2020

University XYZ Database

GROUP A

FRANKLIN OVERTON, HENRY DEPEW, BRET RIEDEL, SAMI JENEDI, AUSTIN SWITZER

Table of Contents

Requirements Analysis and Definition

Revision History

Date	Version	Description	Author
9/15/2020	1.0	First Draft	Whole Group
9/29/2020	2.9	Fixing according to comments from submission	Henry Depew
11/10/2020	3.0	Adjusted Entities table	Bret Riedel

Description of Domain and Requirements

The University Database System is a MySQL database to be used by XYZ University. Requirements include searching and manipulating database entries for campuses, clubs, sports, faculty, schools, programs, courses, prerequisites, sections, students, supervisors, lecturers, grades, and committees. The database system should have varying levels of access depending on the permissions of the person accessing it in order to enforce authorization.

Example Queries

- List all the schools are located in 'Toronto Campus', and sort them by school name.
- List all the programs provided by 'science faculty'.
- Give all the names of the lecturers who are the members of the committee and sort by their name.
- List all supervisor's name and the name of the lecturer they manage. Please sort by supervisor name and lecturer name.
- Give all the lecturers who are not the member of the committee.
- Give the total number of courses for each program.
- Give all the lecturers with the courses they are teaching. Sort by lecturer name.
- Give all the course titles and their corresponding prerequisite course titles.
- Give the top 5 courses which have more students involved.
- Give any students and the Prerequisites they need to take based on classes they are enrolled in currently.
- Add student.
- Add lecturer.
- Delete student.
- Delete lecturer

Business Rules

- Each campus has a different name, address, distance to the city center and the only bus running to the campus.
- Each campus has a club.
- Clubs offer one to many sports.
- Each faculty has a name, dean and building.
- Each school belongs to one faculty only and is located on just one campus.
- Every school has a name and a building assigned to it.
- Each program can be offered by only one school.
- Programs exist within a single school.
- Each program has a unique code, title, level and duration.
- Each course belongs to a program.
- Each course has a unique code and course title.
- Courses may have required prerequisites which are also courses.
- Each of the students is enrolled in a single program of study.
- Students receive a grade for each course they take.
- Every student has a unique ID.
- A lecturer is only allowed to work for one school only.
- Each lecturer is assigned an ID which is unique across the whole university.
- A lecturer reports to only one supervisor.
- Supervisors may supervise one or many lecturers.
- A lecturer can teach many courses.
- A course can be taught by one or multiple lecturers.
- Committees are made up of one to many Lecturers.
- The frequency is determined by the faculty involved.

Entities, Attributes and Relationships

Entity	Attributes	Relationship **(translation of the bus. rules)**
CAMPUS	name, address, distance, bus	Contains a CLUB, Contains one or more SCHOOLs
CLUB	name, campus, building, phone number	Offers SPORTs
SPORT	club, name	Offered by CLUB
FACULTY	name, dean, building	Has: SCHOOLs
SCHOOL	name, faculty, campus, building	Offers: PROGRAMs Belongs to: FACULTY Located on: CAMPUS
PROGRAM	code, title, level, duration, school	Comprised of: COURSEs Enrolls: STUDENTs
COURSE	code, program, title, hours	Instructed by: LECTURERs Taken by: STUDENTs
STUDENT	id, name, program, gpa, birthday	Enrolled in a: PROGRAM Takes: COURSEs (in a year)
LECTURER	id, supervisor, title, office_room, school, name	Teaches: COURSEs Works for a: SCHOOL Has a: SUPERVISOR
COMMITTEE	name, faculty, title, frequency	Has members that are: LECTURERs
SECTION	id, lecturer_id, course_code, building, room, time, day, year, semester	Links: STUDENT, LECTURER and COURSE to resolve many to many relationship
PREREQUISITE	course_code, prerequisite_code	Allows: many to many relationship for COURSES
LECTURER_COMMITTEE	lecturer_id, committee_name, faculty	Allows many to many relationship for LECTURER and COMMITTEE
STUDENT_COURSE	section_id, student_id, grade	Links: STUDENT to SECTION

Conceptual Design Model

Revision History

Date	Version	Description	Author
9/29/2020	1.0	First Draft	Whole Group
10/2/2020	1.1	Add figures to the document	Henry Depew
10/4/2020	1.2	Added description for figure 1.1	Franklin Overton
10/5/2020	1.3	Added description for figure 1.2	Bret Riedel
10/5/2020	1.4	Added description for figure 1.3	Sami Jenedi
10/5/2020	1.5	Added description for figure 1.4	Austin Switzer
10/5/2020	1.6	Final tuning of diagrams and uniformity	Whole Group
11/16/2020	1.7	Add resolved ER diagrams	Whole Group

Campus, Club, Sport

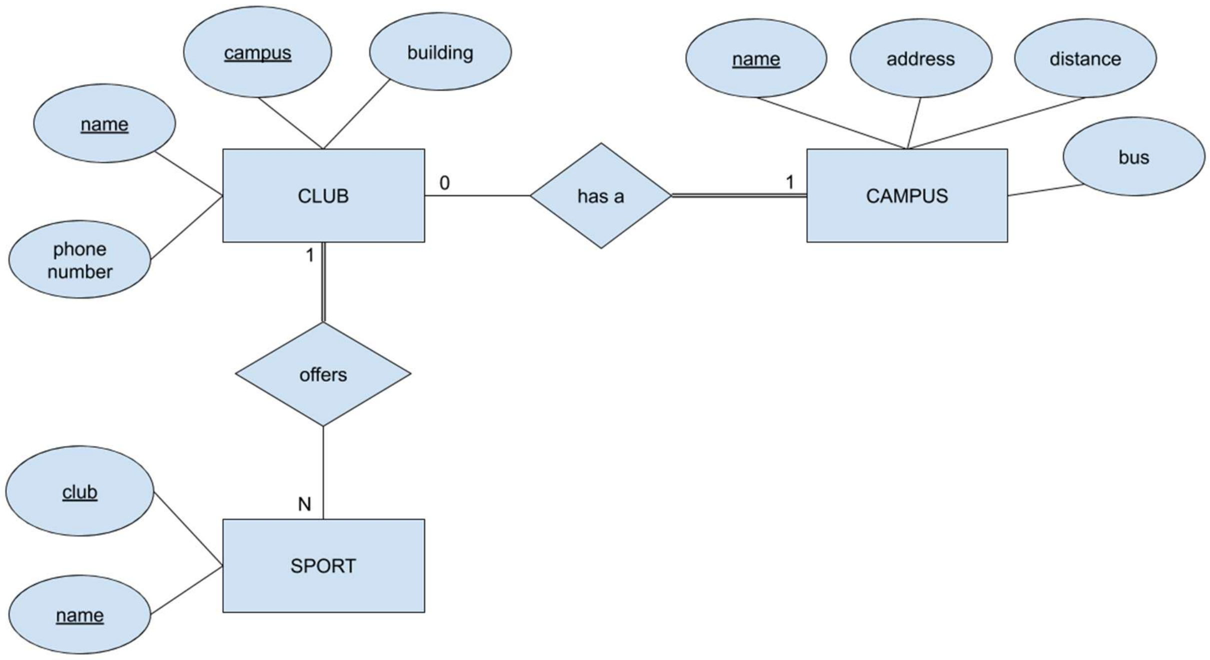


Figure 1.1

The campus has attributes that are the name of the campuses. The name is the primary key to campus. The campus has an address, a distance from the main campus, and it has a bus that goes from one campus to the other. Each campus has a club. There are many clubs. Each club has a building, it is attached to a campus, the club has a name, and each club has a phone number. Its primary keys are a combination of campus and name. Each club offers a sport. There are many sports. Each sport belongs to a club and has a name. Both club and name are a combination of primary keys.

