Information Technology: Research and Development (1982), *1* (73-96)

AN OVERVIEW OF INFORMATION SYSTEMS

W. B. CROFT

Department of Computer and Information Science,
University of Massachusetts, Amherst, MA 01003, USA

(*Received 8 September 1980, revised 22 December 1980*)

ABSTRACT

Information systems are being studied and used by a large section of the computer community. However, the research in this area is usually regarded as directed at one of three distinct types of system — database management systems, document retrieval systems and question-answering systems. This paper describes some of the similarities between these systems and how they may be merged to form an integrated information system. A descriptive model of a general information system is used to provide a framework for an overview and comparison of the three types of system.

The term 'information system' has been applied to an extremely broad range of computer systems, encompassing everything from business accounting systems to the much more complex 'knowledge-based' systems which can take the role of a subject expert in a dialogue with the user. Furthermore, much of the research applicable to information systems has been developed in the very specific areas of database systems, document retrieval and artificial intelligence, and it is not obvious how new techniques in one area may affect techniques used in the other areas or indeed how they affect the design of information systems in general. Therefore it is difficult to decide exactly what properties or features an information system should possess. In the first part of this paper, the important features will be defined by introducing a very general descriptive model of an information system. Three main types of information system, namely database management systems, document retrieval systems and 'question-answering' systems, will then be discussed in terms of this model. The purpose of this discussion is to make clear not only the differences but also the many similarities between these systems. These similarities, which are a major topic of this paper, could be exploited by using techniques from one type of system to improve other systems. A number of examples will be given of the most promising areas of overlap between systems. Finally, the idea of an integrated information system will be presented along with a brief description of two current systems that may provide the impetus for this integration. For another discussion of the general features of information systems, see Minker (1977).

1. A GENERAL INFORMATION SYSTEM

The components of a general information system are shown in Figure 1. This diagram, although very simple, describes the important parts of an information system according to the following definition of its function:

> An information system retrieves information from its database in response to a user's query.

This statement does not say anything about the form of the information retrieved, how it is stored or how it is retrieved. It does not even imply that a computer is necessarily involved, although it will be assumed that we are indeed talking about computer information systems. The importance of this definition is that it identifies the parts of a system which are of interest in this paper. For example, database management systems are not designed solely for information retrieval; they also provide a programming environment to reduce the cost of producing and maintaining application programs. However, by concentrating on the information system function as defined above, we can directly compare a database management system to a document retrieval system or a question-answering system.

This definition of an information system does imply a particular group of users. This type of user, commonly referred to as the *end user*, is primarily interested in retrieving information rather than in inserting, modifying or deleting information. Typical end users in a business information system for a parts supply company would be the executives and the store clerks. The needs of other groups of users, such as application programmers, do not have a direct effect on the design of all information systems and so they are not a major consideration in this paper. However, some of the issues involved with updating information are considered in the sections on file organizations and database management systems.

Each of the components shown in Figure 1 varies in design depending on the particular information system being discussed. For example, the input to the system, which is the users' queries, can vary from formal program-like specifications in the case of database management systems to natural language questions in a question-answering system. Similarly, the output can be graphs, tables or natural language sentences.

The search/retrieve module uses the information requests to search the database and retrieve the appropriate data for presentation to the user. In the case of database management systems, this process is particularly simple since the retrieved items must exactly match the query specification. For both document retrieval systems and question-answering systems, the search process is more complicated and may involve inference. One feature the various systems have in common is that the search process will include an interactive dialogue between the user and the system. This means that the systems must respond to on-line information requests within a reasonable time.

The form of the information stored in the database varies between systems. In a database management system the data consist of well-defined entities and relationships between those entities. The data are well-defined because they have been tailored to a particular application. The data in a document retrieval system consist, at least initially, of a collection of text documents such as journal papers or office memos. In a question-answering system, the data are complex representations of some field of knowledge.

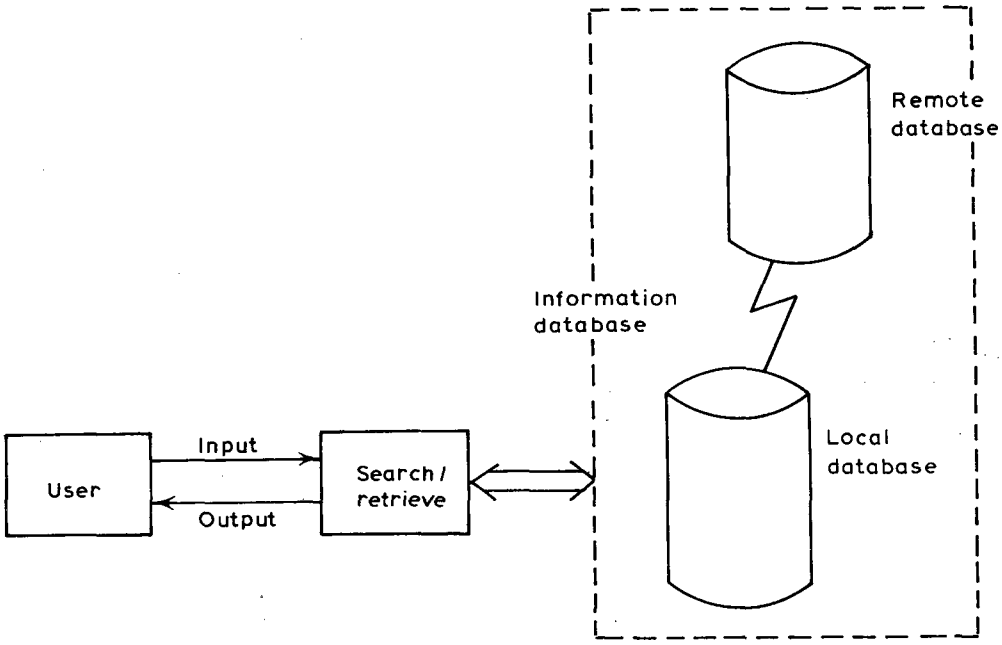

FIG. 1. A general information system

So far it is clear that there are considerable differences between the various types of information system. However, in terms of the low level storage of the information database, there is much common ground. File organizations, which are used to provide efficient, on-line access to the data, are common to all systems. Another important aspect of the information database in these systems is indicated by the communication link in Figure 1. This communication link shows that the information may in fact be distributed over a number of locations. That is, as well as the data which are stored locally, the system may have access to data stored on other devices or systems. Indeed, in some cases the local database may be incomplete if it cannot access the other databases. Networks capable of connecting many computers and storage devices have important implications for the design of information systems and together with file organizations they will be discussed in the next section. In the later sections, the three main types of information systems will be examined in more detail and compared to each other.

2. DATA STORAGE IN INFORMATION SYSTEMS

In this section three possible components of an information system will be examined in more detail. All three are related to the storage of information in the database. File organizations and content-addressable memories provide access to the data on the storage devices. Computer networks make the distribution of both the data and the system itself possible.

2.1 File organizations

A file organization is a means of structuring data (or information) on physical storage devices so that they can be retrieved, stored or deleted efficiently. That is, with the appropriate file organization these operations will be done economically in terms of computer time and storage. All information systems use physical storage devices and therefore they will all have file organizations. There have been many good reviews of the various file organizations (Knuth, 1973; Martin, 1977; van Rijsbergen, 1979) and only a few of the main ones will be mentioned here.

