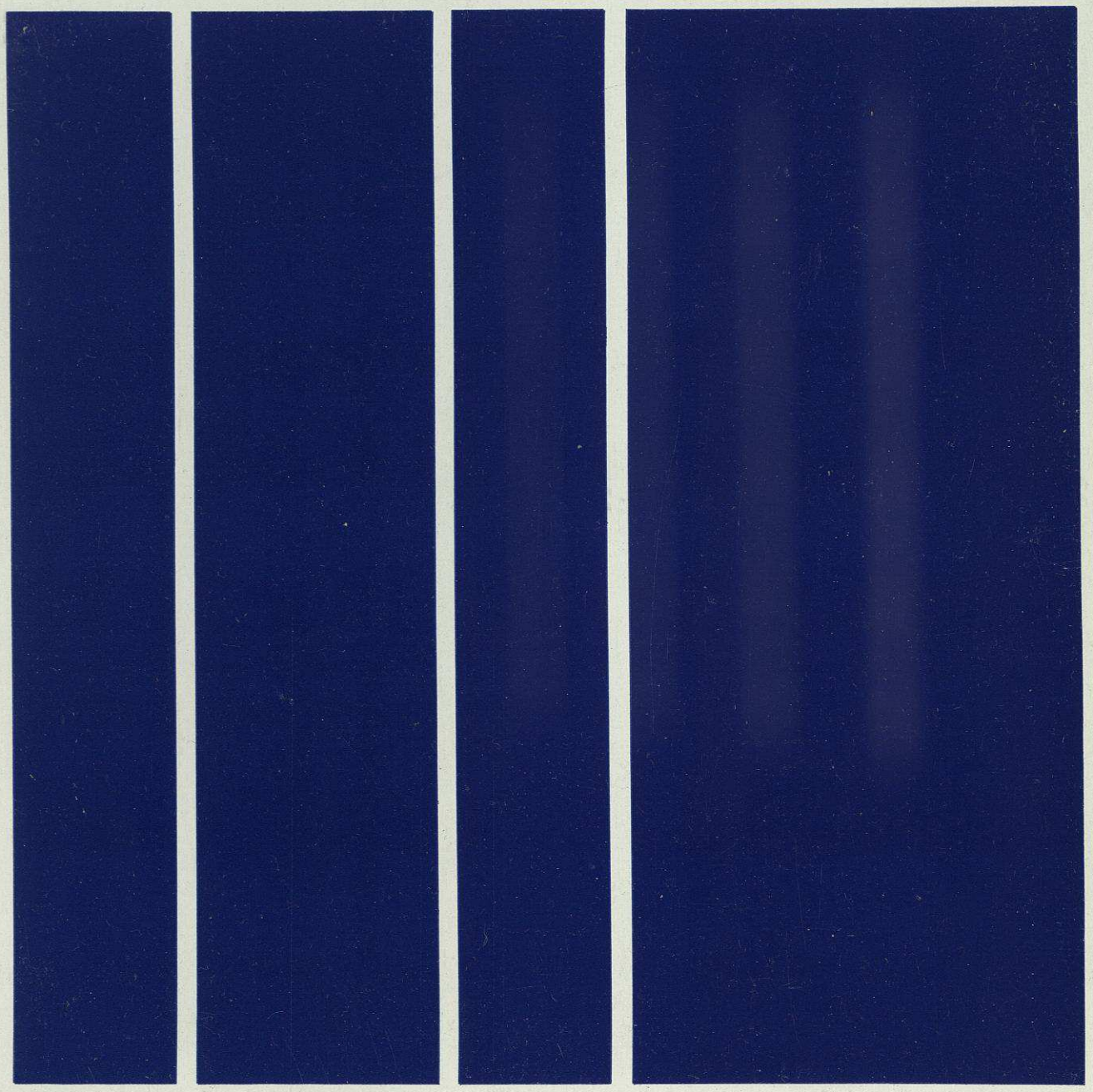
Report Series
No12

Trends in Database Management Systems

June 1979



The Butler Cox Foundation

Abstract

Report Series
No12

Trends in Database Management Systems

by David Flint
June 1979

Trends in business data processing such as increasing user requirements and the convergence of technologies require the integration of data processing systems. Integration may be between operational and management information systems as well as across applications areas.

Integration requires a new approach to data processing — the database approach. Data must be seen as a corporate resource that should be understood and managed in its own right. Database technology is already used successfully by many businesses but it is most effective when used to support the database approach.

This report discusses the concepts underlying database management systems and explains their significance and likely impact on data management. The report also examines relevant trends in hardware and software and discusses the changing market in data management products.

Networking and standards are also discussed insofar as they relate to data management.

The report concludes that the market is a rapidly evolving one and that it offers important opportunities for the management services department to improve the service provided to the enterprise. The report recommends the steps that management services should take to exploit these opportunities.

The Butler Cox Foundation is a research group which examines major developments in its field — computers, telecommunications, and office automation — on behalf of subscribing members. It provides a set of 'eyes and ears' on the world for the systems departments of some of Europe's largest concerns.

The Foundation collects its information in Europe and the US, where it has offices through its associated company. It transmits its findings to members in three main ways:

- As regular *written reports,* giving detailed findings and substantiating evidence.

- Through *management conferences,* stressing the policy implications of the subjects studied for management services directors and their senior colleagues.

- Through *professional and technical seminars,* where the members' own specialist managers and technicians can meet with the Foundation research teams to review their findings in depth.

The Foundation is controlled by a Management Board upon which the members are represented. Its responsibilities include the selection of topics for research, and approval of the Foundation's annual report and accounts, showing how the subscribed research funds have been employed.

Abstract

Report Series
No12

Trends in Database Management Systems

by David Flint
June 1979

Trends in business data processing such as increasing user requirements and the convergence of technologies require the integration of data processing systems. Integration may be between operational and management information systems as well as across applications areas.

Integration requires a new approach to data processing — the database approach. Data must be seen as a corporate resource that should be understood and managed in its own right. Database technology is already used successfully by many businesses but it is most effective when used to support the database approach.

This report discusses the concepts underlying database management systems and explains their significance and likely impact on data management. The report also examines relevant trends in hardware and software and discusses the changing market in data management products.

Networking and standards are also discussed insofar as they relate to data management.

The report concludes that the market is a rapidly evolving one and that it offers important opportunities for the management services department to improve the service provided to the enterprise. The report recommends the steps that management services should take to exploit these opportunities.

The Butler Cox Foundation is a research group which examines major developments in its field – computers, telecommunications, and office automation – on behalf of subscribing members. It provides a set of 'eyes and ears' on the world for the systems departments of some of Europe's largest concerns.

The Foundation collects its information in Europe and the US, where it has offices through its associated company. It transmits its findings to members in three main ways:

- As regular *written reports,* giving detailed findings and substantiating evidence.

- Through *management conferences,* stressing the policy implications of the subjects studied for management services directors and their senior colleagues.

- Through *professional and technical seminars,* where the members' own specialist managers and technicians can meet with the Foundation research teams to review their findings in depth.

The Foundation is controlled by a Management Board upon which the members are represented. Its responsibilities include the selection of topics for research, and approval of the Foundation's annual report and accounts, showing how the subscribed research funds have been employed.

Report Series No.12

TRENDS IN DATABASE MANAGEMENT SYSTEMS

by David Flint

June 1979

Butler Cox & Partners Limited Morley House 26 Holborn Viaduct London EC1A 2BP

TABLE OF CONTENTS

I INTRODUCTION 1

A The Purpose of this Report 1
B The Scope of the Subject 2

II THE PRESENT SITUATION IN DATA MANAGEMENT 3

A The Experience of Users of Database Management Systems 3
B Database Products 6
C Database Applications 7

III THE COMPONENTS OF A DATABASE ENVIRONMENT 8

A Organisational Components 8
B Hardware and Software Components 10

IV IMPORTANT CONCEPTS IN DATA MANAGEMENT 11

A Data Independence 11
B Data Models 16
C Data Analysis 21
D Database Integrity 23
E Automatic Design and Tuning 25

V RELEVANT TRENDS IN HARDWARE 28

A The Cost of Processors and Memory 28
B Associative Memories 31
C Novel Database Processors 32

VI DEVELOPMENTS IN SOFTWARE 34

A Database Managers 34
B Data Dictionaries and Design Aids 38
C Query Languages 40
D System Building Tools 44
E Performance Monitoring and Tuning 46

VII DATABASE AND NETWORKING 49

A Remote Access to Data 49
B On-Line Interrogation and Problem Solving 50
C Distributed Database 52

VIII INDUSTRY STANDARDS 56

A An American Standard for Database Management Services 56
B Other Standards 57
C The Implications of the Database Management Services Standard 57

IX FINDINGS AND RECOMMENDATIONS 58

A Findings 60
B Recommendations 61
C Footnote 62

APPENDIX A LIST OF DATA MANAGEMENT PRODUCTS MENTIONED IN THE TEXT 62

SELECTED BIBLIOGRAPHY 65

I. INTRODUCTION

Most large organisations have now implemented a range of computer-based systems which assist the business at an operational level. The applications of order processing, sales ledger and bill of materials are common examples. Such organisations are now seeing new opportunities to exploit computers to produce benefits for the organisation. Many of these opportunities depend upon the integration of information from different systems that up to now have been quite separate. The opportunities encompass systems at an operational level and also systems that provide support for managers.

Integration is often achieved by giving several applications systems access to a single, unified database. Often, this database will be maintained by a database management system (DBMS). DBMS are now used extensively. They are estimated to be in use in well over 5,000 installations worldwide, and they are being used successfully. A survey conducted by the Foundation whilst this report was being prepared illustrated this quite conclusively.

The main results of this survey are reported in section II.A. Despite the satisfactory experience of DBMS users, however, management services managers remain confused and apprehensive.

This is not surprising. New approaches to data management — for example, the relational approach (discussed later) — are receiving publicity. In addition, the development of databases to date has reinforced the trend towards centralisation and so it is not clear how DBMS should be combined with the more recent trend towards distributed processing.

The relationship between DBMS and the search for new system development methods is also not obvious.

Despite these difficulties there is a clear need for an integrated approach to corporate data and for a DBMS to support this approach. The reasons for this are reviewed briefly in section II.C.

A The Purpose of This Report

The report, having identified the need for DBMS and having shown that experience so far has been good, concentrates on the future trends in the field.

The main purposes of the report are:

1. To explain the concepts that are basic to effective data management. Understanding these concepts is necessary if the significance of future development is to be appreciated. These concepts are discussed in sections III and IV.

2. To explain the expected trends in hardware, software and computer networking which we believe to be relevant to DBMS in the next five years. These trends are explained in sections V, VI and VII, respectively.

3. To consider the prospects for standardisation and the implications for management services management. This is the subject of section VIII.

4. To explain the principal findings of our research and to provide management services management with advice based upon our findings.

The report does not address the subject of the functional integration of data management and office systems. Nor does it consider in detail the interaction between DBMS and distribued processing.

B The Scope of the Subject

Database management is concerned with the effective management of the information acquired by, and generated within, an organisation. It is a wider issue than the selection and use of either a software or a hardware product. It embraces the following concepts:

- Data as a corporate resource.
- Data administration as a function independent of particular project teams.
- The provision and use of tools in order to implement the concept (although the use of a DBMS is not essential to the database approach).
- The revision of systems analysis techniques to embrace the analysis of data.

When we discuss the characteristics of DBMS products in this report we often quote particular products as examples. We have done this either to enable us to be specific or to establish that some capability is within the state of the art. Since this report is not intended as a buyer's guide to DBMS we have not attempted to list all the products that show each characteristic. Some of the products will almost certainly be unfamiliar to some readers and so we have listed all the products mentioned, together with their suppliers, in the appendix.

II. THE PRESENT SITUATION IN DATA MANAGEMENT

A The Experience of Users of Database Management Systems

As a first step towards understanding the current position, we surveyed DBMS last year in order to tap the informed opinion of those managers who have experience with databases. We have published the results separately as "The Experience of Users of Database Management Systems".

One hundred users replied. They represented enterprises from a wide variety of business sectors in both the public and the private sectors both in the UK and continental Europe. Between them they have had over two hundred years of database experience and they have implemented systems in a wide range of applications.

Respondents were asked why they had adopted a DBMS and the reasons they gave, as shown in figure 1, showed that the needs to increase productivity and to improve service to users were

Figure 1 Expectations of Users

You must have had reasons for adopting a DBMS at all. Please read the list of possible benefits and grade them according to the influence they had on your original decision.

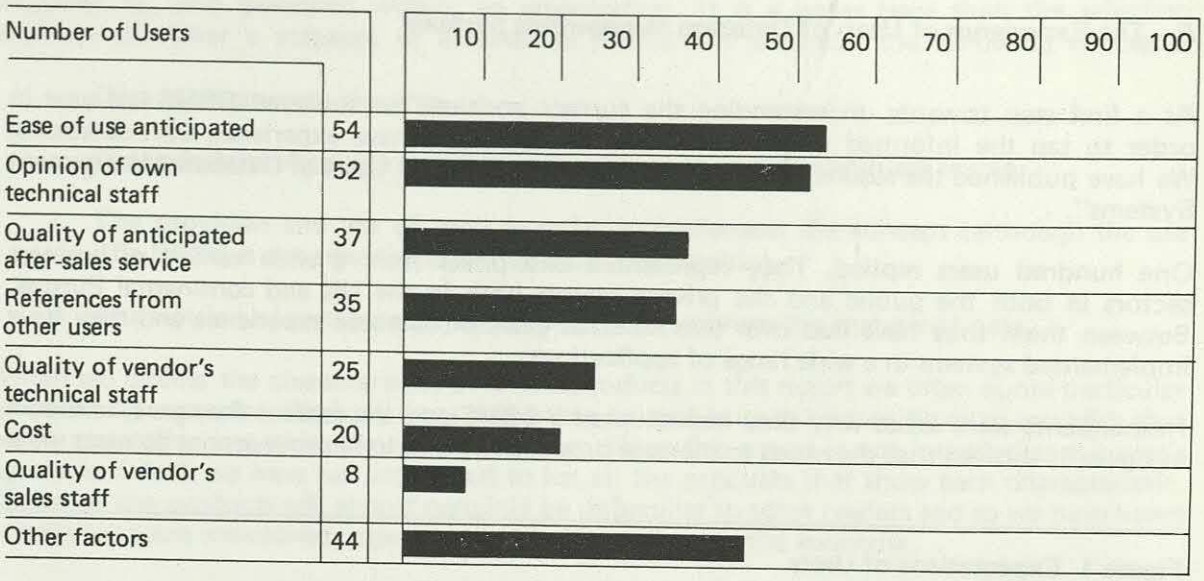
	Number of Users reporting an influence that was:				
	Very Strong	Strong	Weak	Negligible	*No Answer*
Reduced programming effort in development	17	45	27	8	*3*
Reduced programming effort in maintenance	31	36	24	7	*2*
Reduced data duplication	29	35	19	13	*4*
Better data consistency	27	46	13	11	*3*
Faster response to new user requirements	28	51	15	3	*3*

All the suggested benefits were felt to be important by most respondents and no one benefit was of overwhelming importance. The number of respondents reporting reduced programming effort in development as a very strong influence was significantly less than that for the other items.

factors that had influenced their choice of a particular DBMS. Their answers, shown in figure 2, confirmed the conclusion reached above. The small number of respondents who mentioned cost suggests that improving service to users was the major motive in the database decision.

Figure 2 DBMS Selection

Which factors most influenced your choice in favour of the DBMS you actually chose? Pick just three from the list below.



The 'other factors' included 15 reasons why the user had no choice of DBMS and a wide variety of virtues ascribed to the DBMS, eg efficiency, reliability, ease of use.

Respondents were asked to comment also on their general experience of database and on their experience against their expectations. The results are shown in figures 3 and 4 respectively. The fact that 90% of respondents were at least pleased with their experience is a considerable vote of confidence in DBMS and one that we had not expected. In some ways, however, the muted nature of the six negative reactions is just as significant. This general result is confirmed by the comparison of experience with expectations in figure 4. The preponderance of 'better' reports is striking as is the small number of 'worse' reports. Further analysis has shown that all the 'much worse' and many of the 'worse' reports referred to benefits that the respondents had not thought to be important when they were choosing the DBMS. Thus, managers had unrealistically optimistic views only on points that were of limited significance.

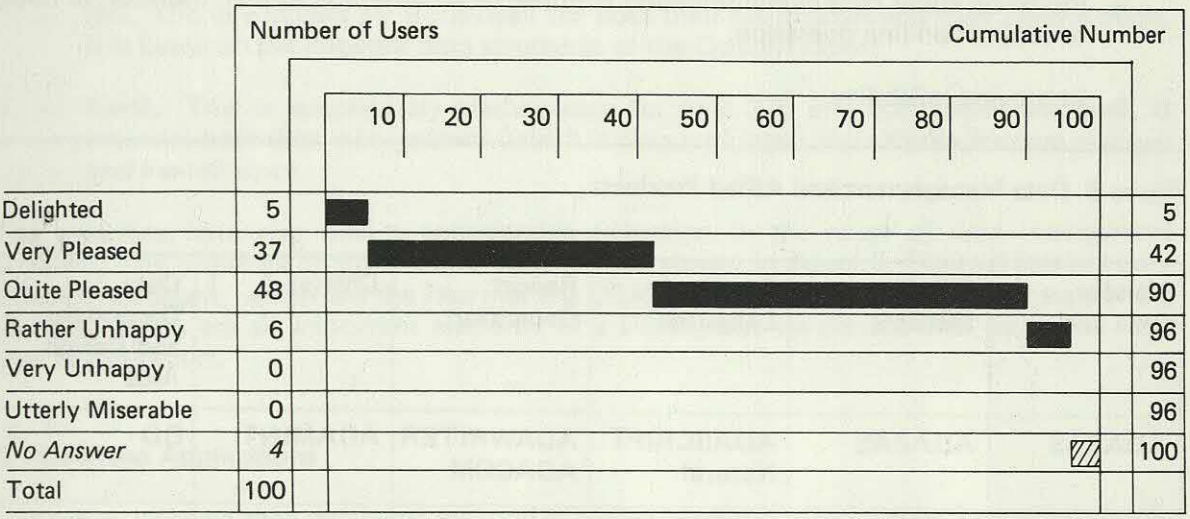
Because only around 110 users responded the results of the survey must be treated with caution. Nonetheless the very clear results and the general consistency of those results with the results of other research allow two major conclusions to be drawn from them:

1. The use of database management systems has been a success. All the products on which we received reports (except some older and more limited offerings) are well regarded by their users, and brand loyalty is high.

2. Most users had expected to achieve significant benefits from their DBMS and most had achieved even greater benefits than they had originally expected. The benefits achieved included: reduced programming effort through greater ease of use, less duplication of data, greater consistency of data, and faster response to new user requirements.

Figure 3 User Satisfaction

How do you feel about your experience with your DBMS?



	Number of Users	Cumulative Number
Delighted	5	5
Very Pleased	37	42
Quite Pleased	48	90
Rather Unhappy	6	96
Very Unhappy	0	96
Utterly Miserable	0	96
No Answer	*4*	100
Total	100	

Figure 4 Users' Experiences

What benefits have you actually achieved? Please read the list of benefits and grade them according to the degree of improvement or deterioration you have witnessed.

	Number of Users claiming a benefit				
Benefit	Much Better than expected	Better than expected	Worse than expected	Much Worse than expected	*No Answer*
Reduced programming effort in development	18	44	15	1	*22*
Reduced programming effort in maintenance	13	43	9	1	*34*
Reduced data duplication	18	57	3	0	*22*
Better data consistency	18	54	1	0	*27*
Faster response to new user requirements	17	38	17	1	*27*

B Database Products

The first DBMSs were embedded in systems for specific application areas. In the late sixties, with the release of IMS, IDS, and so on, DBMSs emerged as products in their own right.

