CECS 323 LAB ONE TO MANY

OBJECTIVE: Getting some experience with physically creating a parent/child relationship and populating it.

INTRODUCTION: This exercise will be one of many "Concept of thing/Instance of thing" examples that we will encounter in this course. The starter SQL for the tables and a few representative inserts can be found here.

You cannot enroll in a **course**, at least not directly. Instead, you must enroll in a **section** of a course. You will create a table for section with the following columns:

- departmentName – varchar(50) not null
- courseNumber – integer not null
- sectionNumber – integer not null
- year – integer not null
- semester – string – This will take on values such as 'Fall', 'Spring' and so forth.
- instructor – varchar(50) not null
- days – varchar(10) not null – This will take on the values:
    - MW
    - TuThu
    - MWF
    - F
    - S
- startTime – integer not null (assume all courses start at the beginning of the hour, and that we are using military time here)
- building – varchar(10) not null
- roomNo – integer not null

PROCEDURE:

1. Run the SQL above to create the Department and Course tables and populate them with sample data.
2. Create the Section table.
    a. Remember to create a referential integrity constraint from Course to Section.
        i. This is an identifying relationship, be sure to reflect that in the referential integrity constraint statement.
    b. Create the primary key for section.
    c. Take a hard look at the columns for this table. Identify all of the candidate keys that you can.
        i. Be sure that there is a business rule for each candidate key, that it is not just a theory of yours that there would never be two sections that agree on all those attributes at once.
        ii. Be sure to **enforce** your candidate keys in the create table statement.
3. Insert several rows into the Section table.
4. Try to add a section into a Course that does not exist.
    a. Cut/paste the output that you get from NetBeans into your report.docx file for later.

   b. Then add that course to the Course table.
   c. Add the proper Department if necessary.

5. Start with the select statement at the end of the SQL that I gave you, and extend it to now include the section number, the startTime, the building, and the roomNo for each section that you have inserted into the database. Include the results from your new and improved query in your report.docx file.
   a. The join has two operands: left and right. Each operand is a relation. The join creates a new relation by matching rows from the left relation to rows on the right relation based on the criteria that you give it in the "on" clause. R1 join R2 join R3 is evaluated as though you coded: (R1 join R2) join R3.
   b. In your new and improved join, you take the result of Department joined to Course, which produces a set of Course records with all of the Department information added, and you join **that** to the Section table to produce a relation of sections that also includes information about the course that the section offers, and the department offering that course. Think about that when you compose the new on clause for your join statement.
   c. Always be careful to join across all of the columns of the primary key/migrated foreign key of the relationship that connects the left hand operator to the right hand operator of your join.

WHAT TO TURN IN:

- The DDL that you used to create your Section table.
- A report out of the rows that you have in the Section table.
  - Right click the Section table in NetBeans Navigator.
  - Select the "View data" menu option.
  - Select all the rows in the output.
  - Right click the selection.
  - Select the "Copy Row Values (With Headers) option.
  - Go to Word, or your favorite text editor
  - Paste the table rows into the document and make a table out of them.
- The console text that you received when you tried to insert a Section for a non-existent course.
- The report that you got by doing your new and improved select statement that joins Department, Course, and Section.
- The text of your new and improved select statement.
- Your team's collaboration document. You can find the template for that on BeachBoard | Content | Student Helps | Lab Collaboration Document.

REFLECTION:

We could have merged all three of these tables together into one massive table. That would have been easier from the standpoint of getting all the information together into one table, but it would have been very redundant because we would end up repeating all the department information for every section of every course within that department. Doing it this way, the department information is stored just once. In addition to wasting space, redundant data leaves us open to the possibility of **contradictory** data. We might have one name for the CECS department chair for one course section, and another name for the CECS department chair for another course section, which clearly makes no sense.