THE ROLE OF DATABASE MANAGEMENT SYSTEMS FOR INVESTIGATIVE DATA

Gary D. Anderson, McMaster University

I. INTRODUCTION:

In this paper, we will attempt to present enough information about database management systems (DBMS) to help a researcher or user of investigative data assess the applicability of a DBMS to his/her setting. Database management philosophy promotes several major themes (1).

A. Standardization of data definition and documentation.
B. Improvement of physical access to data items.
C. Reduction of data redundancy.
D. Independence between the way data is stored and the application program which accesses the data.
E. Simplification of writing application programs.
F. Reduction in the need for program maintenance.
G. Consolidation of updating, retrieval, reporting and utility functions.

This paper is not a definitive coverage of database management or database management systems. It is, rather, a brief introduction to the three database models and commercially available systems which implement these models in various ways. The reader is encouraged to consult the references cited at the end of the paper for a more thorough coverage of the subject. In particular, the series of articles (2) - (6) in Computing Surveys provides an especially useful overview with a complete set of further references.

In the following section, we will first consider the three main data models (hierarchical, network and relational). The second section will then consider the primary features of a database management system. This will be followed by a section outlining the specific advantages and strengths of the database management system approach. We will then provide a partial list of existing systems classified by the type of primary data access method used and data model they adhere to. The next section will highlight considerations of DBMS to be kept in mind for research and investigative data users who wish to interface their DBMS usage with the SAS system. The final section will review the major features of SIR (8), a DBMS with which the author has worked. SIR was designed specifically for research and investigative data users requiring an interface directly to SAS and other statistical systems.

II THE THREE DATA MODELS

All major database management systems in use today incorporate some variation or combination of three primary data models: the hierarchical model, the network model, or the relational model.

A. Hierarchies

The hierarchical model is described by an inverted tree structure among the data record types. Consider the following record types in a school database:

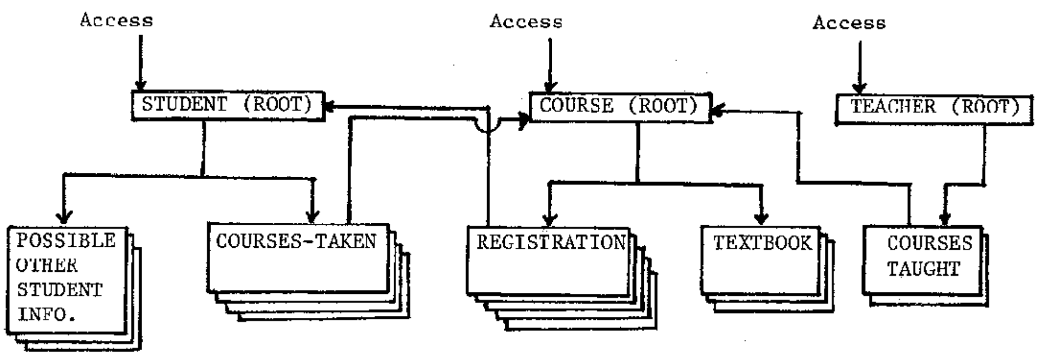
STUDENT (STUDENT-ID, NAME, BIRTHDAY, SEX)
COURSES-TAKEN (STUDENT-ID, COURSE-ID, YEAR)
COURSE (COURSE-ID, COURSE-NAME, PRE-REQ, LEVEL, CREDITS)
REGISTRATION (COURSE-ID, STUDENT-ID, YEAR)
TEXTBOOK (COURSE-ID, TEXT-ID, TITLE, LOCATION, COST)
TEACHER (TEACHER-ID, NAME, RANK, BIRTHDAY, SEX)
COURSES-TAUGHT (TEACHER-ID, COURSE-ID, YEAR)

The double underlined variables in these record types denote the primary identifier for each record type occurrence. The single underlined variable denotes a secondary identifier necessary to establish relationships among the record types and also to uniquely identify an individual record within its primary identifier.

