Z. Meral Ozsoyoglu **EECS 341: Introduction to Database Systems** March 9, 2017

Spring *2017*

Project Assignment Part 2

Initial Report due: Tuesday, March 28th, 2017

Project Presentation Slides due: Monday, April 24th, 2017

Final Report due: Monday, May 1st, 2017, 1:00pm

PART 2 of the project is a team assignment. A team of 2-4 students will be working on each project. Each group needs to submit an initial report (no more than 3 pages) with the following items.

An initial report, due on Tuesday, March 28, is to be submitted with the following sections.

a. Name of the application, team members
b. Application background and what it will be used for
c. Data description, including all constraints that need to be enforced—in English and in an intuitional manner
d. An ER diagram design of your database.

Note: The total number of tables should be around 5-10. Building a web-based user interface **(with the web server residing in another machine)** is optional, and, a simple command-line interface to the database is acceptable. Having said this, building a quality web-based user interface will earn you extra points, up to 8% of your final grade.

Project Presentations: There will be project presentations (~5-10 minutes per project) in class during the last week of classes. Due date for submissions (via Blackboard) of project presentation slides is Monday, April 24, 2017, 1:00pm.

Final project report: Each team needs to submit one final report (no more than 15 pages) with the following items. **The final report+source code+your output screenshots are to be uploaded (as a single zip file) to the Blackboard on Monday, May 1st, 2017.**

Your final project report must contain at the least the following sections.

a. Application background,
b. Data description, including all constraints that need to be enforced,
c. An (updated) ER diagram design of your database,
d. A list of functional dependencies derived from the semantics of your data,
e. The schema of your database, satisfying 3NF (if not BCNF),
f. Example queries supported by your system,
g. Implementation: OS, DBMS, programming language,
h. The role and contributions of each team member (make sure that the contributions of each team member is doable without the implementation contributions of other members),

i. A short presentation/demo in the last a few classes,
j. What you have learned from this project—over and above from what is covered in the course.

HOW IS A WEB-BASED DATABASE IS BUILT (FOR AN APPLICATION) IN PRACTICE?

Below is a list of sections that any DBMS design, implementation, and evaluation should have; however, given that this project is only a part of the course, you are not expected to follow these steps strictly.

1. *Introduction: Overview.* Give an overview of the environment (i.e., a portion of the "real world" as chosen by you) about which a web-based application that deploys a database needs to be developed. I will refer to the application as *application X* and the database as *database D*.

2. *Application Requirements Specifications.* List the requirements specifications of the application X. Note that the requirement specifications of X list in detail what X will do, what types of web interfaces it will have and their functionality (if possible, with mock-up web forms that you can put together), and the user interactions with the system. This section normally is several pages, convincing the reader that enough time is spent thinking over the details of the application, and the designer knows what she/he is about to build as an application.

3. *Database Requirements Specifications.* List the requirements specifications of the database D, i.e., discuss in English (a) data (objects and relationships) to be maintained in the database, and their details, (b) queries and transactions that the implemented system will employ and the possible frequencies of these queries and transactions, (c) events, actions (triggers) of the database, and the (d) integrity constraints associated with the database. This section is usually several pages, illustrating the mastery of the knowledge needed for building a viable application.

   Note: Steps 2 and 3 demonstrate the teams' ability to analyze a problem, and identify and define the computing requirements appropriate to its solution: the project team (1) correctly observes requirements specifications, (2) understands and effectively uses the iterative nature of requirements specification refinement, (3) carefully documents requirements specifications, and (4) correctly identifies the tradeoffs involved in design choices. (5) (Together with steps 5, 6, 7 and 8), the accuracy and performance analysis of the specific algorithm chosen is complete.

4. *ER Data Model Design*. Design the ER data model of your database in detail. List the entities and their attributes. Specify the domain of each attribute. Specify the properties of each attribute (i.e., key, composite/simple, single-valued/multi-valued, derived, incomplete with different nulls, roles, weak/strong entity type, etc.). Specify the relationships and their attributes. List the properties of each relationship (i.e., degree of the relationship, cardinality ratio constraints (1-1, 1-N, N-M), key constraints, participation constraints (total, partial), other application-specific constraints, etc.). Draw the E-R diagram, and incorporate everything discussed in this step into the E-R diagram.

   Note: Step 4, together with the other design and implementation steps, demonstrate the team's ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs: the project team develops a design strategy, including a plan of attack, decomposition of work into subtasks, development of a timetable.

