Generalized data management systems and scientific information

Organisation de Coopération Et de Développement Économiques (ocde)

GENERALIZED DATA MANAGEMENT SYSTEMS AND SCIENTIFIC INFORMATION

Report of a specialist study

SYSTÈMES DE GESTION DE BASES DE DONNÉES ET INFORMATION SCIENTIFIQUE

Rapport d'étude de spécialistes

Published by/Édité par
OECD NUCLEAR ENERGY AGENCY
AGENCE DE L'OCDE POUR L'ÉNERGIE NUCLÉAIRE
38 bd. Suchet, 75016 Paris France
1978

NEA WORKING GROUP ON NUCLEAR ENERGY INFORMATION
GROUPE DE TRAVAIL DE L'AEN SUR L'INFORMATION
DANS LE DOMAINE DE L'ÉNERGIE NUCLÉAIRE

GENERALIZED DATA MANAGEMENT SYSTEMS AND SCIENTIFIC INFORMATION

SYSTÈMES DE GESTION DE BASES DE DONNÉES ET INFORMATION SCIENTIFIQUE

report of the specialist study on computer software
rapport d'étude de spécialistes sur le logiciel d'ordinateur

The use of Generalized Data Management Systems
for handling Scientific Information

L'utilisation de systèmes de bases de données généralisés pour
le traitement de la documentation et des données scientifiques

jointly organized by/organisé conjointement par
OECD NUCLEAR ENERGY AGENCY
AGENCE DE L'OCDE POUR L'ÉNERGIE NUCLÉAIRE
and/et
UNITED STATES DEPARTMENT OF ENERGY

in cooperation with/en coopération avec
U.S. NATIONAL BUREAU OF STANDARDS

Chairman/Président
A. SHOSHANI, LAWRENCE BERKELEY LABORATORY

Secretary and Editor
N. TUBBS, OECD NUCLEAR ENERGY AGENCY

published by/édité par
OECD NUCLEAR ENERGY AGENCY
AGENCE DE L'OCDE POUR L'ÉNERGIE NUCLÉAIRE
38 bd. Suchet, 75016 Paris France

The Organisation for Economic Co-operation and Development (OECD) was set up under a Convention signed in Paris on 14th December, 1960, which provides that the OECD shall promote policies designed:

- to achieve the highest sustainable economic growth and employment and a rising standard of living in Member countries, while maintaining financial stability, and thus to contribute to the development of the world economy;
- to contribute to sound economic expansion in Member as well as non-member countries in the process of economic development;
- to contribute to the expansion of world trade on a multilateral, non-discriminatory basis in accordance with international obligations.

The Members of OECD are Australia, Austria, Belgium, Canada, Denmark, Finland, France, the Federal Republic of Germany, Greece, Iceland, Ireland, Italy, Japan, Luxembourg, the Netherlands, New Zealand, Norway, Portugal, Spain, Sweden, Switzerland, Turkey, the United Kingdom and the United States.

The OECD Nuclear Energy Agency (NEA) was established on 20th April 1972, replacing OECD's European Nuclear Energy Agency (ENEA) on the adhesion of Japan as a full Member.

NEA now groups all the European Member countries of OECD and Australia, Canada, Japan, and the United States. The Commission of the European Communities takes part in the work of the Agency.

The primary objectives of NEA are to promote co-operation between its Member governments on the safety and regulatory aspects of nuclear development, and on assessing the future role of nuclear energy as a contributor to economic progress.

This is achieved by:

- *encouraging harmonisation of governments' regulatory policies and practices in the nuclear field, with particular reference to the safety of nuclear installations, protection of man against ionising radiation and preservation of the environment, radioactive waste management, and nuclear third party liability and insurance;*
- *keeping under review the technical and economic characteristics of nuclear power growth and of the nuclear fuel cycle, and assessing demand and supply for the different phases of the nuclear fuel cycle and the potential future contribution of nuclear power to overall energy demand;*
- *developing exchanges of scientific and technical information on nuclear energy, particularly through participation in common services;*
- *setting up international research and development programmes and undertakings jointly organised and operated by OECD countries.*

In these and related tasks, NEA works in close collaboration with the International Atomic Energy Agency in Vienna, with which it has concluded a Co-operation Agreement, as well as with other international organisations in the nuclear field.

L'Organisation de Coopération et de Développement Économiques (OCDE), qui a été instituée par une Convention signée le 14 décembre 1960, à Paris, a pour objectif de promouvoir des politiques visant :

- à réaliser la plus forte expansion possible de l'économie et de l'emploi et une progression du niveau de vie dans les pays Membres, tout en maintenant la stabilité financière, et contribuer ainsi au développement de l'économie mondiale;
- à contribuer à une saine expansion économique dans les pays Membres, ainsi que non membres, en voie de développement économique;
- à contribuer à l'expansion du commerce mondial sur une base multilatérale et non discriminatoire, conformément aux obligations internationales.

Les Membres de l'OCDE sont : la République Fédérale d'Allemagne, l'Australie, l'Autriche, la Belgique, le Canada, le Danemark, l'Espagne, les États-Unis, la Finlande, la France, la Grèce, l'Irlande, l'Islande, l'Italie, le Japon, le Luxembourg, la Norvège, la Nouvelle-Zélande, les Pays-Bas, le Portugal, le Royaume-Uni, la Suède, la Suisse et la Turquie.

L'Agence de l'OCDE pour l'Énergie Nucléaire (AEN) a été créée le 20 avril 1972, en remplacement de l'Agence Européenne pour l'Énergie Nucléaire de l'OCDE (ENEA) lors de l'adhésion du Japon à titre de Membre de plein exercice.

L'AEN groupe désormais tous les pays Membres européens de l'OCDE ainsi que l'Australie, le Canada, les États-Unis et le Japon. La Commission des Communautés Européennes participe à ses travaux.

L'AEN a pour principaux objectifs de promouvoir, entre les gouvernements qui en sont Membres, la coopération dans le domaine de la sécurité et de la réglementation nucléaires, ainsi que l'évaluation de la contribution de l'énergie nucléaire au progrès économique.

Pour atteindre ces objectifs, l'AEN :

- *encourage l'harmonisation des politiques et pratiques réglementaires dans le domaine nucléaire, en ce qui concerne notamment la sûreté des installations nucléaires, la protection de l'homme contre les radiations ionisantes et la préservation de l'environnement, la gestion des déchets radioactifs, ainsi que la responsabilité civile et les assurances en matière nucléaire ;*
- *examine régulièrement les aspects économiques et techniques de la croissance de l'énergie nucléaire et du cycle du combustible nucléaire, et évalue la demande et les capacités disponibles pour les différentes phases du cycle du combustible nucléaire, ainsi que le rôle que l'énergie nucléaire jouera dans l'avenir pour satisfaire la demande énergétique totale ;*
- *développe les échanges d'informations scientifiques et techniques concernant l'énergie nucléaire, notamment par l'intermédiaire de services communs ;*
- *met sur pied des programmes internationaux de recherche et développement, ainsi que des activités organisées et gérées en commun par les pays de l'OCDE.*

Pour ces activités, ainsi que pour d'autres travaux connexes, l'AEN collabore étroitement avec l'Agence Internationale de l'Énergie Atomique de Vienne, avec laquelle elle a conclu un Accord de coopération, ainsi qu'avec d'autres organisations internationales opérant dans le domaine nucléaire.

AVERTISSEMENT

Les informations publiées dans ce rapport n'engagent pas la responsabilité de l'Organisation de Coopération et de Développement Économiques.

LIST OF PARTICIPANTS IN THE STUDY

Meetings held at OECD, Paris, 11th-13th January 1977
and Lawrence Berkeley Laboratory, USA, 5th-7th October 1977

Austria
Dr. A. Nevyjel, SGAE Wien

France
M. G. Martin, CISI Saclay

Federal Republic of Germany
Dr. H. Behrens, ZAED Karlsruhe

Japan
Dr. T. Yamamoto, Univ. of Tokyo
(also representing ICSU/CODATA)

Netherlands
Dr. H. Rietveld, ECN Petten

United Kingdom
Mr. K. Montgomery, UKAEA Risley

United States
Dr. E. Birss, LLL
Dr. A. Brooks, ORNL
Dr. D. Deutsch, NBS
Dr. C. Dunford, BNL (NNDC)
Ms. P. Fuja, ANL
Dr. W. Gersbacher, Battelle

United States (continued)
Ms. G. Haire, LBL
Dr. V. Hampel, LLL
Dr. J. Hilsenrath, NBS
Dr. H. Honeck, Savannah River Lab.
Dr. K. Hsu, Battelle
Mr. T. Hughes, Library of Congress
Dr. E. Jones, LLL
Dr. D. Knoll, NODC
Dr. D. Richards, LBL
Dr. D. Ries, LLL
Ms. J. Robinson, LBL
Dr. A. Shoshani, LBL (Chairman)
Dr. P. Stevens, Cal Tech
Dr. J. Suich, Savannah River Lab.
Dr. K. Szczesny, Battelle

International Organizations

CERN Geneva	:	Dr. G. Moorhead
CEC/JRC Ispra	:	Dr. S. Perschke
		Dr. J. Petrie
		Mr. J. Powell
IAEA/NDS	:	Ms. P. Attree

Sponsoring organizations

U.S. Department of Energy	:	Dr. C. Gottschalk
OECD Nuclear Energy Agency	:	Dr. P. Johnston, CCDN Saclay
		Dr. A. Schofield, CCDN Saclay
		Dr. W. Schuler, CPL Ispra
		Dr. N. Tubbs (Secretary)

ACKNOWLEDGEMENTS

The organizers wish to thank the above participants for their contributions, whether in written form or in discussion, and to acknowledge many personal contributions to the logistics of the study, as well as that of Lawrence Berkeley Laboratory in hosting the second meeting. Particular mention is due to Mr. T. Hughes, now at the Library of Congress, who has been personally involved with the study from its early planning stages.

CONTENTS

TABLE DES MATIERES

GENERALIZED DATA MANAGEMENT SYSTEMS

Note on the English terminology

In discussions on scientific information, the stock of information to be handled is often called a Data Base. By extension, programs for managing the data collection are sometimes loosely referred to as a data base management system. However, such specialised program systems do not in general constitute a generalized Data Base Management System (DBMS) in the sense in which this term is widely used in computing technology. To avoid confusion in the English text, and to emphasize the important distinction between a DBMS and a set of specialised programs to manage a particular (file-structured) collection of data, we have preferred to use the term Generalized Data Management System (GDMS) throughout the present study. The expression 'Data Base' is normally used in the narrow sense, to mean 'an integrated collection of data managed by a GDMS'.

The term used in French is 'Système de Gestion de Bases de Données' (SGBD), and in this report 'Système de Bases de Données Généralisé' (SBDG), as a more direct translation of GDMS. We believe that no confusion exists about their meaning in French.

AVAILABILITY OF THIS REPORT

A limited number of copies of this report are available for distribution free of charge. Requests should be addressed:

for the United States to :

Dr. C. GOTTSCHALK
Office of Technical Information
Department of Energy
Washington, D.C. 20545
USA

for other OECD Member countries to:

Dr. N. TUBBS
OECD Nuclear Energy Agency
38 Boulevard Suchet
75016 Paris
France

DISTRIBUTION DU RAPPORT

Un nombre limité d'exemplaires de ce rapport est destiné à la distribution gratuite. Les demandes doivent être adressées :

pour les Etats-Unis à :

Dr. C. GOTTSCHALK
Office of Technical Information
Department of Energy
Washington, D.C. 20545
USA

pour les autres pays, membres de l'OCDE, à :

Dr. N. TUBBS
Agence pour l'Energie Nucléaire
de l'OCDE
38 boulevard Suchet
75016 Paris
France

GENERAL INTRODUCTION

This report is aimed at people with scientific background, such as physicists, chemists, biologists, etc. as well as management personnel. Its purpose is to stimulate scientists of all disciplines to consider the advantages of using a generalized data management system (GDMS) for storage, manipulation and retrieval of the data they collect and often need to share. The report should also be of interest to managers and programmers who need to make decisions on the management of scientific (numeric or non-numeric) data. Another goal of this report is to expose the features that a GDMS should have which are specifically necessary to support scientific data, such as data types and special manipulation functions.

