

McGILL UNIVERSITY

Survey of Database Applications and Management Systems

Submitted to: Prof. T.H. Merrett

February 25th, 1977

Bruno Leps

Table of Contents

PART I: DATABASE APPLICATIONS

1.1 The Database vs the Company.....	p. 1
1.2 Survey Objectives.....	p. 2
1.3 Methods.....	p. 3
1.4 Representation and Biases.....	p. 3
1.5 Findings.....	p. 5
1.5.1 Flexibility.....	p. 6
1.5.2 Programmer vs Non-Programmer.....	p. 7
1.5.3 Programmer effectiveness.....	p. 9
1.5.4 Techniques.....	p. 11
1.6 Conclusions.....	p. 14

PART II: DATABASE MANAGEMENT SYSTEMS

2.1 Introduction.....	p. 16
2.2 Flexibility.....	p. 17
2.3 Programmer vs Non-Programmer.....	p. 20
2.4 Security and Privacy.....	p. 21
2.5 User Response.....	p. 22
2.6 Some Practical Considerations.....	p. 24

PART III: THE RELATIONALITY OF IT ALL

3.1 The Relational Approach.....	p. 26
3.2 Concluding Remarks.....	p. 27

PART IV: APPENDIX AND REFERENCES

PART I: DATABASE APPLICATIONS

1.1 The Database vs the Company

Data processing has traditionally been considered as essential but tedious work-- a burden relegated to the clerical staff. Computerization gives the users the opportunity to eliminate the slowness and inefficiency incurred by these manual methods. However, users are finding out that computerization, while eliminating some problems, also creates new ones. It is not the mere replacement of the clerical staff by a machine. As more people use computers (11 years ago there were some 400 computer installations in Canada whereas today, two to three thousand are being installed each year)¹, the problems associated with such a move are becoming prominent. The use of computerized database systems is a major source of the problems for it not only requires important changes within the company organization but also forces users to change their whole outlook on "data".

The use of a database system often warrants the creation of a separate department having to justify its expenditures while meeting the demands of its users at minimal costs in terms of money, time and personnel. Data is promoted from a clerical level to a corporate level within the organization.

The system further requires a high degree of problem specification: the user must know what he wants out of the system. Database systems, unlike manual methods, do not permit vagueness. Notwithstanding the original cost, any future major revision carries along with it a very expensive price tag: the system must be useful, without major changes, for

a significant length of time.

However, as much as database systems are forcing changes within companies, the reverse is also true. The design and structure of database systems are being influenced by their ever growing use within the business community. Thus, a survey seeking to identify the needs of database system users has become imperative, if we are to understand the present and future trends of database technology.

1.2 Survey Objectives

(1) To determine what the computing community is using in terms of database systems.

(2) To determine what the computing community is looking for in a database system.

Our aim was not to accumulate extensive statistical information on the different home-grown or commercial systems in use but to seek qualitative trends in data organization, processing methods, query languages, protection, privacy and security.

From the start of the survey we had decided to limit ourselves to users within the Montreal region. We do not feel that this significantly biased our findings as a highly representative sample of computer applications and users can be found within these geographical limits.

1.3 Methods

Of the 400 computer installations within the Montreal region 100 were selected for an original first contact. Out of these, 60 showed interest in our survey and were sent a copy of the letter and questionnaire(Ref. Appendix) and responded as follows:

(1) Eighteen contributors were interviewed by a researcher with the questionnaire filled in by the researcher during the interview.

(2) Nine users filled in the questionnaire and returned it directly to us.

(3) Two respondents, following an interview with a researcher, filled in the questionnaire and returned it.

1.4 Representation and Biases

INDUSTRY	APPLICATION	NO.
Financial	Accounting	9
	Personnel	4
	Insurance	1
	Banking	1
Materials	Inventory	5
	Planning	3
	Distribution	1
Service	Reservations	2
Misc.		13

TABLE 1

Table 1 shows the breakdown of the different applications represented by the 29 contributors to the survey. This table includes the fact that some users have more than one system. The applications cover the principal database activities of the Manufacturing, Distribution, Financial and Transportation industries². Of the whole spectrum of major computer users the survey has no representative database applications from Government, Utility and Other Services (education, hospitals and the like)*.

