CPSC 304 Project Cover Page

Milestone #: 4

Date: tba

Group Number: 85

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Misty Penner	26615161	d5n1b	baileypenner@gmail.com
Clara He	36882223	h4z2q	clarahct1102@gmail.com
Andrew Cai	10673291	z0a7g	acaiclouds@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Project Description

Our application is a one stop place to get all the info people could possibly need during an election season. It includes information about candidates, ridings, allows people to register as voters, and allows people to vote.

Schema

We mostly stuck entirely to our original schema for the final schema, however we made some changes. When we learned that Oracledb doesn't support booleans, we changed our booleans to integers and use the values 0 and 1 to indicate true and false. Ballot ID also got changed from an Integer to one that the database automatically populates itself, so that it can handle creating new unique IDs.

SQL Script

Can be found as votingApp.sql in the main directory.

SQL Queries

1. Insert
    - ```
      INSERT INTO PostalCode_City_Province ( PostalCode, City, Province ) VALUES ( :postalCode, :city, :province )

      INSERT INTO VOTER ( SIN, Email, FullName, DOB, Voted, Party_PartyName, Role, StreetAddress, PostalCode ) VALUES ( :sin, :email, :fullName, :dob, :voted, :party_PartyName, :role, :streetAddress, :postalCode )
      ```
    - Server-side implementation can be found in appController.js starting at line 84, and at appService.js starting at line 168.
2. Update
    - ```
      UPDATE Voter SET PostalCode = :postalCode WHERE SIN = :sin
      ```
    - (other copies of update query not included for brevity)
    - Server-side implementation can be found in appController.js starting at line 141, and at appService.js starting at line 217.
3. Delete
    - ```
      DELETE FROM Party WHERE PartyName = :partyName
      ```
    - Server-side implementation can be found in appController.js starting at line 184, and at appService.js starting at line 275.
4. Selection
    - ```
      SELECT * FROM Riding` + where + finalQuery, finalQueryObject
      ```

- Server-side implementation can be found in appController.js starting at line 196, and at appService.js starting at line 284.

5. Projection
   - `SELECT ` + projectionParams + ` FROM Party`
   - Server-side implementation can be found in appController.js starting at line 215, and at appService.js starting at line 449.
6. Join
   - `SELECT Voter.FullName, Voter.SIN FROM Voter, Candidate, PollingStation WHERE Voter.SIN = Candidate.SIN AND Candidate.Riding_ID = PollingStation.Riding_ID AND PollingStation.Name = :polling_station_name`
   - Server-side implementation can be found in appController.js starting at line 228, and at appService.js starting at line 482.
7. Aggregation with GROUP BY
   - This query sums up the total number of votes cast in the riding.
   - `SELECT Riding.Name, SUM(Votes) FROM Riding, Candidate WHERE Riding.ID = Candidate.Riding_ID GROUP BY Riding.Name`
   - Server-side implementation can be found in appController.js starting at line 238, and at appService.js starting at line 493.
8. Aggregation with HAVING
   - This query returns all the parties with official party status (at least 13 elected members) and their total seat counts.
   - `SELECT Voter.Party_PartyName, Count(Voter.SIN) FROM Voter, Candidate, Parliament WHERE Voter.SIN = Candidate.SIN AND Candidate.Parliament_Country <> NULL GROUP BY Voter.Party_PartyName HAVING COUNT(*) > 12`
   - Server-side implementation can be found in appController.js starting at line 249, and at appService.js starting at line 505.
9. Nested Aggregation with GROUP BY
   - `SELECT * FROM Riding r WHERE r.Population > (SELECT AVG(r2.Population)FROM Riding r2 WHERE r2.Province_Name = r.Province_Name GROUP BY r2.Province_Name)`
   - Server-side implementation can be found in appController.js starting at line 260, and at appService.js starting at line 515.
10. Division
    - This query returns the policies that all the parties have in common.
    - `SELECT * FROM Policy P WHERE NOT EXISTS ((SELECT Pa.PartyName FROM Party Pa) MINUS (SELECT Po.Party_PartyName FROM Policy Po WHERE P.PolicyName = Po.PolicyName))`
    - Server-side implementation can be found in appController.js starting at line 271, and at appService.js starting at line 525.