It is hoped that the way the report is organized, starting with basic concepts and the terminology of GDMS, then discussing the requirements of GDMS for scientific data, and describing case studies, will be of value to people who have not been exposed to GDMS before. At the same time, the reader more familiar with GDMS can benefit from guidance on available systems, from discussion of other users' experience, and capabilities in GDMS helpful in handling scientific data. More specifically, the report will:

a. Give a clear introduction to what GDMS are, why they may be useful, and what computing hardware is needed.

b. Include a list of the capabilities required in a Generalized Database Management System for handling scientific data. Presented at a time when considerable effort is being invested in GDMS software development, such an inventory may be in time to influence the specifications of this third generation of Data Management Systems.

c. Compare possible alternatives: do-it-yourself software, APL, file management systems.

d. Show by case studies of a variety of existing and potential GDMS applications to scientific data (in different fields, with more or less numerical content) what is involved in GDMS use, and what advantages may result.

e. Survey the direction of development work in GDMS: hardware development trends, software development trends, distributed Database systems, and database conversion.

A GDMS is a system that provides generalized tools for the purpose of defining a database structure, for loading the data, for modification of the data, and for organizing the database for efficient retrieval and formatted output. A data management system is "generalized" when it provides a user-oriented language for the different functions, so that it is possible to define any new database, its internal organization, and to retrieve and modify the data without the need to develop special purpose software (program) for each new database. The main purposes of a GDMS are quoted from a recent survey [1]:

- to make an integrated collection of data available to a wide variety of users;
- to provide for quality and integrity of the data;
- to ensure retention of privacy through security measures within the system; and
- to allow centralized control of the database, which is necessary for efficient

data administration.

From the user point of view GDMS should provide:

- data independence, i.e. that application software does not need to be modified when data or data structures are changed
- languages and facilities to perform the spectrum of data management functions: data definition, data entry and updating, conditional search and data retrieval, and data output and report generation. These facilities could be available in on-line or batch mode depending on the needs of the application.
- the representation and access of both numerical and literal data.

The above points will be discussed in more detail in the next section of the report.

The advantages of using a GDMS are numerous, but for scientists and other users who are not computer specialists (and who cannot afford the time to become expert programmers) the main advantage is the immediate availability of a system for database handling. If their data handling requirements are relatively simple, these may perhaps be satisfied without further programming by use of the query language/report writer facilities of a suitable GDMS. Where user requirements are more complicated, applications programs in a high-level 'host language' (such as COBOL or FORTRAN) may be linked to the database by embedding GDMS Data Manipulation Language statements in those programs. The available programming effort can be concentrated on the specific problems in hand.

The generality of GDMS can sometimes degrade the efficiency of an application compared to a specially developed program, but this consideration is usually outweighed by the savings in software development time and cost, by the availability of the data for numerous applications, and by facilities providing integrity and security and easy modification and manipulation of the data. Database management, like the use of high-level programming languages instead of assembler, is the latest in a long series of compromises in which increased user convenience is traded against computer efficiency. Although initially these trade-offs are expensive, hardware and software have historically tended to evolve in a way which reduces the cost of convenience.

Recognizing that scientific data may have different characteristics from "commercial" data, it is the aim of this study to explore the differences and point out systems that have features amenable to the handling of scientific data. Therefore, the people involved in the study were selected both from the scientific community and the computer field. In particular, cases where GDMS were used for the handling of scientific data are described.

The report introduces first the spectrum of GDMS approaches, techniques and terminology. This is done at a level which describes the functionality of GDMS without going into the details of how it is achieved. Then a survey of (mostly commercially) available systems is given, together with comparative features. This is followed by a discussion of requirements and capabilities needed to deal with scientific data and a description of several data management systems designed specifically to handle scientific data. In the next sections several example scientific applications that use GDMS successfully are described, followed by examples of potential applications now under consideration. Finally, a discussion of future trends in the development of GDMS and related areas is given so that the reader may consider their possible effect on his environment. In the conclusion section (part of the epilogue) we summarize the viewpoints of participants relative to when should GDMS be used and what to expect in using them.

REFERENCE

[1] James P. Fry and Edgar H. Sibley, "Evolution of Data-Base Management Systems," Computing Surveys, Vol. 8, No. 1, March 1976, (this entire issue is devoted to Data Base Management)

PART I
AN INTEGRATED APPROACH TO DATA MANAGEMENT

PARTIE I
UNE STRATEGIE INTEGREE DE GESTION DE DONNEES

AN INTRODUCTION TO GENERALIZED DATA MANAGEMENT SYSTEMS

G. Moorhead, CERN, Geneva
N. Tubbs, OECD/NEA, Paris

I. WHAT IS A DATA MANAGEMENT SYSTEM ?

1. Scientific data handling by computer

Scientific and engineering programs handle data from diverse sources. The process by which data generated in experiments is refined for ultimate use in technology typically covers three stages. Raw data is first collected from experimental equipment, then later analyzed by the individual experimenters for publication. In the final 'evaluation' stage, experimental results are reviewed, compared and aggregated into a final recommended data set, which may be used as the input data for calculations in many different technological applications, and is called 'evaluated' data. A scientific data handling application may be required to accept source data at any one of these stages, and on one or more of a variety of media. They may originally have been punched on cards or paper tape, keyed in at an interactive terminal, or have been read out directly from an experiment on to magnetic tape. At some stage, the data ends up in computer files and an analysis program will read these files to produce a set of results which may be histograms, graphs, computed values or simply listings.

All major computers have a file manager as part of their operating system allowing the programmer to declare named logical files and specify on which physical storage devices they are to reside. The system will organize the physical space and write data to it or read data from it when requested to do so by a user program. A typical analysis program will be written in FORTRAN, and the layout of data in the files will be reflected in its FORMAT statements. This approach causes no problems where the files are created only once, and then used in a single program or slight variants of it.

However there are many situations in which logically interrelated files may be accessed by several different programs. Matters will be complicated further if these files are frequently updated; they may also be subject to independent modification by several authorised users.

It is in such cases that the Generalized Data Management System (GDMS) approach can be of great utility, and there are now many examples of its successful use in scientific or engineering applications. What a GDMS offers is, essentially, a high level language for describing and

manipulating data to complement programming languages such as FORTRAN (which is designed primarily for performing calculations), COBOL or PL/1 in maintaining an integrated and logically consistent data base for use in a variety of different applications.

2. The function of a GDMS

A Generalized Data Management System appears to the user as a software interface separating him and his programs from the operating system and the external storage hardware, in respect of all access to a centrally controlled and integrated data collection, shared by a number of users and referred to as a Data Base. The system provides tools for defining the physical structure of the data base and the logical relations within it, for loading and modifying the data, for protection of the data base against accidental damage and unauthorised access, and for efficient data retrieval. Good special-purpose systems may provide many of these facilities. A data management system is "generalized" when it provides a user-oriented command language for all these different functions, applicable to any new data base regardless of its internal organisation, thus removing the need to write new data handling programs for each new data base.

When a file is read by a conventional analysis program, usually one 'record' is read from it at a time where a record is a collection of 'items' of logically associated information, such as the measurements made at one location and/or at one particular time, during a scientific investigation. A file usually contains only records of the same form, and which can, for example, be read by similar FORTRAN statements. The files to be integrated in a data base may represent an equal number of different record types, each containing data items with varying formal attributes (integer, floating point, character data...). A GDMS offers facilities for handling these records and data items individually or in sets, using data manipulation commands dependent only on the logical structure of the data base. The division of records into files, and the actual disposition of records and files on the physical storage devices, is almost invisible to the user.

It should be remarked that a software package providing most of the basic GDMS facilities described below can itself be written in a higher-level language such as FORTRAN, making use of the standard interface for file-handling provided in that language. Typically, such a package can be developed with a few man-years of effort, but may be found lacking in generality, reliability and efficiency.

II. THE FACILITIES OFFERED BY A GDMS

3. Data independence of user programs

A GDMS allows the user to refer to and retrieve individual items in a record directly by name, without the need to declare the record structure in the user program. The structure of the data base (record structures, names of data items and the relations linking different record types) is declared independently from individual programs, for all applications, in an initial phase known as Data Definition.

Once a data base exists and contains data, a particular user has only to be concerned with the names of those items which are of interest to him, and an application program need only refer to those items which it needs. This facility is by no means negligible as can be seen in a real

life GDMS application for oceanographical research where each record contains no less than 73 items ranging from the latitude and longitude to percentages of different minerals in samples brought up.* The same information could of course have been split between several overlapping files of shorter records, but such a split would bring with it the need to administer, by program and by the inclusion of link information, the relations between corresponding records in the different files. This is in itself a major function of a GDMS.

The removal of explicit information about the physical and logical structure of the data from the user programs to a central schema accessed through the GDMS is said to confer a degree of 'data independence' on the user programs. It will be seen from later articles in this study report that data independence is far from complete in most presently available systems.

4. Updating facilities

The data management system provides updating facilities for storing, modifying or deleting data from the data base. When the data base is updated either by inserting completely new records or by modifying the values of items in existing records, the system automatically validates and converts the data according to the formal attributes of the item, which are usually supplied by the user when the item is named for inclusion in the schema of the data base. An item may be of the type Integer, Character (i.e. alphanumeric text), Date, Real, etc. As an example of validation, '1A34' would not be accepted in place of 1234 as the value of an item named LENGTH and described as Integer. Some systems also allow the user to specify a range or set of acceptable values for an item, or even more general validation conditions.

Rapid retrieval of specified data items, whether from a classical file or from a data base, usually requires the data to be kept physically on a random access storage device (currently discs). One of the more difficult problems in programming data update operations for random access files is to ensure that the file can be recovered if it is corrupted during update following a program error or a computer breakdown. In a GDMS which permits several users to work in parallel updating the data base, it is important not only to be able to correct errors due to one of them, but also to do this without rubbing out the work done in the meantime by the other users. Many GDMS provide utilities for data base recovery following an accident to one of the constituent files, or the inclusion of invalid data. These facilities may be obtained by taking periodic copies of the whole base, supplemented by the logging of all update transactions.

5. Data retrieval

The most important function of a GDMS is to allow the retrieval of data according to prescribed criteria. At its simplest, retrieval may consist of obtaining, in an application program, a single record containing specified values in particular items which have been declared as key items, for example, Item PARTICLE = PROTON and Item PLAB = 1260. An easy variant of this is to retrieve all records satisfied by a range of key item values, for example, PARTICLE = PROTON, and PLAB between 12000 and 13000, the successive records being supplied to the application program on request. However, some useful systems limit the number of possible key items to one, which could simply be an identification number for the record.

*Centre National pour l'Exploitation des Océans, Centre Océanographique de Bretagne, Brest.

It is easy to see that a file can be organized so that records may be retrieved efficiently by declared keys. Indeed, file organizations such as "indexed sequential" or "random access" are provided for this purpose by most operating systems. However, the advantage of a GDMS is that the user has simply to state which of his named items are to be used as keys, and the GDMS takes care of a non-trivial amount of housekeeping in order to set up the files and organize the updating and retrieval.

In addition, a GDMS offers the possibility of easily posing questions of a kind which may or may not have been envisaged when the data base was created. One obvious way of doing this, though not the easiest, is to pass through all the records of a given type in turn looking at the values of some items. The required items would be fetched using CALL's to the GDMS, and the tests would be made using the IF statements of the "host language" such as FORTRAN, from which the CALLs are made. Most GDMS allow records of several different types to be scanned simultaneously, thus allowing the equivalent of multi-file retrieval. Another feature of some systems is free-text search, or examination of an alphanumeric item character by character, looking for a particular substring in it.

The more advanced GDMS provide, in addition, a high-level query language for expressing retrieval criteria in a natural manner. This same language may be used for making updates. The retrieval conditions usually consist of simple predicates in which items are compared to a constant (e.g. PARTICLE = PROTON) or linked together by logical operators, as in (PARTICLE = PROTON OR PARTICLE = ANTIPROTON) AND PLAB GE 12000 AND PLAB LE 13000. As well as retrieving individual records, the GDMS often provides basic statistical functions such as means, variances, regression analysis or even presentation in histogram form of the results of a search.