The following should also be pointed out:

- (1) All but one user have in-house hardware along with in-house programmers.
- (2) All but two applications were on larger machines.
- (3) All but one user had developed their own ad-hoc information retrieval systems before adopting a database system.

Thus the following groups are excluded or unevenly represented in our study:

- (1) Users with Government, Utility and other services applications,
- (2) Users buying machine and programmer time from commercial establishments,
- (3) Users with database applications on minis,
- (4) Users going directly from manual methods to a computerized database system.

* Representatives of each of these three applications were contacted and sent a questionnaire. However, we had not, at the time of writing this report, received any responses.

However, we feel that these omissions do not significantly bias our prime objective of finding out what the user is looking for in a database system. In the future, as the state of the art progresses, it will become imperative to distinguish between each class of user; however, for the present, it remains important to identify those points which users, as a whole, have in common.

At this time, if a major bias does exist in the report, it comes from the willingness of the companies to participate in such studies rather than from the representativeness of the respondents. Nevertheless the range of applications encountered within the different industries allows us to make significant points.

1.5 Findings

We have divided the discussion of our findings into 4 sections:

- (1) Flexibility: the need to adapt.
- (2) Programmer vs Non-Programmer: is one more important than the other ?
- (3) Programmer effectiveness: a main concern.
- (4) Techniques: data structures, security, privacy.

These divisions are perhaps not the usual way of viewing database systems; however, they do represent the users' main preoccupations.

1.5.1 Flexibility

1) Change

One of the major problems facing the database using community is the necessity of being able to adapt constantly to new uses: the system may have been designed for one specific application or as a general system but, as the company grows, new applications arise and old applications have to be broadened.

The results of our survey show that 20 out of 29 contributors have extensively changed their database systems within the last two years. Another 4 are presently conducting year-long studies seeking alternatives to their present systems. In all but three cases, the inability of the present home-grown system to be adapted to new uses without tremendous expenditures in cost and time, inevitably forces the user to search for and adopt a completely new system.

2) Generality

The users need flexible systems. The specialised methods used in the earlier systems are no longer satisfactory. It still remains to be shown whether or not the present systems (be they developed in-house or commercially) will meet the flexibility requirements of the users: this will only be determined after long and extensive usage. We find, however, that users searching for flexibility in different systems usually begin by looking into commercially available packages. One of the reasons is that commercial packages, even specialized ones, must be adaptable to

a wide range of applications, or they would not justify their development costs.

Of the 20 contributors who have changed their system within the last two years 18 have gone from a home-grown system to a commercial package with in-house support. Of the 4 contributors presently studying alternative systems, all are presently looking into commercial packages.

The need for flexibility does not only apply to users wanting to combine different applications under one system but also to users wanting a system for one specific application.

3) Efficiency

Flexibility in system design usually reduces running efficiency. However, we find that the users are willing to pay extra computer costs for the sake of the flexibility.

1.5.2 Programmer vs. Non-Programmer

Extensive efforts are presently being made to allow non-programmers to query databases. These efforts are in terms of using interactive facilities, allowing the non-programmer, through an interface, to access data by the use of simple phrases or codes. The appearance of general query languages even permits the non-programmer to run the equivalent of small programs without programmer help. The trend towards increasing non-programmer activity is clearly shown in Table 2 where 80% of the contributors have both programmers and non-programmers querying their database.

	Programmers only	Non-programmers only	Both	Total
Querying	3	2	20	25
Updating	8	2	15	25

Table 2

The important point to notice however, is that increasing non-programmer activity also increases the need for programmers. This is seen particularly in the case of updates and also in the case of reporting and query applications.

