CS3041

MySQL Project Report

Lee Campbell | campbel2@tcd.ie

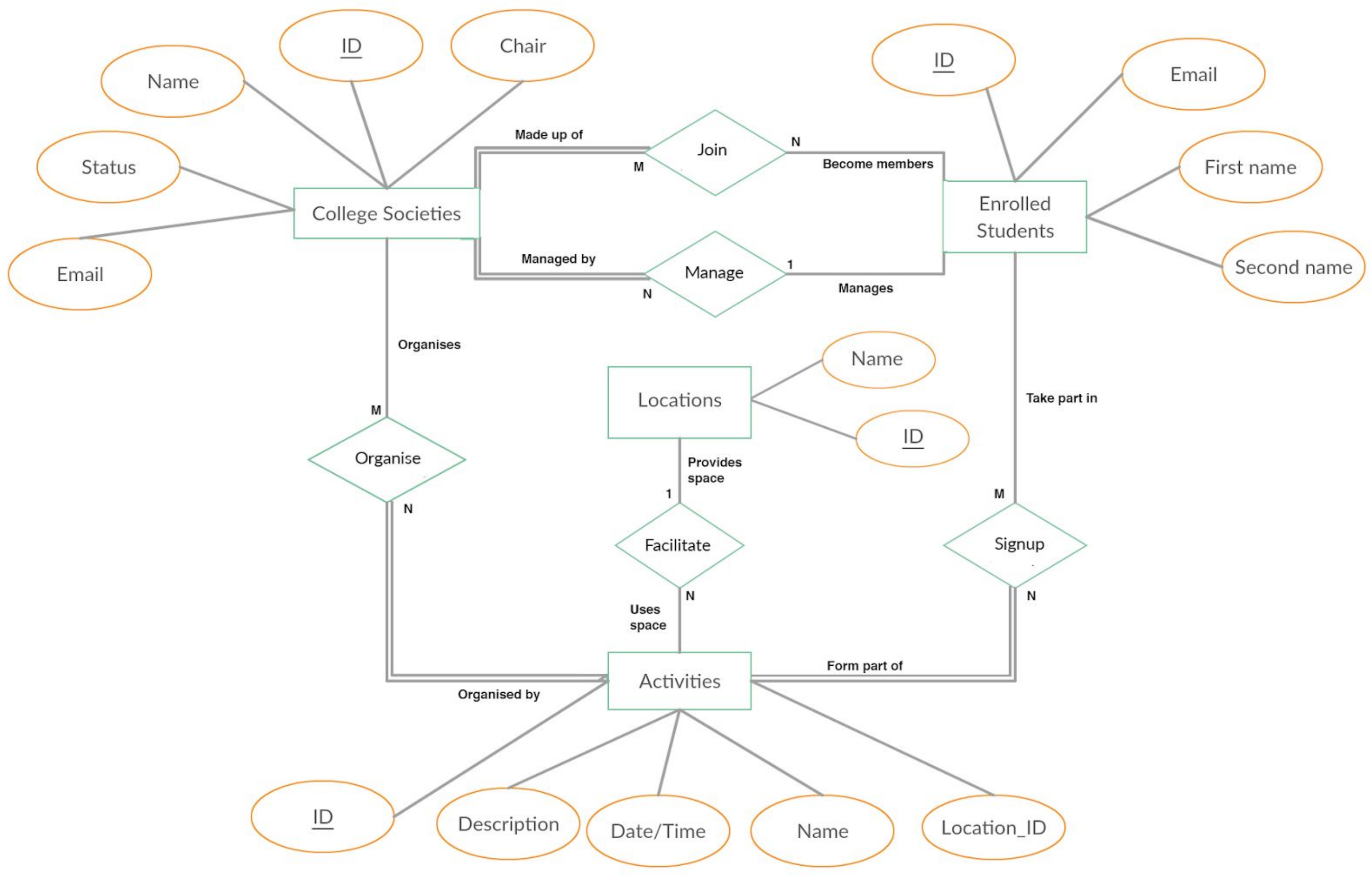
Overview

The given task for this project was to create a database around something of interest to us; using MySQL and based on the concepts we studied over the course of the CS3041 module. As the current Chair of the TEDxTrinityCollegeDublin society, I decided to model a database which represents each of the societies ratified by the Central Societies Committee (CSC), their members and their activities. This database could be used by CSC for both internal administration purposes and externally to provide the student body with information regarding society events for e.g. via it's website which could query the database. While the extent to which you can model information into a database regarding college societies is quite large; in keeping with the scope of this project, I modelled the most common and important data pertaining to college societies in the following entities:

