Fourth generation data management systems

by KEVIN M. WHITNEY
General Motors Research Laboratories
Warren, Michigan

INTRODUCTION AND HISTORY

Many hundreds of programming systems have been developed in recent years to aid programmers in the management of large amounts of data. Some trends in the development of these data management systems are followed in this paper and combined with ideas now being studied to predict the architecture of the next generation of data management systems. The evolution of data management facilities can be grouped into several generations with fuzzy boundaries. Generation zero was the era when each programmer wrote all his own input, output, and data manipulation facilities. A new generation of facilities occurred with the use of standard access methods and standard input/output conversion routines for all programs at an installation or on a particular computer system. The second generation of data management was the development of file manipulation and report writing systems such as RPG, EASYTRIEVE, and MARK IV. Much more comprehensive facilities for the creation, updating, and accessing of large structures of files are included in the third generation of generalized data management systems such as IMS/2, IDS, and the CODASYL specifications. Each of these generations of data management systems marked great increases in system flexibility, generality, modularity, and usability. Before speculating on the future of data management, let us survey this history in more detail.

Standard access methods, such as ISAM, BDAM and SAM which formed the first generation data management facilities were mainly incorporated in programming languages and merely gathered some of the commonly performed data manipulation facilities into standard packages for more convenient use. The main benefit of these standard facilities was to relieve each application programmer of the burden of recoding common tasks. This standardization also reduced program dependency on actual stored data structures and formats. Although these access methods and input/output conversion routines were often convenient to use, they could only accommodate a limited number of different data structures and data types.

As computer systems became more common, more widely used, more highly depended on, and eventually more essential to the functioning of many businesses, a second generation of data management facilities became available to speed the job of generating reports and writing new application packages. These report generators, file maintenance packages, and inquiry systems provided easy-to-use facilities for selecting, summarizing, sorting, editing, and reporting data from files with a variety of formats. A user language simple enough for non-programmers to use was sometimes provided, although using more advanced system facilities often required some programming aptitude. This second generation of data management systems brought non-procedural user languages and a greater degree of program independence from data formats and types.

Much more general capabilities were incorporated into third generation systems such as IDS, IMS, and the CODASYL report specifications. All of these systems emphasize the effective management of large amounts of data rather than the manipulation or retrieval of data items. All have facilities to manage data organized in much more complicated data structures than the sequential structures used by earlier systems. Relationships among data items in many different files are expressible and manipulable in the data management systems. Provisions for the recovery from many different types of system errors, head crashes, erroneous updates, etc. are integral parts of these systems. Audit trails of modifications of the data base are often automatically maintained, and new file descriptions are handled by standard system facilities. In general, many of the functions which previous generations of data management systems left to the operating system or to user application programs are automatically incorporated in these systems. Many of the third generation systems provide a convenient user inquiry language, a well-defined interface for user application programs, and new language facilities to aid the data administrator in the description and maintenance of the data base.

A fourth generation data management system will continue the development of all these trends toward generality, flexibility, and modularity. Other improvements will result from theories and concepts now being tried in experimental systems. Much greater degrees of data independence (from user program changes, from data description and storage changes, from new relationships among the data) will be common; user languages will
become much less procedural; and data manipulation facilities for use in writing application programs will become simpler and more powerful. Concepts from set theory and relation theory will become more widely used as the advantages of a sound theoretical basis for information systems become more widely appreciated. Increasingly, information management systems will make more of the optimization decisions relating to file organization and compromises between different user requirements. The trend to bending the computer toward user requirements rather than bending the user to the requirements of the computer will continue resulting in progressively easier to use systems.

One organization of the facilities of an information management system may be the division into modules shown in Figure 1. This modularity may represent software or hardware (or both) boundaries and interfaces. A data access and control module is needed to manage data flow to and from the storage media. This data management module is used by the data description and manipulation module in providing data description and manipulation at a level less dependent on the storage structure of this data. The user can access data through application processors or through a system provided query language processor. A variety of system services are grouped into the user services module and the data base administrator services module. User services include such conveniences as on-line manuals, help and explain commands, and command audit trails. Data base administrator services include facilities to load and dump data bases, to perform restructuring of data and storage organizations, to monitoring performance, and to control checkpointing and recovery from errors.

Impacting the fourth generation information management systems are the theories and methodologies of data description and manipulation, the relational view of information, the establishment of sound theoretical foundations for information systems, and the development of networks of cooperating processors. Each of these topics will be discussed in one of the following sections. Following those sections is a description of our experiences with RDMS, a relational data management system which illustrates some of these new theories and methodologies.