(i) Updates: the table shows significantly more cases of programmers-only than non-programmers-only where updates are concerned. Allowing non-programmers to do the actual updating of the database demands stricter controls, primarily to make sure that the integrity of the data is preserved; less controls are required if the user, wishing to update the data, puts in a request to the programmer who will, in turn do the actual updating. Having users on the system also implies the need for better measures of security and privacy. Where users do allow non-programmers to update the data, it is usually done via controlled processes. For instance, the non-programmer may be allocated space within records where he can insert his data and periodically programmer written routines will effectively update the database. Another version of this is that the non-programmer will be allocated a workspace where he can enter his data and, according to the demand, the programmer will update the database with the "workspace

data".

(ii) Reporting and query application: Even allowing for the use of a general query language, the bulk of reporting applications must be written by programmers, whether in-house or working for a software firm.

Thus, notwithstanding the advantages of having non-programmers using the database at the company level (e.g. managers have up-to-date information and can thus make better decisions) the programmer remains indispensable.

1.5.3 Programmer Effectiveness

1) Efficiency

Users are concerned with manpower efficiency rather than with machine efficiency, although in few frequently used jobs, optimization is an asset. Even in these cases the user is concerned only to eliminate gross inefficiencies. The philosophy is rather for the programmer to get as many jobs done within the shortest period of time*. To this end:

(1) Users favor the use of the less efficient high-level languages making programs easier to write (Table 3).

(2) Users favor the use of the more costly interactive facilities to do the application programming development; it cuts down

* Of interest is that some users are still faced with situations where it takes 6 months in order to develop a program to produce an ad-hoc report. This perhaps suggests the need for even higher level languages or perhaps that there are some problems with the data structures being used.

on the time it takes to debug the programs.

LANGUAGE MOST EXTENSIVELY USED	NO.
Cobol	10
General Query Language	4
Fortran	2
RPG	2
Basic	1
APL	1
Assembler	2
TOTAL	22

Table 3

In the same way that users favor interactive querying facilities for non-programmers some users favor off-line updating procedures for programmers as is shown in Table 4. Indeed, where programmers are responsible for the actual updating of the database, off-line procedures simplify their task: There is less of a concern for file maintenance, integrity, security and privacy.

	On-line only	Off-line only	Both	Total
Querying	2	2	25	29
Updating	2	8	19	29

Table 4

Finally, some users pointed out that they preferred to have the in-house programmers developing application programs rather

than systems. This is an issue of programmer effectiveness, which like the issue of flexibility, points to the increasing use of commercial package systems.

2) Communications:

Improving the ease of communication between programmers is also a concern to the user. To this end, a great deal of time and effort is spent in order to standardize programming techniques; the overwhelming use of Cobol being perhaps the best proof of this point. The principal goal ~~is~~ to improve the programmer's efficiency by making it easier for him, to correct or modify previously written programs.

1.5.4 Techniques

1) Data Structures

The first step in satisfying the user's need for flexible systems while maximizing manpower efficiency falls on the choice of data structures-- no matter how good the system software or application programs may be, if the data structures are not geared to the user's needs, he will face endless problems. A skyscraper cannot be erected on a foundation intended for a country home.

As commercial packages become more common, the choice of physical and logical data organizations is removed from the system user to the system designer*. It is interesting to note a trend

* One important step which is uniquely a user responsibility is the modelling of the data to the logical structures of the system; a step which in itself can cause havoc if not properly executed.

towards more complex structures, which is partly explained by users wanting to have interactive systems utilized by non-programmers. It is also interesting to note that the majority still use a basically hierarchical type of data organization, as illustrated in Table 5.

DATA ORGANIZATION	NO.
Hierarchical	16
Network	6
Inverted	5
Sequential	3
Hash	2
TOTAL	32

Table 5

However, notwithstanding all the advantages of the hierarchical type of organization, flexibility is certainly not one of them. As a simple example let us consider a Supplier-Parts database hierarchically organized: as long as the queries are directed primarily on the supplier the system runs smoothly and effectively; however, a query of the type "Who are the suppliers of Part-X ?", requires a complete search of the system.

