CSCI 2141 - Intro to Database Systems Group Project

Jayden Taylor, Megan Brock, Khang Tran, Emma Travers

1: Summary

Our group created a Student Records database which contains data about students and their information and relationships within a university. The tables within the database are: Department, Course, Professor, Student, Textbook, and Enrolment. These are all necessary information someone would use to store and manipulate data regarding student records in a university. We created the front-end application using the Java Workbench. The application allows the user to view a menu of the university, which lets them view various information regarding the student records within the university. It not only allows them to view the database's information, but they can edit data by adding, deleting, and manipulating information within the database as well. We created a stored view and procedure as a bonus which allows the user to have more administration over the students information. The stored view lets the user view the fees a particular student has, and the stored procedure allows the user to view all the students who are enrolled in a given course. Overall, our database and database application has all of the necessary information and relations in order for a user to effectively manage the student records for a university.

2: Requirements & Business Rules

For our Student Record database, a university can have many different departments. A department can only have one university, because it would not be logical or effective to have a singular department shared by multiple universities. A department can also have many courses within it. A course can belong to one or more departments, and a course only has one professor and many students. A professor can teach many courses, and a student can also take many courses. A course can also have one or more textbooks for the course, however a textbook can only belong to one course, textbooks are not shared over multiple courses.

3: Conceptual Data Model

Below in **Figure 1** is the conceptual data model for our Student Record database, showing the keys and table relations within the database.

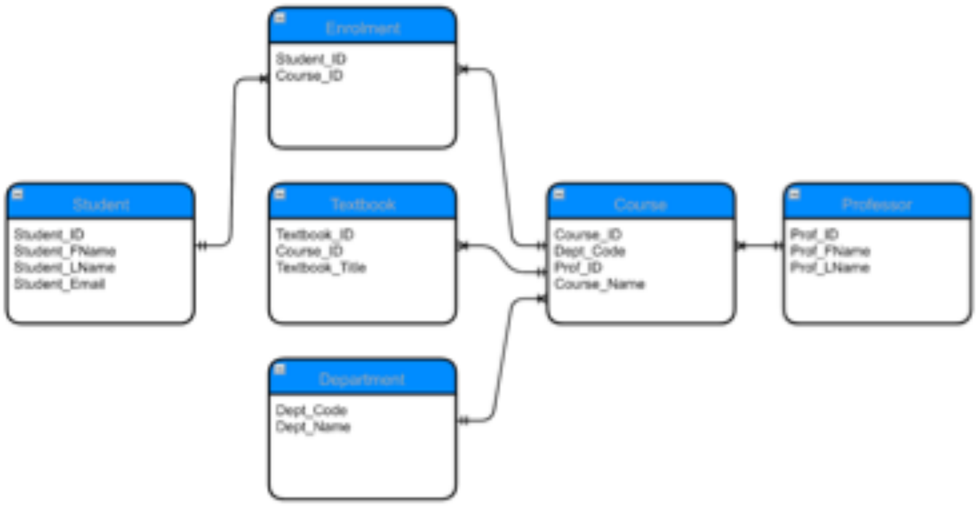

Figure 1: Conceptual Data Model

4: Physical Data Model

Below in **Figure 2** is the physical data model for our Student Record database, created by reverse engineering from MySQL Workbench. Along with the tables and their relations with one another, it includes the bonus view and procedure we created.