Diagram 1 illustrates how this data is represented in a hierarchical model.

The STUDENT record type forms a root segment which owns the COURSES-TAKEN records (as well as any other record types necessary to describe an individual student). The COURSE root record owns the REGISTRATION records and the TEXTBOOK records for the course. Finally, the TEACHER root record owns the COURSES-TAUGHT records. Thus, this database is represented by three hierarchies. Navigation among these hierarchical structures is accomplished by use of the secondary identifiers as shown by the arrows.

DIAGRAM 1:



For example, the COURSES-TAKEN record type has the COURSE-ID as one of its variables. In a given DBMS, this access from the COURSES-TAKEN record will either be accomplished by a physical pointer directly from the COURSES-TAKEN record to the appropriate COURSE root record or by use of a separate index reference to the COURSE record. Although implementation of the hierarchical model lends itself to both physical pointers embedded in the data records or index files separate from the records, most hierarchical systems use the latter approach.

B. Networks

The network model is a more general model. In fact, it can be argued that the hierarchical model is a special case of the network model. The popularity of the hierarchical model, however, relates to its greater simplicity and the fact that so many natural phenomena are more easily and efficiently modelled by it. In fact any data relationship able to be represented by a network model can also be represented by a hierarchical model. The only added cost will (possibly) be some redundancy in the data.

Diagram 2 illustrates a network representation of this same school database.

In the network model, access paths are defined when the data records are described to the DBMS. It is then the responsibility of the system to maintain these paths by appropriate use of index files, linked lists, rings, etc., as the data records are added to the database. Systems which implement a network structure by definition require a wealth of access path methodologies to be available. They generally require greater cost in updating and maintenance of the database but result in greater efficiency when retrieving through the paths (sets) defined. Such systems best lend themselves to relatively static designs since the cost of modifying the design, and thus changing the structure of the database, will be extremely high.

It will be noticed that both the hierarchical and network models require that care be given in the database design stage to defining the required navigational paths. Also, in evaluating a DBMS, it is wise to determine the degree of difficulty involved in adding new paths to an existing database.

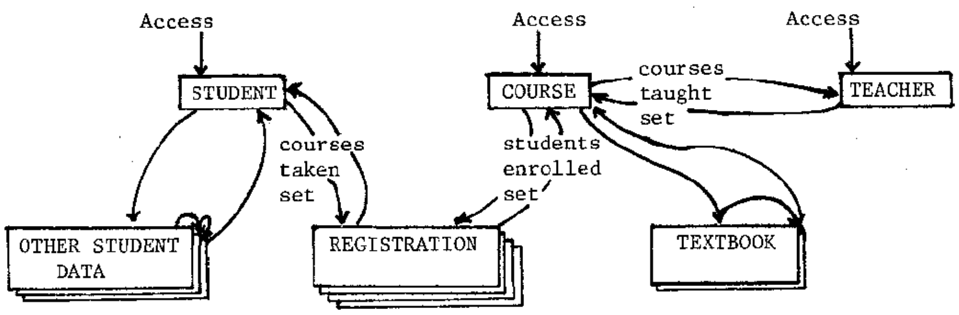
The physical arrangement of the data records for these two models is generally markedly different. The hierarchical model lends itself to an indexed sequential storage approach with the root record and the records it "owns" being located physically together in a variable length record file. This lends itself to very efficient retrieval of information which is hierarchically related (e.g. all information about a specific student or group of students). The network model usually requires that each record type be physically separate (generally in physically separate files). Access to information in related record types therefore, by definition, requires keyed or pointer access into separate files.

In comparing the two model diagrams of the school database, we will observe that the network model did not require the COURSES-TAKEN or the COURSES-TAUGHT record types. Both of these relationships are defined to the network DBMS at data definition time and are automatically maintained by the system. These two record types can thus be considered as redundant data. Whether or not this redundancy actually results in greater physical data storage requirements, however, depends upon the relative requirements of keys and pointers in the network model versus the record types and their keys in the network model. The actual storage requirements may be very similar.