Therein lies an important dichotomy: we have found that users attach the greatest importance to flexibility and yet we find them using hierarchical data structures. Why? This is

surely because the concern for flexibility has led users towards commercial systems, yet most commercial systems have been hierarchically organized. It is apparent that users will abandon their hierarchical packages as more flexible systems (network, inverted and relational) become commercially available--our survey already shows some evidence of this.

The trend towards complex data structures may also be moving the user away from another of his goals: programmer effectiveness. Complex data structures lead to messy programming, whether the programmer is developing a database system or merely interfacing with an existing one. Many problems arise over program accuracy, data integrity, security and privacy that do not arise with simpler structures. This gives rise to long program development times and to semipermanent "bugs" in the systems, damaging both programmer effectiveness and flexibility.

2) Security and Privacy

We define the terms "security" and "privacy" as follows:

Security: The state of protection against system and program failure whether hardware or software.

Privacy: The state of protection against other users or systems querying, changing, deleting or adding data belonging to one user or system--whether these operations be accidental or deliberate.

The increased complexity of database systems and the advent of the non-programmer have caused users to become aware of the

security and privacy facets of these systems. However we find that most of our respondents give only lip-service to security and privacy. They recognize the importance, in principle, of these features, but presently treat them as secondary in practice, probably because they still face problems at more fundamental levels of the system.

Security, however, is of such importance in the transaction oriented systems handling several thousands of transactions per day that some users have responded by using dual data. Others use audit trails, thus allowing for a trace back to the point where the problem started. Variations of this are validation checks on all data being entered or periodical spot checks. For the majority it remains for the individual to be responsible for his data.

Privacy becomes a concern on the multi-user systems where most companies who have commercial systems use the privacy measure inherent in the package. Others use password or terminal access privacy measures.

Better security and privacy measures will undoubtedly become more of a concern to the users when they have resolved some of their more pressing problems.

1.6 Conclusions

We have found that the primary concern of database users is for flexibility - they want systems that can respond to new

demands and accomodate new data. We find that non-programmers require more access to the data but that programmers are as much in demand as ever. We find a great concern for programmer (as opposed to machine) efficiency and a desire to use programmer resources for applications as opposed to system development. We find an extensive use of high-level languages and an increasing use of commercial system packages, due presumably to the concern for flexibility and programmer effectiveness. We perceive some contradictions in our evidence, namely that between the widespread use of hierarchical data structures and the need for flexibility and that between the increasing complexity of data structures and the need for programmer effectiveness. These we interpret as due to the inadequacy of present commercial systems and we foresee an increasing movement of users from home-grown or existing commercial systems to more general commercial systems, preferably ones with essentially simple data organization.

PART II: DATABASE MANAGEMENT SYSTEMS

2.1 Introduction

From the survey we have established some of the user priorities concerning database management systems. We shall now take these priorities and see how the more extensively used commercial management systems meet these requirements by looking at the facilities they offer. The approach will be descriptive and critical. Table 6 gives a brief overview of the systems we shall be dealing with.

SYSTEM	DESCRIPTION ³
ADABAS	Based upon associative addressing implemented through partially inverted files; functional data compression to reduce file size; geared to support very large databases.
IMS	To provide data organization on physical devices and an interface to logical program structure of application programs.
SYSTEM 2000	A generalized DBMS for fast response and access based upon partially inverted files and with a powerful imbedded command language for terminal-based and interactive use.
TOTAL	A host language DBMS for controlling database structure; preformatting disk storage and controlling access along CODASYL lines; designed and oriented for transaction processing rather than retrieval.

TABLE 6

2.2 Flexibility

As mentioned in section 1.5.4 the first step in satisfying the users need for flexible systems while maximizing on manpower efficiency falls on the choice of data structures, and on having a standard approach to describe the contents of the database and establish the relationships between the data.

