CMSC 399
Senior Seminar

Database Project
Proposal

Isabelle Sanford
isanford@brynmawr.edu

1. Summary

[1 sentence summary]

This project is to build a simple version of a database engine, which can be modified and evaluated easily to understand and test the use of various common database features.

2. Problem Statement

Databases are a vital part of everyday life, and something that many programmers will have to interact with regularly throughout their careers. But modern databases are practically inscrutable, because of the level of complexity required for commercial-level databases (ehhh). This isn't usually a problem for the end user, if the user interface is designed well, but there are people who need to actually understand what's going on and how these things work. That's people like database admins, data scientists, and the database engineers who make it all in the first place.

Education(?) about database internals is analagous to education about operating systems - the theory isn't too hard to teach, but practical examples are more difficult. Looking at the internals of a modern database is too much, like trying to teach highschool level chemistry using a graduate-level book. But teaching individual parts of a database is hard because of how intertwined it is. So what's needed is a small, simple 'model' database engine with clear (and alterable!) internals. The alterable point is important: a professor could have pieces of example code of a model database in lectures, but that's still not practical learning like actually using and changing the code yourself is.

3. Proposed Solution

This project is a simple (though functional) database, designed to have most of the basic features of a modern database, but implemented in a clearer and more naive way. It's written in Go, a language which would be reasonable for this kind of database to be written in.

Features include (but are not limited to; more may be added on as the project progresses):

- File Manager can read and write pages directly to the disk
- In-use pages are cached in a buffer pool
- Changes to the database are logged as they occur, which happens in set distinct transactions
- Records are stored in tables in a such a way that the engine knows where and how they're organized (via a catalog of the tables and their metadata)

- Stat Manager keeps track of statistics about tables (e.g. the number of records) in order to plan queries to be more effective
- The engine can take (simple) SQL input from the user and perform the appropriate relational algebra to return a (correct) result

4. Evaluation Plan

The success of this project, beyond the basic functionality of the database, depends on whether the engine is easy to alter and the alterations are easy to test in order to see results. So the evaluation plan is simply to try a bunch of tests changing the database engine and see how easy it is to do correctly and get results. (TBD: exactly what results this means, but it'll probably include time, number of operations, number of buffer pages used or read/writes, or something along those lines.)

Possible tests are being compiled as the database is built, but potential options include:

- The order in which the buffer manager picks unpinned buffer pages to overwrite (affecting the number of reads/writes)
- The stats (both which stats and how precise(?)) which the stats manager uses to plan queries
- How records are stored in files/tables (e.g. kept inside one block or able to span multiple, fixed or variable length, kept in one file or several)

5. Potential Challenges

Making a database engine is hard!

6. Bio

7. Elevator Pitch