CMPUT291

Project 2

Design Report

By

Zhaorui (Teppie) CHEN

Zhengyang (Nicholas) LI

Jiaxuan YUE

Introduction

This report introduces the design of a database which uses various file structures including B-Tree, Hash Table, and Index File, with different search methods. In the report the speed of each structure-method combination is listed. The reason why Index File improves searching speed is briefly discussed.

Method

As it’s mentioned in introduction, the script has three different file structures. The user will need to declare which structure he or she would use to build the database by typing btree, hash, or index as command line arguments. The implementation of B-Tree and Hash is almost the same. In key search method, the program asks for a valid key input and directly search the corresponding value by Berkeley DB get() method. In data search method, the program asks for a valid data input and compare it with all values in data until it is matched. In range search method, the program asks for a valid range of characters, traverse the whole database and put all key-value pair in between the given range into a list.

The implementation of Index File is different from the previous ones in data search method and range search method. In this structure we build two hash table structure databases, one exactly the same as previous one (here we call it *original database*), while the other one is built based on data (here we call it *second database*). For key search method, the program can directly retrieve the data using get() method in *original database* as well as the previous structures. For data search method, since the second data base is built based on data, we can retrieve the key using get() method in *second database*. And lastly for range search method, the program first built up a cursor in the *original database* and find out the smallest key (here we call it *SK*) greater than or equal to lower bound. If this *SK* is lower than the upper bound, the program will let it to be the new lower bound and repeat this procedure until the *SK* is over input higher bound

.

Result

We took five test cases for each kind of searching method. The test result is shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | TEST CASES | B-Tree | Hash Table | Index File |
|  | teppie | 54 | 41 | 57 |
| Retrieve | nicholas | 52 | 41 | 58 |
| By | jiaxuan | 56 | 43 | 61 |
| Key | qwerty | 54 | 53 | 59 |
|  | wasd | 56 | 44 | 58 |
| ave |  | 54.4 | 44.4 | 58.6 |
|  | chen | 236111 | 232951 | 52 |
| Retrieve | li | 237077 | 231287 | 62 |
| By | yue | 237729 | 233073 | 64 |
| Data | abcdefghi | 235551 | 230928 | 53 |
|  | udlr | 235185 | 228297 | 51 |
| ave |  | 236330.6 | 231307.2 | 56.4 |
|  | o-t | 134390 | 147768 | 27090 |
| Retrieve | h-o | 174478 | 179427 | 30573 |
| By | a-y | 91558 | 111134 | 111890 |
| Range | t-y | 102824 | 122529 | 23159 |
|  | n-t | 135263 | 148094 | 27455 |
| ave |  | 127702.6 | 141790.4 | 44033.4 |

**Table 1. Test result**

Discussion

From the result table we could see the speed improvement is quite obvious.

Conclusion