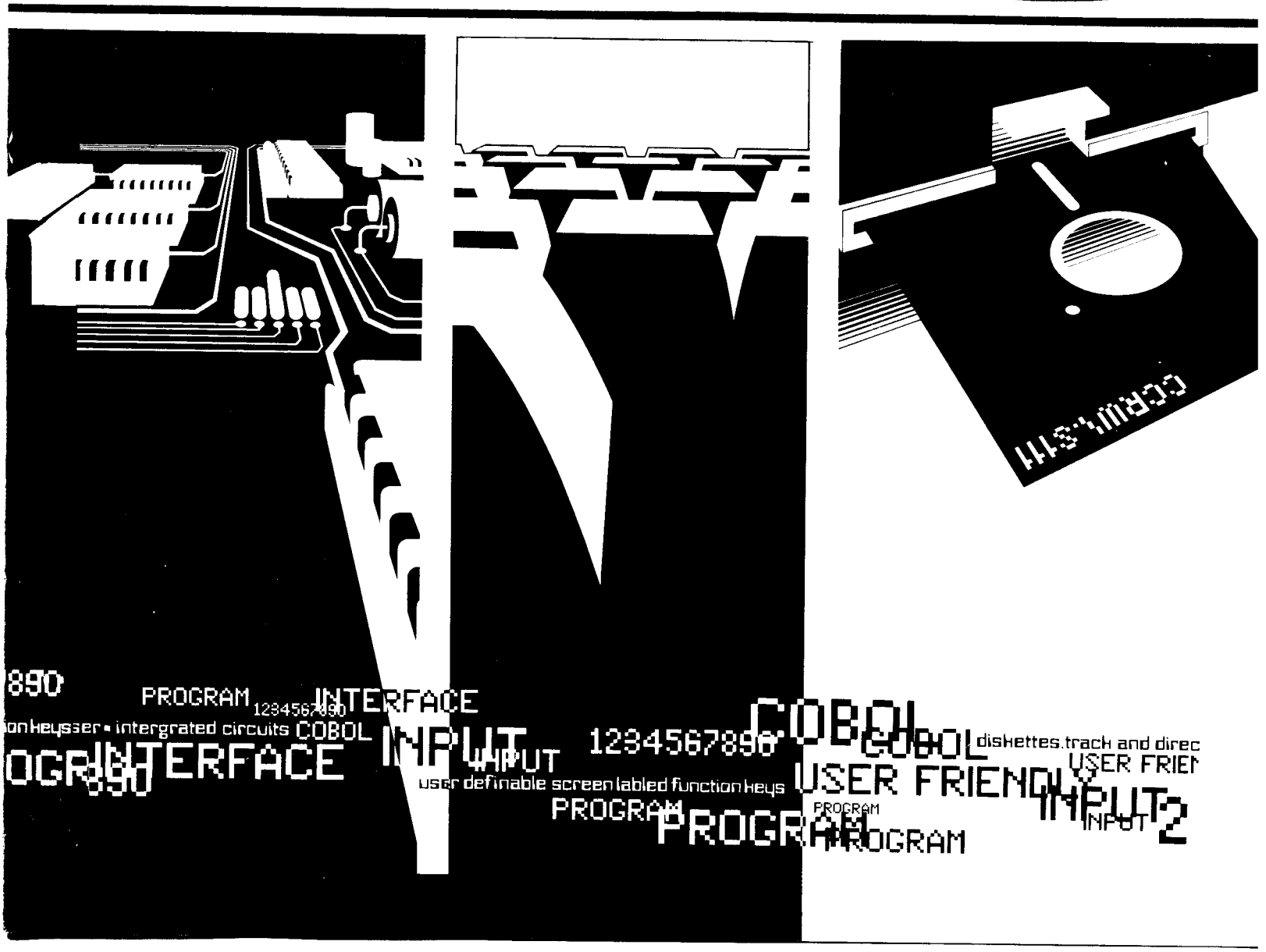
RESEARCH IN WORD PROCESSING NEWSLETTER

South Dakota School of Mines and Technology Rapid City, SD 57701 (605) 394-2481
Volume 6 Number 2 February 1988

International Humanities Conference in Toronto

page 18



RESEARCH IN
WORD PROCESSING
NEWSLETTER

Editors
Bradford A. Morgan
James M. Schwartz

Contributing Editors
Bryan Pfaffenberger
University of Virginia

Tom Carney
University of Windsor

Terrence Erdt
Villanova University

Production Manager
Dan Dorland

Art Director
Francis Jacobs

Typesetting/Graphics
Brenda Boyer

Photography
Dave Adams

Printing
Dick Beshara

Circulation
Wendy Painter

Accounting Manager
Sondra Wagner

Research in Word Processing Newsletter. Volume 6, Number 2. Copyright © 1988 by the South Dakota School of Mines and Technology. All rights reserved. ISSN: 0748-5484. Also indexed by **ERIC, INSPEC, COMPUTER LITERATURE INDEX,** and **SOFTWARE REVIEWS ON FILE.** The *Research in Word Processing Newsletter* is published 9 times a year (September through May) by the South Dakota School of Mines and Technology, Rapid City, South Dakota 57701-3995; (605)394-2481. Postage paid at Rapid City, South Dakota. **SUBSCRIPTION:** $18 per year (U.S.); $24 per year (Canada); $27 per year (foreign). Address all subscription inquiries and manuscripts to the Editors, *Research in Word Processing Newsletter,* SDSM&T, 501 E. St. Joseph, Rapid City, SD 57701-3995. Please allow 4 to 6 weeks for subscription processing. **POSTMASTER:** Send address changes to RWPN, South Dakota School of Mines and Technology, 501 East St. Joseph, Rapid City, South Dakota 57701-3995.