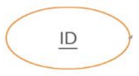
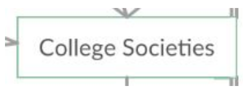
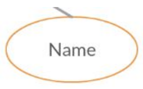
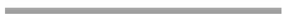
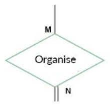
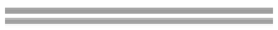
1. *Students_Enrolled*
2. *College_Societies*
3. *Society_Members*
4. *Activities*
5. *Locations*
6. *Activity_Organisers*
7. *Activities_SignUp*

With regards to the Students_Enrolled table, in a real-life implementation of this system, this table would likely be connected to in an external read-only database managed by Academic Registry. However, for the purposes of this project, it will be managed by the system administrator i.e. the CSC.

Entity relationship diagram



Key:

	Primary key i.e. oval with attribute underlined.		Entity
	Attribute		Partial participation
	Relationship and its cardinality		Full participation

Relational schema mapping

In this relational schema mapping, I mapped the conceptual database designed in the previous section (ERM) to a logical database design. The mapping creates relations with simple, single-valued attributes, and constraints using primary keys, unique keys and referential integrity constraints.

Assumptions made:

1. College Societies, Students, Locations and Activities all have unique IDs.
2. A student who is enrolled may participate in an activity without being a member (usually this means an additional fee in real life or reduced access to tickets).
3. College societies have elected a chair from the current body of students enrolled.
4. Members of a college society must be currently enrolled as a student.
5. Not all enrolled students must be a member of a college society.
6. An activity may be organised by more than one college society i.e. a collaboration.
7. If a college society were to be unratified by the Central Societies Committee (CSC) and removed or simply pull out of organising an activity, that activity is not automatically cancelled as it may be a collaboration or the CSC may wish to operate it where for e.g. fees for foreign trips have been paid already.
8. An activity must take place in a single location, sub-activities at different locations which form part of a global activity must be listed as separate activities.
9. It is assumed that Societies/CSC manage the activity signup and membership, and this is not directly accessed by students as e.g. payment might need to be taken before signup.

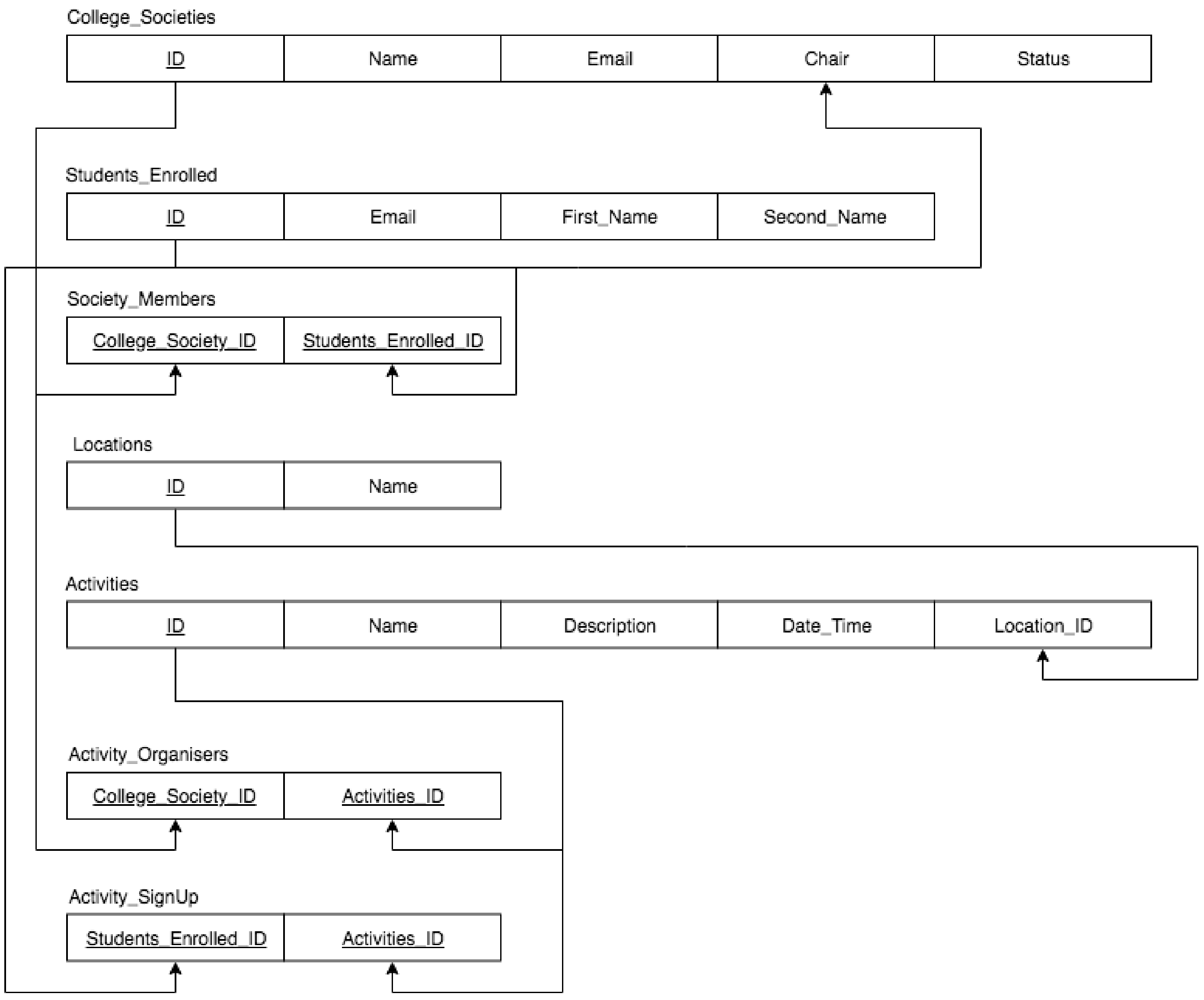
Mapping to Relational Tables:

```
Students_Enrolled (ID, Email, First_Name, Second_Name)

College_Societies (ID, Name, Email, Chair, Status)
                                    -------
Locations (ID, Name)

Activities (ID, Name, Description, Date_Time, Location_ID)
                                              -----------------
Society_Members (College_Society_ID, Students_Enrolled_ID)
                 ------------------------------------------------------------
Activity_Organisers (College_Societies_ID, Activities_ID)
                     -------------------------------------------------
Activities_SignUp (Students_Enrolled_ID, Activities_ID)
                   --------------------------------------------------
```

