Student Club Activities Management System Database

Can ZHOU
19324118
zhouc@tcd.ie

Contents

** The complete copiable codes are in Appendix_0.
** If copying the code **before** Appendix_0 directly, the line number may also be copied down.
** I have also attached the SQL files in the same folder with report.

Background Information & Assumption

I chose to implement a database used for student club activities management system. The management system can let students sign up the activities hosted by clubs, let clubs release the activities, order the venues and facilities they needed in activities and let university staff to supervise activities. This database is designed to meet those functions. I also added some constrains and triggers to restrain the database's relationships. In order to avoid any malicious corruption of data, the privileges are also be granted for each different role.

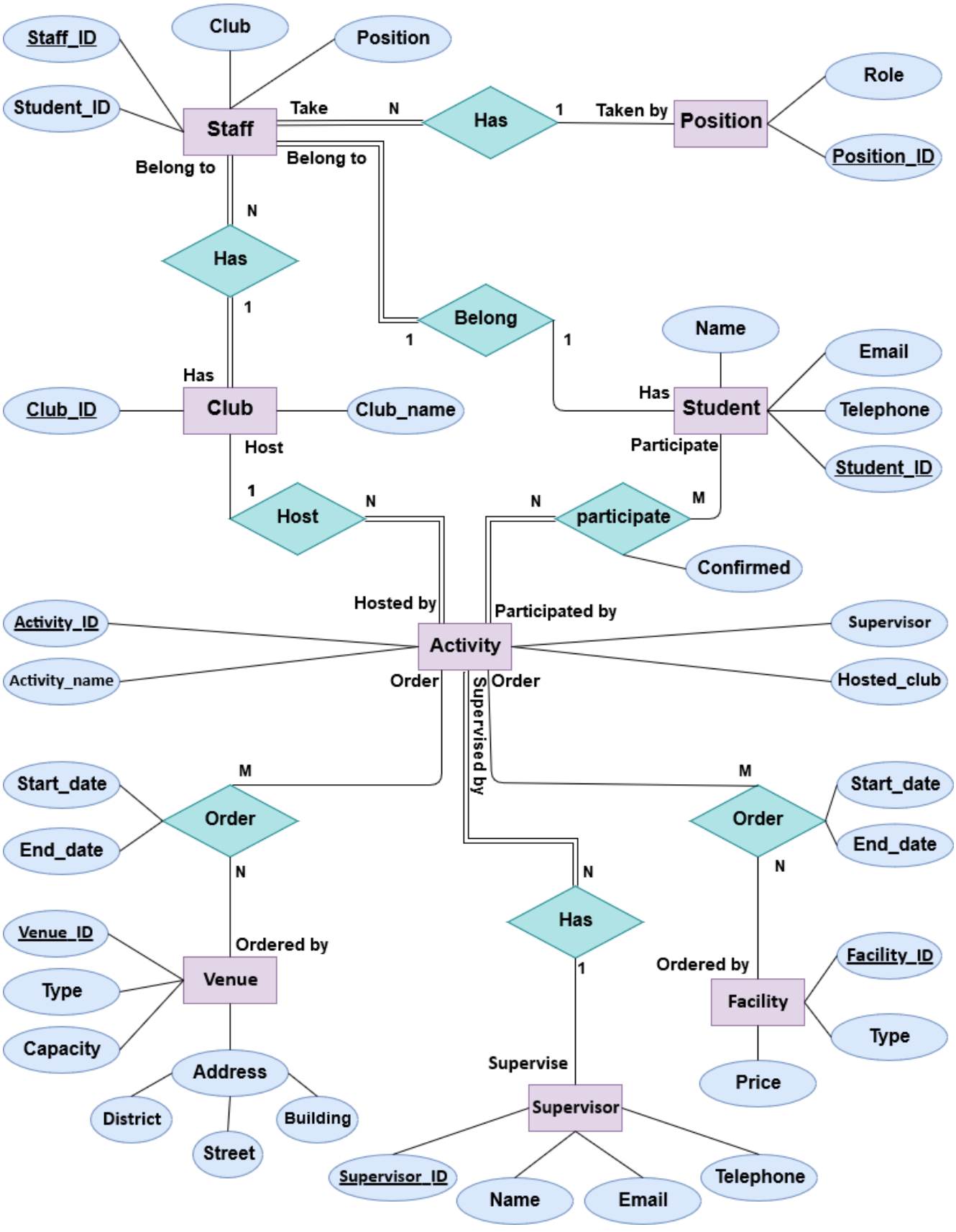
Here are my assumptions:

- There are eight entities in the database: Activity, Club, Staff (club), Position (club), Student, Venue, Facility, Supervisor.
- For Activity:
  - Entity Activity has four attributes: Activity_ID, Activity_name, Supervisor and Hosted_club.
  - Activity can be hosted by each club.
  - Each activity can let many students participated in.
  - Each activity needs a supervisor from university to supervise activity process.
- For Club:
  - Entity Club has two attributes: Club_ID and Club_name.
  - Each club must have staff.
- For Staff:
  - Entity Staff has four attributes: Staff_ID, Student_ID, Club and Position.
  - Club staff are from student.
  - Each staff is only belonged one club.
  - Each staff has only one position (role).
- For Position:
  - Entity Position has two attributes: Position_ID and Role.
  - There are five types of roles: President, Activity Officer, Publicity Officer, Accountant and Member.
- For Student:
  - Entity Student has four attributes: Student_ID, Name, Email and Telephone.
  - Student can sign up many activities.
  - Students' sign up need to be confirmed by club staff.
- For Venue:
  - Entity Venue has seven attributes: Venue_ID, Venue_name, Capacity, District, Street, Building and Available.
  - Venues can be booked by activities and each activity may order many venues.
  - Different venue has different capacity and address.
- For Facility:
  - Entity Facility has four attributes: Facility_ID, Facility_name, Price, Available.

