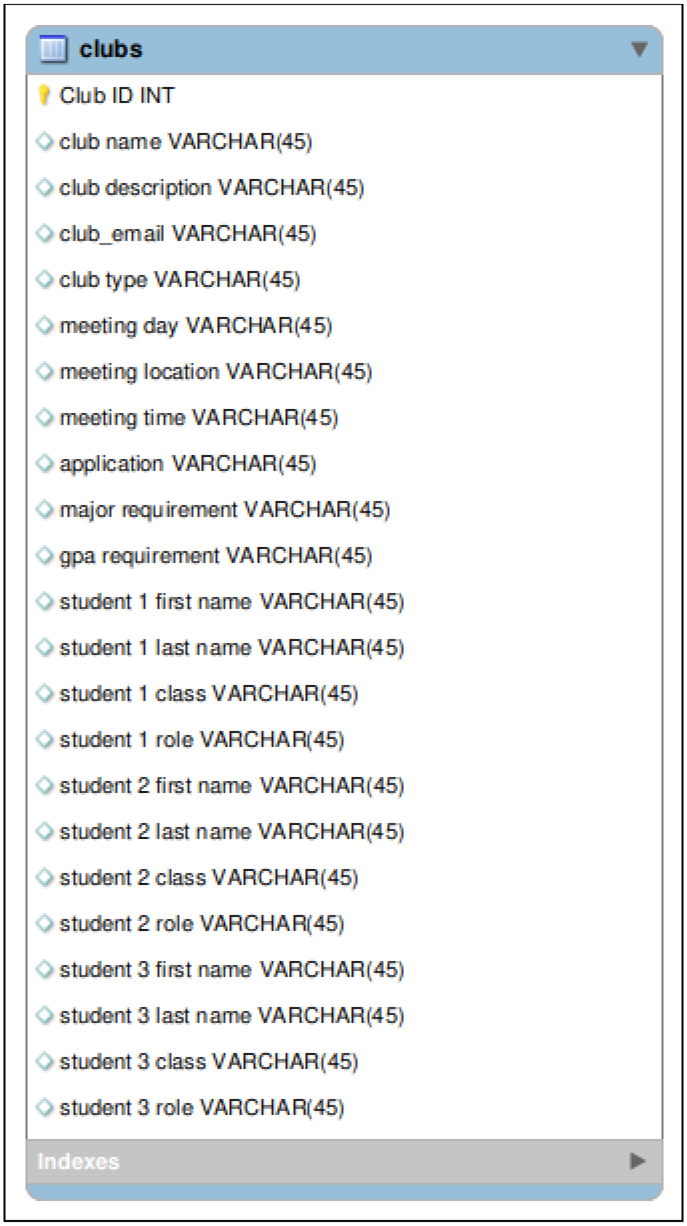
Alec Mattu
INST 327
04/03/2021

UNF

The values here are representative of what appeared in the Assignment 3 instructions.

