OPPORTUNITIES FOR DATA BASE REORGANIZATION

Ben Shneiderman
Department of Computer Science
Indiana University
Bloomington, Indiana 47401

Introduction

Reorganizing the clutter on our desks is a task which each of us would like to have someone else do for us every day, but which we put off for as long as possible. By sorting out the old and worthless items and neatly ordering the current and useful items we can make it easier to locate what we need.

We face the same dilemma with our personal address and telephone books. Old entries become useless, and the scratched out and written over entries make it increasingly difficult to find what we want. Eventually we purchase a new phone book, probably of the same size, and rewrite the useful entries. The old address book can now be disposed of or saved as backup and for historical purposes.

If the contents of the data base are volatile, automating the process does not eliminate the need for reorganization. In fact, the minor inefficiencies which could be tolerated with small files are a costly burden in the massive data bases which are increasingly common. There are numerous techniques for improving data base performance; this paper is concerned with the process known as reorganization. Reorganization is the rewriting of fields of data in such a way that the logical and physical accessing algorithms are unaffected (although the access paths may be altered) except that they now operate more economically. Economics include saving of storage space or execution time or both. In on-line systems there is the additional motivation of reducing the response time.

An earlier mathematically oriented paper (1) focused on the issue of selecting the optimum points for performing a reorganization. The problem is that we would like to have this file in a reorganized form continously, yet we wish to delay the reorganization process for as long as possible since it is so costly. High reorganization cost is a result of the fact that in most cases the entire file must be read and written.

The task of recognizing the optimum reorganization point is difficult since there are a large number of interdependent factors which influence the decision. Relevant considerations include:

* average cost per search
* number of searches per unit time
* number of additions per unit time
* number of deletions per unit time
* reorganization cost

Opportunities For Data Base Reorganization

* reduction in search cost as a result of reorganization
* length of time the data base is expected to remain in use

Since the first six items are functions of time, rather than stable constraints, the task is even more unwieldly. A number of exogenous factors, such as response time constraints for on-line systems, availability of excess computing power during night or weekend shifts and demand for backup copies at fixed time intervals, can further complicate the problem.

By making reasonable assumptions useful analytical results can be obtained (1) . Although fixed time interval reorganizations can be implemented, it is intuitively clear that a stochastic model permitting variable reorganization is preferable. In episodes of high utilization reorganization should take place more frequently than during periods of relative inactivity. Under this model the reorganization takes place when the cost of doing a search reaches a specified level. If the conditions do not permit mathematical simplifications then simulation or even experimentation are the sole recourse.

The problem can be seen in a more familiar format. Deciding when to reorganize is akin to deciding when to buy a new car. The old car continues to cost more per mile (more per search) due to rising repair costs and decreasing mile-per-gallon performance, yet the cost of purchasing a new car (reorganizing) is so high as to prevent our doing so for the longest time. The new car would hopefully have reduced repair bills and improved efficiency thereby reducing the cost per mile. Factors influencing our decision include the current cost per mile, the expected improvement, the cost of a new car, the utilization and a host of other factors. The fact that we can make any decision at all is remarkable!

With cars we have the feedback information of the cost measured in our daily expenditures. With a data base, the cost is hidden and can be enormous. System designers and maintainers (dare I say data base administrators?) have reported to me that their methods for determining whether a reorganization was appropriate included listening to the disk drive to hear if the arm was moving "frequently" or waiting till they got annoyed at the length of terminal response time. Others who rigidly adhere to a fixed time period between reorganizations readily admit to a substantial variation in system utilization.

Whatever the scheme for selecting the reorganization points, the data base maintenance functionary must first acquire the appropriate performance statistics. This is non-trival since the relevant considerations listed earlier are not readily measurable. The average cost per search and the reorganization cost as a function of the size of the data base or the number of overflows may be difficult to obtain. Although a number of systems provide track

utilization information, statistics on number of records, overflows, etc., none relate these values to cost functions. The best source is performance data taken while the system is actually running. Special benchmark runs, called sampling runs or software probes, can be used to collect these statistics. If the burden of software monitoring is large, then hardware monitoring of channels and auxiliary storage devices may suffice.