  - Facilities can be booked by activities and each activity may order many facilities.
  - Each facility has its own price to be borrowed.
- For Supervisor:
  - Entity Supervisor has four attributes: Supervisor_ID, Name, Email and Telephone.
  - Each supervisor can supervise many activities.
- There are four roles in this database: student who participates the activities; supervisor who monitors the activities, club president, the president of club which can organize the activities and club staff who can help their club to host activities.
- Privileges
  - ALL the roles can view the Activity and Club.
  - Supervisors can modify, delete activities they supervised.
  - Students can sign up activities.
  - President and supervisor can view the booking situation of facilities and venues.
  - Only president can add, modify, delete the activities and book venues and facilities for activates.
  - All the club staff can confirm the activities which students signed up and belong to their club is valid.
- I assume the database name is TCD.

Entity Relationship Diagram

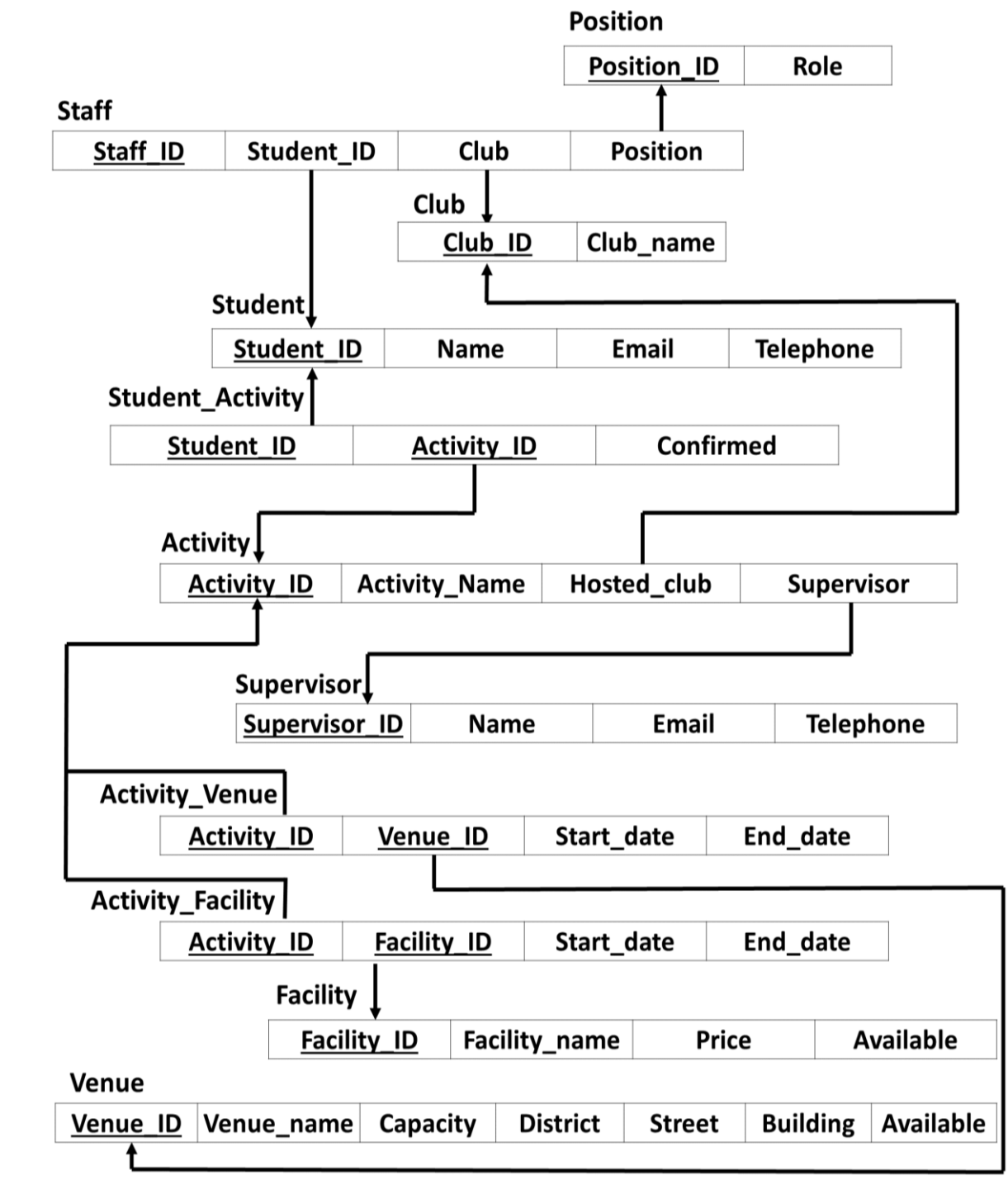
The entity relationship diagram for the database is shown below:



Mapping to Relational Schema

The mapping from the entity relationship diagram to an outline relational schema is shown below:

***Underlined Attributes indicate Primary Keys.**

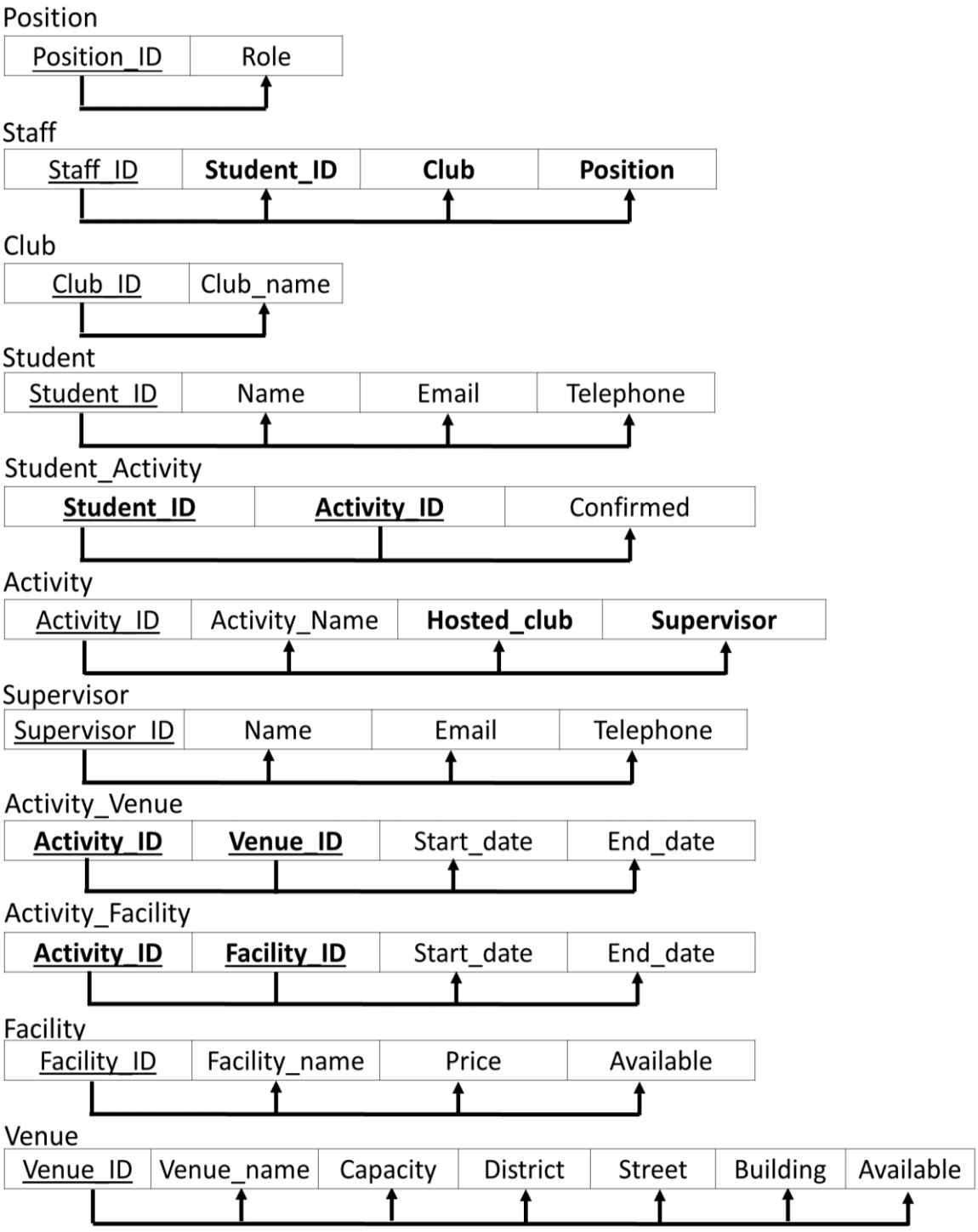


Functional Dependency Diagram

The functional dependency diagram of the database is shown below:

***Underlined Attributes indicate Primary Keys.**

***Emboldened Attributes indicate Foreign Keys.**



Normalisation [Definitions are from lecture slides]

- First Normal Form
  - There is only one value at each intersection of the row-column, not a list of values.
  - The initial relational schema above did not violate First Normal Form.
- Second Normal Form
  - Each non-key column is fully functionally dependent on the entire primary key.
  - The initial relational schema above did not violate Second Normal Form.
- Third Normal Form
  - No non-key attributes is transitively dependent upon the primary key.
  - The initial relational schema above did not violate Third Normal Form.
- Boyce-Codd Normal Form
  - Whenever a functional dependency X→Y holds in the relation R, X is a superkey of R.
  - The initial relational schema above did not violate Boyce-Codd Normal Form.
- The initial relational schema is reached Boyce-Codd Normal Form.

Notice

** The complete copiable codes are in Appendix_0.

** If copying the code **before** Appendix_0 directly, the line number may also be copied down.

Implicit Constraints

- **PRIMARY KEY:**
  - I set primary keys when create table.
    Here are the codes for primary key in each CREATE TABLE:

```
1.  Position:           PRIMARY KEY (Position_ID)
2.  Club:               PRIMARY KEY (Club_ID)
3.  Student:            PRIMARY KEY (Student_ID)
4.  Staff:              PRIMARY KEY (Staff_ID)
5.  Supervisor:         PRIMARY KEY (Supervisor_ID)
6.  Activity:           PRIMARY KEY (Activity_ID)
7.  Student_Activity:   PRIMARY KEY (Student_ID, Activity_ID)
8.  Facility:           PRIMARY KEY (Facility_ID)
9.  Venue:              PRIMARY KEY (Venue_ID)
10. Activity_Facility:  PRIMARY KEY (Activity_ID, Facility_ID)
11. Activity_Venue:     PRIMARY KEY (Activity_ID, Venue_ID)
```

  - Primary key can also be set following the attribute specification.

- **FOREIGN KEY:**
  - I set foreign keys when create table.
    Here are the codes for foreign keys in CREATE TABLE:

```
1.  Staff:              FOREIGN KEY (Student_ID)  REFERENCES Student (Student_ID)
2.                      FOREIGN KEY (Club)        REFERENCES Club (Club_ID)
3.                      FOREIGN KEY (Position)    REFERENCES Position (Position_ID)
4.  Activity:           FOREIGN KEY (Hosted_club) REFERENCES Club (Club_ID)
5.                      FOREIGN KEY (Supervisor)  REFERENCES Supervisor (Supervisor_ID)
6.  Student_Activity:   FOREIGN KEY (Student_ID)  REFERENCES Student (Student_ID)
7.                      FOREIGN KEY (Activity_ID) REFERENCES Activity (Activity_ID)
8.  Activity_Facility:  FOREIGN KEY (Activity_ID) REFERENCES Activity (Activity_ID)
9.                      FOREIGN KEY (Facility_ID) REFERENCES Facility (Facility_ID)
10. Activity_Venue:     FOREIGN KEY (Activity_ID) REFERENCES Activity (Activity_ID)
11.                     FOREIGN KEY (Venue_ID)    REFERENCES Venue (Venue_ID)
```

  - Foreign keys can also be set by ALTER
  - Example:

```
ALTER TABLE STAFF ADD FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID);
```

Semantic Constraints

- **Entity Integrity Constraints**
  - **NULL Keys**
    I have set all primary key attributes with NOT NULL when CREATE TABLE.

- **Referential Integrity Constraints**
  - **NULL Keys**
    I have set all foreign key attributes with NOT NULL when CREATE TABLE.

