CSC 3335 Spring 2016 Database Project

Group 33

Colton Young coltcael@gmail.com

Tyler Guice guicetyler@aol.com

Jennifer Clements clements222@yahoo.com

Steven Hancock [steven\_hancock@aol.com](mailto:steven_hancock@aol.com)

Client:

Dr. Henderson

111 Lake Hollingsworth Dr. PS. 144

Table of Contents:

Title Page ---------------------------------------------------------------------------------------------- 3

Meeting Notes ---------------------------------------------------------------------------------------- 4

Data Dictionary --------------------------------------------------------------------------------------- 5

Business Rules and Constraints ----------------------------------------------------------------- 7

Discussion of Assumptions ----------------------------------------------------------------------- 9

EERD --------------------------------------------------------------------------------------------------- 10

Logical Model in 3rd Normal Form -------------------------------------------------------------- 11

List of SQL Commands to Create the Database -------------------------------------------- 12

List of Data Values Stored in Each Table ----------------------------------------------------- 13

Output From Reports ------------------------------------------------------------------------------- 23

Sample Queries -------------------------------------------------------------------------------------- 24

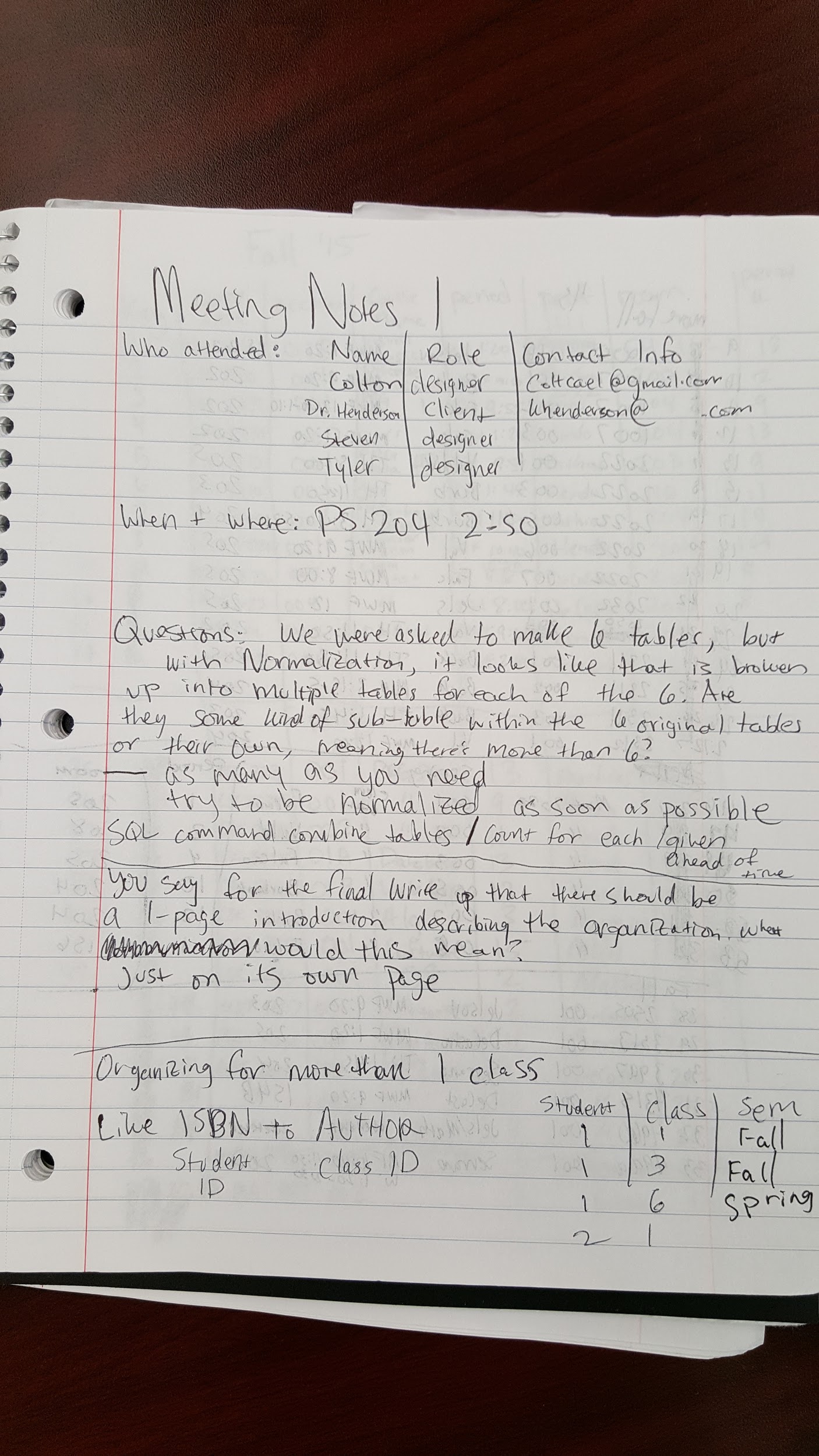
Installation Instructions ----------------------------------------------------------------------------- 40

