DESIGN DOCUMENT

mini project 2

JORDAN SCHRAA + LARA REYES

IMPLEMENTATION

Our main function asks the user to choose 1-6 options, and will keep on giving them the option of choosing until they choose (6) to quit. Each option of searching has it's own seperate function. All functions will output the number of records retrieved and how long it took in microseconds. Essentially, the key search function will try and find the key in the database and from there it will "extract" the value using the key. The data search function will do the reverse. Using the data input by the user, it will try and find the key pair. For btree and hash, it searches for it in linear way. As for trying to find data in a range, we implemented it as the indexfile uses the btree implementation which starts at the bottom and appends all data till you get to the top. As for using hash for our range search, it is a sequential search.

ANALYSIS

From what we've recorded, we've found that overall, all database types are mostly all faster when searching for the key, which makes sense because we've learned that that's how databases are implemented. It's a bit odd though because the data search for our index file database type is faster by 2 milliseconds, which isn't a whole lot and it also differs each time but this is what we came up with on our first complete run. Range search was quite costly in all database types, but we expected that since it has to get a whole range and append it each time it finds something in the range. The hash table took an extreme amount of time finding the records in a specified range. As for data search, it was very sporatic compared to key search and range search. The hash table seemed to be the fastest using the data search and index file was not too far behind in terms of time, but the btree really took a toll and was similar to the time that the hashtable took when it was trying to search within a range.

EXPERIMENTAL RESULTS

Queries	B-Tree	Hash Table	Index File
Key Search	108 ms	123 ms	109 ms
Range Search	65003 ms	373574 ms	65554 ms
Data Search	378423 ms	99 ms	107 ms