During the seventies the vendors have increased the power and useability of their products by:

- Providing support for extra file access methods.
- Adding extra sophistication in describing data structures.
- Providing either data communication features, or an interface to a TP monitor, or both, to support on-line operation.
- Improving reliability.

Figure 5 Data Management and Allied Products

Vendor	Database Manager	Query Language	Report Generator	DMI(s) 1	Data Dictionary and Design Aids
ADABAS	ADABAS	ADASCRIPT Natural	ADAWRITER ADACOM	ADAMINT	DD 3
Cullinane	IDMS	OLQ	CULPRIT	DML	IDD
IBM (370)	IMS	IQF	GIS	DL/1	DD, DBDA
TANDEM	ENSCRIBE	ENFORM		CALLs 2	
ICL (2900)	IDMS	DATA DISPLAY	no	DML	DDS
CINCOM	TIS TOTAL	T–ASK	SOCRATES	CALLs	DD
Informatics	MARK IV	MARK IV			
Hewlett-Packard (3000)	IMAGE	QUERY		CALLs	
Honeywell (Level 66 and below)	IDS II	QRP	PLP	DML	DD
Honeywell (Level 68)	MRDS	LINUS		CALLs	DD

1 DMI – Codasyl data manipulatic language
2 CALLS– Procedure calls, no special features in the language
3 DD – Data Dictionary

The main DBMSs are thus now limited more by the quality of their basic concepts than by any deficiencies in their implementation.

Some well-known current DBMSs are:

- IMS. This is IBM's leading database product. It is based on hierarchies of records cross-linked by pointers.

- Total. This is supplied by an independent software firm, Cincom, and is probably the world's biggest selling DBMS. It is based on a two-level hierarchy. Total is available on a wide variety of computers including several minicomputers.

- IDS. This is supplied by Honeywell for both their mainframes and their Level 6 minis. It is based on the network data structures of the Codasyl report.

- Ramis. This is supplied by Mathematica for IBM 370 and compatible machines. It provides both data management (which is discussed later) and utilities for data retrieval and maintenance.

The seventies have also seen a considerable extension in the range of data management products. The offerings of some major suppliers are shown in figure 5. Figure 5 lists not only database managers, which are the heart of the DBMS, but also a number of other supporting routines. These are all important elements of a DBMS, and we discuss their respective roles later in this report.

C Database Applications

Our survey showed that databases are used in a wide variety of applications, including payroll, engineering, finance, order processing and distribution. In many cases, however, the systems are integrated to only a limited extent with the other systems in an enterprise.

A general trend towards the integration of systems across departmental and functional boundaries can be observed in systems development. This starts with the grouping of single functions into applications systems, a process that is normally achieved using conventional techniques. It continues with the planning of 'families' of related applications, a process that requires the resolution of various anomalies in the representation and definition of data items. A DBMS may be introduced at this stage, and a fair proportion of major enterprises have now reached this stage. Beyond this, there is the possibility of a system that meets the complete information needs of the business. Few companies have so far attempted this since it certainly requires a DBMS and it also involves high costs and significant risks.

A second trend runs in parallel with the first. It is the increasing significance of management information, rather than operational data processing, as an objective of systems development. In the past, management information systems have often been limited and have been quite separate from operational systems. Management's needs for accurate, timely, and consistent information are fuelling the development of systems in which routine processing and management information are combined.

Database contributes to the development of combined systems by providing the raw data from which management reports must be constructed in a clean and well defined form which is free of the presuppositions of particular processing.

Major trends in the development of data processing now clearly point towards the adoption of the database approach to systems development.

III. THE COMPONENTS OF A DATABASE ENVIRONMENT

A Organisational Components

The complex, and usually unfamiliar, nature of data management software will usually require some specialisation amongst those programmers who provide technical support for software. Because one database will typically support a number of application systems the differing requirements of these systems must be reconciled in a single database design. It is now usual to meet these requirements by appointing a database administrator (DBA) within the data processing department who has the following functions:

- Providing technical support for the database, the DBMS, and the data dictionary.
- Monitoring changes in database technology.
- Evaluating and selecting software.
- Choosing the file sizes, placement and access method to support the database.
- Defining procedures for the physical security of the database.
- Consulting on database design and the development of standards.
- Loading, reorganising, and restructuring the database.

There will normally be only one DBA or one DBA team in a company, though where a company has a federal structure there may be one in each constituent company.

In a large enterprise a variety of ways may exist of coding and describing the products, customers, etc. with which the enterprise is concerned. To contain the problems which this causes (and these problems are especially obvious if systems are to be integrated through a database) there is often a data standardisation function. Since this function runs across line departments, but is not directly concerned with computing, it will often be located in a non-DP part of the management services department.

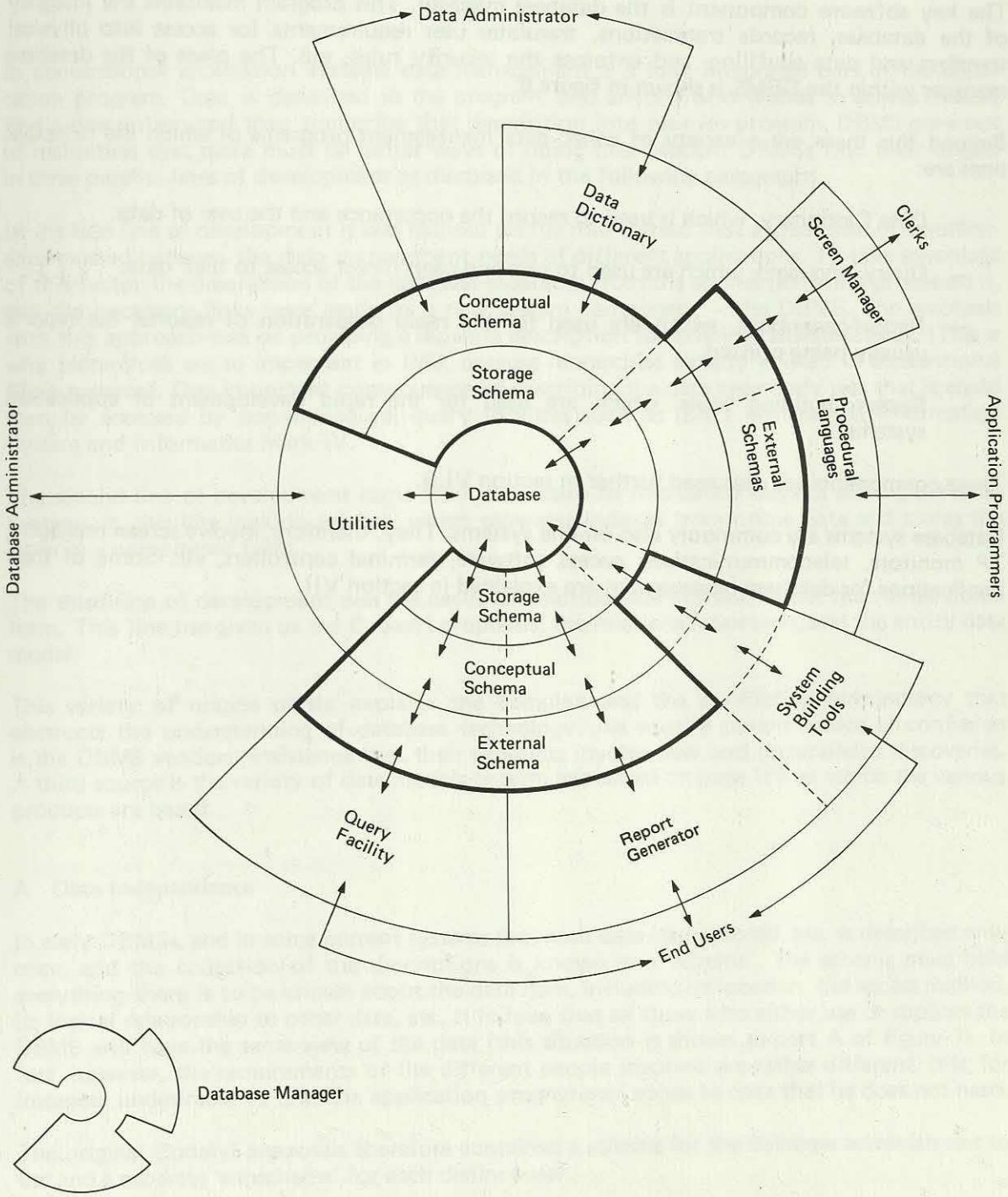
In constructing a database or, indeed, in linking application systems by the transfer of files, various incompatibilities of data definition, accuracy, currency, and format are likely to be encountered.

Sometimes these incompatibilities can be avoided by standardisation prior to the adoption of a database or they may be resolved by purely technical means during the construction of the database. In other cases, the departments for whom the systems are being provided may be unwilling to accept such a resolution and then it will be necessary for someone at a senior level in the company to break the deadlock. This function, which may or may not be a separate post in the corporate hierarchy, is called data administration.

Data administration is, like data standardisation, not basically a DP function, and so it may be

combined with data standardisation. It will often be the task of the data administrator's staff to maintain records of both the occurrence and the usage of data items in the business. Such records may cover non-database and, indeed, non-DP systems, and they will often be held in a computerised data dictionary.

Figure 6 DBMS Components (schematic)



B Hardware and Software Components

At present, databases are usually held on standard disc units, attached to a general-purpose computer, though drums and main store also find application.

In the future, mass stores and specialised file controllers are likely to come into use. There may also be data processors whose only function is to support the database. These possibilities are discussed further in section V.B.

The key software component is the database manager. This program maintains the integrity of the database, records transactions, translates user requirements for access into physical transfers and data shuffling, and enforces the security rules, etc. The place of the database manager within the DBMS is shown in figure 6.

Beyond this there are a variety of other data management programs of which the principal ones are:

- Data Dictionary, which is used to record the occurrence and the user of data.
- Query languages, which are used to give end-users direct access to their data.
- Report-generators, which are used for the rapid preparation of reports, the reports usually being printed.
- System building tools, which are used for the rapid development of application systems.

These components are discussed further in section VI.B.

Database systems are commonly also on-line systems. They, therefore, involve screen managers, TP monitors, telecommunications access software, terminal controllers, etc. Some of their implications for database management are explained in section VII.

IV. IMPORTANT CONCEPTS IN DATA MANAGEMENT

In conventional application systems data management is a fully integrated part of the application program. Data is described in the program, and anyone who wishes to access it must find a description and then transcribe that description into his own program. DBMS grew out of realisation that there must be better ways of doing this. Modern DBMSs find their origins in three parallel lines of development as discussed in the following paragraphs.

In the first line of development it was realised (in the mid-sixties) that a great deal of commonality existed between the data management needs of different applications. To take advantage of this factor the description of the data was separated from the applications that processed it, and the necessary links were made by a new system component – the DBMS. The emphasis with this approach was on providing a separate description for existing data structures. (This is why hierarchies are so important in IMS, because hierarchies already existed in conventional filing systems). One important consequence of describing the data separately was that it could then be accessed by non-procedural query facilities such as IBM's Generalised Information System and Informatics Mark IV.

The second line of development came from the desire to find better ways of storing the data. Systems in this line include Adabas, which separates indexes from prime data and stores the latter in a compressed form.

The third line of development was the desire to describe data in itself, rather than in its stored form. This line has given us the Codasyl proposals, the relational approach, and the entity data model.

This variety of origins partly explains the complex and the conflicting terminology that obstructs the understanding of database technology. An equally potent source of confusion is the DBMS vendors' insistence that their products involve new and unparalleled discoveries. A third source is the variety of data models (a term explained on page 16) on which the various products are based.

A Data Independence

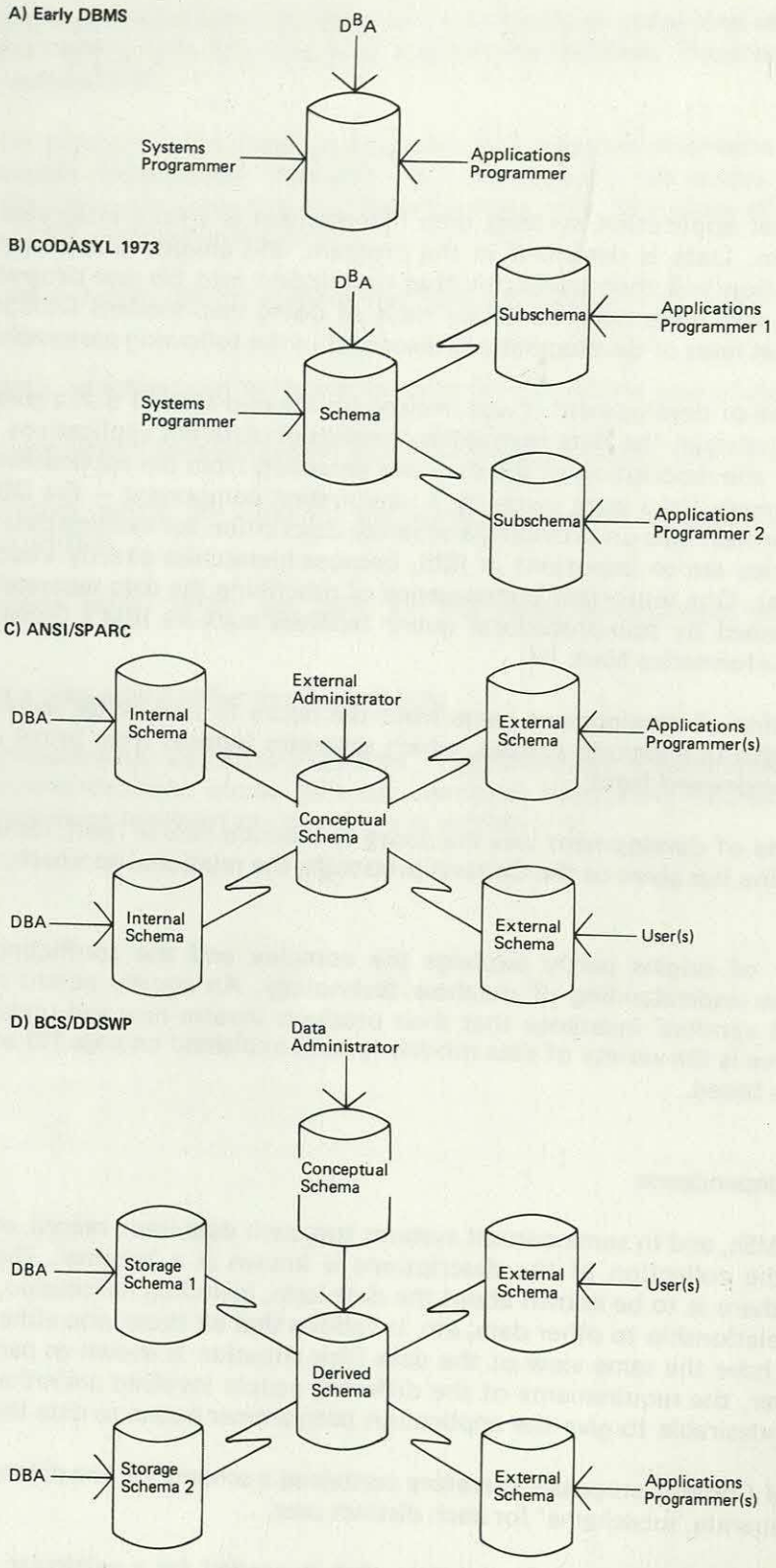
In early DBMSs, and in some current systems too, each data item, record, etc. is described only once, and the collection of the descriptions is known as a 'schema'. The schema must hold everything there is to be known about the data item, including its location, the access method, its logical relationship to other data, etc. It follows that all those who either use or support the DBMS will have the same view of the data (this situation is shown as part A of figure 7). In fact, however, the requirements of the different people involved are rather different. It is, for instance, undesirable to give the application programmer access to data that he does not need.

The original Codasyl proposals therefore contained a schema for the database administrator to use and a separate 'subschema' for each distinct user.

A Codasyl subschema describes the data that is needed for a particular program or suite of

programs. It is a subset of the schema and both a schema and a subschema are shown in part B of figure 7. This concept may be called 'logical data independence'.

Figure 7 DBMS Architectures



The Standards Planning and Requirements Committee of the American National Standards Institute (ANSI/SPARC) after considering what aspects of DBMSs were suitable for standardisation, argued that the physical properties of the data should be stored in a third schema which they proposed to call the internal schema (as shown in part C of figure 7). (They wished to rename the other schemas also). This concept is called 'physical data independence'.

The existence of two internal schemas for a single conceptual schema shows that several different arrangements of computer storage and access methods may correspond to a single logical view of the data.

In most database design methodologies there is a conceptual design, a model of the finished database, long before the DBMS schemas are written, and this is widely called the conceptual data model. The Data Dictionary Systems Working Party of the British Computer Society (BCS/DDSWP) has proposed that this should be held on the computer either in the data dictionary or otherwise. This situation is shown in part D of figure 7, although neither the figure nor the terminology is exactly that used by the DDSWP.

The crucial feature of data independence is the fact that the user does not need to know things that are irrelevant to his task. Logical data independence frees both the user and the programmer from any need to know about the overall structure of the database. It follows that applications programs need not be changed when the overall structure changes (for example, following the expansion of the database to include a new application area which overlaps with the previous one, as shown in figure 8).

A variety of data independence is possible, and some of the more important ones are shown in figure 9 in approximate ascending order of difficulty.

Each of these types can be achieved, and most of them have been implemented, at least on an experimental system. Data independence is most useful to the casual user, but in applications programming, efficiency and clarity may be obtained by taking account of the actual storage structures.

Physical data independence offers the following benefits:

- Improved performance, because the database administrator may vary the physical storage without affecting the data in the views of either the data administrator, the programmer or the end user. Possible changes include blocking factors, device types, hashing algorithms, and physical location.

- Reduced maintenance, because the hardware and the file organisations that are used for storage may be changed without requiring changes to be made to the programs.

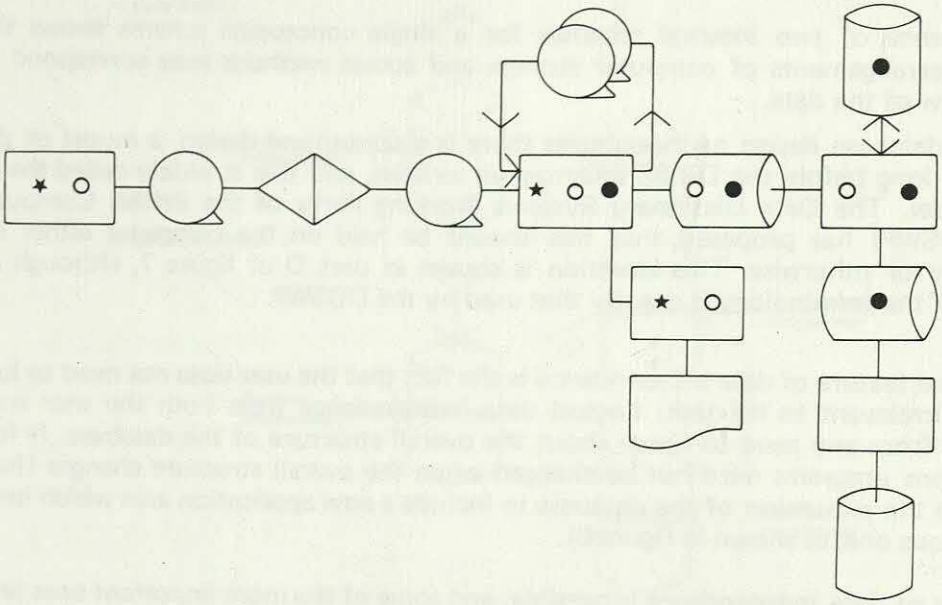
- Easier access by end users, because their views of the data will change less often.

Most DBMSs provide reasonable physical data independence at the lower levels. But they either reveal details of the implementation or depend on logical constructs, such as Codasyl sets, which have, in practice, only one available implementation (in this case, pointer chains). Almost all available DBMSs require all the data to be physically present at one computer centre, and the range of devices supported may be limited (for example, ICL's IDMS does not support their innovative Content Addressable Filestore (CAFS) – which is described in section V.B).