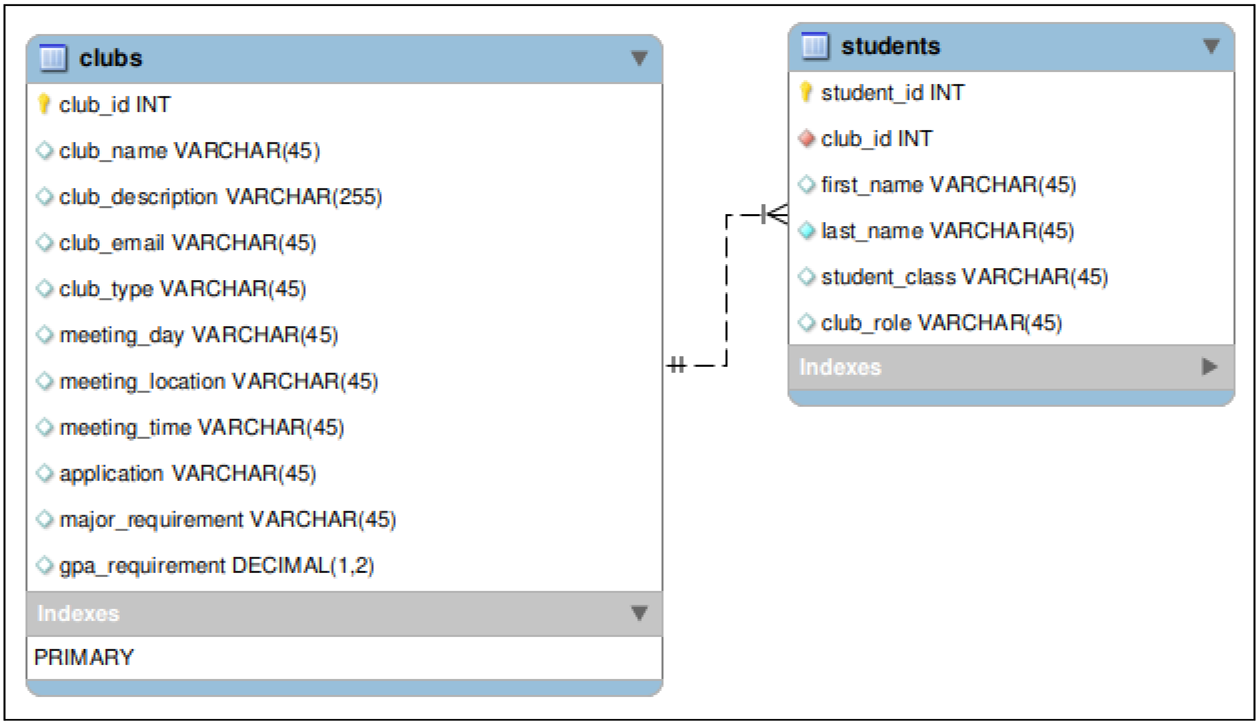


clubs

- Club ID INT
- club name VARCHAR(45)
- club description VARCHAR(45)
- club_email VARCHAR(45)
- club type VARCHAR(45)
- meeting day VARCHAR(45)
- meeting location VARCHAR(45)
- meeting time VARCHAR(45)
- application VARCHAR(45)
- major requirement VARCHAR(45)
- gpa requirement VARCHAR(45)
- student 1 first name VARCHAR(45)
- student 1 last name VARCHAR(45)
- student 1 class VARCHAR(45)
- student 1 role VARCHAR(45)
- student 2 first name VARCHAR(45)
- student 2 last name VARCHAR(45)
- student 2 class VARCHAR(45)
- student 2 role VARCHAR(45)
- student 3 first name VARCHAR(45)
- student 3 last name VARCHAR(45)
- student 3 class VARCHAR(45)
- student 3 role VARCHAR(45)

1NF

Some rewording of the club attribute column names as well as type changes. We’re not far off of the non-normalized form, as the primary interest with 1NF is assigning a unique identity to the table row. Additionally, I moved students to their own table.



clubs

- club_id INT
- club_name VARCHAR(45)
- club_description VARCHAR(255)
- club_email VARCHAR(45)
- club_type VARCHAR(45)
- meeting_day VARCHAR(45)
- meeting_location VARCHAR(45)
- meeting_time VARCHAR(45)
- application VARCHAR(45)
- major_requirement VARCHAR(45)
- gpa_requirement DECIMAL(1,2)