5. *Transforming the ER Model to the Relational Model.* Translate your E-R model (by using explicit transformation) into the E-R model. Each entity should map to a (entity) relation, and each relationship should be accounted for (either represented in an entity relation or as a separate relation). Each entity/relationship attribute should be accounted for in the transformation. List for each relation the primary/candidate keys, foreign keys. Specify the entity and referential integrity

constraints, and discuss why they are satisfied for each relation. Explicitly specify each relation using CREATE TABLE commands of SQL.

6. *Creating your database.* Download your DBMS of choice, and use it. Describe what DBMS you use, and the issues encountered in installing/using it. Then create your database schema, and include in your report actual database CREATE TABLE commands from the DBMS you are using. Use a GUI interface to your DBMS, if possible.

7. *SQL Queries and an Exercise in RA and RC.* List the queries of your project in English, and specify them in SQL, RA and TRC.

8. *Integrity Constraints.* List any integrity constraints in general on the entities and relationships as a whole. Explain how you intend to enforce them; i.e., are you going to build enforcement mechanisms by

    a. Specifying SQL constraints and/or triggers,

    b. Applying database dependency theory (e.g., normalization via functional dependencies) (don't do normalization here; just explain), or

    c. Applying *external* integrity enforcement (e.g., transactions/user-defined procedures, etc). If you need to enforce any constraints at database initialization/update time, list them, and specify how they are to be enforced. Constraint specification needs to be complete in the final report. This section of the report explicitly specifies general constraints, triggers, and stored/external procedures.

9. *Relational Database Design--Applying the Dependency Theory.* Perform normalization by applying all those algorithms you have mastered in the class. More specifically, define your functional dependencies, and find the minimal set of f.d.s. Are all your relations in BCNF or 3NF? If not, apply the algorithms you have seen in the class to decompose and make them BCNF/3NF. Note that your decompositions should be lossless and dependency preserving.

    Note: Step (7) evaluates the team's ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices: The project team (1) understands how sub-problems interrelate and demonstrates ability to integrate prior knowledge into a new problem, (2) fully comprehends the time and space-complexity of the application being designed, (3) observes and completely defines the formal model of the system being designed, (4) correctly identifies the tradeoffs involved in design choices, and (2) uses algorithmic techniques effectively.

10. *Revisiting the Relational Database Schema.* On the basis of the analysis you have done in sections 8-10, the team may perhaps choose to revise the relational database schema (merging/splitting relations) so that (a) the most frequent and most costly queries run faster, and/or (b) integrity constraints are enforced faster/easier.

    Note: Step 9, together with steps 4, 5, 6, 7, 10 and 11, evaluate the team's ability to apply design and development principles in the construction of software systems of varying complexity: The project team (1) can relate design and development concepts to practical problem solving, (2) can defend design and development decisions effectively, (3) uses appropriate resources to locate design and development information needed to solve problems, (4) demonstrates understanding of how various pieces of the software system relate to each other and the whole, (5) formulates

strategies for developing a complex software system, and (6) the software system developed is correct and properly functions.

11. *DBMS Implementation.* Start using the DBMS. Create your schema, constraints, triggers and queries. Populate your database, and run/test your queries, triggers, constraints. Develop and test your stored procedures). Summarize the main components of your code here with proper explanations. Discuss any problems encountered and how you have solved them. If some of the problems require the team to extensively redesign everything, the team may choose to point them out, and not implement them. If some of the stored procedures are too elaborate, scale down your design, explain your decisions, and implement the scaled down version.

12. *Application Implementation—Code* transactions, web forms, web services, etc. Discuss any problems encountered, and how they are resolved.

    Note: Steps 10 and 11 evaluate the team's ability to use current techniques, skills, and tools necessary for computing practice: the project team (1) uses literature- and/or web-based resources effectively in assignments/projects, (2) seeks information on problems from multiple resources including appropriate on-line material, (3) is able to interpret and understand information from a variety of resources, (4) maintains current, state-of-the-art abilities in web and PC use, (5) is able to learn and effectively use stand-alone software or web-based resources, (6) understands and effectively uses digital libraries and other scholarly sources (web-based or not), (7) suggests new approaches and improves on what has been done before, and (8) uses programming techniques effectively.

13. *Conclusions.* Discuss here anything else you want to say, and conclude.

14. *Appendix 1. Installation Manual.* Write an Installation Manual that explains how one can compile and implement your system.

15. *Appendix 2. Users manual.* Write a Users Manual that explains to a naïve user how to start using your application.

16. *Appendix 3. Programmers Manual.* Write a Programmers Manual that explains the classes, functions, etc and their functionality. Post your report, data, etc., on your web page. Supply any password info that I may need to run your application. Submit everything by the deadline posted in the class web pages.