An important distinction to be made for file organizations in information systems is between the *primary* and *secondary* keys of a *record.* A record in the database is a collection of items of information. Generally, it is the smallest unique unit of information referred to by the programs in the system. An example of a record would be a collection of information about an employee of a company. This would be referred to as a particular employee record by the company's information system.

A primary key is the part of the record that uniquely identifies it. Following the example above, the primary key of an employee record would be the employee number assigned by the company. A secondary key refers to a part of the record that is a distinct item of information but that does not uniquely identify the record. That is, more than one record may have the same value for this item of information. Examples of secondary keys in the employee record are the employee's age, birthdate, sex, salary and department.

File organizations in information systems concentrate on retrieval by secondary keys because, in most cases, the user will not know the unique primary key that identifies the record containing the information he wants. Instead, records are usually described by specifying parts of their *content* or, in other words, some of the secondary keys.

One obvious file organization, which is hardly an organization, is simply to store all the records serially on the storage device with no relationships specified between them. To find the records containing a particular set of secondary keys with this organization, all the records in the database must be searched sequentially. Even when a record is found that contains the correct keys, it does not contain any indication of where to find other records that should be retrieved. This *serial file* organization is very efficient in terms of storage since there is no overhead at all, but for large databases the time taken to locate the desired records will be unacceptably long. Other file organizations must then be considered that will decrease the time required to find records but that will necessarily involve larger storage overheads.

The most important organization for secondary key retrieval is the *inverted file.* For each secondary key, the inverted file gives a list of records containing that key. Records containing specific sets of secondary keys can now be found by scanning the appropriate lists. This takes much less time than a serial search. Table 1 compares an inverted file with a serial file for a simple database.

Table 1. Inverted (b) versus serial (a) files

(a)

Emp. no.	*Age*	*Salary*	*Dept.*
123	29	20 000	2
152	35	30 000	2
153	45	28 000	1
162	26	20 000	1

(b)

Age	*Emp. no.*	*Salary*	*Emp. no.*	*Dept.*	*Emp. no.*
26	162	20 000	123	1	153
29	123		162		162
35	152	28 000	153	2	123
45	153	30 000	152		152

The inverted file usually contains pointers to the records in a serial file. In other words, the inverted file is a *directory* for the serial file. The pointers to the required records are found by scanning the inverted lists and then the serial file is accessed to retrieve the full records. This means that, compared to the serial file, using the inverted file as a directory can represent as much as 100% storage overhead (approximately) if every secondary key is inverted. Some of the directory overhead can be reduced by combining the inverted file with the *multilist* organization. In the multilist file there is a list header for each secondary key that points to the start of a chain of records containing that key. The storage overhead, which is the pointers in the chains, is now mostly in the records themselves and the directory overhead is much smaller than for an inverted file. However the chains take longer to search than inverted lists, so hybrid organizations have been proposed (Yang, 1978) that use inverted lists for keys that are important for fast record access and chains for less important keys in order to save storage.

The hybrid organization does not remove some of the major problems associated with inverted files. The two worst problems are the high storage overhead and the difficulty of updating the organization when new records are inserted or old ones deleted. The updating problem arises because the inverted lists themselves can be very large and they are usually maintained in ascending order of the records' primary key. Another problem is that the time to retrieve a set of records depends on the number of secondary keys specified. That is, the more secondary keys specified in the request, the longer the retrieval time. Alternative organizations have been proposed (Abraham *et al.*, 1968; Wong and Chiang, 1971; Bentley, 1975; Rivest, 1976) that attempt to solve some or all of these problems. Unfortunately, none of these organizations can efficiently cope with all the record and query types encountered in information retrieval and so the inverted file is still very common.

So far we have only mentioned single file organizations, for example, a file of employee records. In any real information system there will be many such files, each containing an important entity or record type. These files do not exist in isolation of each other and there are definite relationships among them. These relationships will be discussed in Section 3 but, as an example, consider a hospital system that contains files of patient records, employee records and accounts records. These records are related in that some employees (doctors) are assigned to specific patients and accounts records refer to specific patients. The multilist file organization can be extended to represent explicitly these relationships across files. This is done by connecting related records in different files with chains. Martin (1977) discusses in detail the physical implementation of the record relationships.

2.2 Content-addressable memories

We have seen that file organizations can involve considerable storage overhead and can also be difficult to update. Recently there has been increasing interest in hardware storage devices that can retrieve information specified by content rather than by address or primary key. These *content-addressable memories* can perform the same function as, for example, the inverted file but without the associated disadvantages.

There are two main types of content-addressable memory. The first type (Bird *et al.*, 1977; Berra and Oliver, 1979) uses a small special-purpose fast memory that can be searched with secondary keys. The information is read into this memory in stages from a larger, slower memory. When the entire database has been through the

content-addressable memory, the required records will have been found. The second type (Ozkarahan *et al.*, 1975; Su and Lipovski, 1975) puts extra hardware in the secondary memory so that the secondary key search can be done while scanning the database (see Fig. 2).

In both of these cases, the data are stored on the hardware as a sequential file. That is, there are no directories and therefore no storage overhead. The secondary key searching is done entirely by the content-addressable hardware. Apart from the storage savings, a database stored sequentially can be easily maintained.

At first it may seem that content-addressable memories may entirely replace file organizations. However, they are expensive and can be slow in some cases because of the time required to transfer the data from the slow memory. Therefore content-addressable memories are presently used for specific applications rather than for general-purpose information systems.

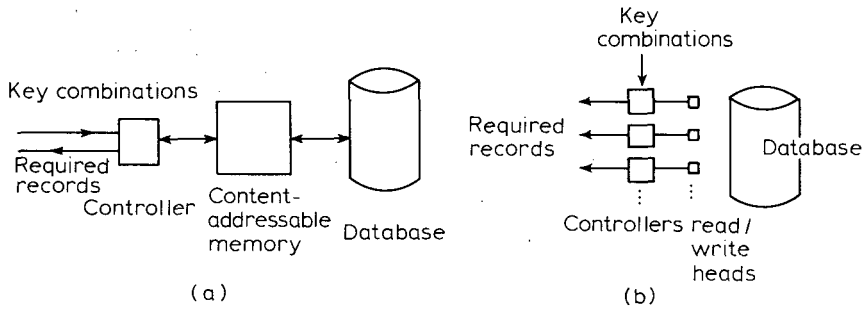

FIG. 2. Two types of content-addressable memories

2.3 Computer networks

The possibility of being able to connect computers and storage devices to each other with links that are both cheap and very fast will have a strong impact on the design of information systems. By using networks the data, the processing power and the tasks of an information system will be distributed among a number of locations and devices.

Networks are implemented at the hardware level through computer communication networks. *Global* computer communication networks have been around for some time and they enable communication between different computers and remote terminals over large distances. Two examples are the ALOHA net (Abramson, 1973), which was originally designed to link remote computer terminals to a central computer and the ARPA net (Kleinrock, 1975) which links many computers of different types. This type of network is expensive and has relatively slow data transfer rates.

Local communication networks have recently attracted a great deal of attention. This type of network is designed to operate over smaller distances than a global network such as, for example, within the premises of a large company. The advantages of local networks are that they are cheap, fast, reliable and can be adapted to many applications. The most important local network designs are the Xerox ETHERNET (Metcalfe and Boggs, 1976) and the Cambridge Ring (Wilkes and Wheeler, 1979).