1)Data structures: Systems are commonly categorized as hierarchical, network or inverted; in fact few of the more advanced commercial systems fall exactly into one of these categories. As can be seen from table 7, they rather use a combination of features. As such, hierarchical and network systems will also have inversion capabilities while inverted packages will support structural descriptions. In fact, with the widening of the "structural capacity" of the systems, it can be predicted that in the future the systems will become very similar. These complex data structures are in use to:

i) permit the database to meet the needs of the application programs,

ii) permit the database to be able to incorporate changes in the data without affecting existing applications and without demanding complete re-organization.

This last point deserves more attention: while these complex data structures make for more flexible systems they do have a serious drawback. The system's response to change is often by way of more pointers, links or indexes with the net result that

the user may not have to re-organize the data but will be faced with a system running sub-optimally. On the long run the user may still have to face the high cost of re-organization. Some IMS users, for example, find it profitable to spend 9 hours of machine time every two weeks simply to re-organize⁴. The point here is to examine the long term operational costs. We must note that System 2000 and ADABAS keep their pointers removed from the actual data. This, while making for even greater machine independence, increases the ability to change without having to go through the actual data. On the larger applications this can be a real asset.

SYSTEM	DATA STRUCTURE
ADABAS	Inverted lists which can be associated into networks; can support any data structure: sequential, tree, network
IMS	Hierarchical with multiple indexing; may obtain network type using pointers in the hierarchical structure.
SYSTEM 2000	Hierarchical with inverted lists on any level of the hierarchical structure.
TOTAL	Network with imbedded pointers with the restriction that a "member" record cannot be the "owner" of another record.

TABLE 7

2) Data definition language(DDL): the DDL removes the burden of describing the contents of the database and the relationships between the data from the application program. This is in contrast to the old method where the data description was, by necessity, imbedded in the application program and thus, any subsequent changes in the data necessitated the re-writing of the application programs. Table 8 gives a brief description of the DDL used by ADABAS, IMS, SYSTEM 2000 and TOTAL.

SYSTEM	DDL ⁵
ADABAS	Database is a collection of files each with records and fields; the fields are defined by name/type/length/format/attribute.
IMS	Application program requires a program specification block which along with the database description forms the logical records of the procedure.
SYSTEM 2000	Own hierarchical format based on repeating groups; independent of the application program.
TOTAL	One format for owner record, 2499 formats for member records; records can be redefined.

TABLE 8

Commercial systems certainly offer users a lot of flexibility

however, no one system gives the ideal solution as there always remains a trade-off to "change", be it complexity or running cost. The importance of this trade-off will only be evaluated after the systems will have been in use for several more years.

2.3 Programmer vs. Non-Programmer

In order to increase non-programmer involvement some commercial systems offer general query language facilities whereby the user may easily query the database. Report writers are also available allowing the non-programmer to generate reports without programmer intervention. Of interest, is the fact that out of the 10 most used DBMS only 4, for the moment, do offer a general query language facility. On the other hand all use a data manipulation language(DML) that functions within a host language. This trend will certainly change with an increase in non-programmer activity but does well reflect the users concern to improve programmer effectiveness: the role of the DML is primarily to eliminate the tedious file handling work from the application program and thus allows the programmers to respond faster to application demands(DMLs are not complete languages but rely on host languages to provide the procedural capabilities) while making programs more general (Ref. Table 9).

Two further facilities are commonly offered by commercial systems in order to improve both programmer and non-programmer

effectiveness:

(1) Commercial systems have facilities for both interactive and off-line processing.

(2) Commercial systems have facilities for handling multi-user tasks.

SYSTEM	DML
ADABAS	Call statements to load, modify, read, find, and delete; uses Assembler, Cobol, Fortran, PL/1.
IMS	Call statements with parameters specifying file name, key, field specifications; uses Cobol, PL/1, Assembler.
SYSTEM 2000	Procedural language interface to Cobol, PL/1, Assembler
TOTAL	Uses CALL statement in any language that supports the CALL statement.

