**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 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*. 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* to allow for input of the keys/data through function calls and had the functions return the time it took for the search to complete. With this we could create a dictionary of database entries as it was generated then randomly select keys and datum to use with testing.

For each query and each data structure we tested 4 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)** |
| |  |  |  |  | | --- | --- | --- | --- | | Queries | BTree | Hash Table | Index File | | Key Search |  |  |  | | Data Search |  |  |  | | Range Search |  |  |  | |

**Discussion**

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

B-Trees also succeed in key searching, while being quick at range searching. However, it appears that B-Trees also fails 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 hash 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 B-Tree 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.

**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 to use.