THE PROFESSIONAL WRITER'S WORKSTATION

Software for Managing Information

Bryan Pfaffenberger

Vannevar Bush, a physicist who did pioneering work on computers during the Second World War, wrote in 1945 of the uses to which information processing technology might be put in peacetime. Scientists of the future, Bush predicted, would use an intelligent desk that he termed MEMEX. A MEMEX is a machine in which

> an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. *It is an enlarged intimate supplement to his memory.*[1]

More than a record keeper, MEMEX would be an aid to thought. People think, Bush noted, using association: "with one item in [the mind's] grasp, it snaps instantly to the next that is suggested by the association of thoughts." MEMEX's real value, therefore, would be to provide people with a device that traces an associative trail through a vast library of material, with near-instantaneous response. A MEMEX-using historian interested in the origins of the bow and arrow, for instance, might begin with an encyclopedia article, continue with relevant historical works, branch into the physics of elasticity, and end with inserting a page of her own notes into MEMEX. The MEMEX's point wouldn't be to replace human thought, but on the contrary, to provide it with a tool that "beats the mind decisively in regard to the permanence and clarity of the items resurrected from storage."[2]

MEMEX doesn't yet exist in the way Bush foresaw; the job of digitizing a scholar's library of books is a big one by any measure, and by no means does every expert believe it worthwhile. Books may be bulky, but they have aesthetic qualities that MEMEX would presumably lack, and they're portable. And the computer power needed to process all that information, moreover, still wouldn't sit on a desk.

Yet the personal computer can indeed provide a MEMEX-like machine on your desk right now. The key to creating this "enlarged intimate supplement to memory" lies in the computer-based storage and retrieval of textual information.

Personal computers permit scholars to store and retrieve textual information in two ways:

> **Personal computer-based information storage and retrieval systems** Using information storage and retrieval software, personal computers can provide the staging ground for individual data bases of research notes, bibliographic citations and annotations, and other collections of information.
>
> **Mainframe computer-based information storage and retrieval systems** Using telecommunications software, personal computers can utilize the massive reference resources of *online data base* services, which maintain millions of bibliographic citations and abstracts in all areas of scholarship and science. The software used by these data base services closely resembles the software available for personal computer information management.

[1] My italics. "As We May Think," originally published in 1945 (*Atlantic Monthly*), reprinted in *Perspectives on the Computer Revolution*, Zenon W. Pylyshyn, ed. (Englewood Cliffs, NJ: Prentice-Hall, 1970), p. 55.
[2] Ibid.

INTRODUCING TEXT-ORIENTED DATA BASE MANAGEMENT SOFTWARE

Software designed to aid the storage and retrieval of text is usually termed text-oriented data base management software. Here's a path through this jungle of terms.

THE DATA BASE CONCEPT

A *data base* is nothing more mysterious than a collection of information pertinent to a particular subject. This collection provides a base or foundation for drawing conclusions or making decisions. A *data base management* program, therefore, is a program for managing (that is, storing, organizing, and retrieving) the information in the data base.

To illustrate the basic concepts underlying data base management, consider a box of three-by-five cards that contain bibliographic citations and notes.

The point of keeping these cards is to store and retrieve bibliographic information. The bibliography's usefulness, however, depends on whether you remember to write down all the necessary information on each card. If, for instance, you leave out the publisher's name, the citation will be next to useless when it comes to preparing the final draft of a journal article: you'll have to go to the library and look it up again. When you set up a card file of bibliographic citations, therefore, you devote some thought to how you want to arrange the information on the cards.

Author
Title
Place of Publication
Publisher
Date of Publication
Abstract
Location

FIGURE 1. Design for a Bibliography Card

If you're especially careful, you'll probably include a set of headings (such as "author," "title," etc.) so you don't forget to write down the necessary information on every bibliographic card. The headings, however, may be implicit in the data base's design and not actually written on every card. That's fine, so long as you remember to include all the necessary information. A typical heading design in shown in Figure 1. A completed index card, containing responses to the headings (now implicit), is shown in Figure 2.

---

de Silva, Lynn
Buddhism: Beliefs and Practices in Ceylon
Colombo, Sri Lanka
Wesley Press
1984

Excellent overview of Theravada Buddhism in Sri Lanka by a sympathetic and scholarly observer. Covered are not only the doctrinal aspects of Buddhist belief and practice but also the so-called "popular" aspects of everyday Buddhism, which have much in common with neighboring Hindu customs and beliefs and rarely receive treatment in works of this type.

In my personal collection.

FIGURE 2. Completed Bibliography Card

SOME BASIC TERMS

With the index card example in mind, it's now possible to define the three fundamental terms of data base management:

Data record A single unit of related information in the data base (corresponding to a single index card in the above example).

Data field A space for entering a particular kind of information in a data record (corresponding, for instance, to the "author" or "abstract" fields in the above example).

Data record format The overall design of data fields is repeated in every record.

LIMITATIONS OF PRECOMPUTER DATA BASES

A card file is a true data base and can be described in the terms just defined. But it has several limitations that you doubtless know only too well:

There's only one way you can file the cards. Suppose, for instance, you're a geologist and you've created a file of rock specimens in your personal collection. You could file them by type of rock (using three sections for igneous, sedimentary, and metamorphic rocks) *or* by the location of their discovery, but not both. What happens, then, if you want a list of all the rocks in your collection from the Mammoth Caves? The only way you can retrieve the information is to search through every single card manually.

You have to alphabetize them and search through them by hand. This problem isn't overwhelming when you've got only 25 or 50 cards, but what

about 200? 500? Unless you've an assitant to help you (rare, these days, and growing rarer), alphabetizing all these cards could discourage you from creating a filing system (or making any more additions to it).

There's only one way you can list facts from your card files: by going through the cards manually, extracting the facts you want, and typing them up, an unbelievably boring job at best (or expensive, if you hire a typist). And when you've finished, you've a data base that's frozen in time. Making any additions to it means retyping the whole thing, doubling the tedium (or expense).

A card files stores information well—so well, in fact, that it's more than difficult to get it back out again. In a bibliography sorted by author, for example, how do you find that classic 1968 work on pesticides by an author whose name you've forgotten? A good fact-crunching system should permit you to retrieve information just as easily as you can store it.

ADVANTAGES OF COMPUTERIZED DATA BASE MANAGEMENT

Data base management programs dramatically cure the retrieval problem by performing the following tasks quickly and almost automatically: *sorting, searching,* creating a *view,* and printing.

Sorting

Most data base management programs include a sorting or ordering command that permits you to arrange the data records in any of the following ways:

Ascending numerical order (1, 2, 3 . . .)
Descending numerical order (. . . 3, 2, 1)
Ascending alphabetical order (a, b, c . . .)
Descending alphabetical order (. . . c, b, a)

Generally, you're given a choice about which data field you'd like to use as a *key* or basis for sorting the records, and that's a handy feature. A major limitation of precomputer data bases is that records could be sorted only on one key (for instance, "author" in the bibliography example used above) and the sorting is so tedious that one is discouraged from repeating it. Computer data base management programs, however, can be used to sort an extensive bibliography repeatedly and quickly. To look only at the most recent works, for instance, a bibliographic data base sorted by author could be resorted in descending numerical order using the "date" field as a key.