Structures Warranting Reorganization

In the past, the volatility of data base systems, changing administrative demands, new implementation techniques and the lack of centralization of the data base administrative function have relegated the problem of reorganization to a secondary consideration. Now, the increasing maturity of data base systems, the growing stability of application and administrative goals and the creation of a central focus at the data base administrator have raised issues of performance, measurement, efficient use of system resources and reduction of costs.

While the data base administrator is responsible for efficient maintenance, it is the responsibility of the system architect/designer to select structures which permit reorganization. Although the variety of structures which permit reorganization is large, most of the patterns can be interpreted in one of three ways: access path reorganization, block overflow reorganization or linked reorganization.

Access path reorganization: In this method the access paths are altered, but the accessing algorithms remain unchanged. The prime example is the reorganization of a binary search tree (2) to reduce the average search length for retrieval of a node (see Figure 1). In this case the average search length is reduced from 6.67 to 3.27 nodes. The symmetric order traversal of the tree remains unchanged and the usual binary tree search algorithm still functions, only faster. No data space is reclaimed.

A second example would be the splitting of a long list into several shorter lists to reduce search time.

Collecting the vital statistics and converting to the values of relevant variables can be tricky, but the payoff in higher performance can be substantial.

Block overflow reorganization: The simple case of block overflow reorganization in Figure 2 depends on the volatility of the data for its success. Deleted nodes, indicated in black, are clogging the blocks and causing new entries to be allocated in new blocks, causing unnecessary access to be made. By eliminating the useless data and collecting records from overflow blocks it is possible to reduce or eliminate overflows. This situation arises in hash coded