Relational Schema Diagram:

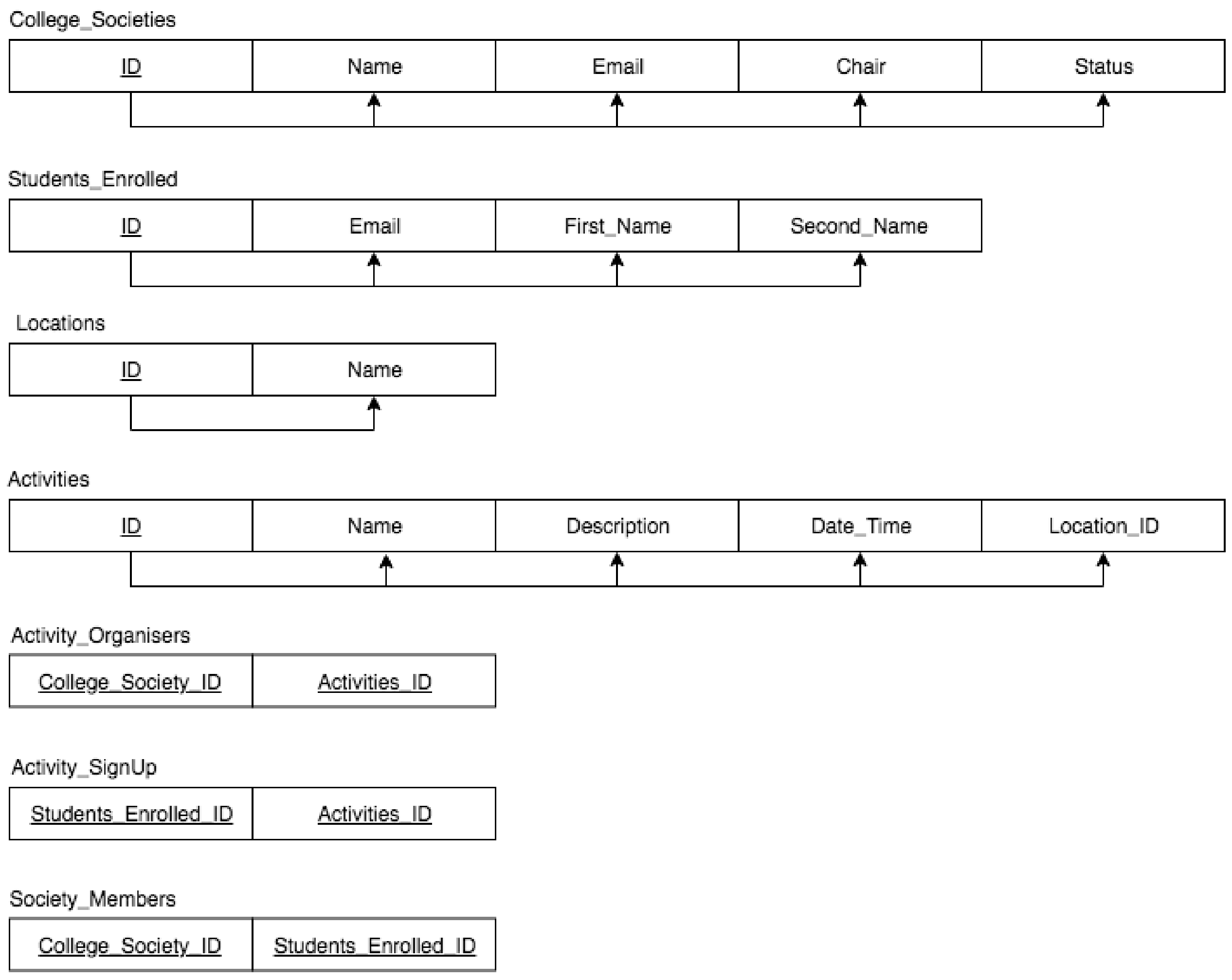


Functional dependency and normalisation

The values of the attribute set X from a tuple in r, uniquely (or functionally) determine the values of the attribute set Y [CS3041 Notes, 2018].

Using this definition of a functional dependency, I created a functional dependency diagram which I used as a tool to measure the appropriateness of groupings of attributes into relations. In turn, I then used it to help ensure the database meets each stage of normalisation up to Boyce-Codd Normal Form, the aim for this project, although higher stages of normalisation do exist.

Functional Dependency Diagram:



Integrity constraints and security policy

Integrity and security are two closely related concepts in database design; while the former is concerned with accidental corruption, the latter is concerned with deliberate corruption. In particular, ensuring the security of the database is a difficult but crucial task for legal (GDPR etc.) and ethical reasons (Personal data etc.).

Integrity constraints:

To help protect against accidental corruption I used some of the following constraints to stop the user for completing an action that would potentially cause such accidental corruption.

Entity integrity constraints:
- Every primary, and hence foreign keys, were specified as NOT NULL.

Referential integrity Constraints:
- Referential integrity was maintained by making all foreign keys NOT NULL, or where this was not desirable such as in the case of a society chair in which the role may be vacated and thus left as NULL until another student is elected to that position; I added BEFORE DELETE Triggers to handle any potential referential integrity issues.

Security policy:

While in particular security policies are used to protect against deliberate corruption or misuse of data, they can also be used to protect against accidental corruption by well-meaning users with too much access.

In the implementation of a security policy for this database, I focused on roles and user authentication. While the Central Societies Committee (CSC), the governing body of all the societies within Trinity are granted full permissions to all tables, individual societies get reduced access to modify tables associated with their activities etc. and individual society members get simple view access to society activities.