The distributed systems that use these local networks consist of a number of nodes connected to each other (Wilkes and Needham, 1980). Some of these nodes are computers of various types, some with large local memories and some with very little. Other nodes, called *servers* (which may also be computers), provide services to the whole network. For example, a *file server* would provide a common place to store and modify data in the network. Small computers attached to the network would then not need an expensive disk memory for local storage. Of course, it is possible to have both local and central data storage as well as multiple file servers. This leads to the type of problems of distributed data mentioned in Section 3. Another example of a server is a *printer server* that provides a printer for all the computers in the network. Non-intelligent terminals in this distributed system are connected either directly to computers in the network or to terminal concentrators, which are also nodes in the network.

3. DATABASE MANAGEMENT SYSTEMS

The concept of a database management system (DBMS) arose from experience with computer systems in a commercial environment. In this environment it was found that the major software cost was in maintaining old application programs or writing new programs when the information in the database was changed in some way. Extensive reprogramming could be required even for minor changes such as including a new item of information in the records. Therefore one of the major aims of a DBMS is to make application programming easier and to make the programs cheaper to maintain. Despite this emphasis, a DBMS provides a powerful information system for users who are not programmers. In this section, we will concentrate mainly on this aspect of a DBMS rather than giving a comprehensive summary of its facilities. For a more general introduction to database management systems see either Date (1977), Martin (1977) or Ullman (1980).

Referring back to Figure 1, there are five components of the general information system. They are

1. The users of the system
2. The input to the system
3. The search/retrieve module
4. The information database
5. The output from the system.

The users of a DBMS are usually the employees of a single company. However they can generate a very wide variety of information requests. For example, the sales manager of the company may wish to see tables and graphs of the sales in a particular period whereas a sales clerk may just want to know how many items of type A are in the store or on order. As well as producing a wide variety of requests, different users of the system will have different views of the information that is stored in the database. Following the example above, the manager will want to be able to look at all of the sales figures available, prices of items, personnel data and many other things. In contrast, the clerk should only be able to access the information relevant to his task, that is, the inventory figures. To see how these different views are accommodated and, indeed, how application programming is made simpler, we have to look at how the information is stored in the database.

The type of information which is stored in a DBMS is very structured. When a computer system using a DBMS is being designed for a particular company, a person (or more accurately an organizational function) called the *database administrator* will study the company's operation in order to decide what *entities* will be important in the system. The entities are represented by records made up of information items or *data fields*. Each of the data fields has a very specific meaning and a very specific format. The fields usually contain numeric data but often can be strings of characters (such as a name or address). The employee record mentioned in Section 2.1 represents a real employee (an entity) and is an example of structured data where each field has a specific meaning.

The database administrator (DBA) not only specifies the record types and the fields in them but also the *logical relationships* between the records. These relationships describe how occurrences of the record types may be linked. For example, in Figure 3 the relationships between four record types are shown in diagrammatic form (following the conventions used in Martin (1977)).

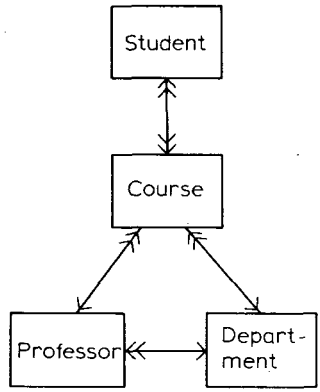

FIG. 3. Four record types and the relationships between them

The four record types are the department record, which contains data items relating to the departments of a university (for example the budget), the professor record, the course description record which, as the name implies, describes the various courses in a university and the student record which describes the students taking the courses. The arrow between the department record and the professor record is an example of a *one-to-many* relationship. This means that each department has many professors but each professor belongs to only one department. This is an example of a real-world constraint that can be represented in this type of diagram. There is a similar relationship between the professor record and the course record and also between the department record and the course record. The arrow between the course record and the student record is an example of a *many-to-many* relationship. Each course may have many students and each student may be taking a number of courses.

There are three main approaches to the design of a DBMS — hierarchical, network and relational (see Martin (1977), Date (1977) or Ullman (1980)). These differ mainly in the complexity involved in expressing the various types of logical relationships. Most of the discussion in this section applies equally to systems based on any of the three approaches.

In summary, the form of the information stored in the database is specified by the DBA. The formal specification of the data fields, record types and relationships between the records is called the *schema* (or conceptual schema). To provide different views of the database for different users, *subschemas* can be specified by either the DBA or the users themselves. The use of subschemas enables the DBA to enforce *privacy* and *security* of the data in the system. That is, users can be prevented from accessing or changing parts of the database. Also, because a subschema can contain completely different record types and even data items than the conceptual schema, the conceptual schema can be changed without affecting application programs that use a particular subschema. This property of a DBMS is called *logical data independence. Physical data independence* means that the actual storage devices or file organizations used by the system can be changed without affecting application programs. This property is also achieved using the conceptual schema together with a separate description of the physical devices and organizations used (the *physical schema*). Another important property of databases defined using schemas is that there is very little redundant storage of data. Apart from the obvious storage savings, this also means that the data are likely to be in a consistent state since updates to the contents of records occur in only one place.

The next component of the information system to be discussed is the input/output of the system. In a DBMS the queries are usually a specification, using a rigid format, of a combination of secondary keys. The combination is often presented as a Boolean expression. The records containing this combination of keys are then retrieved from the database and displayed in a form that is also specified by the user. An example of this type of query expressed in a pseudo-natural language form would be 'List the names and departments of all employees who earn more than $20000 and who work in department 2 or 3.' The secondary keys here are the salary (K1) and the department (K2) and two fields of the retrieved records are to be displayed. The appropriate Boolean expression is (K1>20000) AND (K2=2 OR K2=3). The keys that can be mentioned in the query are exactly those data fields mentioned in the subschemas. Therefore the query has a very well-defined meaning.

Some effort has been made to reduce the burden of a formal query language on the users. An example of an interface designed for end users is the Query-by-example system (Zloof, 1975). In this system, the users formulate their queries by giving examples of possible answers. This interface does, however, still require some expertise in its use.

The search/retrieve module uses the queries to locate and retrieve the specified records. In a DBMS this module is particularly simple. The query exactly specifies a combination of keys and the main function of this module is to efficiently locate the records containing this combination. This type of query-record matching we shall refer to as an *exact match.* The concern for efficiency can lead to complicated problems in optimizing the query evaluation (Yao, 1979) but this does not change the set of records that is eventually retrieved.

Finally, it should be mentioned that it is in DBMS research that the most attention is being paid to distributed information systems. A number of problems are being studied that would arise with any system with distributed data (Adiba *et al.*, 1978). Some of these are

— Should multiple copies of the data be stored?
— Where should the data be stored?
— Which system tasks should be performed at the various nodes in the network?

— How is the user directed to the correct data?
— How can the security and privacy of data be maintained when the data are stored in a number of places and can be updated by several processes operating concurrently?

This section has discussed how a database management system can provide secure and efficient access to structured information. The central definition of the data in the system using a schema enables large systems to be modified without extensive reprogramming. These properties have led to the general use of these systems in the commercial computing environment, but there are limitations in their use as information systems. These limitations are in the form of the query specification, the restricted types of search and the type of data handled by the system.

4. DOCUMENT RETRIEVAL SYSTEMS

Both the nature of the information stored and the algorithms used for searching and retrieving this information make document retrieval systems fundamentally different to database management systems.

