Eric Andrews, Christopher Chapman

CPS377: Database Systems

Professor Justin Brody

Final Project: Empirically Benchmarking Tree-based and Hash-based Database Implementations

For the CPS377 Final Project we plan to develop a C implementation of a Linear Hash Table and a B+-Tree to compare their benefits and downsides. We expect to demonstrate that B+-trees are better at range queries and why hash tables are better optimized for key equality searches.

We will divide the work such that Christopher will write an implementation of a Linear Hash table and Eric will write an implementation of a B+-Tree. These will be written in C, and will each offer four functions: `insert`, `delete`, `keySearch`, and `rangeSearch`. The data structures will be filled with randomly generated "data," using integers as keys with arbitrary data as the associated values. Benchmarking will be done by performing a large number of successive `keySearch` and `rangeSearch` operations on databases of varying sizes, from a couple MB to several GB. We hope to have a consistent, reasonably high-performance environment to run these tests on (e.g. an F&M Linux machine); however, since these tests are all going to be generating comparative rather than absolute data the environment is not of utmost importance. Both implementations will share components like page stucts and page handler.

Both implementations will be written such that the page size can easily be arbitrarily set at compile time. Pages for the individual implementations will vary—the hash table will store one bucket per page, where the B+-tree will store one node per page, thus leading to different page structs within the implementations (the hash bucket struct will have an array member and a single pointer member pointing to overflow buckets; the B+ node struct will have an array of unions of integers (keys) and pointers, with the leaf node pointers being directed at a common page struct). The page struct storing the records themselves will be common to both implementations.

We do not intend to use any significant third-party tools; the C standard library should be sufficient for our purposes. We may need to use some tools in order to mimic page fetching without directly modifying the Linux kernel; this might also be done with a buffer manager.