It is also worth noting as mentioned in the prelude to this report, the table associated with Students_Enrolled in real life would likely - for security and administration reasons - be stored in an external read-only database held by Academic Registry which the database outlined here would connect to. However, due to the scope of this project, this table is stored directly in the societies database and the database administrator, the CSC has the rights to edit this table in this implementation.

I implemented the following roles:

1. **CSC Admin:** As the CSC is the governing body of all student societies within the college, and hence the database administrator, they should have the highest level of access. The CSC has the power to create and assign users to these roles, as well as create additional roles should a need arise.
2. **Society executives:** Society executives are elected by the student body to run particular societies, they are tasked with the administration of society membership records and running activities hence they read and write access to certain elements of the database but much more restricted than the CSC.
3. **Students:** Students have the lowest level of access to the database, with simple read-only access to a number of Views.

Role creation

```
CREATE ROLE csc_admin, csc_society, student;

GRANT ALL ON TCD_SOCS.* TO csc_admin;
GRANT CREATE, DELETE, UPDATE, SELECT ON Society_Members TO csc_society;
GRANT CREATE, DELETE, UPDATE, SELECT ON Activities TO csc_society;
GRANT SELECT ON activities_availible TO students;
```

User creation and role allocation

```
CREATE USER tedx_soc@localhost IDENTIFIED BY 'Secure$6275';
GRANT csc_society TO tedx_soc@localhost;
```

Example operations

View creation:

In general, Views are used for two reasons; either to allow users with restricted access to view selected tables or to form a table based on data that is commonly accessed together. An example of a view I created for the project was Society Activities, which combines together data from the College_Societies, Society_Activities, Activity and Location tables to present a meaningful representation of upcoming activities to students who have restricted view access.