Indexes

PRIMARY

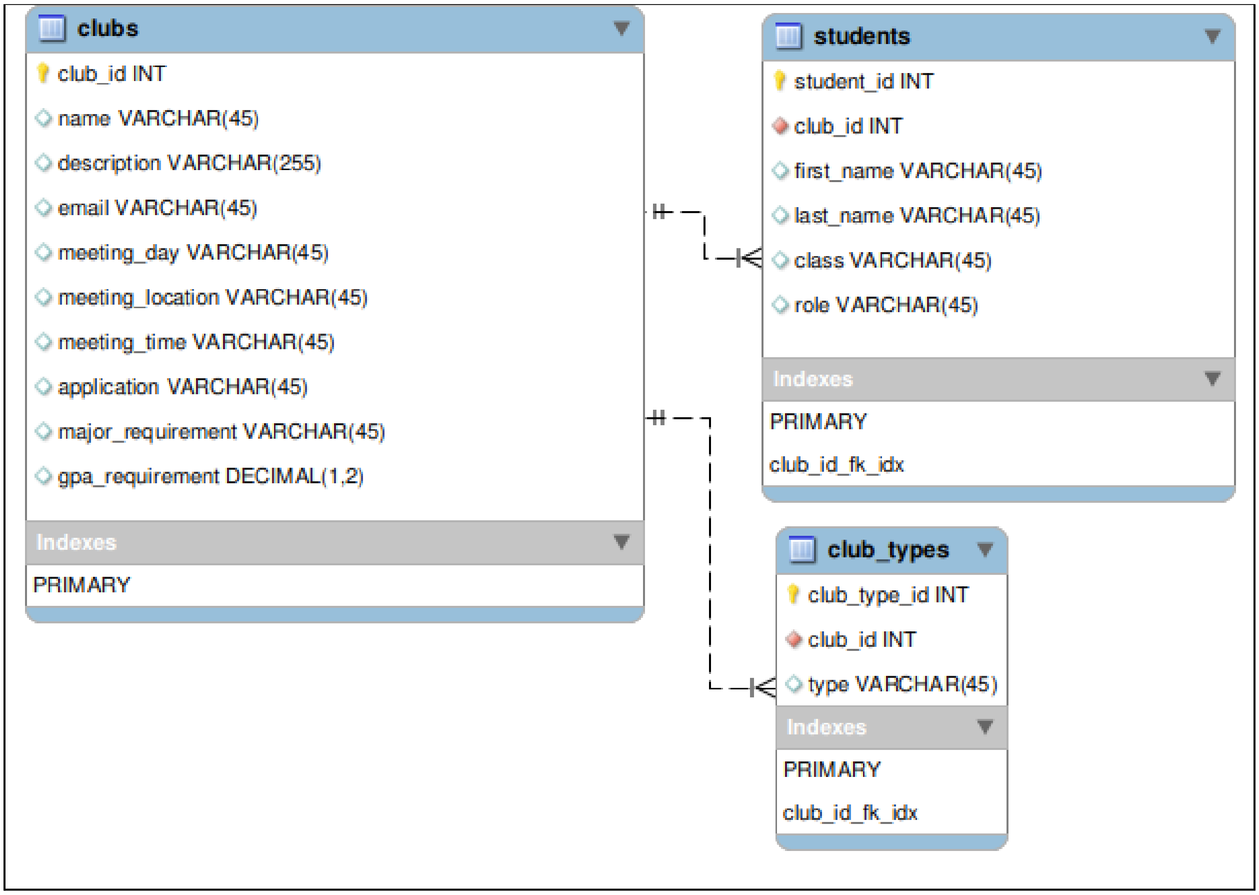
students

- student_id INT
- club_id INT
- first_name VARCHAR(45)
- last_name VARCHAR(45)
- student_class VARCHAR(45)
- club_role VARCHAR(45)

Indexes

2NF

This is where we get to finally organize non-dependent values and split them into their own respective tables. Additionally, since the “club_*” columns belong to the clubs table, we don’t need a prefix for the column names--So they were dropped. At this point, “students” belong to clubs, and cannot be in multiple clubs via PK. Club types (I.E. educational, professional, etc) are now in their own table as well.



clubs

- club_id INT
- name VARCHAR(45)
- description VARCHAR(255)
- email VARCHAR(45)
- meeting_day VARCHAR(45)
- meeting_location VARCHAR(45)
- meeting_time VARCHAR(45)
- application VARCHAR(45)
- major_requirement VARCHAR(45)
- gpa_requirement DECIMAL(1,2)