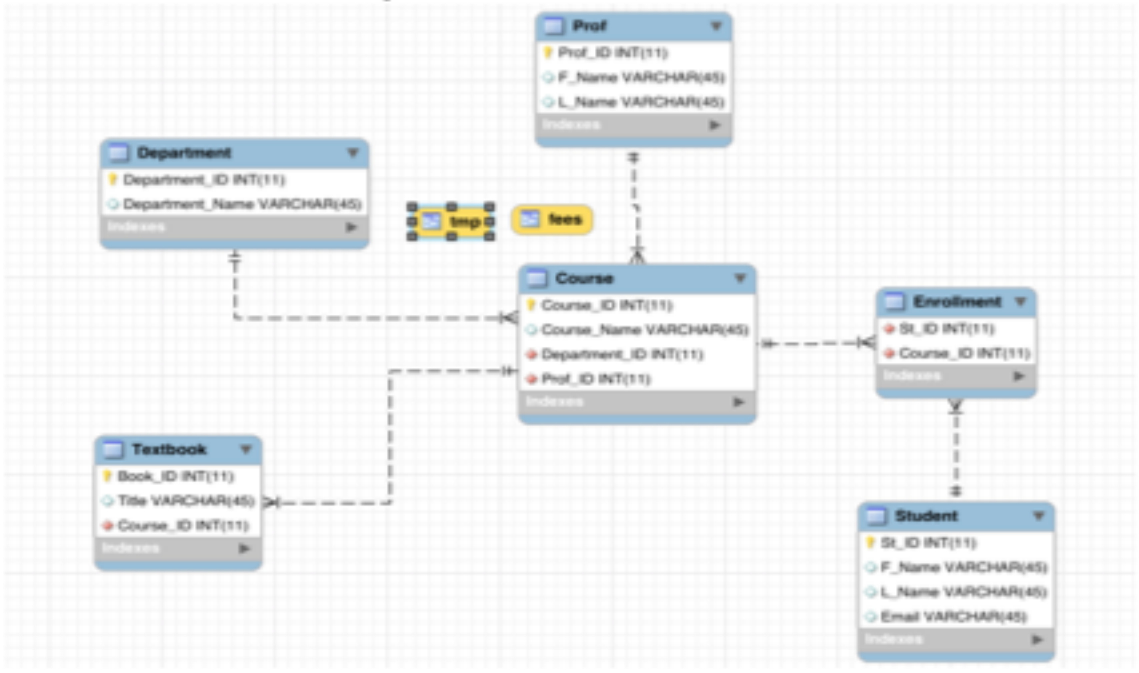

Figure 2: Physical Data Model

5: SQL Scripts

The following SQL scripts are shown in their separate MySQL files, however in the files we provided in our ZIP file the sql file StudentRecordComplete.sql contains all of the queries below in one file.

5.1: Database Creation

Below is the code which creates the Student Record database. It contains all of the table and key creation SQL statements which create the relations within our Student Record Database.

```
CREATE DATABASE IF NOT EXISTS StudentRecord;
USE StudentRecord;

CREATE TABLE IF NOT EXISTS Student (
 `St_ID` INT NOT NULL,
 `F_Name` VARCHAR(45) NULL,
 `L_Name` VARCHAR(45) NULL,
 `Email` VARCHAR(45) NULL,
 PRIMARY KEY (`St_ID`)
);

CREATE TABLE IF NOT EXISTS Prof (
 `Prof_ID` INT NOT NULL,
 `F_Name` VARCHAR(45) NULL,
 `L_Name` VARCHAR(45) NULL,
 PRIMARY KEY (`Prof_ID`)
);

CREATE TABLE IF NOT EXISTS Department (
 `Department_ID` INT NOT NULL,
 `Department_Name` VARCHAR(45) NULL,
 PRIMARY KEY (`Department_ID`)
);

CREATE TABLE IF NOT EXISTS Course (
 `Course_ID` INT NOT NULL,
```

```
 `Course_Name` VARCHAR(45) NULL,
 `Department_ID` INT NOT NULL,
 `Prof_ID` INT NOT NULL,
 PRIMARY KEY (`Course_ID`),
 FOREIGN KEY (`Department_ID`) REFERENCES Department(`Department_ID`),
 FOREIGN KEY (`Prof_ID`) REFERENCES Prof(`Prof_ID`)
);

CREATE TABLE IF NOT EXISTS Enrollment (
 `St_ID` INT NOT NULL,
 `Course_ID` INT NOT NULL,
 FOREIGN KEY (`Course_ID`) REFERENCES Course(`Course_ID`),
 FOREIGN KEY (`St_ID`) REFERENCES Student(`St_ID`)
);

CREATE TABLE IF NOT EXISTS Textbook (
 `Book_ID` INT NOT NULL,
 `Title` VARCHAR(45) NULL,
 `Course_ID` INT NOT NULL,
 PRIMARY KEY (`Book_ID`),
 FOREIGN KEY (`Course_ID`) REFERENCES Course(`Course_ID`)
);
```

