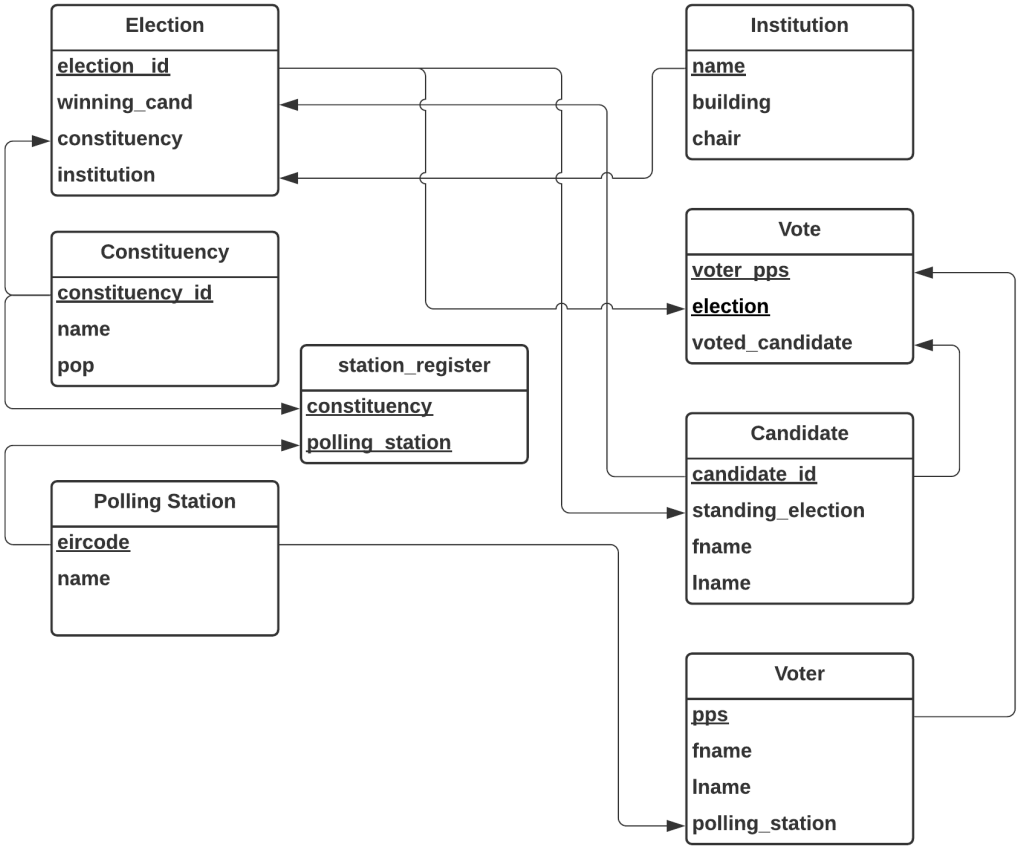

Figure 2: Relationship Schema

The relations generated from the entities themselves are displayed in the Relationship Schema (Figure 2), as well as an additional "Station Register" relation to accommodate the M:N relationship between constituency and polling station, and a "Vote" relation to tidy up the functional dependencies in the "Vote" entity. The functional dependencies for the whole database are in the diagram below (Figure 3):