Some programs permit multiple key sorts, or sorts that employ sorting keys successively to organize the information in a data base. Suppose, for example, a bibliographic data base is to be sorted by author and then by date. First, the program would sort all the records alphabetically (Abrams, Anthony, Bardwell, etc.). Then, when it encounters more than one record by a single author, it would sort those records by date (Abrams 1981, Abrams 1982, Abrams 1984, etc.).

Searching

A data base management program's search command (sometimes called a *query* or *select* command) provides a way to find specific information within the data base.

TYPES OF SEARCHES. Most data base management programs provide the tools needed to do three kinds of searches:

SIMPLE SEARCHES. The simplest search involves the use of a single search phrase, such as "nineteenth century communes." To use just this one search phrase is to ask the computer, in effect, to find all the records that contain the words "nineteenth century communes." Simple searches may produce more records than you can conveniently read in one setting if the search phrase is a general one. They're useful, however, when you're trying to locate all instances of a term or you're trying to pinpoint records pertaining to a topic that you know to be mentioned in only a few records (such as "Frederick, Harriet").

SEARCHES USING A LOGICAL OPERATOR. More finely tuned search questions may be phrased using a *logical operator,* such as AND, OR, or NOT. Using these operators permits you to frame more precise (or more inclusive) search questions than you could using simple search techniques.

MULTIPLE-OPERATOR SEARCHES. Multiple operator searches use search questions written with two or more logical operators.

LOGICAL OPERATORS. Some programs provide search commands equivalent to those found in word processing programs, which let you specify a simple *search question* (a word or phrase) and show you where it's located in the text. Most data base management programs, however, give you more advanced searching functions using logical operators (sometimes called *Boolean operators* after George Boole, the nineteenth-century mathematician who invented the logic on which these operators are based), which let you frame even more precise search questions.

Logical operators, which are expressed as the connective AND, OR, or NOT, permit you to specify a set of criteria about the data records you'd like to see. You might ask, for example, for "all the records where the field Rock Type is equal to 'Igneous' **AND** the field Site is equal to 'Mammoth Caves.' "

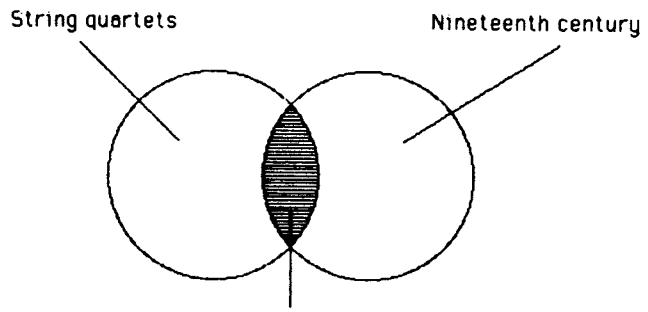


FIGURE 3. Restrictive Search using the AND Operator.

Venn diagrams help to clarify the meaning and function of logical operators. Suppose, for instance, you have a data base of classical recordings that you've labeled by genre ("string quartet" or "piano trio") and period ("eighteenth century" or "nineteenth century"). Phrasing a search question using the AND operator produces a highly restrictive search (Figure 3): only the records that meet both criteria are retrieved.

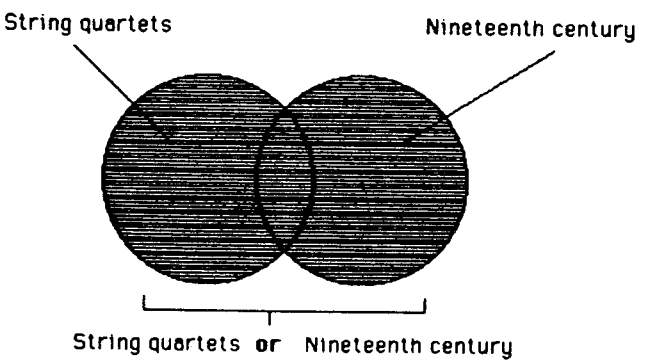


FIGURE 4. Inclusive Search using the OR Operator.

The OR operator is as inclusive as the AND operator is exclusive (Figure 4): records that meet either criteria are retrieved.

The NOT operator permits the exclusion of a subset of the records that contain an undesirable element (Figure 5).

ORDER OF EVALUATION. Just as it's important to understand the order of evaluation when you're using mathematical operators, so too is it important that you understand how a data base management program evaluates search terms. A common order of evaluation puts terms linked with OR first; AND expressions are evaluated second. Therefore, "dolphin AND whale OR porpoise" finds all the records that mention either "whale" or "porpoise" AND also mention "dolphin." That may or may not be desirable. "Dolphin OR porpoise AND whale" is a completely different expression; it finds those records that mention either "dolphin" or "porpoise" and also mention "whale." Understanding these logical operators and their use is basic to data base management proficiency. They play a key role, too, in the searching of online data bases.

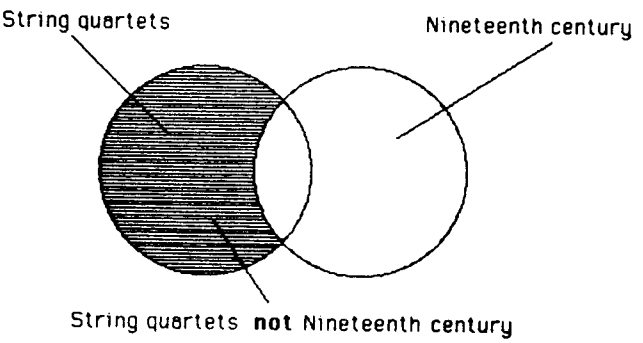


FIGURE 5. Exclusive Search using the NOT Operator.

