Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

59

Edward Hill, Jr.

# A Comparative Study of Very Large Data Bases

Springer-Verlag
Berlin Heidelberg New York

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

59

Edward Hill, Jr.

# A Comparative Study of Very Large Data Bases

Springer-Verlag
Berlin Heidelberg New York 1978

**Author**
Dr. Edward Hill, Jr.
Division of Computer Research
and Technology
Building 12 A, Room 2041 B
National Institute of Health
9000 Wisconsin Avenue
Bethesda, Maryland 20851/USA

AMS Subject Classifications (1970): 68-02, 68A20, 68A50, 68A99
CR Subject Classifications (1974): 4.33

ISBN 3-540-08653-6 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-08653-6 Springer-Verlag New York Heidelberg Berlin

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2141/3140-543210

# PREFACE

This monograph presents a comparison of methods for organizing very large amounts of stored data called a very large data base to facilitate fast retrieval of desired information on direct access storage devices. In a very large data base involving retrieval and updating, the major factor of immediate concern is the average number of accesses to the direct access storage device to complete a request. The average number of accesses to store and retrieve items on a direct access storage device for hashing methods using chaining with separate lists and linear probing is presented. A new algorithm and performance measures for chaining with coalescing lists is presented. New performance measures are presented for storing and retrieving with a binary search tree and a trie stored on a direct access storage device. Algorithms are presented to perform retrieval, insertion, deletion and the inverted file generation operations for an inverted file. New performance measures are presented for an inverted file. The methods are developed using a component concept. A hybrid method involving components is used for the linked files. All methods are analyzed, along with their data structures, to show their effect on the average number of accesses to the direct access storage device while processing a request. Finally, a comparison criterion is developed and each method is compared.

This monograph is based on a D.Sc. dissertation submitted to the Department of Electrical Engineering and Computer Science at The George Washington University in 1977.

## ACKNOWLEDGEMENTS

I am indebted to many people who have contributed in a variety of ways to the completion of this monograph.

First of all, I owe a great deal to Prof. A. C. Meltzer who served as my advisor. Prof. A. C. Meltzer deserves thanks for encouraging me to investigate what has now become the topic of this monograph.

Thanks to Dr. Eugene K. Harris and the Division of Computer Research and Technology of the National Institutes of Health, for their support in my studies in Computer Science.

I am indebted to my wife, Espor and my children Eurica and Edward for their love, understanding and support during the many hours this work has kept me from them.

Finally, I am indeed thankful for my parents Edward and Gertrude who sacrificed so much to aid me during the initial years of my career.

TABLE OF CONTENTS

CHAPTER 1

# INTRODUCTION

This monograph is a comprehensive investigation of methodology for organizing very large amounts of stored data called a very large data base. This investigation surveys the existing methods and presents a new unified notation and approach for evaluation and comparison of these methods. New storage and retrieval algorithms are designed and developed and performance measures for all methods are established. Comprehensive performance evaluations are carried out for each method as a function of critical design parameters. Comprehensive tables and graphs are presented which permit a direct comparison of method performance. The methodology studied here is designed to facilitate rapid storage and retrieval of information which is stored on a direct access storage device.

In a very large data base involving retrieval and updating, the principle concern is the average number of accesses to the direct access storage device to complete a request. Methods and their associated data structures, are analyzed to show their effect on the average number of accesses to the direct access storage device required to process a request. A comparison criterion is developed and each method is analyzed for a measure of its performance.

The size of a data base may be characterized by the number of entities it concerns and the average number of retrieval terms that apply to information about each entity. A data base in which all pointers, lists and indices reside on a disk is called a _very large data base_.

Very large data bases are justified only in very large systems, involving many users using large computer complexes and networks. The analysis presented here is concerned with the problem of designing large systems for processing data bases.

High efficency in processing, storage usage, and retrieval is extremely important in processing a very large data base. An inefficient organization of the data base may account for a very large number of disk accesses and result in impractical processing times. The number of disk accesses is strongly connected to the data base data structure and its processing algorithms. The proper data structure and search algorithms will reduce the number of disk accesses, initial load time and data update time.

The approach taken in this monograph is one of synthesis. The simple component parts of various search techniques and their associated data structure are analyzed. These components are used to compare structures for various component organizations. The primary components are files of physical records and addressing mechanisms used to locate records.

The purpose of Chapter 2 is to introduce a set of concepts which is fundamental in defining both a data structure and a search structure for very large data bases. To compare very large data bases, it is essential that each component of the system be well defined. Chapter 2 introduces definitions that give the structures of the data base precise meaning.