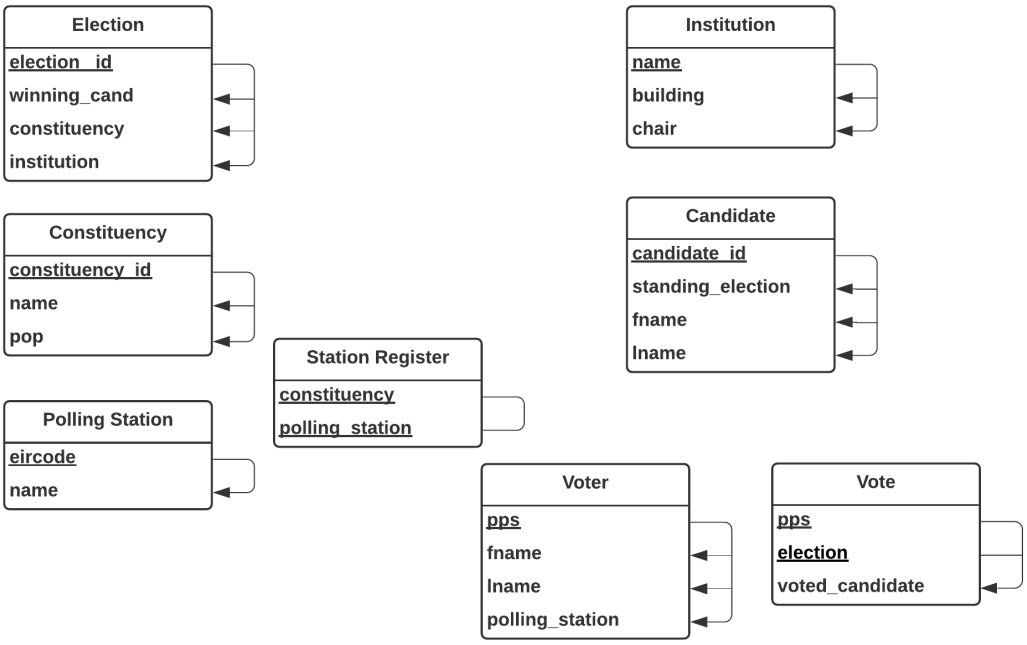

Figure 3: Functional Dependency Diagram

Through the addition of these relations, the database design is Boyce-Codd normal form compliant - all the values are atomic with no repeating groups, all non-key columns are dependent on the primary key in each relation and there are no transitive dependencies in the relations, and all of the functional dependencies have superkeys on the left-hand side. This normalisation form ensures that data duplication, and insertion, update and deletion anomalies. Queries should also be greatly simplified.

To summarise the relations and attributes in the database: *Election*
`election_id`: Unique ID for each election. Primary key.
`winning_cand`:Reference to candidate with the current highest total votes for the election. Foreign key.
`constituency`: The constituency eligible to vote in the election. Foreign key.
`institution`: The institution the seat won in the election is eligible for. Foreign key.
Institution
`name`: The name of the institution. Primary key.
`building`: The name of the building the institution is located in.
`chair`:The name of the chair of the institution, that all winning candidates would report to.
Candidate

`candidate_id`: Unique ID for each candidate. Primary key.
`standing_election`: The election the candidate is standing in. Foreign key.
`fname`: First name of the candidate.
`lname`: Last name of the candidate.
Constituency
`constituency_id`: Unique ID for the constituency. Primary key.
`name`: The name of the constituency.
`pop`: The census population of the constituency.
Polling Station
`eircode`: The eircode address of the polling station. Primary key.
`name`: The local name of the polling station.
Station Register
`constituency`: The constituency referred to by the entry. Joint primary key. Foreign Key.
`polling_station`: The polling station referred to by the entry. Joint primary key. Foreign Key.
Voter
`pps`: The PPS of the voter. Primary key.
`fname`: First name of the voter.
`lname`: Last name of the voter.
`polling_station`: Reference to the polling station the voter is registered for. Foreign key.
Vote
`pps`: Reference to the PPS of the voter. Joint primary key. Foreign Key.
`election`: Reference to the election the vote was cast in. Joint primary key. Foreign key.
`voted_candidate`: Reference to the candidate voted for. Foreign key.