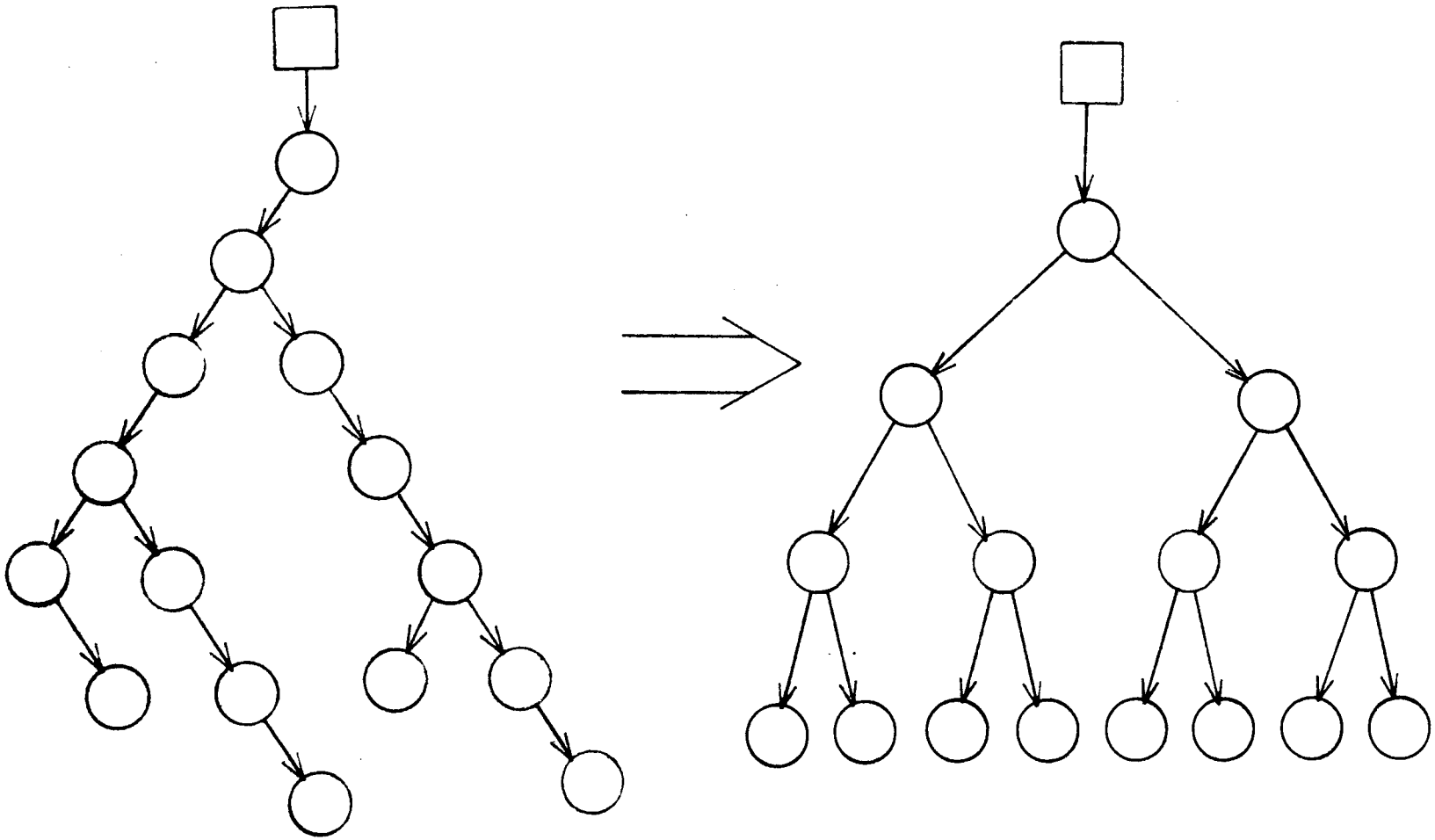

Figure 1: Access Path Reorganization - Binary Tree

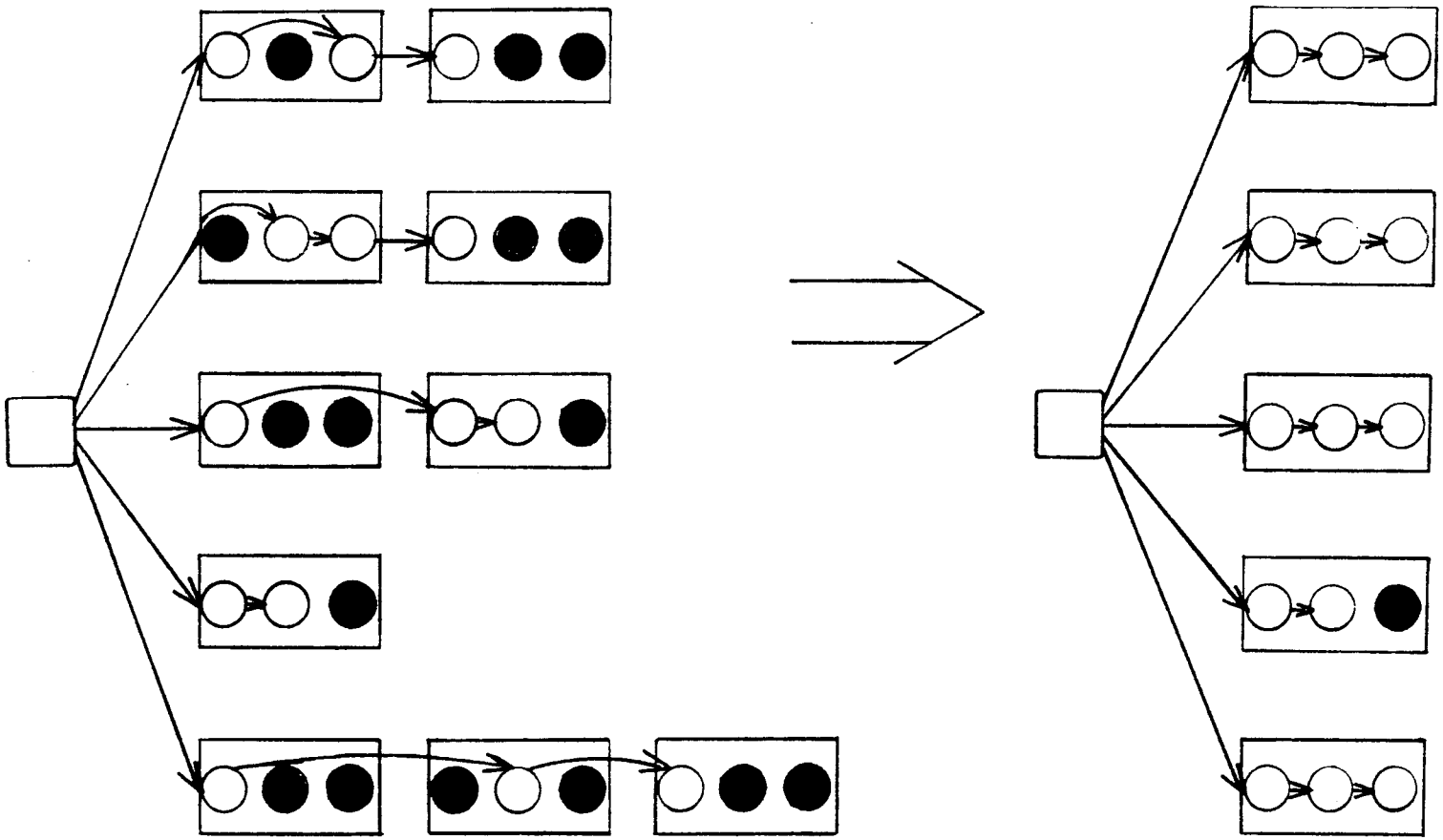

Figure 2: Block Overflow Reorganization

implementations which employ a "bucket" or block which is shared by all keys having the same hash value. As the blocks fill up, overflow blocks are used, but since nodes have been deleted it is possible to compact all of the nodes into a single block. In this example the average number of blocks accessed for a search is reduced from 1.5 to 1.0.

Even if there were no deletions, it would be possible to eliminate overflow blocks. Either the size of the blocks could be increased with only a modest increment in the transfer cost or the number of blocks could be increased. Increasing the numbers of blocks will, however, probably require a rehashing of every entry in the structure.

A structure containing master and detail records is another example of block overflow reorganization. Each record contains some master information such as account number, and a variable number of detail items, such as outstanding orders for this account. As detail items are added and deleted, overflow blocks may be used. Eventually, deletion of detail items will make it reasonable to call for reorganization to reduce the execution time and free storage space.

Linked list reorganization: The abstraction of linked list reorganization in Figure 3 can be interpreted and generalized in many ways. The blocks may be thought of as disk regions on which a simple linked list (a tree in this case) has been stored. As a result of additions and deletions the structure is now spread out over six blocks, but uses only a fraction of the space. By reorganizing the structure onto three blocks the average number of blocks retrieved for a search is reduced from 2.2 to 1.6.

Another interpretation is that the blocks are pages in a virtual memory system and the reorganization is closely related to the process of data compaction. The usual goal of data compaction is to eliminate fragmentation of storage and free-up large contiguous areas of storage. Our concern is primarily the reduction of the search cost. These methods are distinguished from garbage collection, since the latter is only concerned with the recovery of storage space.

Reorganization of a cellular multilist structure is a third example of this type. With highly linked network or cyclic structures the task of linked list reorganization and compaction can be extremely tricky. Determining the optimal placement of nodes on blocks is an additional problem which warrants further study.

Indexed Sequential Access Method Reorganization: It is hardly possible to complete a discussion on reorganization without mentioning ISAM (3). Unfortunately, reorganization of this structure is difficult to analyze since it is a combination of access