Creating View

```
CREATE VIEW activities_availible AS SELECT socs.Name "Organiser", acts.Name
"Activity", acts.Description, acts.Date_Time "When", locs.Name "Location"
FROM Activities AS acts, Activity_Organisers AS orgs, College_Societies AS socs,
Locations AS locs
WHERE orgs.Activities_ID = acts.ID and orgs.College_Societies_ID = socs.ID and
locs.ID = acts.Location_ID;
```

Resulting View

```
+-----------+--------------------+---------------------------------------------------------------------------+---------------------+--------------+
| Organiser | Activity           | Description                                                               | When                | Location     |
+-----------+--------------------+---------------------------------------------------------------------------+---------------------+--------------+
| TEDx      | TEDx Conference    | See exciting speakers                                                     | 2019-01-01 18:00:00 | Regent House |
| TES       | Incubators         | Get investement for your startup                                          | 2019-01-01 17:00:00 | Regent House |
| MOVE      | MOVE Ball          | Get your tux on and support charity                                       | 2019-05-01 14:00:00 | Dining Hall  |
| TEDx      | TEDx Salon         | See TED speakers Mark Pollock and Simone George                           | 2019-09-01 17:00:00 | Dining Hall  |
| TAF       | Design workshop    | Learn the latest in modern arts installations                             | 2019-03-02 12:00:00 | LB01         |
| VDP       | VDP panto          | Support charity and have a laugh at VDP christmas panto                   | 2019-06-01 13:00:00 | LBO4         |
| TEF       | Economic Forum     | Get the lowdown on where the Irish economy is headed from leading experts | 2019-03-06 16:30:00 | TBSI         |
| TEDx      | MOVE Carol Singing | Support charity and get in the christmas spirit                           | 2019-01-06 18:30:00 | Front Sq     |
| TES       | MOVE Carol Singing | Support charity and get in the christmas spirit                           | 2019-01-06 18:30:00 | Front Sq     |
+-----------+--------------------+---------------------------------------------------------------------------+---------------------+--------------+
```

Relational select and table joins across multiple tables:

In this query, I use an Inner Join across multiple tables combined with the Select command to get a table resulting in a list of all enrolled students who are society chairs along with their email addresses. This could be useful for the CSC if when they need to contact a society chair or just get an overview.

Creating Select

```
SELECT socs.Name, students.First_Name, students.Second_Name, students.Email
FROM College_Societies AS socs JOIN Students_Enrolled as students
  ON socs.Chair = students.ID;
```

The resulting table of Society Chairs

```
+------+------------+-------------+-------------+
| Name | First_Name | Second_Name | Email       |
+------+------------+-------------+-------------+
| TEDx | Lee        | Campbell    | name1@tcd.ie |
| TES  | Joe        | Deegan      | name2@tcd.ie |
| MOVE | Daire      | Pryal       | name3@tcd.ie |
| TAF  | Emily      | Feeney      | name4@tcd.ie |
| VDP  | Mellisa    | Barnes      | name5@tcd.ie |
| TEF  | Jack       | Dolan       | name6@tcd.ie |
+------+------------+-------------+-------------+
```

Update operations:

Update operations can be used for a range of actions, in this example, I illustrate how a society chair might be updated to reflect the results of an AGM election.

Creating Update command

```
UPDATE College_Societies SET Chair = 16316007 WHERE ID = 6;
```

A view of the original table, followed by the resulting table after the update

```
+------+------------+-------------+-------------+
| Name | First_Name | Second_Name | Email       |
+------+------------+-------------+-------------+
| TEDx | Lee        | Campbell    | name1@tcd.ie |
| TES  | Joe        | Deegan      | name2@tcd.ie |
| MOVE | Daire      | Pryal       | name3@tcd.ie |
| TAF  | Emily      | Feeney      | name4@tcd.ie |
| VDP  | Mellisa    | Barnes      | name5@tcd.ie |
| TEF  | Jack       | Dolan       | name6@tcd.ie |
+------+------------+-------------+-------------+

Query OK, 1 row affected (0.20 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
+------+------------+-------------+-------------+
| Name | First_Name | Second_Name | Email       |
+------+------------+-------------+-------------+
| TEDx | Lee        | Campbell    | name1@tcd.ie |
| TES  | Joe        | Deegan      | name2@tcd.ie |
| MOVE | Daire      | Pryal       | name3@tcd.ie |
| TAF  | Emily      | Feeney      | name4@tcd.ie |
| VDP  | Mellisa    | Barnes      | name5@tcd.ie |
| TEF  | Philip     | Smyth       | name7@tcd.ie |
+------+------------+-------------+-------------+
```

Trigger commands:

Trigger commands can be used for a wide variety of reasons, in the case of this project I used them for the most part to ensure the integrity of the database on deletion of a tuple. For example, if a student dropped out and in turn was removed from the Students_Enrolled table, a trigger I created would any rows where the student's ID was referenced as a foreign key.