- **Table Constraints**
  - **UNIQUE**
    I have not set unique in any table. Because when I set primary key, unique will be automatically added to primary key.
    And ALTER can be used to add unique as well.
    Example:

```
ALTER TABLE Staff ADD UNIQUE (Student_ID);
```

- **NOT NULL**

  I have set all attributes with NOT NULL when CREATE TABLE

  Example:

```
Student_ID INT NOT NULL
```

- **CHECK**

  I have set some CHECK constraints when CREATE TABLE:

  - For Position table, the Role attribute should in five types: 'President', 'Activity Officer', 'Publicity Officer', 'Accountant', 'Member'.
  - For Student_Activity table, the Confirmed attribute should in 0 or 1.
  - For Venue table, the Available attribute should in 0 or 1.
  - For Facility table, the Available attribute should in 0 or 1.
  - Code:

```
1.  Position:         CONSTRAINT Role_check CHECK(
2.                              Role IN ('President', 'Activity Officer', 'Publicity O
    fficer', 'Accountant', 'Member')
3.                    )
4.  Student_Activity: CONSTRAINT Confirmed_check CHECK(
5.                              Confirmed IN (0,1)
6.                    )
7.  Facility:         CONSTRAINT Facility_available_check CHECK(
8.                              Available IN (0,1)
9.                    )
10. Venue:            CONSTRAINT Venue_available_check CHECK(
11.                             Available IN (0,1)
12.                   )
```

  Check can also be added by ALTER:

  - Example:

```
ALTER TABLE Venue ADD CONSTRAINT Venue_available_check CHECK( Available IN (0,1) )
```

- **DEFAULT**

  I have set some DEFAULT constraints when CREATE TABLE:

  - For Venue table, the value of Available attribute is defaulted to 1.
  - For Facility table, the value of Available attribute is defaulted to 1.
  - Code:

```
1.  Facility: Available INT DEFAULT 1
2.  Venue:    Available INT DEFAULT 1
```

- **TRIGGER**
  - A database trigger is a special stored procedure running while specific actions are taking place within a database. Triggers are set to run when a table's data is modified. Triggers can be defined for running actions such as INSERT, UPDATE, and DELETE before or after DML (Data Manipulation Language).
  - I have set three triggers.
  - Delimiter label is used for executing the trigger in MySQL.
  - facility_price_tigger:
    Before inserting new tuple into table Facility, if price inserted is smaller than 0, then change the insert price to 0.
    Code:

```
1. delimiter //
2. CREATE TRIGGER facility_price_tigger BEFORE INSERT ON Facility
3. FOR EACH ROW
4. BEGIN
5.     IF NEW.Price < 0
6.     THEN SET NEW.Price = 0;
7.     END IF;
8. END; //
```

  - student_staff_tigger:
    Before deleting the tuple into table Student, delete the tuple in table Staff who has the same Student_ID
    Code:

```
1. delimiter //
2. CREATE TRIGGER student_staff_tigger BEFORE DELETE ON Student
3. FOR EACH ROW
4. BEGIN
5.     DECLARE var_id INT DEFAULT 0;
6.     SET var_id = OLD.Student_ID;
7.     DELETE FROM Staff
8.     WHERE Student_ID = var_id;
9. END;//
```

  - student_activity_tigger:
    Before deleting the tuple into table Student, delete the tuples in table Student_Activity which have the same Student_ID
    Code:

```
delimiter //
CREATE TRIGGER student_activity_tigger BEFORE DELETE ON Student
FOR EACH ROW
BEGIN
    DECLARE var_id INT DEFAULT 0;
    SET var_id = OLD.Student_ID;
    DELETE FROM Student_Activity
    WHERE Student_ID = var_id;
END;//
```

Database Security Commands for Access and Security Policy

Database should consider security. In order to avoid any malicious corruption of data, I have set many roles and grant the privileges for each different role.

As in the assumption, there are four roles in this database: student who participates the activities; supervisor who monitors the activities, club president, the president of club which can organize the activities and club staff who can help their club to host activities.

- CREATE ROLE:

```
CREATE ROLE 'student', 'supervisor', 'club_president', 'club_staff';
```

- Grant privileges to roles:
  - ALL the roles can view the Activity and Club.
  - Supervisors can modify, delete activities they supervised.
  - Students can sign up activities.
  - President and supervisor can view the booking situation of facilities and venues.
  - Only president can add, modify, delete the activities and book venues and facilities for activates.
  - All the club staff can confirm the activities which students signed up and belong to their club is valid.
  - Assuming database name is TCD.
  - Code:

```
GRANT SELECT ON TCD.Activity TO 'student', 'supervisor', 'club_president', 'club_staff';
GRANT SELECT ON TCD.Club TO 'student', 'supervisor', 'club_president', 'club_staff';
GRANT UPDATE, DELETE ON TCD.Activity TO 'supervisor';
GRANT SELECT, INSERT ON TCD.Student_Activity TO 'student';
GRANT SELECT ON TCD.Facility TO 'club_president', 'supervisor';
```

```
GRANT SELECT ON TCD.Venue TO 'club_president', 'supervisor';
GRANT SELECT, INSERT, UPDATE, DELETE ON TCD.Activity TO 'club_president';
GRANT SELECT, INSERT, UPDATE, DELETE ON TCD.Activity_Facility TO 'club_president';
GRANT SELECT, INSERT, UPDATE, DELETE ON TCD.Activity_Venue TO 'club_president';
GRANT SELECT, UPDATE ON TCD.Student_Activity TO 'club_president', 'club_staff';
```

- CREATE USER:
  I have created some new users as well, so we can grand users with specific role privileges.
  Code:

```
CREATE USER 'student_0' IDENTIFIED BY '123456';
CREATE USER 'student_1' IDENTIFIED BY '123456';
CREATE USER 'student_2' IDENTIFIED BY '123456';
CREATE USER 'student_3' IDENTIFIED BY '123456';
CREATE USER 'student_4' IDENTIFIED BY '123456';
CREATE USER 'student_5' IDENTIFIED BY '123456';
CREATE USER 'supervisor_0' IDENTIFIED BY '123456';
CREATE USER 'supervisor_1' IDENTIFIED BY '123456';
```

- Grant privileges to users:

```
GRANT 'student' TO 'student_0';
GRANT 'student' TO 'student_1';
GRANT 'student' TO 'student_2';
GRANT 'club_president' TO 'student_3';
GRANT 'club_staff' TO 'student_4';
GRANT 'club_staff' TO 'student_5';
GRANT 'supervisor' TO 'supervisor_0';
GRANT 'supervisor' TO 'supervisor_1';
```

- WITH GRANT OPTION
  - The Role/User with keywords: WITH GRANT OPTION can grant the privileges they had to other users.
  - I thought it is dangerous to do that, so I have not added WITH GRANT OPTION to any role or user.