**DATA DESCRIPTION AND MANIPULATION**

Certainly the most complex and difficult problem facing the designers, implementers, and users of an information management system is the selection of language facilities for the description and manipulation of data. Although many attempts have been made to separate data description from data manipulation, it must be noted that data description and data manipulation are inextricably intertwined. While the declarative statements which describe a data base may indeed be kept separate from the statements in application programs which manipulate the data in the data base, nevertheless the data description facilities available determine and are determined by the data manipulation facilities available. Descriptive statements for vectors aren't very useful without vector manipulation statements, and vice versa.

The description of data may be done at a wide variety of levels of generality ranging from simple statements about the relationships between large sets of data items to explicit details about the actual storage of the data items. Information management systems of the next generation will have data description facilities at a variety of levels to serve different classes of users. At least three main levels of data description can be distinguished, the information structure for the users, the data structure for the data base administrators, and the storage structure for the system implementers and the system.

The information structure which determines the user's view of the data is (ideally) quite abstract, indicating the relationships among various types of data items, but omitting details such as data item types, precisions, field lengths, encodings, or storage locations. The user should also be free of system performance considerations such as indexing schemes or file organizations for efficient retrieval of the data items, particularly since his access requirements may conflict with those of other users. In any event the system or the data administrator is in a better position than any one user to make those arbitrations. An example of the information structure for some data described by a relational model is shown in Figure 2.

A second level of data description is necessary to specify the structure (logical) of the data which will facilitate the efficient retrieval of the data representing the information in the data base. This second level of description is the data structure and represents additional informa-

```
PEOPLE (NAME, AGE, HEIGHT, WEIGHT)
SHIRTS (SHIRT#, COLOR, SIZE, COLLAR)
SLACKS (SLACKS#, COLOR, SIZE, FABRIC)
OWNS-SHIRTS (NAME, SHIRT#)
OWNS-SLACKS (NAME, SLACKS#)
```

Figure 2—An information structure specified by a relational model
tion about the patterns of retrieval expected for information in the data base. The set occurrence selection facility of the CODASYL proposal is an example of this level of data description. CODASYL schema data description statements for the information structure of Figure 2 are shown in Figure 3.

A still more detailed level of data description is the storage structure of the data which represents the actual organization of the data as stored in the system's storage media. At the storage structure level, items of concern include field types, lengths, encodings, relationship pointers and index organizations. Figure 4 shows a diagram of pointers and storage blocks specifying a storage structure for the data described in Figure 3.

Data manipulation facilities must also exist at a variety of levels corresponding to the data description levels. As data description becomes more general, the corresponding data manipulation becomes less procedural and more descriptive. E. F. Codd's relational model of data described in the next section provides an example of a highly non-procedural data manipulation language.

A RELATIONAL MODEL FOR INFORMATION

Information management systems deal fundamentally with things such as people, ages, weights, colors, and sizes which are represented in a computer by integers, floating point numbers, and character strings. A collection of items of similar type is called a domain. Domains may overlap, as for example with the domains of people and of children.

SET IS PEOPLE; OWNER IS SYSTEM.
MEMBER IS PERSON; DUPLICATES NOT ALLOWED FOR NAME.

RECORD IS PERSON.
NAME TYPE CHARACTER 12.
AGE TYPE FIXED; CHECK RANGE 5, 50.
HEIGHT TYPE FIXED.
WEIGHT TYPE FIXED.
#SHIRTS TYPE FIXED.
#SLACKS OCCURS #SHIRTS.
2 COLOR TYPE BIT 4; ENCODING CRTable.
2 SIZE TYPE FIXED.
2 COLLAR TYPE FIXED.
SHIRTS OCCURS #SHIRTS.
2 COLOR TYPE BIT 4; ENCODING CRTable.
2 SIZE TYPE FIXED.
2 FABRIC TYPE FIXED.
SLACKS OCCURS #SLACKS.
2 COLOR TYPE BIT 4; ENCODING CRTable.
2 SIZE TYPE FIXED.
2 FABRIC TYPE FIXED.

Figure 3—A data structure specified in the CODASYL data description language

A relationship is an association among one or more not necessarily distinct domains. "Is married to" is a relationship associating the domains of men and of women. An occurrence of a relationship is an ordered collection of items, one from each domain of the relationship, which satisfy the relationship. Each occurrence of a relationship associating N domains is an ordered collection (John and Mary is an occurrence of the relationship "is married to" if John is married to Mary).

A relation is a set of some tuples satisfying a relationship. The number of domains of a relation is called its degree and the number of tuples of a relation is called its cardinality.