Creating of above Trigger

```
DELIMITER $$
USE TCD_SOCS $$
CREATE DEFINER = CURRENT_USER TRIGGER Student_Before_Delete BEFORE DELETE
ON Students_Enrolled FOR EACH ROW
BEGIN

DELETE FROM Society_Members WHERE Society_Members.Students_Enrolled_ID =
OLD.ID;
DELETE FROM Activities_SignUp WHERE Activities_SignUp.Activities.ID =
OLD.ID;
UPDATE College_Societies SET Chair = NULL WHERE Chair = OLD.ID;

END$$
DELIMITER ;
```

Appendix i

```sql
---------------------------------------------------------------------------
------------
CREATE ENTITY TABLES
---------------------------------------------------------------------------
------------
  CREATE TABLE Students_Enrolled (
  ID INT NOT NULL AUTO_INCREMENT,
  Email VARCHAR(255) NOT NULL,
  First_Name VARCHAR(255) NOT NULL,
  Second_Name VARCHAR(255) NOT NULL,
  PRIMARY KEY (ID));

CREATE TABLE College_Societies (
  ID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  Email VARCHAR(255) NOT NULL,
  Chair INT NULL,
  Status VARCHAR(45) NOT NULL,
  PRIMARY KEY (ID),
  FOREIGN KEY (Chair) REFERENCES Students_Enrolled (ID)

  );

  CREATE TABLE Locations (
  ID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  PRIMARY KEY (ID));

  CREATE TABLE Society_Members (
  College_Society_ID INT NOT NULL ,
  Students_Enrolled_ID INT NOT NULL,
  PRIMARY KEY (College_Society_ID, Students_Enrolled_ID),

  FOREIGN KEY (College_Society_ID) REFERENCES College_Societies (ID),

  FOREIGN KEY (Students_Enrolled_ID) REFERENCES Students_Enrolled (ID));
```

```sql
CREATE TABLE Activities (
  ID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  Description VARCHAR(255) NOT NULL,
  Date_Time DATETIME NOT NULL,
  Location_ID INT NULL,
  PRIMARY KEY (ID),
  FOREIGN KEY (Location_ID) REFERENCES Locations (ID));


CREATE TABLE Activity_Organisers (
  College_Societies_ID INT NOT NULL,
  Activities_ID INT NOT NULL,
  PRIMARY KEY (College_Societies_ID, Activities_ID),

  FOREIGN KEY (College_Societies_ID) REFERENCES College_Societies (ID),

  FOREIGN KEY (Activities_ID) REFERENCES Activities (ID));


  CREATE TABLE Activities_SignUp (
  Students_Enrolled_ID INT NOT NULL,
  Activities_ID INT NOT NULL,
  PRIMARY KEY (Students_Enrolled_ID, Activities_ID),

  FOREIGN KEY (Students_Enrolled_ID) REFERENCES Students_Enrolled (ID),

  FOREIGN KEY (Activities_ID) REFERENCES Activities (ID));


---------------------------------------------------------------------------
------------
CREATE TRIGGERS
---------------------------------------------------------------------------
------------


DELIMITER $$
USE TCD_SOCS $$
CREATE DEFINER = CURRENT_USER TRIGGER Socs_Before_Delete BEFORE DELETE ON
College_Societies FOR EACH ROW
```

```
BEGIN


DELETE FROM Society_Members WHERE Society_Members.College_Society_ID =
OLD.ID;
DELETE FROM Activity_Organisers WHERE
Activity_Organisers.College_Societies_ID = OLD.ID;


END$$
DELIMITER ;



DELIMITER $$
USE TCD_SOCS $$
CREATE DEFINER = CURRENT_USER TRIGGER Activity_Before_Delete BEFORE DELETE
