Northeastern University
CS3200 – Database Design
Spring 2018, Derbinsky

Project Specification

The goal of the project is to gain hands-on experience with designing and implementing a full database application from the ground up. Given a narrative, your group will design a set of relations, populate the database, write a set of contextually useful SQL queries, and then implement a user interface to execute these (parameterized) queries, and view results, in a programming language/stack of your choosing[1].

1 Teams

Students will work in project teams, typically with 4 members. Group membership will be set by the instructor via a random process.

2 Deliverables

The team must produce ...

1. a final **packet**, including all design and implementation documentation, as well as code;
2. and a **presentation**, to be delivered as a link to a YouTube video.

Additionally, each team member will individually submit a **peer evaluation**.

2.1 Packet

The packet should be turned in by one member of the team and must contain at least the following items:

- Report (`report.pdf`), which includes ...
  - A cover sheet with the team members, class/semester, and date
  - Abstract: a 1-page overview of the problem being solved and the solution delivered
  - A detailed textual description of the problem, including what entities are involved, how they relate/constrain each other, and what information needs to be extracted/presented – this is your opportunity to make concrete any assumptions you had to make based upon ambiguities in the supplied narrative
  - ER Diagram(s): 1 global, as many logically separated local views as is appropriate
  - Normalized relations with primary/foreign keys clearly identified; any deviations from the ERD and/or 3NF must be justified in text (including denormalization)
  - Physical design: including justification of any included indexes, denormalization, etc.
  - Screenshots of your running system, demonstrating example input/output of each task
  - A project retrospective, including at least ...
    - what you liked and disliked the most
    - what you found the easiest and the hardest

[1] You MUST use a relational DBMS, and you are NOT allowed to use ORM software.

    - what you learned from doing a large-scale project
  - A conclusion statement: what was produced, what still needs to be done

- A link to a **private** GitHub.com[2] source code repository (`repo.txt`), which includes ...
  - Structure and data dumps (i.e. DDL & DML)
  - Application source code
  - A `README.md` file in the root indicating all of the program names, what they do, and how they are compiled/configured/used (this must be sufficient to get the system up and running in a VM). There must be links to two additional YouTube videos:
    - An annotated walkthrough of a system build/installation: any steps necessary from cloning the repo to a full run
    - A "demo" video, showing input/output examples of *at least* each of the tasks

- Presentation information, which includes
  - A link to a YouTube video (`presentation.txt`)
  - Static version of presentation slides (`presentation.pdf`)

Submit the packet as a single ZIP file.

Late work will not be accepted. **Updates to the repository/videos made after the submission deadline will result in 0% credit.**

2.2 Presentation

The presentation is intended to be a brief overview of the project, including design and implementation. All members of the group are required to visibly and/or audibly contribute to the group presentation; if a group member is not clearly presenting, s/he will receive a reduced individual project grade.

2.2.1 Format

The presentation will be made using PowerPoint (or equivalent) and must cover at least the following aspects of the project:

a) Project Motivation: non-technical overview of the problem being solved

b) System Description: what does the system do? how does this solve the problem? baseline features (i.e. tasks), as well as any additional above-and-beyond functionality your team added

c) System Architecture: what dbms, front-end language(s), and any other component(s)? why? how does everything talk?

d) Database Design: ERD, relational model, source of the data for each table, physical design (note: this can be challenging to communicate effectively given the time constraint and the need for readability)

Your presentation should last 8–10 minutes.

[2] Request for free ASAP: `https://education.github.com`

2.3 Peer Evaluation

Group projects are sometimes looked upon as being "unfair." To combat contribution inequity, each team member's perception of the quantity of work that s/he performed and that of each team member will be analyzed against the perceptions of the team member(s). Through this process, hopefully equity will be achieved that reflects each member's contribution to the group effort.

Each team member will submit a report rating the relative contributions of each team member (including her/himself) using a single number, as well as optional commentary. The aggregate rating for each student will determine the grade that individual receives, relative to the group grade. In order for this process to work effectively there is the need for each group member to be honest and objective; these ratings and comments will be kept confidential.

2.4 Milestone

During the semester there is one intermediate milestone (see the Syllabus for the date) when components of the final packet are due. This deadline is intended to facilitate project progress, as well as serve as a structured opportunity for feedback. It is expected that your final work may differ from what was submitted at this checkpoint; however, the submission will be graded for quality and completeness.

