**Student Club Activities Management System Database**

Can ZHOU

19324118

[zhouc@tcd.ie](mailto:zhouc@tcd.ie)

Contents

[Background Information & Assumption 1](#_Toc25933651)

[Entity Relationship Diagram 3](#_Toc25933652)

[Mapping to Relational Schema 4](#_Toc25933653)

[Functional Dependency Diagram 5](#_Toc25933654)

[Normalisation [Definitions are from lecture slides] 6](#_Toc25933655)

[Notice 6](#_Toc25933656)

[Implicit Constraints 6](#_Toc25933657)

[Semantic Constraints 7](#_Toc25933658)

[Database Security Commands for Access and Security Policy 10](#_Toc25933659)

[CREATE VIEW [With SELECT & JOIN] 12](#_Toc25933660)

[Additional Features of SQL 13](#_Toc25933661)

[Appendix\_0 – MySQL Code 14](#_Toc25933662)

[Appendix\_1 – Showcases for Some Achievements 22](#_Toc25933663)

\*\* 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.

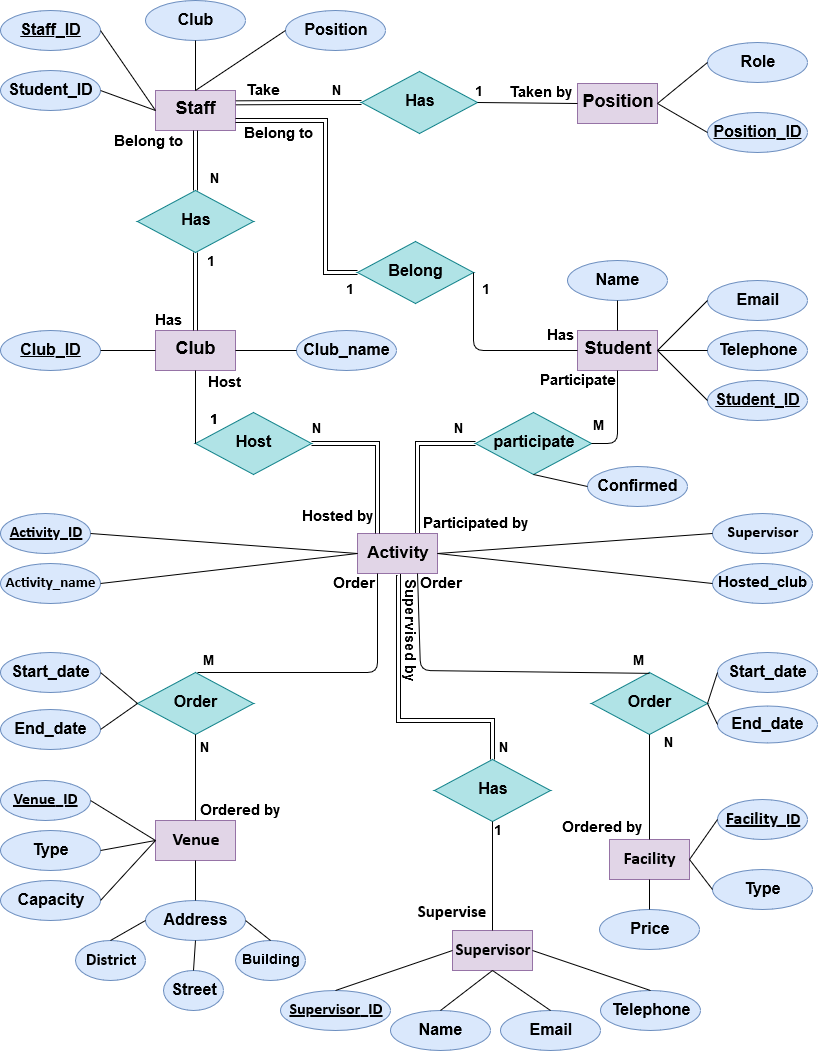
# Background Information & Assumption

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.