TABLE 9

2.4 Security and Privacy

In view of the high degree of user facilities the systems have very sophisticated security and privacy measures (Ref. Table 10). Notwithstanding these measures it still remains to be determined the actual overhead cost of having file access security and recovery/restart facilities; this is certainly an important

point especially when one considers the fact that a clever person will eventually access the data he wants and that the recovery/restart facilities nevertheless involve going from a known back-up through the transaction trail (which is really all that the system provides) to bring the database up to date. Again, the important point is that the systems offer the facilities but at what actual cost.

SYSTEM	SECURITY and PRIVACY
ADABAS	File and field level security for reading and updating; restart and recovery facilities.
IMS	Security control and reconstruct restart.
SYSTEM 2000	Read and update control at field level; audit trails; restart/recovery.
TOTAL	Logging security and recovery.

TABLE 10

2.5 User Response

Table 11 gives the results of a survey conducted in december 1976 by DATAPRO. The users were asked to evaluate their software package. (The rating in each category is expressed in terms of the weighted average calculated on a scale of 4 for "excellent", 3 for "good", 2 for "fair", and 1 for "poor".) Some interesting

points come out of the table:

- (1) Users seem to be generally satisfied with their systems.
- (2) Overall documentation, vendor technical support and training rate lower than the other categories.
- (3) Considering the price difference between ADABAS and the other systems the users do not show significantly more satisfaction.
- (4) The extensive use of TOTAL.
- (5) An important consideration, which is not shown on the table, is the year the users bought the systems; these figures become much more significant if the users have been using these systems over five or more years.
- (6) Generally users are less satisfied with the IMS package.
- (7) Overall the highest ratings are given to TOTAL.

	ADABAS	IMS	SYSTEM 2000	TOTAL
Users Reporting	4	33	21	113
Overall satisfaction	3.3	2.9	3.0	3.5
Throughput/efficiency	3.3	2.4	2.3	3.1
Ease of Installation	3.3	2.2	2.9	3.4
Ease of Use	3.5	2.5	3.5	3.4
Documentation	2.0	2.8	2.4	2.8
Vendor technical support	2.8	2.8	2.7	3.0
Training	2.5	2.9	2.1	2.8
Price(approx.)	\$100,000	\$700/month	\$30,000	\$40,000

2.6 Some Practical Considerations

From the brief overview it is quite evident that on the surface, at least, the present systems have gone a long way in satisfying the users requirements. However some drawbacks remain. We have already mentioned two of them: the systems boast ability to change, yet the user may still have to face costly re-organization in order to avoid complete inefficiency; even though users show little concern for machine time cost, they may be facing extremely high overhead costs in order to have file and field security facilities along with recovery/restart measures. Other points that should be considered include:

(1) Notwithstanding the cost of the basic package what are the costs to administrate, program and monitor the actual implementation of a database management package?

(2) It takes approximately 8 years to absorb the initial implementation costs⁶. Within that period it is quite likely that there will be better hardware facilities (not to mention better software packages) available on the market. To what degree is the present software convertible to new environments? The present complexity of database management systems may in fact stop the user from taking advantage of new hardware developments for the conversion may cost too much. In this sense simpler implementations will be easier to adapt.

(3) What are the long term costs of maintaining the software? Buying a database management package does not mean that the user

can forget about software support. This is shown when looking at one IMS user who found it profitable to have five full time maintenance specialists solely for the IMS software plus five more solely for trimming and tuning the IMS operationally (so that it did not use excessive machine resources)⁷.

These points emanate from two problems:

(1) The relative youth of the systems: as such, it is impossible to determine exactly the costs of maintaining a database management system over a long period of time. This is crucial for it may raise the question about the validity of the whole database management system concept for certain applications.

(2) The complexity of the present systems: the level of complexity is necessary in order to meet the user's requirements. However, the overheads it causes tend to limit the use of database management packages to large applications where, extensive use, completely submerges the high costs of implementation, maintenance and re-organization. Better hardware will certainly decrease these costs and will thus make database packages available to the smaller applications.