Quality of Final Project ----------------------------------------------------------------------------- 41

Title Page

This is a database designed by Group 33 in an effort to produce a cohesive database that combines elements in a table of rooms, classes, fall and spring schedules, faculty, students, sections, periods, and student schedules. It includes a few students and faculty, all the rooms in Polk Science and all CSC and MAT classes. This group was created by Colton Young, Steven Hancock, Tyler Guice and Jennifer Clements.

Meeting Notes

Met with Dr Henderson in class regarding writing queries and requirements for the final presentation on February 15, 2016. He explained that we each need to write six queries of the database and that we could split and include the 8 queries he requested. He gave me the list of project requirements. Jennifer Clements

Data Dictionary

* **Term Schedule** (In our case, this makes up Fall ‘15 and Spring ‘16 tables but would, in practice, extend to any semester’s schedule- Is an associative entity that ties together many other fields regarding where, when, and by who a class is taught, among others
  + ScheduleKey- int(length 11)- This is the primary key for Term Schedule and has no meaning to the term schedule other than to identify a row
  + Course- int(length 11)- This is a foreign key that ties into the primary key of the classes table
  + Sections- int(length 11)- This is a foreign key that ties into the primary key of the sections table
  + Periods- int(length 11)- This is a foreign key that ties into the primary key of the periods table
  + Professor- int(length 11)- This is a foreign key that ties into the primary key of the faculty table
  + Rooms- text(length ~65,000 chars)- This is a foreign key that ties into the primary key of the rooms table
* **Classes**- This table is a listing of all classes in CSC and MAT
  + ClassKey- int(length11)- This is the primary key for Classes and has no meaning to Classes other than to identify a row
  + ClassName- text(length~65,000 chars)- This field gives the name of a class
  + CourseCode-text(length ~65,000 chars)- This field gives the course code used by FSC to denote an instance of a course
* **Sections**- This table gives a primary key that ties into a certain section code (e.g. 002, 01A)
  + SectionKey- int(length11)- This is the primary key for Sections and has no meaning to Sections other than to identify a row
  + Section- text(length~65,000 chars)- This field is a column of all the sections that a course in MAT or CSC can possibly have (i.e. 001-009, 01A, 01B)
* **Periods**- This table enumerates the periods (with the primary key) and collects a period’s days and times (e.g. Period 1 is MWF 8-9:10)
  + PeriodKey- int(length11)- This is the primary key for Periods and has no meaning to Periods other than to identify a row
  + Day- text(length~65,000 chars)- This field gives the days (denoted M for Monday, T-Tuesday, W-Wednesday, H-Thursday, and F-Friday) for a period
  + Time-text(length~65,000 chars)- This field gives the time for a period on certain day(s)
* **Faculty**- This table shows all Faculty names and departments for CSC and MAT professors
  + FacultyKey- int(length11)- This is the primary key for Faculty and has no meaning to Faculty other than to identify a row
  + Name- text(length~65,000 chars)- This field gives the name, both first and last, of a professor
  + Department- text(length~65,000 chars)- This field gives the department in which a professor teaches (i.e. CSC or MAT)
* **Rooms**- This tables shows all of the rooms in Polk Science with their purpose, # of seats (if possible) and tells if these rooms have multiple exits and any additional information
  + RoomKey- text(length~65,000 chars) This is the primary key for Rooms and doubles as both the actual room number and as the reference to a specific row of the table (e.g. 204 is both the key to a certain row and denotes this row as being Polk Science room 204)
  + Purpose- text(length~65,000 chars)- This field gives the purpose of a room from a list of possible choices (Classroom, Lab, StoreRoom, PrepRoom, or Planetarium)
  + Seats- text(length~65,000 chars)- This field gives the amount of individual seats for students to use in a room
  + MultipleExits- boolean(length 1- 0 or 1)- This field gives 0 if a room has only one exit and 1 if a room has more than one exit
  + AdditionalInfo- text(length~65,000 chars)- This field gives any additional infor pertinent to this room’s information (e.g. If it has a special name, if it was inaccessible, or a specific Major that it is exclusive to)
* **Students**- This table gives 6 made up students who take CSC and/or MAT classes at FSC
  + StudentKey- int(length11)- This is the primary key for Students and has no meaning to Students other than to identify a row
  + Name- text(length~65,000 chars)- This field gives both the first and last name of a student
  + Major- text(length~65,000 chars)- This field gives the disciplinary major of a student
* **StudentSchedule**- This table, in practice, collects all courses all students have ever taken at FSC
  + AllStudentScheduleKey- This is the primary key for StudentSchedule and has no meaning to StudentSchedule other than to identify a row
  + StudentID- This is a foreign key that ties into the primary key of the Students table
  + ScheduleID- This is a foreign key that ties into the primary key of the Term Schedule table