Logical data independence overlays physical data independence, and physical data dependencies will usually appear in both the user's and the programmer's views of data as well. Logical data independence (the independence of the global and the user's views of the database) offers the following benefits:

Figure 8 Data Dependence/Data Independence

Data Dependence



★ Processing Change

○ Conceptual Structure Change

● Physical Storage Change

Applications	Data Administration ★	Data Storage ★
Change Data Computer Program Maintenance	Restructuring	Reorganisation

Data Independence

Figure 9 Types of Data Independence

Type of Data Independence	Defined by the things the user need not know
Device Independence	The type(s) of storage device in use.
File Organisation Independence	The organisation of the physical file, e.g. block size.
File Type Independence	The logical organisation of the file, e.g. indexed sequential.
Element Order Independence	The order of elements within the stored record.
Element Format Independence	Whether numbers are stored in binary, display, packed decimal; whether strings are stored compressed, etc.
Access Path Independence	Which data elements are keys and which sort of record the data is in.
Location Independence	The physical location of the data.
Structure Independence	The relationships between the various data.
Language Independence	The programming or query language for which the data was defined.

- Reduced maintenance, because data structure changes will not require recompilation of programs nor will program changes require restructuring of the database (providing that the required data is available somewhere).

- Faster development of new systems, because old applications will not be affected.

- Easier implementation of applications and complex queries, because the programmer can be given a view of the data that suits his thinking and programming language.

At present most DBMSs provide only limited logical data independence. For example, the IDMS subschema is compatible only with COBOL, and it can contain no sets that are not in the schema. The IMS programmer, to take another example, finds that aspects of the physical implementation limit his manipulations. In some cases, users have provided extra data independence of their own, for example:

- Hoskyns Manufacturing Applications Systems (MAS) are written in terms of a virtual database, which is mapped onto the actual database by an access module (five DBMSs are supported).

- Some Total users do not access Total directly, but access it only through software of their own that provides easier and cleaner access.

The following examples illustrate that vendors are moving in the same direction.

- IBM have recently released IMPS, which provides a relational view of an IMS database.

- Cincom's new Total Information System (TIS) provides user and programmer interfaces that are independent of the underlying DBMS, which thus need not be Total Cincom's own DBMS.

Despite its apparent complexity, data independence is a crucial concept in database management, and vendors are extending their products to provide more of it. All data processing staff who work with DBMSs need to understand this concept.

B Data Models

The term 'data model' is normally used to mean a description of the data relevant to one particular applications system or one group of systems. A model is normally embodied in a schema, but it may also be held in a data dictionary. Any model must be built of certain elements, and a set of these elements may, in a more abstract sense, be called a data model. It is data models in this more abstract sense that we discuss below.

Data models are not new, and the COBOL data model, for example, consists of files, records, group items, elementary items, tables, indexes, etc. These are the only ways in which data may be described in COBOL and if some other structure or type is needed, such as a set or a switch, then it must be simulated using the basic elements.

Data models enter into database work at two levels — the conceptual level and the implementation level.

1. Conceptual data models

The conceptual model for a business is produced by data analysis, and we discuss this in the next section. It should be as free as possible of implementation details and, indeed, of any detail that is not essential to an understanding of the basic structure of the data. A conceptual data model should have the following characteristics:

- Simplicity. It should have a minimum number of elements and rules for combining them.

- Power. It should describe any particular structure in a concise way.

- Picturability. It should be possible to represent a structure diagrammatically, because many people find diagrams easier to understand than either words or a mathematical notation.

- Appropriateness. The elements of the model should correspond directly to things in the real world.

- Uniqueness. There should be just one way to represent a thing or a relationship in the real world.

– Natural terminology. It should not be necessary to learn any new concepts or any eccentric meanings of normal words.

These characteristics are, in some cases, conflicting. For instance, some people may find a concise notation difficult to understand.

The models most often suggested for the conceptual level are the entity model and the relational model, though the binary model is favoured in some quarters.

The *entity* model describes data in terms of entities that have attributes and that stand in relationships with one another. An entity is any person, thing, event, document or legal object, etc. that is of interest to the system.

The entity model is easy for DP and user staff to understand, though some training is necessary if they are to use it with precision. It has a few basic concepts and these are drawn from normal discourse. Structures can be shown as diagrams, and indeed this is the most usual method employed (an example is given in figure 10).

The entity model has been implemented by ICL as the conceptual part of the 2900 Data Dictionary System. It is likely that other data dictionaries will be extended to include it.

The *relational* model treats all data as tables. The model is defined formally in quasi-mathematical terms, and from this formal definition come a set of rules for deciding precisely which tables should be constructed. This design process is known as 'normalisation' and the resulting tables are said to be in 'third normal form'. Tables in third normal form have three main advantages:

– They can be updated simply, without creating the anomalies found with other structures.

– They can be understood easily by non-DP staff.

– They can be derived by following the normalisation rules.

Tables are easy to read, but the normalisation rules are difficult to master, partly because they are expressed in mathematical terms. Once they have been mastered, however, the rules represent a valuable tool in analysis and design. Relational structures cannot be presented graphically without, in effect, transforming them into entity structures.

The *binary* model describes associations between data items. Diagrams can easily be drawn (as shown in figure 11), but they tend to become unwieldy. IBM's Database Design Aid (DBDA) is based on the binary model.

Each model must also include integrity conditions which may be expressed either schematically (in the entity model) or mathematically (in the relational model). Only recently has the importance of integrity conditions been fully appreciated.

At the conceptual level, judgments between models still seem to be largely subjective, and, in any case, the theoreticians continue to refine the models on offer. For the moment, the entity model (backed by relational analysis) seems the best choice for data processing.

2. Implementation data models

During database design, the conceptual model must be transformed into descriptions of actual records and their relationships. This is done by adding such necessary details as the size and type of fields and making change in the interests of performance (e.g. adding indexes). In most cases, the implementation model is not sufficiently rich to support the

whole of the conceptual model, and the parts that are left out must be included in the application programs.

Figure 10 An Entity Model

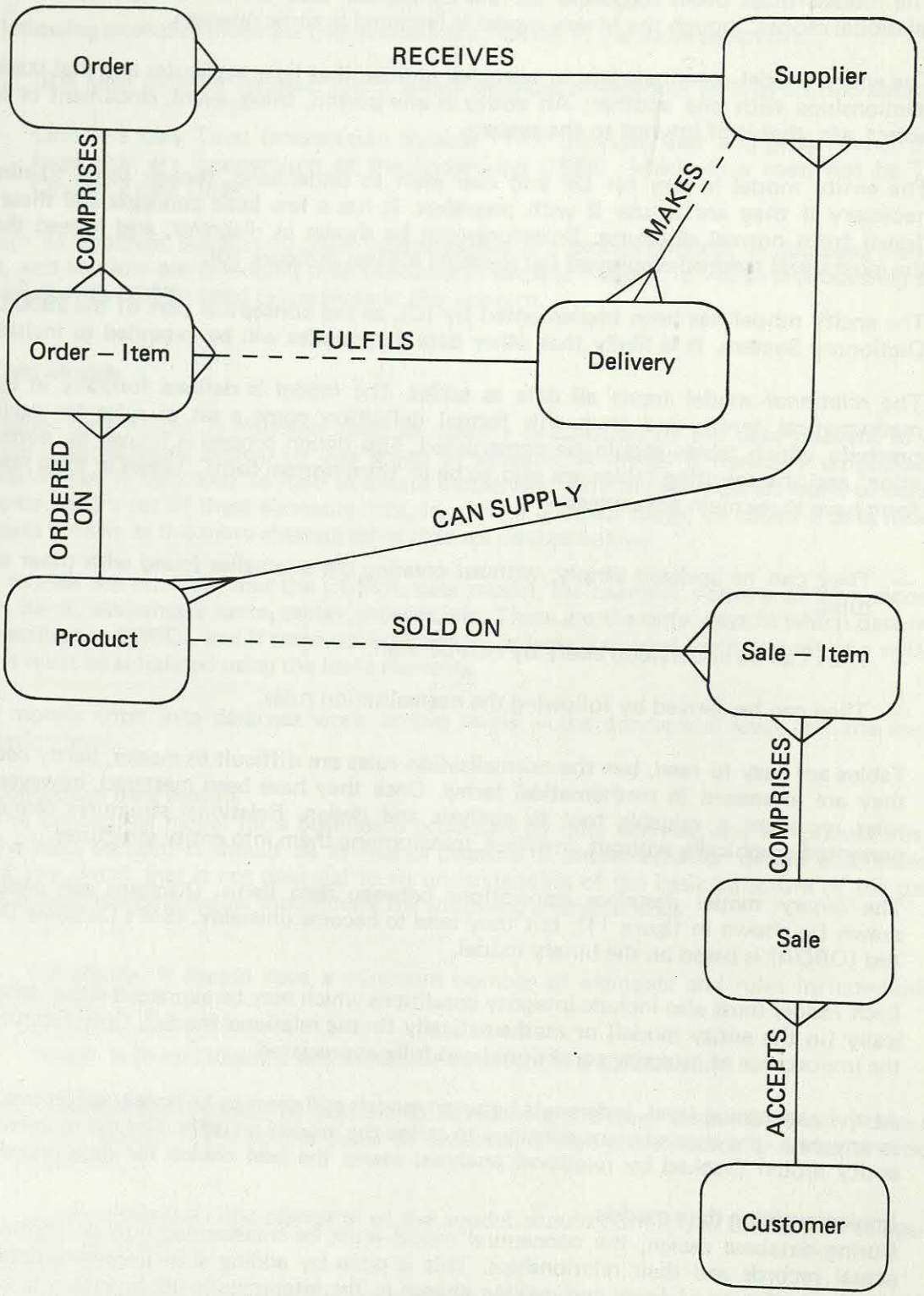
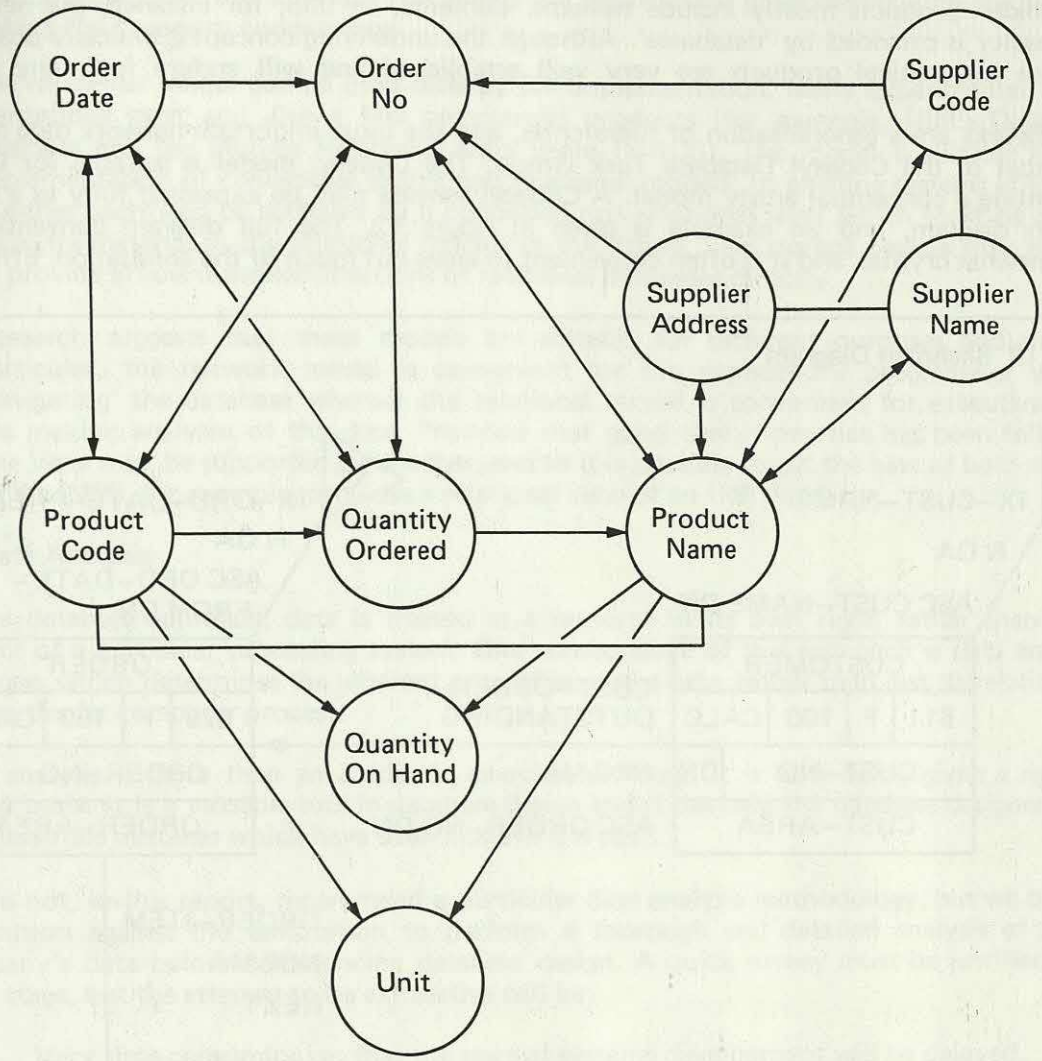


Figure 11 A Binary Model



The implementation model should have the following characteristics:

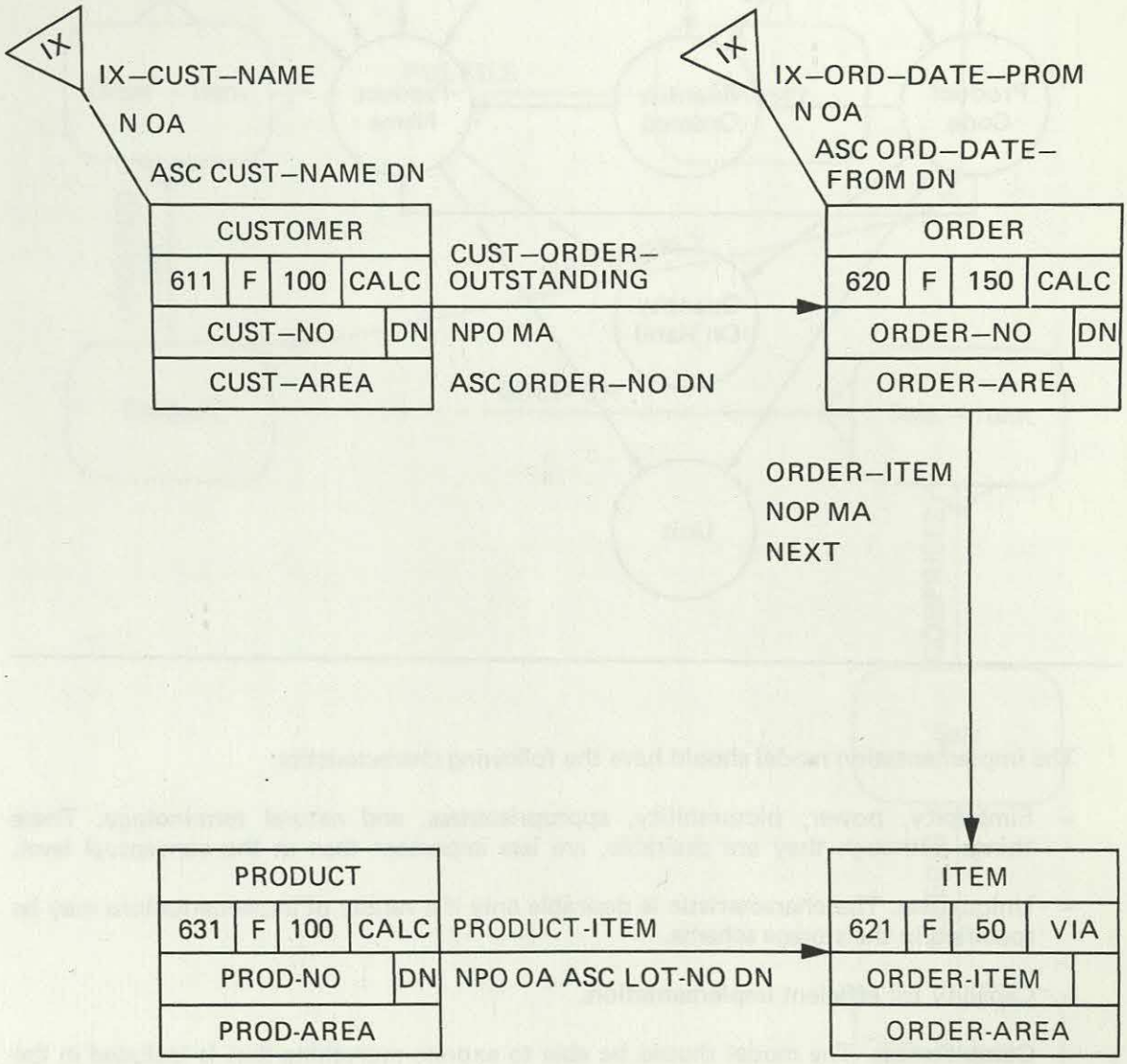
- Simplicity, power, picturability, appropriateness, and natural terminology. These things, although they are desirable, are less important than at the conceptual level.
- Uniqueness. The characteristic is desirable only if a variety of implementations may be specified in the storage schema.
- Capacity for efficient implementation.
- Completeness. The model should be able to express everything that is included in the conceptual model.

The three models most often proposed for implementation are the hierarchical model, the network model, and the relational model.

Hierarchies often occur in the real world. They are therefore found in the conceptual data model and must be implemented. Also, pure hierarchies can be mapped efficiently onto conventional file structures. But hierarchies are not sufficient, and the so-called 'hierarchical' products mostly include network elements. In IMS, for instance, the network capacity is provided by 'databases'. Although the underlying concept is basically obsolete, some hierarchical products are very well established and will endure for many years.

Networks are a generalisation of hierarchies, and the most important network data model is that of the Codasyl Database Task Group. The Codasyl model is suitable for implementing a conceptual entity model. A Codasyl schema may be expressed fully as a Bachman diagram, and an example is given in figure 12. The full diagram convention is somewhat cryptic, and it is often convenient to leave out much of the annotation. Efficient

Figure 12 Bachman Diagram



(Reproduced by courtesy of SCICON)

implementations of the Codasyl model exist (for example, IDS and IDMS) but because the model relies heavily on pointer paths (due to the incomplete separation of the logical and the physical aspects in the model), the database designer is often obliged to exclude some relationships from the schema. This leaves the applications programmer with the task of making the necessary connections.

The *relational* model can be used directly for implementation. Many experimental implementations exist and also a few commercial products (for example, IBM's Query-By-Example and the Multics Relational Data Store (MRDS). The model is suitable for the implementation of conceptual entity and relational models. In existing systems efficiency is either sacrificed or achieved by a concealed non-relational layer which must be guided by extra material in the relational model. In the future, new storage devices may be able to provide efficient implementations of relational databases directly.

Research suggests that these models are suitable for different purposes and that, in particular, the network model is convenient for the applications programmer who is 'navigating' the database whereas the relational model is convenient for executives who are making analyses of the data. Provided that good design practice has been followed, one view may be supported by another, and so it is possible to get the best of both worlds. IBM's IMPS, for example, provides a relational view of an IMS database.

C Data Analysis

In the database approach, data is treated as a resource in its own right, rather than as an adjunct of a particular processing system. One consequence of this approach is data analysis, a process which determines the inherent properties of the data, rather than just its relationship to a particular computer process.

Data analysis is more than an academic exercise, although it is now being given a rigorous logical basis. It is a valuable tool in database design and it can help the database designer avoid the disastrous mistakes which have been made in the past.