5.2: Database Population

The SQL statements below are all of the insert statements which insert sample information about the university, courses, and students into the various tables, for us to use with the front-end application.

```
CREATE DATABASE IF NOT EXISTS StudentRecord;
USE StudentRecord;

INSERT INTO Student VALUES(1001,"John","Smith","Jsmith@email.com");
INSERT INTO Student VALUES(1002,"Selena","Lowry","Selena.Lowry@email.com");
INSERT INTO Student VALUES(1003,"Talia","Harris","TaliaH@email.com");
INSERT INTO Student VALUES(1004,"Roy","Grimes","Roy_G@email.com");
```

```
INSERT INTO Student VALUES(1005,"Abigail","Brook","AbigailB@email.com");
INSERT INTO Student VALUES(1006,"Abdi","Kaur","Akaur@email.com");
INSERT INTO Student VALUES(1007,"Mina","Sharp","Msharp@email.com");
INSERT INTO Student VALUES(1007,"Mina","Sharp","Msharp@email.com");
INSERT INTO Student VALUES(1008,"Grady","Phelps","GHPhelps@email.com");
INSERT INTO Student VALUES(1009,"Reece","Anderson","ReeceA@email.com");
INSERT INTO Student
VALUES(1010,"Brendon","Mcfarland","Brendon_Mcfarland@email.com");
INSERT INTO Student VALUES(1011,"Daniel","Vinson","Dvinson@email.com");
INSERT INTO Student VALUES(1012,"Alan","Tierney","Atierney@email.com");
INSERT INTO Student VALUES(1013,"Nicky","Shepard","NickyS@email.com");
INSERT INTO Student VALUES(1014,"Juno","Mccray","Juno_M@email.com");
INSERT INTO Student VALUES(1015,"Maddy","Page","MaddyPage@email.com");
INSERT INTO Student VALUES(1016,"Garin","Gray","Ggray@email.com");
INSERT INTO Student VALUES(1017,"Hadley","Guerra","Hguerra@email.com");
INSERT INTO Student VALUES(1018,"David","Winters","D_Winters@email.com");
INSERT INTO Student VALUES(1019,"Dana","Jordan","DanaJ@email.com");
INSERT INTO Student VALUES(1020,"Reid","Henry","Reid_H@email.com");

INSERT INTO Prof VALUES(2001,"Jane","Brown");
INSERT INTO Prof VALUES(2002,"Mike","Jones");
INSERT INTO Prof VALUES(2003,"Mary","Matthews");
INSERT INTO Prof VALUES(2004,"Andrew","Kane");
INSERT INTO Prof VALUES(2005,"George","Jackson");
INSERT INTO Prof VALUES(2006,"Ben","Merritt");
INSERT INTO Prof VALUES(2007,"Reggie","Tanner");
INSERT INTO Prof VALUES(2008,"Aaron","Garner");
INSERT INTO Prof VALUES(2009,"Carter","Fraser");
INSERT INTO Prof VALUES(2010,"Elliott","Terry");

INSERT INTO Department VALUES(4001,"Mathematics");
INSERT INTO Department VALUES(4002,"Computer Science");
INSERT INTO Department VALUES(4003,"English");
INSERT INTO Department VALUES(4004,"Chemistry");
INSERT INTO Department VALUES(4005,"Arts");

INSERT INTO Course VALUES(3001,"Calculus I",4001,2001);
INSERT INTO Course VALUES(3002,"Calculus II",4001,2001);
```

```
INSERT INTO Course VALUES(3016,"Programming for Begginers",4002,2004);
INSERT INTO Course VALUES(3003,"Algorithms",4002,2004);
INSERT INTO Course VALUES(3004,"Creative Writing",4003,2007);
INSERT INTO Course VALUES(3005,"Intensive Grammar",4003,2007);
INSERT INTO Course VALUES(3006,"Organic Chemistry",4004,2008);
INSERT INTO Course VALUES(3006,"Organic Chemistry",4004,2008);
INSERT INTO Course VALUES(3007,"Biochemistry",4004,2008);
INSERT INTO Course VALUES(3008,"Introduction to Databases",4002,2006);
INSERT INTO Course VALUES(3009,"History of Painting",4005,2009);
INSERT INTO Course VALUES(3010,"Music",4005,2010);
INSERT INTO Course VALUES(3011,"Fundamentals of Drawing",4005,2009);
INSERT INTO Course VALUES(3012,"Linear Algebra",4001,2002);
INSERT INTO Course VALUES(3013,"Data Structures",4002,2005);
INSERT INTO Course VALUES(3014,"Modern Poetry",4003,2007);
INSERT INTO Course VALUES(3015,"Geometry",4001,2003);

INSERT INTO Textbook VALUES(5001,"Integral Calculus",3002);
INSERT INTO Textbook VALUES(5002,"Basics of Java",3003);
INSERT INTO Textbook VALUES(5003,"Linear Algebra",3013);
INSERT INTO Textbook VALUES(5004,"Sheet Music",3011);

INSERT INTO Enrollment VALUES(1001,3001);
INSERT INTO Enrollment VALUES(1001,3008);
INSERT INTO Enrollment VALUES(1001,3003);
INSERT INTO Enrollment VALUES(1002,3001);
INSERT INTO Enrollment VALUES(1003,3012);
INSERT INTO Enrollment VALUES(1003,3015);
INSERT INTO Enrollment VALUES(1004,3009);
INSERT INTO Enrollment VALUES(1004,3014);
INSERT INTO Enrollment VALUES(1005,3005);
INSERT INTO Enrollment VALUES(1005,3012);
INSERT INTO Enrollment VALUES(1005,3015);
INSERT INTO Enrollment VALUES(1006,3006);
INSERT INTO Enrollment VALUES(1006,3007);
INSERT INTO Enrollment VALUES(1007,3009);
INSERT INTO Enrollment VALUES(1008,3011);
INSERT INTO Enrollment VALUES(1009,3004);
INSERT INTO Enrollment VALUES(1009,3005);
```

```
INSERT INTO Enrollment VALUES(1009,3014);
INSERT INTO Enrollment VALUES(1009,3010);
INSERT INTO Enrollment VALUES(1010,3005);
INSERT INTO Enrollment VALUES(1010,3006);
INSERT INTO Enrollment VALUES(1011,3001);
INSERT INTO Enrollment VALUES(1012,3002);
INSERT INTO Enrollment VALUES(1013,3001);
INSERT INTO Enrollment VALUES(1014,3006);
INSERT INTO Enrollment VALUES(1014,3010);
INSERT INTO Enrollment VALUES(1015,3007);
INSERT INTO Enrollment VALUES(1015,3008);
INSERT INTO Enrollment VALUES(1015,3016);
INSERT INTO Enrollment VALUES(1016,3015);
INSERT INTO Enrollment VALUES(1017,3003);
INSERT INTO Enrollment VALUES(1017,3009);
INSERT INTO Enrollment VALUES(1018,3012);
INSERT INTO Enrollment VALUES(1019,3013);
INSERT INTO Enrollment VALUES(1019,3014);
INSERT INTO Enrollment VALUES(1019,3002);
INSERT INTO Enrollment VALUES(1020,3010);
INSERT INTO Enrollment VALUES(1020,3011);
```