THE WITH OPERATOR. A desirable addition to a program's set of logical operators is the WITH/*n* operator, which lets you phrase a question of the form "find all instances of TERM A that occur within *n* words of TERM B." Consider, for example, searching for "economic stratification." One could simply enter the two words, but most programs would search for precise matches. Were "*stratification*" used where "economic" was implicitly understood, the passage would not be retrieved. Phrasing the search question "stratification WITH (20) economic" returns all instances of "stratification that occur within 20 words of "economic" in either direction. The WITH operator, in other words, lets you put the context into the search question.

WILD CARD SEARCHING. Wild card searching (sometimes called truncation) is a valuable feature that lets you expand your search question's inclusiveness. Consider, for example, searching for "fiction." Without wild cards, the program would not retrieve records containing "fictions," "fictional," "fictitious," or "fictive." To avoid this exclusion of kindred terms, you could use AND operators ("Find fiction AND fictions

AND fictional AND fictitious AND fictive"), but this procedure is tedious and error-prone; you might forget to enter one or more related forms. Wild card characters let you enter a root word such as "soc" (social, society, socialization, etc.) or "publici" (publicity, publicist, publicizes, etc.).

SINGLE-CHARACTER WILD CARD SYMBOLS placed at the end of a word retrieve only those words that contain the root plus one additional character. Often, the single-character wild card symbol is a question mark. The search term "education," for instance, retrieves "education" and "educations" but not "educational."

MULTICHARACTER WILD CARD SYMBOLS placed at the end of a word retrieve all the words that contain the root plus any additional characters, no matter how numerous. Often, the multicharacter wild card symbol is an asterisk. The search term "education*," for instance, retrieves "education," "educations," "educational," and "educationally." Be careful, however, not to truncate the root too much when you're using the multicharacter wild card symbol. Consider searching for terms related to the subject of sociology. The search term "soc*," for instance, returns not only the desirable sociological terms (socialization, society, etc.) but also soccer, socks, Socrates, and many more unrelated ones.

PARENTHETICAL EXPRESSIONS. Programs that permit you to enter parentheses in search expressions let you control the order in which operators are evaluated. As noted above, data base management programs follow a fixed order of precedence when evaluating search terms; if you're unaware of that order, you can obtain spurious results in your search. Parentheses let you tell the computer which expression to evaluate first. Just as it's good practice in mathematical calculation to use parentheses liberally, so too is it in writing search terms. The search question "(porpoise OR dolphin) AND whale" doesn't need the parentheses to work correctly, since the OR expresssion is evaluated first with most programs, but it helps immensely in clarifying the search term's logic. In some cases, adding the parentheses changes the search term's meaning. Consider, for instance, "porpoise OR (dolphin AND whale)." In this search question, the computer is instructed to look for those records that contain "dolphin" together with "whale" OR records that contain "porpoise."

QUANTITATIVE OPERATORS. Some data management programs let you include quantitative operators, such as LESS THAN, LESS THAN OR EQUAL TO, EQUAL TO, GREATER THAN OR EQUAL TO, and GREATER THAN in search questions. One could enter a search question such as "Find all the records which contain a date GREATER THAN OR EQUAL TO 1967." It's important to remember, however, that to the computer everything that's represented for processing is a number, even the letters A through Z. So long as you understand the order in which ASCII characters are represented (the *ASCII collating sequence*, Table 1), you can make use of these operators even when you're searching for nonnumerical material. The word "enormous," for example, is less than the word "minuscule," since it comes before "minuscule" in the collating sequence.

TABLE 1
THE ASCII COLLATING SEQUENCE

Control characters
Space
Punctuation characters !"#$%&'()* + - /
Numbers 0 through 9
Punctuation characters :;◄ = ►?@
Upper-case letters A through Z
Lower-case letters a through z
Additional punctuation characters [|] _'{,}
Extended characters (foreign language, technical, graphics)

Creating a View

The result of a search operation is a subset of the data base—that is, a *view*—that contains only those records that meet the specified criteria. A view is a way of temporarily reducing the size of a data base so that it includes only those records in which you're interested at the moment (for instance, "string quartets AND piano trios but NOT those of the eighteenth century"). Creating the view does not harm or restructure the information contained in the data base.

Printing

Most programs permit you to print out the whole data base or a view of it, producing a *report* of its contents. The better programs let you specify which fields you want printed and give you a good degree of control over the appearance of the report.

A desirable feature for scholarly data base management is the ability to print the data base to an ASCII text file (using only the standard characters as defined by the American Standard Code for Information Interchange). ASCII text files can be read by word processing programs, meaning that the information printed out from the data base can be utilized directly, without retyping, when you sit down to write about it.

DATA BASE MANAGEMENT SOFTWARE

A data base management program, in the broadest sense of the term, is any program that:

Facilitates the storage of massive amounts of information in a data base.

Provides commands for sorting, searching, creating views, and printing the contents of the data base.

TEXT-ORIENTED DATA BASE MANAGEMENT

Not all data base management programs are designed to handle extensive amounts of text. Data base management programs can store, organize, and retrieve three kinds of information, and most specialize in just one or two:

Quantitative data such as serial numbers, census data, sales figures, part numbers.

Graphic images such as maps, diagrams, illustrations of parts, X-rays, or computer-created images.

Text such as correspondence, bibliographic citations, research notes, or the full text of scientific or technical articles.

Most of the data base management programs on today's personal computer market are designed to deal with quantitative data (with limited amounts of text), and for good reason: that's where the market is. Businesses make many uses of quantitatively oriented data base management programs for such matters as maintaining inventories, updating mailing lists, and storing customer records. Although these programs are well suited to business applications, they tend to place restrictions on the amount of text that can be stored (a typical limit is about 1000 characters per electronic "index card"). They're of little use, therefore, for scholars. Graphics-oriented data base management programs may appeal greatly to those who wish to maintain data bases of illustrations, maps, or charts, but they're only now becoming available for personal computers. Our concern, therefore, is with text-oriented data base management programs: programs for the storage and retrieval of large amounts of textual information.

A TYPOLOGY OF TEXT-ORIENTED DATA BASE MANAGEMENT SOFTWARE

Text-oriented data base management programs, defined in the broadest possible sense, can be said to include word processing programs, free format information storage and retrieval programs, text-oriented file management programs, and idea processors.

WORD PROCESSING PROGRAMS

It's possible to use a word processing program's search function for simple information management purposes, but it's not recommended if other alternatives are available. You can use the word processing program, for instance, to create a data file, and the search function will help you find portions of the file in which a specified word or two appears. You're given no tools, however, for more advanced searches using logical operators, for sorting the data base, or for printing a selection of records.

