Database Project
Proposal

Isabelle Sanford
Advisor: Geoffrey Towell

1. Summary

This project is to build a simple version of a database engine, which can be modified and evaluated easily in order to understand database internals and test the impact 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 incredibly safe, fast, generalized, and scalable databases. This isn't usually a problem for the end user (and in fact rather a good thing), if the user interface[?] is designed well, but there are also people who need to actually understand what's going on and how these things work. That means (for instance) database admins, data scientists, and the database engineers who make it all in the first place.

Education(?) about database internals is analagous to that of operating systems or compilers - the theory isn't especially hard to teach, but practical examples are fairly difficult. Looking at the internals of a modern database is too much, 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 internals. Even then, this has only limited practicality if just used in lectures, or if the internal structure of the database is visible but cannot be modified. That's not practical learning.

3. Proposed Solution

This project is a simple (but fully(?) functional) database, designed to have most of the basic features of a modern database but implemented in a clearer (and generally more naive) way. The database engine should also have a set of useful metrics for evaluating its own speed and efficiency, so that the effect of modifications is actually measurable.

It's written in Go, which is a fairly standard(?) language for this kind of software[]. Features include (but are not limited to; more may be added on as the project progresses):

- The 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 structures)
- A statistics manager keeps track of numerical metadata about tables (e.g. the number of records) in order to plan queries more effectively
- 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 three factors: how easy the engine is to alter, how easily the alterations can be tested, and how useful the test results are. So the evaluation plan is simply to try a bunch of interesting tests changing the database engine and to see how easy it is to do correctly and get results. (TBD exactly what test results will specifically be, but it'll probably include things like time, number of operations, number of buffer pages used or read/writes, etc.)

Possible interesting tests are being collected as the database is built, but some 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! It's a complex piece of software which is very interconnected, so flaws are hard to track down and painful to fix. Compounding that problem is my lack of experience with Go (e.g. not knowing how testing or logging works and just using print statements everywhere), and even my advisor doesn't have experience with the low-level parts of Go that are important for talking to the disk. It's entirely possible that large portions of this endeavor will be effectively useless (in terms of the major benefits of structuring it this particular way) because the file reading/writing or the buffer pool of pages aren't interfacing with the OS in the intended way, and it would be difficult to determine whether that's the case.

Evaluation will probably be even harder than making the database. It's not obvious how to even structure evaluating the database, nor what measurements to use. Some non-implemented (or incorrectly implemented) feature could make such a difference to the metrics (or just cancel alterations altogether) that changes get lost in the noise. It's hard to tell that apart from a change that actually

has little or no effect. If any of the metrics involve time, the database would probably have to be hosted on a server (which ideally would happen anyway), which could be difficult to implement.

The last challenge is presentation. This project is non-visual, motivated primarily by learning (rather than an end product), and not particularly common or familiar to most people. Saying "I built a database" is near-gibberish for laypeople, and seems like a much less complex task than it is to the average programmer (?). It's difficult to explain the work clearly without being either very vague or much too detailed. Maybe it will get easier over time, and otherwise this will be a challenge nearly as big as the other two.

6. Bio

Isabelle Sanford is a double CS/Math major with a Physics minor who [???]. She started out "coding" in absurdly complex spreadsheet formulas before she even came to Bryn Mawr, and was delighted to discover that computer science is that but better. Her interests still lie in data analysis and visualization, with a particular focus on clearly communicating technical concepts (with e.g. graphs, websites, dashboards). [more?]

7. Elevator Pitch

Oral version (blank parentheses are a pause, with an optional filler noise) Hi, I'm Isabelle! I'm a programmer who does data analysis and () visual stuff like graphs or websites or dashboards or () presenting technical information. I've made visualizations for SpaceX, I've written a graph-making tool, and I'm literally coding a database right now, so I can do pretty much anything data-related and also communicate it clearly.

Written (basically the above but grammatically correct): Hi, I'm Isabelle! I'm a programmer who's really interested in data analysis as well as visual things like graphs and websites, and using them to communicate technical concepts clearly. I've made visualizations for SpaceX, written a graph-making tool, and am now literally coding a database, so I can do pretty much anything data-related you need and communicate it clearly to anyone.