COMPARATIVE STUDY ON DATA SEARCHING IN LINKED LIST & B-TREE AND B+TREE TECHNIQUES

AHMED ESHTEWI S GIUMA

A dissertation submitted in partial
fulfillment of the requirement for the award of the
Degree of Master of Computer Science (Software Engineering)

The Department of Software Engineering
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

MARCH 2015

DEDICATION

Specially dedicated to…

My parents and my wife, who are always there during the most challenging times. My friends, for the support and help throughout the projects. And to the unsung persons who are involved directly or indirectly.
May God bless us always.

ACKNOWLEDGEMENT

First and foremost I thank God for the strength and courage that had made this humble effort a reality.

I would like to express my deepest gratitude to my final year project supervisor Dr. Mohd Zainuri Saringat for his invaluable advices and guidance throughout this project. His profound knowledge, ideas and support keeps on motivating me to give my all for this project.

I wish to thank all my friends, staff and to those who has directly or indirectly guided and helped me in this project. The knowledge and support that they shared with me will always be remembered.

Lastly, and most importantly I wish to dedicate my appreciation to my beloved father, mother and brothers for always being there for me all these years. Thanks for their unconditional love, encouragement, and support Universiti Tun Hussein Onn Malaysia (UTHM) is also gratefully acknowledged.

ABSTRACT

There are many methods of searching large amount of data to find one particular piece of information. Such as finding the name of a person in a mobile phone record. Certain methods of organizing data make the search process more efficient. The objective of these methods is to find the element with the least time. In this study, the focus is on time of search in large databases, which is considered an important factor in the success of the search. The goal is choosing the appropriate search techniques to test the time of access to data in the database and what is the ratio difference between them. Three search techniques are used in this work namely; linked list, B-tree, and B+ tree. A comparison analysis is conducted using five case databases studies. Experimental results reveal that after the average times for each search algorithms on the databases have been recorded, the linked list requires lots of time during search process, with B+ tree producing significantly low times. Based on these results, it is clear that searching in B- tree is faster than linked list at a ratio of (1: 5). The searching time in a B+ tree is faster than B- tree at the ratio of (1: 2). The searching time in a B+ tree is faster than linked list at the ratio of (1: 8). With that, it can be concluded that B+ tree is the fastest technique for data access.

ABSTRAK

Terdapat banyak kaedah dalam pencarian suatu maklumat dari satu kumpulan data yang banyak. Contohnya seperti mencari nama dalam telefon bimbit. Sestengah kaedah menguruskan data bagi menjadikan proses pencarian lebih efisien. Objektif kaedah yang dibincangan adalah untuk mencari data dengan cepat. Dalam kajian ini, tumpuan kajian adalah pada masa carian dalam pengkalan data yang besar dimana ia adalah satu factor penting dalam menentukan kejayaan dalam carian. Matlamatnya adalah memilih teknik yang paling sesuai dalam carian data didalam pengkalan data dan perbandingan dalam peratus masa capaian diantara teknik teknik tersebut. Tiga jenis carian dikaji iaitu linked list, B-tree dan B+ tree satu analisa perbandingan dibuat dengan menggunakan lima kajian kes. Hasil kajian telah laporkan dimana linked list memerlukan banyak masa dalam carian berbanding B+ tree. Berdasarkan keputusan ini telah menunjukkan carian dalam B- tree adalah pantas berbanding linked list dengan kadar (1:5). Carian masa dalam B+ tree adalah lebih baik berbanding linked list dengan kadar (1:2). Sementara itu carian masa dalam B+ tree adalah lebih laju berbanding linked list dengan nisbah (1:8). Dengan itu, dapatlah dirumuskan B+ tree adalah teknik yang paling laju dalam capaian data.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Data is defined as a set of valuable information with certain similarities, which is usually sorted in such way where it may be easily retrieved by other relevant parties. The Internet or a library is a storage facility providing avenue for the accessibility of data, and such storages are known as databases. Every organization deals with a series of databases respectively. For instance, the police may have a database of criminal records, where a car showroom would have a database of vehicle history. The size of the database directly affects the effectiveness in searching the data. Thus, every data should be traced via a database, based on the following criteria:

(i) Ability to search for a specific item.
(ii) Ability to search for related items to a known item.
(iii) Ability to search in a specific field or fields.
(iv) Ability to combine search terms using Boolean logic.