2 Integrity Constraints

Constraints are used to ensure data accuracy and integrity in the database. All primary keys have been set to be unique and non-null, enforcing both key constraints on primary keys and entity constraints on each relation. Foreign keys that create dependencies between relations create referential constraints which can be violated by insertions, updates and deletions. Therefore, constraints have been set that should cascade the deletions through all dependant relations, and on updates the updates should be reflected. There was no point in setting any defaults in the event of a foreign key reference being deleted or updated, as logic dictates that none of the dependent entities in this database design could exist without the foreign entity they are referring to.

A number of extra constraints were also implemented.

Check constraints were implemented on specific attributes as follows:
`polling_station.eircode` must be a valid eircode (1 letter followed by 2 numbers followed by 2 letters followed by 2 numbers.)

`voter.pps` must be a valid PPS (7 numbers followed by a letter.)

Triggers were also set to ensure that votes being cast (inserted or updated) referenced a candidate actually in the election, and that the voter is registered in a relevant polling station. This was carried out using custom defined functions to reduce duplicate code.

Lastly, a trigger was set to update the winning candidate for an election after the votes table is changed. A procedure was used to simplify this.

The settings for these constraints are in the DDL section (Appendix A).

3 Security Roles & Views

While the integrity constraints protect against accidental inaccuracies in the data itself, further measures are required to prevent deliberate tampering with the database itself.

A number of different user roles are required for the database:
EC: The DBA for the database is employed by the Electoral Commission. They have superuser access to the database, and grant roles and privileges to other users as required. They also set up the elections, institutions, constituencies and polling stations as required. Read, modification and referential permissions are granted to the DBA for all tables. This user was also granted super user permissions so they could define the custom functions. They will be the only account granted account-level privileges, all others do not need this and will be given relation-level privileges only.
Registrar: The officials who register users in the database are the registrars. They need to be able to update the polling Voters table to register new voters for polling stations. The Registrars have read and referential privileges on polling stations, and read and modification privileges on voters.
Polling Official: The polling official needs to see the voters associated with a particular election in a particular polling station and whether they have voted or not. They have read privileges on voter, polling station and elections. Their view also contains a derived column informing them as to whether the voter has already voted or not.
Returner: The returner needs to see who has won each election and where to report the results to. Therefore, they have read privileges on elections and institution.
Voter: The voter has to vote in an election. While the majority of the security around this would be performed by the application accessing the database, the voter application role would need read access on a view that enables them to see what candidates are running in what election and where each voter should vote.

User accounts, in practice, would then be created using these roles as necessary.

A number of views were also defined displaying and joining fields relevant to each role, one for the registrar, polling official, returner and voter. These views are used rather than granting access directly to the tables so that the

users concerned can only view or change data that they absolutely need to.

In practice, of course, significantly more care and thought would go into the security for an elections database used by a real state. Notably, the clear link between voter and vote would not be drawn. This is a very simplified version of a database to address this application, but demonstrates the topics necessary in the context of the assignment.

The SQL implementations of all these views and roles can be found in the DCL section (Appendix B). The "ec" DBA role is actually defined at the start of the DDL section as their superuser privileges are required to define the functions.