- REVOKE
  To remove the privilege, we can use keyword REVOKE.
  Example:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM student;
```

CREATE VIEW [With SELECT & JOIN]

- VIEW table for each club's president information.
  Including Staff_ID, Student_ID, Club_name, Name, Email, and Telephone.
  Information is from tables: Staff, Position, Student, Club.
  Code:

```
1.  CREATE VIEW Club_President AS
2.  SELECT
3.      Staff.Staff_ID, Staff.Student_ID, Club.Club_name, Student.Name, Student.Email,
    Student.Telephone
4.  FROM
5.      Staff, Position, Student, Club
6.  WHERE
7.      Position.Role = 'President'
8.      AND
9.      Staff.Student_ID =  Student.Student_ID
10.     AND
11.     Staff.Club = Club.Club_ID
12.     AND
13.     Staff.Position = Position.Position_ID;
```

- VIEW table for each club's account information.
  Including Staff_ID, Student_ID, Club_name, Name, Email, and Telephone.
  Information is from tables: Staff, Position, Student, Club.
  Code:

```
1.  CREATE VIEW Club_Accountant AS
2.  SELECT
3.      Staff.Staff_ID, Staff.Student_ID, Club.Club_name, Student.Name, Student.Email,
    Student.Telephone
4.  FROM
5.      Staff, Position, Student, Club
6.  WHERE
7.      Position.Role = 'Accountant'
8.      AND
9.      Staff.Student_ID =  Student.Student_ID
10.     AND
11.     Staff.Club = Club.Club_ID
12.     AND
13.     Staff.Position = Position.Position_ID;
```

Additional Features of SQL

- PL/SQL
  - Actually, MySQL does not have PL/SQL. PL/SQL is a stored procedure language specific to Oracle.
    (Reference: https://stackoverflow.com/questions/9808876/pl-mysql-does-it-exist/9809225)
  - But MySQL has Stored Programs as well. It has similar uses with PL/SQL.
  - I can declare variables in MySQL and use them like in PL/SQL.
  - Example:

```
1. BEGIN
2.     DECLARE var_id INT DEFAULT 0;
3.     SET var_id = OLD.Student_ID;
4.     DELETE FROM Student_Activity
5.     WHERE Student_ID = var_id;
6. END;//
```

  - It also has been used in triggers.
- AUTO_INCREMENT
  - I have used AUTO_INCREMENT for some primary keys.
  - The attribute AUTO INCREMENT can be used to create a unique identity for new rows.
  - I thought it is safer and easier to use AUTO_INCREMENT to set primary key values which is number automatically.
  - Code:

```
1. Position:          Position_ID INT NOT NULL AUTO_INCREMENT
2. Club:              Club_ID INT NOT NULL AUTO_INCREMENT
3. Student:           Student_ID INT NOT NULL AUTO_INCREMENT
4. Staff:             Staff_ID INT NOT NULL AUTO_INCREMENT
5. Supervisor:        Supervisor_ID INT NOT NULL AUTO_INCREMENT
6. Activity:          Activity_ID INT NOT NULL AUTO_INCREMENT
7. Facility:          Facility_ID INT NOT NULL AUTO_INCREMENT
8. Venue:             Venue_ID INT NOT NULL AUTO_INCREMENT
```

Appendix_0 – MySQL Code

- CREATE DATABASE --- Assuming name is TCD
- CREATE TABLE
- INSERT
- CREATE VIEW
- CREATE ROLE/USER & GRANT
- TIGGER

CREATE DATABASE --- Assuming name is TCD:

```
CREATE DATABASE TCD;
USE TCD;
```

CREATE TABLE:

```
CREATE TABLE Position (
    Position_ID INT NOT NULL AUTO_INCREMENT,
    Role VARCHAR(100) NOT NULL,
    PRIMARY KEY (Position_ID),
    CONSTRAINT Role_check CHECK(
        Role IN ('President', 'Activity Officer', 'Publicity Officer', 'Accountant', 'Memb
er')
    )
);

CREATE TABLE Club (
    Club_ID INT NOT NULL AUTO_INCREMENT,
    Club_name VARCHAR(100) NOT NULL,
    PRIMARY KEY (Club_ID)
);

CREATE TABLE Student (
    Student_ID INT NOT NULL AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Telephone VARCHAR(100) NOT NULL,
    PRIMARY KEY (Student_ID)
);

CREATE TABLE Staff (
```

```
    Staff_ID INT NOT NULL AUTO_INCREMENT,
    Student_ID INT NOT NULL,
    Club INT NOT NULL,
    Position INT NOT NULL,
    PRIMARY KEY (Staff_ID),
    FOREIGN KEY (Student_ID) REFERENCES Student (Student_ID),
    FOREIGN KEY (Club) REFERENCES Club (Club_ID),
    FOREIGN KEY (Position) REFERENCES Position (Position_ID)
);

CREATE TABLE Supervisor (
    Supervisor_ID INT NOT NULL AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Telephone VARCHAR(100) NOT NULL,
    PRIMARY KEY (Supervisor_ID)
);

CREATE TABLE Activity (
    Activity_ID INT NOT NULL AUTO_INCREMENT,
    Activity_name VARCHAR(100) NOT NULL,
    Hosted_club INT NOT NULL,
    Supervisor INT NOT NULL,
    PRIMARY KEY (Activity_ID),
    FOREIGN KEY (Hosted_club) REFERENCES Club (Club_ID),
    FOREIGN KEY (Supervisor) REFERENCES Supervisor (Supervisor_ID)
);