Indexes

- PRIMARY

students

- student_id INT
- club_id INT
- first_name VARCHAR(45)
- last_name VARCHAR(45)
- class VARCHAR(45)
- role VARCHAR(45)

Indexes

- PRIMARY
- club_id_fk_idx

club_types

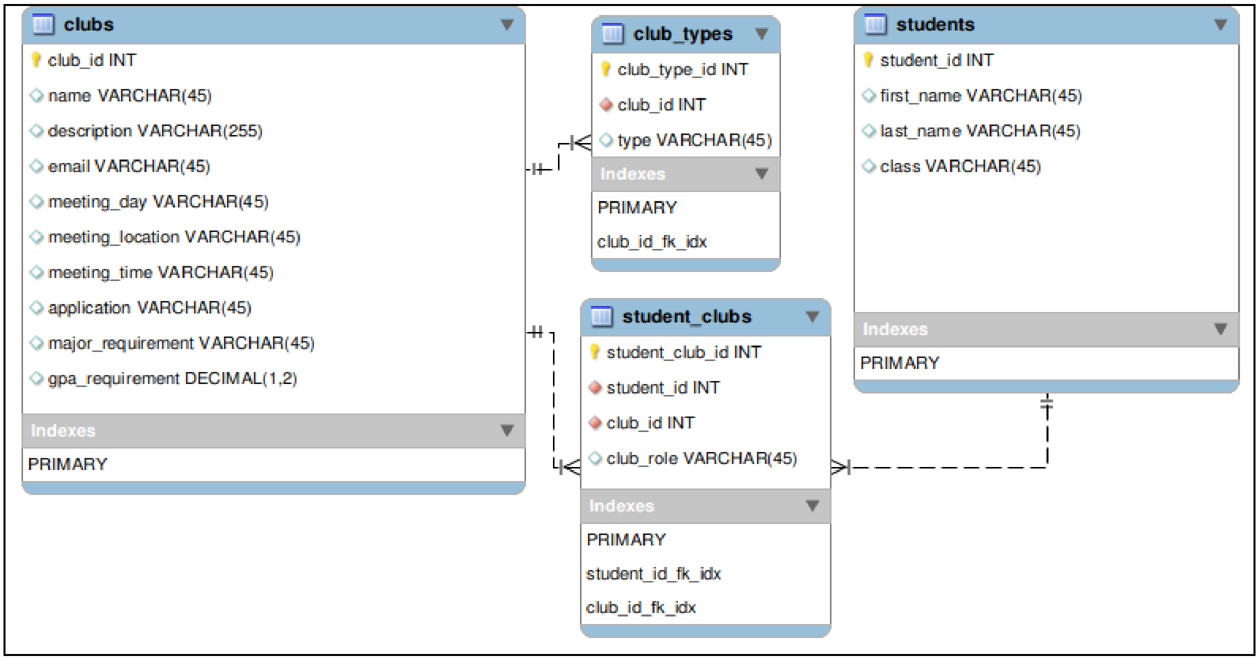
- club_type_id INT
- club_id INT
- type VARCHAR(45)

Indexes

- PRIMARY
- club_id_fk_idx

3NF

Now, in order to prevent unnecessary data duplication, let’s make it so students can be in multiple clubs; In order to do this, we need to create a new linking table. By making this transition, we will also move the student's club role into that linking table, as well. The student “class” column is independent of the club, so this will remain in place.