Chapter 3 introduces the necessary direct access storage device terminology to analyze algorithms. Since the data base and all pointers are stored on a direct access storage device these terms influence the implementation of any algorithm to search and store records in the data base.

Chapter 4 introduces record processing methods using hashing. The methods analyzed are chaining with separate lists, chaining with coalescing lists and linear probing, denoted by CS, CC and LP respectively. A new search algorithm using coalescing list is introduced and analyzed. New performance measures are presented for chaining with coalescing lists using a general bucket size. A mixture probability distribution is developed to apply the 80-20 rule using the performance measures.

Record processing using tree methods are introduced in Chapter 5. Tree methods are summarized. Algorithms are presented for TREE and TRIE. New performance measures are presented for the TREE and the TRIE. The notation for the TREE and the TRIE is BS and T respectively.

Chapter 6 introduces record processing using linked files, denoted by LL. The component approach used in this monograph is demonstrated. The directory search is by linear probing and the list of files are organized using the chaining with separate list method.

Record processing using an inverted file structure is presented in Chapter 7. The notation used for the inverted file is IF. Algorithms are presented and analyzed for the retrieval, insertion, deletion and inverted file generation operations. New performance measures are presented for an inverted file.

In chapter 8 a criterion for comparison of large data bases is presented. The concepts defined in earlier chapters are used to define various attributes of the comparison criterion. The attribute list is used to define a comparison operator. This comparison operator defines precisely the methods, distribution, relation and other attributes in every comparison.

Chapter 9 presents the conclusions which have been reached on the basis of this research.

CHAPTER 2

# DATA BASE STRUCTURE

This chapter defines the record structures and data structures for the data bases used in this monograph. A collection of data, organized in some fashion, is called a data base. Within a data base both organization and content assume importance. Organization of a database consist of the grouping of records and the ordering of records within those groups. This is done to increase the chance of locating known data. It is not enough to know that a particular data item may be in the data base; one must also know how to find it.

Search algorithms are used to locate records in data bases. Two factors that affect search algorithms are the record structure and the data structure of the data base. Basic definitions for a record structure and data structure are presented in this chapter. This is done to explicitly define the notions that are used in later chapters.

## 2.1. Record Structure

The basic unit which is processed in a data processing system is called an _item or record_. An item is made up of two parts: the key and the data. A _key_ K is that part of a record that distinguishes the record from all other records. The _length of the_ key is the number of digits or characters in the key. The data is that part of a record which is not the key. A set of records is a file, and a set of files is called a data base.

## 2.2. Data Structure

The structures discussed here are those relationships that exist between records and files in a data base. Many definitions used are modifications of those in references [10, 20, 35].

The basic unit in a data structure is called a node. A node may consist of a key, data, record, file, or a data base. A structure that involves only linear relative positions of the nodes is called a linear list. When the list nodes are modified in a way such that one node contains the address of its successor in addition to its normal content , the list is a linked list and the address is called a pointer. A linked list contains a special node to indicate the last node in the list called the termination node. Most lists are allocated from a linked area of storage called an available storage pool. Therefore, every list must have a pointer to the beginning of the list. A variable (called a link variable) points to the first node of a list and contains the address of the first node of the list. An example of a linked list with a link variable called FIRST is illustrated in figure 2.1.

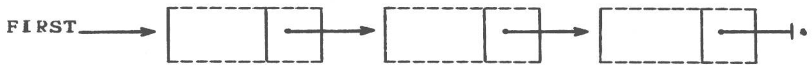

Figure 2.1. Linked List.

A special node in a list located at the beginning of the list, is called a list_head. Many times it is necessary to organize a list where in the last node points to the first node. Such a list is called a circular_linked_list. In a circular linked list, sometimes called a ring_list, it is a common pratice to include a list head node. The list head node is useful in many search algorithms using lists, because it indicates a node with known attributes.

Often, it is necessary to have two pointers in each node. A list with two pointers in each node, where one points to the node predecessor called the left_link (LLINK) and the other points to the node successor called the right_link (RLINK), is called a double linked_list. An example of this structure is presented in figure 2.2.

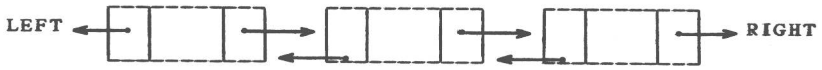

Figure 2.2. Doubly Linked Lists.

For large files the lists tend to be long, and extended searches may be required if the list length is not controlled. In all lists discussed so far the list lengths were unrestricted and each list had one starting point. By restricting the list