CREATE TABLE Student_Activity (
    Student_ID INT NOT NULL,
    Activity_ID INT NOT NULL,
    Confirmed INT DEFAULT 0,
    PRIMARY KEY (Student_ID, Activity_ID),
    FOREIGN KEY (Student_ID) REFERENCES Student (Student_ID),
    FOREIGN KEY (Activity_ID) REFERENCES Activity (Activity_ID),
    CONSTRAINT Confirmed_check CHECK(
        Confirmed IN (0,1)
    )
);

CREATE TABLE Facility (
```

```
    Facility_ID INT NOT NULL AUTO_INCREMENT,
    Facility_name VARCHAR(100) NOT NULL,
    Price FLOAT(16) NOT NULL,
    Available INT DEFAULT 1,
    PRIMARY KEY (Facility_ID),
    CONSTRAINT Facility_available_check CHECK(
        Available IN (0,1)
    )
);

CREATE TABLE Venue (
    Venue_ID INT NOT NULL AUTO_INCREMENT,
    Venue_name VARCHAR(100) NOT NULL,
    Capacity INT NOT NULL,
    District VARCHAR(100) NOT NULL,
    Street VARCHAR(100) NOT NULL,
    Building VARCHAR(100) NOT NULL,
    Available INT DEFAULT 1,
    PRIMARY KEY (Venue_ID),
    CONSTRAINT Venue_available_check CHECK(
        Available IN (0,1)
    )
);

CREATE TABLE Activity_Facility (
    Activity_ID INT NOT NULL,
    Facility_ID INT NOT NULL,
    Start_date DATE NOT NULL,
    End_date DATE NOT NULL,
    PRIMARY KEY (Activity_ID, Facility_ID),
    FOREIGN KEY (Activity_ID) REFERENCES Activity (Activity_ID),
    FOREIGN KEY (Facility_ID) REFERENCES Facility (Facility_ID)
);

CREATE TABLE Activity_Venue (
    Activity_ID INT NOT NULL,
    Venue_ID INT NOT NULL,
    Start_date DATE NOT NULL,
    End_date DATE NOT NULL,
    PRIMARY KEY (Activity_ID, Venue_ID),
    FOREIGN KEY (Activity_ID) REFERENCES Activity (Activity_ID),
```

```
    FOREIGN KEY (Venue_ID) REFERENCES Venue (Venue_ID)
);
```

INSERT:

```
INSERT INTO Position (Role) VALUES ('President');
INSERT INTO Position (Role) VALUES ('Activity Officer');
INSERT INTO Position (Role) VALUES ('Publicity Officer');
INSERT INTO Position (Role) VALUES ('Accountant');
INSERT INTO Position (Role) VALUES ('Member');

INSERT INTO Club (Club_name) VALUES ('Squash club');
INSERT INTO Club (Club_name) VALUES ('Basketball club');
INSERT INTO Club (Club_name) VALUES ('Football club');
INSERT INTO Club (Club_name) VALUES ('Tennis club');
INSERT INTO Club (Club_name) VALUES ('Swimming club');

INSERT INTO Student (Name, Email, Telephone) VALUES ('Ava', 'Ava@tcd.ie', '111111111');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Mike', 'Mike@tcd.ie', '111111112');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Leo', 'Leo@tcd.ie', '111111113');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Ella', 'Ella@tcd.ie', '111111114');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Jack', 'Jack@tcd.ie', '111111115');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Ann', 'Ann@tcd.ie', '111111116');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Harry', 'Harry@tcd.ie', '111111117')
;
INSERT INTO Student (Name, Email, Telephone) VALUES ('Mia', 'Mia@tcd.ie', '111111118');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Poppy', 'Poppy@tcd.ie', '111111119')
;
INSERT INTO Student (Name, Email, Telephone) VALUES ('Lucy', 'Lucy@tcd.ie', '111111110');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Lilly', 'Lilly@tcd.ie', '111111121')
;
INSERT INTO Student (Name, Email, Telephone) VALUES ('Max', 'Max@tcd.ie', '111111122');
INSERT INTO Student (Name, Email, Telephone) VALUES ('David', 'David@tcd.ie', '111111123')
;
INSERT INTO Student (Name, Email, Telephone) VALUES ('Alex', 'Alex@tcd.ie', '111111124');
INSERT INTO Student (Name, Email, Telephone) VALUES ('Luke', 'Luke@tcd.ie', '111111125');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s1', 's1@tcd.ie', '111111126');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s2', 's2@tcd.ie', '111111127');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s3', 's3@tcd.ie', '111111128');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s4', 's4@tcd.ie', '111111129');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s5', 's5@tcd.ie', '111111130');
```

```
INSERT INTO Student (Name, Email, Telephone) VALUES ('s6', 's6@tcd.ie', '111111131');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s7', 's7@tcd.ie', '111111132');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s8', 's8@tcd.ie', '111111133');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s9', 's9@tcd.ie', '111111134');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s10', 's10@tcd.ie', '111111135');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s11', 's11@tcd.ie', '111111136');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s12', 's12@tcd.ie', '111111137');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s13', 's13@tcd.ie', '111111138');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s14', 's14@tcd.ie', '111111139');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s15', 's15@tcd.ie', '111111140');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s16', 's16@tcd.ie', '111111141');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s17', 's17@tcd.ie', '111111142');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s18', 's18@tcd.ie', '111111143');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s19', 's19@tcd.ie', '111111144');
INSERT INTO Student (Name, Email, Telephone) VALUES ('s20', 's20@tcd.ie', '111111145');

INSERT INTO Staff (Student_ID, Club, Position) VALUES (1, 1, 1);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (2, 2, 1);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (3, 3, 1);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (4, 4, 1);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (5, 5, 1);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (6, 1, 2);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (7, 2, 2);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (8, 3, 2);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (9, 4, 2);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (10, 5, 2);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (11, 1, 3);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (12, 2, 3);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (13, 3, 3);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (14, 4, 3);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (15, 5, 3);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (16, 1, 4);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (17, 2, 4);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (18, 3, 4);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (19, 4, 4);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (20, 5, 4);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (21, 1, 5);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (22, 2, 5);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (23, 3, 5);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (24, 4, 5);
INSERT INTO Staff (Student_ID, Club, Position) VALUES (25, 5, 5);
```