First, let us consider the information in the database. A *document* is a body of natural language text such as a journal article, a newspaper article or even an office memo. The function of a document retrieval system is to retrieve those documents or, more frequently, *references* to those documents which are *relevant* to the user's query. The notion of relevance will be discussed later but it is clear that the query will, in some way, have to be compared to the content of the documents. But, whereas for the DBMS this comparison is simple because the data is highly structured (for example, the employee record), in the case of a document retrieval system the raw data (the text) is very unstructured. In some systems the full text of the documents is stored in the database (Mead Data Central, 1975), but in most cases the raw data is structured by representing the contents of the documents by a set of keywords or terms. The process of representing text by terms is known as *indexing* and this can be done either manually or automatically (see Sparck Jones (1974) for a survey of automatic indexing techniques). A typical automatic process involves the following steps

1. Remove common words (e.g., and, the).
2. Produce stems by removing suffixes.
3. Replace stems by numbers and count occurrences of these stems in the text.

The text used for indexing is usually an abstract or summary of the document rather than the full document. The indexing process just described is very simple and usually produces reasonable document representatives. More complicated statistical models can be used for indexing in an effort to produce better representatives (Harter, 1975).

The records in a typical document retrieval system database consist, therefore, of some document identification data (title, authors, journal, date, etc.) and a set of index terms which may or may not have weights associated with them to indicate their relative importance to the documents. These terms are similar to the data fields of a DBMS record but there are important differences. A term in a document has a less well-defined meaning than a field in a typical DBMS record. For example,

describing a document by the term 'programming' is less specific than describing an employee by the data item 'salary = $20000'. Also, the number of terms that describe a document varies widely compared to the relatively stable number of fields in a DBMS record and, more importantly, the terms in a document representative are a small selection from a very large vocabulary (the English language) whereas the number of different fields in a typical DBMS application is much smaller.

In a DBMS the logical relationships between different record types are recorded in a schema. In a document retrieval system, there is really only one type of record (the document representative) but some types of relationships are stored in the database. A statistical technique called *document clustering* (van Rijsbergen, 1979) can be used to group documents (we shall drop the word representative) that have similar sets of index terms. This technique attempts to make explicit relationships between documents whose contents are semantically similar. The same type of technique can be applied to index terms to indicate which terms tend to occur together. A *term clustering* attempts to relate, using an automatic method, terms that are synonyms or components of a common phrase.

We can, therefore, regard the information database in a document retrieval system as a network (Fig. 4) in which there are two types of nodes — documents and terms. Connecting these nodes there are links that represent relationships between them. Associated with a link is a weight that indicates the strength of the relationship. These links and weights are derived using the statistical procedures mentioned previously. There are document–term links that indicate what terms represent the content of a document and how important they are, document–document links that connect documents with similar contents and term–term links that connect similar terms.

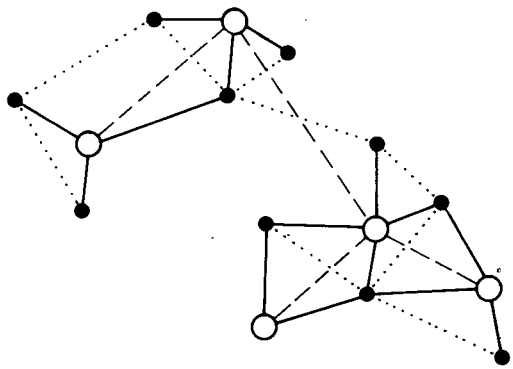

FIG. 4. A document retrieval database viewed as a network
(o = documents; ● = terms; unbroken line = document–term links;
broken line = document–document links; dotted line = term–term links)

The users of a document retrieval system are more homogeneous than those of a DBMS. They are all interested in retrieving relevant documents and they want the system to search all the literature it knows about, perhaps within certain dates. Although different users may be interested in different levels of performance by requiring, for example, a few highly relevant documents or as many relevant

documents as possible, it is difficult to separate the needs of these users in the same way that the needs of a manager and a store clerk can be separated in a DBMS. The usual type of query is a description of the content of documents that the user wants. In most commercial systems such as DIALOG (Lockheed, 1976), this means that the user will specify a Boolean combination of terms. However, in most experimental systems, it is assumed that the user will give a natural language statement of interest that will be analyzed with the same indexing techniques that were used on the documents (Salton, 1968). Another way of describing the content of the required documents is to give the system an example of a relevant document.

The search/retrieve module can be either very simple or quite complex. In the commercial systems, the documents are retrieved by doing an exact match with the query. That is, documents containing the Boolean combination of terms specified in the query are retrieved. However the user will not necessarily consider these documents to be relevant because the query, whether in Boolean or natural language form, is really only an indication of the user's interest. Given the uncertainty involved with the notion of relevance, it does seem unreasonable that a system will entirely exclude documents that do not quite match the query specification. Similar objections can be raised to full-text systems (Mead Data Central, 1975) where the retrieved documents must contain exactly the text string specified by the user. Therefore in experimental systems, instead of using this all-or-nothing approach to retrieval, the documents are ranked in decreasing order of probability of relevance. That is, the first documents presented to the user will be those most likely to be relevant.

An obvious method of producing this document ranking is to use a *similarity measure* (van Rijsbergen, 1979) that measures the number of common terms between a query and a document normalized in some way to be a number between 0 and 1. An example of a similarity measure for documents and queries represented by binary index terms is Jaccard's Coefficient,

$$\frac{|Q \cap D|}{|Q| + |D|}$$

where Q is the set of query terms, D is the set of document terms, and $|\ |$ is the modulus of the set.

Recently, a great deal of theoretical and experimental work has been done in this area which has clarified many of the techniques involved with document ranking (Yu and Salton, 1975; Robertson and Sparck Jones, 1976; van Rijsbergen, 1977; Croft and Harper, 1979). Probabilistic models of searching have been proposed that use estimates of how frequently terms occur in the relevant and non-relevant document sets for a given query to infer the probable relevance of documents. It can be shown that, under certain reasonable assumptions, this method of ranking will give optimal system effectiveness in terms of the recall/precision measures mentioned in the next section (Robertson, 1977).

In a document retrieval system, the concept of the *effectiveness* of retrieval is very important. Because the retrieved documents are not necessarily relevant, systems have to be compared by measuring how well the particular system or technique separates the relevant and non-relevant documents. Most of the evaluation measures that are used are based on two quantities, *Recall*, which is the proportion of relevant documents that are retrieved, and *Precision*, which is the proportion of retrieved

documents that are relevant. In an operational system, recall is very difficult to measure directly and so this type of performance evaluation is usually only done with experimental systems.

Other search techniques have been proposed that in some cases, improve retrieval effectiveness. One of these uses (in effect) the document-document links shown in Figure 4 by searching *document clusters* (Croft, 1980). Instead of comparing the query to the separate documents in the database, it is compared to the representatives of document clusters. Because clustering the documents tends to eliminate individual differences and indexing mistakes, the effectiveness of the system, particularly in terms of precision, can be improved. Similarly, the term-term links in Figure 4 can be used to give a more effective search strategy (Harper and van Rijsbergen, 1978). In this case the query is expanded by adding terms which are closely related to those mentioned by the user.