C. Relations

The relational model is growing in popularity and promises to be the approach of the future. As an SAS user, the reader will be familiar with the approach of the relational model as SAS uses a relationally oriented approach to management of data sets. In the relational model, each of the school record types would be considered as a separate relation (data table). In a purely relational model DBMS, the relational algebra would be applied to these relations at the time of each request to project out courses taught by a given teacher or student registered for a given course for example. In practice, present DBMS implementations of the relational model generally combine the use of some type of key structure to the relations. This should not be considered necessarily bad (a point hotly disputed by relational purists) because the combination can lead to both relational simplicity and indexed efficiency. It should be noted that the COURSES-TAKEN relation is not required for an implementation

DIAGRAM 2:



of the relational model as the REGISTRATION relation is sufficient to establish any desired relationships between the STUDENT relation and the COURSE relation. The COURSES-TAUGHT relation may be necessary if more than one teacher is involved with the same course (a many to many or plex relationship).

III. MAJOR DBMS FEATURES

Having considered the three primary data models, let us look briefly at the major features of a DBMS.

A. Data Description Language (DDL)

The data description language or schema definition capability provides the means of mapping the particular conceptual model to be implemented including all data items, record types and their relationships, into the DBMS. Once the description has been provided, the DBMS then supports the necessary structures (through indexes, pointers, sets, etc.) automatically. The vocabulary of the DDL and the way in which it models identical information will vary considerably from one DBMS to another. The Conference on Data Systems Languages (CODASYL) has made a considerable effort during the 1970's to encourage standardization in data description language area of commercial DBMS systems. Many major DBMS developments in the 1970's adhered, at least to some degree, to the CODASYL standards. Recent interest in the relational model which is not encompassed in the CODASYL proposal, is leading to major departures from this as a standard.

Regardless of how a particular DBMS allows a user to define structures to it, the most important thing is the appropriateness of the structures to be defined. The user of a DBMS must give careful thought to the design of a database and how it should be modelled before attempting to define it to the system. The data description language is intended to provide great flexibility in implementing structures it is told to implement. It has no way of telling the user whether these are appropriate structures for his problem.

The DDL capabilities of commercial DBMS systems vary significantly in the flexibility they provide for data documentation. The major emphasis is toward commercial data requirements. Thus, the investigative data user will generally find a number of things he needs are missing or poorly supported. Among these are:

1. extended variable labelling capabilities;
2. lack of categorical data support (e.g. value labels);
3. no concept of multiple missing values; and
4. little support for scientific data types.

Further, the orientation toward the COBOL language of most DDL implementations does not appeal strongly to investigative data users who more likely have a FORTRAN orientation.

B. Data Storage Support

A DBMS will contain, internally, methods for storing data records, with necessary physical pointers, indexes, etc., to support the structures defined in the data description language. A major advantage of a DBMS is the fact that it assumes this responsibility and frees the database developer from this system task.

C. Data Manipulation Language (DML)

Various methods are provided by a DBMS to support the users access to the database both for adding and changing data and for retrieving data. Most systems rely on a host language interface approach. This assumes that most database applications (data entry, updating, retrievals, reports, etc.) will be written in the host language (e.g. COBOL or FORTRAN). The DBMS merely provides a series of special calling routines from the host language that allow the programs to read or write records or data items in the database.

Although the host language interface allows the greatest degree of flexibility in application development using a DBMS, it requires a very high level of programming support. Consequently most DBMS systems provide additional data manipulation language support including:

1. Built-in high level retrieval languages. This may range from a simple-to-use query command language to a high level procedural language.
2. The provision of built-in procedures for listing, tabling and report generating.
3. Direct interfaces to other systems (like SAS).