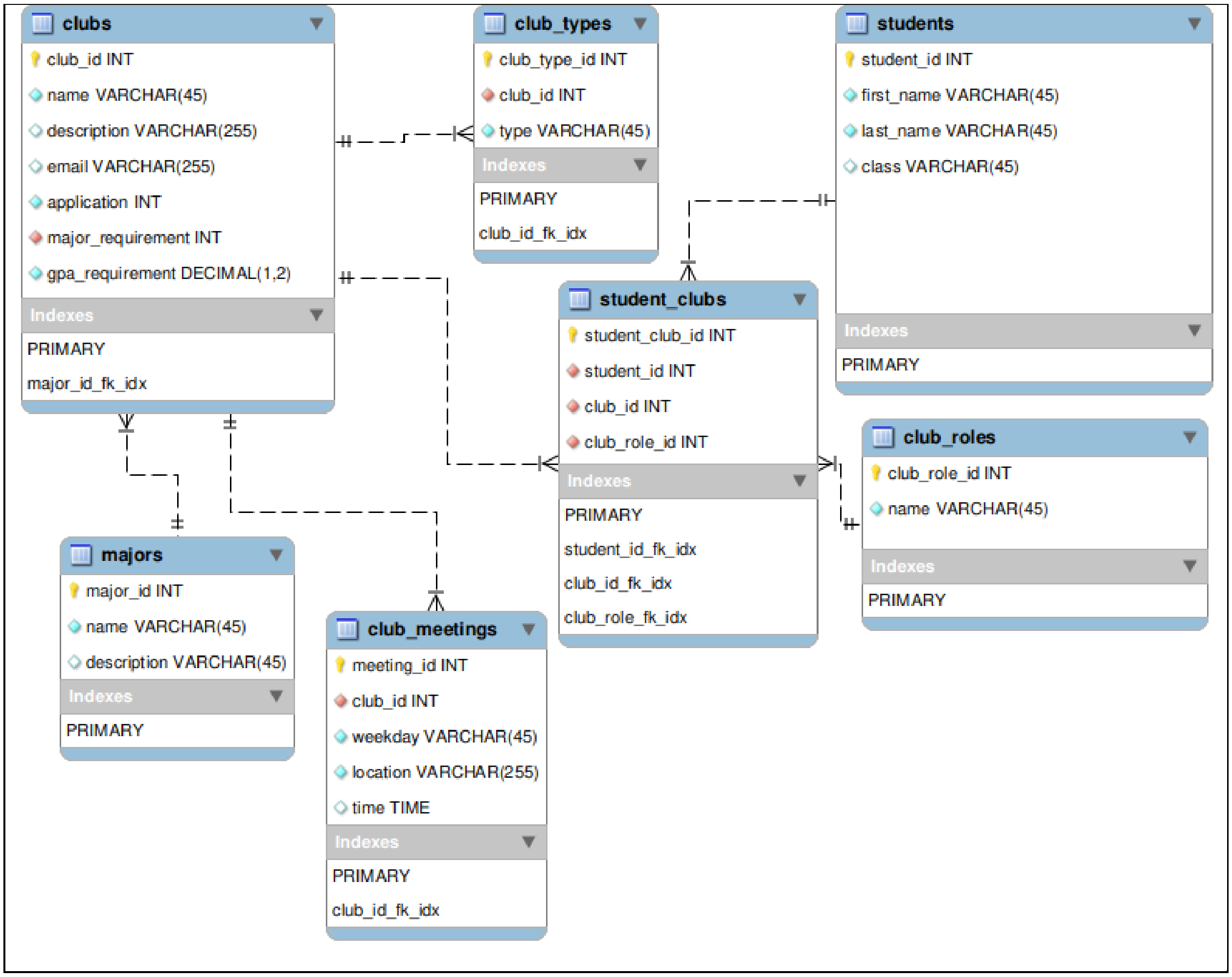
clubs
club_id INT
name VARCHAR(45)
description VARCHAR(255)
email VARCHAR(45)
meeting_day VARCHAR(45)
meeting_location VARCHAR(45)
meeting_time VARCHAR(45)
application VARCHAR(45)
major_requirement VARCHAR(45)
gpa_requirement DECIMAL(1,2)
Indexes
PRIMARY

club_types
club_type_id INT
club_id INT
type VARCHAR(45)
Indexes
PRIMARY
club_id_fk_idx