Appendices

A DDL

```
CREATE DATABASE voting_system;
CREATE USER 'ec'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON voting_system TO 'ec'@'localhost' WITH GRANT OPTION;
GRANT CREATE USER, CREATE ROLE TO 'ec'@'localhost';
GRANT SUPER ON *.* TO 'ec'@'localhost';

CREATE TABLE institution (
                                name VARCHAR(24) NOT NULL,
                                building VARCHAR(24) NOT NULL,
                                chair VARCHAR(24) NOT NULL,
                               PRIMARY KEY (name)
);

CREATE TABLE constituency (
                                 constituency_id INT NOT NULL AUTO_INCREMENT,
                                 name VARCHAR(24) NOT NULL,
                                 pop INT NOT NULL,
                                PRIMARY KEY (constituency_id)
);

CREATE TABLE election (
                              election_id INT NOT NULL AUTO_INCREMENT,
                              winning_cand INT,
                              constituency INT NOT NULL,
                              institution VARCHAR(24) NOT NULL,
                             PRIMARY KEY (election_id),
                             FOREIGN KEY (constituency) REFERENCES constituency(con
                                 ON DELETE CASCADE ON UPDATE CASCADE,
```

```
                                FOREIGN KEY (institution) REFERENCES institution(name)
                                    ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE polling_station (
                                        eircode NCHAR(7) NOT NULL CHECK (eircode RLIKE
                                        name VARCHAR(24) NOT NULL,
                                       PRIMARY KEY (eircode)
);

CREATE TABLE station_register (
                                         constituency INT NOT NULL,
                                         polling_station NCHAR(7) NOT NULL,
                                        PRIMARY KEY (constituency, polling_station),
                                        FOREIGN KEY (constituency) REFERENCES constitue
                                            ON DELETE CASCADE ON UPDATE CASCADE,
                                        FOREIGN KEY (polling_station) REFERENCES polli
                                            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE candidate (
                                 candidate_id INT NOT NULL AUTO_INCREMENT,
                                 standing_election INT NOT NULL,
                                 fname VARCHAR(8) NOT NULL,
                                 lname VARCHAR(8) NOT NULL,
                                PRIMARY KEY (candidate_id),
                                FOREIGN KEY (standing_election) REFERENCES election(e
                                    ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE voter (
                           pps NCHAR(8) NOT NULL CHECK (pps RLIKE '[0-9]{6}[A-Z]'),
                           fname VARCHAR(8) NOT NULL,
                           lname VARCHAR(8) NOT NULL,
                           polling_station NCHAR(7) NOT NULL,
                          PRIMARY KEY (pps),
                          FOREIGN KEY (polling_station) REFERENCES polling_station(
                              ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE vote (
                         pps NCHAR(8) NOT NULL,
                         election INT NOT NULL,
                         voted_candidate INT NOT NULL,
                        PRIMARY KEY (pps, election),
                        FOREIGN KEY (pps) REFERENCES voter(pps)
```

```
                        ON UPDATE CASCADE ON DELETE CASCADE,
                    FOREIGN KEY (election) REFERENCES election(election_id)
                        ON UPDATE CASCADE ON DELETE CASCADE,
                    FOREIGN KEY (voted_candidate) REFERENCES candidate(candida
                        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE FUNCTION check_voter_valid(v_id NCHAR(8), election INT)
    RETURNS BOOLEAN
    DETERMINISTIC
BEGIN
    DECLARE is_valid BOOLEAN;
    SET is_valid = IF(
            (v_id IN (
                SELECT v.pps FROM voter v
                                    INNER JOIN polling_station p ON v.polling_
                                    INNER JOIN station_register sr ON p.eircod
                                    INNER JOIN constituency c on sr.constituen
                                    INNER JOIN election e on c.constituency_id
                WHERE e.election_id = election
            )),
            TRUE, FALSE);
    RETURN is_valid;
END;

CREATE FUNCTION check_candidate_valid(c_id INT, election INT)
    RETURNS BOOLEAN
    DETERMINISTIC
BEGIN
    DECLARE is_valid BOOLEAN;
    SET is_valid = IF(
            (c_id IN (
                SELECT candidate_id FROM candidate
                WHERE standing_election = election
            )),
            TRUE, FALSE);
    RETURN is_valid;
END;

CREATE FUNCTION check_vote_valid(c INT, v NCHAR(8), e INT)
    RETURNS BOOLEAN
    DETERMINISTIC
BEGIN
    DECLARE cand_valid BOOLEAN;
    DECLARE voter_valid BOOLEAN;
    DECLARE vote_valid BOOLEAN;
```

```
    SET cand_valid = check_candidate_valid(c, e);
    SET voter_valid = check_voter_valid(v, e);

    SET vote_valid = IF(cand_valid IS TRUE && voter_valid IS TRUE,
TRUE, FALSE);
    RETURN vote_valid;
END;


CREATE TRIGGER check_vote_insert_valid
    BEFORE INSERT ON vote FOR EACH ROW BEGIN
    DECLARE valid BOOLEAN;
    SET valid = check_vote_valid(NEW.voted_candidate, NEW.pps, NEW.election);
    IF (NOT VALID) THEN
        SIGNAL sqlstate '45000'
            SET MESSAGE_TEXT = 'Vote insert not valid!';
    END IF;
END;

CREATE TRIGGER check_vote_update_valid
    BEFORE UPDATE ON vote FOR EACH ROW BEGIN
    DECLARE valid BOOLEAN;
    SET valid = check_vote_valid(NEW.voted_candidate, NEW.pps, NEW.election);
    IF (NOT VALID) THEN
        SIGNAL sqlstate '45000'
            SET MESSAGE_TEXT = 'Vote update not valid!';
    END IF;
END;

CREATE PROCEDURE update_winning_cand(IN e INT) BEGIN
    DECLARE winner INT;
    SET winner = (SELECT voted_candidate FROM vote
                  WHERE election = e
                  GROUP BY voted_candidate
                  ORDER BY COUNT(voted_candidate)
                      DESC LIMIT 1);
    UPDATE election SET winning_cand = winner
    WHERE election_id = e;
END;


CREATE TRIGGER update_winning_cand_update
    AFTER UPDATE ON vote FOR EACH ROW
    CALL update_winning_cand(NEW.election);

CREATE TRIGGER update_winning_cand_insert
```

```
    AFTER INSERT ON vote FOR EACH ROW
    CALL update_winning_cand(NEW.election)
```