FREE-FORMAT INFORMATION STORAGE AND RETRIEVAL SYSTEMS

A free-format information storage and retrieval system (FFISR) lets you see a word processing program to set up your data base. Two kinds of FFISRs are now available for personal computers: *automatic indexing* and *controlled vocabulary* programs.

Automatic Indexing FFISRs

Automatic indexing FFISRs let you create a massive data base that includes as many as several thousand distinct text files, each created with a word processing program. The program automatically indexes every significant word in the whole data base, with the exception of unimportant "noise words" such as article or prepositions. As they create the index, they note where the word is located.

Automatic indexing FFISRs represent the ultimate in free-format data base management. No restrictions are placed on how the material is written up; the software can put an article, a chapter from a book, a loosely organized file of research notes, and a set of bibliographic citations and abstracts into the same massive data base. Furthermore, no special preparation of the word processor-created manuscript file is necessary; you simply tell the program which files you want indexed, and away it goes.

How, then, does the program distinguish one data record from another? It doesn't: automatic indexing FFISRs treat the data record concept arbitrarily. One program (FYI 3000) considers a paragraph of text as a unit, and shows you retrieved paragraphs. Another (ZyIndex) takes a whole disk file as a unit, and shows you the beginning of the file in which the information you want is stored. Pressing a button takes you to all the screenfuls of text in that file in which the search terms you entered are mentioned.

ADVANTAGES. Automatic indexing FFISRs are exceptionally easy to use. Unlike controlled vocabulary FFISRs (see below), you don't have to worry about making up controlled vocabulary key words or putting

symbols in to demarcate the records. They operate very rapidly, moreover, because they search an index (rather than the actual text). This feature makes these programs suitable for truly massive data bases containing up to 10,000 pages or more of material.

DISADVANTAGES. The chief disadvantage of the automatic indexing FFISR is the *full text search* technique it employs. You're not searching a carefully controlled and deliberately selected set of search terms, the way you do when you use a controlled vocabulary FFISR. On the contrary, you're searching through the entire text, and that means you're certain to retrieve a high proportion of irrelevant material. Here's an illustration.

Let's suppose you want to search for material on Type I supernovas in a data base of information on supernovas. The program will take you with supreme accuracy to all instances of the term "Type I." Among those instances will be several passages that are mainly about Type I supernovas. That's fine; that's the target. The problem is that you'll also see material that's about something else but mentions Type 1 supernovas peripherally. Notes on Type II supernovas, for example, will almost invariably mention Type I supernovas, but the term may be mentioned only in passing.

The major drawback of automatic indexing FFISRs, in sum, is that they can tell you with supreme accuracy what a passage contains, but they don't give you any way to indicate what it is *about*.

EXAMPLE: ZYINDEX PROFESSIONAL. (ZyLab Corporation, 233 East Erie St., Chicago, IL 60611, for the IBM Personal Computer and PC-compatibles), indexes files created with a wide variety of word processing programs.[3] You can create a single data base containing 5000 disk files, each 125,000 words (500K or more) in length. ZyIndex includes an unusually complete set of logical operators, including OR, AND, NOT and WITH/*n*. Available also are parenthetical expressions as well as single-character and multicharacter wild cards, extremely attractive features indeed.

The WITH/*n* operator is of particular value since ZyIndex considers an entire disk file to be equivalent to a single data record. A 10,000-word disk file, for instance, might contain the words "business" and "software" but have nothing to do with business software; indeed, the two words could be separated by a dozen pages. Searching for "business WITH/15 software," however, makes sure that the file won't be retrieved unless the two words are closely related somewhere in the text. What makes this feature so desirable is that it gives you a way to overcome (if only partially) a major drawback of automatic indexing FFISRs, namely, their inability to label what passages of text are about. Using the WITH/*n* operator gives you some assurance that when you search, you'll retrieve a document that's in some way about business software rather than a completely irrelevant one that happens to mention the two terms separated by vast gulfs of text.

ZyIndex's prowess was amply demonstrated during General William Westmoreland's libel suit against CBS in 1984 and 1985. Legal researchers used the program to index and search daily court transcripts, which totaled 10,000 pages at the end of the trial. The program indexed each day's court transcripts, 200 pages in length or more, in about 10 minutes, giving the lawyers by late afternoon a complete record, searchable in seconds, of everything that had transpired in the courtroom up to that point. In the legal setting, having rapid access to massive transcripts can prove invaluable in plotting courtroom strategy. If you need to know just who said what about malice in the trial, for instance, you can find out in seconds.

RECOMMENDED APPLICATIONS. Automatic indexing FFISRs are especially recommended for creating and searching massive textual data bases of unstructured or loosely structured material, such as court transcripts or chronologically organized field notes. Because these programs automatically construct an index to these data bases, they provide sophisticated access to the information the data bases contain without requiring tedious or costly modifications to the information. They're less useful for storing and retrieving material that can be separated into distinct data records, each of which is about something in particular (for example, a data base of notes on specific articles).

Writers can use an automatic indexing FFISR to create a data base of their manuscripts. Suppose, for example, you want to check what you've written and published previously on a particular subject. With your disk-based manuscripts indexed by a FFISR, you can load the program, type in the search terms, and in seconds find all the passages in everything you've written that pertain to search terms.

Controlled Vocabulary FFISRs

Controlled vocabulary FFISRs let you set up a massive textual data base with distinct data records and give you a way to indicate what each data record is about. That's their prinicpal advantage—and, as will be seen, their principal liability.

[3]As of this writing, EasyWriter II, Microsoft Word, Multimate, Palantir, Smart Word Processor, Volswriter, Deluxe II, Wang Word Processor, Word Perfect, WordPlus PC, WordStar, WordStar 2000, Xywrite II, and all ASCII-based text editors such as ProofWriter, Edix, and so on.

Each record is stored with a set of *key words*, or index terms that you deliberately type into the record. The program creates an index that contains only those words (and no others); the text of the record is not indexed. In other words, you control the index's vocabulary.

ADVANTAGES. Like automatic indexing FFISRs, controlled vocabulary FFISRs operate rapidly because they search the index (not the actual text). They are exceptionally well suited, therefore, for truly massive textual data bases.