School, Faculty, Program

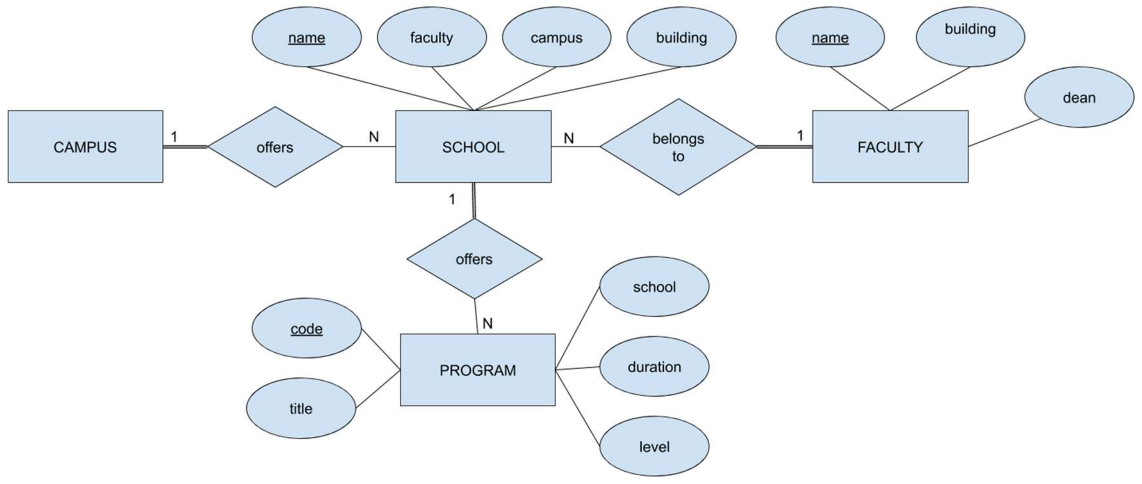


Figure 1.2

Each school can have many programs, but each program can belong to only one school. Each program will have the following attributes: title, school, duration, level, and a unique code that is its primary key. Each faculty can have many schools, but each school can belong to only one faculty. Each faculty will have the following attributes: building, dean, and a unique name that is its primary key. Each school will have the following attributes: faculty, campus, building, and a unique name that is its primary key.

Student, Program, Course

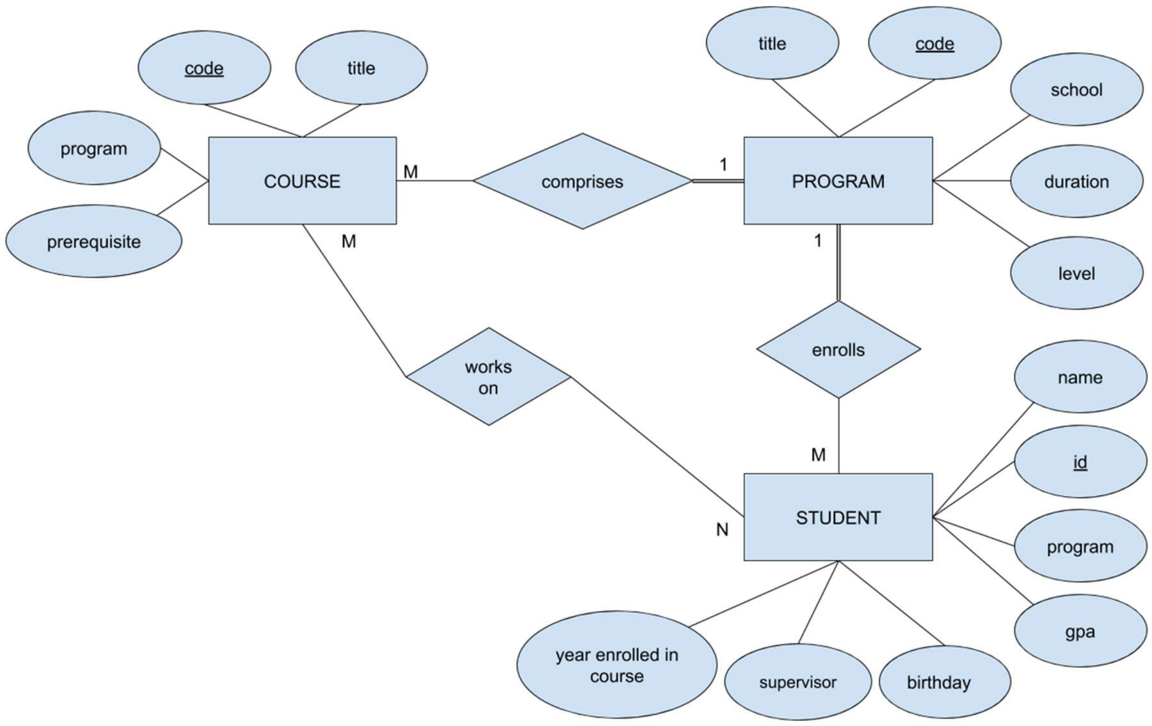


Figure 1.3

Each student will have the following attributes: name, gpa, birthday, supervisor, year enrolled in course, and a unique code that is its primary key. Each student must be enrolled in only one program, but a program can have many students. Each course will have the following attributes: program, prerequisite, title, and a unique code that is its primary key. Each program will have the following attributes: title, school, duration, level, and a unique code as its primary key.

Committee, Lecturer

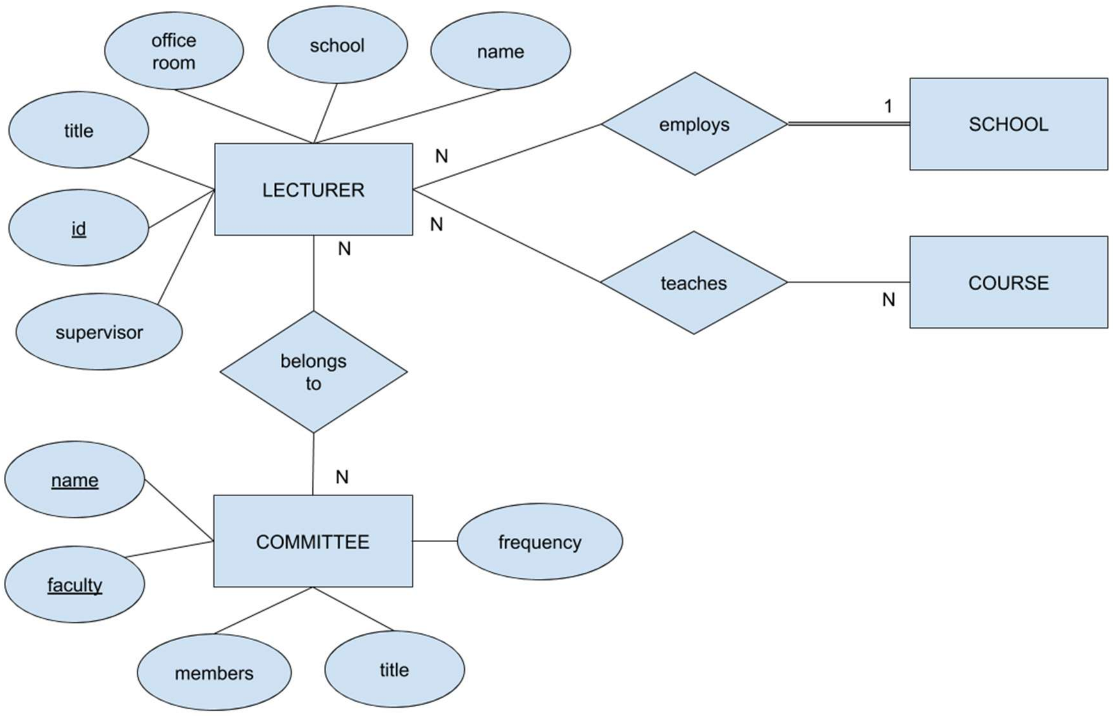


Figure 1.4

Each lecturer has a unique id which acts as its primary key, as well as a name, school, office room, title, and supervisor. A lecturer can only be employed by one school, but a school can employ any number of lecturers. Lecturers belong to any number of committees. Committees have a unique name and faculty, which act in competition as a primary key, in addition to members, a title, and frequency. A committee can be made up of any number of lecturers.