An important part of the implementation of the probabilistic models of searching and, indeed, of a number of document retrieval systems, is *relevance feedback*. In this process, originally introduced by Salton (1968), the user is presented with an initial set of retrieved documents that may be, say, the top ten documents from a ranking using the user's query. The user then decides, on the basis of the titles or abstracts shown, which documents in this group are relevant. The system then uses this information about relevance to retrieve a new set of documents. In the simple versions of relevance feedback, the terms in the identified relevant documents are used to modify the user's initial query whereas in the probabilistic model the new information is used to give better estimates of the occurrences of terms in the relevant set of documents.

To summarize this section, document retrieval systems are designed to retrieve text documents and a number of techniques that are not used in a DBMS such as automatic indexing, document ranking and relevance feedback are needed to give effective performance.

5. QUESTION-ANSWERING SYSTEMS

Question-answering systems are information systems that use artificial intelligence techniques. The term itself is a little out of date and 'expert systems' better describes the type of system currently being developed. As the former name implies, these systems are designed to answer direct questions from the user, rather than indirectly satisfying the information need by retrieving documents. For example if a doctor asked 'What is the effect of caffeine on the nervous system?', a question-answering system would reply with a list of the physical effects of caffeine rather than with a list of references to documents about this subject. This means that the system has a much greater 'understanding' both of natural language and of the information it deals with than a document retrieval system. Much of the extra capability comes from a very detailed representation of knowledge stored in the database.

The information or knowledge in the database usually describes a very specific subject. This is not only because systems that are expert in a narrow field of knowledge can be used in many situations but also because the current methods used for representing the knowledge have many limitations. Examples of expert systems are the PROSPECTOR system (Duda *et al.*, 1977) which acts as a geological consultant and the MYCIN system (Randall *et al.*, 1977) which helps physicians decide on treatment for blood infections. Two of the main formalisms used for

knowledge representation are *productions* and *semantic nets* (Nilsson, 1980). A production system consists of a global database, a set of rules or productions which operate on the database and a control strategy for activating these rules. The rules are often derived from an 'expert's' knowledge of a subject as in MYCIN or PROSPECTOR. A production has two parts — the situation recognition part and the action part. The situation recognition part specifies a combination of facts or pieces of evidence from the database that triggers the production. The action part specifies the new facts that are asserted when the production is triggered. The following is a typical MYCIN production:

If the infection type is primary-bacteremia,
the suspected entry point is the gastrointestinal tract,
and the site of the culture is one of the sterile sites,
then there is evidence that the organism is bacteroides.

The facts deduced by one set of productions can trigger other productions and so on. An *and/or tree* represents how a conclusion can be reached by several alternative chains of productions. Figure 5 gives an example of such a tree. The *and* arcs indicate that all the facts mentioned must have occurred before the production is triggered. An *or* arc indicates that any of the facts mentioned can occur for the production to be triggered. In Figure 5 production P2 is triggered only if facts F2, F3 and F4 have occurred. For example, production P2 could be something like 'If a student has taken courses C101, C102 and C103 then the student knows operating systems.' Production P7 could be 'If students know about theory, operating systems and artificial intelligence then they are qualified for a computer science degree.' The

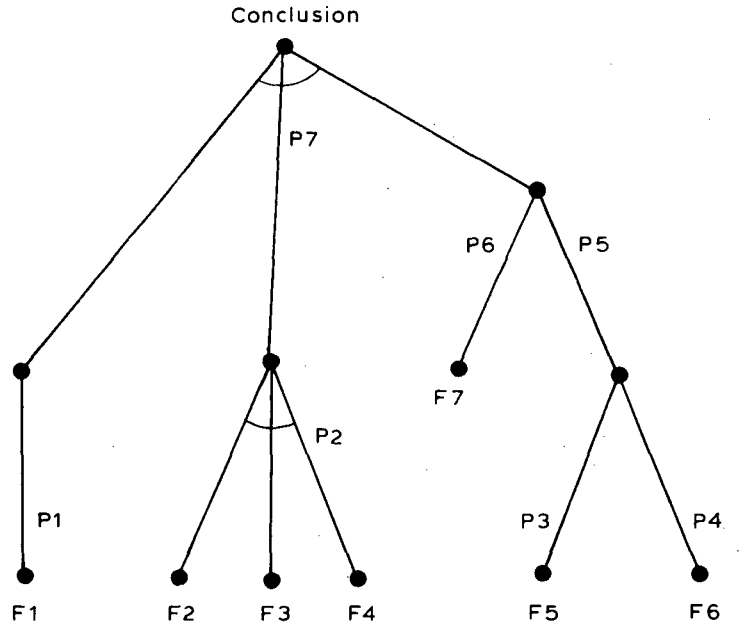

FIG. 5. An AND/OR tree (F = fact, P = production)

other productions would relate to qualifications in theory and artificial intelligence.

The most common way of representing the knowledge in the global database is with graph structures, called semantic nets, which are based on first-order predicate calculus. A semantic net consists of nodes connected by labelled directed arcs. There are two main types of nodes — object nodes and relation nodes. Arcs are used to connect relation nodes to other nodes. Figure 6 gives two examples of semantic nets from the PROSPECTOR system. In Figure 6(a), the semantic net represents the statement 'Entity-1 is composed of rhyolite.' The labels on the arcs are the names of the arguments of this statement. Figure 6(b) shows a way of representing taxonomic hierarchies in a semantic net. This section of the net represents the statement 'Galena is an element of the lead sulfides that is a subset of the sulfides that is a subset of the minerals.'

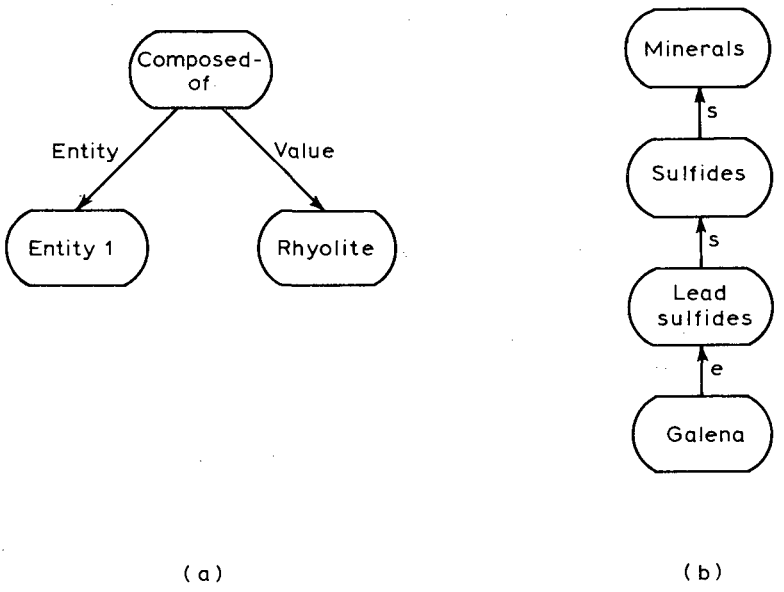

FIG. 6. Two examples of semantic nets

We can summarize the purpose of semantic nets with a statement from (Schubert *et al.*, 1979): 'The knowledge required to perform an intellectual task generally lies in the semantic vicinity of the concepts involved in the task.' This statement says that most of the knowledge about a concept in the database is very closely connected to that concept in terms of the number of links that must be traversed in the semantic net.

As mentioned previously, the knowledge representation in a question-answering system will, as in PROSPECTOR, contain elements of both of these representations. That is, the database will contain specific facts (*extensional data*) and general rules that permit other facts to be deduced from the data (*intensional axioms*).

