University of British Columbia, Department of Computer Science

CPSC 304

Cover Page for Project Part Project Formal Specification

Date: ___ October 23, 2018_____

Group Members:

Name	Student Number	CS Userid	Tutorial Section	Email Address
Brandon Chung	11371168	m1u0b	T1H	b.chung8612@gmail.com
Cole O'Brien	32693160	f9v0b	T1H	rcoleobrien@gmail.com
Jiali Fan	56214307	t4f2b	T1E	16302010062@fudan.edu.cn
Adle Ker Hue Chiam	22669162	k4y0b	T1D	adlekerhuechiam@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 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

Deliverable 1: Identify the different kinds of users that your application will service.

Type of Users	Actions
Common Users	1. Befriend other users 2. Follow Company accounts 3. Join Groups(become a member) 4. Create Groups(become the founder) 5. Publish post 6. Comment on post 7. Like post 8. Create Events 9. RSVP to Events 10. Play games (gaming session record)
Company	1. Has Employees 2. Join Groups(become a member) 3. Create Groups(become the founder) 4. Publish post 5. Comment on post 6. Like post 7. Create Events 8. RSVP to Events 9. Play games
Admin	1. Monitors group's post 2. Edit group's information

Deliverable 2: Insert Statements.

Table	Function	Statement
Users	Create a new user. (UID uses AUTO_INCREMENT)	INSERT INTO Users (Name, Password, Email) VALUES ('John', 1234567, '1234567@gmail.com')
Common_Users	Keep record of a common user's ID. (Suppose User whose ID is 5 is a common user)	INSERT INTO Common_Users VALUES (5)
Company	Keep record of a company's ID (Suppose User whose ID is 7 is a company and type is airline)	INSERT INTO Company VALUES (7, 'Airline')
Employs_Employee	Keep record of employees employed by certain company. (EMID uses AUTO_INCREMENT) Suppose the employee named Lucy is employed by a company whose CID is 7.	INSERT INTO Employs_Employee (CID, Ename) VALUES (7, 'Lucy')
Follow	Keep record of the follow relationship between a common user and a company.	INSERT INTO Follow VALUES (5, 7, '2018-10-20')
Befriend	Keep record of the friendship between two users.	INSERT INTO Befriend VALUES (4, 5)
Game	Keep record of games. (GID uses AUTO_INCREMENT)	INSERT INTO Game (Name, Type, Description) VALUES ('Super Mario', 'Adventure', 'So much fun!')
Play	Keep record of games played by users. Suppose user whose UID is 5 played a game whose	INSERT INTO Play VALUES (1, 5, 100)

	GID is 1 and got a score of 100.	
Creates_Event	Keep record of events with UID of its creator, name, time, date, location and description. (EID uses AUTO_INCREMENT)	INSERT INTO Creates_Event (UID, Name, Time, Date, Location, Description) VALUES (5, 'Skiing', '08:00:00', '2018-12-20', 'Vancouver', ' So much fun!!')
RSVP	Keep record of users participating in some events.	INSERT INTO RSVP VALUES (5, 1, '08:00:00', '2018-10-20')
Post	Keep record of posts with UID of its author, location, time, date, photo_file and text. (PostID uses AUTO_INCREMENT)	INSERT INTO Post (UID, Location, Time, Date, Photo_File, Text) VALUES (5, 'UBC', '09:00:00', '2018-10-20', '1.jpg', 'Studying')
Like	Keep record of which posts are liked by some users.	INSERT INTO Like VALUES (5, 1)
User_Comments_Posts	Keep record of which comment users left under the posts. (CommentID uses AUTO_INCREMENT)	INSERT INTO User_Comments_Posts (UID, PostID, Text) VALUES (5, 10, 'So cool!')
Group	Keep record of groups with its name, description, type, founder ID, and founder. (GID uses AUTO_INCREMENT)	INSERT INTO Group (Name, Description, Type, FounderID, Founder) VALUES ('UBC CPSC304', 'Group for UBC CPSC304 2018 Fall', 'Class', 5, 'John')
Group_Contains_Post	Keep record of the group a post is in. (Suppose the post's ID is 100 and it belongs to the group with GID 10)	INSERT INTO Group_Contains_Post VALUES (10, 100)
Admins	Keep record of the administrators of all groups.	INSERT INTO Admins VALUES (6, 'Mary', 'Manager')

Admin_Monitors_Agg_GP	Keep record of the aggregation of ‘The posts posted by the users are monitored by the administrators’	INSERT INTO Admin_Monitors_Agg_GP VALUES (1, 5, 50, FALSE, ‘Inappropiate’)

Deliverable 3: Delete Statements.