Full Diagram

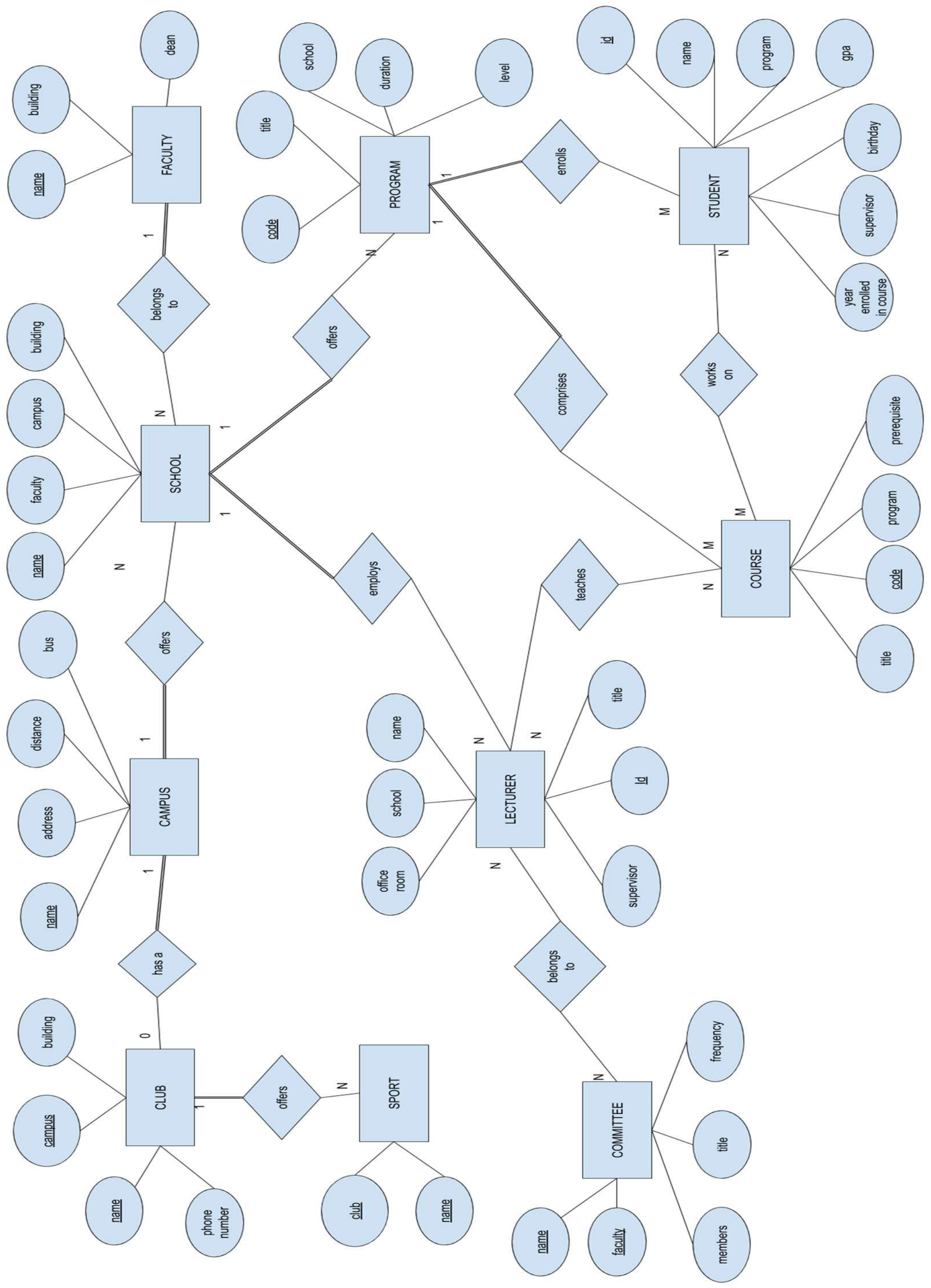


Figure 1.5

This is a composite of figures 1.1 through 1.4.

ER Diagrams After Resolving Many to Many Relationships

Before finalizing the data model, the many to many relationships must be resolved. The following figures demonstrate the addition of four new tables to satisfy these many to many relationships.

CAMPUS, CLUB and SPORT

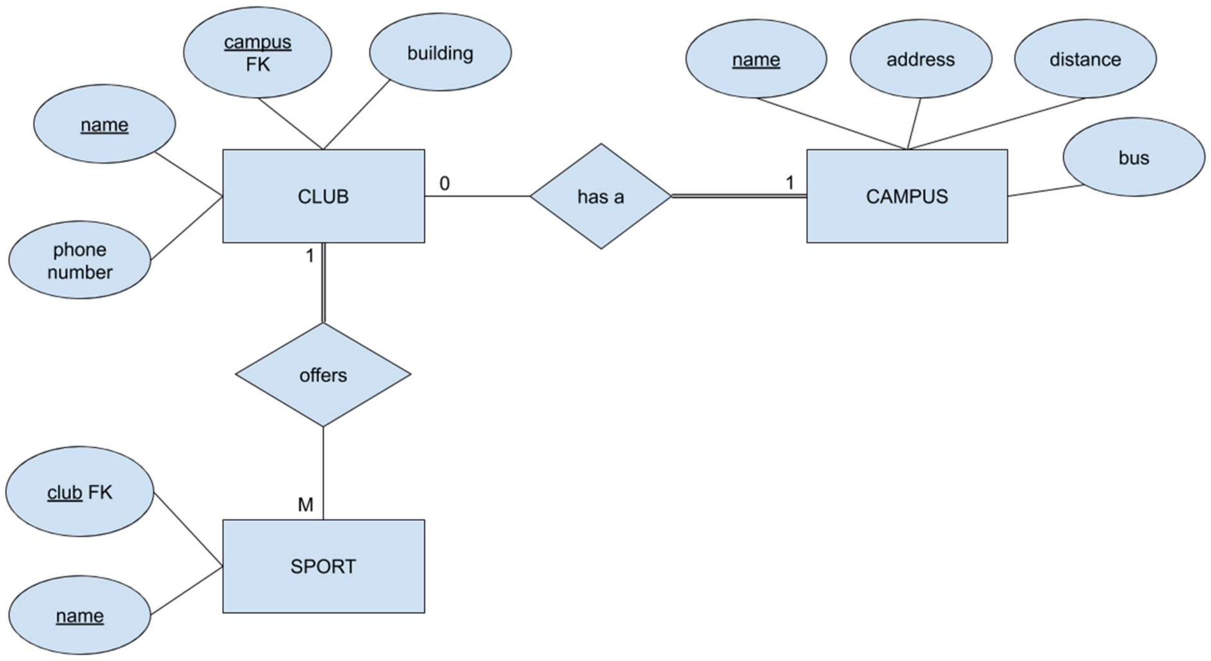


Figure 1.6

This figure updates some attributes from figure 1.1.