The search/retrieve module of a question-answering system must use the knowledge representation to answer questions from the users. In general this is a difficult problem for which there is no single solution (see Nilsson (1980) for a

review of AI searching techniques). However, question-answering systems do have the ability to use general rules and specific facts to deduce answers to questions that are not explicitly stored in the database. It is this ability to do deductive reasoning that is a major difference between question-answering systems and database management systems. Document retrieval systems do retrieve by inference, but it is a statistical form of inference rather than the symbolic form used in question-answering systems.

The search/retrieve module will develop an answer to a question by using the facts stored in the database, the general rules in the database and any *heuristic* techniques that may be useful. Heuristic techniques may, for example, be used to prevent the search from going in non-productive directions. As an example of a deductive search, consider a production system made up of several and/or trees similar to that in Figure 6. There are several conclusions that can be reached by a number of alternative chains of productions. A search strategy called *forward chaining* starts at the bottom of the and/or trees using the known facts. It then follows all possible paths in the trees until a conclusion is reached. In some systems it is more efficient to use a strategy known as *backward chaining*. Using this strategy, a conclusion is hypothesized and the tree is searched in a top-down fashion to see if it is possible to work backwards to the known facts. With either of these strategies, the user may be asked to supply new facts to fit in with the productions. Backward chaining has the additional advantage that this process of asking new facts will seem more directed.

The input to a question-answering system is natural language. Rather than processing the language using statistical techniques, the syntax of the input is analyzed and related to the semantics of the knowledge stored in the database. The first part of this process is the *lexical* analysis that identifies the words in the text. A *lexicon* provides a list of words that the system knows about, possibly with some extra syntactic and semantic information (such as what parts of speech the word or a variant may appear in). The syntactic analysis uses a *grammar* to define the valid syntactic constructions for a sentence in the language. The grammars required to describe a reasonable subset of the syntactic forms allowed in the English language can be very complex.

The most common way of representing a grammar in a natural language processing system is with an *augmented transition network* (ATN) (Woods, 1970). Figure 7 gives a simple (not augmented) transition network for the simple *context-free* grammar (Hopcroft and Ullman, 1979) of Figure 7(a). This grammar defines a set of simple English sentences, where A is an adjective, N is a noun, V is a verb and D is a determiner. For example, the fourth rule defines a noun phrase (NP) as an arbitrary number of adjectives followed by a noun. Note how the network implements the recursive definition of the fourth rule. The grammar is made up of a set of *rewrite rules*, *non-terminal* symbols (S, DNP, VP, V, DET, NP, N, A) and *terminal* symbols (not shown). To be able to implement the complexity of a *context-sensitive* grammar, the transition network is augmented by allowing conditions to be tested and actions taken on the arcs of the network.

The output of a question-answering system is also in natural language. Since the answer to a user's question will be obtained in a formal internal representation, a procedure will be needed to *generate* English from the internal description.

We have seen that question-answering systems incorporate powerful techniques for understanding language, representing knowledge and solving problems. Many problems do remain however in designing effective techniques for these systems. The main limitation is that the techniques have been tested only with databases

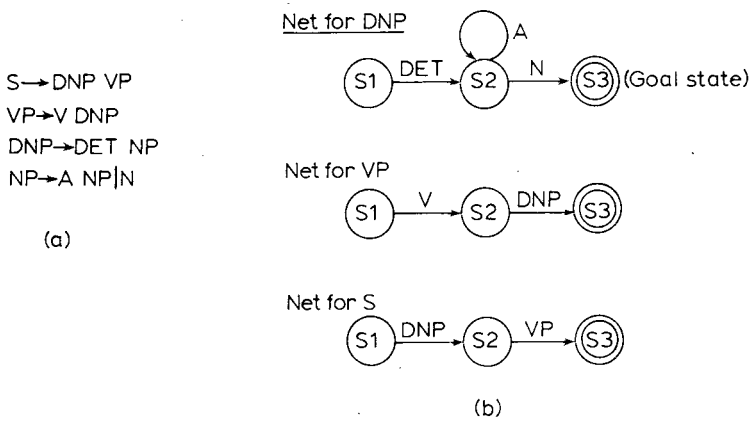

FIG. 7. A transition network for a context-free grammar

containing very specific knowledge. It is not known whether they would work on a larger body of knowledge.

6. THE INTERFACES

Table 2 summarizes the features of database management systems, document retrieval systems and question-answering systems in terms of the model of information systems introduced in Figure 1. In the following sections we will discuss how current versions of these systems could be improved by incorporating techniques from the other systems.

Table 2. A summary of information systems

System	*Input*	*Output*	*Search/Retrieve*	*Database*
DBMS	Boolean combination of secondary keys, primary keys	Required records in specified format	Exact match	Structured data. Records and relationships defined by schema. Provides privacy, security and data independence
Document Retrieval	Boolean combination of terms. Natural language queries analyzed statistically, example documents	List of references to documents	Documents ordered by probability of relevance. Search strategies are such that retrieved documents need not have all terms in common with the query	Less structured data (text). Statistical analysis used to extract terms and to define document–document, term–term relationships
Question-answering	Natural language queries analyzed syntactically	Natural language answers	Uses sets of rules, knowledge representation and heuristics to deduce answers	Complex representation of expert knowledge, e.g., semantic nets

6.1 Database management systems and question-answering systems

An obvious application of AI techniques to database management systems is to provide a natural language interface. The interface would encourage people such as company managers to use the system because these people often find a program-like interface too time-consuming to master. A great deal of research is going on in this area (Hendrix *et al.*, 1978; Waltz, 1978; Kaplan, 1979). One approach in this research has been to use the information in the schema of a particular DBMS to help in the analysis of the natural language. That is, the grammar includes information that is specific to an application. A less system-dependent and therefore more portable system would be produced by using a general interface to convert natural language into a formal internal language. This internal language would then be translated into a string of commands for a particular DBMS.

There has also been widespread interest in improving conceptual schemas so that they will include more semantic information about the data. At the simplest level, this may involve improving the *semantic integrity* of the data by including rules or procedures that describe allowable data values and that can be used to check for data consistency (Hammer and MacLeod, 1975; Brodie, 1978). For example, one of the rules may say that the salary of an employee should be less than the salary of his manager. At a higher level, the conceptual schemas can be extended to look more like semantic nets (Wong and Mylopoulus, 1977 review this topic; see also Bachman and Daya, 1977; Codd, 1979). That is, relationships such as those shown in Figure 6 are included in the schema. In this case some forms of inference may be used in a DBMS system in order to solve a broader range of queries. For example, answering a query like 'What departments are the heaviest users of resources?' would involve an understanding of 'heaviest', 'users' and 'resources' in the context of a particular system. In effect, database management systems may become simple 'expert' systems. However, the full range of AI techniques may not be needed as the type of information stored in a DBMS, because it is tailored to a particular application, is very well-defined. It is, after all, the structured nature of the data that enables a DBMS to provide efficient and secure access to the database.

