COMP 3380 Project

Version 2 (Oct 14)

Concept

Students will design and implement their own database based on publicly available data. Students will model the database using the models learned in class, apply techniques learned in class, and use SQL to formulate and run some queries.

Group Setup

~~Projects may be done individually or in groups of up to five. Students are free to select their own group. ONE member per group should leave a message stating all group members in the appropriate Discord channel.~~

~~All students who have not assigned themselves to a group (or stated they are working alone) by October 1, 2021 will be randomly assigned to one.~~

Groups have been assigned. Anyone wishing to leave their group must inform the professor BEFORE October 31, 2021.

Only one member per group should submit files for each part of the project. Make sure to include the names and student numbers of all members of the group in any written components.

Grading Rubric

Parts 1-3 below will be graded on two components: Concept and Execution. Concept consists of the creative decisions made by the students (e.g., what data to choose, what queries to create) and defines the upper bound of the possible grade in that section. Execution consists of the technical work required to manipulate the data (e.g., drawing an ER diagram, writing the SQL code). The best projects will have ambitious and creative concepts that are executed well.

I have provided general guidelines for what will be considered an A, B, or C concept score for each section in that section, as well as a short list of the technical tasks required for the execution component. An A+ tier is not provided, as an A+ on the project would require students going above and beyond expectations in a creative way not easily quantified here.

Part 1 – The Data

Part 1.1 – Data Discovery

The first part of the project involves finding some data to analyze. One viable source is the Winnipeg Open Data portal (https://data.winnipeg.ca), though students are free to use other sources. Any kind of data is viable, but it must either be open for public use (e.g., under a Creative Commons license), or you must have the explicit written permission of the data's owner to use it.

Data from a public source but acquired by someone other than the owner (such as a fan wiki for a video game) is acceptable, but students must acknowledge the original owner of the copyright.

Students may aggregate data from several sources, but all data must ultimately be connected. Students will have to draw an ER diagram with this data (see Part 1.2), and that diagram ***must*** be a connected graph.

Part 1.2 – Database Design

Students must draw an ER model (including EER components, if appropriate) which represents the database they have chosen to create. The model must include participation and cardinality constraints, as well as a brief justification for each. Students are reminded that justifications should consist of explaining the "why" of constraints, not merely putting them into words (e.g., saying "not every Song is written by an Artist" is a bad justification; saying "some Songs were written by unknown Artists and thus aren't in the Wrote table" is a better one).

Students must convert their ER model to a relational model, and then normalize that model as much as is possible according to the rules and standards discussed in class and in the lectures.

A submission for this part will include the ER model and the final (post-merge and post-normalization) relational model.

Part 1 Grading Expectations

Technical Components

- Drawing an ER/EER diagram
- Justifying all participation/cardinality constraints
- Translating an ER diagram to a relational model, including merging
- Normalizing a relational model.

C Tier

- Data which ultimately breaks down into 5 or fewer tables and/or consists of a total of fewer than 100 rows
- Data centralized in one main table, with all other tables mostly being support for it
    - A support table is a table that is usually small in both arity and cardinality, and is mostly used for lookup purposes (e.g., a table that just has Rank and Salary, where you can lookup a Salary based on Rank)
- An ER diagram that is weakly connected (i.e., Entities are only connected to one or two other Entities through relationships).

B Tier

- Data which breaks down into 5-10 tables and/or consists of a total of more than 100 rows
- Data centralized in two or three main tables, with other tables being support for them
- An ER diagram that is well connected (i.e., some Entities are connected to many other Entities).

A Tier

- Data which breaks down into more than 10 tables and/or consists of a total of more than 1000 rows
- Data which has a low ratio of support tables to main tables
- An ER diagram that is strongly connected (i.e., you can't disconnect it by removing one relationship)

Part 2 – The Database

Part 2.1 – Database Creation

Students must create the database in DB2, including populating it with all the data. Students are encouraged to find code-based ways to add the data records rather than entering them all manually one at a time.

The database created must match the final relational model from Part 1.

Part 2.2 – Database Queries

Students must think of queries which make sense (i.e., answer interesting questions a user might want to know) related to the database, and then write the SQL code necessary to answer those queries. Students must include at least one query which uses GROUP BY, at least one query which uses ORDER BY, and at least one query which uses an aggregate function.