SCHOOL, FACULTY and PROGRAM

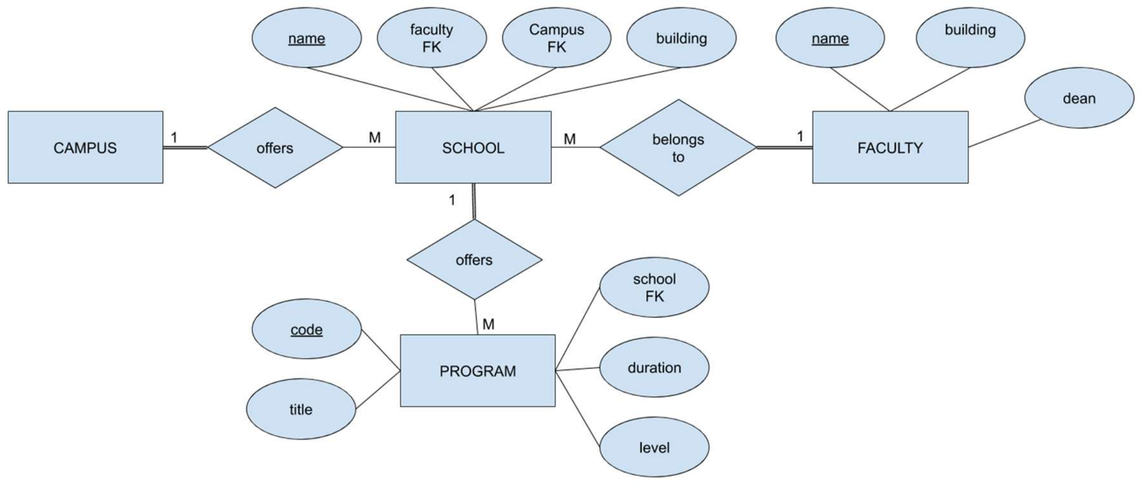


Figure 1.7

This figure updates some attributes from figure 1.2

COURSE, PROGRAM and STUDENT with additions

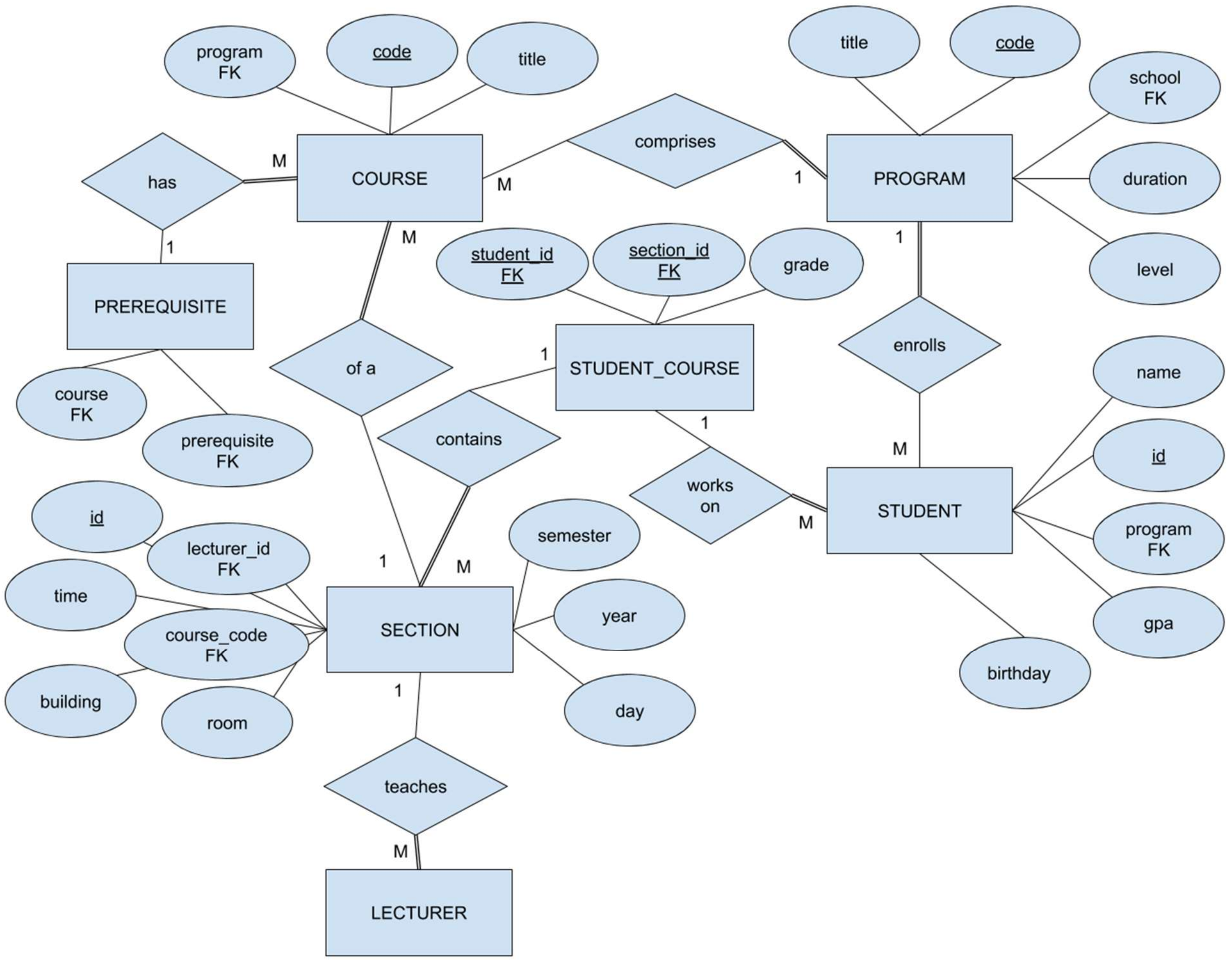


Figure 1.8

This includes new tables SECTION, STUDENT_COURSE and PREREQUISITE onto figure 1.3

COMMITTEE and LECTURER with additions

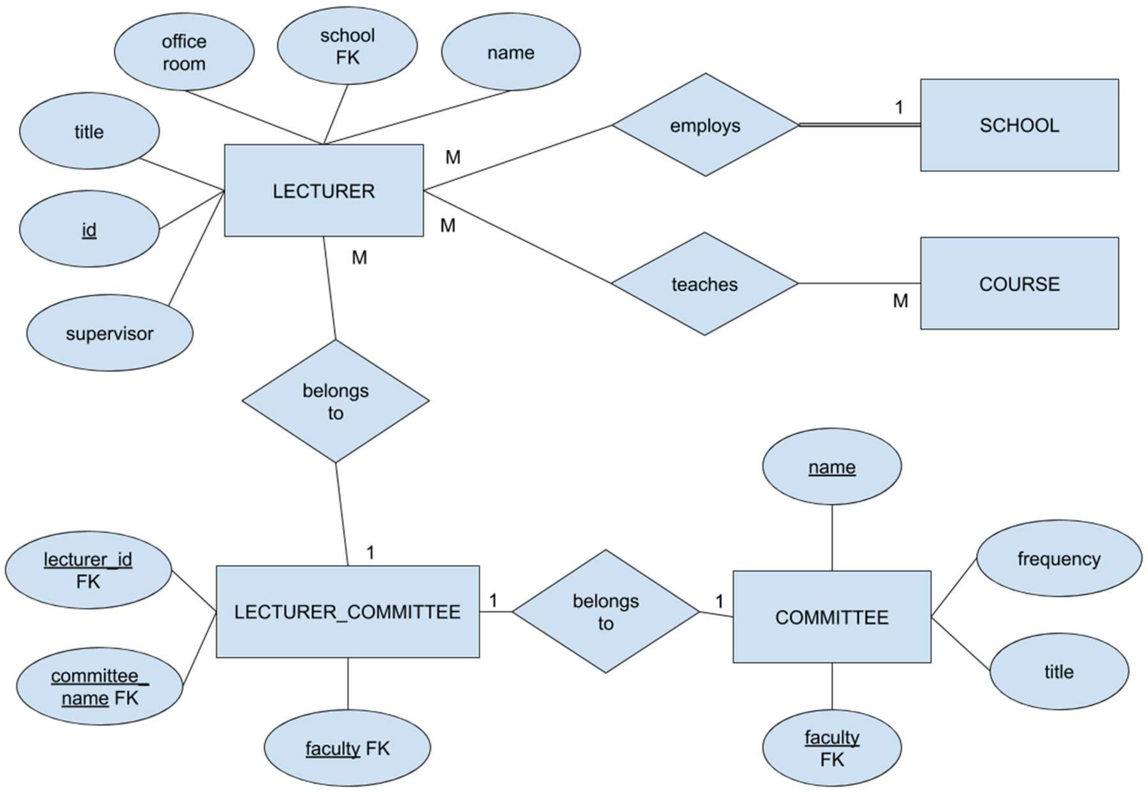


Figure 1.9

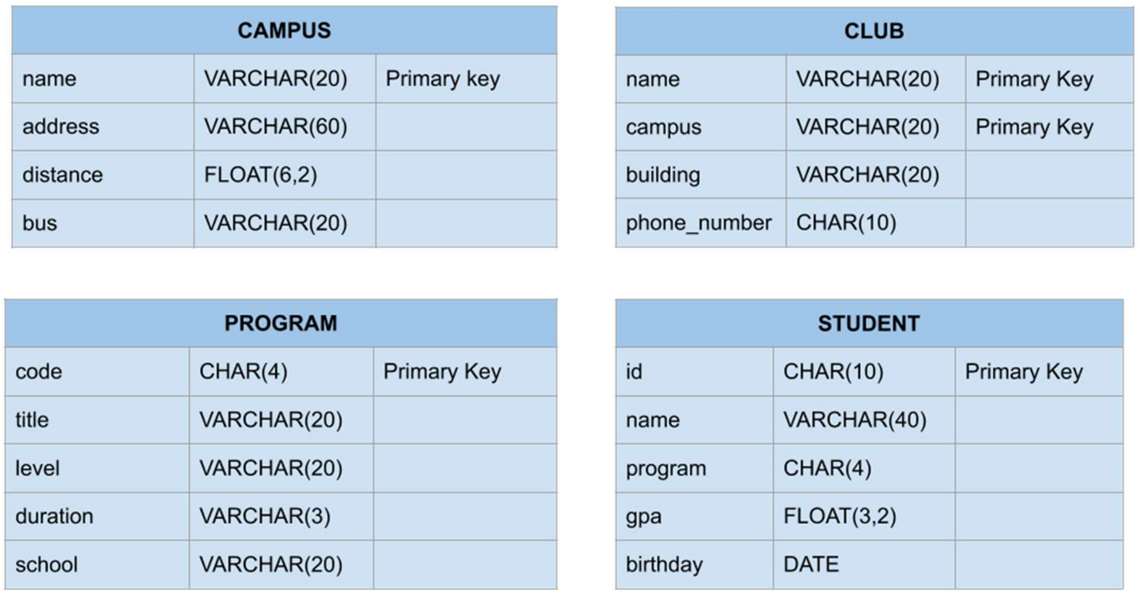
Adds the LECTURER_COMITTEE table to resolve many to many relationships and modifies some attributes from figure 1.4.