Table	Function	Statement
Users	1. Delete a user whose UID is 6 2. Delete a user whose email is '1234567@gmail.com'	1. DELETE FROM Users WHERE UID = 6 2. DELETE FROM Users WHERE Email = '1234567@gmail.com'
Common_Users	Delete a common user whose UID is 5	DELETE FROM Common_Users WHERE UID = 5
Company	Delete a company whose CID is 7	DELETE FROM Company WHERE CID = 7
Employs_Employee	1. Delete the employment relation of an employee named Lucy 2. Delete the employment relations of a company whose CID is 7	1. DELETE FROM Employs_Employee WHERE Ename = 'Lucy' 2. DELETE FROM Employs_Employee WHERE CID = 7
Follow	1. Delete the follow relations of a user whose UID is 5. 2. Delete the follow relations of a company whose CID is 7. 3. Delete the follow relation between user with UID 5 and company with CID 7.	1. DELETE FROM Follow WHERE UID = 5 2. DELETE FROM Follow WHERE CID = 7 3. DELETE FROM Follow WHERE UID = 5 AND CID = 7
Befriend	1. Delete the friendships of a user whose UID is 5. 2. Delete the friendship between users with UID 5 and 10. (When we insert the tuple, we specify that UID1 < UID2)	1. DELETE FROM Befriend WHERE UID1 = 5 OR UID2 = 5 2. DELETE FROM Befriend WHERE UID1 = 5 AND UID2 = 10.
Game	1. Delete the game with GID 8. 2. Delete the games with Name 'Super Mario'	1. DELETE FROM Game WHERE GID = 8 2. DELETE FROM Game WHERE Name = 'Super Mario'
Play	1. Delete the records of game with GID 8.	1. DELETE FROM Play WHERE GID = 8

	2. Delete the records of user with UID 5.	2. DELETE FROM Play WHERE UID = 5
Creates_Event	1. Delete events through UID 2. Delete events through EID	1. DELETE FROM Creates_Event WHERE UID = 5 2.DELETE FROM Creates_Event WHERE EID = 9
RSVP	1. Delete the records of events user with UID 5 participates in. 2. Delete the records of users participating in the event with EID 10. 3. Delete the record of user with UID 5 participating in event with EID 10.	1. DELETE FROM RSVP WHERE UID = 5 2. DELETE FROM RSVP WHERE EID = 10 3. DELETE FROM RSVP WHERE UID= 5 AND EID = 10
Post	1. Delete all the posts from user with UID 5.2 2. Delete the post with PostID 23.	1. DELETE FROM Post WHERE UID = 5. 2. DELETE FROM Post WHERE PpstID = 23.
Like	1. Delete all the likes for post with PostID 23. 2.Delete the like for post with PostID 23 from user with UID 5.	1. DELETE FROM Like WHERE PostID = 23. 2. DELETE FROM Like WHERE PostID = 23 AND UID = 5.
User_Comments_Posts	1. Delete all the comments for post with PostID 23. 2.Delete the comments for post with PostID 23 from user with UID 5. 3. Delete one specific comment with CommentID 50.	1. DELETE FROM User_Comments_Posts WHERE PostID = 23 2. DELETE FROM User_Comments_Posts WHERE PostID = 23 AND UID = 5 3. DELETE FROM User_Comments_Posts WHERE CID = 50
Group	Delete a group with GID 14.	DELETE FROM Group WHERE GID = 14
Group_Contains_Post	1. Delete all the records of the posts under the group with GID 14. 2. Delete a specific post with PostID 23.	1. DELETE FROM Group_Contains_Post WHERE GID = 14 2. DELETE FROM Group_Contains_Post WHERE PostID = 23

