CS4.301: Data and Applications **Due: 7PM, 29th August 2020**

Project Phase #1

Instructor: Kamal Karlapalem *Released:* 20th August 2020

1 The Task

Consider a **mini-world** of your choice and come up with the data requirements for the database design and functional requirements for database operations.

A mini-world is a set of users and how they will use the database you design—for example, a school examination system used by students, teachers, and more.

2 Database Requirements

Your mini-world should result in data requirements that have:

1. At least *five* entity types
2. At least *one* entity with two key attributes
3. At least *two* weak-entity types
4. At least *five* relationship types (which should include cardinality ratios and participation constraints)
5. At least *one* ($n > 3$) degree relationship type.
6. At least *one* subclass
7. Few composite, multi-valued, derived attributes

2.1 Bonus

1. Relationship type with the same participating entity type in **distinct roles**
   - Example: In a COMPANY database SUPERVISION relationships between EMPLOYEE (in the role of supervisor) and EMPLOYEE (in the role of subordinate)
2. Relationship with cardinality constraints constraint (specifies a *minimum* or *maximum* participation)
   - Example: An employee can manage at *most one* department.

There will be a small weight for mini-worlds that are unique with a very specific purpose and specialized applications. For such mini-worlds above requirements can be relaxed.

3 Functional Requirements

In your mini-world, you will be required to have applications that operate on the database. This program will query and/or update the database. Example: Check for a student's grades from last year. These operations are functional requirements.

The functional requirements to meet are:

1. Retrieval:
   (a) At least **one** query function for each:
      i. **Selection** Example: "Retrieve complete data tuples of students belonging to UG2K19"
      ii. **Projection** Query to enable the users to search the database by a particular attribute. Example: "Names of all courses with $\geq$ 50 students.
      iii. **Aggregate** function (SUM, MAX, MIN, AVG). Example: "Maximum score secured in the midterm exam".
      iv. **Search**: Search (partial text match) for entries in an entity, matching for subparts of the entries. Example: "APP" will match with "APPLE".

   (b) Analysis: At least **two** analysis reports to be generated:
      - Example: "Number of students that scored above average in the midterm exam in all courses" Note: We expect that these reports convey something about the relationship between entities and are not simple selection operations from a single entity. To do so, you have to use the Join operator.

2. Modification:
   (a) At least *one* insertion of data, check for violations of integrity constraints.
   (b) At least *one* update operation
   (c) At least *one* delete operation

4 Submission

One member per team must upload a PDF document (named as <teamname>.pdf - without the '>' & '<') with the following details:

1. A paragraph or two describing the mini-world, purpose of the database, and users of the database. What will the users do with the database?
2. Database requirements section
   (a) Relationship between entities
3. Functional requirements section

5 Points to Remember

As always, it's good to get an early start on the assignment! Try keeping the following points in mind when you get to work:

- The requirements document is simply a description of what your database has to store and what functionality you provide on top of it.

  - Example: If your database requirements describe the relations between students, their grades, courses, and instructors, the functional requirements describe what operations should be possible - such as look for a report card.

- Specify constraints that your data may have. If an attribute has a specific, non-trivial domain, please specify the domain. If there are cardinality constraints, participation constraints. For instance, if an object of one entity can be related to only one other object of another entity type, specify. Example: an employee can belong to only one department.

- You may not understand some of the concepts like join, aggregate, etc. as of now, do not worry about that. You will know them well enough by the time you have to build the app.

- You will build your ER models and your final program based on these sets of requirements. Hence, try to think about this in detail and keep in mind that this cannot be changed **drastically** later.

Best of luck!