We do not, in this report, recommend a particular data analysis methodology, but we do wish to caution against the temptation to perform a thorough and detailed analysis of all the company's data before commencing database design. A quick survey must be justified at an early stage, but the attempt to be exhaustive will be:

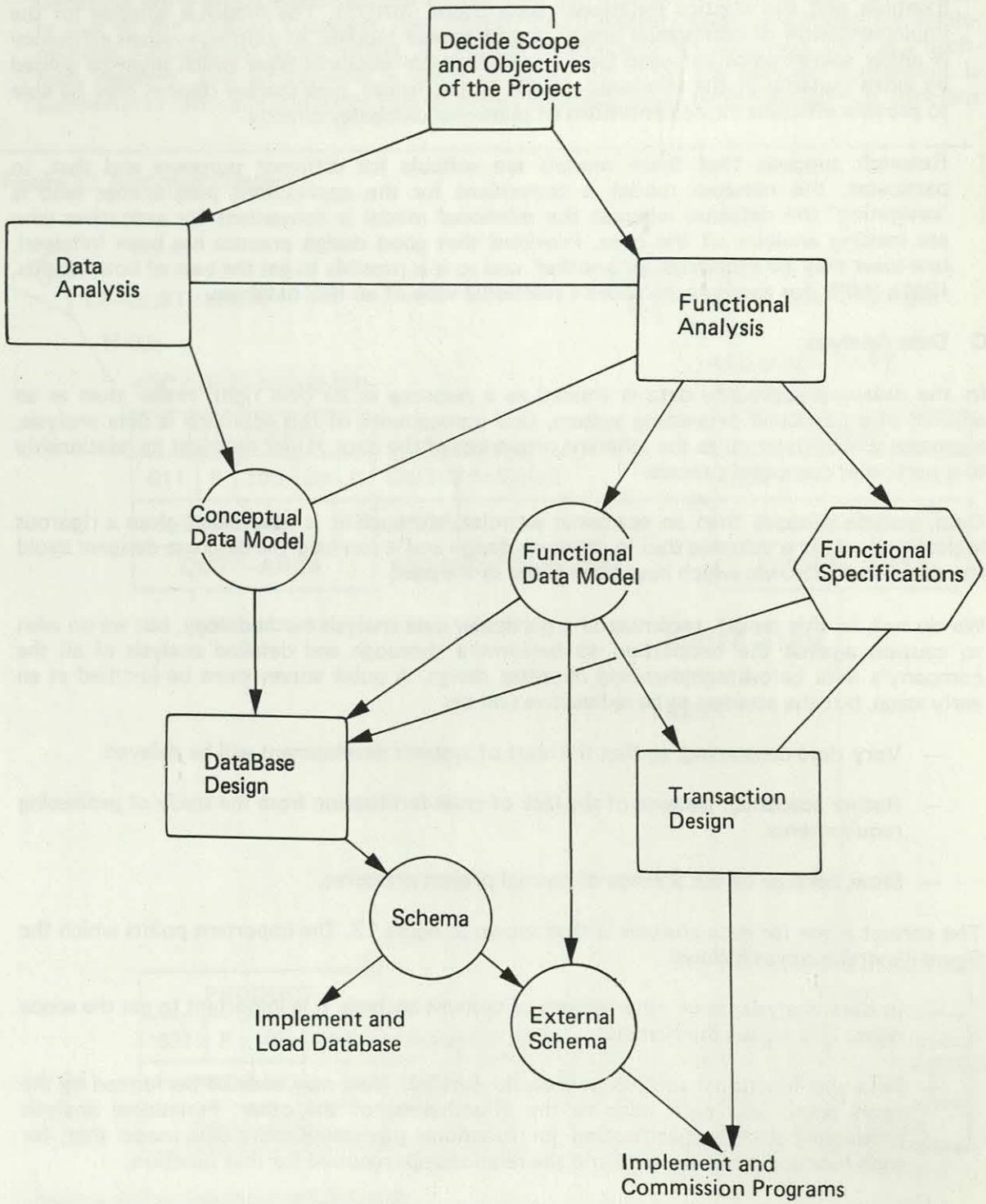
- Very time-consuming, so that the start of systems development will be delayed.
- Rather academic, because of the lack of cross-fertilisation from the study of processing requirements.
- Slow, because of the absence of normal project pressures.

The correct place for data analysis is that shown in figure 13. The important points which the figure illustrates are as follows:

- In data analysis, as in other phases of systems analysis, it is important to get the scope right. This is thus the first step.
- Data and functional analysis proceed in parallel. They may even be performed by the same team, and each adds to the effectiveness of the other. Functional analysis produces both the specification for functional processing and a data model that, for each function, gives the data and the relationships required for that function.
- In database design a database structure is produced which can support the required functional data models and the expected processing load, this load being determined

from the function specifications. Design requires a thorough knowledge of the data management software to be used, and this need not be a DBMS.

Figure 13 The Place of Data Analysis in Computer Systems Development



Data analysis will normally start by a discussion with user management and staff about the people, places, objects and events that are important to the business. These are the 'entities' about which information (the 'attributes') will be stored. The relationships between the entities must also be documented. It is important that user personnel should be actively involved in this phase of the work, and it seems best to document the results as an entity model.

Analysis will then continue with the existing records of the business. Where possible the results should be recorded in a data dictionary.

The results of this stage should be checked by the relational technique of 'normalisation' and only normal forms (which are the result of normalisation) should be allowed into the model.

As an alternative, the normalisation process may be applied directly to the existing computer files and clerical records of the business. This approach is faster, but it may fail to uncover anticipated changes in the business. In addition, it will not encourage user staff to become involved.

Data analysis is valuable even when it is known that a DBMS will not be used. It provides a thorough understanding of the data before design work starts, which is the time when errors can be corrected most cheaply.

It could even be used during the development of manual systems, but there is little evidence that this has been done in the past.

Data analysis also provides a sound base from which the need for a DBMS can be assessed.

In our view, all systems work, with the possible exception of the very simplest, will benefit from the use of data analysis. It should be made an integrated part of the development process, in the sense of figure 13, rather than an exceptional technique that is used only on database projects.

D Database Integrity

The integrity of a system may be compromised by any of the following:

- A clerical error made in entering a transaction.
- An error in the applications program.
- An error in the DBMS and its utilities.
- An error in the hardware, the operating system or the TP monitor.
- An error made by the system operators, and especially an error made in recovering from another error.

Where applicable, conventional and proven methods (validation of input data, file reconciliations and processing reconciliations) will be used to ensure integrity but these are often less effective in an on-line system.

Database presents special problems. The distinctive features that make integrity a greater problem and the appropriate remedies are given in figure 14.

The database audit program is an important tool. It works by reading all or part of the database and checking that certain logical conditions hold good. It checks for example:

- That there is a delivery address for every order.
- That every waiver has been authorised by someone who has the requisite authority.

A random sample may be taken, or some part of the database that is of particular interest may be selected. Offending records may be removed, report, or both. The functions of the audit program are:

- To detect errors, so that they can be corrected (unrecognised errors can be very expensive).
- To reveal patterns of errors, so that changes in either procedures or vetting rules may be made to prevent any recurrence.
- To provide a quantitative measure of the quality of the database for managers, the data administrator, and the auditors.

Figure 14

Feature	Remedies
1. There is less redundancy and this makes it more difficult to recover from crashes because data cannot so easily be reconstructed from related files.	Data Administration Controlled Redundancy
2. Since data no longer belongs to a single application the manager or the department sponsoring an application will no longer feel as responsible for it.	Data Administration Security System
3. A database program has access to the whole database and it is difficult to be sure that it is updating only the parts intended. This is especially true if the database may be updated by people who are not DP professionals.	Logical Data Independence Security System Database Audit
4. The DBMS and its associated utilities are more complicated than the corresponding components of conventional file management systems; and the file management systems may be present as well.	Database Audit

Integrity conditions are often identified during data analysis, and the entity model provides ways of showing various sorts of conditions in the conceptual model. Also the DBMS may be asked to check integrity conditions whenever it does an update. In theory, integrity conditions should make the audit program redundant, but in practice the audit program will probably be required for the purposes of:

- Policing those conditions that are too expensive to check in real time.
- Detecting any errors there may be in the DBMS, the hardware, the utilities, etc.

Real time checking is potentially expensive, but it has the following major benefits:

- The database quality is assured (apart from system faults), and so the reports that are derived from the database are of high and consistent quality.
- Validation and update programming is simplified, because the integrity checking need not be included in the applications program (and cannot be by-passed).
- Systems generation by users is, consequently, both easier and safer. It allows more complex tools to be supplied to end users and it takes the load off the central DP department.
- The control of the data administrator is extended, and so his ability to manage the data resource is improved.

Some existing commercial systems already either provide non-procedural ways to state conditions (for example, Nomad) or provide ways in which procedural tests may be added to the schema (for example, IDMS). In other cases, certain conditions are implicit in the data structures (for example, IMS).

IBM's prototype relational DBMS, System R, provides a number of well designed features for ensuring integrity including:

- Vetting rules for individual fields.
- Vetting rules for groups of related records.
- Relationships between fields in a record.
- Relationships between records of different types.

The conditions are expressed in a non-procedural language called Sequel.

In addition to these facilities at the logical level, the DBMS should include facilities that detect and repair errors at the physical level. Errors may include disc space that is lost through errors in the space management routines and pointers that do not point to records of the correct type.

E Automatic Design and Tuning

Data analysis is essentially a human activity. However, a good data dictionary can help with the documentation of the activity and can detect loose ends that have been left by the analysis team. (Data dictionaries are discussed in section VI.B.) If, as we recommended in part C of this section, the result of data analysis is a normalised entity model, implementation will involve two design stages:

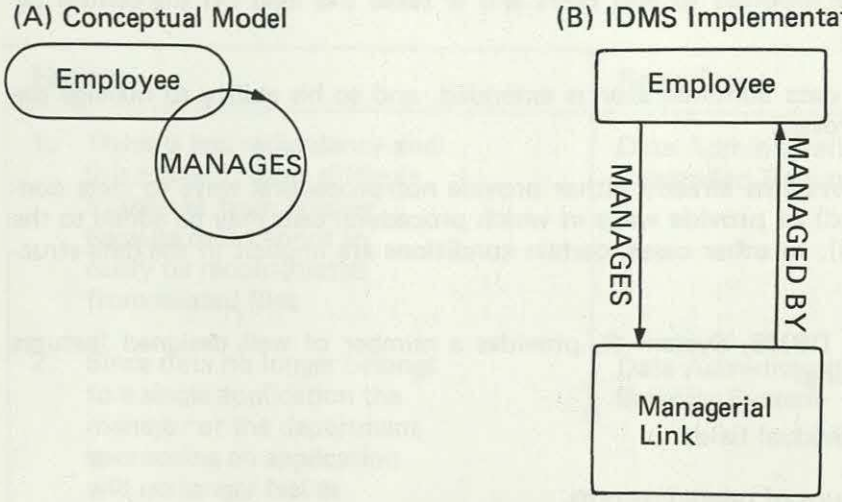
1. Logical design. In this stage the conceptual entities, attributes, relationships and integrity conditions are translated into the terms of the implementation data model.

2. Physical design. In this stage the derived schema is translated into a storage schema.

In logical design, the designer must select for computer implementation only those entities, attributes and relationships that are necessary for the systems that are being built. Since a sophisticated data dictionary will record the use of data by the required functions, it provides the basis for the automation of this selection. However, in any practical system, the designer will need to be able to override the system.

Design problems arise because of conflicts between the conceptual and the implementation data models. For instance, an involuted relationship is quite proper in a conceptual data model – (a) in figure 15 – but is not supported by the Codasyl set construct. In IDMS, then, it must be implemented as in (b) in figure 15.

Figure 15



It has been established that the various data models that are currently in use are equally complete in their ability to represent data structures, and so it will certainly be possible to convert automatically from one model to another. It would be better, however, to choose a better implementation data model, and a number of vendors are moving in this direction.

In physical design, the designer will select the access method for each record (or segment), decide blocking factors, design hashing algorithms, etc. These decisions will then be recorded both in the storage schema and the derived schema, although only the storage schema would be involved if physical data independence was complete.

Experience commonly shows that the use that is made of database systems, especially systems that have online query facilities, is signficantly different from that expected by the designer.

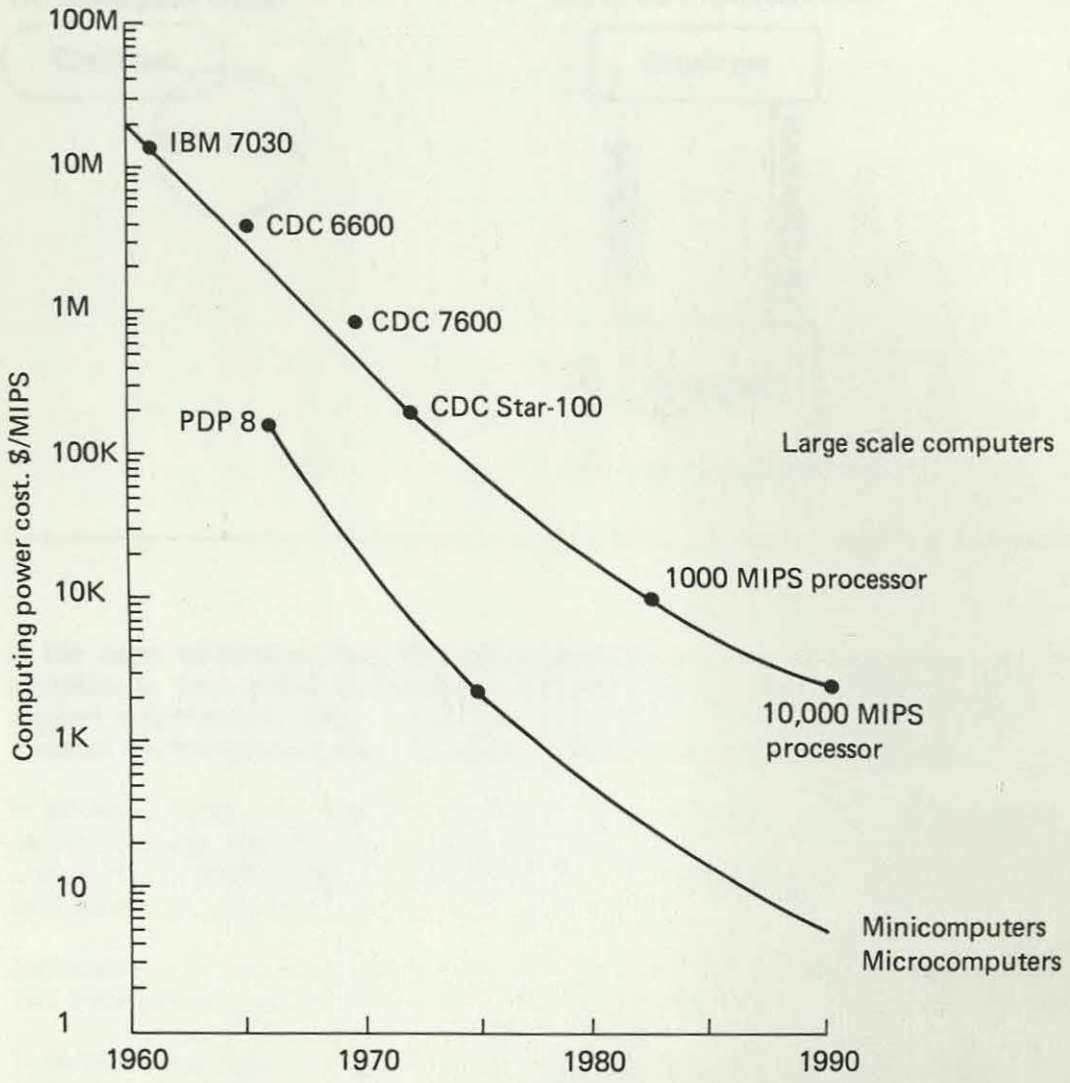
Even if the estimates are right initially, changes in the business will invalidate them sooner or later. Consequently, it will always be beneficial to monitor the performance of the system and to tune it either for higher performance or greater efficiency.

The tuning problem is now well understood theoretically. Work by Professor Stocker and his colleagues at the University of East Anglia, for instance, has shown that a self-optimising system that adjusts its physical storage to accommodate changing patterns of use is quite practical, although so far only experimental systems have been built. Performance prediction aids of a more conventional kind are already available from various sources, and they include analytical models based on queueing theory and complex simulation models. These aids will assist the database administrator until such time as the DBMS is able to take over the whole task.

A The Cost of Processors and Memory

There is an underlying trend of falling hardware costs, and one example of this — the trend in processing power — is shown in figure 16, and it can be seen from this that by 1985 a one

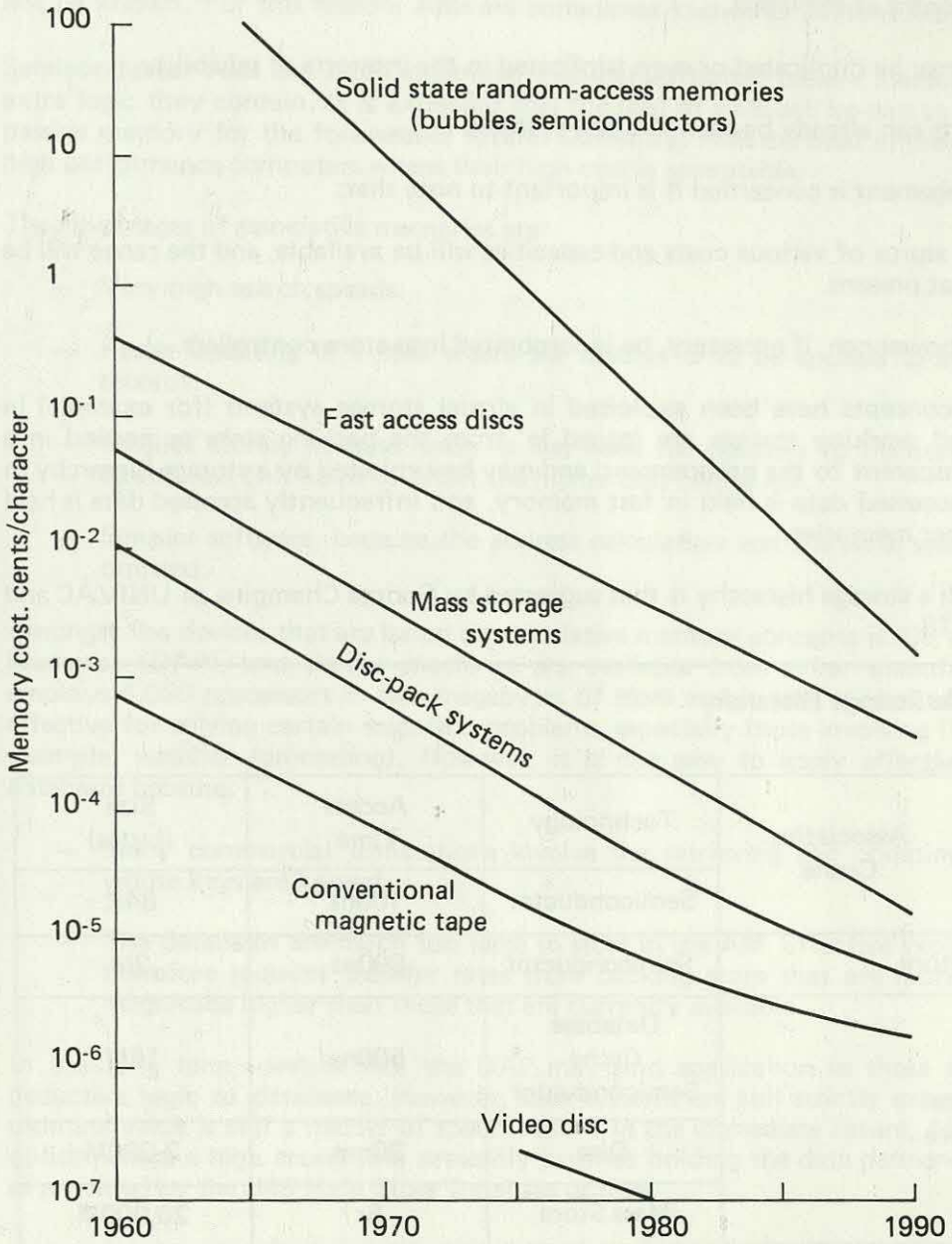
Figure 16 Computer Power Cost



million instruction per second (1 MIPS) microprocessor (which is approximately equivalent to the power of an IBM 3032 or an ICL 2972) is expected to cost £50. It does not follow, of course, that IBM will actually be selling a 3032 cpu for £50, but that is the expected production cost. The remainder of the selling price will pay for software, support, marketing, and so on.