B DCL

```
CREATE VIEW registrar_view AS
SELECT v.pps, CONCAT(v.fname, ' ', v.lname) AS voter_name, ps.eircode, ps.name A
FROM voter v
        INNER JOIN polling_station ps on v.polling_station = ps.eircode;

CREATE FUNCTION has_voter_voted(v NCHAR(8), e INT) RETURNS BOOLEAN
    DETERMINISTIC
BEGIN
    DECLARE voted BOOLEAN;
    SET voted = IF(v IN (SELECT pps FROM vote WHERE election = e), TRUE, FALSE);
    RETURN voted;
END
;


CREATE VIEW polling_official_view AS
SELECT e.election_id, v.pps, CONCAT(v.fname, ' ', v.lname) AS candidate_name,
       ps.eircode, ps.name AS polling_station, has_voter_voted(v.pps, e.election
FROM voter v
        INNER JOIN polling_station ps on v.polling_station = ps.eircode
        INNER JOIN station_register sr on ps.eircode = sr.polling_station
        INNER JOIN constituency c on sr.constituency = c.constituency_id
        INNER JOIN election e on c.constituency_id = e.constituency;

CREATE VIEW returner_view AS
SELECT e.*, i.building, i.chair FROM election e
                        INNER JOIN institution i on e.institution = i.name;

CREATE VIEW voter_view AS
SELECT DISTINCT CONCAT(c.fname, ' ', c.lname) AS cand_name, e.election_id, ps.ei
FROM candidate c
        INNER JOIN election e on c.standing_election = e.election_id
        INNER JOIN constituency co on e.constituency = co.constituency_id
        INNER JOIN station_register sr on co.constituency_id = sr.constituency
        INNER JOIN polling_station ps on sr.polling_station = ps.eircode
        INNER JOIN voter v on ps.eircode = v.polling_station;

CREATE ROLE registrar;
GRANT SELECT, UPDATE ON registrar_view TO registrar;
```

```
CREATE ROLE polling_official;
GRANT SELECT ON polling_official_view TO polling_official;
CREATE ROLE returner;
GRANT SELECT ON returner_view TO returner;
CREATE ROLE voter;
GRANT SELECT ON voter_view TO voter;
GRANT INSERT ON vote TO voter;
```