Admins	Delete an adminstrator with AID 15	DELETE FROM Admins WHERE AID = 15
Group_Member	1. Delete all members under group with GID 4 2. Delete a specific member from the group.	1. DELETE FROM Group_Member WHERE GID = 4 2. DELETE FROM Group_Member WHERE GID = 4 AND MID = 10
Admin_Monitors_Agg_GP	1. Delete the record for a specific post 2. Delete the records for the posts under a specific group.	1. DELETE FROM Admin_Monitors_Agg_GP WHERE PostID = 23 2. DELETE FROM Admin_Monitors_Agg_GP WHERE GID = 10

Deliverable 4: Update Statements.

Table	Function	Statement
Users	Update the user's information.	UPDATE Users SET Email = '7654321@gmail.com' WHERE UID = 5
Company	Update the company type.	UPDATE Company SET Type = 'Business' WHERE CID = 7
Employs_Employee	Update the employee's information.	UPDATE Employs_Employee SET CID = 17 WHERE EMID = 10
Game	Update the name or type of the game.	UPDATE Game SET Name = 'Sudoku' AND Type = 'Puzzle' WHERE GID = 2
Play	Update the score the user gets in the game.	UPDATE Play SET Score = 200 WHERE GID = 2 AND UID = 5
Creates_Event	Update the information of the event.	UPDATE Creates_Event SET Name = 'Party' AND Location = 'UBC' WHERE EID = 14
Post	Update the information of the post.	UPDATE Post SET Location = 'UBC' AND Date = '2018-10-21' AND Text = 'Mid!' WHERE PostID = 23
Group	Update the group information.	UPDATE Group SET Name = 'Camping' AND Type = 'Friend'
Admins	Update the name or type of an administrator.	UPDATE Admins SET Name = 'Lucy' AND Title = 'Clerk'
Admin_Monitors_Agg_GP	Update the approval state and reason.	UPDATE Admin_Monitors_Agg_GP SET ApprovalState = 'True' AND Reason = 'Appropiate'

Deliverable 5 and 6: Join Statements.

Function	Statement
Table of users and the companies they follow, with possible search functionality. Ex. Find Company names followed by User named “Jeff Geoff” by joining User with Follows, then joining that with Company	SELECT c.name FROM users u INNER JOIN follows f on u.UID == f.UID INNER JOIN company c on f.CID == c.CID WHERE u.name == “Jeff Geoff”
Table of Users and Posts they Like. Ex find UID of all Users that like Post with postID “0987”	SELECT u.UID FROM users u INNER JOIN like on u.UID == like.UID INNER JOIN post p on like.postID == p.postID WHERE post.postID == “0987”
Table of groups and the posts they contain. Ex find all postID’s in Group “SQL Squirrels”	SELECT p.postID FROM group g INNER JOIN has h on g.GID == .GID INNER JOIN post p on h.postID == p.postID WHERE g.name == “SQL Squirrels”
Table of users and the games they play. Ex find the high score for game “Bad Rats 2”	SELECT MAX(p.score) FROM user u INNER JOIN play p on u.UID p.UID INNER JOIN game g on p.GID == g.GID WHERE g.name == “Bad Rats 2”

Deliverable 7: At least one query must be an interesting GROUP BY query (aggregation). Describe it.

Function	**Statement**
Count of games play sessions by type (popularity of various types of games)	SELECT COUNT(G.ID), G.type FROM [games] G JOIN play P ON G.ID = P.GameID GROUP BY G.type ORDER BY COUNT(G.ID)
Count of groups memberships by type (popularity of various types of groups using membership)	SELECT COUNT(G.ID), G.type FROM [groups] G JOIN members M ON G.ID = M.GgroupID GROUP BY G.type ORDER BY COUNT(G.ID)