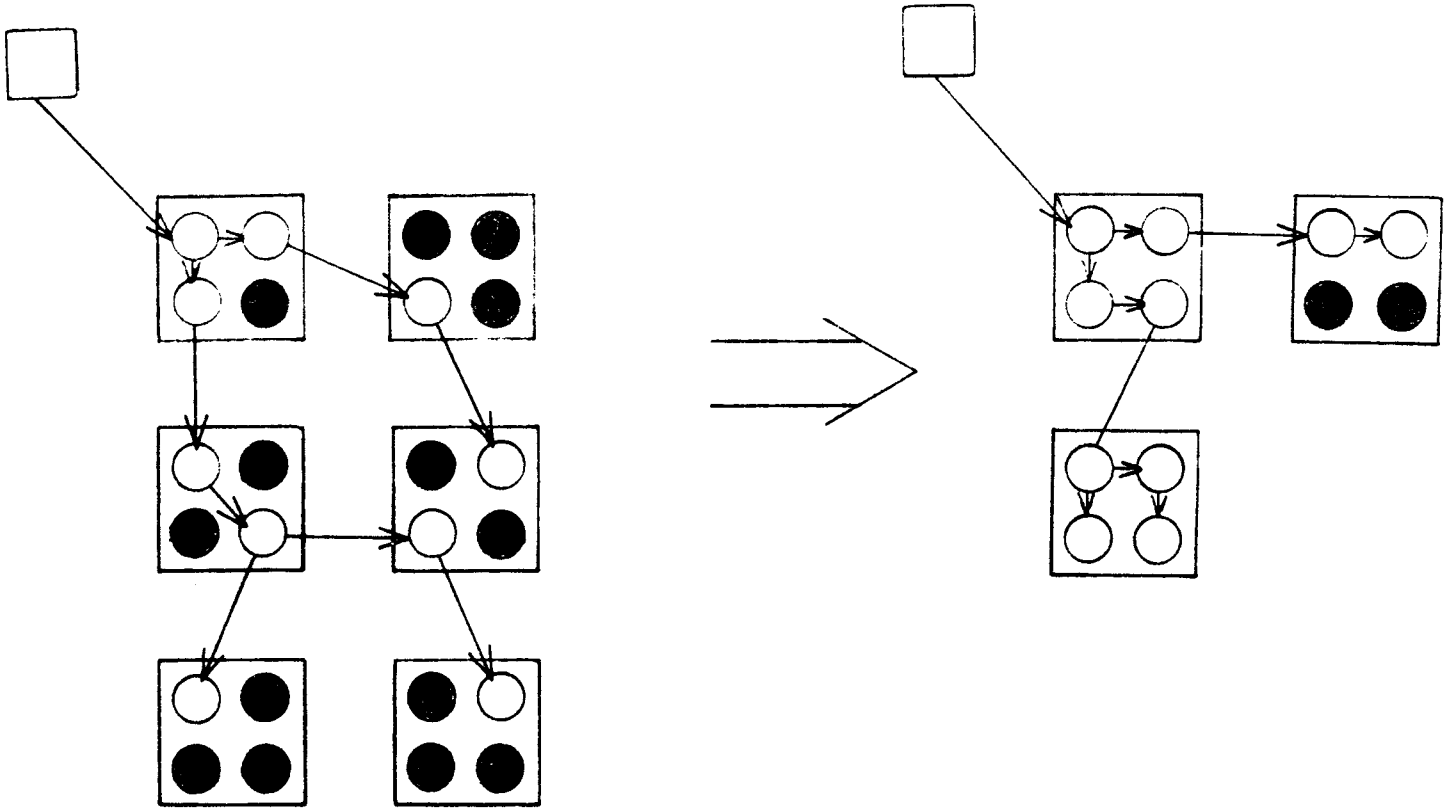

Figure 3: Linked List Reorganization

path and block overflow reorganization. New access paths are created by additional index entries and the overflow records which cause multiple accesses are removed. Although the system provides statistics on the number of records, number of overflows, etc., it can be difficult to estimate the cost of the relevant variables in determining when reorganization is to take place. Well documented experiments would be helpful in this area, since this method is widely used.

Conclusion

As data base systems attain maturity and stability the pressure to optimize performance will compel the system designer to actively consider every aspect of data base reorganization. The structure which can be reorganized, efficient algorithms for reorganization and the optimum reorganization points will all play vital roles.

References

1) Shneiderman, Ben, Optimum Data Base Reorganization Points, CACM 16, 6 (June 1973), pp.362-365.

2) Martin, W.A. and Ness, N.D., Optimizing Binary Trees Grown with Sorting Algorithm, CACM 15, 2 (February 1972), pp.88-93.

3) IBM System/360 Operating System Data Management Services, GC26-3746, IBM, White Plains, N.Y.

4) Wilfred J. Hansen, Personal Communication.

5) Knuth, Don, The Art of Computer Programming: Vol. I, Fundamental Algorithms, 2nd Edition: Addison-Wesley, (Reading, Massachusetts, 1973).

6) Knuth, Don, The Art of Computer Programming: Vol. III, Sorting and Searching, Addison-Wesley, (Reading, Massachusetts, 1973).

7) Martin, James, Computer Data-Base Organization, Prentice-Hall (Englewood Cliffs, N.J., To Be Published).