Data Model Mapping

Revision History

Date	Version	Description	Author
11/9/2020	1.0	First Draft – placing diagrams in document	Henry Depew
11/16/2020	2.0	Corrections of data as errors were found	Bret Riedel

Campus, Club, Program and Student



CAMPUS		
name	VARCHAR(20)	Primary key
address	VARCHAR(60)	
distance	FLOAT(6,2)	
bus	VARCHAR(20)	

CLUB		
name	VARCHAR(20)	Primary Key
campus	VARCHAR(20)	Primary Key
building	VARCHAR(20)	
phone_number	CHAR(10)	

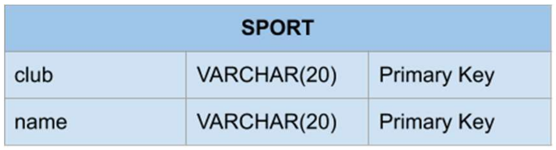
PROGRAM		
code	CHAR(4)	Primary Key
title	VARCHAR(20)	
level	VARCHAR(20)	
duration	VARCHAR(3)	
school	VARCHAR(20)	

STUDENT		
id	CHAR(10)	Primary Key
name	VARCHAR(40)	
program	CHAR(4)	
gpa	FLOAT(3,2)	
birthday	DATE	

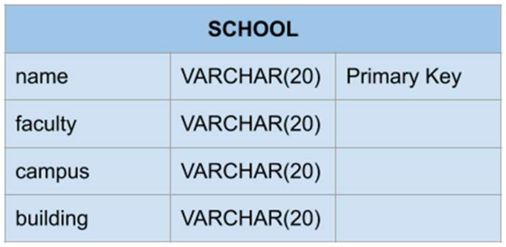
Figure 2.1

The columns and column data types for the CAMPUS, PROGRAM, CLUB and STUDENT tables also specifying the primary key(s).

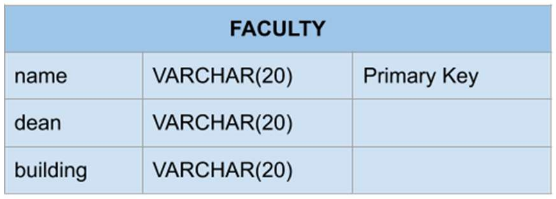
Sport, Faculty, Lecturer, School, Course and Committee



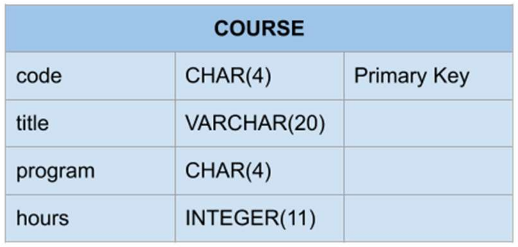
SPORT		
club	VARCHAR(20)	Primary Key
name	VARCHAR(20)	Primary Key



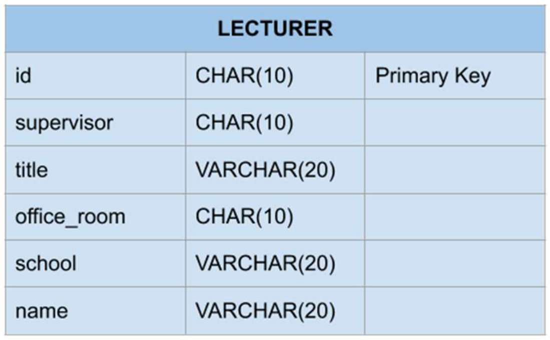
SCHOOL		
name	VARCHAR(20)	Primary Key
faculty	VARCHAR(20)	
campus	VARCHAR(20)	
building	VARCHAR(20)	



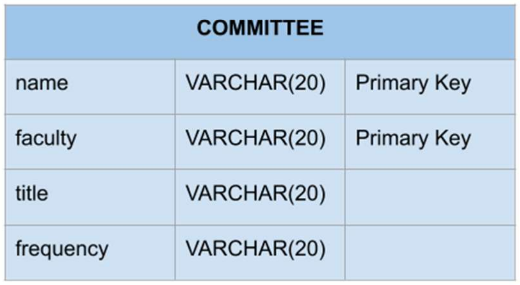
FACULTY		
name	VARCHAR(20)	Primary Key
dean	VARCHAR(20)	
building	VARCHAR(20)	



COURSE		
code	CHAR(4)	Primary Key
title	VARCHAR(20)	
program	CHAR(4)	
hours	INTEGER(11)	



LECTURER		
id	CHAR(10)	Primary Key
supervisor	CHAR(10)	
title	VARCHAR(20)	
office_room	CHAR(10)	
school	VARCHAR(20)	
name	VARCHAR(20)	

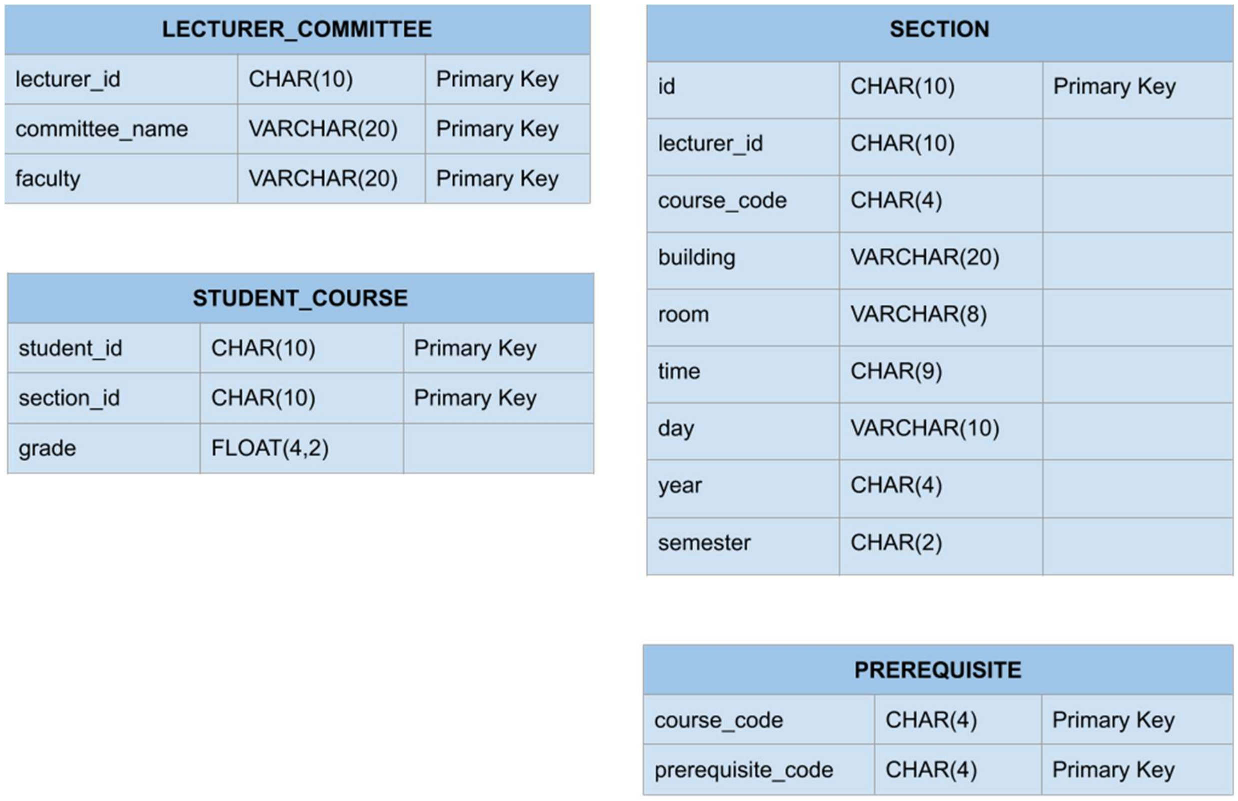


COMMITTEE		
name	VARCHAR(20)	Primary Key
faculty	VARCHAR(20)	Primary Key
title	VARCHAR(20)	
frequency	VARCHAR(20)	

Figure 2.2

The columns and column data types for the SPORT, FACULTY, LECTURER, SCHOOL, COURSE and COMMITTEE tables also specifying the primary key(s).

