line 33 Forward.
lines 34, 35 Retrogression.
line 36 The stack vanished because the first component has no other partner except for the second one in this case.

Other three informational homologues (structures 1 through 3 in Figure 5) were also generated from the LT 189 in the same manner as above.

6. Data Comparison. As mentioned before, the generated structures 1 through 3 for compound I were not found in the file; thus no comparison was carried out. Structures 4 and 5 found in the file and their NMR and IR spectra are observed as shown in Table X.

G and S values for the NMR of compound I are exactly equal to those of structure 4, and the differences between I and structure 5 in G and S values exceed the extent of the allowance described in the text ($\Delta G \leq 0.03$ and $\Delta S \leq 0.04$). Similarly there is no difference between lines of numerals expressing IR and I and structure 4, while a significant difference between I and 5 in position and in intensity is observed as $M = 8$ and $E = 5$, respectively (cf. $M < 6$ and $E < 20$). Therefore the structure of compound I is suggested to be 4 as far as the present filed data are used.

REFERENCES AND NOTES

(1) S. R. Heller, G. W. A. Milne, and R. J. Feldmann, *J. Chem. Inf. Comput. Sci.*, **16**, 232 (1976).
(2) J. Lederberg, G. L. Sutherland, B. G. Buchanan, E. A. Feigenbaum, A. V. Robertson, A. M. Duffield, and C. Djerassi, *J. Am. Chem. Soc.*, **91**, 2973 (1969).
(3) G. Beech, R. T. Jones, and K. Miller, *Anal. Chem.*, **46**, 714 (1974).
(4) N. A. B. Gray, *Anal. Chem.*, **47**, 2426 (1975).
(5) L. A. Gribov and M. E. Elyashberg, *J. Mol. Struct.*, **9**, 357 (1971).
(6) H. Abe and P. C. Jurs, *Anal. Chem.*, **47**, 1829 (1975).
(7) S. Sasaki, Y. Kudo, S. Ochiai, and H. Abe, *Mikrochim. Acta*, 726 (1971).
(8) S. Sasaki, "Automated Chemical Structure Analysis Systems" in "Determination of Organic Structure by Physical Methods", Vol. 5, F. C. Nachod and J. J. Zuckerman, Ed., Academic Press, New York, N.Y., 1973.
(9) (a) C. A. Shelley, H. B. Woodruff, C. R. Snelling, and M. E. Munk, "Interactive Structure Elucidation" in ACS Symposium Series, No. 54, "Computer-Assisted Structure Elucidation", D. H. Smith, Ed., American Chemical Society, Washington, D.C., 1977, p 92. (b) H. B. Woodruff and M. E. Munk, *J. Org. Chem.*, **42**, 1761 (1977).
(10) R. E. Carhart, D. H. Smith, H. Brown, and C. Djerassi, *J. Am. Chem. Soc.*, **97**, 5755 (1975).
(11) Y. Kudo and S. Sasaki, *J. Chem. Doc.*, **14**, 200 (1974).
(12) Y. Kudo and S. Sasaki, *J. Chem. Inf. Comput. Sci.*, **16**, 43 (1976).
(13) S. Sasaki, Y. Yotsui, and S. Ochiai, *Bunseki Kagaku*, **24**, 213 (1975).
(14) B. A. Knock, I. C. Smith, D. E. Wright, R. G. Ridley, and W. Kelley, *Anal. Chem.*, **42**, 1516 (1970).
(15) S. Sasaki, H. Abe, Y. Kudo, and T. Yamasaki, "CHEMICS: A Computer Program System for Structure Elucidation of Organic Compounds" in ref 9a, p 108.
(16) MS analysis will be introduced in the near future.
(17) The value of 179th element is calculated after completing the 178th dimensional vector (cf. eq 4).