D. Concurrent User Support

Because a database is often considered as a data storage bank for an organization, it will generally be necessary to allow multiple users to access it at the same time.

A DBMS usually addresses the problem of multiple write access by some type of record or file locking. Allowing multiple users to read a database concurrently is relatively simple. Proper management of multiple writers, however, is a much more significant problem. The main difficulties center around maintaining the integrity of the data and avoiding deadlocks among users.

E. Database Security

Again, with multiple users accessing the same set of data, it becomes important to provide flexibility in the type of security mechanisms supported. Approaches available are:

1. Subschema capability. This restricts groups of users to specified parts of the overall database schema. Their programs and procedures can access only the data in their subschema.
2. Multiple levels of password protection.
3. Encoding of the data so that it is readable only if passed through appropriate decoding routines.

F. Database Integrity Support

A DBMS system will provide various mechanisms to insure that both the data in the database and the physical structuring of the data is protected at all times. These include:

1. Transaction logging or journalling. This implies that a copy of every modification to the database is stored in a separate file. This file will contain all information sufficient to restore the database should it be destroyed or to back out transactions which did not fully complete or were incorrectly done.
2. Checkpointing. Essentially this provides points in a long process from which processing could restart in case of application failure.
3. Data integrity checking facilities. This includes such things as data validity checks at data entry (range, valid values and consistency checks) and utilities to check indexes and data paths within the database for validity.

F. General Operational Utility Support

Database management systems provide a series of build-in utilities designed to support maintenance operations on the database. Among these are:

1. loading and unloading utilities;
2. restructuring utilities for reorganizing the database to allow for new record types, record modifications or defining new access paths;
3. a database statistics utility;
4. utilities for subsetting the database in various ways; separating out specified record types, disengaging specified parts of the database from the rest, etc.;
5. utilities for rejoining subsetted parts or for combining multiple databases; and
6. utilities for tracing problems in the database (e.g. bad indexes or pointers) and assisting in recovery from such problems.

IV. ADVANTAGES OF THE DBMS APPROACH

There are a number of inherent advantages to the DBMS approach. The value of each of these to a given user community will depend very much upon the specific circumstances of that community. It is realized that an advantage to one community of users may actually be considered a disadvantage to another. We will list some of the more commonly cited advantages and let the reader judge their importance to his setting:

A. Reduction in data redundancy

A DBMS implies central storage of data with multiple users and applications accessing this central database. This has the advantage that updating is restricted to one set of data rather than being required on multiple sets in various locations. This central storage of data in a database does not mean all analysis has to take place in the database itself. For example, database retrievals can, at regular intervals in time, produce SAS data sets directly from the database for local analysis by specific local user groups.

B. A greater level of data independence

Because all access to data in a database is controlled through the DML, programs accessing the database are not usually affected by database changes. The DML effectively masks the users application program from the physical structure of records and their organization in the database.

C. Significant reduction in data sorting and unnecessary record processing

Because records are addressable by key or pointer directly, access to a specific record or a well-defined set of records (e.g. an enrolment report for a given student) can be accomplished with minimal processing. The reader should be aware, however, that access to summary information over which an access path was not defined in the data description language may be extremely costly in some DBMS's.

D. Online access to data

A DBMS will usually support an interactive interface through which direct display and modification of data can be achieved.

E. Machine independence

Most major systems are implemented across multiple mainframe computers. Thus, if care is taken in the host language programs to ensure their portability, a database can be moved relatively easily from one type of mainframe to another.

F. Standardization of methods

A DBMS provides a standard approach to handling databases within a user community. This leads to better documentation, more flexible utilization of personnel and overall greater efficiency in use of resources.