5.3: Views & Stored Procedures

The SQL statements below contains the stored procedure and view we created in our database for bonus marks. The view we created allows the user to view a particular students fees for the university (which is their tuition fee, dependent on how many courses they are taking at the university). The stored procedure we created allows the user to view all of the students in a given course (such as all of the students in course Introductions to Databases).

```
CREATE DATABASE IF NOT EXISTS StudentRecord;
USE StudentRecord;

#temp view for supquery in Fees
```

```
CREATE VIEW tempView AS SELECT ST_ID, COURSE_NAME,

  CASE    WHEN DEPARTMENT_NAME = 'Mathematics'
        THEN 500

     WHEN DEPARTMENT_NAME = 'Computer Science'

        THEN 550
        WHEN DEPARTMENT_NAME = 'Chemistry'

           THEN 600

        WHEN DEPARTMENT_NAME = 'Arts'

           THEN 450

        ELSE 400

    END AS 'Fee'
FROM ENROLLMENT NATURAL JOIN COURSE NATURAL JOIN DEPARTMENT;


#Stored View (bonus marks)
#Shows students fees

CREATE VIEW Fees AS (

SELECT CONCAT(F_NAME, ' ', L_NAME) AS 'Student Name',
COUNT(COURSE_ID) AS 'Number of Courses',
CASE    WHEN COUNT(COURSE_ID) >= 3

        THEN 'Full time'
    ELSE 'Part time'
END AS 'Student Type',
SUM(Fee) AS 'Student Fees'
FROM STUDENT NATURAL JOIN ENROLLMENT NATURAL JOIN COURSE NATURAL
```

```
JOIN DEPARTMENT
NATURAL JOIN tempView
GROUP BY ST_ID);


#Stored Procedure (bonus marks)
#Shows students in given course

DELIMITER //
CREATE PROCEDURE CourseView (IN Course VARCHAR(100))
CREATE PROCEDURE CourseView (IN Course VARCHAR(100))
BEGIN
	SELECT student.St_ID, F_Name, L_Name
	FROM student, enrollment, course
	WHERE student.St_ID = enrollment.St_ID
		AND enrollment.Course_ID = course.Course_ID
		AND Course_Name =  Course;
END //
```