The cost trends for various types of memory are shown in figure 17. It can be seen that the

Figure 17 Memory Cost Trends



rapid decline in memory cost is expected to continue for at least the next decade. The direct effects of these cost trends are:

- The cost of including a microprocessor and some store in any device will be trivial compared to the cost of the electromechanical parts and the cost of programming.

- Mainframe computers will have very large main stores as a matter of course. (By 1985 a single memory chip will hold half a megabyte and will cost £50). If the necessary reliability can be achieved, stores of several hundred megabytes will become feasible.

- Large disc stores will remain an important part of computer systems for the foreseeable future. However, semiconductor and magnetic bubble devices will displace them for rapid access applications and mass storage devices will displace them for large volume slow response functions. A one thousand megabyte disc drive will be available within eighteen months at the latest.

- Hardware may be duplicated or even triplicated in the interests of reliability.

Each of these effects can already be seen in a few systems.

As far as data management is concerned it is important to note that:

- A range of stores of various costs and capacities will be available, and the range will be wider than at present.

- Processing power can, if necessary, be incorporated into store controllers.

Storage hierarchy concepts have been exploited in virtual storage systems (for example) in which modules and working storage are 'paged in' from the backing store as needed in a manner that is transparent to the programmer, and may be exploited by a storage hierarchy in which frequently accessed data is held in fast memory, and infrequently accessed data is held in slower and cheaper memories.

An example of such a storage hierarchy is that suggested by George Champine of UNIVAC and described in figure 18.

Figure 18 A Possible Storage Hierarchy

<table>
<tr><td rowspan="2">Processor</td><td rowspan="2">Associative Cache</td><td>Technology</td><td>Access Time</td><td>Size (bytes)</td></tr>
<tr><td>Semiconductor</td><td>100ns</td><td>64K</td></tr>
<tr><td colspan="2">Main Store</td><td>Semiconductor</td><td>800ns</td><td>2M</td></tr>
<tr><td colspan="2" rowspan="3">Data Store</td><td>Database Cache Semiconductor</td><td>500ns</td><td>16M</td></tr>
<tr><td>Disc</td><td>30ms</td><td>2,000M</td></tr>
<tr><td>Mass Store</td><td>5s</td><td>20,000M</td></tr>
</table>

Such a hierarchy would combine costs typical of the slower devices, with a mean access time close to those of the faster ones, whilst providing the user with access to the whole of the

stored data. The effectiveness of such hierarchies has been confirmed both by research and by the experience of systems that use a storage hierarchy for the database (for example Tandem's Enscribe).

B Associative Memories

In conventional memories, data is read or written one record at a time, and the physical address of the record must be known first. An associative memory (AM) has in-built logic that enables it to read or write a number of records in parallel. Moreover, the AM can be told to retrieve, or to modify every record that contains some value, so that the physical address need not be known. For this reason, AMs are sometimes known as Content Addressable Memories.

Semiconductor AMs are more expensive than conventional, passive memories because of the extra logic they contain. It is expected that the cost of AMs will be two to three times that of passive memory for the foreseeable future. Currently, AMs are used in the paging hardware of high performance computers where their high cost is acceptable.

The advantages of associative memories are:

- Very high search speeds.
- Faster updating in a case where the change is to be applied to a number of similar records.
- Smaller stores, because there is less need for pointers to maintain the information structure. (This helps to offset the higher cost of the AM.)
- Simpler software, because the address calculation and the serial search routines can be omitted.

Amongst the devices that are based on associative memory concepts is ICL's Distributed Array Processor (DAP), and similar machines are available from other manufacturers. The DAP employs 4,096 processors in two megabytes of store and is sold at about £600,000. It is very effective for solving certain scientific problems, especially those involving fluid mechanics (for example, weather forecasting). However, it is not easy to apply effectively to commercial databases because:

- Many commercial transactions involve the retrieving and updating of single records whose keys are known.
- The databases are much too large to store in the AM. Effective exploitation of the AM therefore requires transfer rates from backing store that are more than an order of magnitude higher than those that are currently available.

In the long term, devices like the DAP may find application in those systems that apply deductive logic to databases. However, such systems are still strictly experimental, and their ultimate value is still a matter of specification. In the immediate future, AMs are an attractive option where a high access rate presently justifies holding the data permanently in main store as provided by the IMS Main Store Database option.

A compromise exists between holding the whole database in AM or in passive storage only. In ICL's Content Addressable File Store (CAFS) all the data is held on a disc drive that has been modified to allow reading from all heads in parallel. The CAFS controller can read and search half a cylinder in a single revolution, and it can conduct a number of such searches in parallel. In retrieval, CAFS is ten to fifty times faster than conventional search techniques.

It is currently being evaluated by the British Post Office for an on-line telephone directory inquiry system.

CAFS is valuable not only for the rather peculiar requirements of directory enquiries, but also for implementing the retrieval and manipulation of sets of records that are needed in both query systems and relational databases. Content addressable stores based on discs are likely to be supplemented, and ultimately superseded, by stores based on magnetic bubbles.

C Novel Database Processors

Hardware developments are of interest to the DP manager only if they lead to new products. The possible new database processor products are:

- An up-rated disc controller, which is compatible with existing devices but provides faster access and higher throughput.
- An enhanced disc controller, which provides the retrieval and manipulation of sets of records, rather than of just single records or blocks.
- A back-end processor, which provides all database management functions for one or more mainframes.
- A database machine to which intelligent terminals might be connected either directly or via network processors.

Up-rated disc controllers reduce the problems of contention for disc space and so they support higher transaction rates and allow transactions to access more records whilst giving an acceptable response time. The latter is important in database query and analysis work. The increasing size of disc stores and the falling cost per character are largely achieved by the increases in recording density. Since disc access times fall more slowly than density increases their trends lead to increased contention. Improved controllers are thus necessary even to maintain present performance. Many current disc controllers contain more logic than they use, and so it is likely that mainframe vendors will use this logic to extend capacity (with or without associative memory).

Enhanced disc controllers provide both increased performance and new facilities. To exploit these new facilities, however, will require changes to the data management software, and it is not yet clear precisely what mixture of hardware and software will give the best results. These devices will be introduced initially for special purposes (such as the use of CAFS for directory enquiries mentioned earlier), and they will be integrated into general-purpose DBMSs only slowly.

Back-end processors offer advantages in security and integrity, rather than in pure performance. Where one back-end processor can support several applications machines, the expense of holding several copies of the DBMS is saved, but it is saved at the cost of extra data communications. With its Multi-Computer Support feature, Cullinane has provided for users to establish back-end processors for IDMS. In initial versions the back-end machine must be either an IBM 370 or a machine compatible with it, but Cullinane is actively considering supplying its own hardware.

From the viewpoint of the independent DBMS vendors, back-end machines have the following advantages:

- Portability across ranges of mainframes.
- The ability to lock the user into the DBMS.

- Less dependence on the mainframe supplier's hardware and operating systems.

However, for the vendor, back-end machines have the disadvantage that he must:

- Provide hardware support.
- Compete with hardware manufacturers in a market that is becoming increasingly competitive and in which he may have no previous experience.
- Innovate in both hardware and software, if he is to obtain the full benefits. This will increase both his development and his maintenance problems.

The independent manufacturers are therefore likely to come to differing conclusions about the wisdom of making this move.

Database machines that support terminals directly are not really distinct from mainframes and minicomputers with DBMSs. An interesting marketing initiative is National CSS's marketing of an IBM-compatible super mini, the NCSS3200, to run their Nomad DBMS. This is important in that it brings a powerful DBMS within the reach of the smaller user.

Very big database machines will find application in public database services (as discussed in Report No. 10: Public On-line Information Retrieval Services).

Some technical questions remain unanswered but the main constraints on introducing novel database processors are commercial ones. IBM, to take a critical case, will not wish to introduce any product which adversely impacts their IMS rental base. On the other hand, IBM will wish to keep everyone, and especially their competitors, off balance and will continue to enhance their product line. They will also change the interfaces between the mainframe and the disc controller as they upgrade the latter. IBM's own need to maintain and enhance software will limit the speed at which such changes can be introduced. As in the past, independent vendors will be able to handle the new IBM interfaces within six to twelve months of IBM product release.

We discussed the several components of a DBMS earlier in this report – figure 5 showed what some of the major vendors offer, and figure 6 showed the relationships between these components.

The principal vendors are extending their product lines to include at least most of the software discussed in this section. At the same time each vendor is trying not only to meet his customer's changing needs but also to so commit his customers to his products as to make it difficult for them to migrate to competitive products.

To win new customers, vendors are also providing interfaces to each other's products. While this adds to the complexity of the market it also provides opportunities for the more advanced user.

The major types of products discussed are:

- Database managers. These are the heart of DBMS and they are discussed in A below.
- Data dictionaries and design aids. These are used in data and systems analysis and in logical database design. They are discussed in B below.
- Query languages. These comprise batch-oriented report generators and high-level data manipulation languages. They are discussed in C below.
- System building tools These are those DBMSs that are intended to provide an alternative to conventional systems development techniques. They are described in D below.
- Performance prediction and monitoring aids. These are the tuning options provided by the database managers. They are discussed in F below.

A Database Managers

The database manager (DBM) is the central element in a DBMS. It provides access to the database and enforces privacy, security, and integrity on behalf of the data administrator. For the purposes of this discussion, the DBM is taken to include the utilities responsible for reorganising and restructuring the database, for integrity checking, and for maintaining the schemas.

In this section a number of significant features of DBMs are discussed and special emphasis is placed on those that are relevant to the improved understanding of databases given in section III.

As with any software product, it is desirable that a DBM should be reliable, well documented, well supported by the vendor, and efficient. It is also desirable that training and consultancy should be available. Some of these points are covered by the annual Datapro survey of software products, and the results of their most recent survey are shown in figure 19.

Figure 19 How Users Rated the Popular Database Management Systems

Package and Vendor	Number of Users reporting	Weighted Average User Ratings: Overall Satisfaction	Throughput/Efficiency	Ease of Installation	Ease of Use	Documentation	Vendor Technical Support	Training
*ADABAS, Software ag of N.A.	28	3.5	3.1	3.5	3.3	2.8	3.1	3.1
Datacom/DB, Applied Data Research	15	2.9	3.1	3.0	3.2	2.5	2.9	2.8
DBMS–10/20, Digital Equipment Corp	6	2.8	2.5	3.0	2.8	2.5	2.2	2.7
DBOMP, IBM Corp., DPD	25	2.8	2.5	2.1	2.5	2.3	2.3	2.3
DL/1 DOS/VS, IBM Corp., DPD	36	2.8	2.5	2.5	2.5	2.5	2.7	2.8
DL/1 Entry, IBM Corp., DPD	8	3.0	2.6	2.6	3.1	2.5	3.0	2.8
DMS–II, Burroughs Corp.	30	3.4	3.3	3.4	3.4	2.5	2.6	2.5
DPL, National Information Systems	6	3.5	2.3	3.8	3.5	3.0	3.0	3.2
*IDMS, Cullinane Corp.	42	3.5	3.3	3.5	3.4	3.1	3.5	3.3
IMAGE/1000, Hewlett-Packard Co.	9	3.0	2.9	3.4	3.0	2.5	2.7	3.3
*IMAGE 3000, Hewlett-Packard Co.	30	3.5	3.3	3.7	3.6	3.2	3.0	2.7
IMS, IBM Corp., DPD	34	2.9	2.4	2.2	2.5	2.8	2.8	2.6
INQUIRE, Infodata Systems, Inc.	8	3.8	2.8	3.6	3.3	2.9	3.3	2.9
SYSTEM 2000, MRI Systems	24	3.3	2.9	3.1	3.2	2.8	2.8	3.0
TOTAL, Cincom Systems, Inc.	108	3.2	3.1	3.2	3.2	2.7	2.7	2.7

Meaning of Rating: 4 – Excellent; 3 – Good; 2 – Fair; 1 – Poor
*Datapro Honour Roll

Source: *Datamation,* December 1978

The main features of database managers are discussed below. For particular purposes other features may be of special interest and their absence does not necessarily mean that they will not be relevant to an evaluation.

1. **Storage options**

A wide variety of storage options are needed in order to cope with the variety of ways and frequencies in which data is manipulated in different systems. For instance, a relationship between entities of two types may be implemented by:

- Storing the corresponding records together.
- Holding the physical address of one record in the other.
- Linking the records of one type by pointers (which are physical addresses).
- Holding the key of one record in the other or in a separate record.
- Using an index.

These options are not equally valuable. In general, physical addresses should be used with restraint and only where high performance is needed. In such cases, however, they are most valuable.

A similar range of choice is desirable for the implementation of entities and attributes, for the placement of data on disc, and for the choice of backing store devices.

No single option is ideal. The database administrator needs a wide range of choice, together with the facilities for converting from one to another without impact on the conceptual model. The reorganisation facilities should include:

- Consolidating of overflow areas on indexed sequential files.
- Changing hashing algorithms.
- Changing device types.
- Changing the implementation of a relationship.

Relational DBMSs usually have a variety of these options available. They are *not* usually implemented solely by files that contain only records of a single type ('flat files'). The advantages of two database managers may be combined if one allows access to data maintained by the other. Ramis, for instance, may be used on IMS and Adabas databases, and Codasyl 'ON procedures' may also be used to access alien data.

The present tendency is to give the database administrator a wider choice of storage options and to provide better utilities for reorganisation.

2. **Physical data independence**

The virtues of physical data independence were discussed earlier. In almost all current DBMSs this is incomplete in one or more of the following ways:

- The nature and the sizes of data items are visible (except for Nomad).
- The combination of fields into records (or segments) is visible (eg Adabas).
- The mechanism used to implement relationships is visible (eg Total).
- File access methods, including the existence of indexes, may be visible (eg IMS).

Consequently, any change in these structures will require corresponding changes in the programs.

Relational systems, such as IBM's System R, by contrast usually possess a high degree of physical data independence.

In the older DBMSs there is often no clear distinction between the logical and the physical views of data. An IMS hierarchy, for instance, is both a logical relationship between segments and a description of the physical storage. The original Codasyl proposals were also deficient in this area. However, the suppliers of Codasyl implementations do now seem to have accepted the need for separate storage schemas and this means that physical data independence will increase. The hierarchical products are less likely to improve.

3. **Logical data independence**
   The virtues of logical data independence were also discussed above. Codasyl subschemas allow both the user and the programmer to be shown a subset of the records and sets in the schema. More powerful facilities than this are possible, and System R for example, allows a user 'view' (the equivalent of a subschema) to include records that are not physically present, but are created from other records as needed.

   In Codasyl systems, it is possible to arrange that any attempt to access a record causes a user-written procedure to be executed. This procedure can then create the desired record from other database records, or, indeed, from data outside the database entirely.

4. **Recovery**
   Following the failure of either a program, processor or disc file it should be possible to recover and to continue with only a little delay.

   The failure of a program should cause the loss of that single transaction only, and any updates that it has already done should be undone ('backed out'). If necessary, other transactions may be reversed in order to ensure integrity.

   The failure of a processor will cause the loss of all current transactions, but the loss should be limited to those transactions that are not yet completed. IDMS and IMS are among the systems that meet this requirement.

   The failure of a disc file may cause transactions to be aborted. The rest of the system should be able to continue processing whilst the lost data is reconstituted from back-up files and the log.

   Facilities of this sort require the logging of copies of database records ('images') both before and after modification, and the ability to run only part of the system after a partial failure.

5. **Sequential processing**
   Despite the predominance of on-line working in database applications it is often necessary to process large parts of the database sequentially. Some DBMSs do not support this, and it is necessary to have an extract-sort-process-update sequence outside the database. ICL's IDMS will shortly provide an option to view part of the database as a sequential file (FAME). This may also be seen as adding to logical data independence.

6. **Divisibility**
   It should be possible to reorganise parts of the database and to continue processing if part becomes corrupt. Hierarchical systems usually treat the various hierarchies separately for these purposes. Codasyl 'realms' provide a more flexible facility of the same kind.

7. **Integrity checks**
   The need for integrity checks was discussed earlier. No commercially available system

provides adequate features for integrity checking, though Codasyl 'ON procedures' may be used.

8. **Testing**
   The DBMS should provide test data generators and comparison facilities. For example, IMS has a test facility which will:

   - Create a database that contains arbitrary values but conforms to the schema.
   - Compare 'before and after' versions of a database.
   - List the database.

9. **Restructuring**
   It will certainly be necessary from time to time to add fields to records and new record types to the database. Utilities for this should be available and generally are. It should not be necessary to unload the whole database (and the IDMS utility does not even require the records in question to be unloaded).

B Data Dictionaries and Design Aids

Data dictionaries are a growth area in the database field, and there is still a considerable lack of clarity as to the functions they should provide. Early dictionary products either recorded the structures created and the calls written in the program or else served merely to support a query language. More recent products (IBM's IMS/DD or ICL's 2900 DDS for instance) play a more active role by serving as the source from which the database description or schema respectively are generated. These differences in concept make it difficult to compare the products.

A fully developed data dictionary should assist the fact finding activity and also the data and functional analysis. It should therefore be possible to record the details of clerical and non-database systems. This is possible with, for instance, 2900 DDS or Data Manager, but not with most other products. It should also be possible to record the conceptual data model. The only product that currently supports this is 2900 DDS, although others are currently being extended. Figure 20 shows the elements of the entity data model used in 2900 DDS.

IBM's Data Base Design Aid (DBDA) accepts, as input, details of the data items established by fact finding and details of their inter-relationships, and it derives implementation data structures from them. DBDA seems to be the only product to provide this degree of support. In our view the data dictionary can be valuable at three major stages of a development project as discussed below.