V. A CLASSIFICATION OF SOME EXISTING SYSTEMS

In considering a few of the presently most widely used commercial systems, we will classify them by the type of model they implement (hierarchical, network or relational) and by the type of physical access method they use. Access methods are primarily of two types: physically linked DBMS and inverted DBMS. In the physically linked systems, the addresses of related records are stored as part records themselves so that linkage paths exist within and among the records. The DBMS then follows these paths to access records. The inverted DBMS depends upon separate index files containing one set of indexes for each data item on which access is desired. Pointers are then maintained from the indexes back to the original data records. The inverted DBMS usually carries out operations on the index files prior to any direct data access.

The following DBMS systems were chosen merely as representatives of the various categories. A comprehensive review of available systems is well beyond the scope of this paper.

A. Physically linked DBMS

1. TOTAL is a DBMS package offered by Cincom

Systems Incorporated. It is one of the oldest major DBMS systems and is presently the most widely used DBMS with over 3000 user sites. TOTAL first became available in 1969 and is now available on more than a dozen separate mainframes including several of the mini computers. On IBM systems, TOTAL is presently the most serious competitor for IBM's own IMS/VS system.

TOTAL implements a restricted network model. Its implementation is somewhat like the CODASYL model, although it is not a CODASYL system. Restrictions in TOTAL include its inability to express hierarchies of more than two levels, lack of support for variable length records and limited flexibility in indexing methods. TOTAL is a very efficient system in use of resources (perhaps as a result of its restrictions) and thus has been able to be run on very small machines.

TOTAL supports data independence to the individual item level since program access to the database is by item not record.

2. IMS/VS is IBM Corporation's major DBMS offering and is the second most widely used IBM compatible DBMS with probably over 1500 installations. IMS/VS is a hierarchical system. Data in IMS is structured into a network-like structure. These network relationships are remodeled, by the use of a pseudo database description (DBD), into new logical views that appear hierarchical to the user.

   Data independence in IMS is to the record level but not to the individual item since all database access is by record (segment).

3. IDMS offered by the Cullinane Corporation is a very attractive DBMS package for IBM hardware with a fast growing user community (about 500 sites). Data structures under IDMS follows the CODASYL specifications for networks.

   Cullinane promotes a total data management concept, of which IDMS is one component. Included in the total system, in addition to IDMS, is teleprocessing support, an on-line query facility, a data dictionary and various report generation facilities. This modular concept in DBMS offerings is becoming increasingly popular among vendors.

B. Inverted DBMS

1. SYSTEM 2000 is one of the best known DBMS available with versions implemented on UNIVAC and CDC as well as on IBM systems. System 2000 is a product of the INTEL Corporation. There are presently over 700 installations using SYSTEM 2000. In addition to host language support for COBOL, FORTRAN and PL/1, SYSTEM 2000 allows a user to interface with the database directly through a high level language consisting of English-like commands. SYSTEM 2000 has a highly developed multi-threaded feature, which allows multiple application access to the same database with full control of concurrent updating.

   SYSTEM 2000, unlike many other inverted DBMS, implements a hierarchical structure. The system allows up to 32 levels in a given tree. The trees are related by tables called hierarchical location tables. Although powerful, SYSTEM 2000 does not achieve the data networking possible in other DBMS's.

2. ADABAS is a full-scale DBMS produced by the German-based corporation Software AG. Over 550 sites presently use this DBMS world-wide. It features a very high level of data independence and general separation between logical and physical concerns. Overall, ADABAS is one of the foremost representatives of the class of inverted DBMS.

3. INQUIRE is marketed by Infodata, Inc., has been marketed since 1969, and more recent releases have greatly enhanced its DBMS offerings. There are probably about 200 sites presently using INQUIRE. Data independence to the item level is supported but the system features no subschema capability. One of INQUIRE's strong features is a comprehensive text handling capability making it useful for document databases.

4. MODEL 204 is marketed by Computer Corporation of America. There are probably about 100 installations of MODEL 204. A major feature of MODEL 204 is a completely self-contained language oriented for easy interactive inquiry which may also be used for updating and for batch processing. The system has given notable attention to search capabilities with a powerful set of relational and numeric operators. It also features some integrated mathematical operations, a feature left to the host language interface in most DBMS's.