C DML

```
INSERT INTO institution
VALUES
('Dail', 'Leinster House', 'Joe Bloggs'),
('Seanad', 'Leinster House', 'Boe Jloggs'),
('European Parliament', 'Brussels Building', 'Donald Tusk'),
('Cork County Council', 'Cork City Hall', 'Arnold Parsimmons'),
('Dublin City Council', 'Dublin City Hall', 'Cian Walsh');

INSERT INTO constituency (name, pop)
VALUES
('Kerry', '50000'),
('Dublin', '1000000'),
('Cork', '500000'),
('Galway', '100000'),
('Limerick', '150000');

INSERT INTO election (constituency, institution)
VALUES
(1, 'Dail'),
(1, 'Seanad'),
(2, 'Dublin City Council'),
(3, 'Cork County Council'),
(4, 'European Parliament');

INSERT INTO polling_station
VALUES
('V23RW88', 'Aghatubrid NS'),
('A12BC34', 'Dublin City Hall'),
('X12RW43', 'Cork City Hall'),
('A43BC12', 'LimCity Secondary School'),
('A12DE54', 'Kerry County Hall'),
('D44HJ87', 'Galway College'),
('A22IM98', 'Blackrock College');
```

```
INSERT INTO station_register
VALUES
(1, 'V23RW88'),
(1, 'A12DE54'),
(2, 'A12BC34'),
(2, 'A22IM98'),
(3, 'X12RW43'),
(4, 'A43BC12'),
(5, 'D44HJ87');

INSERT INTO voter
VALUES
('1234567A', 'A', 'B', 'A12BC34'),
('1234567Z', 'A', 'B', 'A12BC34'),
('1234567B', 'B', 'C', 'A12BC34'),
('1234567C', 'D', 'E', 'A12BC34'),
('1234567D', 'F', 'G', 'X12RW43'),
('1234567E', 'H', 'I', 'X12RW43'),
('1234567F', 'J', 'K', 'A12DE54'),
('1234567G', 'L', 'M', 'A12DE54'),
('1234567H', 'N', 'O', 'V23RW88'),
('1234567I', 'Q', 'P', 'A12DE54'),
('1234567J', 'R', 'S', 'A22IM98'),
('1234567L', 'U', 'X', 'A22IM98'),
('1234567K', 'Y', 'V', 'V23RW88'),
('1234567M', 'Z', 'W', 'A22IM98'),
('1234567N', 'AA', 'BB', 'A22IM98'),
('1234567O', 'AB', 'BD', 'V23RW88'),
('1234567P', 'AC', 'BR', 'A43BC12'),
('1234567Q', 'AV', 'BQ', 'A43BC12'),
('1234567R', 'AG', 'BY', 'D44HJ87'),
('1234567S', 'AM', 'BY', 'D44HJ87');

INSERT INTO candidate (standing_election, fname, lname)
VALUES
(1, 'Cian', 'Walsh'),
(1, 'Andrew', 'Garfield'),
(1, 'James', 'Joyce'),
(2, 'Cian', 'Qalsh'),
(2, 'Randrew', 'Sarfield'),
(2, 'Matthew', 'Henry'),
(3, 'Spatthew', 'Menry'),
(3, 'Ondrow', 'Gorfald'),
(3, 'Kiera', 'Cullen'),
(4, 'Nessa', 'Walsh'),
(4, 'Cathal', 'Walsh'),
```

```
(4, 'Oscar', 'Wilde'),
(5, 'Ada', 'Lovelace'),
(5, 'George', 'Boole'),
(5, 'Robert', 'Emmet'),
(5, 'Martin', 'Emms'),
(5, 'Karl', 'Marx');

INSERT INTO vote
VALUES
('1234567O', 1, 1),
('1234567K', 1, 2),
('1234567H', 1, 2),
('1234567L', 3, 7),
('1234567M', 3, 7);
```