Privacy of data may be ensured by limiting a user's access to certain files (or record types), or to a logical subset of the data base, which in some GDMS is referred to as a subschema. The degree of discrimination between users which can be imposed by these privacy locks depends on the system : several systems can allow a user access to specified data items only within a given record type.

Sometimes the user is allowed, essentially, to declare beforehand the type of question he expects to ask frequently, and the system then creates "access paths" for faster retrieval at the expense of updating time and storage space. However, the possibility of asking unforeseen questions without writing and testing a special program is one of the more pleasant features of GDMS.

For commercial applications, presentation of the retrieved output in formatted and sorted reports with headings and subheadings, footings, summaries, sub-totals, etc., is a very important requirement. The fact that sophisticated report generators are available in some form or other with standard GDMS may not matter for the majority of scientific applications, but it can be extremely useful in a few cases where printed compilations are required. Unfortunately graphical output, which is obviously desirable for scientific applications, is not usually available with report generators, but clearly a host language which interfaces to both a GDMS and a graphics package offers this possibility at the cost of some coding.

6. Control of redundancy between data

A data base carries, in more or less integrated form, the data which would otherwise be dispersed over a number of overlapping files. In the case of the files, their logical overlap is expressed by recording the same values for various data items in two or more files. Within the data collection seen as a whole, and stored in an integrated data base, these repeated data may be seen as redundant. The structural information expressed in the overlapping files by repetition of data values is now handled by the GDMS, and may appear as address pointers or cross-indexing transparent to the user.

One reason for avoiding the inclusion of redundant information in the data base is that it wastes storage space. A more important one (since many data bases occupy more storage space than the files they replace) is that redundancy which is not controlled by the GDMS itself may result in errors which will seriously corrupt the data base. At the level of data items redundancy can be reduced by limiting the number of times a given item is recorded in the data base : system structural information linking the different record types referring to the item will replace repetition of the item. Another method at the item level is to store once only any long text which appears many times as the value of an alphanumeric item, or any set of values of different items which are always associated with a particular value of one other item, for example the properties of a chemical compound. The important feature for the user is that he need keep in only one place the full descriptive data concerning, for example, a particle or a chemical compound which is known elsewhere in the data base by a short name or code. A reduction in quantity of data stored may bring an improvement in quality.

III. DATA STRUCTURES

The basic facilities which are provided by the majority of GDMS have now been outlined. In descriptions of current GDMS, much emphasis is usually placed on the kind of logical structure in which the user is allowed to store his data. In present-day practice, this logical structure is inextricably bound up with the physical structure or accessing methods used. The logical structures offered by various GDMS may differ sufficiently to influence the choice in favour of some rather than of others for a particular class of applications, but not usually enough to invalidate the use of any given GDMS altogether.

7. Hierarchical structure within a single record

A common kind of logical structure is the intra-record hierarchy in which items within a record are organized in a hierarchy or tree structure. This may be restricted to be a two-level tree in which an item at the first level may consist of an indefinite number of sub-items, rather like a vector, at the second level. More generally, some systems allow a multilevel hierarchy in which sub-items themselves can consist of sub-items, and so on; for example, an elementary particle can decay into several particles, any one of which may decay into other particles, etc. Naturally, the system looks after all the internal pointers needed to implement such a hierarchy and the user searches through the tree in an application-oriented manner. Historically, intra-record structures of this type formed the basis of early GDMS because the records could be maintained in a sequential file on magnetic tape. The main problem was that the entire file had to be copied when an update was performed (necessary anyway for tape files), while a search might likewise require the whole file to be read.

Examples of systems using intra-record hierarchies are the early INFOL system, still used for scientific applications in CERN and originally written by T.W. Olle around 1965, and the Oak Ridge ORCHIS system. Fig. 1 shows an intra-record hierarchy; references to individual systems discussed will be found in the list at the end of this report.

8. Hierarchies of records

As falling costs made it possible to store large data bases on rotating mass storage devices such as disks or drums, with their semi-random access capability, retrieval from an intra-record hierarchical structure became more efficient because (provided one knew where to look for it) finding an individual record no longer involved a search of the whole file. It became possible to allow GDMS to handle several files as a single data base, with user access in milliseconds to any item in any one of them. The use of disc storage led to the development of more powerful structures, increasingly oriented by user requirements rather than machine constraints.

An extended hierarchical structure, as implemented in SYSTEM 2000 and shown in Fig. 2, may be supplemented by inverted indices allowing specified data items to be accessed directly at the disc address given in the index rather than by following the hierarchy down from the top. Different levels in the logical hierarchy are represented by distinct sets of records (called 'repeating groups') rather than by items in one record. Such a hierarchical structure does not permit direct links between items or records in different hierarchies.

9. Network structures

Logical network structures are based on the concept of interlocking 'sets' each consisting of an owner record and one or more member records. A set can be implemented as a circular linked list, and may be searched in logical sequence by entering at the owner record and accessing members in succession until the required member record is identified.

Depending on whether a given record may be both owner of one set and member in another, it may or may not be possible to represent a hierarchy directly using only sets of this type : Fig. 3 makes the point clear. The very widely-used TOTAL system allows records to be owners or members in many sets, but not to be both owner and member. Although more levels can be represented indirectly, direct representation is thus limited to a two-level hierarchy. Systems conforming to CODASYL specifications (see Section 13 below) do not have this restriction, and can represent full hierarchies.

The greater power of networks, as compared to hierarchical structures, lies in the possibility of associating one record type with (almost) any other. Rather than adapting the data base structure to the limited set of associations permitted by a hierarchical structure, the user may start by defining the record types of interest to his application, and then express in the data base schema all the associations that exist between them. A pure hierarchical structure can be seen as a degenerate case of a network, in which no record may be a member in more than one set.

10. Relational data bases

The relational model, developed later and so far implemented in a number of experimental systems, is a more user-oriented approach to data base structures. In this model, a data base is viewed as a collection of n-ary relations or homogeneous tables, each row of which is analogous to a record containing n items, none of which can have multiple occurrences. When defining relations, consistency and non-redundancy can be guaranteed by following a set of formal rules which ensure that the relations are all in so called Third Normal Form.

There is a closed algebra of operations called join, projection etc., which can be performed on relations, and interrogation of a relational data base consists in applying this algebra. The user is freed from specifying access paths when defining relations, though access paths are indeed used when making joins, and cannot be entirely ignored. No relational GDMS is yet commercially available. Fig. 4 shows schematically how a hierarchy may be represented by a collection of relation tables in which data are stored.

11. Physical storage of data

It must be made clear that the logical structures which can be expressed by a GDMS need not be directly reflected in the way in which data records are stored on disc. In systems based on network structures, for example, records accessed by their logical keys are often distributed at apparently random disc addresses within a system-defined file. These disc addresses are often obtained from the logical keys by a 'hashing' algorithm and the aim of this procedure is to ensure even distribution of data over the available disc space. In SYSTEM 2000 the structural information which constitutes the hierarchy is stored in 'hierarchical location tables' (indices) separate from the data records themselves, which are stored on disc in approximately the order in which they are loaded. Sets (in network systems) and hierarchical structure are usually expressed by address pointers, and sets or hierarchies may well be physically intermingled and disordered as compared to the logical order expressed by the system-generated pointers.

Although the physical layout of the data base may not be very similar to the logical structures it represents, the performance of the system may still be very much affected by the way the data storage is 'tuned' to match the logical structure and the commoner user program paths through the data base. Conversely, it may degrade performance to represent a data base directly as the logical view of the user, while performance of a particular program may in any case be degraded when the data base structure is designed to satisfy the overall performance requirements of multiple applications.

IV. NOMENCLATURE

Some further features of present-day GDMS will now be considered in order to introduce terms which will be used later in this report.

The innovations in data base structures which became possible with the use of disc storage have been discussed in Sections 8 - 11 above. Data stored in random access disc files may be reached directly by obtaining the disc location (its 'address') through a randomizing algorithm ('hashing') or from an index (an index is a directory in which a logical key value may be looked up to find the physical locations of the records in which that key value occurs and which may be implemented internally in various ways, including hashing). Pointers in a data base context refer to addresses generated by the GDMS and embedded in the data base records to give directly the physical location of the adjacent records in the logical structure. Pointers were already used in the intra-record hierarchies of the early tape storage GDMS, and are the main support for the implementation of CODASYL networks.

12. An example of early GDMS : INFOL

Most of the important characteristics of current GDMS can be found in the later versions of the simple INFOL system mentioned in Section 7. INFOL is an entirely self-contained system with its own query update language. In contrast to most later systems, it cannot be accessed by CALL statements from a host language such as COBOL or FORTRAN. These user languages allow commands at a 'high level' of logical abstraction, in contrast to 'low level' languages (such as assembler language), much closer to the machine language of the computer in use. Although the original INFOL was written in assembly code for CDC 3600/3800 computers, it has acquired portability by being rewritten in standard FORTRAN. Another new feature is the possibility of its being used interactively (that is from a terminal with keyboard) as well as in batch mode (that is using card reader and line printer).

INFOL was designed for tape storage, and has a sequential file structure with an intra-record hierarchy. The data is described by the user in an establishment phase using what would now be called a Data Description Language (DDL). The data description, which in this case consists only of logical descriptions of items in a record is stored in the file separately from the data itself. Only one item in the record can have the privilege of being a key item, and records in the file are ordered according to the value of that item. An item may be declared to be multiple (more generally called repeating); that is it may have an indefinite number of occurrences.

The initial loading of data into an empty data base is known as population of the data base. In the case of INFOL, this may be done in the updating phase, in which data already loaded may also be modified or deleted. Items are checked or validated before being inserted. An example of automatic validation in INFOL is that a date is checked to see if it is a proper calendar date, even taking into account leap-years. Because updating is performed by copying the complete sequential file, a back-up copy is usually available, and thus integrity is easily ensured, at the expense of computer efficiency.

Retrieval is performed in an interrogation phase, using a query language, in which search criteria can be specified, as already described above. As an INFOL record has only one key item and as the file is sequential anyway, virtually any query is an unforeseen question and implies an examination of all the records in the file. When the desired records have been retrieved, they can be displayed partially or completely, in an easily prescribed manner using a modest report generator, which lacks the full

spectrum of facilities for page layout provided by more sophisticated report generators.

Finally INFOL allows restructuring by change of description of existing items, or by addition of new items. This is facilitated by the fact that after any change to data or description, the entire data base is copied.

13. CODASYL systems

The 1971 report of the CODASYL Data Base Task Group defined standards for a GDMS to be used primarily with COBOL as the host language. It proposed the use of a network structure (see Section 9) to model the relations between records in the data base. Hierarchical relationships are a simple special case of a chain of network sets, and so can also be represented.

The report gives the detailed syntax for a Data Definition Language (DDL), and a Data Manipulation Language (DML) for inclusion in the COBOL language. A Data Base Description is called a schema, and the view of the schema which an individual COBOL user needs, or is allowed to have, is called a sub-schema. In the DDL the user is allowed a choice of accessing methods for records in sets and by record type, and thus can take account of efficiency for foreseen retrievals. Using the DML, the COBOL programmer figuratively navigates through the network as he moves from set to set.

No query language is proposed in the CODASYL report, while the question of multi-user interaction is dealt with to some extent. Rules are prescribed for avoiding the conflicts which can arise when two users try to update simultaneously the same or closely related data. Also security at all levels is provided for restricting access to information in the data base.

Some well-known CODASYL systems are IDMS (Cullinane Corp., for IBM, ICL and other computers), DMS 1100 (Univac), DBMS-10 and -11 (DEC), and IDS-II (Honeywell).