Business Rules and Constraints

A course may have more than one section.

A room can house only one class at a time, must have sufficient seats, and be suited to the purpose of the course.

A professor can only teach one class at any given time of the day. ( ie.. one class at 8:00am MWF)

A department can offer many courses.

Students can attend only one class at any given time of the day. (ie… one class at 10:00am T TH)

**Relationships**

Departments ---- offer ---- courses

Rooms ---- house ---- classes

Course ---- generates ---- class

Professors ---- teach ---- classes

Students ---- attend ---- classes

Section ---- provides expansion on ---- course

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Term Schedule** (Fall 15 and Spring 16)

Attributes: Schedule key, course, sections, periods, Professor, and Rooms

**Classes**

Attributes: Class key, Class name, Course code

**Sections**

Attributes: Section key, Sections(for MAT and CSC)

**Periods**

Attributes: Period key, Day, Time

**Faculty**

Attributes: Faculty key, Name, Department

**Rooms**

Attributes: Room key, Purpose, Seats, Multiple Exits(boolean), Additional Info

**Students**

Attributes: Student key, Name, Major

**Student Schedule**

Attributes: All Student Schedule key, Student ID, Schedule ID

Assumptions

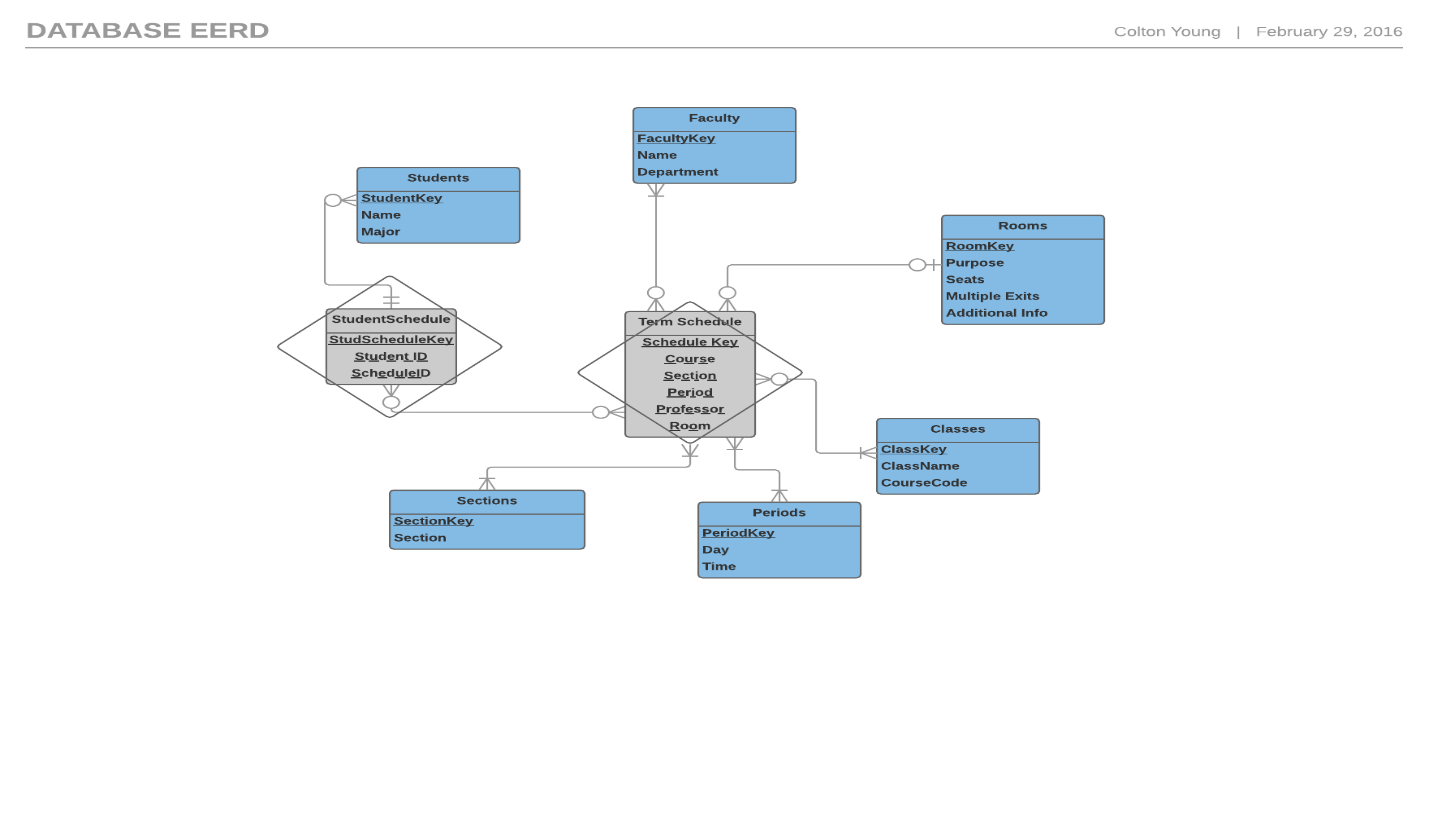
We assumed each table would need to be related by a primary key. This was assumed because we questioned how we would reference each location in a specific table. If this is not correct, changes would not be necessary. The database would still be intact regardless of the object we use as the primary key.