2.4.1 Expectations: ERD, Mapping, Data, Queries, Repo

You are to submit the following items as a single ZIP file:

- A detailed textual description (`description.pdf`) of the problem, including what entities are involved, how they relate/constrain each other, and what information needs to be extracted/presented – this is your opportunity to make concrete any assumptions you had to make based upon ambiguities in the supplied narrative
- ER Diagram(s) (`erd.pdf`): 1 global, as many logically separated local views as is appropriate
- Normalized relations (`logical.pdf`) with primary/foreign keys clearly identified; any deviations from the ERD and/or 3NF must be justified in text (including denormalization)
- DDL (`ddl.sql`): SQL that instantiates your normalized relations in a DBMS of your choosing (must be identified in the comments)
- DML (`dml.sql`): SQL that populates your tables (each relation should have at least 10 rows, moreso as necessary to demonstrate tasks/reports)
- Task Queries (`tasks.sql`): SQL that implements each of the required tasks (provide comments where appropriate for parameterization, sequencing, etc)
- Report Queries (`reports.sql`): SQL that implements each of the 5 complex reports (provide comments describing the purpose, as well as justification of the complexity based upon the scoring algorithm)
- Repo (`repo.txt`): a link to a **private** repository on GitHub.com with a non-empty `README.md` file

3 Grading

The group project grade is based upon ...

Presentation (20%) Includes aspects of professionalism (how the group presented itself, was the presentation rehearsed, did it fit well into the time allowed, production quality) and the degree to which the required sections were covered (see above).

Database Design (20%) Includes the ERD and resulting relational schema (was the design correct semantically, as well as technically both in diagrams and code; were tables normalized to at least 3NF; was a reasonable physical design applied and justified, relative to queries).

Task/Report Queries (30%) Correct and efficient implementation of the supplied tasks, as well as context-appropriate design & correct/efficient implementation of additional 5 report queries.

Product and Documentation Quality (20%) Includes the sample data (quantity, quality, strategy and implementation for gathering/producing the data), the report (did it have all required content, was the writing grammatically correct and professionally presented), the source code (was it commented; were user inputs sanitized; were good security measures taken, such as salting passwords), and the ability to install, run, and use the system. Note that a "pretty" and usable interface is encouraged, and may be given bonus points for outstanding work, but is not required (i.e. if command-line tools yield a fully functional system, all associated points will be awarded).

Intermediate Milestone (10%) The submitted components will be evaluated based upon the criteria in this section. While it is understood that the work will change by final submission, final-submission quality is expected.

As described above, each team member will submit a peer evaluation report. These evaluations are a serious statement and are used to re-distribute up to 50% of the grade on the project: if all group members agree they put in equal share, each individual grade will be equivalent to the group grade, but those who did less will receive up to 50% lower individual grade, and those who did more will receive up to 50% greater. No late submissions will be accepted for this evaluation. If you do not submit an evaluation it will be assumed that you did not perform your fair share of the work and your grade will suffer as a result.

3.1 Report Complexity

There are three classes of queries you will write for this project:

- Queries for your system to install/run (e.g. creating tables, populating tables, and simple queries to add/update/remove entities)
- One or more queries to implement each of the required narrative tasks
- **5 complex reports**

For the last item, you must come up with at least 5 queries that you highlight as "reports;" that is they extract useful information out of your database and present it in an effective fashion. **At least 3 of these queries must be parameterized via user input.**

To help you gauge the complexity of these report queries, please reference the following approximate scoring algorithm. Note: actual grading of queries will be subjective, and so the following guidelines do not guarantee that your queries will be considered "complex" (especially if you try to game the system). Queries that are insufficiently complex will receive partial credit; many trivial/easy/moderate queries will not add up to fewer complex queries (and only the first 5 will be graded).

3.1.1 Score Your Query

Start with a score of 0, add points for each applicable question:

- # Tables joined (1–2:0 points, ≥3:1 point)
- Non-inner/natural join? (no:0 points, yes:1 point)
- # of subqueries (0:0 points, 1:1 point, >2:2 points)
- # queries comprising result via union/intersect (0:0 points, ≥1:1 point)
- Aggregate function(s)? (no:0 points, yes:1 point)
- Grouping? (no:0 points, yes:1 point)
- # ordering fields (≤1:0 points, >1:1 point)
- # WHERE/HAVING conditions not for joins (≤1:0 points, >1:1 point)
- Non-aggregation functions or expressions in SELECT/WHERE? (no:0 points, yes:1 point)
- Strong motivation/justification for the query in the domain? (no:0 points, yes:1 point)

3.1.2 Score Meaning

Your score falls into one of the following buckets:

Trivial 0 - 1 points

Easy 2 - 3 points

Moderate 4 - 5 points

Complex >5 points