EECS 341: Introduction to Databases

Spring 2015

Final Project Reports

- **Final Project report**: Each team should prepare only one report document (no tar/gzip/etc files; no multiple files; no code), and it should only be in pdf format (no other formats are accepted). Each Project team member should upload the team's report with the additional section for individual comments to the blackboard separately (note that, multiple uploads, one for each member is for grading purposes).
- **FINAL PROJECT: upload all of your code as a zip-file (not tar, not gzip, not anything else).**
- Each project final report document name should have the **last names of all team members concatenated with dots**, followed by any descriptive phrase of your choice. E.g., Final Project report by John Black, Chuck Green, and Mary Zales about a restaurant database application should be entitled: "Black.Green.Zales.FinalProjectReport.RestaurantDatabaseApplication".
- You reports must follow technical report writing standards:
    - title page with project title, team members, e-mails, date, course number and title.
    - Numbered sections, subsections, etc.
    - Please see the detailed instructions and outline below
- Your reports must be detailed, and respond to each and every one of the items requested below.

Outline for Final Project Reports:

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. This section should be several pages, convincing me that you have spent enough time thinking over the details of your application, and you know what you build as an application.
3. *Database Requirements Specifications.* List extensively 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 should also be several pages, illustrating your mastery of the knowledge needed for building a viable application.
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.). **(I assume you have already have your E/R model checked and verified by me or one of the TA's earlier).** Draw the E-R diagram, and incorporate everything discussed in this step into the E-R diagram.

5. *Transforming the ER Model to the Relational Model.* Translate your E-R model (by using explicit transformation) into the Relational 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 learn how to 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. Show your actual commands in the report, if possible (e.g., screen shots, etc).

7. *SQL Queries and an Exercise in RA and RC.* List the queries of your project in English. Choose the **toughest** 5 of them, and specify them in SQL, RA and TRC. Be creative with queries: you can easily envision much more sophisticated queries,

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. That is, in this section of your final report, you must explicitly specify general constraints, triggers, and stored/external procedures (for procedures, I do not want code; only the function specs).

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.

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

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 you to extensively redesign everything, you may choose to point them out, and not implement them. If some of your stored procedures are too elaborate, scale down your design, explain your decisions, and implement the scaled down version.

12. *Revisiting the Whole Project* **(This step and step 13 have to be answered by each project member separately, and uploaded to the blackboard as a separate document**).

    - Discuss any problems encountered overall during the whole project progress. Any scaling down needs to be explained here.
    - What else would you have changed given more resources?
    - What else would have been useful to add to the system?
    - How could your project evolve into a commonly-used application?
    - **Explicitly state how much of the code is developed by you as a team member.**

13. *Team work* (**This step has to be answered by each project member separately uploaded in the same document as answers to step 12**).

    - First, specify in detail who did what.
    - Next, specify what went well as a team and what did not?
    - Did you have team meetings, and how often?
    - Did you share information effectively?
    - Was the division of labor fair?
    - Who did what type of decision making?
    - Did you disagree/agree on any topic, and how did you resolve them?

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

The final version of the project is to be turned in as a single zip file on blackboard, including

- A README file in the top-level folder that explains what is where, etc. Include usage instructions for the interfaces. Everything should be in a single zip file so that when I unzip it, I can read the README file, follow the directions, and run your project.
- *Final Project report in pdf*
- *Installation Manual* that explains how one can compile and implement your system.
- *Users manual* that explains to a naïve user how to start using your application.
- *Programmers Manual* that explains the classes, functions, etc., and their functionality. Submit everything by the deadline posted on Blackboard (**Monday, April 27, 2015, 3:00pm**) .

15. *Code.* At the time of your demo, prepare to walk through your code with me or the TA if needed, and to turn it in so that we can compile and run if needed.