Lab 07 – Entity Relationship Diagrams

(Database Design ERDs)

Objective:

- Students work with a simple scenario to find the entities and the relationship between entities.
- Students learn to define the relationship cardinality and optionality.
- Students learn to design a simple database using an ERD model.
- Students learn how to fix many-to-many relationships in an ERD.

Submission:

Students MAY work in partners for this lab. It is best to complete this lab through collaboration and shared thought.

Using the draw.io website, save your lab file as a PDF file. You need to submit a single PDF file for this lab. Both names, studentID's of the students, and the current date must be in the output document (make a box on the page and put the info in it). ***Submission without names and the date will receive a mark of zero (0).***

ONLY ONE SUBMISSION PER PARTNERSHIP PLEASE

Tasks:

Draw the corresponding crow's foot ERD for the following scenario. M:N relationships should be broken into two 1:M relationships using a bridge entity. Label PK attributes and FK when applicable.

Bridge/Junction Entity: The **bridge/junction** entity is used to eliminate the many-to-many relationships. This **entity** sits between the two **entities** with the many to many relationships and this composite **entity** shares the primary keys from both tables.

Show all PK's and FK's with your solution.

First you need to find all entities. Next, list the attributes for each entity. Next, you need to find the relationship between each two entities if there is any relationship. Fix all *many-to-many* relationships. Finally, make sure all PKs and FKs are properly defined.

For all entities, list their attributes. For example, for the students, we need to store student ID, student name and last name, email address, address, and phone number.

You do not need to define all possible attributes. Just determine some important attributes.
For each entity, you need to define a primary key. If there is a relationship between two entities make sure you define the corresponding foreign key in the child table.

- ***Seneca College*** contains many departments.
- Each department has many programs, but every program belongs to only one department.
- Every department has many professors.
- A professor can work for only one department.
- A program has many courses.
- A course can be a requirement of many programs.
- A professor can teach many courses. A course can also be taught with many professors, via sections.
- Each section is only taught by one professor (ignore the summer term)
- A program has many students studying in that program.
- A student can study multiple programs. However, a student has to register in least in one program.
- A student may take many courses.
- A student, however, can be off from school and do not take any courses.
- A course can be taken by many students.
- A new course may not be available yet so the course may not be taken by any student.
- A student may have a program advisor. Having an advisor is optional for students so some students may not have any advisor.
- An advisor can have one or more students assigned to them.