ON Activities FOR EACH ROW
BEGIN

DELETE FROM Activity_Organisers WHERE Activity_Organisers.Activities.ID =
OLD.ID;
DELETE FROM Activities_SignUp WHERE Activities_SignUp.Activities.ID =
OLD.ID;




END$$
DELIMITER ;


DELIMITER $$
USE TCD_SOCS $$
CREATE DEFINER = CURRENT_USER TRIGGER Student_Before_Delete BEFORE DELETE
ON Students_Enrolled FOR EACH ROW
BEGIN

DELETE FROM Society_Members WHERE Society_Members.Students_Enrolled_ID =
OLD.ID;
DELETE FROM Activities_SignUp WHERE Activities_SignUp.Activities.ID =
OLD.ID;
```

```
UPDATE College_Societies SET Chair = NULL WHERE Chair = OLD.ID;



END$$
DELIMITER ;

DELIMITER $$
USE TCD_SOCS $$
CREATE DEFINER = CURRENT_USER TRIGGER Location_Before_Delete BEFORE DELETE
ON Locations FOR EACH ROW
BEGIN

UPDATE Activities SET Location_ID = NULL WHERE Location_ID = OLD.ID;



END$$
DELIMITER ;

---------------------------------------------------------------------------
------------
CREATE VIEWS
---------------------------------------------------------------------------
------------

CREATE VIEW society_chairs AS SELECT socs.Name, students.First_Name,
students.Second_Name, students.Email
FROM College_Societies AS socs JOIN Students_Enrolled as students
  ON socs.Chair = students.ID;

CREATE VIEW all_societies AS SELECT socs.Name, socs.Email "Society-Email",
socs.Status, students.First_Name "Chair_FN", students.Second_Name
"Chair_SN"
FROM College_Societies AS socs LEFT JOIN Students_Enrolled as students
  ON socs.Chair = students.ID;

CREATE VIEW elections_needed AS
SELECT * FROM College_Societies WHERE College_Societies.Chair is Null;

CREATE VIEW activities_availible AS SELECT socs.Name "Organiser", acts.Name
```

```
"Activity", acts.Description, acts.Date_Time "When", locs.Name "Location"
FROM Activities AS acts, Activity_Organisers AS orgs, College_Societies AS
socs, Locations AS locs
WHERE orgs.Activities_ID = acts.ID and orgs.College_Societies_ID = socs.ID
and locs.ID = acts.Location_ID;

//All society Members
SELECT socs.Name, students.First_Name, students.Second_Name, students.Email
FROM College_Societies AS socs, Society_Members AS members,
Students_Enrolled as students
WHERE socs.ID = members.College_Society_ID and students.ID =
members.Students_Enrolled_ID;

//TEDx Members
SELECT socs.Name, students.First_Name, students.Second_Name, students.Email
FROM College_Societies AS socs, Society_Members AS members,
Students_Enrolled as students
WHERE socs.ID = members.College_Society_ID and students.ID =
members.Students_Enrolled_ID and socs.ID = 1;

---------------------------------------------------------------------------
------------
CREATE ROLES
---------------------------------------------------------------------------
------------

CREATE ROLE csc_admin, csc_society, student;
GRANT ALL ON TCD_SOCS.* TO csc_admin;
GRANT CREATE, DELETE, UPDATE, SELECT ON Society_Members TO csc_society;
GRANT CREATE, DELETE, UPDATE, SELECT ON Activities TO csc_society;
GRANT SELECT ON activities_availible TO students;
```