```
INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Chloe', 'Chloe@tcd.ie', '21111111
1');
INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Daisy', 'Daisy@tcd.ie', '21111111
2');
INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Ruby', 'Ruby@tcd.ie', '211111113'
);
INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Teddy', 'Teddy@tcd.ie', '21111111
4');
INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Adam', 'Adam@tcd.ie', '211111115'
);

INSERT INTO Activity (Activity_name, Hosted_club, Supervisor) VALUES ('Act_1', 1, 1);
INSERT INTO Activity (Activity_name, Hosted_club, Supervisor) VALUES ('Act_2', 2, 2);
INSERT INTO Activity (Activity_name, Hosted_club, Supervisor) VALUES ('Act_3', 3, 3);
INSERT INTO Activity (Activity_name, Hosted_club, Supervisor) VALUES ('Act_4', 4, 4);
INSERT INTO Activity (Activity_name, Hosted_club, Supervisor) VALUES ('Act_5', 5, 5);

INSERT INTO Student_Activity (Student_ID, Activity_ID) VALUES (26,1);
INSERT INTO Student_Activity (Student_ID, Activity_ID) VALUES (27,2);
INSERT INTO Student_Activity (Student_ID, Activity_ID) VALUES (28,3);
INSERT INTO Student_Activity (Student_ID, Activity_ID) VALUES (29,4);
INSERT INTO Student_Activity (Student_ID, Activity_ID) VALUES (30,5);

INSERT INTO Facility (Facility_name, Price) VALUES ('Fac_1', 1);
INSERT INTO Facility (Facility_name, Price) VALUES ('Fac_2', 2);
INSERT INTO Facility (Facility_name, Price) VALUES ('Fac_3', 3);
INSERT INTO Facility (Facility_name, Price) VALUES ('Fac_4', 4);
INSERT INTO Facility (Facility_name, Price) VALUES ('Fac_5', 5);

INSERT INTO Venue (Venue_name, Capacity, District, Street, Building) VALUES ('Ven_1', 100,
 'D2', 'S1', 'B1');
INSERT INTO Venue (Venue_name, Capacity, District, Street, Building) VALUES ('Ven_2', 100,
 'D2', 'S1', 'B2');
INSERT INTO Venue (Venue_name, Capacity, District, Street, Building) VALUES ('Ven_3', 100,
 'D2', 'S1', 'B3');
INSERT INTO Venue (Venue_name, Capacity, District, Street, Building) VALUES ('Ven_4', 100,
 'D2', 'S1', 'B4');
INSERT INTO Venue (Venue_name, Capacity, District, Street, Building) VALUES ('Ven_5', 100,
 'D2', 'S1', 'B5');
```

```
INSERT INTO Activity_Facility (Activity_ID, Facility_ID, Start_date, End_date) VALUES (1,
1, '2019-12-01', '2019-12-07');
INSERT INTO Activity_Facility (Activity_ID, Facility_ID, Start_date, End_date) VALUES (2,
2, '2019-12-01', '2019-12-07');
INSERT INTO Activity_Facility (Activity_ID, Facility_ID, Start_date, End_date) VALUES (3,
3, '2019-12-01', '2019-12-07');
INSERT INTO Activity_Facility (Activity_ID, Facility_ID, Start_date, End_date) VALUES (4,
4, '2019-12-01', '2019-12-07');
INSERT INTO Activity_Facility (Activity_ID, Facility_ID, Start_date, End_date) VALUES (5,
5, '2019-12-01', '2019-12-07');

INSERT INTO Activity_Venue (Activity_ID, Venue_ID, Start_date, End_date) VALUES (1, 1, '20
19-12-01', '2019-12-07');
INSERT INTO Activity_Venue (Activity_ID, Venue_ID, Start_date, End_date) VALUES (2, 2, '20
19-12-01', '2019-12-07');
INSERT INTO Activity_Venue (Activity_ID, Venue_ID, Start_date, End_date) VALUES (3, 3, '20
19-12-01', '2019-12-07');
INSERT INTO Activity_Venue (Activity_ID, Venue_ID, Start_date, End_date) VALUES (4, 4, '20
19-12-01', '2019-12-07');
INSERT INTO Activity_Venue (Activity_ID, Venue_ID, Start_date, End_date) VALUES (5, 5, '20
19-12-01', '2019-12-07');
```

CREATE VIEW:

```
CREATE VIEW Club_President AS
SELECT
    Staff.Staff_ID, Staff.Student_ID, Club.Club_name, Student.Name, Student.Email, Student
.Telephone
FROM
    Staff, Position, Student, Club
WHERE
    Position.Role = 'President'
    AND
    Staff.Student_ID =  Student.Student_ID
    AND
    Staff.Club = Club.Club_ID
    AND
    Staff.Position = Position.Position_ID;

CREATE VIEW Club_Accountant AS
SELECT
```

```
    Staff.Staff_ID, Staff.Student_ID, Club.Club_name, Student.Name, Student.Email, Student
.Telephone
FROM
    Staff, Position, Student, Club
WHERE
    Position.Role = 'Accountant'
    AND
    Staff.Student_ID =  Student.Student_ID
    AND
    Staff.Club = Club.Club_ID
    AND
    Staff.Position = Position.Position_ID;
```

CREATE ROLE/USER & GRANT:

```
CREATE ROLE 'student', 'supervisor', 'club_president', 'club_staff';

GRANT SELECT ON TCD.Activity TO 'student', 'supervisor', 'club_president', 'club_staff';
GRANT SELECT ON TCD.Club TO 'student', 'supervisor', 'club_president', 'club_staff';
GRANT UPDATE, DELETE ON TCD.Activity TO 'supervisor';
GRANT SELECT, INSERT ON TCD.Student_Activity TO 'student';
GRANT SELECT ON TCD.Facility TO 'club_president', 'supervisor';
GRANT SELECT ON TCD.Venue TO 'club_president', 'supervisor';
GRANT SELECT, INSERT, UPDATE, DELETE ON TCD.Activity TO 'club_president';
GRANT SELECT, INSERT, UPDATE, DELETE ON TCD.Activity_Facility TO 'club_president';
GRANT SELECT, INSERT, UPDATE, DELETE ON TCD.Activity_Venue TO 'club_president';
GRANT SELECT, UPDATE ON TCD.Student_Activity TO 'club_president', 'club_staff';