PART III: THE RELATIONALITY OF IT ALL

3.1 The Relational Approach

Better hardware will not solve the complexity faced by the applications programmer: in order to simplify the logical structures faced by the programmers we may have to adopt a new approach as that suggested by the relational model.

The relational model is currently receiving a lot of attention as a viable alternative to the network and hierarchical type of logical data structures. It is by no means the only alternative but is currently receiving most of the attention for it has the advantages of offering a rigorous method to structure data while being simple.

With its tabular-like structures the user is not faced with any type of links or pointers. Furthermore the rules it provides to structure data tends to yield more data independence, reduces redundancy and thus also reduces the likelihood of anomalies within the data. The real strength of the relational model is that the simplicity of the data structures allows the user through simple operators to deal with entire files (tables) at once. Through the use of known operators (using mathematical formulations as in relational algebra and relational calculus) the manipulation language is thus powerful and simple.

Presently the relational model has one main drawback--performance. All implementations to date have been on rather small databases. For the moment the strength of the relational model is also its weakness: as it operates on entire files at once,

retrieving time tends to be longer than with the other conventional approaches and it needs large core requirements. However for modest size databases requiring flexibility and a lot of bulky querying the relational approach is very useful.

Hardware which would allow large sections of memory to be addressed at once would render obsolete existing systems and would place the relational model in the forefront. It remains that the relational model offers a means of simplifying the complexity presently encountered and thus needs to be investigated further.

3.2 Concluding remarks

Throughout the survey we have tried to take the user's view of database applications and database management systems. Although we have tried to cover a wide range of topics we do leave a lot of unanswered question. What is the role of the database administrator? Where will minicomputers or even microprocessors fit in? What data is essential and what only frills? Which database features are essential and which only fashions? What are users experiencing about legal implications and confidentiality of data? Can we expect standardized techniques to develop which will ultimately meet the needs of all database users?

The database management systems presently available meet the users requirements. There is, however, a definite need for more investigations of these systems in order to determine their actual cost. For the moment the complexity at the system software level would seem to be dependant on the development of new hardware;

such hardware could possibly replace some functions currently being handled by software, and would lessen the burden presently placed on it. At the user end there is also a need to simplify present data structures and to give a rigorous method for the structuring of the data. The relational approach offers a possible solution dependant on better performance.

PART IV: APPENDIX AND REFERENCES

APPENDIX



McGILL UNIVERSITY

Survey of Database Applications and Management Systems

The School of Computer Science and the Faculty of Management at McGill University have undertaken a study of database applications and management systems in Montreal.

The aim of the committee is to prepare and make available to the database community a report on the different databases and systems in use. The report will be a survey rather than detailed and will seek trends in data organization, processing methods, query languages, protection, privacy and security. We will follow the format used by the CODASYL report "A Survey of Generalized Database Management Systems"

We are circulating this letter to computer users, manufacturers and software houses in Montreal to ask your support in supplying information about your database applications and services. A researcher will be contacting you in the near future and we would be grateful if you would answer his questions. In the meantime, we enclose a copy of the questionnaire which we use in the interviews: please use this to prepare for the researcher's visit, or alternatively, fill it out and return to:

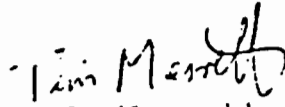
Database Study
c/o Prof. T.H. Merrett
School of Computer Science
McGill University
P.O.BOX 6070, Station A
Montreal, H3C 3G1

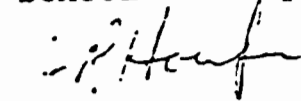
We plan to hold a meeting on January 28th of all contributors and any others interested to present our finding prior to publication of the report.

All responses will be kept confidential and the report will be a summary only, containing no references to individual companies unless permission has been expressly received.

Please direct any inquiries you may have to Mr. Bruno Leps c/o School of Computer Science at 392-8275.

Sincerely,


T.H. Merrett,
Associate Professor,
School of Computer Science.


H.R. Howson,
Associate Professor,
Faculty of Management.