Part 1 Grading Expectations

Technical Components

- Creating a database using DDL
- Populating a database using DDL
- Writing queries in SQL

C Tier

- Fewer than 10 queries, all (or most) of which are ***uninteresting***
    - An ***uninteresting*** query is one which can be easily solved by a human just looking through the data.
        - "Find all Pokemon with an ATK stat over 100" is an example of an uninteresting query
    - Some uninteresting queries are okay (especially if they help set up more interesting ones), but students should try to minimize the amount they use.

B Tier

- Fewer than 10 queries, at least half of which are interesting.
- More than one query which uses GROUP BY, ORDER BY, and/or aggregate functions

A Tier

- More than 10 queries, all (or almost all) of which are interesting.
- More than one query which uses GROUP BY, ORDER BY, and/or aggregate functions

Additional Tips

- Using GROUP BY and aggregate functions is an easy way to add interesting-ness to queries, as they are hard for humans to do on the fly.
- Queries must first and foremost be ***relevant*** to the data and reasonable to ask. A query that is a hundred words long and nested four layers deep might be interesting by the above definition, but it doesn't count if it's so specific or so convoluted that no user would ever reasonably want to know the answer.
- This is not a query optimization course. I don't care if queries aren't as efficient as they could possibly be, as long as they run in a reasonable amount of time (at most a few seconds), that's good enough.

Part 3 – Interaction

Students must create a front-end interface to their database. This can be done in any programming language (Java or Python are recommended) and should provide a user the ability to interact with the database in specific ways. The design of this is up to the student, so long as the following requirements are met:

1. The user cannot freely enter SQL commands (as this would be a security risk in the real world)
2. The user can access the answer to any of the queries the students thought of in Part 2
    a. Given point #1, this must be a selection of some kind (dropdown, buttons, a menu, etc.).
3. The user can request the contents of any table in the database.

Technical Components

- Access a database from a programming language
    - Be able to query the data and return results
- Write a user interface for database interaction

C Tier

- A command-line interface for both input and output
- Above requirements (user cannot enter SQL freely, etc.) satisfied

B Tier

- A command-line, GUI interface, or webpage for input and command line output
- Above requirements (user cannot enter SQL freely, etc.) satisfied
- No SQL visible to the user at all (all queries shown to them in English, not code)

A Tier

- A simple GUI or webpage for both input and output (no interaction on the command line at all)
- Above requirements (user cannot enter SQL freely, etc.) satisfied
- No SQL visible to the user at all (all queries shown to them in English, not code)
- An option for the user to get the output some other way (such as a CSV file with the results of a query)

Additional Tips

- A well-explained interface is good, an intuitive one that requires no explanation is better
- In general, the less work a user must do to get the results they want, the better
    - Clicking a mouse is better than typing, and the fewer clicks, the better.
- The interface does not have to be pretty, but good aesthetics can only help the grade.

Part 4 – Project Report

Students must assemble a short (1-3 pages) written report detailing their project. This report will be the first thing the markers read when evaluating the final project, so it should be a sufficient introduction to the work. At minimum, the report should include all of the following:

- A summary of the data
    - Why was it chosen?
    - What does it consist of?
    - How large is it? (File size, number of records)
- A discussion of the data model
    - Why was it broken it down into those tables?
    - Did students face any difficult choices when deciding on how to set up the model?
        - Tricky participation/cardinality ratio decisions could go here
- A summary of the database
    - List each of the final tables, along with its cardinality and arity
- A list of the queries implemented in Part 2 (and thus available through the interface in Part 3)
    - Students do not need to write out the SQL code, just a text explanation of what the query returns
- A summary of the interface
    - This should include both a description of how it was created (what language was it written in, including any specific libraries) and instructions on how to use it.

Submission

The project will be submitted in several parts at various dates.

1. Part 1.1 will be submitted on Crowdmark and is due on October 31, 2021. See the Crowdmark assignment for details.
2. Part 1.2 will be submitted on Crowdmark and is due November 30, 2021. More details TBA on Crowdmark.
3. All remaining components will be due on December 23, 2021. Details your final submission will be announced later in the term, but will likely consist of:
    a. A copy of or link to your database (Part 2)
    b. A program or link for your database interaction (Part 3)
    c. Your project report (Part 4)