C. Relational DBMS

It is difficult to say too much about available relational DBMS. First, there are no full DBMS implementations of the rational model presently commercially available on IBM mainframe hardware. Some have said that the about-to-be-delivered SQL/DS is a relational DBMS. SQL/DS, IBM's structured Query Language, is scheduled for delivery in February 1982. Although it does not appear to be a full relational DBMS in its initial release, it is relational and seems to be IBM's first major, announced step in the direction of a relational DBMS.

Relational DBMS systems are making their strongest showing on smaller computers. INGRES, probably the first major, commercially available relational DBMS is available on DEC 11 equipment including the VAX. INGRES is a product of Relational Technology Inc. Two other commercial DBMS offerings of the relational model have been announced within the last 18 months and should be mentioned: ORACLE, a product of Relational Software Inc. was announced in June, 1980 and RELATE/3000 by Computer Resources Inc. in December, 1980. RELATE/3000 was designed exclusively for the Hewlett Packard 3000 computer.

VI. DBMS CONSIDERATIONS UNIQUE TO THE INVESTIGATIVE AND RESEARCH DATA USER

The database management systems we have described so far are primarily designed for a commercial database environment. The research database management environment is often different in a number of respects. Since we assume many of the readers of this paper are in this latter category, let us now consider some of the ways in which these two environments do differ:

A. Availability of resources

The commercial database administrator usually has a major corporate commitment behind him. He can thus justify sufficient computer, software and personnel resources to meet the task of developing a major database application. The research designer is often more limited. His data requirements may be large and complex but his user base is not sufficient to justify a major commitment of resources. He must share the computer with others; he must be able to access affordable DBMS software; and he must work with limited personnel support.

B. Programming support

In the commercial environment, the database administrator is trained in database design and has a staff of applications programmers skilled in writing host language programs in COBOL, PL/1 or FORTRAN to interface with a host language driven DBMS. The research user is generally not trained in database design and must use support staff with minimal programming background. Consequently, the DBMS must support internal language capabilities which are both user friendly, and sufficiently powerful to meet the full scope of retrieval needs.

C. Design flexibility

Although the ability to alter the database is important in both the commercial and research environments, it is especially important to the researcher. Almost by definition, his original design will change (to add new or overlooked data items, to accommodate new or altered sources of data, etc.). The DBMS system he uses must provide utilities for adding new record types, for adding or deleting items from an existing record type and for rearranging data within a record type.

D. Concurrent usage

The need for concurrent updating capability is usually less critical in the research environment than the commercial field. The researcher generally has a small group of staff with one person being responsible for database updating activities. The nature of research studies are also often such that data updating lends itself to an occasional volume operation rather than an interactive database update through a query system. Such volume operations can easily be scheduled at times when they will not interfer with retrieval and reporting operations. Concurrency may indeed be important for some research database activities, but it is unlikely to involve high volume activity sufficient to warrant multi-threaded retrieval access to the database.

E. Research data support

As we have mentioned earlier in this paper, the typical commercial DBMS will lack many data support characteristics needed by the researcher. These include categorical data support, multiple missing value handling and comprehensive arithmetic computational support. Also, none of the major commercial DBMS have made provisions for direct interfaces to statistical systems such as SAS.

