

Kevin Turkington
CS340
12/2/17
Final Project

Overview:

This project is simply a outline of a university universe. It models the relationship between departments to buildings, buildings to classes, departments to classes, and classes to students.

Db Outline:

On a more detailed description, a single department can be associated with any number of buildings a one to many relationship. Buildings as previously stated can be associated with only one department, but are also associated with many classes (again one to many relationship). Classes have a total of three different types of relationships, it can be associated to a single department and building. For example 'introduction to databases' is associated to the computer science department and the 'Ecampus' building in the real world. Classes also contain a many to many relationship with students, however a student can only enroll in a class once. And Finally students contain only one relationship the many to many relationship as previously stated with classes.

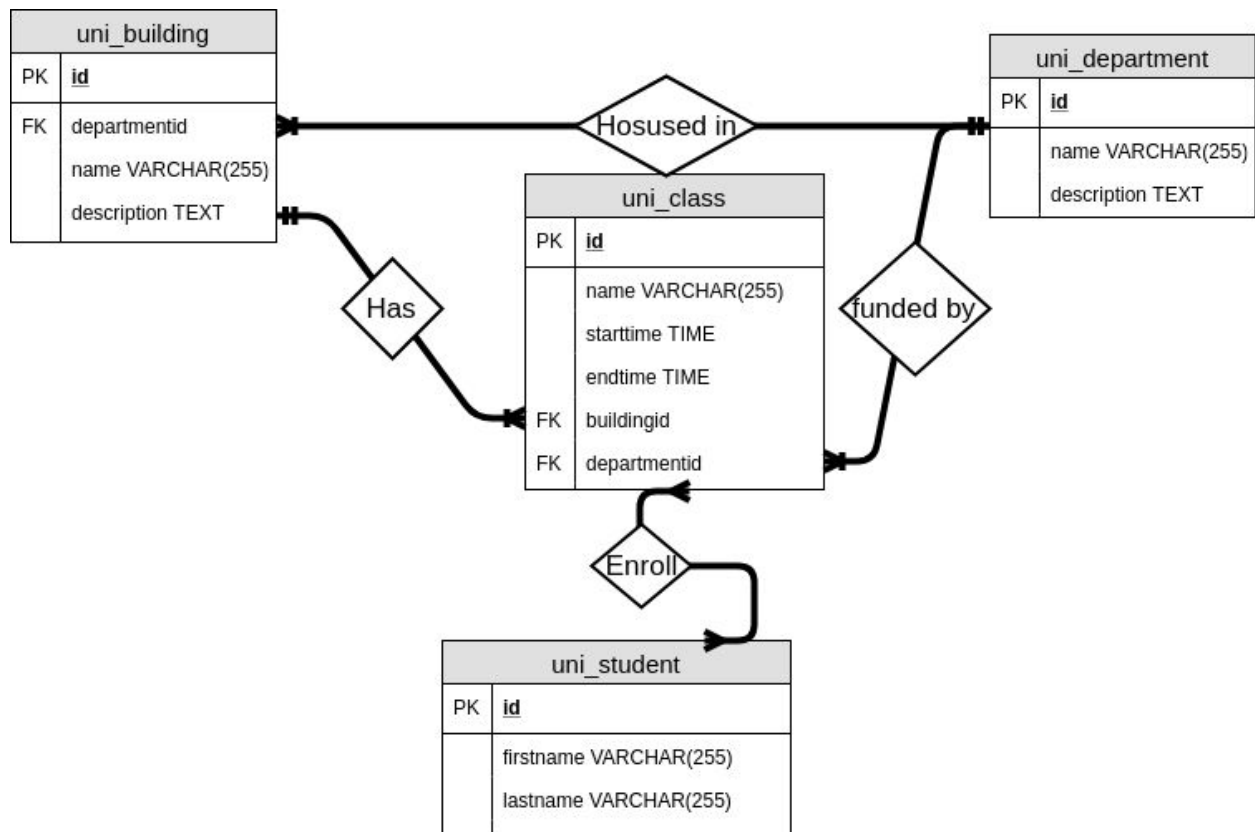
Build Instructions:

1. git clone [git@github.com:zainkai/university_app.git](https://github.com/zainkai/university_app.git)
2. cd university_app
3. npm install
4. cd src/server/db
5. mkdir protected
6. cd protected
7. touch dbcon-dev.ts
8. vim dbcon-dev.ts

```
import * as mysql from 'mysql';
export const pool = mysql.createPool({
  connectionLimit : 10,
  host            : 'mysql.eecs.oregonstate.edu',
  user            : 'cs340_youengrusername',
  password        : 'last-4-digits-of-your-osu-id',
  database        : 'cs340_youengrusername'
});
```

9. npm run build-all
10. npm run start

ER diagram:



Db Schema:

*These can be found in dbModel.

```
CREATE TABLE `uni_department` (  
  `id` INT AUTO_INCREMENT PRIMARY KEY,  
  `name` VARCHAR(255) NOT NULL,  
  `description` TEXT,  
  CONSTRAINT UNIQUE (`name`)  
) ENGINE=InnoDB;
```

```
CREATE TABLE `uni_building` (  
  `id` INT AUTO_INCREMENT PRIMARY KEY,  
  `departmentid` INT NOT NULL,  
  `name` VARCHAR(255) NOT NULL,  
  `description` TEXT,  
  FOREIGN KEY(`departmentid`) REFERENCES `uni_department`(`id`),  
  CONSTRAINT UNIQUE (`name`)  
) ENGINE=InnoDB;
```

```
CREATE TABLE `uni_class` (  
  `id` INT AUTO_INCREMENT PRIMARY KEY,  
  `name` VARCHAR(255) NOT NULL,  
  `starttime` TIME,  
  `endtime` TIME,  
  `buildingid` INT NOT NULL,  
  `departmentid` INT NOT NULL,  
  FOREIGN KEY(`buildingid`) REFERENCES `uni_building`(`id`),  
  FOREIGN KEY(`departmentid`) REFERENCES `uni_department`(`id`)  
) ENGINE=InnoDB;
```

```
CREATE TABLE `uni_student` (  
  `id` INT AUTO_INCREMENT PRIMARY KEY,  
  `firstname` VARCHAR(255) NOT NULL,  
  `lastname` VARCHAR(255) NOT NULL,  
  CONSTRAINT `fullname` UNIQUE (`firstname`, `lastname`)  
) ENGINE=InnoDB;
```

```
CREATE TABLE `uni_class_enrollment` (  
  `id` INT AUTO_INCREMENT PRIMARY KEY,  
  `studentid` INT NOT NULL,  
  `classid` INT NOT NULL,  
  FOREIGN KEY(`studentid`) REFERENCES `uni_student`(`id`),  
  FOREIGN KEY(`classid`) REFERENCES `uni_class`(`id`),  
  CONSTRAINT UNIQUE (`studentid`, `classid`)  
) ENGINE=InnoDB;
```

Db Queries:

Departments:

--Get a specific department

```
SELECT * FROM uni_department  
WHERE id=[department id]
```

--Get all departments

```
SELECT * FROM uni_department
```

--Add a new department

```
INSERT INTO uni_department (name,description)  
    values ([Department Name],  
           [Department Desc]  
    );
```

--Get only the id and name of a department

```
SELECT id,name FROM uni_department`
```

Buildings:

--Get a specific building

```
SELECT b.id, b.departmentid, b.name, b.description, d.name AS departmentName  
    FROM uni_building b  
    INNER JOIN uni_department d ON b.departmentid=d.id  
WHERE b.id=[building id]
```

--Get all buildings

```
SELECT b.id, b.departmentid, b.name, b.description, d.name AS departmentName  
    FROM uni_building b  
    INNER JOIN uni_department d ON b.departmentid=d.id
```

--Add a new building

```
INSERT INTO uni_building (name,description,departmentid)
    values ([Name],
            [description],
            [associated department id]
    );
```

--Get only building name and id

```
SELECT id,name FROM uni_building
```

Class:

--Get a specific class

```
SELECT c.id, c.name,
       c.starttime, c.endtime,
       c.buildingid, c.departmentid,
       d.name AS departmentName, b.name AS buildingName

FROM uni_class c
INNER JOIN uni_department d ON c.departmentid=d.id
INNER JOIN uni_building b ON c.buildingid=b.id
WHERE c.id=[Class id]
```

--Get all classes

```
SELECT c.id, c.name,
       c.starttime, c.endtime,
       c.buildingid, c.departmentid,
       d.name AS departmentName, b.name AS buildingName

FROM uni_class c
INNER JOIN uni_department d ON c.departmentid=d.id
INNER JOIN uni_building b ON c.buildingid=b.id
```

--Add a new class

```
INSERT INTO uni_class (name,starttime,endtime,buildingid,departmentid)
    values ([name of class],
            [start time (currently unused)],
            [end time (currently unused)],
            [associated building id],
            [associated department id]
    );
```

--Get all class names and ids.

```
SELECT id,name FROM uni_class
```

Student:

--Get a specific student

```
SELECT *,
    (SELECT COUNT(id) FROM uni_class_enrollment WHERE studentid=s.id) AS
classCount
    FROM uni_student s
WHERE id=[student id]
```

--Get all students

```
SELECT *,
    (SELECT COUNT(id) FROM uni_class_enrollment WHERE studentid=s.id) AS
classCount
    FROM uni_student s
```

--Add a new student

```
INSERT INTO uni_student (firstname,lastname)
    values(
        [first name},
        [last name]
    );
```

--Update a specific students infromation

```
UPDATE uni_student
    SET firstname=[new student first name],
        lastname=[new student last name]
```

WHERE id=[student id]

--Delete a specific student

DELETE FROM uni_student WHERE id=[student id]

Enrollments:

--Get a specific students enrollments

```
SELECT ce.studentid AS studentid, ce.classid AS classid,  
       s.firstname AS studentFirstName, s.lastname AS studentLastName,  
       c.name AS className, ce.id AS id  
FROM uni_class_enrollment ce  
     INNER JOIN uni_class c ON ce.classid=c.id  
     INNER JOIN uni_student s ON ce.studentid=s.id  
WHERE s.id=[student id]
```

--Get all enrollments

```
SELECT ce.studentid AS studentid, ce.classid AS classid,  
       s.firstname AS studentFirstName, s.lastname AS studentLastName,  
       c.name AS className, ce.id AS id  
FROM uni_class_enrollment ce  
     INNER JOIN uni_class c ON ce.classid=c.id  
     INNER JOIN uni_student s ON ce.studentid=s.id
```

--Add a new enrollment

```
INSERT INTO uni_class_enrollment (studentid,classid)  
    values(  
        [student id],  
        [class id]  
    );
```

--Delete an enrollment by id

DELETE FROM uni_class_enrollment WHERE id=[enrollment id]

--Delete all of a students specific enrollments

DELETE FROM uni_class_enrollment WHERE studentid=[student id]