This simple structure is adequate to represent a great variety of information structures. Certain data manipulation facilities arise naturally from the relational description of information. The basic operations are the traditional operations of set theory (union, intersection, difference, etc.) and some new operations on sets of relations (projection, join, selection, etc.). Projection reduces a relation to a subset of its domains, while join creates a relation by combining two component relations which have one or more common domains. Selection extracts the subset of the tuples of a relation which satisfy some restriction on the values of items in a particular domain. In combination with the usual control and editing commands, these operations provide a convenient and non-procedural data manipulation language. Note that nothing need be known about the data or storage structures of the information represented by a relational model. The user may concern himself entirely with the domains of interesting objects and their relationships.

Figure 4—A storage structure specified by a pointer diagram
THEORETICAL FOUNDATIONS FOR INFORMATION SYSTEMS

As information systems become increasingly complicated, it will be more important to base their designs on sound theoretical foundations. Although it has always been customary to test a system as comprehensively as possible, larger systems can never completely be tested. Thus it is important to find other methods of verifying that a system will function as specified. Current work in proving program correctness has this same aim with regard to programs. In this section three contributions to the theoretical foundations of information system structure will illustrate some possible effects of new theory on system design. D. L. Childs\(^1\) has devised an ordering for set structures useful in the implementation of set operations. E. F. Codd\(^2\) has developed the relational calculus as a sound basis for user languages, and W. T. Hardgrave\(^3\) has proposed a method for eliminating ambiguous responses to queries of hierarchical data structures.

Childs proposed a general set theoretic data structure based on a recursive definition of sets and relations. His theory guarantees that it is always possible to assign unique key values to any set element in such a structure. These key values may be used to create an ordered representation of the data structure. Set operations are much more efficient on ordered sets than on unordered sets. Thus the theory leads to efficient implementations of complex structures.

In a series of papers, E. F. Codd has developed the relational model of data which was explained in the previous section. One important theoretical result of this theory is a proof that any relational information in a data base of relations can be retrieved by a sequence of the basic relational and set operations defined in the previous section of this paper. Furthermore, it is possible to estimate the system resources necessary to answer the query without answering it. Thus a user can be warned if he asks a particularly difficult query. Although not all queries on a data base of relations can be answered as relations, the inclusion of functions on relations (counts, sums, averages, etc.) guarantee a very wide range of legal queries. This theory assures a system design that a general purpose system will not suddenly fail when someone asks an unexpected new query.

Research into storage structures underlying the relational data model by Date \& Hopewell\(^4\) show that a variety of possible anomalies in the storage, updating, and retrieval of information do not arise when the information is stored in Codd’s canonical third normal form. Thus by using a relational storage structure for data, certain types of consistency are automatically assured. These studies show also a variety of efficient methods of implementing a relational data model.

A third investigation into the fundamentals of information systems design is W. T. Hardgrave’s study of information retrieval from tree structured data bases with Boolean logical query languages. Hardgrave analyzed anomalies resulting from the use of the “not” qualifier in boolean queries. Finding unavoidable problems with the usual set theoretic retrieval methods, he formulated additional tree operations which separate the selection of data items from the qualification of items for presentation. These capabilities may be considered a generalization of the HAS clause of the TDMS system. This study not only focuses attention on possible multiple interpretations for some Boolean requests applied to items at different levels of a tree hierarchy, but also presents more severe warnings about possible problems with the interpretation of network structured data such as in the CODASYL proposal.

NETWORKS OF COOPERATING PROCESSORS

Continuing decreases in the cost of data processing and storage equipment and increases in the cost of data manipulation software will bring further changes in the architecture of information management systems. The example of Figure 5 shows the natural trend from software modularity toward hardware modularity.

![Figure 5](From the collection of the Computer History Museum (www.computerhistory.org))
ware modularity of third generation systems to hardware modularity of fourth generation systems.

This trend toward hardware modularity has been under way for many years. Control Data Corporation has advocated the use of independent peripheral processors to handle some of the more mundane functions of computing systems such as input/output spooling, paging, disk and drum interfaces, etc. The use of front end processors such as the IBM 3705 to handle communications functions independently of the central processor is already common. IBM uses the Integrated Storage Controller, a small independent processor, to control 3330 disk drives. Special purpose computer systems for Fourier transforms and for matrix manipulation are being used as peripheral or attached processors in current computing systems.

Two specific examples of independent processors in data management systems are intelligent remote inquiry terminals and the data base computer. While intelligent remote terminals are already common, independent data base computers are still in research laboratories. These data base machines consist of an independent control processor and a large storage media. The processor not only manages the control commands necessary to handle the storage media, but also can perform logical extraction or selection of data records to reduce the amount of data transmitted to the host processor. As more experience with independent data base computers is accumulated, they will assume additional tasks such as selecting data compaction and compression methods for optimal data storage and selecting indexing methods for optimal data access.