6.2 Database management systems and document retrieval systems

It is possible that a document retrieval system could be implemented using a DBMS. This would provide an information system capable of dealing with both the structured business data and text documents. The bibliographic information associated with the documents, for example, authors, publication dates and journal titles would be handled as normal secondary keys by the DBMS. Then, to provide effective retrieval using the content of the documents, various techniques such as document ranking would have to be provided by routines in the DBMS. Dattola (1979) describes such an implementation. It should be noted that this type of implementation does not improve the performance of a document retrieval system and many of the DBMS features such as subschema definition would not be used. It is, in fact, difficult to make these implementations efficient because of the large number of possible secondary keys in a document database. The main advantage would be to make document retrieval available in the same package as the DBMS.

6.3 Document retrieval systems and question-answering systems

A document retrieval system is entirely data-driven. That is, the information represented in the database is directly derived by statistical procedures from the documents in the system. In a question-answering system, the information is a formal representation of the knowledge about a subject and at present there is no method of automatically deriving this representation. This is the major reason that document retrieval systems can be used for databases of hundreds of thousands of documents representing many subjects. As new documents representing new subject areas come into the system, the information representing these new subject areas (new terms and relationships) will appear in the database. This is not true of question-answering systems and so they are currently limited to very specific subject areas (small knowledge representations).

There are some similarities between these systems. For example, both systems use lexical analysis of the user's input. The 'knowledge representation' in a document retrieval system shown in Fig. 4 does contain a notion of semantic vicinity as in semantic nets. In this simple network, documents and terms that are semantically similar will be closely connected. Also, a document retrieval system does use inference to decide which documents to retrieve — the relevance of documents is not explicitly represented in the database. Nevertheless, it cannot be denied that the two systems represent basically different approaches to information retrieval — one using statistical reasoning, the other symbolic reasoning.

It can be argued that the full range of AI techniques may not be needed for effective document retrieval. Some small experiments using syntactic procedures and more complex representations of the knowledge in documents have given disappointing results (Salton, 1968). However it is interesting to speculate as to what current AI techniques may be usefully applied to document retrieval systems.

Two of the important issues in artificial intelligence research are the representation of knowledge and the control strategies which govern how the system searches that knowledge. The same issues are important for document retrieval. The best method for representing the knowledge in a document retrieval system has long been a topic of debate, although not expressed in these terms. The network of Figure 4 seems to be a reasonable step towards providing a more powerful and flexible representation without adding the complexity of a semantic net. Having such a representation means that many different types of search strategies can be implemented. For example, the system may be able to choose between searching using a probabilistic model, searching using document clusters and searching using Boolean query formulations. A control strategy will then be needed to choose the appropriate strategy for a particular user and for a particular stage of the overall search. If the search strategies based on formal models have difficulty locating relevant documents in some cases, heuristic search techniques may be needed. An obvious strategy similar to backward chaining is to look at the closely connected terms and documents of a document identified as relevant. If no relevant document has been identified, the system may encourage the user to browse through the database to find interesting terms.

Another area of overlap is user models. A document retrieval system builds up a model of the user in terms of estimating which are the important terms in the relevant set through relevance feedback. It should be possible to get a better model by using a natural language interface to analyze the query and to interact with the user before the search is started. For example, this type of interface may help the

system decide whether the user is more interested in recall or precision and also what the important terms are. In one particular AI system (Rich, 1979), stereotypes of different types of users were provided to give the system extra information. For example, if a person identified themselves as a 'scientist' and 'feminist', the system would assume certain characteristics related to these categories during the search. These stereotypes may not be as useful in a document retrieval system because of the large population of users and the wide variety of information needs, even for one user over a period of time. However, they could provide some general information such as the differences between a physicist and a chemist in their views of a particular scientific database.

One further possibility is for the document retrieval system to retrieve documents and then, using a syntactic analysis of the text and a limited knowledge base, to answer simple questions about the content of the document. An example of this type of question is 'What are the main conclusions of this paper?' This question-answering capability would be more feasible in a limited application such as an office system for a company.

7. AN INTEGRATED INFORMATION SYSTEM

We have seen how the various types of information system relate to each other and also how these systems may, to some degree, be integrated. An integrated information system would combine techniques from the systems previously discussed and would therefore be able to handle a wider range of data than any one of these three systems. The following section is mainly speculative but it is intended to show what could be implemented using current techniques.

The basis for the integrated information system will be a database management system. The reason for this is that a DBMS provides a number of system features that any information system should have, as well as a means of formally defining the logical structure of the database. The main features that a DBMS provides are security of data, consistency of data, different views of the data for different users and a means of specifying file organizations for efficient retrieval. Another important feature that a DBMS would provide is the ability to run a distributed information system on a number of personal workstations connected by a local network.

The information database of the DBMS will contain a formal description of the data in the system. This formal description, although similar to a standard DBMS data model (such as the relational model), may contain extra semantic information to provide integrity constraints and perhaps a simple question-answering capability as mentioned in Section 6.1. It is unlikely at the present that a large-scale AI representation would be used in the database because of the difficulty in defining and maintaining these representations for anything but simple knowledge domains.

The integrated information system should have the capability of indexing and retrieving text documents. This capability would significantly extend the range of data that could be handled by the system without requiring the large time and storage overheads involved in many AI systems. The software incorporating the statistical analysis and retrieval routines for text documents could be implemented using the DBMS.

The main user interface to the integrated information system would be natural language. A natural language interface would enable casual users to communicate

more effectively with the computer, especially in the case of retrieving text documents. A more formal language would still be provided for specific tasks such as choosing output formats, defining databases and updating information.

The search/retrieve module should be capable of doing more than exact matching. In the case of text documents, the most effective search method would be to combine two or three alternative strategies based on probabilistic models and heuristics with a highly interactive dialogue. For more structured data, the system should be capable of deductive reasoning using the formal data definition.

The integrated information system then appears to be a distributed database management system with a natural language interface, some capability of doing deductive searches and routines for representing and retrieving text documents. The question is whether there is any incentive to build such a system. There are two types of information system currently available which may indeed provide the vehicles for an integrated information system. These will be discussed in the following sections.

7.1 Word processing systems

Office information systems consisting of work stations with powerful local processing capability connected to each other and to peripheral devices via a local network are currently being developed. As the processing power of these systems increases, it is inevitable that a DBMS will be incorporated into the network to handle the data management problems of the office. However, the major product of the word processing stations is text documents. Office memos, letters and reports are all being generated with these stations. Currently, there is no way of retrieving these documents apart from knowing exactly where they are stored. That is, there is no provision for retrieval by content. Document retrieval techniques such as automatic indexing are an obvious solution to this problem. Therefore, in the near future, office information systems may have distributed database management systems with document retrieval capability.

A major goal of the word processor manufacturers is to get the work stations into the managers' offices. In other words, they want to provide an information system for the managers and this has led them to investigate natural language interfaces. Therefore, the future office information system represents an excellent example of an integrated information system.

7.2 Home information systems

Home information systems, such as the British Post Office's PRESTEL, provide access to information for users in their homes. PRESTEL uses a specialized television receiver and a telephone connection to provide an interactive interface to the database which contains information on many subjects, from cooking recipes to reference material. Presently, users specify their interests by menu selection. That is, at each stage of the search the user is given a choice of paths to follow. The user often selects items of interest simply by scanning a large number of alternatives. The manufacturers of these systems are interested in developing home information systems which have more sophisticated interfaces and retrieval techniques. Text documents are obviously an important part of the database and so document retrieval techniques are applicable. A true home information system would provide

local processing and data storage as well as the access to general information. So, as in the case of the office information systems, the home information system will use integrated techniques.