A controlled vocabulary FFISR lets *you* control the vocabulary with which you'll be searching. The only information a controlled vocabulary FFISR will retrieve when searching is the information you have deliberately placed in the key word field. Controlled vocabulary searching is recommended when it's vital to reduce the retrieval of irrelevant information or to indicate what a data record is about.

Suppose you're setting up a data base of research notes on supernovas. Supernovas are divided into Type I and Type II, and a particular item in the literature is likely to report research on one or the other. But an article on Type I is likely to mention Type II peripherally. If you're controlling the search vocabulary and, after reading an article, you know it's on Type I supernovas, you can put Type I in the key word list (but not Type II). That way, when you're searching for articles on Type I supernovas, you'll retrieve essays that are mainly about Type I supernovas (and none that are mainly about Type II supernovas).

DISADVANTAGES

The chief disadvantage of controlled vocabulary FFISRs is that information in the rest of the data record is not indexed and cannot, therefore, be retrieved if the key word list does not include any reference to it. That can be an advantage. But it often happens that when you're indexing you're looking at an article from a certain angle, even though you may not be aware of it. Later, you might want to look at the same literature from another angle, but you may have neglected to index the article with that second article in mind. It's always hard to predict in advance the different ways you might want to look at your notes or a bibliographic citation in the future.

Another major disadvantage of controlled vocabulary searching is that it's tedious to format the file with the necessary markers and write your own key words. Controlled vocabulary FFISRs can use files you've created with your word processing program, but you have to format the file first with special markers that tell the program where data records begin and end. Moreover, you have to think up key words for each record and type them in. For huge information management projects, formatting the file and sorting all the information into distinct data records could prove prohibitively tedious.

```
*C
Vijaya, Samaraweera, "The Evolution of a Plural Society," in
K.M. de Silva (ed.), Sri Lanka: A Survey. Honolulu: Univ.
Press of Hawaii, 1977.

Abstract: This short essay well summarized the development of
the classic, strife-torn "plural society"—a society marked by
ethnic rivalry, nationalism, and conflict—after the rise of mass
political participation in colonial Ceylon and, especially, after
the island's independence. Particular attention is paid to the rise
of nationalist political organizations (particularly S.W.R.D.
Bandaranaike's Sri Lanka Freedom Party [SLFP]) among the
island's dominant ethnic population, the Sinhalese, which led to
the adoption of Sinhala as the country's "sole official
language," much to the dismay of minority Tamil speakers.

*K

PLURAL SOCIETY / ETHNICITY / ETHNIC CONFLICT /
COLONIAL PERIOD / INDEPENDENCE / TAMIL /
SINHALA / SINHALESE / BANDARANAIKE / SRI LANKA
FREEDOM PARTY / SLFP / 1958 RIOTS / LANGUAGE

*E
```

FIGURE 6. SuperFile Data Record

EXAMPLE: SUPERFILE. SuperFile (FYI, Inc.) well illustrates controlled vocabulary FFISRs. To create a data base with SuperFile, you use your word processing program to write up the material you want to store and retrieve (say, a set of bibliographic citations and abstracts). So that SuperFile will know how to demarcate the data records, you insert special signal characters, prefaced with an asterisk, to inform the program how to tell where one record ends and the next one begins. Figure 6 shows a typical SuperFile entry, the example being drawn from a bibliographic data base. The *C marker tells the program that a data record begins; the *E marker tells it that the record has ended.

The *K marker tells SuperFile that the words to follow are key words; that is, they're words that are especially pertinent to the content of the record. SuperFile searches only for the words you've listed in this special key word section. You may include up to 250 key words in each key word field.

This example points up the limitations of FFISRs in general. Suppose, for example, you wished to search for "nationalism," a term that's relevant to this data record. But SuperFile won't retrieve it. You weren't looking at this citation from that angle when you wrote the key words; you were concentrating on ethnicity

and language conflict. SuperFile would tell you that no records existed that contained information on this topic when in fact there is at least one (and perhaps many more).

Like automated indexing FFISRs, controlled vocabulary FFISRs let you create truly massive textual data bases (even when you're using a floppy disk drive-based system). SuperFile, for example, lets you create a massive data base containing up to 20,000 separate disk files, each containing millions of characters. The data base can be distributed over a maximum of 255 floppy disks.

SuperFile gives you a powerful set of searching options. You can construct a complex, well-focused search question using logical operators, the connectors AND, OR, and NOT, to form search questions such as "show me all the data records that mention 'Sinhala' AND 'language conflict' but NOT the ones that mention '1958 Riots.' "

RECOMMENDED APPLICATIONS. Controlled vocabulary FFISRs are an excellent choice when you want to create a large textual data base and, at the same time, exercise conscious control over how the information in it is organized for retrieval purposes. These programs operate rapidly and can work with massive amounts of text. At the same time, they let you demarcate distinct data records and indicate precisely what the text in the records is about.

An astronomer whose research notes include entries that are about Type I supernovas (but contain references to Type II supernovas), for instance, will appreciate the ability to label a record as being about Type I supernovas. Moreover, controlled vocabulary searching can provide a way to bring about a working version of Vannevar Bush's MEMEX, with its permanently encoded trails of association that provide illuminating pathways through the material.

Bear in mind, however, the limitations of the controlled vocabulary FFISR: you have to format the file with markers and create your own key words. That's a lot of work. If you start with a massive amount of material (say, 3000 pages of field notes), you might find the job so tedious (or expensive) that you'll give up before finishing the project. In such cases, automatic indexing FFISRs are recommended. Controlled vocabulary FFISRs are excellent choices when you're starting a data base from scratch and plan to add small amounts of material to it steadily (for example, a page or two of reading notes per day).

TEXT-ORIENTED FILE MANAGEMENT PROGRAMS

Text-oriented file management software (FMS) represents the next step up in complexity and power from free-format information retrieval systems. The major advantage of a text-oriented FMS is that you can combine the virtues of controlled vocabulary and full text searching without suffering either one's liabilities.

Text-oriented file management programs do not require you to create a data base with your word processing program. Instead, they have their own word processing functions (which are rudimentary, but sufficient for their intended purposes) built in. The most important difference is that a file management program lets you design your own data record format, using a pattern of named data fields (such as, for instance, "author," "citation," and "annotation") that appears on each data record. You can sort the records on any of the fields you've defined (you can sort your bibliography, for example, by author, by date, or by call number). When you're printing, you don't have to print the whole record; you can print only the fields you want printed (leaving out, for instance, the abstracts in a bibliographic data base so that only the citations are printed). The information contained in the data records is, in short, highly structured, and that gives you the ability to manipulate it in many ways.