A Compact and Efficient File Structure for Searching Large Generic-Keyed Databases. An Application to Mass Spectral Data

R. GEOFF DROMEY*
Research School of Chemistry, Australian National University, Canberra, A.C.T. 2600, Australia

Received July 1, 1977

Conventional file structures do not satisfactorily handle large generic-keyed databases. Seemingly a compromise must always be made between storage requirements and retrieval efficiency. A new inverted bitmap file structure that does not involve a high storage cost is suggested as a viable alternative to existing systems. It requires less storage and is faster for retrieval than other systems that are currently being used. Implementation and performance evaluation for a mass spectral database are given.

INTRODUCTION

Key-based information storage and retrieval systems usually fall into one of three categories. The simplest of these is the single-key file, where each record possesses just one key which may or may not be unique. A chemical name file and a molecular formula file would fit into this category. Efficient methods for handling this type of system are readily available.[1,2] The second category is typified by a bibliographic file, where each record is characterized by perhaps an author, title, and several keywords. Methods for handling these systems with their relatively few keywords per record are also straightforward.[3] It is in the third category, where each record may consist of many generic keys, that the real difficulties arise. Application of existing methods to this problem has always resulted in a tradeoff between storage and processing efficiency.

The present paper represents an attempt to design an efficient system for handling large databases in the third category. The aim has been to present a file structure which is simple, easy to construct, and yet at the same time is highly economical on both storage requirements and retrieval times.

Lefkovitz[4] has discussed the characteristics of generic-key files and suggested that a hybrid inverted list-bitmap file structure is a practical way of handling these systems. This hybrid file design is tailored to cope with the Zipfian-like[5] distribution among keys that almost invariably exists for large generic-keyed databases. The small proportion of keys occurring at high frequency is stored in fixed-length inverted bitmaps while the majority of keys, which occur only infrequently, are stored in inverted lists. This approach certainly economizes on storage but at the cost of introducing very inefficient Boolean operations between the two data structures (bits and lists). For any given system there is a degree of uncertainty as to what is the most useful mix between bitmaps and lists. The hybrid file approach, although clearly superior to other available systems, still must trade efficiency for storage. A desirable goal would therefore seem to be to devise a data structure that could exploit the processing efficiency of bitmap systems without incurring the excessive storage costs that they conventionally entail.

Zatocoding[4,6] has been suggested as a possible bitmap-oriented solution to the large database dilemma. Its drawbacks are that it involves a sequential rather than inverted search and that the super-imposed code can produce a significant risk of false retrievals. Lefkovitz[4] gives a detailed account of why zatocoding would appear to be unsatisfactory for very large systems.

Efficiency demands that some type of inverted bitmap be used for large databases. The storage problem still remains.

* Address correspondence to author at Department of Computing Science, University of Wollongong, Wollongong, N.S.W. 2500, Australia.

0095-2338/78/1618-0222$01.00/0

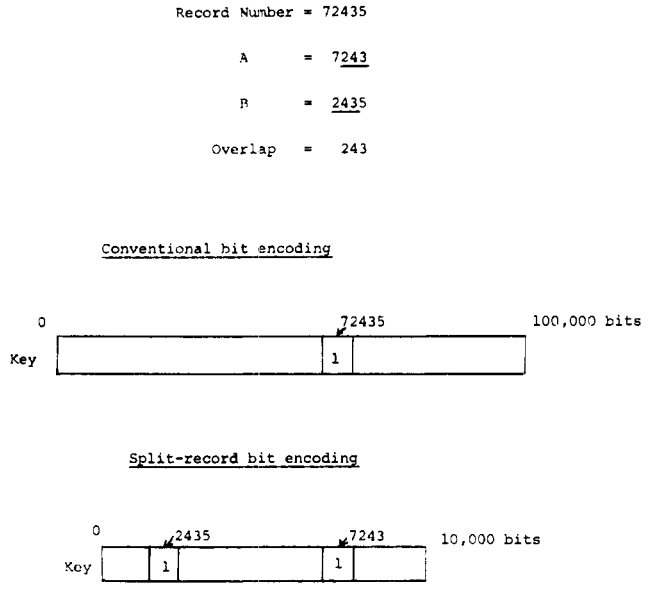

Figure 1. Comparison of conventional and split-record number bitmap encoding.