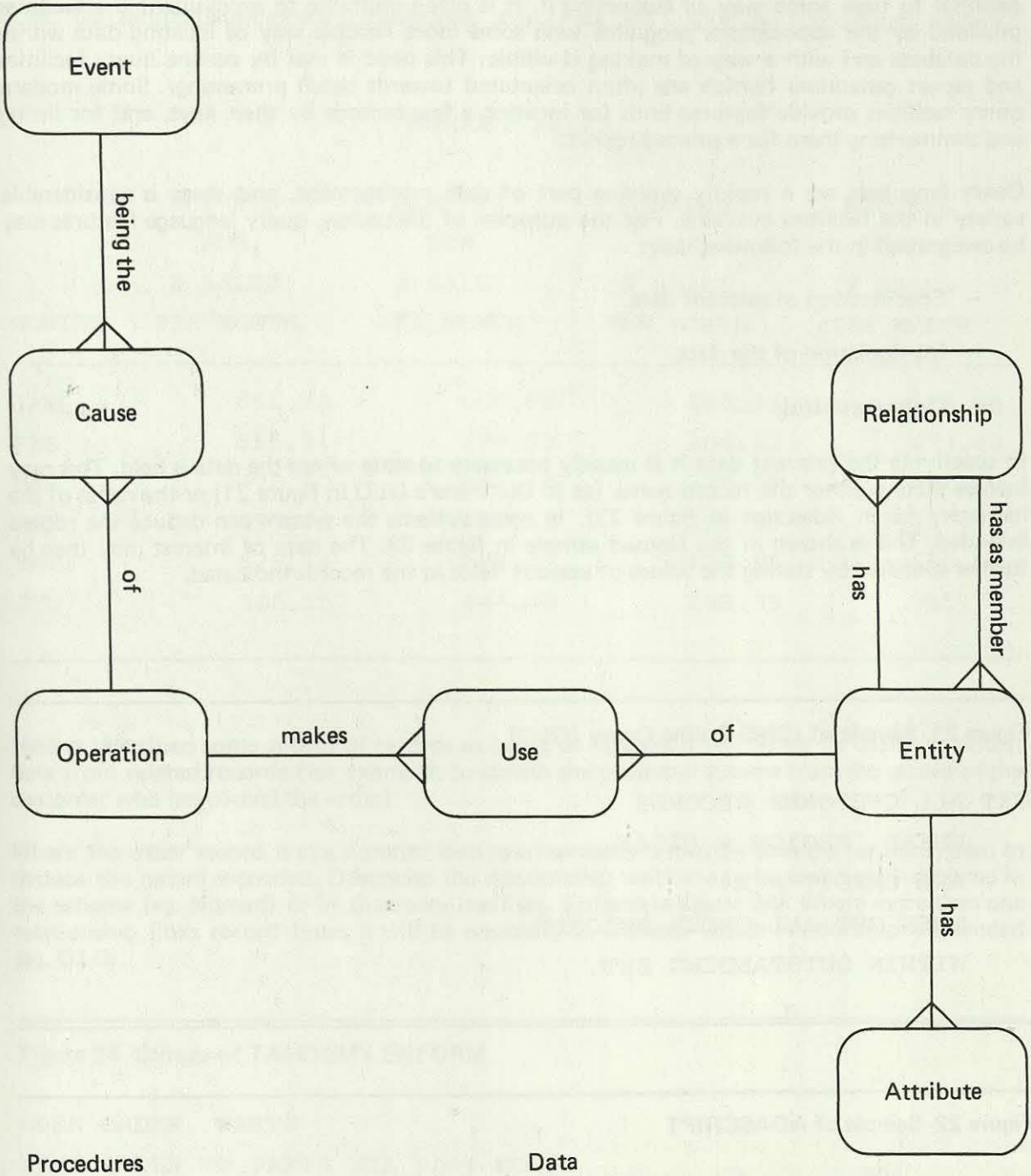
During design, representations will be chosen for the entities, attributes, relationships and conditions in the conceptual model. The dictionary should record the links between the conceptual and the implementation models.

During implementation, schemas and subschemas can be generated from the dictionary to save the labour of transcription and also to enforce naming standards, etc.

After implementation, the dictionary serves as documentation for maintenance and data standardisation work.

It is highly desirable that all systems, and not just database systems, should be recorded in the dictionary. Experience shows, however, that if this task requires programmers to prepare separate documentation it will not get done. To be used in this way the dictionary must be able to capture the necessary data from source programs. Better still, the dictionary should be able to generate both source programs and printed documentation from the same dictionary

Figure 20 Elements of the Entity Data Model used in the Conceptual Part of DDS 2900



entries.

The basic reason for using a data dictionary is that there can be no central view of data without it, and without such a view there can be no administration of that data. The data dictionary is an essential tool of the data administration function, the role of which is discussed at greater length in section III.A.

C Query Languages

Once a substantial part of an enterprise's data has been entrusted to the database, it becomes essential to have some way of inspecting it. It is often desirable to supplement the facilities provided by the applications programs with some more flexible way of locating data within the database and with a way of making it visible. This need is met by on-line query facilities and report generators (which are often orientated towards batch processing). Some modern query facilities provide features both for locating a few records by their keys, and for listing and summarising them for a printed report.

Query languages are a rapidly evolving part of data management, and there is considerable variety in the facilities available. For the purposes of discussion, query language features may be categorised in the following way:

- Specification of relevant data.
- Manipulation of the data.
- Output control.

In specifying the relevant data it is usually necessary to state where the data is held. This may involve stating either the record name (as in Cullinane's OLQ in figure 21) or the name of the hierarchy (as in Adascript in figure 22). In some systems the system can deduce the record intended. This is shown in the Nomad sample in figure 23. The data of interest may then be further specified by stating the values of various fields in the records indicated.

Figure 21 Sample of IDMS Online Query (OLQ)

```
GET ALL CUSTOMER RECORDS
    WHERE (REGION = 01545
    AND CREDIT = X)
    THEN GET ALL ORDER RECORDS
    WITHIN OUTSTANDING SET.
```

Figure 22 Sample of ADASCRIPT

```
FIND ALL RECORDS IN PERSONNEL-FILE
    WITH EDUCATION = LAWYER OR ENGINEER
    AND AGE FROM 40 THRU 60
AND SORT BY NAME
AND DISPLAY NAME, EDUCATION, AGE, SALARY.
```

Figure 23 Sample of NOMAD

```
list by months across prodname sum (sales) title 'product sales'
```

PRODUCT SALES

MONTHS	BLIVETS SUM $ SALES PER MONTH	JARVERS SUM $ SALES PER MONTH	LINKERS SUM $ SALES PER MONTH	WIDGETS SUM $ SALES PER MONTH
JAN	651.38	400.68	168.23	534.90
FEB	556.91	208.59	204.12	473.40
MAR	470.10	442.18	278.08	581.20
APR	497.56	161.11	253.55	1,327.45
MAY	576.14	282.21	193.95	688.53
JUN	316.16	444.48	243.36	885.77

Having identified some subset of records as being of interest it will often be useful to obtain data from related records (for example, to obtain the customer's name from the record of the customer who has placed the order).

Where the other record is at a superior level in a hierarchy it may be possible for the system to deduce the record intended. Otherwise the relationship will have to be specifically declared in the schema (eg. Nomad) or in the query itself (eg. Enform in figure 24). Where more than one relationship links record types it will be necessary to indicate which relationship is intended (eg. OLQ).

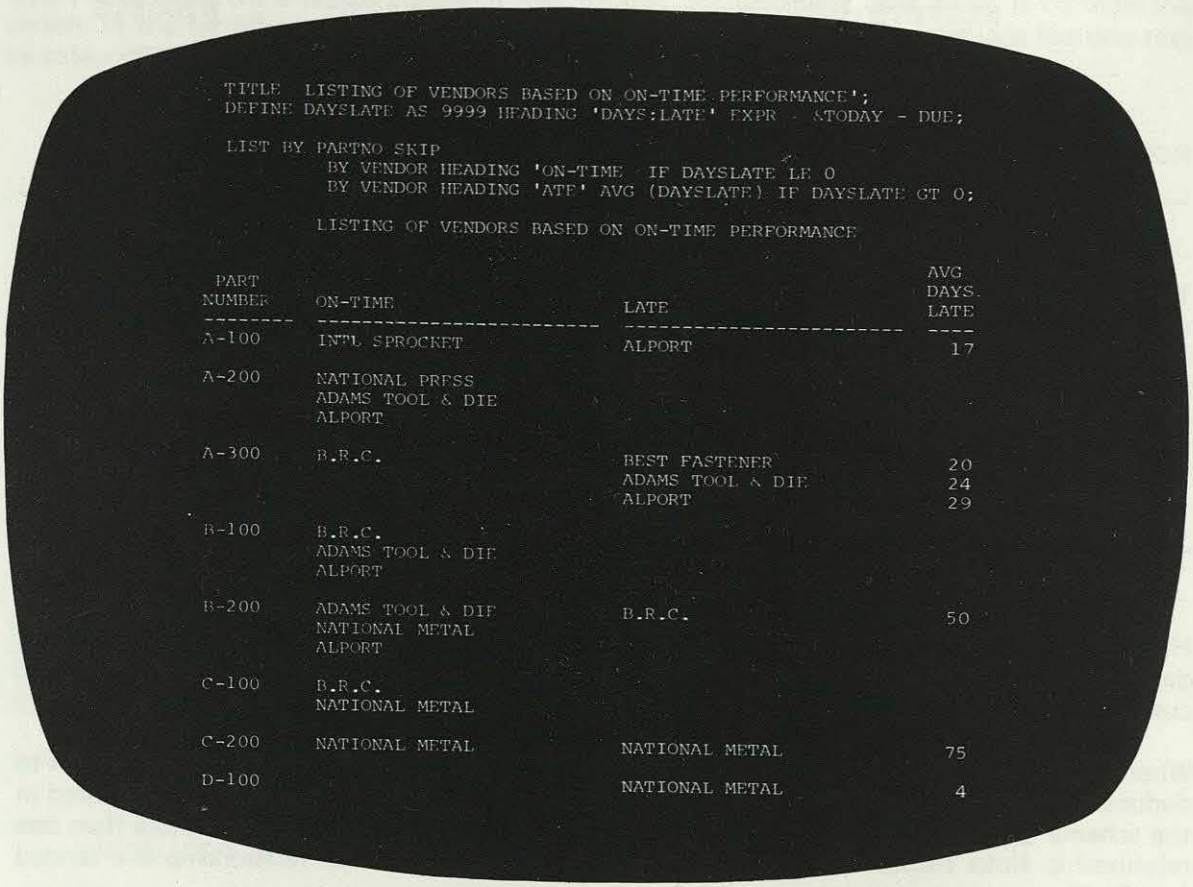
Figure 24 Sample of TAMDEM's ENFORM

```
OPEN ORDER, PARTS
LINK ORDER TO PARTS VIA PART-NO
LIST BY ORDER-NUMBER,
    CUSTOMER, PART-NO, QUANTITY, COST
    (QUANTITY*COST) HEADING "TOTAL-COST",
    WHERE CUSTOMER = JONES
    SUBTOTAL;
```

When the user has obtained the data he may wish to analyse it. A good query language should allow new quantities to be calculated from those stored, using simple arithmetic (as shown in the Enform example in figure 24). It should also provide statistical functions.

It should also be possible to specify the data of interest using the statistical functions (as shown in the second Nomad sample in figure 25).

Figure 25 Second Sample of NOMAD



```
TITLE  LISTING OF VENDORS BASED ON ON-TIME PERFORMANCE';
DEFINE DAYSLATE AS 9999 HEADING 'DAYS:LATE' EXPR  &TODAY - DUE;

LIST BY PARTNO SKIP
        BY VENDOR HEADING 'ON-TIME  IF DAYSLATE LE 0
        BY VENDOR HEADING 'ATE' AVG (DAYSLATE) IF DAYSLATE GT 0;

        LISTING OF VENDORS BASED ON ON-TIME PERFORMANCE

 PART                                                          AVG
NUMBER   ON-TIME                    LATE                      DAYS.
--------  ------------------------  ------------------------  LATE
A-100     INTL SPROCKET             ALPORT                     17

A-200     NATIONAL PRESS
          ADAMS TOOL & DIE
          ALPORT

A-300     B.R.C.                    BEST FASTENER              20
                                    ADAMS TOOL & DIE           24
                                    ALPORT                     29

B-100     B.R.C.
          ADAMS TOOL & DIE
          ALPORT

B-200     ADAMS TOOL & DIE          B.R.C.                     50
          NATIONAL METAL
          ALPORT

C-100     B.R.C.
          NATIONAL METAL

C-200     NATIONAL METAL            NATIONAL METAL             75

D-100                               NATIONAL METAL              4
```

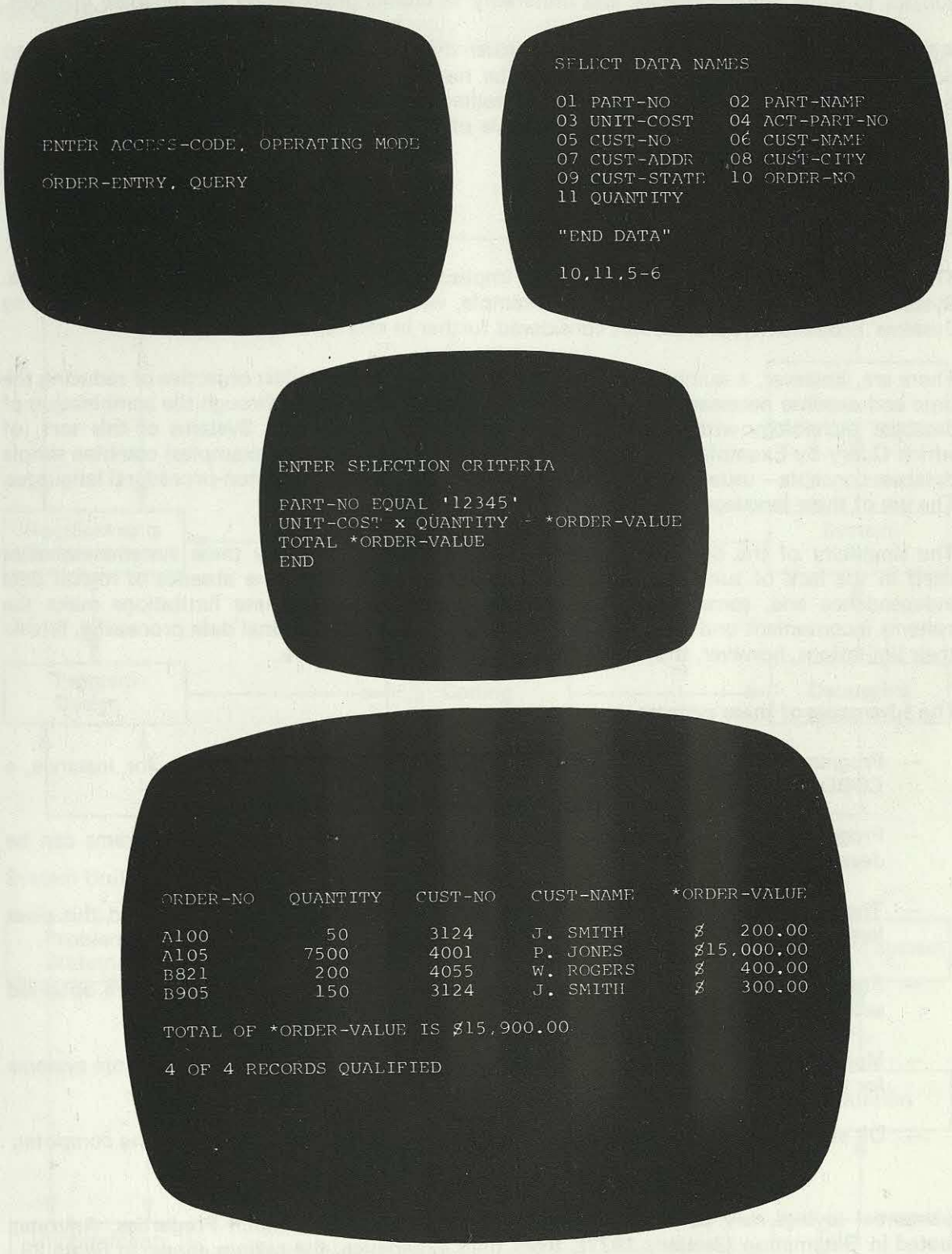
When the required analysis has been obtained, it should be possible to inspect it on a VDU and, if required, to have it printed. It should also be possible to fit printed output onto paper of various sizes. Existing languages provide a variety of features in this area.

A good query language should provide editing facilities, including table look-up at the output stage. Nomad, again, provides some interesting features (as shown in figure 25).

Output in the form of graphs and histograms is very convenient for some purposes, and the query facility may provide this as an alternative to tables.

In general, non-procedural languages are to be preferred to procedural ones, because they are easier both to understand and use. Non-procedural query languages can be very compact, and, for instance, the length and complexity of the COBOL program equivalent to the Nomad query shown in figure 23 illustrated this. Non-procedural query languages are also relatively easy to maintain.

Figure 26 Sample of T—ASK



Most query facilities allow complex queries to be stored and invoked with parameters. This allows the data processing department to provide users with tailored reports more quickly, more cheaply, and more flexibly than conventional languages do. Queries should certainly be subject to some privacy checks, and preferably to checks provided by the database manager.

All the languages discussed above assume either that the user knows enough to enable him to construct his own queries, or that he can be restricted to queries that others define on his behalf. Alternatively, a dialogue can be conducted between the system and the user to establish his needs. Viewdata provides one example of this, and Cincom's T—ASK, shown in figure 26, provides another.

D System Building Tools

The view of data as an entity in itself also implies a new view of data processing as a whole. Systems based on this view, ISDOS for example, were discussed in Report No.11: Improving Systems' Productivity, and are not considered further in this report.

There are, however, a number of DBMSs that have the more modest objective of reducing the time and expense necessary to implement data processing systems through the combination of database technology with high level, data manipulation languages. Systems of this sort (of which Query-By-Example, Ramis, Mark IV, Nomad and MRDS are examples) combine simple database concepts — usually based on hierarchies — with sophisticated non-procedural languages. The use of these languages for queries has been discussed earlier.

The simplicity of the database concepts that are typically used by these systems expresses itself in the lack of tuning options for data storage and access, the absence of logical data independence and, sometimes, the lack of a central schema. These limitations make the systems inconvenient and expensive to use for high-volume operational data processing. Within their limitations, however, they present a very attractive alternative.

The advantages of these systems are:

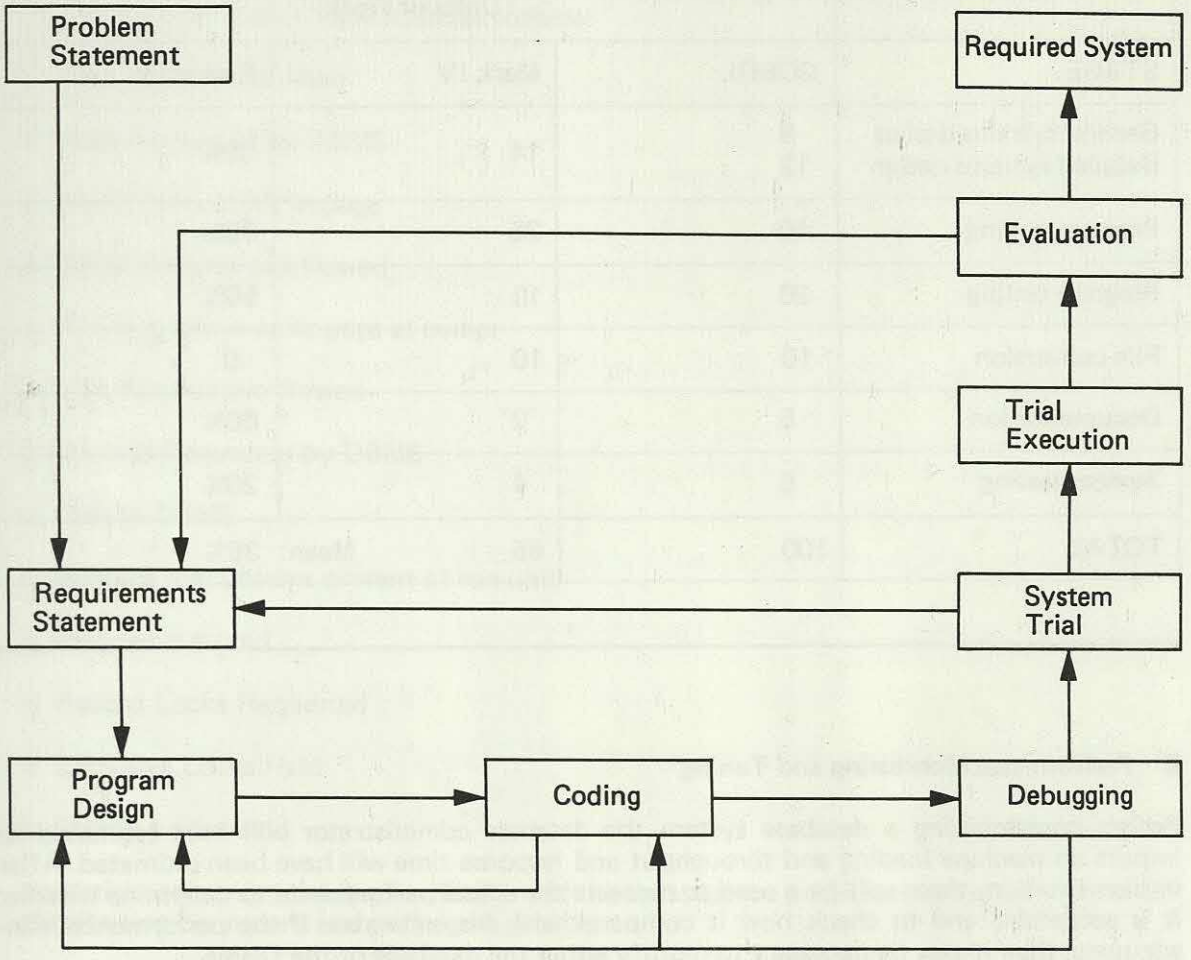
- Programs in higher level languages can be discussed with users as, for instance, a COBOL program cannot.
- Programs can be written faster and more easily. This means that programs can be developed for jobs that would not be worth doing with conventional tools.
- The implementation cycle is greatly shortened (as shown in figure 27), and this gives less opportunity for both misunderstanding and delay.
- Faster implementation means that the benefits of the resulting system are obtained sooner.
- More rapid implementation enables the data processing staff to produce more systems for users.
- DP staff become orientated more towards the business and less towards the computer.