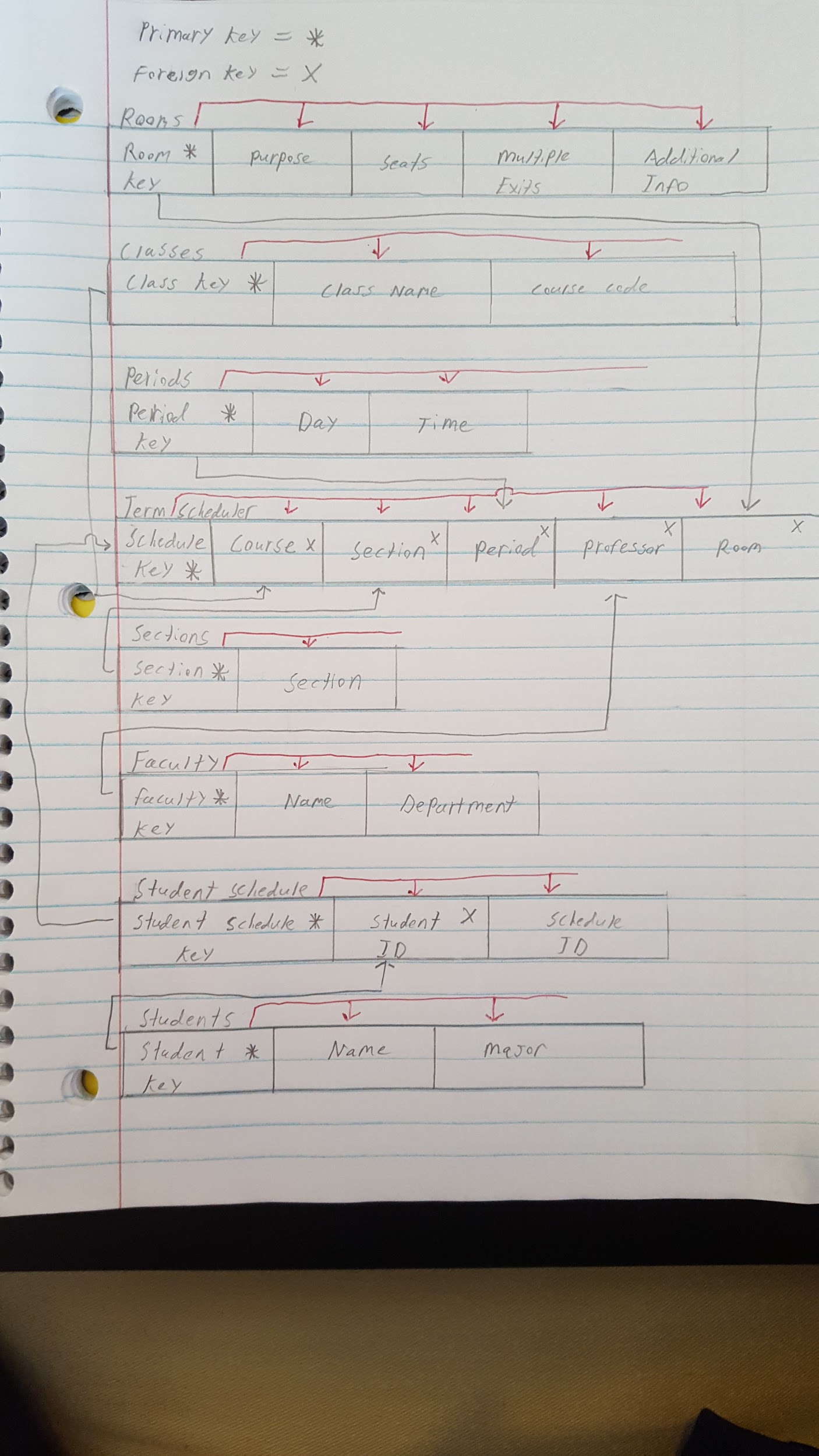
Additionally, we assumed that each student would have a number associated as the key and this would be used to create a new table for student schedules. We assumed this because we needed a way to reference the classes each student had taken. If we are wrong, the student schedule table may be unnecessary and would need to be removed.

Another assumption we made is that once a class is entered into the course catalog, it must be immediately added to the course schedule at least once. Thus, it would be mandatory many classes to a schedule. This came from us questioning how a new class would be added to the schedule. We decided that we were wrong and changed the relationship of classes to schedule to optional many classes to a schedule.

Finally, we made an assumption on separating the fall and spring schedules. When organizing the the schedules we assumed that this would make it easy to distinguish between the classes in each section. This assumption came from the question of how we would initially design the database. The primary key is different for each class in each section, so if we are wrong, it shouldn't affect much. Although it may make more logical sense to put the two together if we are wrong.

EERD



Logical Model in 3rd Normal Form

List of SQL Commands to Create the Database

CREATE DATABASE *team33database*;

CREATE TABLE ‘team33database’.’classes’ ( ‘ClassKey’ INT NOT NULL, ‘ClassName’ TEXT NOT NULL, ‘CourseCode’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’rooms’ ( ‘RoomKey’ TEXT NOT NULL, ‘Purpose’ TEXT NOT NULL, ‘Seats’ INT NOT NULL, ‘Multiple Exits?’ BOOLEAN NOT NULL, ‘Additional Info’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’faculty’ ( ‘FacultyKey’ INT NOT NULL, ‘Name’ TEXT NOT NULL, ‘Department’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’period’ ( ‘PeriodKey’ INT NOT NULL, ‘Day’ TEXT NOT NULL, ‘Time’ TEXT NOT NULL) ENGINE = InnoDB;

