Eric Andrews, Christopher Chapman

CPS377: Database Systems

Professor Justin Brody

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

GitHub: `https://github.com/CWMChapman/DBMS-Implementation`

Presentation Preferred Date: December 1, 2-5pm

Milestone 1 Report

Page Manager

Done

- Page Manager: the page manager is a simple global struct that logs page I/Os. It does this through the `getPage` and `putPage` functions, which accept a `pageptr` and update the page manager's read/write counters.
- Underlying Structs: the core of the project is three different page structs, the size of which is set by a single pre-processor directive. Through a couple of unions and structs, these are all referred to by a `pageptr` struct containing a pointer and a type identifier; these can then be passed to the `getPage` and `putPage` functions to "read" and "write" from disk (i.e. update I/O counters).

Planned—For Final Submission

Once the data structures are fully implemented and polished, if time permits we will add some simulation of buffer pages. Right now the page manager naïvely assumes that every read and write must go to disk. Further granularity in the data might also be nice—for example, which reads and writes are the cost of traversing the data structure and which are the cost of retrieving the data from memory.

Additionally, the way in which the page manager "stores" on disk is currently very naïve: it tracks `curRecordPage`, and every time a record is added it puts it on that page. When the page is full, it "writes" to disk and makes a new page. This works great so long as you only ever insert items. For the final submission, we will work out a somewhat more sophisticated and efficient solution; if time permits, we may also introduce clustered and unclustered data.

We also plan on developing a `deleteRecord` function which will properly delete a record from a record page and will be used by the B+ Tree and Linear Hashing schemes when deleting records.

B+ Tree

Done

- Initialization and underlying structs. Much of this work was actually done for the page manager.
- Insert. Doesn't quite work yet—there are still some bugs with splitting root—but the basic function is up and running. Hopefully those bugs will be resolved by end-of-day today.

In Progress—For Next Milestone

- Delete and search. Both should be done by next milestone, including both equality and range searches. Work on those will start as soon as insert is reliable.

Planned—For Final Submission

- As currently planned, the B+ tree should be done by Milestone 2; all further work would be on optimizing it, adding nuance to the page manager, and actually gathering data.

Linear Hash

Done

- Initialization and underlying structs for the hash table backend.
- Insert. This works, but the hash table isn't able to split buckets and or increase the number of buckets yet. Overflow buckets are not yet implemented either.
- Search/Lookup. This works, but once again, since I haven't implemented overflow buckets, searching through overflow buckets isn't working yet either.

In Progress—For Next Milestone

- Overflow buckets, splitting and adding hash buckets will be implemented by next milestone (1 week).

Planned—For Final Submission

- Deleting will be implemented by final submission. The are difficulties surrounding deletion because if we remove all the records from a record page, we don't want to have multiple empty record pages in memory. This will likely be implemented by the page manager.
- Actual benchmarking.