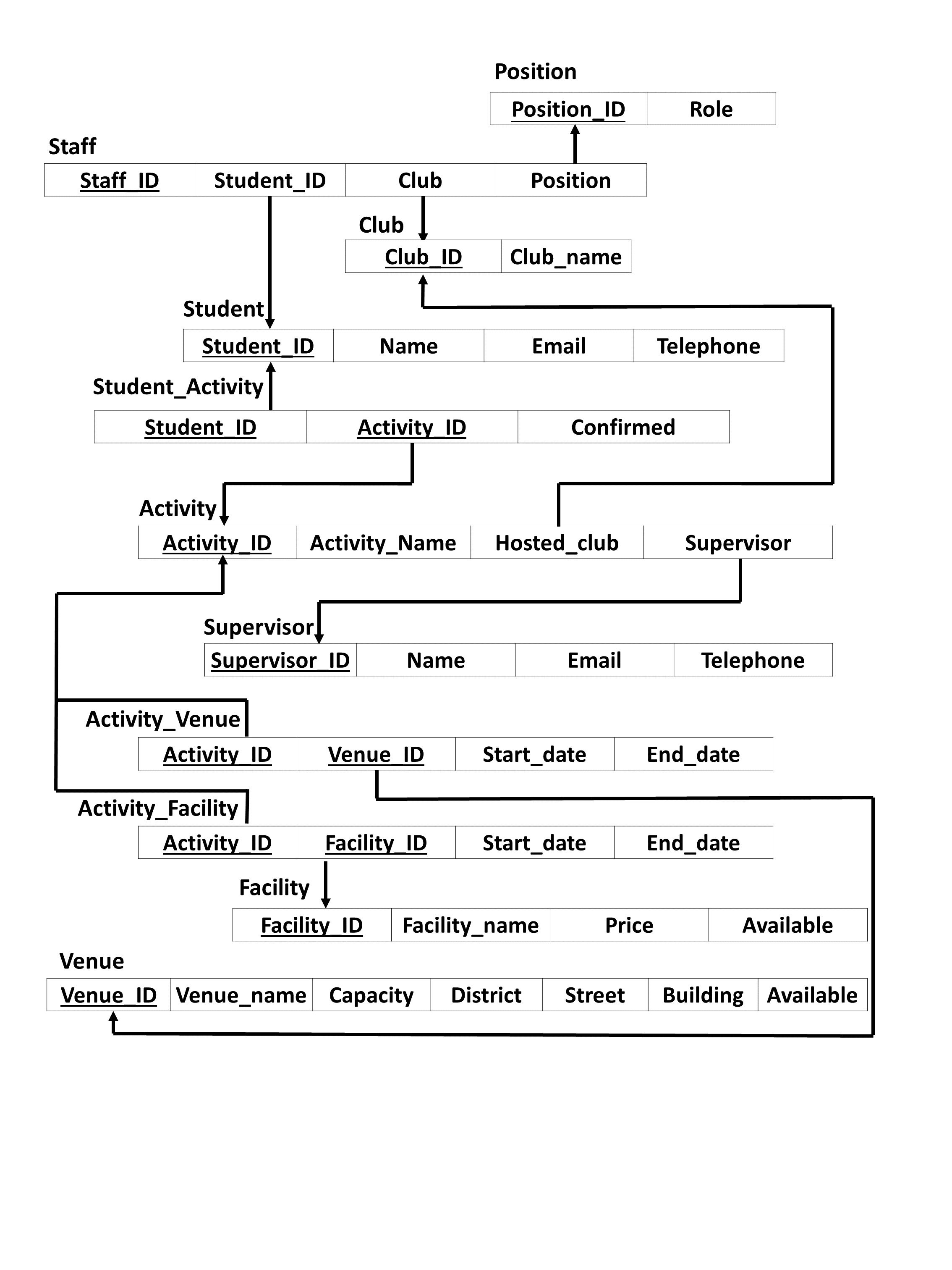
# Entity Relationship Diagram

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.**

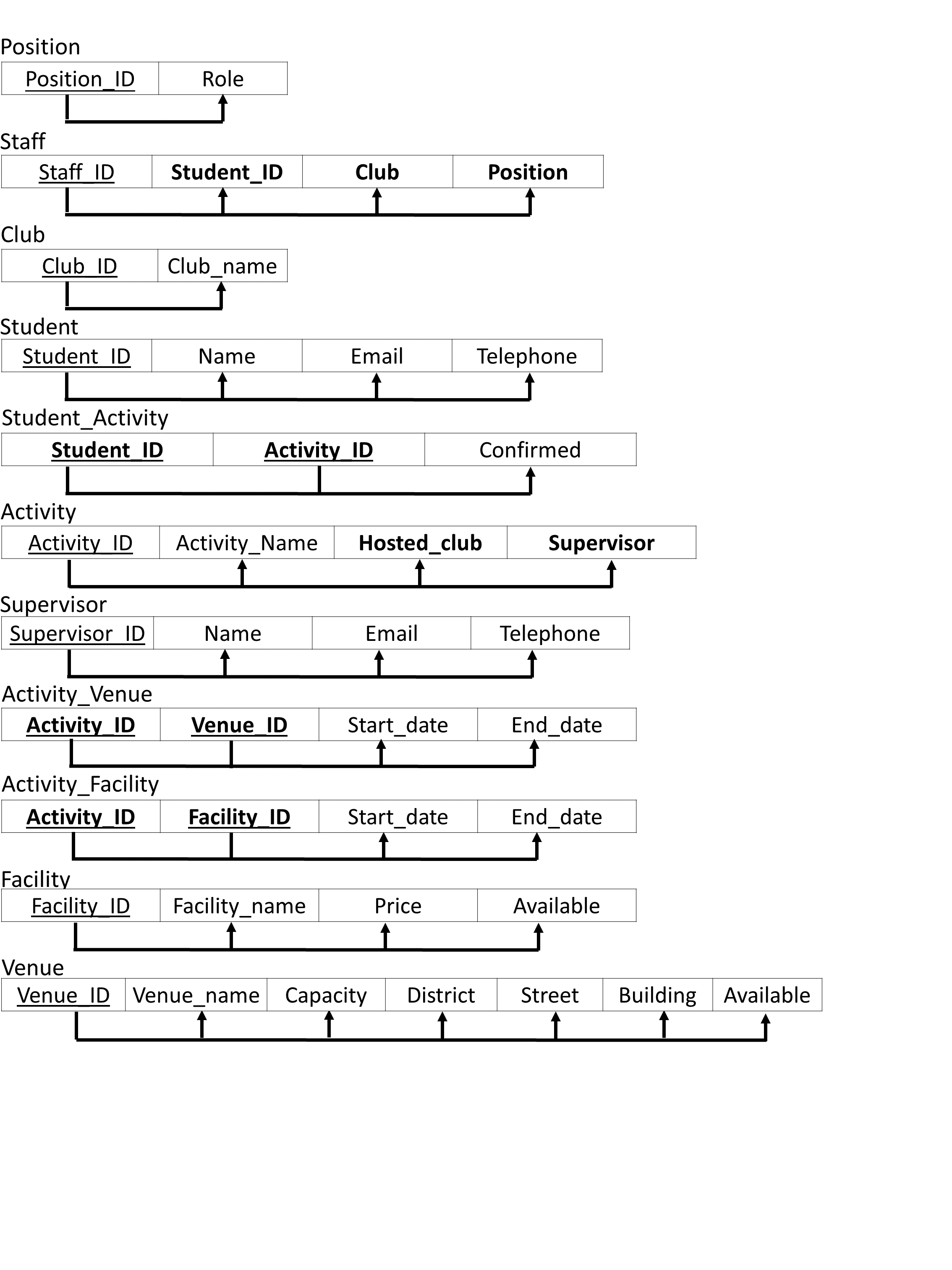


# Functional Dependency Diagram

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 Officer', '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:

1. **delimiter** //
2. **CREATE** **TRIGGER** student\_activity\_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** Student\_Activity
8. **WHERE** Student\_ID = var\_id;
9. **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:

1. **GRANT** **SELECT** **ON** TCD.Activity **TO** 'student', 'supervisor', 'club\_president', 'club\_staff';
2. **GRANT** **SELECT** **ON** TCD.Club **TO** 'student', 'supervisor', 'club\_president', 'club\_staff';
3. **GRANT** **UPDATE**, **DELETE** **ON** TCD.Activity **TO** 'supervisor';
4. **GRANT** **SELECT**, **INSERT** **ON** TCD.Student\_Activity **TO** 'student';
5. **GRANT** **SELECT** **ON** TCD.Facility **TO** 'club\_president', 'supervisor';
6. **GRANT** **SELECT** **ON** TCD.Venue **TO** 'club\_president', 'supervisor';
7. **GRANT** **SELECT**, **INSERT**, **UPDATE**, **DELETE** **ON** TCD.Activity **TO** 'club\_president';
8. **GRANT** **SELECT**, **INSERT**, **UPDATE**, **DELETE** **ON** TCD.Activity\_Facility **TO** 'club\_president';
9. **GRANT** **SELECT**, **INSERT**, **UPDATE**, **DELETE** **ON** TCD.Activity\_Venue **TO** 'club\_president';
10. **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:

1. **CREATE** USER 'student\_0' **IDENTIFIED** **BY** '123456';
2. **CREATE** USER 'student\_1' **IDENTIFIED** **BY** '123456';
3. **CREATE** USER 'student\_2' **IDENTIFIED** **BY** '123456';
4. **CREATE** USER 'student\_3' **IDENTIFIED** **BY** '123456';
5. **CREATE** USER 'student\_4' **IDENTIFIED** **BY** '123456';
6. **CREATE** USER 'student\_5' **IDENTIFIED** **BY** '123456';
7. **CREATE** USER 'supervisor\_0' **IDENTIFIED** **BY** '123456';
8. **CREATE** USER 'supervisor\_1' **IDENTIFIED** **BY** '123456';

* Grant privileges to users:

1. **GRANT** 'student' **TO** 'student\_0';
2. **GRANT** 'student' **TO** 'student\_1';
3. **GRANT** 'student' **TO** 'student\_2';
4. **GRANT** 'club\_president' **TO** 'student\_3';
5. **GRANT** 'club\_staff' **TO** 'student\_4';
6. **GRANT** 'club\_staff' **TO** 'student\_5';
7. **GRANT** 'supervisor' **TO** 'supervisor\_0';
8. **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', 'Member')

    )

);

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', '211111111');

INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Daisy', 'Daisy@tcd.ie', '211111112');

INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Ruby', 'Ruby@tcd.ie', '211111113');

INSERT INTO Supervisor (Name, Email, Telephone) VALUES ('Teddy', 'Teddy@tcd.ie', '211111114');

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, '2019-12-01', '2019-12-07');

INSERT INTO Activity\_Venue (Activity\_ID, Venue\_ID, Start\_date, End\_date) VALUES (2, 2, '2019-12-01', '2019-12-07');

INSERT INTO Activity\_Venue (Activity\_ID, Venue\_ID, Start\_date, End\_date) VALUES (3, 3, '2019-12-01', '2019-12-07');

INSERT INTO Activity\_Venue (Activity\_ID, Venue\_ID, Start\_date, End\_date) VALUES (4, 4, '2019-12-01', '2019-12-07');

INSERT INTO Activity\_Venue (Activity\_ID, Venue\_ID, Start\_date, End\_date) VALUES (5, 5, '2019-12-01', '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

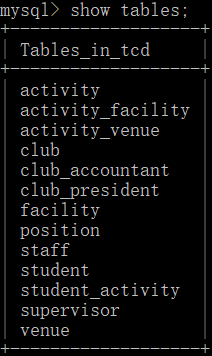
    WHERE Student\_ID = var\_id;

END;//

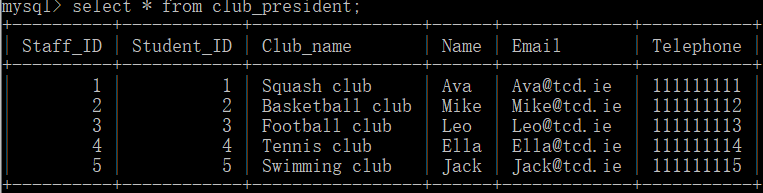
# Appendix\_1 – Showcases for Some Achievements

* TABLE --- Overview
* VIEW --- Club\_President
* CREATE ROLE/USER & GRANT --- club\_president & student\_0
* TIGGER --- Student

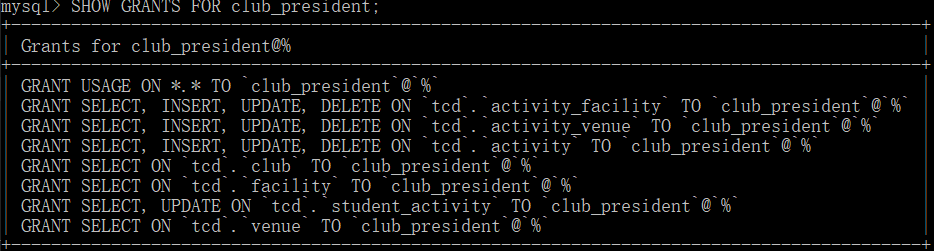
TABLE --- Overview:

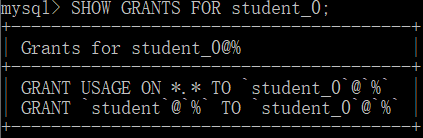


VIEWS --- Club\_President:



CREATE ROLE/USER & GRANT --- club\_president & student\_0:





TIGGER --- Student:

