CECS 323 LAB DENORMALIZATION

OBJECTIVE: Getting some experience with two different forms of denormalization.

INTRODUCTION: Denormalization happens when the designer consciously and carefully backs away from a fully normalized model to either meet a performance requirement, or to make the job of navigating through the data easier for the developers. In this assignment, we will perform two types of denormalization (there are more) so that you get an idea of what the issues are that come along when you make this decision.

The first denormalization clones a single column from the parent table (department in this case) down into a child (Course in this case). The motivation might be that we always need to know who the chair is every time that we access the course, and the join is taking too long. So, we copy the value of the department chair into the course, which is redundant. Now we have a different problem to solve. The chair that shows up in the Course rows must match the value of the chair in the corresponding Department row. So, we will build a couple of triggers to make sure that those always stay in sync. The basic DDL to start with can be found here. Note, there are a few things that are different in MySQL than they were in Derby, and you will need to address those before you get these create table statements to work.

The **second** denormalization is simpler. We are just going to merge the Department table into the Course table. That saves us a join, but now all of the department information gets repeated each time we build a course in that department. We need to make sure that we never have contradictory data in the database. If, for instance, one course says that the department office is in one room of ECS, and another course in the same department says that the department is in another room of ECS, that's a contradiction.

PROCEDURE:

1. For the **first** denormalization:
   a. Add the chair field from the Department table to the Course table.
   b. Create an on insert trigger in Course that will copy the value of the department chair from Department into Course.
   c. Create an on update trigger in Department that will update all of the courses in that department if the chair gets changed at the department level.
   d. Write an on update trigger in Course that will make sure that you can**not** change the value of the chair from within Course. You have to do that in the Department table.
   e. Write test inserts to create some representative departments and courses, and then try to change the department chair up updating a row in the Department table.
      i. What happened?
      ii. Why did that happen?
2. For the **second** denormalization, imagine that you have decided to **replace** Department & Course with a single table: deptCourse. That, of course, removes all need for joining between the two tables for **any** Department information, but it now means that two courses within the same

department might disagree with each other regarding the department chair, the department office building, or the department office number:

a. Create a table, call it deptCourse.
   i. This table has all the Department columns in it as well as the Course columns.
   ii. The primary key is deptName **and** courseNumber.
   iii. A candidate key is deptName and courseName.
b. Add an on insert trigger to the deptCourse table that checks to make sure that there are no other rows in the table that disagree with the new row regarding any of the department information.
c. Add an on update trigger to the deptCourse table that checks to make sure that no one is able to change any of the departmental information such that there are any discrepancies between courses within the same department.

WHAT TO TURN IN:

- The DDL that you used to create your deptCourse table.
- All your trigger code in a .sql file.
- Your DML that you use to put sample data into the tables.
- Console output showing the results that you received from testing your triggers as well as documentation telling me what you did to get that response from you triggers.
- Your team's Collaboration document. You will find a template for that at BeachBoard | Content | Student Helps | Lab Collaboration Document.