Deliverable 8, 9, 10: Other Queries

```
CREATE TABLE Users(
        UID             INTEGER,
        Name            CHAR(20),
        Password        CHAR(20),
        Email           CHAR(30),
        PRIMARY KEY (UID))
```

```
CREATE TABLE Common_Users(
        UID             INTEGER,
        PRIMARY KEY (UID),
        FOREIGN KEY (UID) REFERENCES Users
                                ON DELETE CASCADE
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Company(
        CID             INTEGER,
        Type            CHAR(10),
        PRIMARY KEY (CID),
        FOREIGN KEY (CID) REFERENCES Users
                                ON DELETE CASCADE
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Employs_Employee(
        EMID            INTEGER,
        CID             INTEGER,
        Ename           CHAR(20),
    PRIMARY KEY (EMID, CID),
    FOREIGN KEY (CID) REFERENCES Company
                        ON DELETE CASCADE
                        ON UPDATE CASCADE)
```

```
CREATE TABLE Follow(
        UID             INTEGER,
        CID             INTEGER,
        FollowDate      Date,
    PRIMARY KEY (UID, CID),
    FOREIGN KEY (UID) REFERENCES Common_Users
                                ON DELETE CASCADE
                                ON UPDATE CASCADE,
    FOREIGN KEY (CID) REFERENCES Company
                                ON DELETE CASCADE
                                ON UPDATE SET NULL)
```

```
CREATE TABLE Befriend(
        UID1            INTEGER,
        UID2            INTEGER,
        PRIMARY KEY (UID1, UID2),
    FOREIGN KEY (UID1, UID2) REFERENCES Common_Users
                                ON DELETE CASCADE
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Game(
        GID             INTEGER,
        Name            CHAR(20),
        Type            CHAR(20),
        Description     VARCHAR(1000),
        PRIMARY KEY (GID))
```

```
CREATE TABLE Play(
        GID                     INTEGER,
        UID                     INTEGER,
        Score                   INTEGER,
        PRIMARY KEY (GID, UID),
        FOREIGN KEY (GID) REFERENCES Game
                                ON DELETE SET NULL
                                ON UPDATE CASCADE,
        FOREIGN KEY (UID) REFERENCES Users
                                ON DELETE SET NULL,
                                ON UPDATE CASCADE
```

```
CREATE TABLE Creates_Event(
        EID                     INTEGER,
        UID                     INTEGER,
        Name                    CHAR(20),
        Time                    TIME,
        Date                    DATE,
        Location                CHAR(20),
        Description             VARCHAR(1000),
        PRIMARY KEY (EID),
        FOREIGN KEY (UID) REFERENCES Users
                                ON DELETE SET NULL
                                ON UPDATE CASCADE)
```

```
CREATE TABLE RSVP(
        UID             INTEGER,
        EID             INTEGER,
        Time            TIME,
        Date            DATE,
        PRIMARY KEY (UID, EID),
        FOREIGN KEY (UID) REFERENCES Users
                                ON DELETE SET NULL
                                ON UPDATE CASCADE)
        FOREIGN KEY (EID) REFERENCES Create_Events
                                ON DELETE SET NULL
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Post(
        PostID                  INTEGER,
        UID                     INTEGER,
        Location                CHAR(30),
        Time                    TIME,
```

```
        Date                DATE,
        Photo_File          CHAR(100)
        Text                VARCHAR(1000)
        PRIMARY KEY (PostID),
        FOREIGN KEY (UID) REFERENCES Users
                            ON DELETE SET NULL
                            ON UPDATE CASCADE)
```

```
CREATE TABLE Like(
        UID                 INTEGER,
        PostID              INTEGER,
        PRIMARY KEY (UID, PostID),
        FOREIGN KEY (UID) REFERENCES Users
                            ON DELETE CASCADE
                            ON UPDATE CASCADE)
        FOREIGN KEY (PostID) REFERENCES Post
                            ON DELETE CASCADE
                            ON UPDATE CASCADE)
```

```
CREATE TABLE User_Comments_Posts(
        CommentID           INTEGER
        UID                 INTEGER,
        PostID              INTEGER,
        Text                VARCHAR(1000),
        PRIMARY KEY (CommentID),
        FOREIGN KEY (UID) REFERENCES Users