Compaction of bitmaps by replacing long strings of zeros by integers is not attractive because it introduces further processing and file maintenance complexities. Bitmaps can lead to high efficiencies only when they are fixed in length.

A HIGHLY COMPRESSED SPLIT-RECORD-NUMBER INVERTED BITMAP

The preceding discussion has laid down the characteristics for an efficient retrieval system. The problem that remains to be solved is how a fixed-length inverted bitmap structure can be encoded so that the storage is economical.

In conventional bitmaps each key is assigned a fixed-length linear array of bits with the total number of bits corresponding to the total number of records on a one-to-one basis. Close to an order of magnitude reduction in the inverted bitmap file size can be obtained by using a somewhat different approach to record number encoding. The mechanism for the new approach to bit-encoding is best described by example. The five-digit record number

$$R = 72435$$

may be taken to consist of two overlapping four-digit values A and B as shown

$$R \rightarrow \overset{A}{\overbrace{72435}}\underset{B}{}$$

where $A \rightarrow 7243$ (high-order component), $B \rightarrow 2435$ (low-order component), and overlap $\rightarrow 243$ (three-digit overlap). Bit encoding for the record number 72435 then involves setting the bits $A = 7243$ and $B = 2435$ in an array of only 9999 bits (an order of magnitude less than the original bitmap); that is, record numbers in the range 0–99 999 are accommodated in an array of only 9999 bits (Figure 1). Split-record number encoding by its nature involves some degradation in record number representation even though there is strong protection afforded by the three-digit overlap.

One possible drawback of split-record encoding in the present context is that in using the three-digit overlap to transform the paired four-digit codes back to their original five-digit representation it may be found that by chance some bit combinations will result in record numbers appearing to be set that do not belong to the key under consideration. There are two factors which act strongly to reduce the record number representation ambiguity to insignificance. The first is that the bit density for most keys is very low in generic files (the Zipfian characteristic) and so ambiguous combinations are unlikely. Secondly, the key searches for generic data act strongly as a mechanism for reducing the possibility that a combined Boolean result will retrieve erroneous records. That is, even if by chance a number of ambiguous records are present in the bit array for a given key, there is little likelihood that the bits corresponding to the erroneous record numbers will be able to survive beyond the Boolean combination of several search keys. The results for a mass spectral retrieval system (discussed later) suggest that the predicted elimination of ambiguity is borne out in practice.

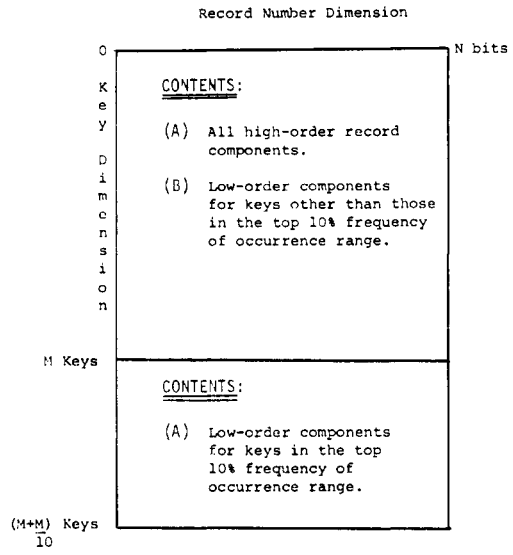

Figure 2. Split-record bitmap implementation that takes into account the high frequency of occurrence of a small proportion of keys.

In almost all generic-keyed files there is a small proportion of keys that occur with high frequency (e.g., some keys may occur in as many as one-quarter of the records present). Obviously, if a small set of keys in this high-frequency category were to be combined in a search, there would be a significant chance that erroneous record numbers would be retrieved. At a small additional cost in storage this possibility can be reduced to near insignificance. The first step is to identify the keys that are, say, in the top 10% of the frequency of occurrence distribution. For this set of keys, only the high-order (A) components are stored in the main body of the file. The corresponding low-order components are set in a small separately allotted key space (Figure 2). This configuration requires a small amount of additional processing for retrieval. If a second array is used for holding intermediate Boolean results from the low-order keys that have been designated as high frequency, then the latter can be prevented from being eligible to act as potential high-order components for record number reconstruction via the three-digit overlap criterion. This precaution can significantly reduce the risk of erroneous record number retrievals for searches involving only high-frequency keys. The extra costs in storage space and retrieval times are small.