8. CONCLUSION

The three main types of information system — database management systems, document retrieval systems and question-answering systems — have been presented in terms of a general descriptive model. The interfaces between these systems were then discussed.

The main points in the paper are as follows:

1. The three systems can be viewed as three approaches to the task of retrieving information, but there are sufficient differences in the type of information they deal with to justify their separate existences.
2. Advanced AI techniques from question-answering systems can be (and in some cases are) used to improve database management systems and document retrieval systems.
3. Document retrieval systems use sophisticated methods to represent and retrieve text documents and the techniques used in these systems may soon appear in office and home information systems.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the help of Dr. Jack Wileden and, in particular, Mohammad Dadashzadeh.

REFERENCES

Abraham, C. T., Ghosh, S. P. and Ray-Chaudhuri, D. K. (1968) File organization schemes based on finite geometries. *Information and Control 12*, 143–163.

Abramson, N. (1973) Another alternative for computer communications. *Fall Joint Computer Conference*, 695–702.

Adiba, M., Chupin, J. C., Demolombe, R., Gardarin, G. and Le Bihan, J. (1978) Issues in distributed database management systems: A technical overview. *Proceedings of the 4th Conference on Very Large Data Bases*, 89–110.

Bachman, C. W. and Daya, M. (1977) The role concept in data models. *Proceedings of the 3rd Conference on Very Large Data Bases*, 464–476.

Bentley, J. L. (1975) Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 509–517.

Berra, P. B. and Oliver, E. (1979) Associative array processor utilization in data base management. *Computer 12*, 53–61.

Bird, R. M., Tu, J. C. and Worthy, R. M. (1977) Associative/parallel processors for searching very large textual databases. *Proceedings of the 3rd Workshop on Computer Architecture for Non-Numeric Processing, ACM SIGMOD Newsletter 9*, 8–16.

Brodie, M. L. (1978) Specification and verification of database semantic integrity. University of Toronto (Technical Report CSRG-91).

Codd, E. F. (1979) Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems 4*, 397–434.

Croft, W. B. (1980) A model of cluster searching based on classification. *Information Systems 5*, 189–195.

Croft, W. B. and Harper, D. J. (1979) Using probabilistic models of document retrieval without relevance information. *Journal of Documentation 35*, 285–295.

Date, C. J. (1977) *An introduction to database systems.* 2nd Edition. Reading, MA: Addison-Wesley.

Dattola, R. M. (1979) FIRST — Flexible information retrieval system for text. *Journal of the American Society for Information Science 30*, 29–35.

Duda, R. O., Hart, P. E., Nilsson, N. J. and Sutherland, G. L. (1977) Semantic network representations in rule-based inference systems. *Pattern-directed Inference Systems* (D. A. Waterman and F. Hayes-Roth, eds.) New York: Academic Press.

Hammer, M. M. and MacLeod, D. J. (1975) Semantic integrity in a relational database system. *Proceedings of the 1st Conference on Very Large Data Bases*, 25–47.

Harper, D. J. and van Rijsbergen, C. J. (1978) An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation 34*, 189–216.

Harter, S. P. (1975) A probabilistic approach to automatic keyword indexing. Part I: On the distribution of specialty words in a technical literature. Part II: An algorithm for probabilistic indexing. *Journal of the American Society for Information Science 26*, 197–206, 280–289.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D. and Slocum, J. (1978) Developing a natural language interface to complex data. *ACM Transactions on Database Systems 3*, 105–147.

Hopcroft, J. E. and Ullman, J. D. (1979) *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley.

Kaplan, S. J. (1979) Cooperative responses from a portable natural language data base query system. University of Pennsylvania (Technical Report MS-CIS-79-26).

Kleinrock, L. (1975) *Queueing Systems*, Vol. 1, 2. New York: Wiley Interscience.

Knuth, D. E. (1973) *The Art of Computer Programming*, Vol. 3, *Sorting and Searching*, Reading, MA: Addison-Wesley.

Lockheed Information Systems. (1976) A brief guide to DIALOG searching. Palo Alto, California.

Martin, J. (1977) *Computer Data-Base Organization.* New Jersey: Prentice-Hall.

Mead Data Central, Inc. (1975) LEXIS: A primer. New York.

Metcalfe, R. M. and Boggs, D. R. (1976) Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM 19*, 395–404.

Minker, J. (1977) Information storage and retrieval — A survey and functional description. *SIGIR Forum 12*, 1–108.

Nilsson, N. J. (1980) *Principles of Artificial Intelligence.* Palo Alto, California: Tioga.

Ozkarahan, E. A., Schuster, S. A. and Smith, K. C. (1975) RAP — An associative processor for database management. *Proceedings of the AFIPS Conference 44*, 379–387.

Randall, D., Buchanan, B. and Shortliffe, E. (1977) Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence 8*, 15–46.

Rich, E. (1979) Building and exploiting user models. Carnegie-Mellon University (Technical Report CMU-CS-79-119).

Rivest, R. L. (1976) Partial match retrieval algorithms. *SIAM J. Computing 5*, 19–50.

Robertson, S. E. (1977) The probability ranking principle in IR. *Journal of Documentation 33*, 294–304.

Robertson, S. E. and Sparck Jones, K. (1976) Relevance weighting of search terms. *Journal of the American Society for Information Science 27*, 129–146.

Salton, G. (1968) *Automatic information organization and retrieval.* New York: McGraw-Hill.

Schubert, L. K., Goebel, R. G. and Cercone, N. J. (1979) The structure and organization of a semantic net for comprehension and inference. *Associative Networks* (N. V. Findler, ed.) New York: Academic Press.

Sparck Jones, K. (1974) Automatic indexing. *Journal of Documentation 30*, 393–432.

Su, S. Y. W. and Lipovski, G. J. (1975) CASSM: A cellular system for very large data bases.

Proceedings of the 1st Conference on Very Large Data Bases, 456–472.
Ullman, J. D. (1980) *Principles of database systems.* Maryland: Computer Science Press.
van Rijsbergen, C. J. (1977) A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation 33*, 106–119.
van Rijsbergen, C. J. (1979) *Information Retrieval.* London: Butterworths.
Waltz, D. L. (1978) An English language question-answering system for a large relational database. *Communications of the ACM 21*, 526–539.
Wilkes, M. V. and Wheeler, D. J. (1979) The Cambridge digital communication ring. *Proceedings of the Local Area Communications Network Symposium.* Boston, MA.
Wilkes, M. V. and Needham, R. M. (1980) The Cambridge model distributed system. *Operating Systems Review 14*, 21–29.
Wong, E. and Chiang, T. C. (1971) Canonical structure in attribute based file organization. *Communications of the ACM 14*, 593–597.
Wong, H. K. T. and Mylopoulus, J. (1977) Two views of data semantics: A survey of data models in artificial intelligence and database management. *INFOR 15*, 344–383.
Woods, W. A. (1970) Transition network grammars for natural language analysis. *Communications of the ACM 13*, 591–606.
Yang, C. S. (1978) A class of hybrid list file organizations. *Information Systems 3*, 49–58.
Yao, S. B. (1979) Optimization of query evaluation algorithms. *ACM Transactions on Database Management Systems 4*, 133–155.
Yu, C. T. and Salton, G. (1975) Precision weighting — An effective automatic indexing method. *Journal of the ACM 23*, 76–88.
Zloof, M. M. (1975) Query by example. *Proceedings of the National Computer Conference 44*, 431–438.