14. Future GDMS development

Relational GDMS were discussed in Section 10, and it is this data model which is currently receiving the most attention in new systems development. Independently of the data model used, it was felt important to establish a broad outline of standards for future GDMS designs. The 1975 ANSI SPARC proposals, which have been widely accepted, foresee a clear separation between (a) the external schema through which the data is accessed by user programs, and which may be relational, network or whichever interface is more natural to the user, (b) the conceptual schema which carries the intrinsic structure of the enterprise being modelled, and (c) the internal schema which controls the physical storage in the data base. The internal schema can be tuned, for example by adding or removing access paths, to match data base use. A system which starts to approach this design is EDMS (by CDC, Brussels, and some universities).

Future GDMS can be expected to have user interfaces allowing queries to be made in more or less natural language, or in a powerful formal language according to convenience. The data bases themselves may be distributed over many computers in a communications network.

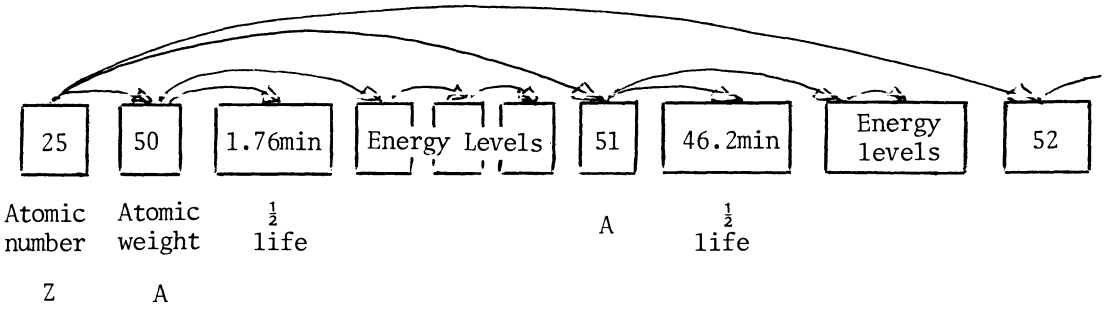


Fig. 1: An intra-record hierarchy of 3 levels: one record per element. The example shows part of a record for manganese, Z = 25. The arrows show the logical structure.

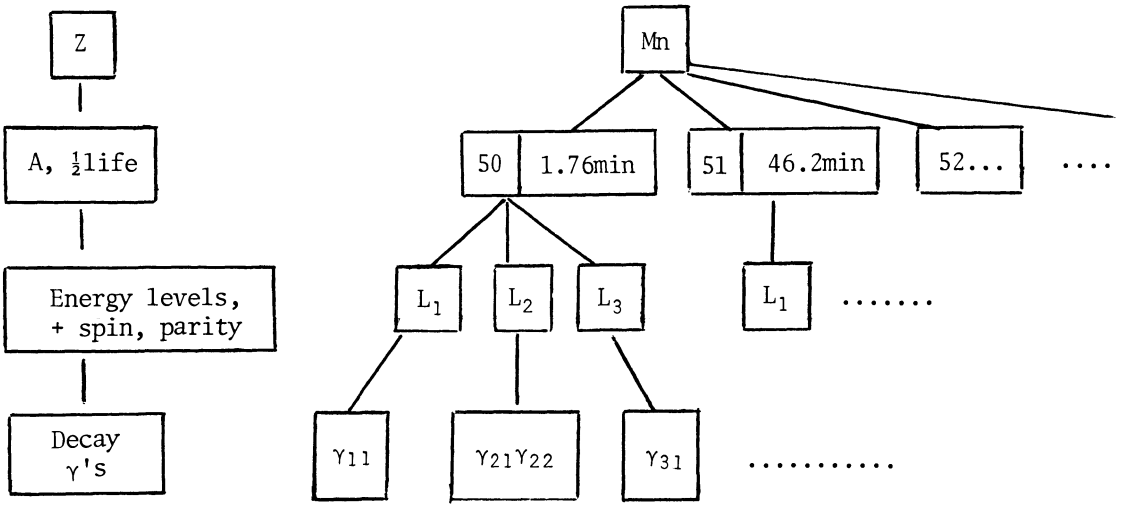
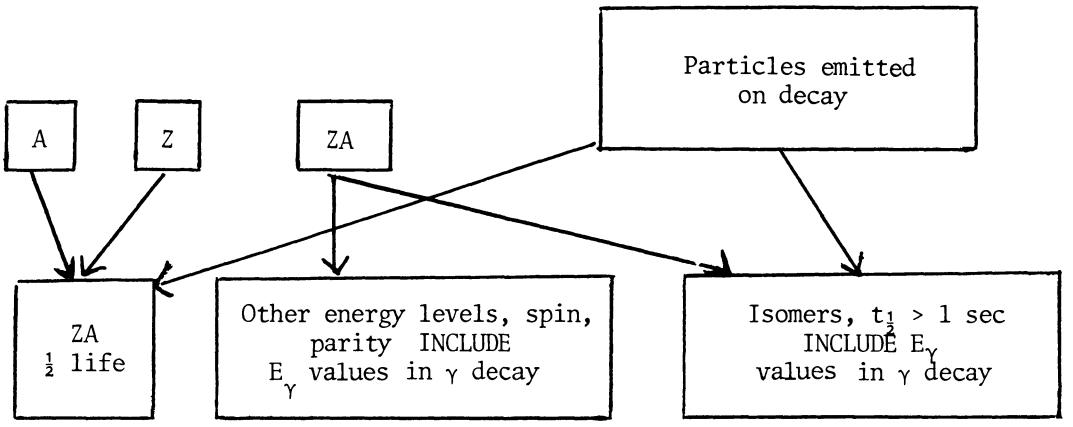
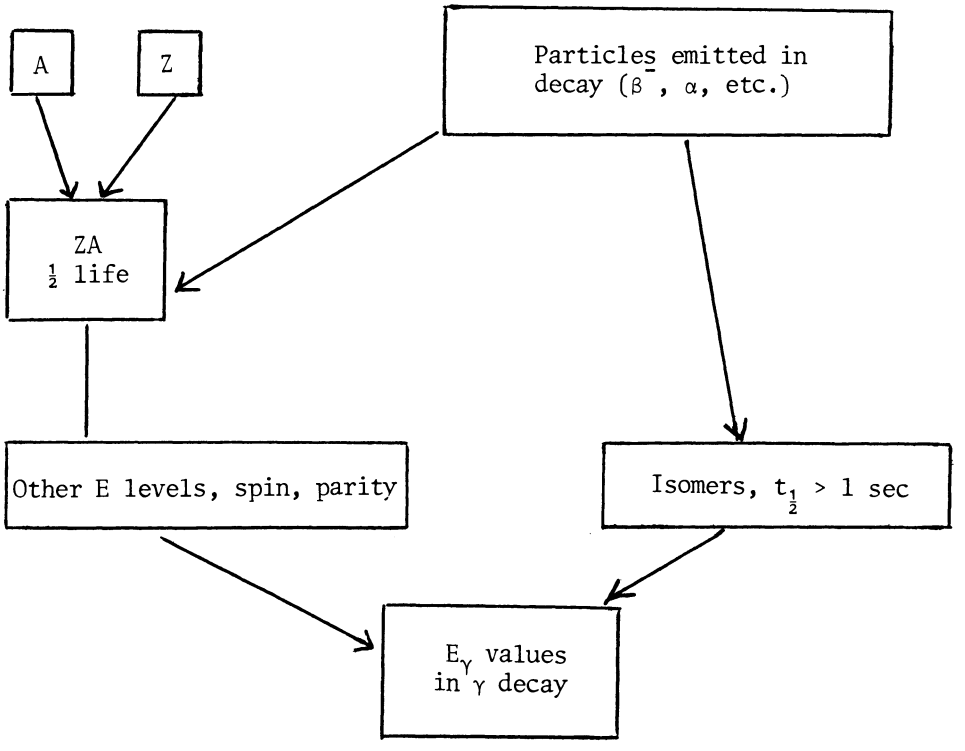


Fig. 2: A full hierarchial structure showing more nuclear structure details. Each box on the left defines a repeating group, and the example on the right shows how one Z branch of the hierarchy might be populated.



a) Example of a limited network structure



b) Example of a full network structure

Fig. 3: Examples of network structure, showing further nuclear structure details.

Domain	Z	A	½ life	...
Tuple 1	25	50	1.76 min	...
2	25	51	46.2 min	...
3	25	52	...	...
...	...	...		

Domain	Z	A	E-level	Spin	Parity
Tuple 1	25	50	E_1	...	...
2	25	50	E_2	...	...
3	25	51	E_3	...	...
4	25	51	...	...	...
...	...	...			

Domain	Z	A	E-level	Gamma	...
Tuple 1	25	50	E_1	γ_{11}	...
2	25	50	E_2	γ_{21}	...
3	25	50	E_2	γ_{22}	...
4	25	50	E_3	γ_{31}	...
...	...	...			

Fig. 4: Parts of relations representing the nuclear structure details of Fig. 2. The E_i and γ_{ij} would be actual values in a real data base.

CHARACTERISTICS OF GENERALIZED DATABASE MANAGEMENT SYSTEMS

D. Deutsch and E. Fong

Institute for Computer Sciences and Technology
National Bureau of Standards*
Washington, DC 20234 U.S.A.

Generalized database management systems are complex and diverse software products that are used increasingly by organizations of all types. While many applications of database technology are highly successful, others do not meet expectations. An important determinant of success appears to be a close match between application requirements and database management system capabilities. This paper describes characteristics that differentiate database management software packages and presents in an appendix a list of software products and the computer systems on which they are available.

Key words: Acquisition sources; feature description; GDMS; generalized database management systems; product characteristics; software.

1. INTRODUCTION

The complexity and diversity of modern Generalized Database Management Systems (GDMS) make them difficult to characterize. An understanding of the current state of GDMS technology is important, however, for those considering the use of these powerful software tools.

Database management systems are software products used for implementing application systems. The increasing use of GDMS packages attests to the success of many database applications. Unfortunately, many others do not achieve desired levels of performance within expected resource expenditure levels. Often unsuccessful systems utilize GDMS software that is not well matched to application requirements. This paper is aimed at those whose interest in the application of GDMS technology exceeds their knowledge of these new software tools. The following sections review database management acquisition sources and characteristics of GDMS

---

*This work was supported in part by the U. S. Department of Energy (formerly Energy Research and Development Administration) under Interagency Agreement No. EA-77-A 01-6010, Task No. A050-TI.

currently available in the marketplace. The appendix contains a list of commercially available GDMS and related software packages and the hardware on which they are operable.

Specific database management products and their vendors are identified for illustrative purposes in the following sections and in the appendix. Inclusion or omission of specific systems should not be construed as a judgement, endorsement, or recommendation by the National Bureau of Standards.

2. GDMS ACQUISITION SOURCES

Database management software can be acquired in several ways and from many sources. Acquisition methods include lease (rental) and purchase agreements where GDMS products are considered separate from other system software, and "bundled" arrangements where GDMS software is paid for as part of a complete package of hardware and software. GDMS providers include: hardware vendors, software vendors, computer services, universities, and other sources. Each of these sources is discussed in turn below. Tables listing a few GDMS packages available from each of the source types are presented for illustrative purposes.

2.1 Hardware Vendors

Computer hardware suppliers have traditionally provided operating systems and support software to their customers. GDMS software is also marketed in this way. Like other software products, GDMS packages are in some cases priced separately and in others "bundled" with the hardware. In either case, the hardware vendor is also responsible for GDMS software support. Many users believe that dealing with a single vendor eliminates possible conflicts and clarifies responsiblity. Conversely, detractors of this acquisition method claim hardware vendors have a vested interest in developing software that "locks-in" users. That is, GDMS as well as other software products may be developed in a manner that makes it difficult to transport applications to other hardware systems. Some GDMS provided by hardware vendors appear in Table I.