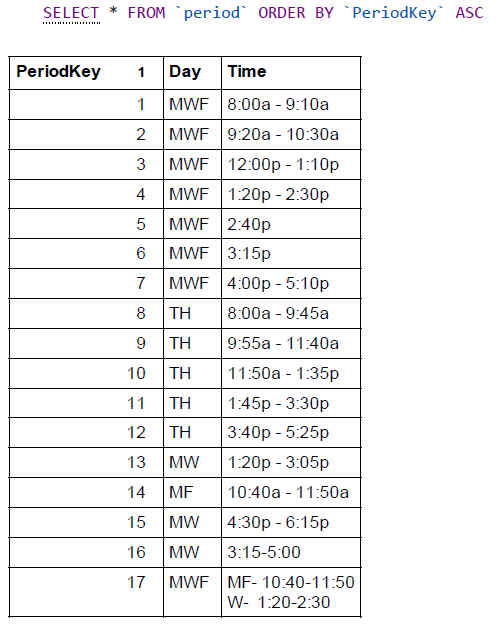
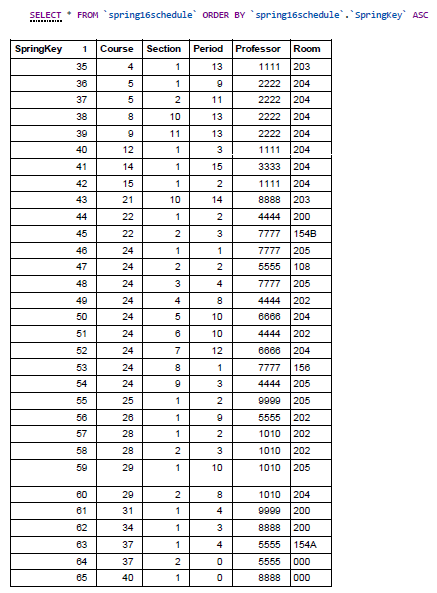
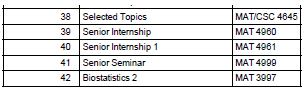
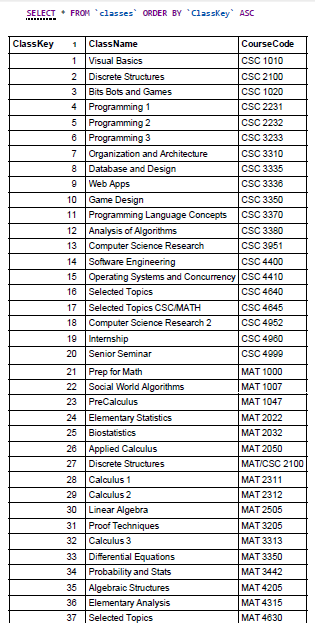
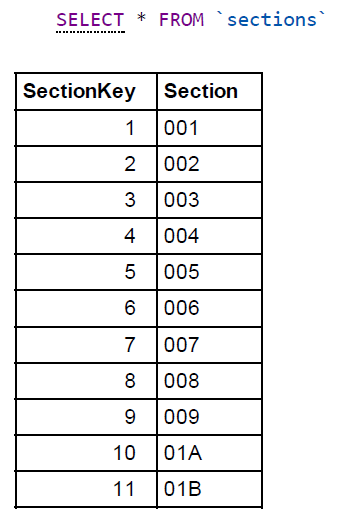
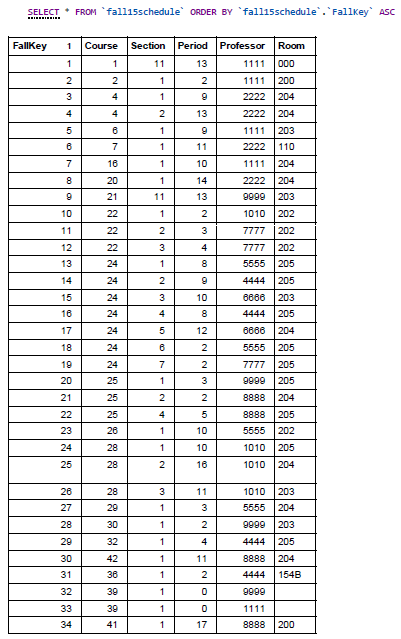
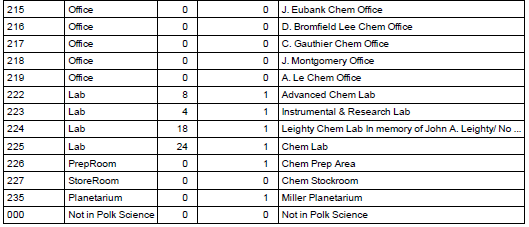
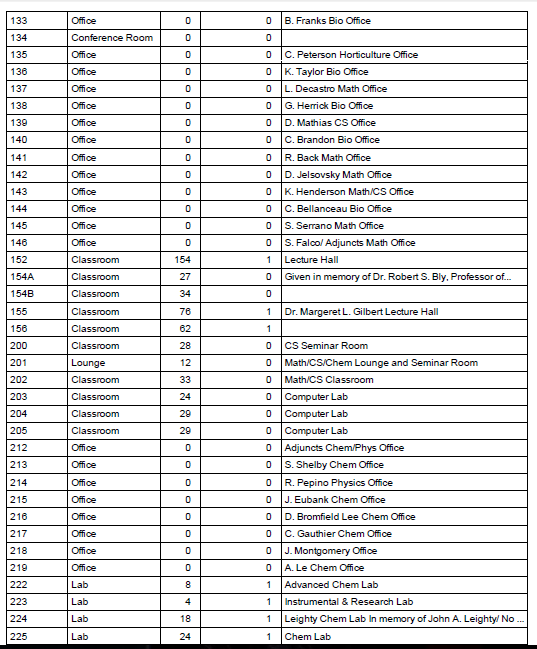
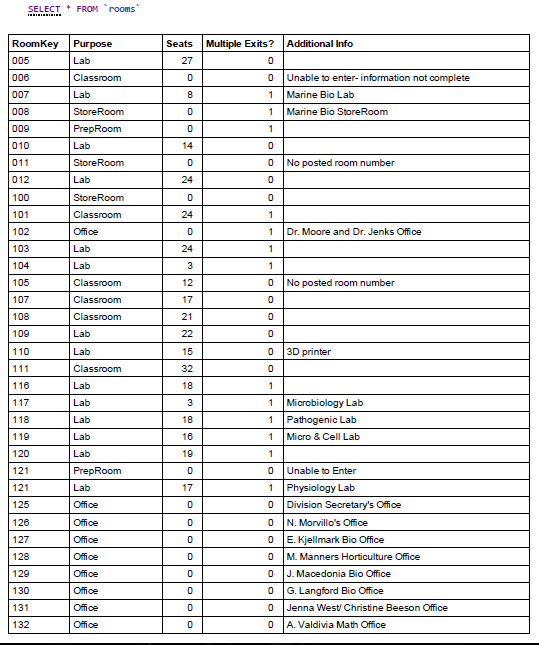
