University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___4_____

Date: ___Nov 29, 2024_____

Group Number: ___99_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Alex Kwok	70386099	g8j7	a.kwok0191@gmail.com
Byeori Kim	74612821	p2i1w	Bk.byeori.kim@gmail.com
Jun Lee	21913603	v2l8o	Junemessi040714@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

University of British Columbia, Vancouver
Department of Computer Science

Project Description

This project focuses on designing and implementing a database to model food charity management across Canada. The database captures the intricate network of food logistics, donor management, recipient tracking, and inventory optimization. By centralizing this information, the database aims to streamline the operations of food charities, ensuring that donations are efficiently distributed to communities in need. A robust data model enables charities to better allocate resources, reduce food waste, and optimize delivery routes, improving their overall impact. Furthermore, such a system provides transparency and accountability, which is crucial for maintaining trust with donors and stakeholders.

This database achieves several key objectives. It tracks the demographic profiles and needs of food recipients, helping charities understand and address disparities in access to food support. By managing donated food inventory, it minimizes spoilage and ensures timely distribution based on demand. The system also monitors donor activity, enabling charities to recognize frequent contributors and identify opportunities for new partnerships. Overall, this tool empowers food charities to operate more efficiently, maximize their impact, and support the growing number of individuals and families relying on their services.

SQL DDL Script

project_g8j7_p2i1w_v2l8o/src/scripts/ in our repo, file named databasesubmission.sql.

- **note that we have a separate function called to drop all tables automatically, so not needed in our script that runs in our code. we have two sql files in our repo, one being with drop tables (for submission) and one being without. they both work but the other seemed cleaner in terminal.**

link:
https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_g8j7_p2i1w_v2l8o/blob/main/src/scripts/databasesubmission.sql

Changes to Schema

Our overall schema remains the same as in milestone 2. We made the suggested corrections to our normalization tables in response to feedback given to us by our TA. For completeness, we reviewed our entire milestone 2 submission and found some discrepancies that we corrected. This includes normalizing the charity worker entity based on the functional dependency that we specified, and minor corrections to out-of-place/ missing attributes.

An observation that we noted as we decided on our queries was the many normalized tables that we had. Almost every relation in our model was normalized which added complexity to our SQL queries. In retrospect, some functional dependencies were not necessary which would avoid the need to normalize to BCNF. We decided to remove the EventNameDate table (part of the DonationEvent relation) and AddressProvince table (part of the Charities relation): both tables were the result of normalization. This decision was made because we felt the functional

dependencies that we proposed were not too strong and it added unnecessary complexity to our SQL queries. Lastly, we added “ON DELETE CASCADE” to all our SQL DDL statements which we missed in milestone 2.

SQL Query Description

All Our SQL Queries can be found in this path: src/appService.js!

INSERT Query

This code is found on line 163. This query enables adding new recipients to the recipients table.

```
`INSERT INTO Recipients (SinNum, EventID, Age, ContactNum, Gender)
 VALUES (:SinNum, :EventID, :Age, :ContactNum, :Gender)`,
```

UPDATE Query

This code is found on line 191. This query enables the updating of existing recipients' information in the recipients table.

```
  UPDATE Recipients
  SET ${updateFields.join(', ')}
  WHERE SinNum = :SinNum
```

DELETE Query

This code is found on line 417. This query enables the deletion of donors in the FoodDonor table.

```
'DELETE FROM FoodDonors WHERE DonorID = :donorId'
```

University of British Columbia, Vancouver
Department of Computer Science

Selection Query

This code is found on line 283. This query enables the selection on the charity table with an arbitrary number of clauses of AND/OR.

`'SELECT CharityID, Name, Address FROM Charities'` + dynamic additions of WHERE … with:

```
if (whereClauses.length > 0) {
    sqlQuery += ' WHERE ';

    sqlQuery += whereClauses[0];

    for (let i = 1; i < whereClauses.length; i++) {
        sqlQuery += ` ${logicalOperators[i-1]} ${whereClauses[i]}`;
    }
```

Projection Query

This code is found on line 381. This query enables projection of any columns found in the charity table.

```
const query = `SELECT ${selectedColumns} FROM Charities`;
```

Join Query

This code is found on line 238. This query joins the FoodReceived table with the Recipients table under the condition that the Sin number is equal.

```
`SELECT
    FR.SinNum,
    R.ContactNum
FROM
    FoodReceived FR
JOIN
    Recipients R ON FR.SinNum = R.SinNum
WHERE
    FR.FoodID = :foodID`,
```

Aggregation with GROUP BY Query

This query is found on line 339. This query lists all the ages and it's count number of recipients in the database.

```
`SELECT Age, COUNT(*) AS AGE_COUNT
 FROM Recipients
 GROUP BY Age`
```

Aggregation With Having Query

This query is found on line 261. This query finds the event ids with more than 2 attendees (recipients) and returns the average age of recipients in those events.

```
`SELECT
    R.EventID,
    ROUND(AVG(R.Age), 2) AS AVG_AGE,
    COUNT(*) AS RECIPIENT_COUNT
FROM
    Recipients R
GROUP BY
    R.EventID
HAVING
    COUNT(*) >= 2`
```

Nested aggregation With GROUP BY query

This code is found on line 356. This query finds the event IDs where the average age of attendees (recipients) in that event is less than or equal to the average age of all recipients in the database.

```
`SELECT DE.EventID, DE.EventName, AVG(R.Age) AS AVG_EVENT_AGE
FROM DonationEvent DE
JOIN Recipients R ON DE.EventID = R.EventID
GROUP BY DE.EventID, DE.EventName
HAVING AVG(R.Age) <= ALL (
    SELECT AVG(R2.Age)
    FROM Recipients R2
    GROUP BY R2.EventID
```

Division Query

This code is found on line 393. This query finds the contact number of recipients who have attended all Donation Events (EventID).

```
SELECT DISTINCT R1.ContactNum
FROM Recipients R1
WHERE NOT EXISTS (
    SELECT E.EventID
    FROM Recipients E
    WHERE NOT EXISTS (
        SELECT R2.ContactNum
        FROM Recipients R2
        WHERE R2.ContactNum = R1.ContactNum AND R2.EventID = E.EventID
```