Table I - GDMS Provided by Hardware Vendors

PACKAGE NAME	HARDWARE VENDOR	MAINFRAME
DMS-II	Burroughs Corp	B6700/7700
DMS-170	Control Data Corp.	CDC 6600
DMS-1100	UNIVAC	UNIVAC 1100 Series
IDS-II	Honeywell	Honeywell 60/600/6000
IMAGE/3000	Hewlett Packard	HP3000 CX
IMS	IBM	IBM 360/370

2.2 Software Vendors

The complexity and level of support required by a modern GDMS makes this type of software a likely candidate for development as a proprietary package. Software vendors provide support including installation, training, documentation and sometimes assistance in application design. Proponents of independent software organizations believe they are more responsive to user requirements because of competitive pressures than are hardware vendors. Many independently developed GDMS packages are available for IBM hardware; fewer are available for other hardware. Examples of GDMS offerings by software vendors are listed in Table II.

Table II - GDMS Provided by Software Vendors

PACKAGE NAME	SOFTWARE VENDOR	MAINFRAME
ADABAS	Software Ag	IBM 360/370 Siemens 4004 UNIVAC 70 series
IDMS	Cullinane Corp	IBM 360/370 UNIVAC 70 series
INQUIRE	Infodata System, Inc.	IBM 360/370
MODEL 204	Computer Corporation of America	IBM 360/370
System 2000	MRI System Corp.	IBM 360/370 CDC 6000 series UNIVAC 1100 series

2.3 Computer Services

Access to GDMS software can be acquired through the purchase of computer services. The user pays for on-line and/or batch access to a hardware system that provides one or more GDMS packages (often at an additional charge). This type of arrangement is sometimes advantageous: it allows access to GDMS software without acquisition of specific hardware; the burden of installation and maintenance of the GDMS software falls on the computer service organization rather than on the user; and a prototype application can be developed prior to a total commitment to a specific GDMS. A disadvantage can be cost; time-sharing (on-line) and service-bureau (batch) charges often exceed those of in-house GDMS installations. Some organizations do not like to be dependent on outside computer service firms because they believe it reduces their control over critical applications and information.

GDMS provided by computer time-sharing and service firms vary with the hardware systems used. Table III shows some GDMS packages provided by major domestic U.S. computer service organizations.

Table III - GDMS Provided by Computer Service Organizations

GDMS PACKAGE NAME	COMP SERVICE	MAINFRAME
System 2000 DML ALADIN	INFONET	UNIVAC 1108
System 2000	Cybernet	CDC 6600
DMS-2	GE MARK III	Honeywell 6088
System 2000 System 1022 INQUIRE RETRIEVE	Tymshare	IBM 370/158 PDP 10 IBM 370/158 SIGMA 9

2.4 Universities

Many of today's commercial GDMS packages began as research projects in universities and other laboratories. Several state-of-the-art systems are currently available from their developing institutions. While these systems generally employ innovative techniques, they are rarely as complete and as fully tested as those available commercially. The systems are frequently distributed free or with only a nominal charge, but testing, maintenance and support can not be expected to meet commercial standards. The systems are therefore typically of interest primarily to other research installations. Examples of GDMS implemented by universities and available for a nominal fee appear in Table IV.

Table IV - GDMS Implemented by Universities

PACKAGE NAME	IMPLEMENTER	MAINFRAME
INGRESS	U. of California Berkeley	PDP-11
ZETA	U. of Toronto	PDP-11
UNIBASE	U. of Florida	IBM 360/370 or any machine with full ANS Cobol 74

2.5 Other Sources

GDMS are also available from sources other than those listed above. In the United States a number of GDMS have been developed with Federal government funding, and are therefore in the public domain. Some of these systems are available at no charge, but like the university developed systems they may lack even rudimentary maintenance and updating support. Other, however, are available through software vendors; these are

typically priced and supported like proprietary software. Table V includes representative systems developed initially under U. S. government contract.

Table V - GDMS implemented with U. S. Federal Funding

PACKAGE NAME	IMPLEMENTER	MAINFRAME
GIM 2	TRW	IBM 360/370 PDP-11
NIPS	Defence Intelligence Agency	IBM 360/370
MIDMS	Defence Intelligence agency	IBM 360/370 Honeywell 6000 series
WWDMS	Honeywell	Honeywell 6060
MIRADS	National Aeronautics & Space Administration	UNIVAC 1108

3. FEATURES OF GDMS SOFTWARE

Despite the universality claimed by many GDMS software vendors, all current systems favor one pattern of use over another. A method for describing and classifying GDMS software is necessary so that capabilities can be matched to application requirements. Unfortunately, there is no taxonomy for GDMS that is both comprehensive and unambiguous.

This paper discusses some of the features that characterize and differentiate GDMS software packages. It is recognized that specific GDMS products may not be easily described in this framework; some may seem to fit in several classes while others may not be properly described by any. The feature analysis approach, however, does appear to be useful for understanding the nature of any differences among currently available GDMS software packages.

The feature list approach for description of database management software is not new. Two previous efforts were widely distributed and provided the framework for this analysis: the CODASYL Technical Report of May 1971 [1], and NBS Technical Note 887 issued in November 1975 [4].

The features of GDMS software are described below under five major headings: computer environment, secondary storage structures, user interfaces, security features, and implementation orientation. Each heading is further subdivided as necessary to describe more detailed capabilities and functions.

3.1 Computer Environment

Whenever a GDMS is acquired it must be carefully matched to the computer environment in which it will be used. GDMS packages are generally designed to operate on a specific manufacturer's hardware and to interface with particular operating systems, telecommunications packages, and the like. Some important considerations regarding computer hardware and systems software are discussed below.

3.1.1 Hardware. Database management systems are generally developed for one or more specific mainframes or family of central processors. With the exception of systems designed by hardware vendors specifically for their own equipment, GDMS packages are most often developed to operate on IBM equipment because of the large number of IBM installations. Nevertheless, an increasing number of these software packages are developed specifically or converted for use on hardware manufactured by other vendors. Even when a GDMS will be used on hardware produced by a specified vendor, the configuration must be adequate to support the GDMS software. Some hardware characteristics that impact GDMS software are:

* Mainframe
    - -- manufacturer and model
    - -- minimum main memory requirement
    - -- required optional features
* Secondary Storage
    - -- media and speed
    - -- minimum capacity
* Input/Output Capabilities (e.g. terminals supported)

3.1.2 Systems software. GDMS packages must be closely linked to systems software; indeed, it is sometimes useful to think of a GDMS as an extension of the operating system. This close working relationship causes the GDMS to be very sensitive to apparently minor (from the user's perspective) differences in systems software. Types of systems software that must be carefully matched to specific versions required by candidate GDMS include:

* Operating System
* Communications/Teleprocessing Interface
    - -- terminals supported
    - -- re-entrant versus self-modifying (see implementation orientation below)
* Input/Output Access Methods
* Language Translators (Compilers/Assemblers)
    - -- for GDMS source code
    - -- for programs interfacing with GDMS

3.2 Secondary Storage Structures

Database management systems are concerned with storing, maintaining and retrieving large quantities of data. To accomplish these objectives, GDMS provide mechanisms for structuring data on mass-storage devices. Two conceptual views of data characterize modern GDMS packages: the logical, or user's view, and the actual physical representation of data in computer storage. Each is discussed below.

3.2.1 Logical view of data. The logical view of data deals with the users' perception of data without concern for how data items are physically stored. The sum of all aspects of the logical view of data for a GDMS is sometimes referred to as the applicable "data model". Characteristics of logical data views are listed below.

* Basic Units
    - -- fields/attributes
    - -- records/tuples
    - -- files/relations/realms
* Operations
* Data Types
    - -- character/fixed point/floating point
    - -- fixed versus variable length
* Logical (Data) Structure
    - -- flat
    - -- hierarchial (tree)
    - -- network (plex)
    - -- other
        - relational
        - set-theoretic

3.2.2 Physical representation of data. Physical organization and representation of data on mass-storage is an important determinant of GDMS efficiency. The user need not be intimately aware of the details of physical data representation; indeed, a single logical data view can be implemented using various physical representations. However, physical storage strategies differentiate alternative GDMS packages and are the result of major design decisions and trade-offs. An understanding of aspects of physical organization such as those listed below is useful.

* Basic Unit of Stored Data
    - -- character
    - -- computer word
    - -- field/attribute value
    - -- record/tuple

* Physical (Storage) Structure

  -- sequential

  -- indexed sequential

  -- random

* Degree of Physical Separation of Data and Relationships

* Treatment of Redundancy

  -- redundant recording of data at all levels

  -- exploit redundancy to reduce database size:
  - at field/attribute level
  - across records/tuples in files/relations
  - across entire database

3.3 User Interfaces

The interfaces between users and GDMS vary greatly both with respect to capabilities that can be invoked and ease of use. Five aspects of user database interface are discussed below: data definition, external linkages, database maintenance, data retrieval, and user aids.

3.3.1 Data definition. Generalized database management systems provide a facility for data definition that is separate from data manipulation procedures. Data definition languages (DDL) take different forms; some are highly tabular, others have positional (card column) orientations, and a few are relatively free form. A DDL is used for specifying names, types, and special characteristics for data entities. Relationships among data item classes are usually communicated through the DDL, for example, which fields are to function as selection "keys." Security and integrity restrictions may also be specified with the DDL. The resulting definition is sometimes called a "schema".

3.3.2 External Linkages. User written programs often require access to data maintained by a GDMS. Some GDMS provide an external programming language interface as the sole or primary access mechanism. In addition to these "host language" GDMS, some systems which are "self-contained" (in that they have their own query language) also provide procedural language interfaces. External linkages usually take the form of subroutine calls that can be embedded in programs written in languages such as COBOL, FORTRAN, PL/1 or assembly language. Some host language systems provide GDMS macros that are inserted in user programs and translated into the appropriate subroutine calls by a preprocessor. Some GDMS provide no interface at all to external programming languages.

3.3.3 Database maintenance. Database maintenance refers to the initial loading of a database, database restructuring, and updating existing databases. These three maintenance tasks are described and features characterizing alternative GDMS products are listed in the following paragraphs.

Database Loading - provides the initial instance of the database. This process is sometimes termed "population" as it involves filling out the data framework established by the Data Definition Language with an actual "population" of data instances. Aspects of population facilities include:

-- nature of data source
- media (i.e., cards, tape, disk)
- organization (usually sequential)

-- data integrity features
- data type checks
  (numerical versus alphabetical, e.g.)
- range checks
  (age between 21 - 65, e.g.)
- match to value list
  (items must appear on a list of
  permissible items)
- procedural checks
  (right user, right time, etc.)

-- data conversion
- encryption
- character translation
- encoding/decoding

Database Restructuring - is the modification of the data definition for an existing database. Possible modifications include adding or deleting data elements and changing relationships among data item classes. The most straightforward way to restructure is to unload the database, change the definition, and populate the database using the modified schema. Some GDMS products provide utility programs for assisting the user in the restructuring task. Others allow limited restructuring without going through the tedious and costly unloading and repopulating process. A few GDMS software packages offer extensive restructuring capabilities that are automatically invoked via a structure modification language.

Database Updating - is concerned with changing the data contents of an existing database without modifying the data definition. Data instances are added, changed and deleted using GDMS updating features. GDMS packages having their own query language often have a similar language for updating. Other systems allow updating from high-level language programs. Some aspects of database updating facilities are listed below.

-- level where update may occur
- individual data items/attribute values
- data records/tuples
- files/relations
- entire databases

-- pointer consistency assurance
(system automatically adjusts pointer
references when necessary)

-- Integrity constraints
(validation of updates similar to that
provided for database population)

-- security features (see below)

-- lockout (protection against simultaneous update by 2 or more users)
   - level of lockout (record, file, etc.)
   - type of access precluded (update only, update and query access, etc.)

-- restart and recovery (see section 3.4.3., below)

3.3.4 Data retrieval. Retrieval features of GDMS packages allow users to select and extract data from a database, to order and perform calculations on that data, and to format and display the results. Data retrieval capabilities vary widely in available GDMS software. Characteristics of data retrieval mechanisms are listed below.