The most noticeable problem in the world of computer science and information technology would be the storage and retrieval of data. There are applications and search engines which are capable to access a large virtual database in a short period of time. Nevertheless, the scope of the hits on the desired data might be large, to an extent that the user still cannot find what he/she is looking for. However, there are certain infrastructures applicable for retrieval of data efficiently. The most common search structure would be the multi way balanced B-tree. As the name suggests, it consists of leaf and internal, or also known as the nodes. The internal nodes are basically the trace index to the leaf nodes, whereas the leaf nodes are the data carrier. As for this infrastructure is by far the most effective method in the maintenance of disk data (Askitis *et al.,* 2009).

Other search structures exist namely, the linked list and the B+ tree described in the following paragraphs.

In the context of computer science, linked list are a structured data, used in retrieval of sequential objects, allowing flexibility to add or remove intermediate elements in the sequence. Instead of having a series of arrays, linked list consists of nodes, that stores value and reference of the next node. Though the insertion and removal of nodes are fast, the access to the elements could be slow since in order to access node ten, the link would go through the first nine nodes if no removals were made. Random access elements on the other hand are accessed arbitrarily (John Wiley & Sons, 2010).

A B+ tree consists of a root, which may be a leaf or a node with more than two children, in where the actual number of children for a node is denoted as *m*. The root is an exception. The primary value of a B+ tree is in the stored data for efficient retrieval in a block-oriented storage context such as the file systems. Unlike the binary search trees, B+ trees have high fan outs or pointers to children nodes in a certain node (Navathe *et al.*, 2010).

1.2 Problem Statement

One of the problems that faces large databases users is the noticeable lateness of data retrieval which can lead to boredom and the loss of user's time by waiting for the completion of data access and retrieval process. In order to minimize the searching time and the loss of the data, many of the programmers and developers of software engineering development have designed several techniques that can help to increase the searching speed and also provide a good compromise for databases users. Developers have developed many of the algorithms that do the searching process and all the work to achieve the fastest time in the data retrieval process. But there is a difference between these algorithms in terms of speed, there are high-speed algorithms and other medium-speed and slow speed. That make databases designers find it difficult to determine which algorithm is faster. Because of that researchers have compared between many of the techniques used in order to determine the fastest technique and facilitate the selection of any appropriate algorithm in the search process. In this research comparative study will be conduct on the three algorithms (linked list, B-tree and B + tree) to determine the fastest and also to determine the percentage difference between the three algorithms. In this study research five different sized databases will be used in order to get more accurate results.

1.3 **Project Objectives**

The objectives of this research are summarized as follow:

(i) To develop and implement linked list, B-tree and B+tree by using one of the programming languages.

(ii) To compare the three proposed techniques using the five case studies depending on the different sizes of the data.

(iii) To evaluate and analysis results based on time and identify any faster technique , and calculate the amount of the difference between them.

1.3 Project Scope

This research focuses on the problem of time search in databases. Therefore, linked list, B-tree and B+tree techniques will be used to test the speed of access to data in the database and will be compared using the five case studies.

1.4 Outline of the Report

This research consists of five chapters. Chapter 1 is an overview of the project and the main objectives of the project. It consists of the scope of work covered and the project's objectives. Chapter 2 illustrates the literature review of the project. It also gives a brief explanation in general information about automated testing for database system in this project. Chapter 3 discusses the methodology used to obtain the entire objectives of this project and tools. Chapter 4 explains the implementation and the detailed steps in this work as well as the results and discussion. Chapter 5 includes the objectives achieved, disadvantages, future work, and conclusion of the project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introductions

Historically, memory limit was restricted, so extensive information accumulations must be put away on databases, which utilize information structures, for example, linked list and B-trees. With the accessibility of expansive memories, this confinement has been loose. Correspondingly, various new requisitions have risen in such fields as bio-informatics and computational semantics that oblige looking immense accumulations in memory. A B-tree-like information structure implicit memory is still a great answer for such issues (Helen, 2011).

Nodes are arranged in a certain way that they communicate sequentially in a linked list. In a basic structure, under the least complex structure, every previous node acts as a predecessor of the current node, and every current node acts as a successor of the previous node. Removal and addition of nodes are dynamic, where it could be done from any point in the list.

Connected records are easily comparable as they store information beneficial to the customer. A similar structure of connected records would store the similar type of data. The interchange methodologies and the functionality of connected records would be a good research to conduct on (Nick, 2010).

A linked list stockpiling is effective in such way that a client does not have to worry about the relevancy of data acquired. Linked list rundown information stockpiling is where the information are retrieved haphazardly. The incorporation of linked list in corresponding channels, organization of binary trees, stack building, queues in programming, and overseeing social databases creates an ease in access.

The exhibits are the most widely recognized information structure used to store data. Mostly, clusters are helpful in terms of linguistic assistance in getting to any component via its record number (Nick, 2010).