                            ON DELETE SET NULL
                            ON UPDATE CASCADE)
        FOREIGN KEY (PostID) REFERENCES Post
                            ON DELETE CASCADE
                            ON UPDATE CASCADE)
```

```
CREATE TABLE Group(
        GID                 INTEGER,
        Name                CHAR(20),
        Description         VARCHAR(1000),
        Type                CHAR(20),
        FounderID           INTEGER,
        Founder             CHAR(20),
        PRIMARY KEY (GID),
        FOREIGN KEY (UID) REFERENCES Users
                            ON DELETE SET NULL
                            ON UPDATE CASCADE)
```

```
CREATE TABLE Users_Becomes_Members(
        UID                 INTEGER,
        MID                 INTEGER,
        PRIMARY KEY (UID,GID),
        FOREIGN KEY (UID) REFERENCES Users
                            ON DELETE CASCADE
                            ON UPDATE CASCADE,
```

```
        FOREIGN KEY (GID) REFERENCES Members
                                ON DELETE CASCADE
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Group_Contains_Post(
        GID                     INTEGER,
        PostID                  INTEGER,
        PRIMARY KEY (GID, PostID),
        UNIQUE (PostID),
        FOREIGN KEY (GID) REFERENCES Group
                                ON DELETE CASCADE
                                ON UPDATE CASCADE
        FOREIGN KEY (PostID) REFERENCES Post
                                ON DELETE CASCADE
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Members(
        MID                     INTEGER,
        UID                     INTEGER
        Name                    CHAR(20)
        Title                   CHAR(20)
        PRIMARY KEY (MID)
        FOREIGN KEY (UID) REFERENCES Users
                                ON DELETE CASCADE
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Admins(
        AID                     INTEGER,
        Name                    CHAR(20),
        Title                   CHAR(20),
        PRIMARY KEY (AID))
```

```
CREATE TABLE Group_Member(
        GID             INTEGER,
        MID             INTEGER,
        PRIMARY KEY (GID, MID)
        FOREIGN KEY (GID) REFERENCES Group
                                ON DELETE CASCADE
                                ON UPDATE CASCADE
        FOREIGN KEY (MID) REFERENCES Users
                                ON DELETE CASCADE,
                                ON UPDATE CASCADE)
```

```
CREATE TABLE Admin_Monitors_Agg_GP(
        AID                     INTEGER,
        GID                     INTEGER,
        PostID                  INTEGER,
        ApprovalState           BOOLEAN
        Reason                  VARCHAR(1000),
        PRIMARY KEY (AID, GID, PostID),
        FOREIGN KEY (AID) REFERENCES Users
                                ON DELETE CASCADE
```

```
                    ON UPDATE CASCADE,
FOREIGN KEY (GID) REFERENCES Group
                    ON DELETE CASCADE
                    ON UPDATE CASCADE,
FOREIGN KEY (PostID) REFERENCES Post
                    ON DELETE CASCADE
                    ON UPDATE CASCADE)
```

Deliverable 11: You need to have at least one view for your database, created using the CREATE VIEW statement in SQL. It should be a proper subset of a table. When the user wants to display all the data from this view, they will get all the columns specified by the view, but not all of the columns in the underlying table.

- You don't have to give the SQL for the view yet; just indicate the subset of which table(s) that will be used, and for which kind of user.

Function	**Statement**
Table View of all friendships. Names of who is friends with who. Ex. (Jeff, George)	CREATE VIEW friendships AS SELECT u1.Name, u2.Name FROM [Common Users] u1 JOIN befriend B ON u1.ID = B.UID1 JOIN Common Users u2 ON u2.ID = B.UID2
Table View of all games of type X. Ex. Games of type "builder"	CREATE VIEW buiderGames AS SELECT * FROM Games WHERE type = "builder"

Deliverable 12: Division of Labor

Brandon Chung:

1. Code & embed part of the SQL statements in a program.
2. Document the project.

Cole O'Brien:

1. Create the tables and other database objects.
2. Document the project.
3. Test each set of queries.

Jiali Fan:

1. Create data for the tables
2. Embed part of the SQL statements in the program.
3. Test each set of queries.

Adle Ker Hue Chiam:

1. Code & embed part of the SQL statements in a program.
2. Test each set of queries.