Exploration of the Efficiency of Data Structures with B-Trees, Hash Tables, and Index Files.

Devon Upton (doupton)
Bennett Hreherchuk (hreherch)

Summary

Using Berkeley DB (BDB) we wanted to determine the efficiency of various data structures that BDB has to offer, focusing on B-Trees, Hash Tables, and the added efficiency of using an index file (a secondary file with data as the keys and keys as the data values) with these structures. We measured three types of searches and the time it took for each data structure to execute them. The three searches were:

1) Key Search: Search for a given key's point of data in the database.
2) Data Search: Search for the keys that have the matching data.
3) Range Search: Search for the data identified by a range of keys.

Purpose

To explore what the strengths of various database types are with respect to typical queries run on databases.

Method

We used python3 and the bsddb3 library for BDB implementation in our python file called *mydbtest.py*. We created a terminal interface that could build a database based on the command line argument and allow for the three types of searches based on stdin input from the terminal.

For the testing purposes we modified the code of *mydbtest.py* and created a python program *experiment.py* to run tests and automatically compute the average time for each query. The testing values we chose for the project were randomly selected once and hardcoded into our testing program.

For each query and each data structure we tested the four random queries (of either key, data, or a range that produced 100-200 results) to the database and took the average time of the tests, collecting the results and writing it into table 1 below.

Result

Table 1: Queries and their time taken with various data structures
(in microseconds)

	Data Structure		
Query Type	B-Tree	Hash Table	Index File
Key Search	115	112	110
Data Search	590470	664654	146
Range Search	1438	710923	1625

Discussion

From table 1 it is clear to see that Hash tables excel at key searching while failing to be as efficient as other data structures in the other two queries.

B-Trees are also efficient in key searching, while being quick at range searching. However, it appears that B-Trees also fail to produce data search results in a reasonable time.

For the index file, we found that a B-Tree primary database associated with a secondary hashing database as the index file produced very reasonable results for all three queries. We used the index file only for data searching, while maintaining use of the primary B-Tree database for key searching and range searching. As a result, the setup with the index file caused data searching to be thousands of times faster than just B-Tree or hash tables alone. We chose a hash table for the secondary database because hash tables have constant time in accessing keys (which would be the data in this case) giving us the theoretical fastest time.

Conclusion

The most well rounded database from our testing appears to be the B-Tree with the hashing index file as it can perform all three queries faster than a B-Tree or hash table alone, with respect to data search at least. While this is good, we never took into account the memory costs, size of the databases, or the costs of insertion, deletion, and maintenance which could be other indicators of which database style would fit general use purposes.