B-tree is a tree information structure that keeps information sorted, where logarithmic insertions and cancellations are easy. The B-tree is a generalization of a binary inquiry tree in that a node can have more than two branches. Unlike the common tree structures, the B-tree have improved framework and composes numerous information. It is commonly used in databases and document frameworks. It is an effective method in placing and retrieving records in a database. However, the significance of the alphabet B has not been theoretically expressed. The B-tree calculation saves time since a medium exist to run through the existing records, with a fast moving algorithm (Margaret, 2009).

2.2 Data Structure of Linked Lists

Linked lists consists of data and link. Via the link, each data element contains location information about the next immediate element. The index name is basically the pointer variable name in the linked list. The following Figure 2.1 illustrates a linked list, addressed as scores, which consists of four elements. An example of an empty linked list, or a null pointer is shown in Figure 2.1.

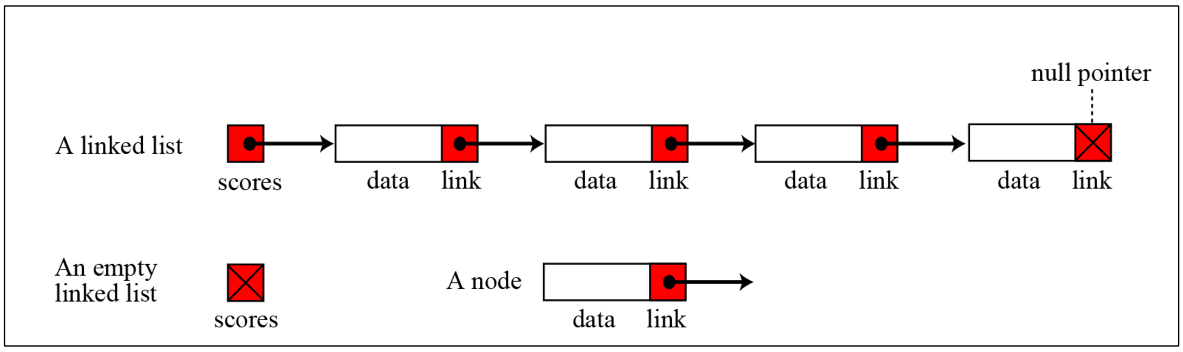

Figure 2.1: Linked Lists (Behrouz & Firouz, 2008)

Each linked list should be named in such way that it could be differentiated from the elements and the nodes itself. Figure 2.2 displays the name of a selected linked list, which is the head pointer that directs the link to the first node in the linked list. A node would only have implicit rather than explicit name (Behrouz & Firouz, 2008).