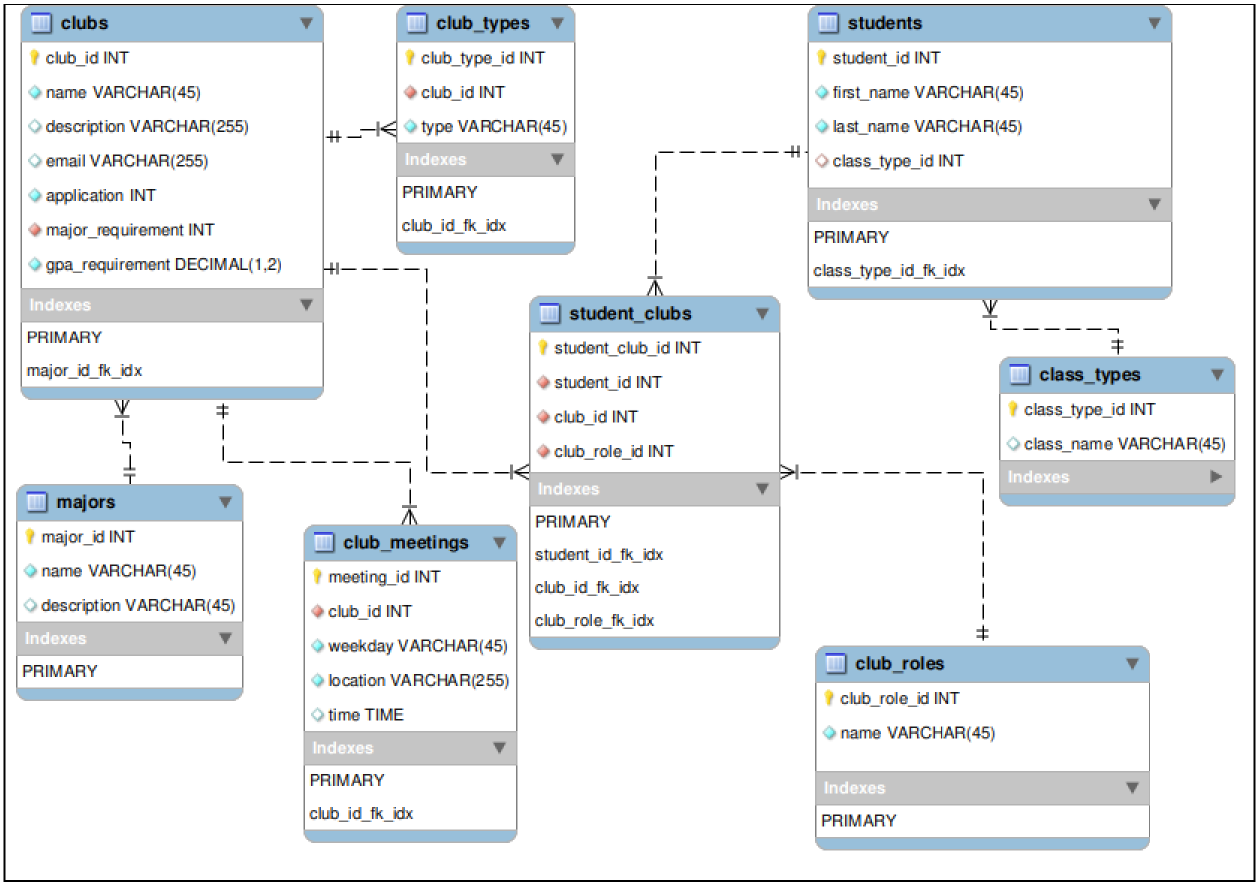
students
student_id INT
first_name VARCHAR(45)
last_name VARCHAR(45)
class VARCHAR(45)
Indexes
PRIMARY

student_clubs
student_club_id INT
student_id INT
club_id INT
club_role VARCHAR(45)
Indexes
PRIMARY
student_id_fk_idx
club_id_fk_idx

Although meetings could potentially remain on the clubs table, for sake of data integrity and logging, let’s move it to it’s own table. Additionally, I created a table “club_roles” which is a textual description of what each role is and this table is referenced by “student_clubs.” At this point I’ve made most of the columns NOT NULL, and added more specific requirements of columns (I.E. meeting_time became a TIME type column, clubs.application became 0/1 for true or false). See below for the final (3NF) normalized table. One final note, I also converted major_requirement to a foreign key, where a NULL value indicates that the club is open to all--If it’s open to more than one major, we would need to use a linking table like student_clubs (it doesn’t appear as if it would be).