Lecturer_Committee, Student_Course, Section, Prerequisite



LECTURER_COMMITTEE		
lecturer_id	CHAR(10)	Primary Key
committee_name	VARCHAR(20)	Primary Key
faculty	VARCHAR(20)	Primary Key

STUDENT_COURSE		
student_id	CHAR(10)	Primary Key
section_id	CHAR(10)	Primary Key
grade	FLOAT(4,2)	

SECTION		
id	CHAR(10)	Primary Key
lecturer_id	CHAR(10)	
course_code	CHAR(4)	
building	VARCHAR(20)	
room	VARCHAR(8)	
time	CHAR(9)	
day	VARCHAR(10)	
year	CHAR(4)	
semester	CHAR(2)	

PREREQUISITE		
course_code	CHAR(4)	Primary Key
prerequisite_code	CHAR(4)	Primary Key

Figure 2.3

The columns and column data types for the LECTURER_COMMITTEE, STUDENT_COURSE, SECTION and PREREQUISITE tables also specifying the primary key(s). These tables primarily represent many to many relationships of other tables.

Foreign Key Relationships

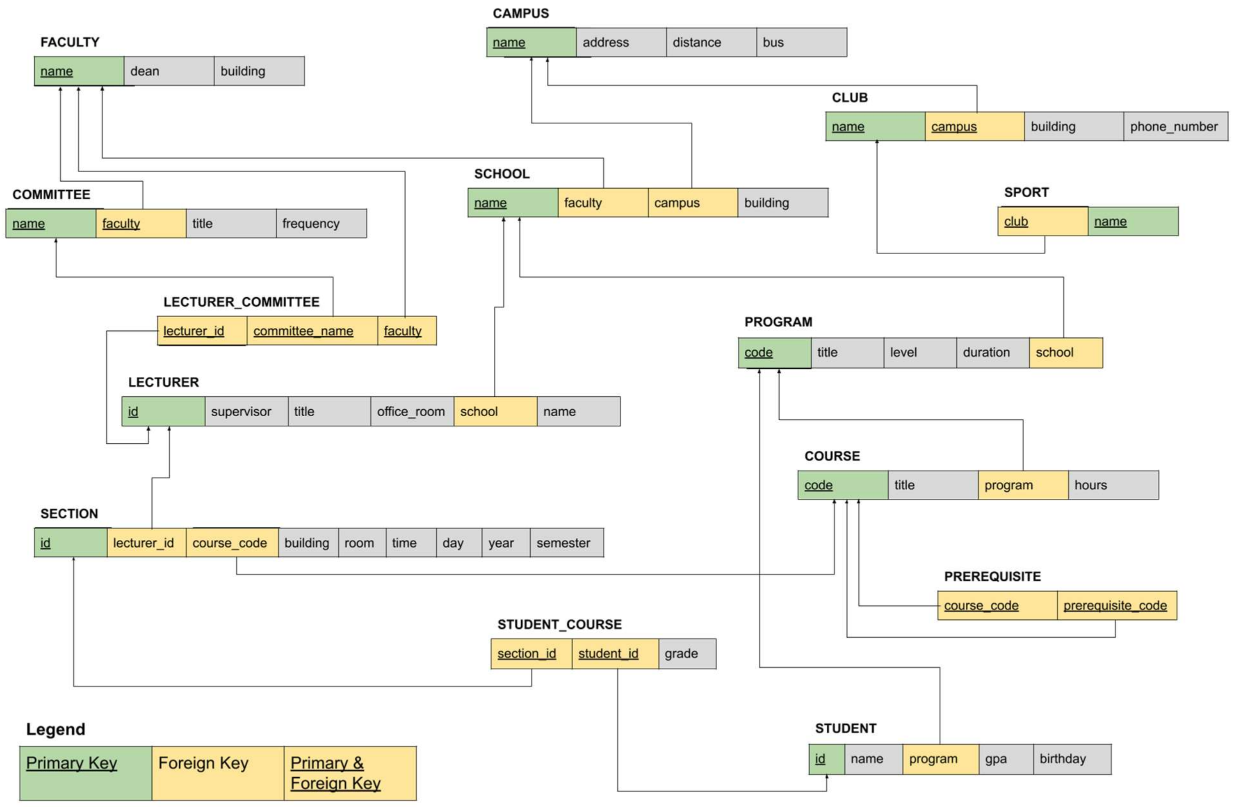
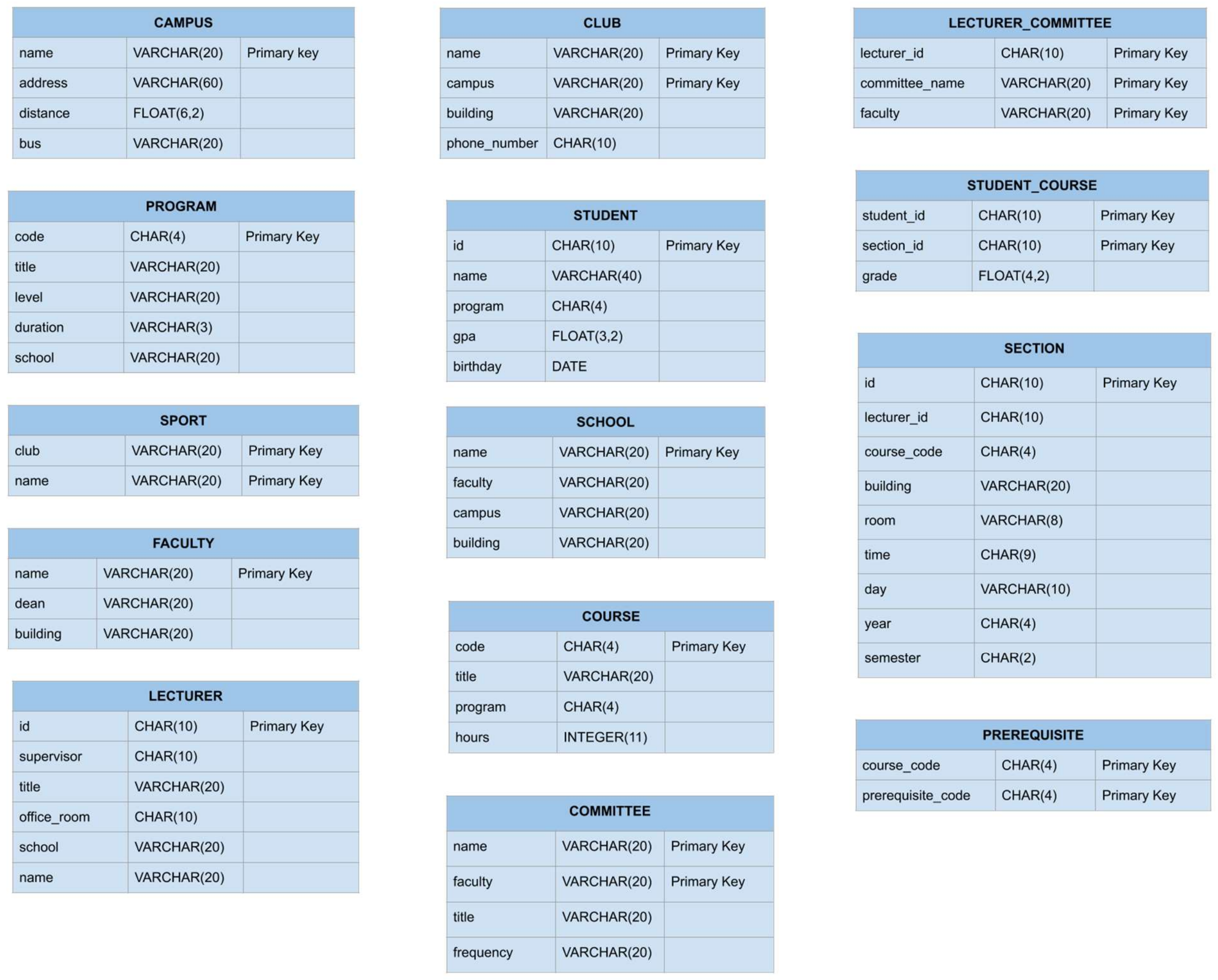


Figure 2.4

Displaying the foreign key relationships between all the tables in the University Database.