Selection - is concerned with identification of the specific data items that are of interest to the user. The selection process may be specified in a "host" programming language through calls to the GDMS, or may be communicated to the GDMS through a "query" language. In the latter case, several query language features that should be considered when evaluating alternative GDMS include:

-- general language format
   - system prompted, with "menu" choices
   - English-like, with keyword commands
   - programming language-like

-- specification of selection criteria
   - conditional expressions (e.g. AGE GREATER THAN 32)
   - logical operators (SALARY EQ 10K AND AGE LE 65)
   - range searching (e.g. AGE BTWN 20, 50)
   - testing for presence/absence
   - alphanumeric selection aids (e.g. partial strings, "don't care" characters, multiple word phrases)

-- predefined queries
   - storage
   - retrieval
   - modification

-- range of query targets
   - single file
   - multiple file

Output - involves the extraction and presentation of selected data instances. While all GDMS packages provide some output facilities, their capabilities range from standard "unformatted" displays to sophisticated report writers. Some GDMS require that user written programs do all output. Many GDMS offer an array of output facilities. Features of output facilities include:

-- method of obtaining output
- user written programs
- selection among options - i.e., response to system prompted dialogue
- standard "unformatted" output

-- display media flexibility
- primary system I/Ø device (e.g., terminal)
- on-line printer
- off-line printer
- graphic display
- other, e.g. photocomposition devices

-- machine readable files produced
- media
- format

-- report formatting
- titles
- headers
- totals and subtotals

Computation - facilities usually in the form of arithmetic and statistical functions allow users to process retrieved data. Built in capabilities most often include the determination of totals, averages, counts, maxima and minima. Some systems also provide limited statistical functions such as calculation of variances and standard deviations. Of course, many GDMS allow user programs to access databases; when these systems are employed, any computation possible in the host (user program) language can be done on data retrieved from the database.

Sorting - is the ordering of retrieved data in a specified sequence using one or more data items as "sort keys". Some GDMS provide sorting facilities that can be invoked on-line; others are available only to batch users. When a GDMS does not have an integral sorting capability, one is often available as a utility program in the operating system library. Features that differentiate available GDMS sorting facilities include:

-- mode of access
- on-line
- batch

-- sort key formulation
- number of data items
- requirement for sort keys to be indexed

-- sequence options

-- user specified collating sequences

3.3.5 User aids. User aids include features designed to assist in search formulation and other aspects of GDMS usage. Types of user aids and their characteristics are listed below:

* Search formulation aids including:

  -- phonetics, i.e., same sounding but differently spelled data names recognized in search requests

  -- synonyms

  -- access to (display of) data element dictionary

  -- "natural English" recognized

* Other GDMS usage aids such as:

  -- "HELP" or "EXPLAIN" command

  -- on-line documentation

  -- "browsing" feature to scan data

3.4 Security Features

GDMS security features provide mechanisms for selectively limiting access to the database, for identification of legitimate users, and for backup, restart and recovery. Each of these security functions is discussed below.

3.4.1 Database protection. Modern GDMS packages provide security "locks" at various levels; that is, classes of users may be allowed to reference specified portions of the database and precluded from accessing others. Some GDMS can differentiate according to type of activity, allowing, for instance, retrieval but not update. Important characteristics of GDMS protection capabilities are described in the following paragraphs.

Protection level - is concerned with the logical levels at which users can specify access controls. Many GDMS permit the user to define security locks at several different levels including: for the entire database; for specific files, relations or realms; and for specific data elements or attributes. More advanced systems may be able to differentiate among records or tuples according to the contents of data fields; for instance, allowing mid-level managers to retrieve personnel records for all those under their supervision but not for their superiors.

Activity differentiation - is the ability to differentiate among classes of users and to provide different database access privileges to each user class. Some users may be allowed only to retrieve, some to retrieve and modify, others to create and delete data, and a few to restructure and define databases.

3.4.2 Authorization mechanisms. Database management systems use different strategies for identifying legitimate users and determining access privileges. Active and passive mechanisms are employed. Active security schemes generally use passwords to differentiate among classes of users. Passive mechanisms recognize and differentiate users based on unique identifiers known to the system software and hardware; frequently used passive identifiers include those maintained by the system accounting/log software and unique identification numbers embedded in remote terminal hardware.

3.4.3 Backup, restart and recovery. To protect against system failure due to physical equipment or software error, most GDMS provide backup facilities and procedures for restart and recovery of lost transactions and database entries. Backup mechanisms also provide the audit trail needed for proper accountability and internal control. Types and characteristics of backup, restart and recovery facilities that differentiate GDMS software are listed below.

* Log tape (audit trail)
    -- record all transactions against database
    -- record only database changes
* Session restart
    -- system maintained working data set
    -- automatic versus user coded procedures
    -- "check point" recording of database and procedural status
* Database recovery
    -- user coded
    -- GDMS provided procedures
* Restart
    -- automatic restart and recovery
    -- user invoked restart and recovery

3.5 Implementation Orientation

This final feature class is a catch-all for GDMS characteristics that do not fall in the previous four categories. The heading, implementation orientation, refers to the fact that many of the features describe implementation details and design decisions that impact the orientation of a GDMS; that is, they affect the type, mode and level of usage for which the systems are most suited. Included in this potpourri of system characteristics are the items discussed below.

3.5.1 Host language versus own language. Most GDMS currently available were initially developed as either host language or self-contained systems with their own user interface language (see "user Interfaces - External Linkages" above). Today many host language systems offer query language processors and many own language GDMS provide procedural language

interfaces. Nevertheless, the initial orientation of the GDMS often is indicative of system strengths and weaknesses.

3.5.2 Mode of use. Closely related to the host versus self-contained dichotomy is the orientation of GDMS toward batch or on-line usage. Host language systems tend to be batch oriented while own language GDMS are designed primarily for on-line query processing. Because GDMS are embedded in and closely tied to operating system hosts they share many of the same performance characteristics including any orientation toward on-line or batch operation.

Available systems differ with respect to the capabilities that are available to the on-line user and those that can be performed in batch mode. Few GDMS provide all system facilities on-line. GDMS capabilities that may be either on-line and/or batch include:

* Data definition
* Database maintenance
    -- population
    -- update
    -- restructuring
* Data retrieval

3.5.3 Re-entrant versus self modifying. When software is implemented so that it does not modify itself it is termed re-entrant. Single copies of re-entrant GDMS program modules can simultaneously serve multiple users. Conversely, if GDMS software is not re-entrant, each active user must have a copy of the database programs resident in main memory.

3.5.4 Teleprocessing mode. Data communications software provides the link between the remote terminal user and the system. If the data communication system used by a GDMS is single-thread, it serves only one user at a time; when the teleprocessing interface software receives a transaction it serves only that user until all requirements have been satisfied and the result is returned to the user. Multi-thread communication software allows for convenient use by more than one user.

3.5.5 Centralized versus dispersed. GDMS have traditionally been designed to maintain one or more large databases at a central processing location. This is contrasted to distributed database environments where data and/or processing facilities are distributed among multiple hardware systems. Distributed databases are receiving increasing attention in universities and research laboratories. While all of the implementation problems associated with distributed processing have not been solved or even identified, it is evident that database software must bear a substantial part of the burden imposed by dispersed storage and processing of data [7].

3.5.6 Design trade-offs. When the array of available GDMS packages is reviewed, it is important to recognize that all systems represent design and implementation compromises. Two important GDMS software design trade-offs are described in the following paragraphs.

Update versus retrieval speed - Fast response to queries is generally achieved through complex data structuring that requires substantial machine resources and time to update. The converse is also generally true; rapid updates can occur only when the data structuring used to speed retrieval is relatively simple and easy to modify.

Response time versus mass-storage utilization - the complex data relationships necessary for fast response to user commands generally require substantial amounts of secondary storage; one modern GDMS generates an object database that is seven to eight (7 to 8) times as large as the source data when all fields are inverted (indexed). On the other hand, one research system that achieves a data explosion factor of less than one by exploiting redundancy does so at the cost of increased processor time and slower response to some user demands.

4. SUMMARY

Many database management systems have been developed in recent years. The proliferation of these software tools makes the selection of GDMS for specific applications difficult. The previous sections presented information designed to assist the potential user of GDMS technology by surveying acquisition sources and describing important features that characterize and differentiate GDMS products. These general guidelines are applicable to specific database management software evaluation and selection tasks.

REFERENCES

1. CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems, CODASYL Systems Committee Technical Report, May 1977.

2. Datapro Research Corporation, A Buyer's Guide to Data Base Management Systems, Datapro Feature Report, Delran, New Jersey, Sept 1976.

3. Fife, D. W. et.al, A Technical Index of Interactive Information Systems, National Bureau of Standards, Technical Note 819, March 1974.

4. Fong, E., J. Collica, and B. Marron, Six Data Base Management Systems: Feature Analysis and User Experiences, National Bureau of Standards, Technical Note 887, Nov. 1975.

5. Koehr, G. J., et. al., Data Management Systmes Catalog, MITRE Corporation Report MTP-139, Jan 1973.

6. International Computer Programs Inc., A Catalog of Saleable Software, ICP Quarterly, Indianapolis, Indiana, 1973.

7. Rothnie, J. and N. Goodman, A Study of Updating In a Redundant Distributed Database Environment, Technical Report CCA-77-01, Computer Corporation of America, February 1977.

8. UCLA Extension Program, Comparative Data Base Management Systems Conference, Course Notes, University of California, Los Angeles, California, 1975.

APPENDIX: CANDIDATE SOFTWARE PACKAGES

The list of software systems contained in this Appendix was compiled over a period of years at the U. S. National Bureau of Standards. Every attempt was made to make this compendium comprehensive and error-free. Because of the dynamic nature of the database management field, however, errors and omissions are inevitable. Any such deficiencies are solely the responsibility of the authors.

The compilation and maintenance of the list was motivated by the desire to assist potential users of database management and related software by providing a catalog of available products. To serve a variety of users, the criteria for inclusion of software systems were broad, and consisted of the following:

1. The software system had to be classified either as a database management system, as a retrieval and report formatting system, or as a bibliographic and text searching system.

2. The software system could not have been designed solely for in-house use; it had to be available to the general public.

3. The software system had to be applicable to a range of information processing problems; that is, it had to be generalized rather than designed for a special processing purpose.

4. The software system had to be operational.

It should be noted that these criteria allowed the inclusion of software not strictly classified as generalized database management systems. Included in the list are software products that are correctly described as bibliographic and text searching systems, or as retrieval and report writing systems. An attempt has been made to label in the 'remarks' column systems falling in these categories to differentiate them from true GDMS. Systems that are available only for minicomputer installation are also noted.

The decision to be inclusive rather than limiting the entries in this appendix to only products fitting some narrow definition for GDMS was motivated by the belief that potential users are not concerned with whether a specific system is strictly defined as a GDMS; what they want is knowledge of candidate software from which they can select tools for solving their particular problems. It is in this spirit that the list is presented.

System trade names, vendors or other system sources, and descriptive remarks appear in tabular form in the following listing. Inclusion of a system in no case implies a recommendation or endorsement by the National Bureau of Standards. Similarly, the omission of a system does not imply that its capabilities are less than those of included systems. The information presented was obtained primarily from existing literature and new products announcements [1-6,8].

Package Name	Supplier
ADABAS	Software Ag
AKSESS	Response Technology Inc.
ASAP	Information Associates, Inc.
ASI-ST	Applications Software, Inc.
BASIS	Battelle Memorial Institute
BRS	Bibliographic Retrieval Service
CULPRIT	Cullinane Corp.
DATA CENTRAL	Meade Technology, Inc.
DATACOM	Computer Information Management Co.
DBMS-10	RAPIDATA
DBMS-11	Digital Equipment Corp.
DIALOG	Lockheed Research Lab.