McGILL UNIVERSITY

Sample survey questionnaire containing summary of results

DATA BASE SURVEY

Note: The numbers in brackets represent the number of contributors for each category. The numbers in the right hand column represent the number of contributors who answered the question.

<u>Company:</u>	N/A	
<u>Interviewed:</u>	N/A	
<u>Name of System:</u>	N/A	<u>TOTAL</u>
<u>Functions of System:</u>	(Some users have more than one system)	29
<u>Financial:</u>	Accounting(9)	
	Personnel(4)	
	Insurance(1)	
	Banking(1)	
<u>Materials:</u>	Inventory(5)	
	Planning(3)	
	Distribution(1)	
<u>Service:</u>	Reservations(2)	
	Misc. (13)	
<u>Status</u>	(date of implementation and of revisions):	29
	Less than two years(20); greater than two years(9)	
<u>Hardware used:</u>		23
	Mini-computers(2); others(21)	
<u>DATA:</u>		
	Total amount of data on system: 10^6 (3), 10^7 (1), 10^8 (4), 10^9 (1) bytes	
	How many files: 10^0 (1), 10^1 (2), 10^3 (2) files	
	File structure: Hierarchical(16), Network(6), Inverted(5),	29
	Sequential(3), Hash(2)	



McGILL UNIVERSITY

DATA(cont'd)

	<u>TOTAL</u>
Range of file size:(approx) <u>10⁵(1), 10⁶(1), 10⁷(4), 10⁸(1) Bytes</u>	7
Record size range:(approx) <u>10²(4), 10³(5) Bytes</u>	9
Growth rate: <u>12%/year(2)</u>	2

ACCESS:

How many accesses: <u>10⁰(2), 10³(3), 10⁵(1), 10⁶(1) per day</u>	7
WHO accesses the data: <u>Programmers only (3),</u>	25
<u>Non-Programmers only (2), Both (20).</u>	
HOW is the data accessed: <u>Interactive only(2)</u>	29
<u>Off-line only (2)</u>	
<u>Both (25)</u>	
Average amount of data accessed (Record, a whole file,)	
<u>Record(4),</u>	5
<u>Whole file(1)</u>	
Language(s) used: <u>Cobol(10), General Query Language(4),</u>	22
<u>Fortran(2), RPG(2), Basic(1), APL(1)</u>	

UPDATES:

WHO does them: <u>Programmers only(8),</u>	25
<u>Non-programmers only(2), Both(15)</u>	
HOW are updates done: _____	29
<u>Interactive(2)</u>	
<u>Off-line(8)</u>	
<u>Both(20)</u>	



McGILL UNIVERSITY

UPDATES(cont'd)

WHEN are updates done: Continuously(6), Daily(5) TOTAL
18

User's discretion(4), Varies with corporate decision(2),
Monthly(1).

HOW much data is changed during a given update:(approx)____
No data

SECURITY:(HOW and WHEN(if applicable))

Back-ups: Daily(9), User's discretion(4), 19
Two to three times per day(2), Data kept dually(2),
Once a week(1), Monthly(1).

Privacy: Package Privacy(6), Terminal access(4), 17
Password(4), Terminal access and Password(2), No problem(1).

Consistency: User responsibility(3), Audit trail(2), 7
Validation on input(1), Spot checks(1).

COMMENTS:(Do you have any comments on the questionnaire; If you wish to add anything about your system that is not covered in the survey please feel free to do so): _____

References:

- 1- 1975 Canadian Computer Census, prepared by: Canadian Information Society, toronto: Whitsed Publishing Limited, January 1976,p.25.
- 2- Ibid.,p.23.
- 3- Sprowls, R.C. Management Data Bases. U.S.A.: 1976, p. 103-113.
- 4- Data Base Management. International Computer State of the Art Report
C. Boon and C.J.Bunyan, ed. Maidenhead, Berkshire, England:Infotech Information Limited, 1973, p.490-520.
- 5- Management Data Bases , p.103-113.
- 6- Data Base Management,p.490-520
- 7- Ibid., p.490-520.