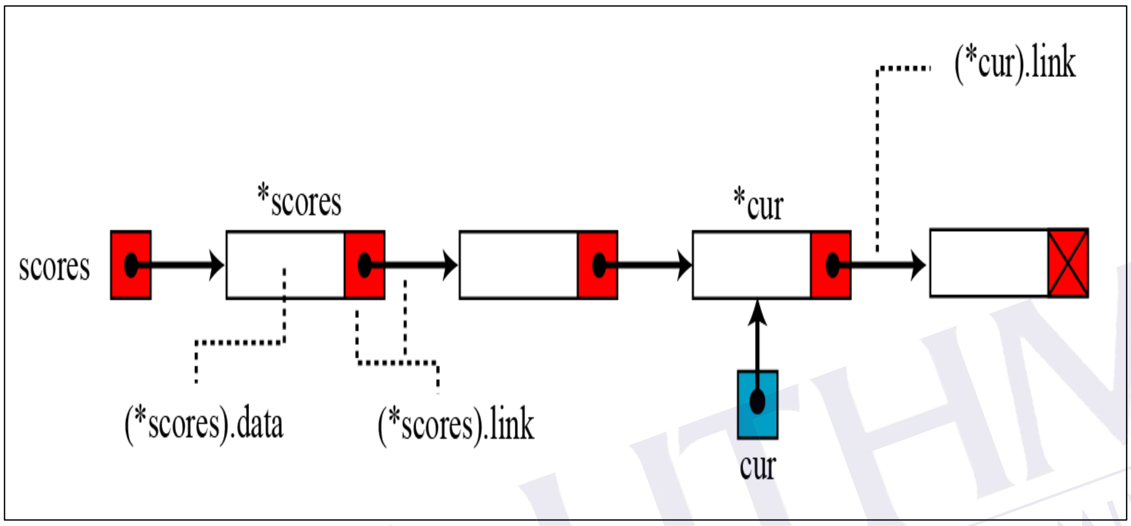

Figure 2.2: The Name of a Linked List Versus The Names of Nodes

2.2.1 Searching in Linked List

Two separate pointers, known as previous (pre) and current (cur) are used in nodes. In the initial stage of a search, the pre pointer would be null, whereas the cur pointer would be linked to the first node of the link. The algorithm of this search structure links these two pointers all the way towards the end of the list. If the target value is bigger than the values in the entire list, the movement of the pointers would be slow. Figure 2.3 illustrates a linked list search algorithm with the pre and cur pointers (Behrouz & Firouz, 2008).

```
Search algorithm for linked list

Algorithm : Search linked list (target, list)
Purpose : Search the list using two pointers: per and cur
Post : None
Per : The linked list (head pointer ) and target value
Return : the position of per and cur pointers and the
value of the flag (true or false )
{
Per ←  null The previous value= null
Cur ←  list Current value
While (target  < (*cur).data )
{
Per ←  cur Cur ←  (*cur).link
}
     If the Current value = flag= true
If ((*cur).data=target ) flag ← true
Else flag ← false
Return (cur ,per ,flag)
}
```

Figure 2.3: Search Algorithm for Linked List (Behrouz & Firouz, 2008)

2.2.2 Advantages and Disadvantages for Linked List

In simple terms, linked lists are a basic chain containing nodes or data, linked via pointers that points the current data towards the next data.

2.2.2.1 Advantages

All data linked in the list are from the similar group or search field. These are some advantages of linked list:

(i) The information structure consumes low external memory during run time as it is a real time system.

REFERENCES

Achakeev, Daniar, & Bernhard Seeger. (2013). "Efficient bulk updates on multiversion B-trees." *Proceedings of the VLDB Endowment*, 6(14), pp. 1834-1845.

Askitis, N., Zobel, J. (2009). B-tries for disk-based string management. *VLDB J. 18*, pp.157–179.

Behrouz Forouzan & Firouz Mosharraf. (2008). Foundations of Computer Science, 2nd edition, Thomson Learning. UK. pp. 11.27-11.50.

Blevins, Jason R. (2009). "A generic linked list implementation in Fortran 95". *ACM SIGPLAN Fortran Forum. Vol. 28. No. 3.*

Braginsky, Anastasia & Erez Petrank. (2012). "A lock-free b+ tree." *Proceedings of* the 24th *ACM symposium on Parallelism in algorithms and architectures.*

Helen A. (2011). "The universal B-Tree for multidimensional indexing. General concepts," *World Wide Computing and its Applications*, pp. 198–209.

John Wiley & Sons.(2010). Horstmann, Cay S. Java Concepts: Compatible with Java 5, 6 and 7, *Congress Cataloging.* USA, pp. 630-631.

Lefteris Kellis & Dani Mart. (2013). B- Tree. *Laxmi Publications*, pp.10.

Margaret Rouse. (2009). "The ubiquitous B-tree," *ACM Computing Surveys*, vol. 11, no. 2, pp. 121–137.

Nick Parlante. (2010). "Linked List Problems". *Acta Informatica*, vol. 9, pp. 1–21.

Prabhakar Gupta & Vineet. (2010). Design and analysis of algorithms. *PHI Learning Private Limited*, pp.170–171.

Ramez Elmasri & Shamkant B. (2010). Fundamentals of database systems (6th ed).*Upper Saddle River, N.J. Pearson Education*, pp. 652–660.

Rize, Jin, Hyung-Ju Cho & Tae-Sun Chung. (2013). "A group round robin based b-tree index storage scheme for flash memory devices." Proceedings of the 8th *International Conference on Ubiquitous Information Management and Communication.* ACM.

Satinder, Bal. Gupta & Aditya Mittal. (2009). Introduction to Database Management System. *Laxmi Publications*, pp. 67.

Thomas Cormen, Charles Leiserson, Ronald Rivest, & Clifford Stein. (2009). Introduction to Algorithms. (3rd ed). *MIT press. USA*, pp. *441*.

Timnat, Shahar, Alex Kogan & Erez Petrank. (2012). "Wait-free linked-lists." *Principles of Distributed Systems. Springer Berlin Heidelberg, pp.* 330-344.

Timnat, Shahar & Erez Petrank. (2014). "A practical wait-free simulation for lock-free data structures." *Proceedings of the 19th ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM.

Yuxing, Zhu & Jun Gong. (2014). "A real-time trajectory indexing method based on MongoDB." Fuzzy Systems and Knowledge Discovery (FSKD), 11th *International Conference on. IEEE.*