Substantial savings may be obtained. Flynn and Kimber of McCulloch Properties, Arizona, quoted in Datamation (January 1977), from their experience, the savings shown in figure 28.

The principal barrier to the widespread adoption of these systems appears to be the difficulty that DP staff have in believing the claims that have been made for them.

Figure 27 Development Cycles

Conventional Programming



System Building

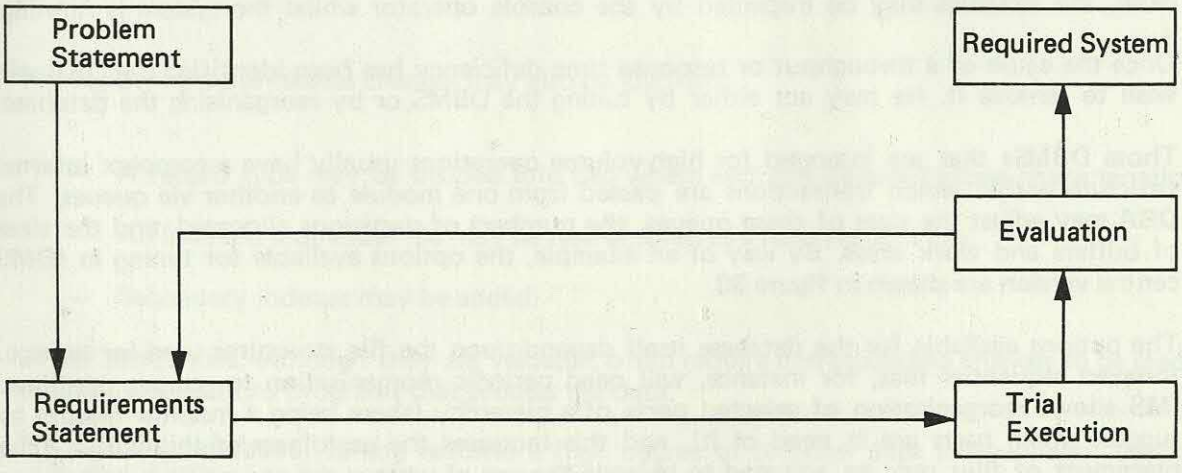


Figure 28 Savings Achieved with Mark IV

	Units of Work		
STAGE	COBOL	Mark IV	Reduction
General systems design Detailed systems design	8 12	14	30%
Program coding	40	25	38%
Program testing	20	10	50%
File conversion	10	10	0
Documentation	5	2	60%
System testing	5	4	20%
TOTAL	100	65	Mean: 35%

E Performance Monitoring and Tuning

Before implementing a database system the database administrator will have estimated its impact on machine loading and throughput and response time will have been estimated. After implementation, there will be a need to measure the actual performance to determine whether it is acceptable and to check how it compares with his estimates. If the performance is inadequate, then it may be necessary to modify either the database or the DBMS.

DBMSs usually collect some statistics internally. Others may be obtained from an analysis of the log. (The statistics collected by the IBM version of IDMS are shown in figure 29). In some cases, the statistics may be inspected by the console operator whilst the system is running.

Once the cause of a throughput or response time deficiency has been identified the DBA will wish to remove it. He may act either by tuning the DBMS or by reorganising the database.

Those DBMSs that are intended for high-volume operations usually have a complex internal structure within which transactions are passed from one module to another via queues. The DBA may adjust the sizes of these queues, the numbers of partitions allocated, and the sizes of buffers and work areas. By way of an example, the options available for tuning in IDMS central version are shown in figure 30.

The options available for the database itself depend upon the file structures used for storage. Indexed sequential files, for instance, will need periodic reorganisation to remove overflow. IMS allows reorganisation of selected parts of a hierarchy (there being a monitor feature to suggest which parts are in need of it), and this increases the usefulness of this option. The placement of files may be adjusted to balance the use of various drives, and it is sometimes possible to split a highly active file across several drives.

Figure 29 IDMS Statistics

"The bulk of the available statistics are accumulated whilst IDMS is running and are available during program execution or alternatively they may be obtained from the log file at a later date. These statistics include:

- o Pages Read and Written
- o Pages Requested by DBMS
- o CALC Records hit in page
- o CALC Records overflowed
- o VIA Records in same page as owner
- o VIA Records overflowed
- o Records Requested by DBMS
- o Calls to DBMS
- o Records that became current of run unit
- o Fragments stored
- o Record Locks Requested
- o Exclusive Locks Held
- o Shared Locks Held

Statistics concerning the space utilisation of the database are provided both space distribution per record and per page is given".

(Reproduced by courtesy of Scicon).

The following more radical changes may also be justified:

- A record type may be divided into two where each has different access characteristics.
- An indexed sequential file may be restructured as a hash random file.
- Secondary indexes may be added.

These techniques, although they are valuable, often require corresponding changes to be made in those applications programs that process the data.

To be fully effective, tuning requires a high degree of physical data independence behind which the DBA can optimise the storage schemas for optimum performance. Improvements in the understanding of tuning, however, will allow much of the DBA's role in this to be auto-

Figure 30 IDMS Central Version Tuning Options

- Maximum number of run-units that may sign on to IDMS–CV concurrently
- External wait time – time that CV will allow between database calls from a run-unit before aborting that run unit. Values can be set for specific programs or as a global default.
- Internal wait time – time that CV will wait for a database resource (DB-key, Area etc) requested by a run unit before aborting that run init. Values can be set for specific programs or as a global default.
- Check times – frequency with which the resource controller checks each run unit for abend condition.
- Maximum number of subschemas and database procedures that can be loaded during each CV session.
- Size of the program pool where non-resident subschemas and database procedures are loaded.
- Size of the storage pool where DMCL buffers, database-key locks etc. are held.
- Number of database keys that can be locked by a single run-unit.
- Number of database keys that can be locked by all run units at one time.
- Number of times a run-unit can be pre-empted for a resource.

mated. Indeed, those IMS products that select parts of an ISAM file, and then reorganise them, may be seen as a step in that direction. Existing relational systems normally perform a variety of optimisations automatically.

Tuning is a vital function for the DBA, and it may easily make the difference between a successful system and an unworkable one. In the longer term, however, increased automation of the tuning function and falling hardware costs will make this a progressively less attractive area for the application of scarce technical skills.

VII. DATABASE AND NETWORKING

On-line access and database management are both major trends in data processing. But they are trends that often lead in opposite directions. The rest of this section identifies the conflicts and discusses ways of dealing with them.

A Remote Access to Data

Most major companies now provide at least some of their staff with on-line access to corporate data. Those that do not are mostly building the necessary systems. It is quite common for the various systems in an enterprise to use incompatible hardware, software or telecommunications methods.

In an attempt to rationalise the situation, or to avoid proliferation, some companies have concentrated their data processing on a central facility. Systems integration has pushed in the same direction, and the mainframe manufacturers have not been slow to stress these factors. Centralisation has also had other benefits, such as the rapid availability of engineering and programming support after a crash.

Organisations that have centralised have, however, found the following disadvantages:

- When the users are located at several sites there are high communications costs.
- Processing power costs more than a cluster of smaller machines would. It seems that Grosch's law no longer applies.
- If there is a single central machine it never has quite enough power. Consequently, either effort has to be put into complex optimisation, or useful jobs have to be kept off the machine, or more hardware has to be purchased.
- User departments feel that they are losing control of their systems.
- Throughput is limited by contention for the database.
- The database is very vulnerable to accident, sabotage, and industrial action.
- The high data volumes, the high transaction rates, the 'domino effect' consequences of a failure and the inherent complexity of the large mainframe require extra work and expense to ensure reliability.

On-line access to a single database may be obtained using standard software. Standard solutions are normally adequate, but they have their difficulties. Moreover, improvements in hardware, software and communications services will make this increasingly straightforward.

Standard solutions can also handle the following configurations:

- A number of sites have their own copies of the database they refer to exclusively. This approach is very suitable for systems that require retrieval only.

- The data is divided, without duplication, between a number of sites, each of which may update its local data only.

- The data is divided, but although duplication exists, the duplicate copies do not need to be kept aligned from moment to moment. It is satisfactory to align the duplicate copies either each evening or at the end of the week.

Until recently, any further interconnection would require the writing of special message-switching software. However, the network architectures announced by IBM, Tandem, DEC, etc. will now allow access to data on both local and remote machines, provided that either the user or the programmer knows which particular machine or which particular database he wishes to access. The network in these products provides message services, but it does not provide any integration between databases.

For many practical cases these facilities are adequate. And if the network is also used to move batches of updates round the system, data need never be more than a few hours out of date.

Companies who need real-time updating of duplicated data will still have to write their own telecommunications software. (One example of this, the distributed network system of the Swiss Credit Bank, was described to the London Management Conference of the Butler Cox Foundation in April 1978).

This composite approach has the following disadvantages:

- It requires staff who have telecommunications skills, and staff with these skills may not be readily available.

- It requires extra procedures when recovering from both line and machine failures.

- It is expensive, and this restricts it to large enterprises or to those organisations (banks for instance) to whom the functions are absolutely vital.

- It distracts attention from business problems.

The question of a true distributed DBMS is discussed further in part C.

B On-line Interrogation and Problem-Solving

Interrogation and the running of analysis programs against the database form an increasingly large element in database processing. It is an element that differs from simple transaction processing in the following ways:

- Its volume and its nature are difficult to predict, especially if the system is to be used to support non-recurring managerial decisions.

- Processing requirements are high, especially if engineering calculations or graphics are involved.

- Data access requirements are high. A simple request to report all orders that are more than five days overdue may result in thousands of accesses. By contrast, a guideline is sometimes given that no more than thirty disc accesses should be made per transaction.

- The needs of the users are very variable. One executive may be an occasional user who requires more help in formulating his request than does a regular clerical user. Another may be an experienced APL programmer who wishes to run a financial model against the database.

- In the absence of a true distributed database, the user may need to access several different databases in different locations, and those databases may, perhaps, use different DBMSs.

- The user may need to access data from a public database and to integrate it with corporate data.

1. **The problem of variety**
In a large company, several languages are likely to be needed for different types of user. Query languages were discussed in section VI.C, but it should be mentioned here that most proprietary languages can access data at only one location at one time. An exception to this is Tandem's Enform which can access any number of databases if the user knows their locations.

Where a network architecture exists it may be possible to make several databases look like one to the query facility.

Access to several databases may be obtained from a programming language, but the work necessary to do this may be considerable. On the IBM 370, for instance, APL cannot access IMS databases at all and COBOL programs may access only one IDMS database in a run. The necessary open access can be obtained however. For example, on Multics (that is, Honeywell Level 68) all languages have access to two database systems, IDS2 and Multics Relational Data Store (MRDS). Also, IBM and MIT have developed an experimental system called General Management Information System (GMIS) under which a variety of languages and statistical systems can call data from a variety of DBMSs.

These systems apart, it is usually easier to extract the required data and to re-format it for the language or the utility that is to be used than it is to find a general solution. The more users, languages, and utilities there are, the greater will be the overhead of this approach. It has the advantage, though, of also solving the resource contention problem discussed below.

No current commercial system provides access to remote databases of different kinds, but several could have such facilities added by skilful software programming.

In the long term the solution lies in improved logical data independence in the DBMS. One interesting development that is relevant here is ICL's decision to provide a subschema option in 2900 IDMS which will allow a program or a utility to treat part of the database as a file (that is, without using the data manipulation language). This will allow file-based utilities and applications programs to be applied unchanged to the database, and this is a valuable facility.

2. **The problem of performance**
The size and the unpredictability of both the processing and the access load makes it difficult to provide reasonable response to *ad hoc* enquiries without ruining the response to those clerical users who are performing operational transactions. Fundamentally, this difficulty is the result of limitations in the basic design of present DBMSs and TP monitors, and therefore the best solution will lie in improving these products. Short of that, the following partial solutions exist:

- The required data may be extracted to a separate system using an extract utility running at low priority. The separate system can then be dedicated to the required manipulations. One computer aided design centre at Cambridge is developing a manufacturing system in which a separate processor performs graphics and engineering calculations. The same principle is also used for text processing on the Xibus.

- A query facility with its own priority system may be chosen. ASI's Inquiry runs long

queries at a much lower priority level than is given to short queries in order to limit their impact on the system.

- Long queries may be performed as overnight batch tasks, provided that the system can recognise them and store them appropriately.

- The required data may be extracted by an overnight batch run. It can then be made available to a suitable utility or language system running as a timeshared program.

- Long transactions may be split into several short ones by the programmer by issuing a reply to the user or by having the transaction suspended. This is practical of course only when the transaction is programmed by the DP staff, rather than by the user.

In the long term these problems will be dealt with by improvements to those scheduling systems that DBMSs use, and also by associative hardware to speed accesses to the data.

C Distributed Database

A distributed DBMS is one that supports a single unified view (a conceptual schema) of data that is physically distributed between a number of processors at different locations.

The storage of any particular type of data may be:

- Localised. This is appropriate for corporate level financial data and for manufacturing data where there is only one factory.

- Replicated. This is appropriate for a price list that must be available at several sales offices.

- Partitioned. This is appropriate for customer accounts that are to be held at the responsible sales office, or for manufacturing data where there are several factories.

A simple example is given in figure 31.

It is the task of the distributed DBMS to route accesses either to the correct location, when the data is held only once, or to the cheapest (which is usually the nearest) location when several copies are held. The distributed DBMS must also ensure that updates are applied to all copies of replicated data. In the interests of data independence changes in placement should be invisible both to the user and the application programmer, and this means that the location should be specified through the storage schema.

A fully distributed database has the following advantages over a centralised one:

- Reduced communications costs, because the data that is referred to most frequently is stored locally.

- Reduced dependence on a single complex mainframe.

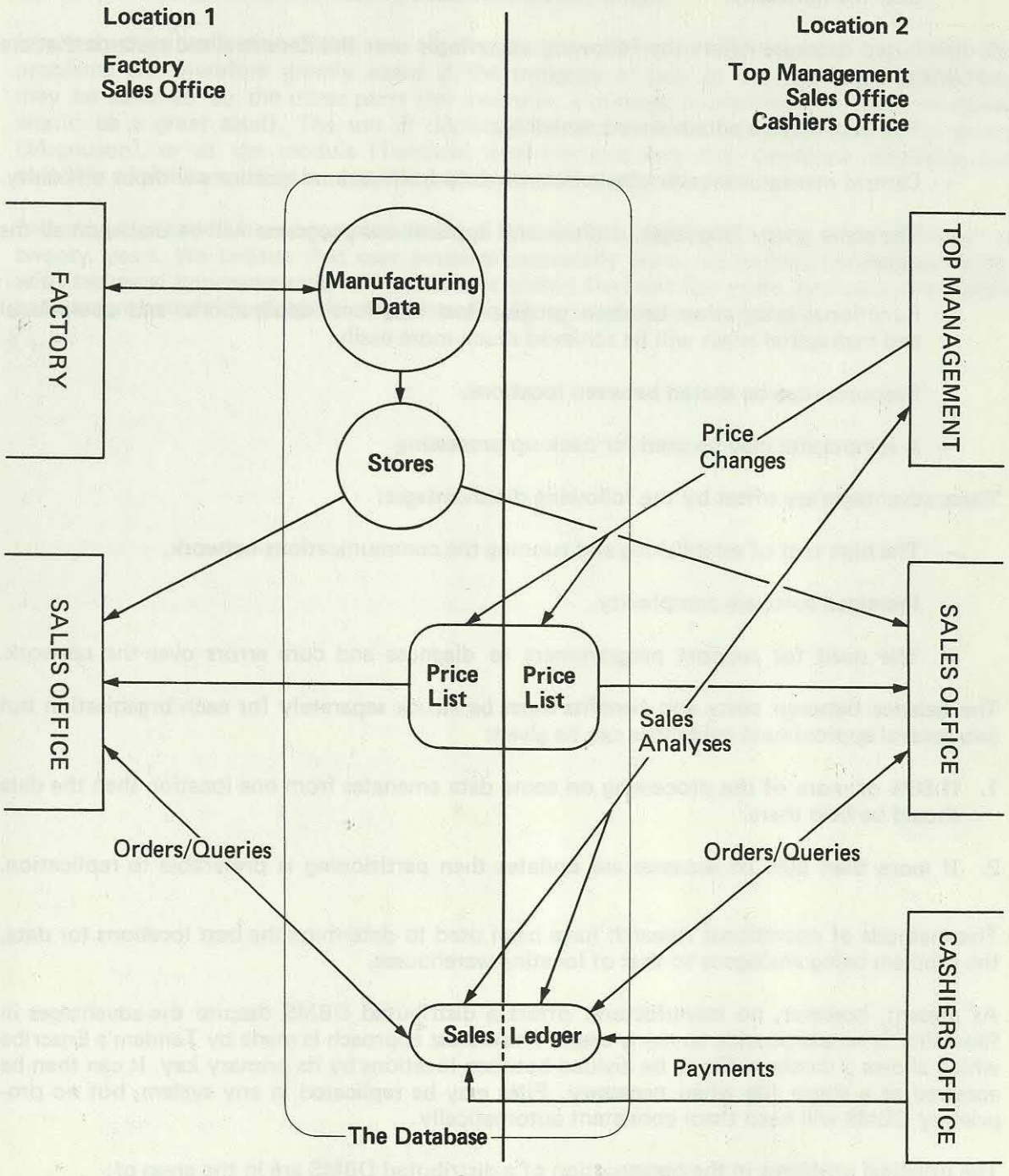
- The ability to tailor the facilities at each location to local needs.

Potential disadvantages of a fully distributed database include:

- Increased software complexity.

- Extra processing in order to maintain consistency.

Figure 31 A Simple Example of a Distributed Database



Manufacturing Data: Centralised at Factory
Stores Data: Centralised at Factory
Price List: Replicated at Sales Offices
Sales Ledger: Partitioned between Sales Offices

Arrows show principal information flows between the functions shown and the database.

- The need to replicate hardware, particularly electromechanical devices.

- The need to diagnose and cure faults in machines at remote sites (which may be done over the network).

A distributed database offers the following advantages over the decentralised systems that are not linked together:

- The data may be administered centrally.

- Central management can obtain data directly from various locations without difficulty.

- The same query languages, utilities and applications programs will be usable on all the data.

- Functional integration between geographical locations, applications, and operational and managerial levels will be achieved much more easily.

- Resources can be shared between locations.

- A remote site may be used for back-up processing.

These advantages are offset by the following disadvantages:

- The high cost of establishing and running the communications network.

- Increased software complexity.

- The need for support programmers to diagnose and cure errors over the network.