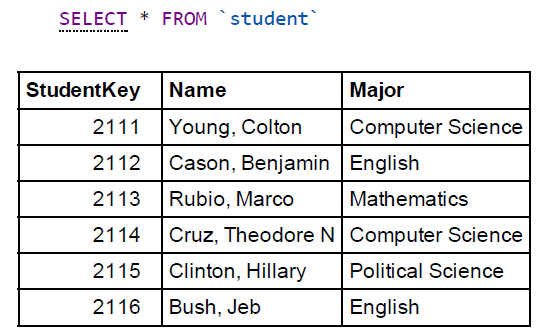
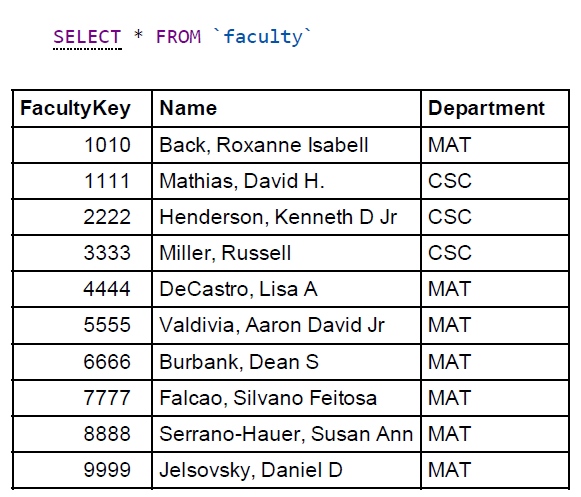
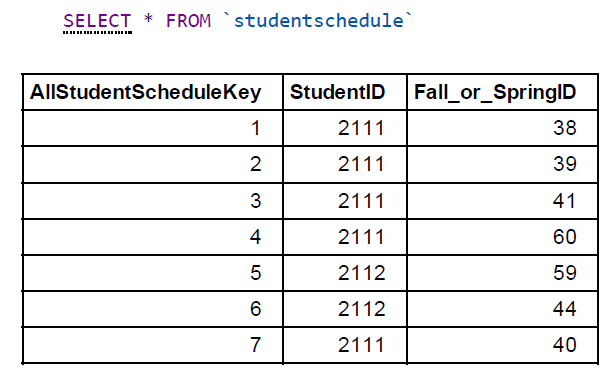
CREATE TABLE ‘team33database’.’sections’ ( ‘SectionKey’ INT NOT NULL, ‘Section’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’student’ ( ‘StudentKey’ INT NOT NULL, ‘Name’ TEXT NOT NULL, ‘Major’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’studentschedule’ ( ‘AllStudentScheduleKey’ INT NOT NULL, ‘StudentID’ TEXT NOT NULL, ‘Fall\_or\_SpringID’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’spring16schedule’ ( ‘SpringKey’ INT NOT NULL, ‘Course’ INT NOT NULL, ‘Section’ INT NOT NULL, ‘Period’ INT NOT NULL, ‘Professor’ INT NOT NULL, ‘Room’ TEXT NOT NULL) ENGINE = InnoDB;

CREATE TABLE ‘team33database’.’fall15schedule’ ( ‘FallKey’ INT NOT NULL, ‘Course’ INT NOT NULL, ‘Section’ INT NOT NULL, ‘Period’ INT NOT NULL, ‘Professor’ INT NOT NULL, ‘Room’ TEXT NOT NULL) ENGINE = InnoDB;

List of Data Values Stored in Each Table

Installation Instructions

1. Unzip project folder
2. Open XAMPP
3. Start Apache
4. Start MySQL
5. Click Admin on the MySQL row of XAMPP
6. Click New at the top of the left-most column
7. Enter a database name in the “Database name” field (can be whatever you want, to keep it logical, you might name it team33database.)
8. Click Create
9. Click Import on the top-most toolbar of phpMyAdmin
10. Click “Choose File”
11. Using your file explorer, find the file team33database.sql that is a part of our project folder
12. Scroll to the bottom of the phpMyAdmin page and click “Go”
13. In the far-left column, find the name that you chose for this database- click the file and you have successfully installed our database.

Quality of Final Project

1. The project is complete. We have comprehensively filled out each of the required fields. The project is also correct. No errors exist that would give faulty results and each column and row is logically and correctly filled out.
2. We tested our database by sorting each column of each table and looked for repeated values (which would have come about by typing errors.) We fixed all of these errors. We also tested it by making sure all SQL commands gave correct results.
3. Some next steps we might take would be to fill out complete student schedules for all students. We might also follow the same process to implement all professors from FSC, all rooms at FSC, all classes, and all students at FSC. Another step we could take would be to merge Fall and Spring schedules into one table called TermSchedule (which we would also do with each additional semester schedule after these two) and just add one field at the end of this new table that allows us to designate which semester the specific class took place in. This would require another associative entity (table) however, where we assigned keys to each of the semesters (i.e. a two-column table of key and semester, e.g. 1 -Fall‘15, 2-Spring’16, 3-Fall’16…)