Computer	Remarks
IBM 360/370 Siemens 4004 Univac 70 and 9000	
Burroughs B5500 Burroughs B5700	
IBM 360/370	
IBM 360/370 UNIVAC 70/40 UP	Report Writing System
CDC 6000 Series UNIVAC 1100 Series Sigma 7 & 9 DEC PDP 10 & 20	Bibliographic Text-Searching System
IBM 360/370	Bibliographic Text-Searching System
IBM 360/370 UNIVAC SERIES 70	Report Writing System
IBM 360/370	Bibliographic Text-Searching System
IBM 360/370	
DEC PDP-10	
DEC PDP-11	
IBM 360/370	Bibliographic text-searching system

Package Name	Supplier
DMARS	First Data Corporation
DML	Computer Sciences Corp.
DMS 1100	Sperry UNIVAC
DMS 170	Control Data Corp.
DMS-II	Burroughs Corp.
DMS-2	General Electric Time-Sharing Service
DS/3	System Development Corp.
DYL-260	DYLAKOR Computing Systems
EASYTRIEVE	Pansophic Systems, Inc.
EDMS	Xerox Information Systems Group (XDS)
EXTRACTO	AQUILA BST (1974) LTD.
GIM	TRW Systems Group
GIS	IBM Corporation

Computer	Remarks
DEC PDP-10	
UNIVAC 1108	
UNIVAC 1100 Series	
Cyber 170	
Burroughs B6700/7700 Burroughs B1700	
Honeywell 6088	
IBM 360/370	
IBM 360/370	Report Writing System
IBM 360/370	Report Writing System
XDS SIGMA 6/7/9 XDS SIGMA 560	
IBM 360/370 Honeywell UNIVAC SIEMENS UNIDATA	Report writing System
IBM 360/370 UNIVAC 1100 Series DEC PDP/11	
IBM 360/370	

Package Name	Supplier
IDMS	Cullinane Corp.
IDS-II	Honeywell Information System
IMS	IBM Corporation
IMAGE	Hewlett-Packard
INQUIRE	INFODATA Systems, Inc.
INGRES	University of California, Berkeley
IRS	SIGMA Data Computing Corporation
MAGNUM	TYMSHARE Time-Sharing Service
MANAGE	Xerox Data Systems
MASTER CONTROL	Lawrence Livermore Lab.
MARK IV	Informatics, Inc
MARS VI	Control Data Corp.
MDBM	Honeywell

Computer	Remarks
IBM 360/370 UNIVAC Series 70 DEC PDP-11 ICL 2900	
Honeywell 60 Honeywell 600 Honeywell 6000	
IBM 360/370	
HP2000/3000	Mini-computer DBMS
IBM 360/370	
DEC PDP-11	
IBM 360/370	Retrieval and Report Writing System
IBM 360/370	
SIGMA 5/6/7	
CDC 6600, 7600	Bibliographic Text-Searching System
IBM 360/370 UNIVAC Series 70	Retrieval and Report Writing System
CDC 6000 Series	
Honeywell Series 60	

Package Name	Supplier
MIRADS	NASA Marshall Space Flight Center
MODEL 204	Computer Corporation of America
MULTIBASE	Computer System International, Inc.
NOMAD	National CSS Inc.
OLIVER	On-Line Systems, Inc.
ORBIT	System Development Corp.
QUERY 5/QUERY 3	AZREX, Inc.
Qwick Qwery	Consolidated Analysis Centers, Inc.
RAMIS	Mathematica, Inc.
REALITY	Microdata Corporation
RECON	NASA, Science and Technical Information
SCORE	Programming Methods (GTE)

Computer	Remarks
UNIVAC 1108	
IBM 360/370	
IBM 360/370	
IBM 360/370	
DEC PDP-10 and Up	
IBM 360/370	Bibliographic Text-Searching System
IBM 360/370 DEC PDP-10 CDC 600 and CYBER Burroughs B3500	Retrieval and Report Writing system
IBM 360/370 UNIVAC 1108 CDC 3150, 6600 Xerox SIGMA 5 Honeywell 600/6000	
IBM 360/370	
REALITY	Mini-computer DBMS
IBM 360/370	Bibliographic Text-Searching System
IBM 360/370 Burroughs Honeywell CDC UNIVAC NCR	Retrieval and report writing system

Package Name	Supplier
SPIRES	Stanford University
STAIRS	IBM Corporation
SYSTEM 1022	TYMSHARE Time-Sharing Service
SYSTEM 2000	MRI System Corp.
TOTAL	CINCOM Systems, Inc.
UL/1	UNIVAC Division, Sperry Rand Corporation
UNIBASE	U. of Florida

Computer	Remarks
IBM 360/370	Bibliographic Text-Searching System
IBM 360/370	Bibliographic Text-Searching System
DEC PDP-10	
IBM 360/370 CDC 6000 Series UNIVAC 1100 Series	
IBM 360/370 CDC Honeywell UNIVAC VARIAN DEC PDP-11 IBM S/3	
UNIVAC 70/45F UNIVAC 70/55F UNIVAC 70/60F	
IBM 360/370 or any machine with full ANS COBOL 74	

GDMS commercially available in Japan (January 1977)

T. Yamamoto
University of Tokyo

Reference : Joho Shori (Information Processing Society of Japan), Vol. 17, No. 10 (Oct. 1976, a special issue on data base systems)

A. Domestic machines, supplied by a mainframe manufacturer

Company	Name of DBMS	Name of Computer System	Nature of DBMS (Origin)
Toshiba (TOSBAC)	IDS/II	ACOS.77 series	CODASYL DBTG
Nippon Electric (NEC)	ADBS	ACOS.77 series	CODASYL DBTG
	IDS	ACOS.77 series	Network (Honeywell)
UNIVAC Japan	DMS/190	OUK 9400 OUK 90 series	CODASYL DBTG
Hitachi (HITAC)	ADM	H8000 series M series	Hierarchical (IMS)
	PDM	H8000 series M series	Network
Fujitsu (FACOM)	INIS	230 series	(Hierarchical?)
	AIM	M series	Network
Mitsubishi (MELCOM)	EDMS	COSMO/700-900	CODASYL DBTG
	DMS-5	COSMO 500	CODASYL DBTG subset

B. Foreign machines, supplied by a mainframe manufacturer

Burroughs (DMS-II)
CDC (DMS-170)
IBM (IMS/VS)
UNIVAC (DMS1100)

C. Independent packages (known to T.Y.)

ADABAS
BASIS
IDMS
System 2000
TOTAL

COST CONSIDERATIONS FOR GENERALIZED DATABASE MANAGEMENT SYSTEMS

D. Deutsch, E. Fong, and J. Collica

Institute for Computer Sciences and Technology
National Bureau of Standards*
Washington, DC 20234 U.S.A.

One important factor that must be considered when evaluating whether generalized database management software should be used is cost. A methodology and a framework based on the application life cycle for estimating costs associated with potential applications of these new software tools is proposed. Important classes of costs are identified and discussed. The problem of comparing costs for database oriented versus traditional software systems is also considered. Finally, budget guidelines for estimating total life cycle costs for generalized database management applications appear in an appendix.

Key words: Application; cost; database management; GDMS; life cycle; methodology.

1. INTRODUCTION

1.1 Motivation

The proliferation of Generalized Database Management System (GDMS) packages in recent years has been accompanied by an especially dramatic increase in use of GDMS for diverse applications in organizations of all types. The large number of available GDMS products and range of their applications merely add to the compexity of the management decision of whether to use traditional or database oriented software tools. The determination of costs and benefits associated with the use of database mangement software is a necessary step in evaluating whether GDMS technology should be applied. While some literature does address cost-benefit aspects of GDMS software [1], there is no accepted methodology for evaluating potential GDMS applications. This paper addresses the cost component of the GDMS application evaluation problem.

---

*This work was supported in part by the U. S. Department of Energy (formerly Energy Research and Development Administration) under Interagency Agreement No. EA-77-A 01-6010, Task No. A050-TI.

1.2 Scope

A thorough analysis of a potential GDMS application requires knowledge of both costs and benefits for the proposed system. Furthermore, determination of the relative trade-offs between GDMS and traditional software development approaches requires analyses of costs and benefits for non-GDMS based systems as well. The size and complexity of the cost-benefit analysis problem for GDMS versus non-DMS based systems preclude development of a complete solution here. A necessary first step, the cataloging of cost factors, and an overview of a methodology for estimating GDMS application costs is presented in the following sections.

The question of benefits derived from GDMS versus non-GDMS based applications is not considered. The reader should be aware, however, of the importance of the benefits dimension when evaluating potential uses of GDMS technology. While costs of GDMS based systems may exceed those for applications using traditional software, database systems often provide information that would not be produced by traditional software. Indeed, the most important determinant of success is often the benefits derived from the GDMS application. Costs must be within reasonable limits, of course; but, many database management applications that cannot be justified on the basis of reduced costs are considered overwhelming successes because of the additional information and flexibility they provide.

1.3 Overview

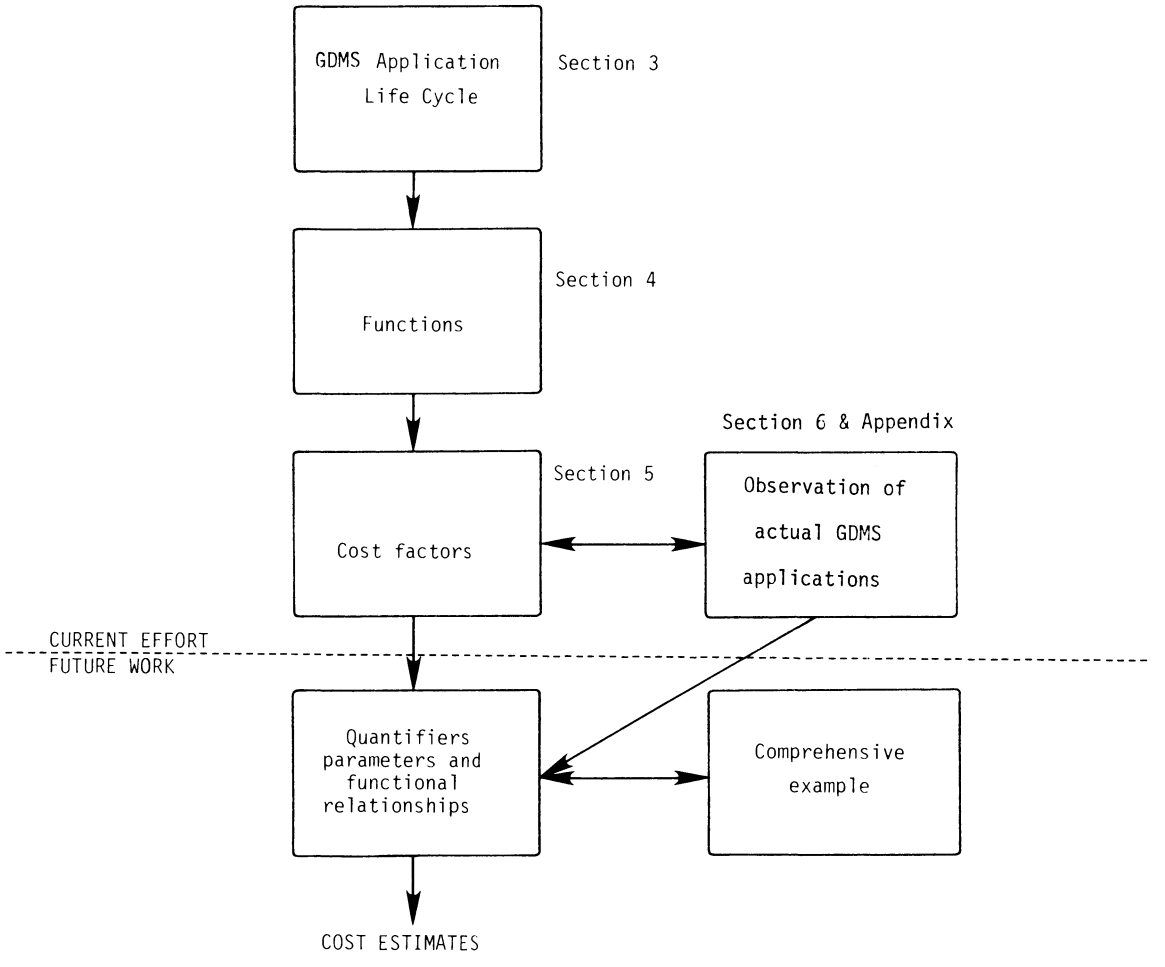
The following sections describe a proposed GDMS cost evaluation methodology based on identification of GDMS life cycle functions and related cost factors. First an overview of the costing methodology is presented. Then the GDMS application development life cycle is discussed. Next functions associated with each of the life cycle phases are listed. Cost characteristics and a framework for costing GDMS applications are presented in the following section. Finally, hypotheses pertaining to costs for GDMS versus traditional software systems are discussed. The appendix contains budget guidelines for estimating GDMS application costs derived from the authors' experience with several actual systems and applications.