Full List of Tables



CAMPUS		
name	VARCHAR(20)	Primary key
address	VARCHAR(60)	
distance	FLOAT(6,2)	
bus	VARCHAR(20)	

PROGRAM		
code	CHAR(4)	Primary Key
title	VARCHAR(20)	
level	VARCHAR(20)	
duration	VARCHAR(3)	
school	VARCHAR(20)	

SPORT		
club	VARCHAR(20)	Primary Key
name	VARCHAR(20)	Primary Key

FACULTY		
name	VARCHAR(20)	Primary Key
dean	VARCHAR(20)	
building	VARCHAR(20)	

LECTURER		
id	CHAR(10)	Primary Key
supervisor	CHAR(10)	
title	VARCHAR(20)	
office_room	CHAR(10)	
school	VARCHAR(20)	
name	VARCHAR(20)	

CLUB		
name	VARCHAR(20)	Primary Key
campus	VARCHAR(20)	Primary Key
building	VARCHAR(20)	
phone_number	CHAR(10)	

STUDENT		
id	CHAR(10)	Primary Key
name	VARCHAR(40)	
program	CHAR(4)	
gpa	FLOAT(3,2)	
birthday	DATE	

SCHOOL		
name	VARCHAR(20)	Primary Key
faculty	VARCHAR(20)	
campus	VARCHAR(20)	
building	VARCHAR(20)	

COURSE		
code	CHAR(4)	Primary Key
title	VARCHAR(20)	
program	CHAR(4)	
hours	INTEGER(11)	

COMMITTEE		
name	VARCHAR(20)	Primary Key
faculty	VARCHAR(20)	Primary Key
title	VARCHAR(20)	
frequency	VARCHAR(20)	

LECTURER_COMMITTEE		
lecturer_id	CHAR(10)	Primary Key
committee_name	VARCHAR(20)	Primary Key
faculty	VARCHAR(20)	Primary Key

STUDENT_COURSE		
student_id	CHAR(10)	Primary Key
section_id	CHAR(10)	Primary Key
grade	FLOAT(4,2)	

SECTION		
id	CHAR(10)	Primary Key
lecturer_id	CHAR(10)	
course_code	CHAR(4)	
building	VARCHAR(20)	
room	VARCHAR(8)	
time	CHAR(9)	
day	VARCHAR(10)	
year	CHAR(4)	
semester	CHAR(2)	

PREREQUISITE		
course_code	CHAR(4)	Primary Key
prerequisite_code	CHAR(4)	Primary Key

Figure 2.5

The full showing of tables and their columns, datatypes and primary keys.

Physical Database

Revision History

Date	Version	Description	Author
11/16/2020	1.0	Adding the groups into the deliverable	Henry Depew

SQL DDL Statements

TABLE CREATION

```
CREATE TABLE FACULTY(
        name VARCHAR(20) NOT NULL,
        dean VARCHAR(20) NOT NULL,
        building VARCHAR(20) NOT NULL,
        PRIMARY KEY (name)
);
CREATE TABLE COMMITTEE(
        name VARCHAR(20) NOT NULL,
        faculty VARCHAR(20) NOT NULL,
        title VARCHAR(20) NOT NULL,
        frequency VARCHAR(20) NOT NULL,
        PRIMARY KEY (name, faculty),
        FOREIGN KEY (faculty) REFERENCES FACULTY(name)
);
```

```
CREATE TABLE SCHOOL(

        name VARCHAR(20) NOT NULL,

        faculty VARCHAR(20) NOT NULL,

        campus VARCHAR(20) NOT NULL,

        building VARCHAR(20) NOT NULL,

        PRIMARY KEY (name)

        FOREIGN KEY (faculty) REFERENCES FACULTY(name),

        FOREIGN KEY (campus) REFERENCES CAMPUS(name)

);

CREATE TABLE CAMPUS (

        name VARCHAR(20) NOT NULL,

        address VARCHAR(60) NOT NULL,

        distance FLOAT(6,2) NOT NULL,

        bus VARCHAR(20) NOT NULL,

        PRIMARY KEY (name)

);

CREATE TABLE CLUB (

        name VARCHAR(20) NOT NULL,

        campus VARCHAR(20) NOT NULL,

        building VARCHAR(20) NOT NULL,

        phone_number CHAR(10) NOT NULL,

        PRIMARY KEY (name, campus),

        FOREIGN KEY (campus) REFERENCES CAMPUS(name)

);
```

```
CREATE TABLE PROGRAM (
        code CHAR(4) NOT NULL,
        title VARCHAR(20) NOT NULL,
        level VARCHAR(20) NOT NULL,
        duration VARCHAR(3) NOT NULL,
        school VARCHAR(20) NOT NULL,
        PRIMARY KEY (code),
        FOREIGN KEY (school) REFERENCES SCHOOL(name)
);
CREATE TABLE STUDENT (
        id CHAR(10) NOT NULL,
        name VARCHAR(40) NOT NULL,
        program CHAR(4) NOT NULL,
        gpa FLOAT(3,2) NOT NULL,
        birthday DATE NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (program) REFERENCES PROGRAM(code)
);
CREATE TABLE SPORT (
        club VARCHAR(20) NOT NULL,
        name VARCHAR(20) NOT NULL,
        PRIMARY KEY (club, name),
        FOREIGN KEY (club) REFERENCES CLUB(name)
);
```

```
CREATE TABLE COURSE (
        code CHAR(4) NOT NULL,
        title VARCHAR(20) NOT NULL,
        program CHAR(4) NOT NULL,
        hours INTEGER NOT NULL,
        PRIMARY KEY (code),
        FOREIGN KEY (program) REFERENCES PROGRAM(code)
);

CREATE TABLE LECTURER (
        id CHAR(10) NOT NULL,
        supervisor CHAR(10) NOT NULL,
        title VARCHAR(20) NOT NULL,
        office_room CHAR(10) NOT NULL,
        school VARCHAR(20) NOT NULL,
        name VARCHAR(20) NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (school) REFERENCES SCHOOL(name)
);

CREATE TABLE LECTURER_COMMITTEE (
        lecturer_id CHAR(10) NOT NULL,
        committee_name VARCHAR(20) NOT NULL,
        faculty VARCHAR(20) NOT NULL,
        PRIMARY KEY (lecturer_id, committee_name, faculty),
        FOREIGN KEY (lecturer_id) REFERENCES LECTURER(id),
        FOREIGN KEY (committee_name) REFERENCES COMMITTEE(name),
        FOREIGN KEY (faculty) REFERENCES FACULTY(name)
);
```

```
CREATE TABLE STUDENT_COURSE (

        student_id CHAR(10) NOT NULL,

        section_id CHAR(10) NOT NULL,

        grade FLOAT (4,2) NOT NULL,

        PRIMARY KEY (student_id, section_id),

        FOREIGN KEY (section_id) REFERENCES SECTION(id),

        FOREIGN KEY (student_id) REFERENCES STUDENT(id)

);

CREATE TABLE SECTION (

        id CHAR(10) NOT NULL,

        lecturer_id CHAR(10) NOT NULL,

        course_code CHAR(4) NOT NULL,

        building VARCHAR(20) NOT NULL,

        room VARCHAR(8) NOT NULL,

        time CHAR(7) NOT NULL,

        day VARCHAR(10) NOT NULL,

        year CHAR(4)NOT NULL,

        semester CHAR(2) NOT NULL,

        PRIMARY KEY (id),

        FOREIGN KEY (lecturer_id) REFERENCES LECTURER(id),

        FOREIGN KEY (course_code) REFERENCES COURSE(code)

);

CREATE TABLE PREREQUISITE (

        course_code CHAR(4) NOT NULL,

        prerequisite_code CHAR(4) NOT NULL,

        PRIMARY KEY (course_code, prerequisite_code),

        FOREIGN KEY (course_code) REFERENCES COURSE(code),

        FOREIGN KEY (prerequisite_code) REFERENCES COURSE(code)

);
```

INSERT STATEMENTS

FACULTY

```
INSERT INTO FACULTY VALUES ("Science", "Teresa Small", "Moon");

INSERT INTO FACULTY VALUES ("Humanities", "Carl Sagan", "Dunkin");
```

CAMPUS

```
INSERT INTO CAMPUS VALUES ("Toronto", "1 Main St", 4, "Main Line");

INSERT INTO CAMPUS VALUES ("Vancouver", "54 First St", 3.7, "Circulator");
```

SCHOOL

```
INSERT INTO SCHOOL VALUES ("Natural Science", "Science", "Toronto", "Eaton");

INSERT INTO SCHOOL VALUES ("Writing", "Humanities", "Vancouver", "Fraser");

INSERT INTO SCHOOL VALUES ("Engineering", "Science", "Toronto", "Mallott");
```