IMPLEMENTATION OF A SPLIT-RECORD INVERTED BITMAP FOR A MASS SPECTRAL DATABASE

A large file of mass spectra is typical of generic-keyed databases. There are a large number (more than 1000) of possible keys (masses), most of which occur relatively infrequently. There are also a small number (perhaps 40–60) that occur with high frequencies. A mass spectral database

Table I. Inverted File Storage Requirements for 32768 Mass Spectra

method	storage required (words)
conventional inverted bitmap	4710400
inverted list	3682220
hybrid inverted list-bitmap	900702[a]
split-record bitmap	665600

[a] See discussion in text.

is thus ideal to test out the performance of a split-record inverted bitmap system.

There is one complication with mass spectral data: the variability in ion current intensity must be taken into account. Traditionally, in inverted list files for mass spectral databases, an intensity match is considered to be made if the query peak and the file peak (for a particular mass) match to within a predefined intensity range.[7] This matching procedure involves an arithmetic range comparison and so it is relatively inefficient. The intensity match can be reduced to a simple Boolean match by special encoding of the intensity data.

McLafferty et al.[8] have shown that the significance of mass spectral peaks follows a logarithmic dependence. For the present purposes of inverted file construction it was therefore found convenient to divide the 0% → 100% ion intensity range into five doubling ranges: 0 → 6.25, 6.25 → 12.5, 12.5 → 25.0, 25.0 → 50.0, 50.0 → 100.0. Each mass was assigned five possible keys corresponding to the five intensity ranges. The occurrence of a peak in a given range required that bits be set for the key in that range and the ranges on either side; e.g., a peak of 22% would require keys to be set for the ranges 6.25 → 12.5, 12.5 → 25.0, and 25.0 → 50.0 for the appropriate mass. Intensities in the two end ranges require additional settings only in the appropriate, immediately adjacent range, in each case.

For the present system because of file constraints it was found to be computationally more convenient to work with the octal representation of record numbers. This approach led to a file reduction factor of only eight rather than ten over a conventional inverted bitmap. The split-record inverted bitmap was set to handle a file of up to 32768 (100000 octal) records or spectra. A bitmap length of only 10000 octal bits (256 16-bit words per key) was needed for this file. The number of keys used was 2600 (note there were five keys per mass). Because of their very infrequent occurrence, masses above 500 amu were mapped onto the range 400–500 amu (e.g., mass 745 was set as though it were 445). Only the two most intense peaks per 14 amu were used. This is standard practice for mass spectral data.[7] The range 40–100 amu encompasses most of the high-frequency masses in the database. The high-order and low-order components for this range were encoded separately in the way described earlier. No mass below 40 amu was encoded. In the final configuration, a file reduction factor of about seven was obtained relative to a standard inverted bitmap.

COMPARISON OF THE SPLIT-RECORD INVERTED BITMAP WITH OTHER SYSTEMS

It is important to make a concrete comparison of the split-record bitmap with other available systems. This can be done readily for the mass spectral database implemention. The storage requirements of the various systems for a file of 32768 spectra are summarized in Table I. All storage has been standardized for a 16-bit word size. Only the split-record file and the conventional inverted bitmap were actually implemented. Available data, however, allowed storage calculations for the two other systems. Table I shows that the split-record implementation compares very favorably with the other three systems. Storage for the hybrid system was calculated on the basis that 90% of the file activity was concentrated in only 10% of the keys. This ratio is rather optimistic. A more realistic estimate of the distribution of activity among the keys would result in the hybrid structure requiring more than *twice* the storage of the split-record file. Zatocoding would certainly take significantly less storage but, as Lefkovitz has pointed out,[4] it is not practical for very large databases.