Appendix ii

```
---------------------------------------------------------------------------
------------
INSERT DATA
---------------------------------------------------------------------------
------------

--------------------
STUDENTS ENROLLED
--------------------

INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316001, 'name1@tcd.ie', 'Lee', 'Campbell');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316002, 'name2@tcd.ie', 'Joe', 'Deegan');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316003, 'name3@tcd.ie', 'Daire', 'Pryal');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316004, 'name4@tcd.ie', 'Emily', 'Feeney ');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316005, 'name5@tcd.ie', 'Mellisa', 'Barnes');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316006, 'name6@tcd.ie', 'Jack ', 'Dolan');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316007, 'name7@tcd.ie', 'Philip', 'Smyth');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316008, 'name8@tcd.ie', 'Matthew', 'Taylor');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316009, 'name9@tcd.ie', 'Steven', 'Grant');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316010, 'name10@tcd.ie', 'Ian', 'Fennel');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316011, 'name11@tcd.ie', 'Matthew', 'Kavanagh');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316012, 'name12@tcd.ie', 'Sarah', 'Hennesy');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316013, 'name13@tcd.ie', 'Emma', 'Quinn');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
```

```
(16316014, 'name14@tcd.ie', 'Sarah', 'Thompson');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316015, 'name15@tcd.ie', 'Clara', 'Hogan');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316016, 'name16@tcd.ie', 'Rory', 'Booth');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316017, 'name17@tcd.ie', 'Sophie', 'Campbell');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316018, 'name18@tcd.ie', 'Mollie', 'Campbell');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316019, 'name19@tcd.ie', 'Paddy', 'Campbell');
INSERT INTO Students_Enrolled (ID, Email, First_Name, Second_Name) VALUES
(16316020, 'name20@tcd.ie', 'Michelle', 'Campbell');

--------------------
COLLEGE SOCIETIES
--------------------

INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (1,
'TEDx', 'tedxtcd@csc.tcd.ie', 16316001, 'Confirmed');
INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (2,
'TES', 'tes@csc.tcd.ie', 16316002, 'Confirmed');
INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (3,
'MOVE', 'move@csc.tcd.ie', 16316003, 'Confirmed');
INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (4,
'TAF', 'taf@csc.tcd.ie', 16316004, 'Confirmed');
INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (5,
'VDP', 'vdp@csc.tcd.ie', 16316005, 'Confirmed');
INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (6,
'TEF', 'tef@csc.tcd.ie', 16316006, 'Confirmed');
INSERT INTO College_Societies (ID, Name, Email, Chair, Status) VALUES (7,
'VIS', 'vis@csc.tcd.ie', NULL, 'Confirmed');

--------------------
LOCATIONS
--------------------

INSERT INTO Locations (ID, Name) VALUES (1, 'Regent House');
INSERT INTO Locations (ID, Name) VALUES (2, 'Dining Hall');
INSERT INTO Locations (ID, Name) VALUES (3, 'LB01');
INSERT INTO Locations (ID, Name) VALUES (4, 'LBO4');
INSERT INTO Locations (ID, Name) VALUES (5, 'TBSI');
INSERT INTO Locations (ID, Name) VALUES (6, 'Front Sq');

--------------------
ACTIVITIES
--------------------

INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (101, 'TEDx Conference', 'See exciting speakers ', '2019-01-01
18:00:00', 1);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (102, 'Incubators', 'Get investement for your startup','2019-01-01
17:00:00', 1);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (103, 'MOVE Ball', 'Get your tux on and support charity',
'2019-05-01 14:00:00', 2);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (104, 'Design workshop', 'Learn the latest in modern arts
installations', '2019-03-02 12:00:00', 3);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (105, 'VDP panto', 'Support charity and have a laugh at VDP
christmas panto', '2019-06-01 13:00:00', 4);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (106, 'Economic Forum', 'Get the lowdown on where the Irish economy
is headed from leading experts', '2019-03-06 16:30:00', 5);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (107, 'TEDx Salon', 'See TED speakers Mark Pollock and Simone
George', '2019-09-01 17:00:00', 2);
INSERT INTO Activities (ID, Name, Description, Date_Time, Location_ID)
VALUES (108, 'MOVE Carol Singing', 'Support charity and get in the
christmas spirit', '2019-01-06 18:30:00', 6);

--------------------
SOCIETY MEMBERS
--------------------

INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (1, 16316007);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (2, 16316008);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
```

```
VALUES (3, 16316009);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (4, 16316010);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (5, 16316011);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (6, 16316012);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (6, 16316007);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (5, 16316008);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (4, 16316009);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (3, 16316010);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (2, 16316011);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (1, 16316012);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (4, 16316013);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (2, 16316007);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (3, 16316008);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (5, 16316010);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (6, 16316011);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (2, 16316012);
INSERT INTO Society_Members (College_Society_ID, Students_Enrolled_ID)
VALUES (3, 16316013);

--------------------
ACTIVITY ORGANISERS
--------------------

INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (1, 101);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
```

```
VALUES (2, 102);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (3, 103);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (4, 104);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (5, 105);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (6, 106);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (1, 107);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (1, 108);
INSERT INTO Activity_Organisers (College_Societies_ID, Activities_ID)
VALUES (2, 108);

--------------------
ACTIVITIES SIGNUP
--------------------

INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316007, 101);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316008, 102);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316009, 103);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316010, 104);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316011, 105);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316012, 106);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316007, 107);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316008, 108);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316009, 101);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316010, 102);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
```

```sql
(16316011, 103);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316012, 104);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316013, 105);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316007, 106);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316008, 107);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316009, 108);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316010, 106);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316011, 107);
INSERT INTO Activities_SignUp (Students_Enrolled_ID, Activities_ID) VALUES
(16316012, 108);
```