VII. A DBMS DESIGNED FOR THE RESEARCH AND INVESTIGATIVE DATA ENVIRONMENT

Scientific Information Retrieval (SIR) is a DBMS marketed by SIR Inc. SIR presently has an installed base of about 180 sites. This system was designed specifically for the research environment in an effort to make available to the research community the required aspects of a DBMS system while adding features specific to the research environment. The major features of the SIR/DBMS are as follows:

A. support for a full hierarchical data model with multiple hierarchies able to be connected into network structures;

B. the provision of an SPSS-like data definition (schema) language;

C. built-in data editing including range, value and consistency checking;

D. a wide range of volume data input and update procedures;

E. multiple level data security to the item and record levels;

F. a comprehensive retrieval and updating language with a structured syntax and full arithmetic and logic operations;

G. several built-in descriptive statistical procedures, the ability to write retrievals directly to system files for SAS, BMDP, and SPSS, and the ability to write retrievals directly to formatted record files;

H. a report generator with sorting and multiple, nested breakpoints;

I. the provision of an interactive subsystem which contains a text-editor, storage for user written procedures and an interactive retrieval processor;

J. a CALL (macro) facility which allows parameters to be passed to user written procedures;

K. automatic journal file logging of database modifications; and

L. an interface to SIR/FORMS, a screen-oriented data entry, display and quality control system.

SIR/DBMS was developed using SIR/TRAN (9), a macro-preprocessor program that has made it possible to convert SIR/DBMS, relatively quickly to run on a variety of computer systems (i.e. IBM, CDC, UNIVAC, DEC VAX, PRIME, PERKIN ELMER and SEIMENS). Presently, SIR/DBMS is available on IBM under OS (interactive usage requires TSO) and CMS.

VIII. CONCLUSION

This paper has discussed aspects of DBM's from the prospective of the model they implement (hierarchical, network or relational), their potential advantages, some example systems and aspects of their use in research and investigative data situations. DBMS usage is growing rapidly. Users of SAS will increasingly find themselves involved with a DBMS in the future. This involvement will likely be in one of two settings.

The first of these settings is rather an indirect involvement. This is the setting in which the DBMS is a source of data sets which are subsequently analyzed in SAS but where the SAS user has little or no other direct involvement with the database. This situation will be common where SAS is used as a statistical and graphical complement to the corporate or institution-wide database.

The second setting involves a more intimate and direct involvement with the DBMS on the part of the SAS user. This is the situation in which a research DBMS, such as SIR, is used as the database management tool in a study in which SAS is the primary analysis system. Here the SAS user is likely to have major involvement in using the DBMS as well as SAS.

In both of these settings, a good, easy-to-use interface to SAS is extremely important. SIR, a system design for use in the investigative setting, recognized the need for such an interface and consequently made provisions for it. Commercial DBMS package designers have yet to recognize this need. It is undoubtedly in the best interest of the SAS user community to nake strong representation to the commercial SBMS vendors regarding this need.

REFERENCES

1. Ross, R.G. Data Base Systems - Design, Implementation, and Management, AMACOM, 135 West 50th St., New York, NY 10020, 1978.

2. Sibley, E.H. "The Development of Data-Base Technology", Computing Surveys 8 (1) Mar 1976 1-5.

3. Fry, J. and Sibley, E.H. "Evolution of Database Management Systems", Computing Surveys 8 (1) Mar 1976, 7-42.

4. Chamberlin, D. "Relational Database Management Systems", Computing Surveys, 8 (1) Mar 1976, 43-66.

5. Taylor, R. and Frank, R.L. "CODASYL Database Management Systems" Computing Surveys 8 (1) Mar 1976, 67-103.

6. Tsichritzis, D.D. and Lochovsky, F.H. "Hierarchical Database Management: A Survey", Computing Surveys, 8 (1) Mar 1976, 105-123.

7. Michaels, A.S., Mittman, B. and Carlson, C.R. "A Comparison of the Relational and CODASYL Approaches to Database Management" Computing Surveys, 8 (1) Mar 1976, 125-151.

8. Robinson, B.N. et al, SIR - Scientific Information Retrieval Users Manual, SIR Inc., P.O. Box 1404, Evanston, IL 60204, 1980.

9. Gazdzik, W., Karpel, L.C., Miller, A.H., Robinson, B.N. and Cohen, E. "SIR/TRAN - A Macro-Preprocessor for Software Portability and Maintenance", COMPSTAT 1980 Proceedings.