Table II. Performance Comparison for Conventional and Split-Record Inverted Bitmaps for 50 Spectra Retrievals[a]

performance characteristic	split-record inverted bitmap	conventional inverted bitmap
av no. of peaks to retrieve a spectrum free of false drops	5.9	5.1
av no. of Boolean word matches to retrieve a spectrum	420	345
inverted file size (16-bit words)	665600	4710400

[a] One spectrum was not resolved after nine peaks in both systems. The falsely retrieved spectrum was found to be an isomer of the test compound.

To evaluate the effects of ambiguity introduced by the split-record encoding scheme, a series of comparisons was made for spectrum retrieval by a standard inverted bitmap and the split-record system. A set of 50 spectra known to be present in the database of 20000 spectra was used for the comparison. The results for the two systems are summarized in Table II. The conclusion that can be drawn from the comparison is that on average one additional peak (key) is needed for the split-record system to retrieve a given spectrum free of false drops. This is a small price to pay for the savings in storage and gains in efficiency of the split-record over other file structures. The most efficient strategy for retrieving spectra was to start the search with peaks at highest mass and move down the mass scale. This follows from the fact that peaks at high mass occur less frequently in mass spectra. All retrieval comparisons were made by starting the search at the high mass end of the spectrum.

It was not possible to establish meaningful retrieval time comparisons for the two files because disk space limitations made it necessary to implement the conventional inverted bitmap on magtape. Both implementations were carried out on a time-shared PDP 11/45 computer. Peak-searching response was immediate for the disk-based, split-record implementation.

Results for an inverted list implementation have not been included in Table II because it is not easy to make meaningful direct comparisons with the bitmap systems. It is, however, generally accepted that intersecting of inverted lists is significantly less efficient than performing Boolean operations on fixed length bitmaps.[4]

The split-record file forms an integral part of a mass spectral search system that includes molecular formula, molecular weight, chemical name, and mass spectrum retrieval options. The data structure is such that the mass spectrum search can be restricted by either a molecular formula, or molecular weight presearch, if desired.

CONCLUSION

The split-record inverted bitmap file structure is seen to compare favorably with other information retrieval systems for large generic-keyed databases. It goes beyond the storage–retrieval efficiency dichotomy in that it requires less storage and it exhibits more efficient retrieval times than other available systems. The results for the mass spectral database suggest that any ambiguity introduced by split-record encoding is reduced to insignificance by the combined effects of the Boolean search and the low bit density for most keys. The generality of the split-record file system makes it potentially suitable for handling with economy a wide range of difficult

information storage and retrieval problems. For maximum efficiency radixes other than 8 and 10 would be needed for some implementations.

ACKNOWLEDGMENT

The author would like to thank Dr J. K. MacLeod for his comments and Mrs. Greta Pribyl and Ms. Lorraine Scarr for their help in preparing the manuscript. The author also wishes to thank Dr. G. W. A. Milne and Dr. S. R. Heller for providing the mass spectral database.

REFERENCES AND NOTES

(1) C. T. Meadow, "The Analysis of Information Systems", Melville Publishing Co., Los Angeles, Calif., 1973, pp 314–344.
(2) N. Wirth, "Algorithms + Data Structures = Programs", Prentice-Hall, Englewood Cliffs, N. J., 1976.
(3) R. G. Dromey, "A Compact Free-Keyword File Structure for Author-Title-Keyword Searching", *J. Chem. Inf. Comput. Sci.*, **18**, 160 (1978).
(4) D. Lefkovitz, "The Large Data Base File Structure Dilemma", *J. Chem. Inf. Comput. Sci.*, **15**, 14 (1975).
(5) G. K. Zipf, "Human Behaviour and the Principle of Least Effort", Addison-Wesley Publishing Co., Cambridge Mass., 1949, pp 19–55.
(6) C. N. Mooers, "Zatocoding Applied to Mechanical Organization of Knowledge", *Am. Doc.*, **2**, 20 (1951).
(7) S. R. Heller, "Conversational Mass Spectral Retrieval System and Its Use as an Aid in Structure Determination", *Anal. Chem.*, **44**, 1951 (1972).
(8) G. M. Pesyna, F. W. McLafferty, R. Venkataraghavan, and H. E. Dayringer, "Statistical Occurrence of Mass and Abundance Values in Mass Spectra", *Anal. Chem.*, **47**, 1161 (1975).