CREATE USER 'student_0' IDENTIFIED BY '123456';
CREATE USER 'student_1' IDENTIFIED BY '123456';
CREATE USER 'student_2' IDENTIFIED BY '123456';
CREATE USER 'student_3' IDENTIFIED BY '123456';
CREATE USER 'student_4' IDENTIFIED BY '123456';
CREATE USER 'student_5' IDENTIFIED BY '123456';
CREATE USER 'supervisor_0' IDENTIFIED BY '123456';
CREATE USER 'supervisor_1' IDENTIFIED BY '123456';

GRANT 'student' TO 'student_0';
GRANT 'student' TO 'student_1';
GRANT 'student' TO 'student_2';
GRANT 'club_president' TO 'student_3';
```

```
GRANT 'club_staff' TO 'student_4';
GRANT 'club_staff' TO 'student_5';
GRANT 'supervisor' TO 'supervisor_0';
GRANT 'supervisor' TO 'supervisor_1';
```

TIGGER:

```
delimiter //
CREATE TRIGGER facility_price_tigger BEFORE INSERT ON Facility
FOR EACH ROW
BEGIN
    IF NEW.Price < 0
    THEN SET NEW.Price = 0;
    END IF;
END;//

delimiter //
CREATE TRIGGER student_staff_tigger BEFORE DELETE ON Student
FOR EACH ROW
BEGIN
    DECLARE var_id INT DEFAULT 0;
    SET var_id = OLD.Student_ID;
    DELETE FROM Staff
    WHERE Student_ID = var_id;
END;//

delimiter //
CREATE TRIGGER student_activity_tigger BEFORE DELETE ON Student
FOR EACH ROW
BEGIN
    DECLARE var_id INT DEFAULT 0;
    SET var_id = OLD.Student_ID;
    DELETE FROM Student_Activity
    WHERE Student_ID = var_id;
END;//
```

Appendix_1 – Showcases for Some Achievements

- TABLE --- Overview
- VIEW --- Club_President

- CREATE ROLE/USER & GRANT --- club_president & student_0
- TIGGER --- Student

TABLE --- Overview:

```
mysql> show tables;
+-------------------+
| Tables_in_tcd     |
+-------------------+
| activity          |
| activity_facility |
| activity_venue    |
| club              |
| club_accountant   |
| club_president    |
| facility          |
| position          |
| staff             |
| student           |
| student_activity  |
| supervisor        |
| venue             |
+-------------------+
```

VIEWS --- Club_President:

```
mysql> select * from club_president;
+----------+------------+-----------------+------+-------------+-----------+
| Staff_ID | Student_ID | Club_name       | Name | Email       | Telephone |
+----------+------------+-----------------+------+-------------+-----------+
|        1 |          1 | Squash club     | Ava  | Ava@tcd.ie  | 111111111 |
|        2 |          2 | Basketball club | Mike | Mike@tcd.ie | 111111112 |
|        3 |          3 | Football club   | Leo  | Leo@tcd.ie  | 111111113 |
|        4 |          4 | Tennis club     | Ella | Ella@tcd.ie | 111111114 |
|        5 |          5 | Swimming club   | Jack | Jack@tcd.ie | 111111115 |
+----------+------------+-----------------+------+-------------+-----------+
```

CREATE ROLE/USER & GRANT --- club_president & student_0:

```
mysql> SHOW GRANTS FOR club_president;
+-----------------------------------------------------------------------------------------------------------+
| Grants for club_president@%                                                                               |
+-----------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `club_president`@`%`                                                                |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `tcd`.`activity_facility` TO `club_president`@`%`                 |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `tcd`.`activity_venue` TO `club_president`@`%`                    |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `tcd`.`activity` TO `club_president`@`%`                          |
| GRANT SELECT ON `tcd`.`club` TO `club_president`@`%`                                                      |
| GRANT SELECT ON `tcd`.`facility` TO `club_president`@`%`                                                  |
| GRANT SELECT, UPDATE ON `tcd`.`student_activity` TO `club_president`@`%`                                  |
| GRANT SELECT ON `tcd`.`venue` TO `club_president`@`%`                                                     |
+-----------------------------------------------------------------------------------------------------------+
```

```
mysql> SHOW GRANTS FOR student_0;
+---------------------------------------+
| Grants for student_0@%                |
+---------------------------------------+
| GRANT USAGE ON *.* TO `student_0`@`%` |
| GRANT `student`@`%` TO `student_0`@`%` |
+---------------------------------------+
```

TIGGER --- Student:

```
mysql> show triggers like 'student'\G;
*************************** 1. row ***************************
             Trigger: student_staff_tigger
               Event: DELETE
               Table: student
           Statement: BEGIN
    DECLARE var_id INT DEFAULT 0;
    SET var_id = OLD.Student_ID;
    DELETE FROM Staff
    WHERE Student_ID = var_id;
END
              Timing: BEFORE
             Created: 2019-11-29 13:28:23.27
            sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
             Definer: root@localhost
character_set_client: gbk
collation_connection: gbk_chinese_ci
  Database Collation: utf8mb4_0900_ai_ci
*************************** 2. row ***************************
             Trigger: student_activity_tigger
               Event: DELETE
               Table: student
           Statement: BEGIN
    DECLARE var_id INT DEFAULT 0;
    SET var_id = OLD.Student_ID;
    DELETE FROM Student_Activity
    WHERE Student_ID = var_id;
END
              Timing: BEFORE
             Created: 2019-11-29 13:28:23.28
            sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
             Definer: root@localhost
character_set_client: gbk
collation_connection: gbk_chinese_ci
  Database Collation: utf8mb4_0900_ai_ci
2 rows in set (0.00 sec)
```