The balance between costs and benefits must be struck separately for each organisation but two general approximate guidelines can be given:

1. If 80% or more of the processing on some data emanates from one location then the data should be held there.

2. If more than 50% of accesses are updates then partitioning is preferable to replication.

The methods of operational research have been used to determine the best locations for data, the problem being analogous to that of locating warehouses.

At present, however, no manufacturer offers a distributed DBMS despite the advantages in flexibility it would provide to many users. The closest approach is made by Tandem's Enscribe which allows a database file to be divided between locations by its primary key. It can then be accessed as a single file when necessary. Files may be replicated in any system, but no proprietary DBMS will keep them consistent automatically.

The principal problems in the construction of a distributed DBMS are in the areas of:

- Interfacing dissimilar machines, although this is eased if a single vendor supplies both machines and software.

- Maintaining integrity, especially after a hardware or program failure. If a failure leaves updates partially completed at a number of locations, and one or more of these has failed, the updates must be backed out of the database. This will require separate recovery actions at each location, and these must be synchronised.

- Avoiding deadlocks. This problem is familiar with centralised DBMSs but the optimum solutions may be different in the distributed case due to the increased time and cost and the lower reliability involved in network accesses. In particular, methods aimed at preventing rather than curing deadlocks are preferable.

In short, the distributed DBMS has all the worst problems of database and networks. The problems are therefore greatly eased if the integrity of part of the system is assured and may be assumed by the other parts (for example, a message mechanisms of 100% reliability would be a great asset). The use of duplicate hardware at the circuit (IBM), at the board (Magnuson), or at the module (Tandem) level can give very high hardware reliability, but software of similar quality is currently possible only for very simple tasks.

Industry spokesmen have suggested that full distributed DBMS will not be available for ten to twenty years. We believe that user pressure (especially from US multinationals), combined with technical improvements, will produce it within the next five years. And once one vendor has demonstrated this capability the pressure will be on the other to follow.

VIII. INDUSTRY STANDARDS

Industry standards for DBMSs will be of greatest importance for those enterprises that anticipate transporting their applications from one computer system to another, although they may find that a commercial product is available on both the computer systems. Industry standards will, however, be of general value if they bring greater stability to the DBMS scene. In this section we review briefly the most important standards activity under way at the moment and comment on its significance.

A An American Standard for DBMS

The American National Standards Institute (ANSI) has voted to produce database standards based on the Codasyl proposals, and the work is expected to be complete within the next five years. There will be standards for three languages and, although some slippage will probably occur, the target dates are as shown in figure 32.

Figure 32 The American National Standards Institute Database Standards

Language	Target Date	Remarks
COBOL Data Manipulation Language	1980	The DML verbs will be included in the next standard, due in 1980.
FORTRAN Data Manipulation Language	1983	The DML verbs will be in the next standard but this must be at least five years on from the present FORTRAN standard, which was 1978.
Data Description Language	1983	DDL is the biggest of the three jobs and standards-making work is slow.

Codasyl is currently working on a Data Storage Description Language (DSDL); that is, a language for the storage schema. Because it is desirable to separate the physical and the logical aspects of the schema a DSDL standard is likely. However, it will probably not be produced until at least the mid-eighties.

The importance of a standard derives, not from its inherent merits, but from the likelihood that it will become available on a wide range of machines. Since Burroughs and IBM (and especially IBM) are strongly opposed to the Codasyl database proposals, the new standard will become generally available only if the US government insists (as it did with COBOL) that every commercial machine it buys shall support it. However, the momentum behind standardisation is, we believe, now sufficient to put the promulgation of a standard beyond doubt, no matter what IBM may do by way of a relational DBMS.

Even without the US government's intervention, vendors will make 'standard' DBMSs available on most commonly used types of hardware. European users who feel suspicious of the likely quality of the standard should note that Codasyl is open to non-US members, and also that groups in this country have already made valuable contributions to its activities.

B Other Standards

Other standards work relevant to database includes the definition of a common information retrieval language for Euronet, and the attempts both by an ANSI/SPARC working group under Charles Bachman and by a committee of the International Standards Organisation to develop a common framework into which all computing will fit. Neither seems likely to be very influential in the near future.

C The Implications of the DBMS Standard

The advantage of using any standard product arises less from the inherent merits of the standard than from the possibility of portability. Choosing a Codasyl DBMS thus has the following advantages:

- Experienced staff are easier to get, although this is also true of any IBM *de facto* standards.
- A change to a very different type of hardware is eased.
- A change to a different, but compatible, DBMS is possible.

On the other hand, these considerations are less than overwhelming at present because:

- There will be no standard until 1983, and the standard will then be different both from the present Codasyl proposals and the state of the present products.
- The ancillary software, query languages, data dictionaries, and utilities differ between standard products, and tend to lock in the user. Avoiding this involves a tough-minded attitude to vendors and may involve more trouble than it is worth.
- IBM may produce a hardware-supported relational DBMS on the long-awaited H series that is so flexible as to make the alternative DBMSs obsolete overnight. This would be contrary to IBM practice, and there is some doubt as to whether it is technically possible. Nevertheless, it might happen. If it did happen, IBM could have substantial difficulty in transferring their IMS users to the new system, and so IMS is not especially attractive as an interim solution.

The choice of a DBMS remains a matter of judgment in a complex market. A Codasyl DBMS is strongly indicated only if a major hardware change is anticipated.

IX. FINDINGS AND RECOMMENDATIONS

A Findings

The principal findings of our research are:

1. Database management systems have potential for alleviating the backlog of work that now confronts most system development departments. Query languages and system building tools, which are becoming elements in most suppliers' product ranges, will allow at least some end-users to develop and enhance systems for themselves.

   This trend has implications for the future role of management services. It will tend to change the present role from one of constructing systems to one of advising and providing tools for end users.

2. Users' experience with database management has been favourable. In almost all cases, users have achieved the benefit that they expected to achieve and, in many cases, their achievement has exceeded their expectations. This experience indicates that, if the introduction of a DBMS is well planned, the DBMS is most likely to be successful and produce worthwhile benefits.

3. Until at least the mid-1980s the proprietary DBMS will exert a centralising influence on data processing. This influence conflicts with most other influences which are pushing organisations towards dispersing both equipment and responsibility.

   Those organisations that wish to take advantage of the benefits of both distributed processing and of DBMSs in the near future will necessarily have to pioneer. This pioneering effort will need to be in two areas:

   - Data administration. The planning and control of the physical location of data will add a new dimension to the data administration function.

   - Software. Network functions concerned with the integrity of the database etc will need to be developed. Clearly, highly-skilled people will be needed for this task.

4. Much of the attraction of the DBMS lies in the ability it provides to integrate systems that previously were separate. This attraction stems from the potential benefits that an organisation can obtain from the new facilities which come with integration. However, integration does pose some potential problems because reliability, recovery, and complexity of implementation become increasingly important. The trade-off between the potential benefits of integration and the possible costs associated with failure needs to be made consciously by the management of an organisation.

5. A separate database administration (DBA) function is vital if a DBMS is to be used effectively. The software is too complex to expect every project team to master. In the longer term, however, the DBA role will decline. As database becomes a standard data processing technique, all analysts will acquire the skills needed to recognise the cases where a DBMS

is appropriate and to design a database. Improved interfaces and greater data independence will reduce the amount of technical detail that the applications programmer must master. Tuning will be performed by software, and the falling cost of hardware will make tuning less useful anyway. Similar trends will affect the DBA's other responsibilities.

6. Many large organisations are now turning their attention to systems for planning, control and decision support which can be built on top of functional systems. DBMSs clearly enable organisations to exploit such opportunities. However, if these new systems are to be effective they may require skills (for example, in management science) which are not commonly found in management services.

7. The database approach involves much more than merely using a DBMS. It involves recognising the importance of data as a corporate resource and treating it accordingly.

   An organisation that wishes to adopt the database approach does not necessarily have to use a DBMS. (Conversely, an organisation that wishes to use a DBMS does not necessarily have to adopt all the principles of data management discussed earlier.)

   Central to the database approach is the recognition of the need for data administration, and the need to include data analysis as a necessary part of the process of systems analysis.

   If the concept of 'data as a corporate resource' is to be implemented successfully, it needs to be understood both by management services staff and by line management. Unless they both understand the principles concerned, it is most unlikely that the inevitable conflicts of interests will be successfully resolved. This poses a challenge to management services, since it requires selling to line departments a concept to which they are unlikely to be well disposed.

8. With a few exceptions the database approach is, in our view, the preferred future approach. We believe that those organisations that are not currently employing a DBMS should concentrate effort on when and how to move towards the database approach. Adopting the database approach involves more than just selecting and successfully implementing a DBMS. Before the software tools are introduced all concerned should be educated in the concept involved and data administration and data analysis techniques should be introduced.

9. The relational model has received disproportionate attention. It should be regarded as an extremely valuable technique which is to be used in the process of data analysis. It is applicable to the analysis and the design of all systems and, in particular, will make any database more flexible and more easy to extend.

   The use of the relational model will help to minimise the effects of moving from one DBMS to another – and possibly one type of hardware to another. It also prepares for the use of relational DBMSs which can be expected to provide considerably more hardware assistance than current products provide.

10. The concepts associated with data analysis and data management are still fluid, and further understanding can be expected to lead to new and superior products. As in other fields, current suppliers will need to evolve new products if they are to protect their existing market base.

    This evolutionary process will lead some of those suppliers who have previously been associated only with software products into hardware supply. Equally, some hardware suppliers can be expected to add DBMS to their product range.

11. As hardware costs continue to fall exponentially, the life-cycle of the equipment is shortening. For most organisations the selected DBMS (or its derivative) is likely to outlast a

number of generations of hardware. The selection of a DBMS is, therefore, a particularly crucial factor in ensuring the success of future application systems. Our research indicates that those organisations that undertake a selection exercise in which they review the leading contenders do seem to achieve better results than those that do not do this.

12. Over the next few years, vendors will introduce products to complete the range of facilities that they offer. These ancillaries include data dictionaries, query languages and system building tools. Beyond that, new products will include support for relational and distributed databases.

13. Standards for DBMSs will emerge over the next few years. We expect an ANSI standard by about 1983, and we expect that this will be based on existing Codasyl proposals.

B Recommendations

Each organisation will have its own data processing strategy and will already have progressed some way in the field of data management. Our recommendations below will, therefore, apply differently to different organisations depending upon their individual circumstances. Some will be able to incorporate our recommendations in their future plans, whilst some will wish to use them to audit their achievements.

Our recommendations are:

1. Any data processing plan that covers the next three years or more should include the progressive adoption of the database approach. Clearly it is both sensible and necessary to proceed through the right pilot project. Such a project should be one that enhances the likelihood of success and one that poses the minimum threat in the event of failure or delay.

2. The database approach is at least as important as the DBMS tools.

   It is necessary to train systems analysts in data analysis skills and to incorporate data analysis in the standard method of the development department. These steps will yield immediate benefits irrespective of the adoption of a DBMS.

   The database approach also implies the separate administration of data. Data administration needs to be set up as a separate function that has the authority to arbitrate in any situation where the various interests of end-users and systems development staff are in conflict. In practice, full autonomy is likely to come from the recognition both by end-users and systems staff of the value of this approach, rather than by enforcement. Clearly, the function of data administration needs to be both encouraged and supported in its early days.

3. The DBMS is one of the most important elements in the data processing manager's tool kit. It will probably be with him longer than either the hardware and at least some of the software in his installation.

   The choice of the DBMS is a decision that will have a very considerable influence on the options available within data processing. It will, therefore, affect the quality of service which is provided to the organisation. This in its turn will affect the way in which management services is regarded.

   The choice, therefore, deserves both the attention of senior management and the use of the best possible resources. Those concerned with the selection need to be able to take a 'management' view as well as a technical view.

4. Query languages which can be used by end-users are now available. Other developments are providing end-users with ready access to the terminal facilities needed to use them.

   We recommend that management services investigate the use of query languages by end-users as one way of alleviating the shortage of skilled systems staff. Clearly, this involves not only selecting the right query facility but also identifying an end-user (or users) best equipped to use it. As with all other innovations, it is important to limit the scope of the implementation and to choose the best available circumstances.

5. The selection of new hardware must take account of the availability of good DBMS facilities – both those that are available now and those which can be expected in the future. It is likely that the quality of future products will be closely linked with:

   – The market base of the product.

   – The number of competing product ranges.

   In general, those who are using IBM or IBM-compatible processors are likely to have the widest range of alternatives available to them, but they will have the disadvantage that they need to choose between them.

6. A number of vendors now offer ancillaries that enhance the basic DBMS. We recommend that organisations when selecting data dictionaries, query languages, etc., consider products offered by vendors other than the vendor of the chosen DBMS.

7. Those organisations that wish to implement a fully distributed database have to choose between:

   – Waiting at least five years for suitable DBMS products to become available, and waiting for other organisations to succeed in implementing them.

   – Going it alone. Proceeding with a distributed database involves both new data administration responsibilities and (probably) designing and implementing software to ensure the integrity of the database.

   We recommend that organisations review carefully the available skills and experience in the relevant fields before embarking on such a venture.

C Footnote

In many respects, database management is still in its infancy. It will be of particular interest to see how it copes with distributed processing and with the convergence of data with other forms of information. In view of the significance which we attribute to the database approach, we believe that the Butler Cox Foundation should continue to monitor progress in database technology and experience. The Foundation should report again on the position within the next two years.

APPENDIX: A LIST OF DATA MANAGEMENT PRODUCTS MENTIONED IN THE TEXT

Name	Vendor(s)	Notes
Adabas	Adabas Software Ltd	DBMS
Adacom	Adabas Software Ltd	Batch report writer
Adamint	Adabas Software Ltd	Interface for programming languages
Adascript	Adabas Software Ltd	Query facility
Adawriter	Adabas Software Ltd	Report writer
Content Addressable File Store (CAFS)	International Computers Ltd	Specialised hardware
Culprit	Cullinane Inc	Report writer for IDMS
Distributed Array Processor (DAP)	International Computers Ltd	Specialised hardware
Database Design Aid (DBDA)	International Business Machines Ltd	Design aid for DL/1 database
Data Dictionary (DD)	International Business Machines Ltd Adabas Software Ltd	
Data Dictionary System (DDS)	International Computers Ltd	Runs on 2900 machines
Enform	Tandem Computers Inc	Query facility
Enscribe	Tandem Computers Inc	DBS
Data Language 1 (DL/1)	International Business Machines Ltd	Data Manipulation Language and subschema language for IMS. May also be used independently.
Generalised Information System (GIS)	International Business Machines Ltd	Query facility for IMS

Generalised Management Information System (GMIS)	International Business Machines Ltd	Developed by IBM and MIT. Not commercially available, it was described in ACM Translations on Database Systems, December 1976, 344-369
Integrated Data Dictionary (IDD)	Cullinane Inc	For use with IDMS
Image	Hewlett Packard Ltd	DBMS
Integrated Data Management Systems (IDMS)	Cullinane Inc	Codasyl DBMS
Integrated Data Management Systems (IDMS)	International Computers Ltd	Codasyl DBMS
Integrated Data Store (IDS)	Honeywell Ltd	Codasyl DBMS
Integrated Data Store (IMPS)	International Business Machines Ltd	Provides a relational view of an IMS database
Information Management System (IMS)	International Business Machines Ltd	'Hierarchical' DBMS
Inquiry	ASI	Query facility for IMS databases
Interactive Query Facility (QF)	International Business Machines Ltd	Query facility for IMS database
Liar	International Business Machines Ltd	Performance estimating tool for IMS systems
Mark IV	Informatics Inc	
Multics Relational Data Store (MRDS)	Honeywell Ltd	Relational DBMS available on Honeywell Level 68
Natural	Adabas Software Ltd	High level language for Adabas databases
Nomad	CSS International Ltd	Available as a bureau service or (in US) on NCSS 3200 mini
On-line Query (OLQ)	Cullinane Inc	Query language for IDMS database
Prompt	Theorem Series 1	System building tool
Ramis II	Mathematica Ltd	System building tool

Socrates	Cincom Systems International Ltd	Batch report writer
System R	International Business Machines Ltd	Full function relational DBMS. It is not commercially available though currently undergoing beta-trials with users in Europe and the US. System R is described in ACM Transactions on Database Systems (1976) (2) 97-137
T–ASK	Cincom Systems International Ltd	Query facility for Total database
Total Information System (TIS)	Cincom Systems International Ltd	DBMS – provides extra data independence for Total and other databases
Total	Cincom Systems International Ltd	DBMS
Query-by-Example (QBE)	International Business Machines Ltd	Relational DBMS
Query (1)	Hewlett Packard Ltd	Query language
Query (2)	Honeywell Ltd	Query language
Xibus	Xionics Ltd	

SELECTED BIBLIOGRAPHY

BERG, J.L. (ed): The Next Steps: National Bureau of Standards, 1976. Available in the UK from the British Computer Society.

BRITISH COMPUTER SOCIETY: Data Dictionary Systems Working Party Report; 1977.

CHEN, P.P.–S: The Entity-Relationship Model – toward a unified view of data: ACM Transactions on Database Systems (1976); 1 (1), 9-36.

DATA, C.J: An introduction to Database Systems; Addison-Wesley, 1975.

GILBERT, J.C: Can Today's MIS Manager make the translation? Datamation (1979); 24 (3), 141-51.

MARTIN, J: Computer Data-Base Organisation; Prentice-Hall, 1975.

NOLAN, J.R: Managing the crises in data processing; Harvard Business Review (1979); 57 (2), 115-126.

PALMER, I.R: Database System – A Practical Reference; CACI, 1975.

SENKO, M: Data Structures and data processing in database systems past, present and future; IBM Systems Journal (1977); 16 (3), 208-258.

WIORKOWSKI, G.K. and WIORKOWSKI J.J: Does a Data Base Management System Pay Off?: Datamation (April) 1978: 109-114.

YAO, S.F. et al: Database Systems; Computer (September 1978).

Abstract

Report Series
No12

Trends in Database Management Systems

by David Flint
June 1979

Trends in business data processing such as increasing user requirements and the convergence of technologies require the integration of data processing systems. Integration may be between operational and management information systems as well as across applications areas.

Integration requires a new approach to data processing – the database approach. Data must be seen as a corporate resource that should be understood and managed in its own right. Database technology is already used successfully by many businesses but it is most effective when used to support the database approach.

This report discusses the concepts underlying database management systems and explains their significance and likely impact on data management. The report also examines relevant trends in hardware and software and discusses the changing market in data management products.

Networking and standards are also discussed insofar as they relate to data management.

The report concludes that the market is a rapidly evolving one and that it offers important opportunities for the management services department to improve the service provided to the enterprise. The report recommends the steps that management services should take to exploit these opportunities.

The Butler Cox Foundation is a research group which examines major developments in its field – computers, telecommunications, and office automation – on behalf of subscribing members. It provides a set of 'eyes and ears' on the world for the systems departments of some of Europe's largest concerns.

The Foundation collects its information in Europe and the US, where it has offices through its associated company. It transmits its findings to members in three main ways:

- As regular *written reports,* giving detailed findings and substantiating evidence.

- Through *management conferences,* stressing the policy implications of the subjects studied for management services directors and their senior colleagues.

- Through *professional and technical seminars,* where the members' own specialist managers and technicians can meet with the Foundation research teams to review their findings in depth.

The Foundation is controlled by a Management Board upon which the members are represented. Its responsibilities include the selection of topics for research, and approval of the Foundation's annual report and accounts, showing how the subscribed research funds have been employed.

The Butler Cox Foundation

Butler Cox & Partners Limited
Morley House, 26-30 Holborn Viaduct, London EC1A 2BP
Tel 01-353 1138, Telex 8813717-GARFLD

SISDOCONSULT
20123 Milano – Via Caradosso 7 – Italy
Tel 86.53.55/87.62.27

Akzo Systems B.V.
Velperweg 76, Arnhem, The Netherlands
Tel 85-662629

Butler Cox & Partners Limited
216 Cooper Center, Pennsauken, New Jersey 08109, USA
Tel (609) 665 3210

Printed in England