A Simple Tree-Structured Line Formula Notation for Representing Molecular Topology

R. GEOFF. DROMEY*

Research School of Chemistry, Australian National University, P.O. Box 4, Canberra, A.C.T. 2600, Australia

Received June 15, 1977

A new linear notation for representing molecular topology is described. This canonical number-based approach is able to match the encoding power and systematics of existing systems (e.g., WLN) while using a much simpler encoding formalism. As such, the system should be much easier to use both manually and computerwise.

INTRODUCTION

Considerable effort has been expended in exploring linear notations for computer representation of molecular topology. Wiswesser line notation[1] (WLN), Dendral,[2] Hayward,[3] IUPAC,[4] and Skolnik,[5,6] are among the most concise and powerful general approaches to representation of molecular topology. These notations have become widely accepted among information scientists, and there would appear to be little point in adding another notation to the list unless it possessed some significant advantages over the current notations. A tree-structured "numeric" linear notation is proposed as a viable alternative *because it is able to offer a much simpler encoding (and decoding) formalism than existing methods.* The system has been designed to be computationally more attractive and to possess better indexing properties. The basis alphabet for this system is the hexadecimal character set (a 4-bit code), that is, the numerals from Ø to 9 and the characters from A to F. With this concise "alphabet" (cf. WLN which uses 40 characters—an 8-bit code on most computers) and, on a relative scale, only a minimal set of rules and conventions, it is possible to match the encoding power and systematics of existing notations. The first task that must be faced in designing a notation is the atom representation convention. A glossary of representation conventions is given in Tables I and II.

1. CONVENTIONS FOR REPRESENTING ATOMS

A very sound design criterion in existing systems has been to choose wherever possible symbols familiar to chemists. To try and meet this design criterion in the present system, the three most commonly occurring elements in organic compounds (apart from hydrogen), carbon, oxygen, and nitrogen, are assigned numeric representations according to their common valences, e.g., carbon = 4, oxygen = 2, and nitrogen = 3. The most abundant isotopes of all other elements in the periodic table take on a representation which is a function of their valency and atomic weight. The reason for this choice will become clear when the precedence rules are discussed. The basic format is

A[valence][atomic weight(rounded)]

Some examples are

A135 = chlorine A232 = sulfur A2Ø9 = beryllium

The symbol "A" (A for *A*tom) is used *only* for atomic representation. Elements with atomic weights greater than 99 and the representation of other than the most abundant isotopes are discussed in a more detailed report.[7]

2. RULES OF PRECEDENCE FOR ENCODING MOLECULAR STRUCTURES

To obtain an encoding scheme that is unique and unambiguous it is necessary to use a hierarchical set of precedence rules. In order to make the rules simple to apply they have been framed in terms of valency, connectivity, and atomic weight, concepts which are both very familiar to chemists and easy to work with. The central rules encompassing these concepts can be stated as follows.

General Rules of Precedence: At each stage in encoding a structure always choose to encode first

(A) *that connected path with an atom of smallest valence attached earliest*;

(B) *where minimum valence does not resolve the path choose to encode first the path with the atom of smallest atomic weight attached earliest*;

(C) *if resolution still has not been made, encode along the path that contains an atom with the least number of atoms attached earliest*;

(D) *finally, if none of the other constraints resolves the path, choose to encode along the path that has an atom with the least number of hydrogens attached earliest.*

This set of rules is applied hierarchically wherever precedence must decide which atom (or ring) is to be encoded next.

* Adress correspondence to author at Department of Computing Science, University of Wollongong, Wollongong, N.S.W. 2500, Australia.

0095-2338/78/1618-0225$01.00/0