One last task is to move students.class values to their own table, and just reference the class ID as a foreign key.



And that’s a 3NF database.

SQL Script

```
-- MySQL Script generated by MySQL Workbench
-- Sat 03 Apr 2021 05:01:15 PM EDT
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_Z
ERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`majors`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`majors` (
  `major_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(45) NULL,
  PRIMARY KEY (`major_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`clubs`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`clubs` (
  `club_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `application` INT NOT NULL,
  `major_requirement` INT NULL DEFAULT NULL,
  `gpa_requirement` DECIMAL(1,2) NOT NULL,
  PRIMARY KEY (`club_id`),
  INDEX `major_id_fk_idx` (`major_requirement` ASC) VISIBLE,
  CONSTRAINT `major_id_fk`
    FOREIGN KEY (`major_requirement`)
    REFERENCES `mydb`.`majors` (`major_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`class_types`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`class_types` (
  `class_type_id` INT NOT NULL AUTO_INCREMENT,
  `class_name` VARCHAR(45) NULL,
  PRIMARY KEY (`class_type_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`students`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`students` (
  `student_id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `class_type_id` INT NULL,
  PRIMARY KEY (`student_id`),
  INDEX `class_type_id_fk_idx` (`class_type_id` ASC) VISIBLE,
  CONSTRAINT `class_type_id_fk`
    FOREIGN KEY (`class_type_id`)
    REFERENCES `mydb`.`class_types` (`class_type_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`club_types`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`club_types` (
  `club_type_id` INT NOT NULL AUTO_INCREMENT,
  `club_id` INT NOT NULL,
  `type` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`club_type_id`),
  INDEX `club_id_fk_idx` (`club_id` ASC) VISIBLE,
  CONSTRAINT `club_id_fk`
    FOREIGN KEY (`club_id`)
    REFERENCES `mydb`.`clubs` (`club_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`club_roles`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`club_roles` (
  `club_role_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`club_role_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`student_clubs`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`student_clubs` (
  `student_club_id` INT NOT NULL AUTO_INCREMENT,
  `student_id` INT NOT NULL,
  `club_id` INT NOT NULL,
  `club_role_id` INT NOT NULL,
  PRIMARY KEY (`student_club_id`),
  INDEX `student_id_fk_idx` (`student_id` ASC) VISIBLE,
  INDEX `club_id_fk_idx` (`club_id` ASC) VISIBLE,
  INDEX `club_role_fk_idx` (`club_role_id` ASC) VISIBLE,
  CONSTRAINT `student_id_fk`
    FOREIGN KEY (`student_id`)
    REFERENCES `mydb`.`students` (`student_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `club_id_fk`
    FOREIGN KEY (`club_id`)
    REFERENCES `mydb`.`clubs` (`club_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `club_role_fk`
    FOREIGN KEY (`club_role_id`)
    REFERENCES `mydb`.`club_roles` (`club_role_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`club_meetings`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`club_meetings` (
  `meeting_id` INT NOT NULL AUTO_INCREMENT,
  `club_id` INT NOT NULL,
  `weekday` VARCHAR(45) NOT NULL,
  `location` VARCHAR(255) NOT NULL,
  `time` TIME NULL,
  PRIMARY KEY (`meeting_id`),
  INDEX `club_id_fk_idx` (`club_id` ASC) VISIBLE,
  CONSTRAINT `club_id_fk`
    FOREIGN KEY (`club_id`)
    REFERENCES `mydb`.`clubs` (`club_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```