CITATION	Vijaya, Samaraweera, "The Evolution of a Plural Society," in K.M. de Silva (ed.), Sri Lanka: A Survey. Honolulu: Univ. Press of Hawaii, 1977.
ABSTRACT	This short essay well summarizes the development of the classic, strife-torn "plural society"—a society marked by ethnic rivalry, nationalism, and conflict—after the rise of mass political participation in colonial Ceylon and, especially, after the island's independence. Particular attention is paid to the rise of nationalist political organizations (particularly S.W.R.D. Bandaranaike's Sri Lanka Freedom Party [SLFP]) among the island's dominant ethnic population, the Sinhalese, which led to the adoption of Sinhala as the country's "sole official language," much to the dismay of minority Tamil speakers.
KEY WORDS	PLURAL SOCIETY, ETHNICITY, ETHNIC CONFLICT, COLONIAL PERIOD, INDEPENDENCE, TAMIL, SINHALA, SINHALESE, BANDARANAIKE, SRI LANKA FREEDOM PARTY, SLFP, 1958 RIOTS, LANGUAGE

FIGURE 7. FMS Data Record

Advantages

To illustrate the advantages of text-oriented FMS, consider setting up a data base with a field for a record's full text and a second field for controlled vocabulary key words (Figure 7).

Suppose you're interested in nationalism, but only as it pertains to Sri Lanka. To retrieve relevant records, you could search in the following way: "Find all the records in which the field ABSTRACT contains NATIONALISM and the field KEY WORDS contains SRI LANKA." Note that searching this way returns all the records that mention nationalism in the full text of the abstract, as long as they also mention Sri Lanka in the key word field. What you've done, in essence, is pointed the search with great accuracy toward precisely those records which are about Sri Lanka and mention nationalism in any way.

To illustrate the great virtues of this search flexibility with another example, let us return to Type I and Type II supernovas. A record that's *about* Type I supernovas will have that term in its key word list. You can pinpoint your search, then, to just those articles that are about Type I supernovas without worrying about retrieving irrelevant articles that mention Type I supernovas only peripherally (but are really about something else). At the same time, you can also search the full text of the abstracts for concepts or terms you may have neglected to index. You can search, for example, for all the records which are about Type I supernovas but which also mention the Magellanic Clouds in the abstract field. And if you want to see every record that mentions Type I supernovas, however peripherally, you can do a full text search of the abstract field.

Disadvantages
The major disadvantages of text-oriented file management programs are their slow speed and limited data base size.

Because most of these programs search the full text of each and every record, rather than an index, operation becomes increasingly sluggish in direct proportion to the number of records in the data base. A good FFISR can search 100 data records per second; a file management program may take from one to several minutes to do the same job.

These programs usually limit the size of the data base to disk capacity. If you're using a hard disk, of course, you'll have enough disk space to create a large data base, but the slow operation of the FMS then becomes a problem.

Example: Notebook II
(Pro/Tem Software, 814 Tolman Drive, Stanford, CA 94305, available for the IBM PC and PC-compatible computers; Notebook I, an earlier version, is available for most computers.) Notebook II is a text-oriented information management system that's in a class by itself. Designed for scholars and scholarly applications, Notebook II is a sophisticated file management program that's specifically designed for the storage and retrieval of text.

In its IBM PC version, Notebook permits you to enter up to 30,000 characters in each of up to 50 data fields per record, and gives you the full panoply of FMS utilities (searching, logical operators, sorting, selection, and printing). The program is exceptionally easy to use. Scholars will appreciate Notebook II's ability to make full use of the IBM PC's extended charcter set, which includes many foreign languages and scientific symbols. Notebook II is of special interest to scholars because it's designed to work with a particularly ingenious Pro/Tem product called Bibliography.

Example: Microsoft File
(Microsoft Corporation, 16011 N.E. 36th Way, Redmond, WA 98073, for the Apple Macintosh) Microsoft File can be viewed as a text-oriented file management program. Up to 1023 data fields may be defined for each data record, and each data field may contain up to 32,767 characters. All the foreign language and scientific symbols available with Mac's Option key are available to Microsoft File. In addition to its excellent text-handling features, Microsoft File permits the user to establish numerical fields and provides tools for performing computations on them (however, no built-in statistical functions are provided).

Recommended Applications
Because of their slow speed and data base size limitations, text-oriented file management programs aren't well suited to storing and retrieving massive textual data bases containing thousands of pages of text. They're better suited to storing and retrieving information from smaller data bases, such as a scholar's annotated bibliography of several hundred citations and abstracts.

IDEA PROCESSING PROGRAMS

To include idea processing programs such as ThinkTank under the rubric "data base management software" is to stretch that term's definition. Idea processing programs, are, in essence, aids for creating an outline, and they were originally intended for writing applications. Nevertheless, for certain purposes such programs can indeed be viewed as tools that can facilitate the storage and retrieval of information, and they're considered here from that angle.

What makes certain idea processing programs suitable for data management purposes is their provision of text storage feature. The programs that include this feature let you store, under the headings of an outline, free-format entries that can contain tens of thousands of characters. One could create, therefore, an idea processor-based data management system that organizes free-format textual entries under a complex set of headings and subheadings.

+ Outline for Research Monograph
  + Introduction
  + The Problem
  + The Method
  + The Data
  + Results
  + Interpretation
  + Conclusions

FIGURE 8. ThinkTank Outline

Example: ThinkTank
(Living Videotext, Inc.) is the first outline-oriented file management program, and it well exemplifies the strengths (and limitations) of this type of software for data base management. You can use ThinkTank to create a complex outline, which could, for instance, represent a plan for a research monograph (Figure 8).

The plus signs (+) in front of the headings indicate that they precede a subordinate heading (or headings) that may be visible or hidden from view. The hidden, or collapsed, headings may be made visible if you wish (Figure 9).