CLUB

```
INSERT INTO CLUB VALUES (“Top”, “Toronto”, “Blasian”, “555-1234”);

INSERT INTO CLUB VALUES (“Varsity”, “Vancouver”, “Corder”, “555-5678”);
```

SPORT

```
INSERT INTO SPORT VALUES (“Top”, “Rowing”);

INSERT IINTO SPORT VALUES (“Varsity”, “Football”);

INSERT INTO SPORT VALUES (“Varsity”, “Soccer”);
```

COMMITTEE

```
INSERT INTO COMMITTEE VALUES (“Advisory”, ”Science”,  “Important”, “Bimonthly”);

INSERT INTO COMMITTEE VALUES (“Advisory”, ”Humanities”, “Important”,
“Bimonthly”);

INSERT INTO COMMITTEE VALUES (“Health and Safety”, ”Humanities”,  “Important”,
“Annually”);
```

LECTURER

```
INSERT INTO LECTURER VALUES (“1231231231”, “9879879879”, “Professor”, “340”,
“Natural Science”, “Chad Warden”);

INSERT INTO LECTURER VALUES (“9879879879”, NULL, “Professor”, “112”, “Writing”,
“Anthony Perez”);

INSERT INTO LECTURER VALUES (“4564564564”, “9879879879”, “Professor”, “220”,
“Engineering”, “Bob Roberts”);
```

PROGRAM

```
INSERT INTO PROGRAM VALUES (“PHSX”, “Physics”, “Undergrad”, “130”, “Natural
Science”);

INSERT INTO PROGRAM VALUES (“ENGL”, “English”, “Undergrad”, “125”, “Writing”);

INSERT INTO PROGRAM VALUES (“MECH”, “Mech Engineering”, “Undergrad”, “130”,
“Engineering”);
```

COURSE

```
INSERT INTO COURSE VALUES (“P210”, “Intermediate Physics”, “PHSX”, 3);

INSERT INTO COURSE VALUES (“M130”, “Basics”, “MECH”, 4);

INSERT INTO COURSE VALUES (“E100”, “Intro to English”, “ENGL”, 3);

INSERT INTO COURSE VALUES (“P100”, “Intro to Physics”, “PHSX”, 3);

INSERT INTO COURSE VALUES (“E200”, “Poetry”, “ENGL”, 3);
```

STUDENT

```
INSERT INTO STUDENT VALUES (“1324567890”, “Bob Smith”, “PHSX”, “3.71”, ”1980-01-
01”);

INSERT INTO STUDENT VALUES (“4445556666”, “Jane Doe”, “MECH”, “1.88”, “2000-10-
15”);

INSERT INTO STUDENT VALUES (“7778889999”, “Burt Mackin”, “PHSX”, “2.50”, “1984-
12-01”);

INSERT INTO STUDENT VALUES (“3338884444”, “Rob Roy”, “ENGL”, “4.00”, “1988-09-
24”);

INSERT INTO STUDENT VALUES (“4564564567”, “Jimmy Buffet”, “MECH”, “3.10”, “1991-
11-05”);
```

SECTION

INSERT INTO SECTION VALUES (“9876543210”, “1231231231”, “P100”, “Fraser”, “154”, “1410-1500”, “MWF”, “2019”, “FA”);

INSERT INTO SECTION VALUES (“6668887777”, “9879879879”, “E200”, “Fraser”, “387”, “800-915”, “TuTh”, “2020”, “SP”);

INSERT INTO SECTION VALUES (“3332224444”, “4564564564”, “M130”, “Mallott”, “215”, “1330-1500”, “MW”, “2020”, “SU”);

STUDENT_COURSE

INSERT INTO STUDENT_COURSE VALUES (“9876543210”, “1324567890”, “92.74”);

INSERT INTO STUDENT_COURSE VALUES (“6668887777”, “4445556666”, “84.00”);

INSERT INTO STUDENT_COURSE VALUES (“3332224444”, “7778889999”, “64.55”);

LECTURER_COMMITTEE

INSERT INTO LECTURER_COMMITTEE VALUES ("1231231231", "Advisory", "Science");

INSERT INTO LECTURER_COMMITTEE VALUES ("9879879879", "Advisory", "Humanities");

INSERT INTO LECTURER_COMMITTEE VALUES ("4564564564", "Advisory", "Science");

INSERT INTO LECTURER_COMMITTEE VALUES ("1231231231", "Health and Safety", "Science");

PREREQUISITE

INSERT INTO PREREQUISITE VALUES ("P210", "P100");

INSERT INTO PREREQUISITE VALUES ("E200", "E100");

Sample Queries

1) List all the schools are located in 'Toronto Campus', and sort them by school name.

```
SELECT name as "School Name"

FROM SCHOOL

Where campus LIKE '%Toronto% '

ORDER BY name;
```

2) List all the programs provided by 'science faculty'.

```
SELECT p.title as "Program Title"

FROM PROGRAM p

INNER JOIN SCHOOL s

        ON p.school = s.name

WHERE s.faculty LIKE 'Science';
```

3) Give all the names of the lecturers who are the members of the committee and sort by their name.

```
SELECT name as Member

FROM LECTURER l

INNER JOIN LECTURER_COMMITTEE lc

ON l.id = lc.lecturer_id

WHERE  lc.committee_name LIKE 'Advisory' && lc.faculty LIKE 'Science'

ORDER BY l.name;
```

4) List all supervisor's name and the name of the lecturer they manage. Please sort by supervisor name and lecturer name.

```
SELECT t1.name as Supervisor, t2.name as Subordinate

FROM LECTURER t1

JOIN LECTURER t2

ON t1.id = t2.supervisor;
```

5) Give all the lecturers who are not the member of the committee.

```
SELECT name as “Lecturer Name”

FROM LECTURER t1

LEFT JOIN LECTURER_COMMITTEE t2

ON t1.id = t2.lecturer_id

WHERE t2.lecturer_id IS NULL;
```

6) Give the total number of courses for each program.

```
SELECT program as “Program Code”, COUNT(code) as “Total Courses”

FROM COURSE

GROUP BY program;
```

7) Give all the lecturers with the courses they are teaching. Sort by lecturer name.

```
SELECT name as “Lecturer Name”, s.course_code as “Course Code”

FROM LECTURER l

INNER JOIN SECTION s

On l.id = s.lecturer_id

WHERE l.id = s.lecturer_id

ORDER BY l.name;
```

8) Give all the course titles and their corresponding prerequisite course titles.

```
SELECT c.title as “Course Title”,  c2.title as “Prerequisite Title”

FROM COURSE c

LEFT JOIN PREREQUISITE pr

ON c.code = pr.course_code

LEFT JOIN COURSE c2

ON pr.prerequisite_code = c2.code;
```

9) Give the top 5 courses which have more students involved.

```
SELECT c. title as “Course Title”, s.course_code as “Course Code”, COUNT(sc.section_id) as
“Student Count”

FROM SECTION s

INNER JOIN STUDENT_COURSE sc

ON s.id = sc.section_id

INNER JOIN COURSE c

ON c.code = s.course_code

GROUP BY s.course_code

ORDER BY COUNT(sc.section_id) DESC

LIMIT 5;
```

10) Give any students and the Prerequisites they need to take based on classes they are enrolled in currently.

```
SELECT s.name as Name, pr.prerequisite_code as ‘Missing Class’, pr.course_code as ‘Required
by’

FROM STUDENT s

INNER JOIN STUDENT_COURSE sc

ON s.id = sc.student_id

INNER JOIN SECTION sn

ON sn.id = sc.section_id

INNER JOIN PREREQUISITE pr

ON sn.course_code = pr.course_code

WHERE (s.id, pr.prerequisite_code) NOT IN

   (SELECT sc.student_id, sn.course_code FROM SECTION sn

   INNER JOIN STUDENT_COURSE sc

   ON sc.section_id = sn.id);
```

11) Add a Lecturer.

```
INSERT INTO LECTURER VALUES (“<id>”, “<supervisor_id>”, “<title>”, “<office_room>”,
“<school>”, “<name>”);
```

12) Update Lecturer with respect to school.

```
UPDATE LECTURER SET school = <school name> WHERE id = “<id>”;
```

13) Add Section that the new lecturer is teaching

```
INSERT INTO SECTION VALUES (“<id>”, “<lecturer_id>”, “<course_code>”, “<building>”,
“<room>”, “<time>”, “<day>”, “<year>”, “<semester>”);
```

14) Delete a Lecturer.

```
DELETE FROM LECTURER WHERE id = ‘<lecturer id>’;
```