RDMS, A RELATIONAL DATA MANAGEMENT SYSTEM

To gain some experience with the advantages and limitations of data management using a relational model of information, the RDMS system was designed and implemented in PL/I on a large virtual memory computer system. The system consists of three main sections, a command (query and data manipulation) language interpreter, set and relational data manipulation facilities, and an interface to the operating system. This modular design resulted in a readily portable group of set and relation manipulation facilities with rigidly defined interfaces to the user programs and to the operating system. Sets are used internally by the system in self-describing data structure which catalogs each user's sets and their characteristics.

Because one of the main benefits of a relational model of information is its simplicity, care was taken to keep the data description and manipulation languages simple. All data managed by RDMS is organized in relation sets viewed by the user only as named collections of named domains. The user need not concern himself with the type of a domain, nor its precision, nor its storage representations. All data manipulation facilities store their output in sets which may be manipulated by other RDMS commands. The command format is simple and consistent containing an output set name (if the command produces an output set), a keyword identifying the command, and the parameters of that command. These parameters may be set names, domain names, domain values, and character strings to label output displays. For example, a command which extracts a subset of a set is: "CHEAP_ WIDGET _ SET = SUBSET OF WIDGET _ SET WHERE COST LT 100.00". Commands are typed on a keyboard, and the system responds with a full screen graphic display.

RDMS commands may be grouped in four main classes: Set control commands which manipulate entire sets (create, destroy, save, catalog, uncatalog, universe, etc.); Display commands which display or print the contents of sets (list, graph, histogram, print, etc.); Set manipulation commands which specify, modify, analyze, select, and combine the contents of sets (union, intersection, subset, join, statistics, summary, set, domains, etc.); and Special purpose commands which provide miscellaneous facilities for system maintenance, bulk input and output, and assorted user conveniences (explain, command list and trace, read from, describe, etc.).

Several small data analysis and manipulation applications were tried using RDMS to test its adequacy for flexible data manipulation. A medical records analysis was particularly informative because the problems to be solved were specified by persons with neither programming nor data base experience. We were given thirty data items of eight different data types for each of several hundred mother-child pairs and asked to find any effects of a particular medication. The large amount of information displayed on the graphic console and the power of individual commands were demonstrated by answering an initial set of 41 questions with only 35 commands. Some of the more pleasing features of the system are the following. Combining individual commands into very complex requests is greatly facilitated by maintaining all data in sets used both for inputs and outputs of commands. Histograms and graphs of data from sets may either be displayed on the graphic terminal or printed on the printer. A permanent record of all commands, any screen display, and the contents of any set can be printed. Mistakes can often be undone by the REMEMBER and FORGET commands which provide an explicit check-pointing facility for each set. The main complaints from users were the paucity of data types available, the difficulty of editing erroneous or missing data items, and the inability to distinguish missing data by a special null code.

We feel the RDMS system was a successful and useful experiment from the viewpoints of both the system user and the system implementer. Sets of relations manipulated by powerful descriptive commands are usable in real information systems. Non-programmers can readily adapt to the relational model of data and the corresponding types of data manipulation commands. For the system implementer, the relational data structure provides a
simple and consistent collection of data description and
manipulation facilities with which to build a variety of
information systems.

SUMMARY

Data management has evolved through at least three dis­
tinct stages and is entering a fourth, from access methods,
to file management systems, to data management sys­
tems, and toward information systems. In this evolution,
the user's facilities for dealing with large amounts of
information have become more general, flexible, extensi­
ble, and modular. Gradually he has been unburdened of
various tedious levels of detail allowing him to focus his
attention more directly on the relationships among var­i­
ous types of information and their manipulation. These
trends will continue, spurred on by advances in informa­
tion system design theory and in computing system hard­
ware and software.

ACKNOWLEDGMENTS

The author is indebted to E. F. Codd, C. P. Date, G. G.
Dodd, and A. Metaxides for many long hours of discus­
sion on the subjects in this paper.

REFERENCES

1. CODASYL Data Base Task Group April 1971 Report. Available
from ACM.
2. Codd, E. F., "A Relational Model of Data for Large Shared Data
cedings of IFIP, 1968.
4. Codd, E. F., "Relational Completeness of Data Base Sublan­
guage," Proceedings of Courant Institute Symposium on Data Base
6. Hardgrave, W. T., "BOLTS: A Retrieval Language for Tree Struc­
tured Data Base Systems," Proceedings of the Fourth Interna­
tional Conference on Information Systems (COINS-72), December
1972.
(RDMS)," Proceedings of the Fourth International Conference on
Information Systems (COINS-72), December 1972.