+ Outline for Research Monograph
  + Introduction
  + The Problem
    + Survey of the Literature
    + Defining the Problem
    + Statement of Hypothesis
  + The Method
  + The Data
  + Results
  + Interpretation
  + Conclusions

FIGURE 9. Expanded Heading

ThinkTank permits you to place up to 20,000 characters (or 900 lines) of free-format text under any subheading of the outline. The program, therefore, isn't merely an outlining program; it's also useful for storing massive amounts of text (up to the limits imposed by your computer's disk drive). In contrast to free-format or file management programs, however, this information is sorted under hierarchically organized headings and subheadings (Figure 10). You may restructure the framework of headings and subheadings if you wish.

Advantages
Managing data with an idea processing program makes good sense when the information to be stored must be subsumed under a complex, hierarchically organized structure. An excellent example is a data base of lecture notes, handouts, and other material for a course. You're not stuck with the structure you've chosen because the program gives you tools for reorganizing it.

+ Outline for Research Monograph
  + Introduction
  + The Problem
    + Survey of the Literature
      + Jenkins 1956
      + Rodale 1959
      + Peterson and Roberts 1962
      + Radkin 1968
      + Chen 1975

      Chen, Li, "Urban Development in the Third World," Urban Affairs and Redevelopment 3:5 (September, 197), pp. 23-28.

      Chen directly addresses the theories of Peterson and Roberts (1962) and Radkin (1968) and presents new evidence that contradicts their predictions. Rates of internal migration and seasonal migration in Third World cities, Chen argues, cannot be squared with their theories. Chen introduces a new model which represents a more accurate and sensitive estimator of urban development processes considering several factors connected with migration.
    + Defining the Problem
    + Statement of Hypothesis
  + The Method
  + The Data
  + Results
  + Interpretation
  + Conclusions

FIGURE 10. Free-Format Textual Entry

Disadvantages
Because idea processing programs such as ThinkTank do not include sophisticated search features such as logical operators, they make it difficult to search for information stored in the text areas, which are usually hidden from view. The best guide to the contents of the text areas is the headings themselves. When the information to be stored cannot be readily categorized by the framework of headings and subheadings, idea processors are likely to impede rather than assist data

base management. For that reason, an idea processing program works well for storing and organizing research notes under the organizational framework of a planned writing project, as in the above example. But it would be much less useful for creating a general data base of research notes in which the overall structure is of little prominence or concern. For that purpose, free-format information storage and retrieval systems excel.

Recommended Applications

Because the hierarchical organization of the stored material is so pronounced, idea processing programs are at their best for storing and retrieving data when the overall structure of the material is known in advance. This isn't to deny, of course, that the structure can be changed with the program's restructuring commands. Even so, the software's limited searching capabilities make other programs such as FFISRs or FMS more appealing when the overall structure to be imposed on the material is vague or temporary.

Where that structure is clear, however, idea processing programs can work beautifully as data base management programs. Consider for example, a biolgist who wishes to store her research notes under Linnaean classifications. A major heading for the order Cetacea, for example, could be expanded to reveal the suborders Mysticeti and Denticete; expanding Denticete could reveal further biological classifications or a series of articles listed in alphabetical order by author's last name.

INFORMATION MANAGEMENT HORIZONS

What you've just learned about personal computer data base management programs will pave the way for your exploration of online data base services. These services let you connect your computer, via telephone lines, to huge mainframe computers whose auxiliary memories contain as many as 75 million bibliographic citations and abstracts. After you've made the connection, your computer becomes, in essence, a control terminal of the service's computer.

The software these services use is fully comprehensible in the terms established here. In essence, this software—the software you'll actually use after you connect with the service—is much like the text-oriented file management software discussed above and exemplified by the program Notebook. The bibliographic data records have distinct data fields, and you can search them by using controlled vocabulary techniques (there's a separate key word field) or full-text searching of the abstracts. The logical operators (OR, AND, NOT, and WITH), wild cards (single-character or multicharacter), and parenthetical expressions work just the same way. The only difference is that, unlike personal computer-based FMS, the mainframe software creates a FFISR-like index and searches it instead of the individual records. The result is extremely fast retrieval time even with data bases containing a million or more records.

Online data base services provide a significant part of the personal computer's MEMEX-like capabilities, and you'll surely want to explore them. For now, learning how to use a personal computer-based data base management program provides excellent training and preparation for going online. Remember that when you're doing online searching you're being charged for every second you're connected, so online searching isn't a good way to become familiar with the intricacies of searching with logical operators. It's far better to practice with a personal computer-based data base management program first.

RESOURCES

On creating a data base with a word processing program, see Dara Pearlman, "Managing Data with a Word Processor," *Popular Computing* (Feb. 1984), pp. 160]163, and her "Throw Out Your Index Cards," *PC: The Independent Guide to IBM Personal Computers* 4:4 (Feb. 19, 1985), pp. 331-332. For extensive illustration, see Chapter 6 of my *Macintosh for College Students* (Berkeley: Sybex Computer Books, 1984).

For a readable overview of theoretical issues in the design of data base management systems, see Michael Lesk, "Computer Software for Information Mangement," *Scientific American* 251:3 (Sept. 1984), pp. 162-173. A standard technical work on data base software technology for mainframes and minicomputers is C.J. Date, *An Introduction to Data Base Systems: Vol. 1*. 3rd ed. (Reading, MA: Addispn-Wesley, 1981), and *An Introduction to Data Base Systems: Vol. II* (Reading, MA: Addison-Wesley, 1983). Less technical is Date's lucid and readable work on data base management from the user's viewpoint, *Database: A Primer* (Reading, MA: Addison-Wesley, 1984).

For do-it-yourselfers, a fine introduction to data file programming (suitalbe for any computer which runs BASIC, not just the IBM PC) is Alan Simpson, *Data File Programming on Your IBM PC* (Berkeley: Sybex Computer Books, 1984).

An excellent automatic indexing FFISR is FYI 3000 (FYI, Inc., P.O. Box 26481, Austin, TX 78755, for the IBM Personal Computer and PC-compatibles). See Hunter McCleary, "FYI 3000: The Unconventional Database Management Program," *Database* 7:4 (Dec. 1984), pp. 49-53.

[***Ed. Note:*** Portions of the above appeared in Contributing Editor Bryan Pfaffenberger's *The Scholar's Personal Computing Handbook: A Personal Guide*, published by Little Brown and Company.]

Bryan, an anthropologist, teaches in the College of Engineering and Applied Science at the University of Virginia in Charlottesville.