2. PROPOSED GDMS COST EVALUATION METHODOLOGY

Any methodology for evaluating costs must identify specific factors contributing to total cost. A well established approach is to describe the process for which costs must be determined first at a high level, and then in increasingly greater detail until quantifiable cost factors have been identified [2]. Such an approach is proposed for evaluating GDMS costs.

The proposed cost evaluation methodology starts with a GDMS application life cycle description of the phases, from perception of need to operation, that all database applications go through. For each life cycle phase, functions are identified. Then, specific cost factors are associated with each function. Finally, units of measure and important parameters for describing cost factors are incorporated in functional relationships suitable for deriving cost estimates.

Figure 1 graphically depicts the proposed GDMS application cost evaluation methodology. Also included on the figure are references to the subsequent sections of this report. The dotted line separates the topics addressed by this paper from those left for future work; no attempt is made to determine quantifiers, parameters or functional relationships, nor to present a comprehensive example of the cost evaluation methodology. The following three sections consider the nature of the GDMS application life cycle, a preliminary list of functions performed within each of the life cycle phases, and approaches to the GDMS application costing problem respectively.

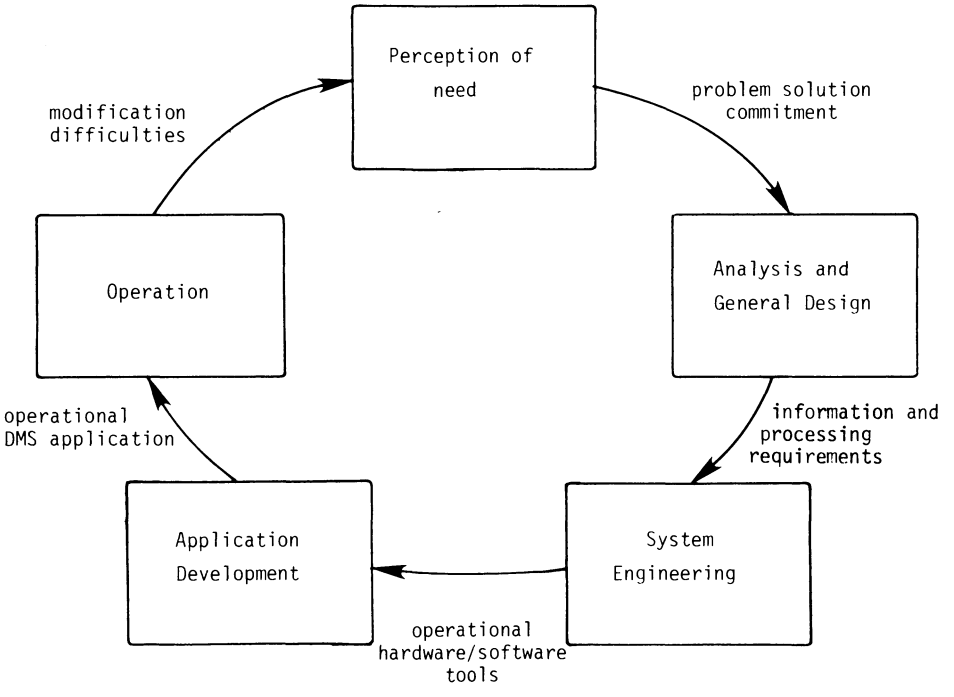


GDMS APPLICATION COST EVALUATION METHODOLOGY
(With Reference to Report Sections)

Figure 1

3. GDMS APPLICATION LIFE CYCLE

The life cycle concept is widely accepted and used for understanding and controlling software application systems. Although there is no agreement on specific phases and terminology, there is a consensus regarding the essential characteristics of life cycles [3,4]. The life cycle phases illustrated in Figure 2 are both consistent with others appearing in the literature and descriptive of GDMS applications throughout all stages of their existence.



GDMS APPLICATION LIFE CYCLE PHASES

Figure 2

The GDMS application life cycle is initiated by a perception of need and passes through four other phases: analysis, system engineering, application development, and operations. Each phase is discussed briefly below.

3.1 Perception of Need

The application life cycle begins when there is a recognition of an information processing problem or need for information. This recognition must be at an organizational level that is sufficiently high to commit the resources required to develop a problem solution. When there is a predisposition toward one solution technique this phase may include a study to determine the feasibility of the proposed approach. Once an information requirement is recognized, this life cycle phase is concerned with establishing a mechanism for developing a system to satisfy the need.

3.2 Analysis and General Design

The analysis and general design phase deals with surveying information requirements and preparing general specifications for processing systems to satisfy these requirements. This life cycle phase includes systems analysis and general design activities similar to those carried out in traditional (non-GDMS based) software development efforts. Information and processing requirements are documented in sufficient detail to support subsequent design, acquisition and development decisions.

3.3 System Engineering

System engineering is concerned with identifying and acquiring the software and hardware capabilities necessary for satisfying requirements identified during the analysis and general design phase. Activities occuring during this phase are all directed toward providing the tools required to solve the information processing problem. For database oriented systems, acquisition of the most suitable GDMS software tool can be a complex and costly problem that has no parallel in non-GDMS based software development projects.

3.4 Application Development

The application development phase includes all of the activities necessary to build an information processing system. Using the hardware and software tools acquired during system engineering, this phase is concerned with the design and implementation of a GDMS application system. Tasks carried out include the creation of data dictionary/directory entries, design and definition of logical and physical database structures, and construction and integration of the application system. Embodied in this phase are the traditional "programming" tasks of coding, debugging, testing and documenting.

3.5 Operation

The last life cycle phase includes all of the activities related to the day-to-day use of the application system. Operation is concerned also with maintenance and modification of the application system. Maintenance must be performed because imperfections ("bugs") are present in even the best software. Modifications are required because organizations and

their environments are dynamic; the systems that serve them can not be static.

Throughout the life of an information system, continual maintenance and modification is necessary in order to respond to changing requirements and to retain an acceptable performance level. Note that the essence of the life "cycle" concept is reflected by the arrow from operation to perception of need in Figure 2. When changes that are necessary to maintain performance and/or to meet changing requirements become too difficult or costly, the cycle begins again with a new perception of need.

4. GDMS APPLICATION LIFE CYCLE FUNCTIONS

The proposed cost evaluation methodology requires the identification of specific functions performed within the framework of the GDMS application life cycle. The objective is to catalog activities for which costs are incurred. Then, the function list can be used as a guideline for those evaluating costs associated with existing or proposed GDMS applications.

In the following sections, functions are listed and briefly discussed for each of the five life cycle phases defined above:

4.1 Perception of Need Functions

4.1.1 High-level recognition of problem. Top management must be aware of an information processing deficiency and must be willing to commit the resources required for developing a solution. While this recognition function may not require any resource expenditures, many organizations establish procedures and managerial committees for monitoring information processing activities. A primary reason for establishing this type of oversight mechanism is to recognize deficiencies before they become critical.

4.1.2 GDMS feasibility study. Often, in addition to recognizing an information processing deficiency, there is a predisposition for employing a particular solution mechanism. A feasibility study attempts to determine the gross feasibility of a particular solution approach through a cursory review of system requirements and their match to the capabilities of the proposed solution. Feasibility studies are frequently carried out for potential GDMS applications. When an existing system made up of traditional software is straining under the load of continually changing requirements, there is often a strong inclination to gain the needed flexibility though the use of GDMS technology.

4.1.3 Establish system development mechanism. After recognizing an information processing problem, making a commitment to find a solution, and determining gross feasibility, this life cycle phase is concerned with establishing a problem solving mechanism. This task may require no resources other than the time to schedule and assign in-house personnel. On the other hand, if assistance is required from outside the organization, some costs may be incurred selecting preferred consultation arrangements.

4.2 Analysis and General Design Functions

4.2.1 Determine information requirements. The analysis and specification of application requirements is an important and time consuming task. Information requirements must be specified and documented to provide a basis for subsequent design decisions. The resulting requirements specifications must indicate what information is required, when and how frequently it must be provided, and must include quantitative and qualitative descriptions of the desired information products.

4.2.2 Develop processing specifications. Based on the information requirements, a general design for a hardware/software system is produced. That is, a processing system capable of satisfying the identified information needs is defined. For database oriented applications, this function includes the specification of necessary GDMS characteristics and features; if a competitive acquisition process is anticipated, GDMS feature descriptions can take the form of a formal Request for Proposal (RFP).

4.3 System Engineering Functions

4.3.1 Hardware/software acquisition. Selecting and acquiring hardware/software systems from among candidate configurations is the first step in the system engineering phase. The specifications prepared during the analysis and general design phase are matched to potential processing tools. Database management system feature requirements are compared to capabilities of available software packages.

The acquisition task can vary from a cursory review of existing resources to a full-fledged competitive hardware and software acquisition. Where hardware resources are fixed, it is important to recognize that GDMS packages frequently require specific hardware features and capacities and operate only under certain versions of operating systems and support software. Consequently, the decision to use any GDMS software that is not currently installed and operational must be considered carefully. A prudent approach is to both install and use, possibly on a prototype application, a new GDMS to assure its viability (see "installation and testing" below).

4.3.2 Documentation and training. Hardware/software tools must be supported by adequate documentation and knowledgeable support personnel. Database management software frequently requires a sizeable investment for training of technical personnel. Instruction and materials may be paid for separately or they may be "bundled," that is included in the cost of the GDMS software. Even when there is no additional charge for training, a substantial commitment of personnel time is required.

4.3.3 Installation and testing. Installation of database management software is a complex task that often requires time, personnel and machine resources similar to those necessary for installing a new operating system. Indeed, installing a database management system frequently requires major modifications in the existing operating system and other support software. Even after an apparently successful GDMS installation, the capabilities of the newly installed software must be thoroughly tested. It is prudent to prepare a benchmark or prototype application to test the range of GDMS capabilities; vendors' claims and promises cannot be substituted for demonstrated performance in the new computer environment.

4.4 Application Development Functions

4.4.1 Data dictionary/directory development. The bridge between information requirements and a GDMS based solution is a comprehensive Data Dictionary/Directory (DD/D) describing all data elements in the database. Ideally, a DD/D is prepared at the beginning of the application development phase or even earlier. [5,6,7] Regardless of the life cycle phase in which it occurs, a data dictionary/directory development effort requires substantial amounts of time and of machine and personnel resources. Entries are required for each identifiable data element; the number of elements in a database of even moderate size is surprisingly large.

4.4.2 Database design. The database design task is concerned with specifying the data structures and storage structures that will be used for the application. Data structures are logical relationships among data items that reflect the users' perception of the database. Storage structures are physical mechanisms used for recording data on secondary storage. The database design process must determine the logical and physical structures that will provide the greatest flexibility and efficiency for current and future applications. Of course, data and storage structures must be chosen from among those supported by the selected GDMS.

4.4.3 Data definition. Data definition is the formal encoding and recording of the database design using the data definition facility provided by the GDMS. Most GDMS have a Data Definition Language (DDL) that is similar to definition facilities in high-level programming languages such as COBOL; DDL's are considerably more powerful, however. After DDL declarations are formulated, they are input to the GDMS definition module. Errors detected by the GDMS are corrected and the processing is repeated in a manner analogous to the compilation of a computer program.

Data definition is complete when the GDMS is ready to accept raw data instances as input. (see "Database population" below). It should be noted that data definitions are not necessarily static. As requirements change and experience is gained, the data definition must be modified to satisfy new requirements and to increase efficiency.

4.4.4 Database population. Database population is concerned with the actual loading of raw data. This initial bulk loading is distinguished from data entry and update facilities. The latter are intended for handling relatively smaller amounts of data after the database has been established. Raw data instances must be recorded in machine readable form before they can be used for population. Many GDMS require specific input formats and/or ordering of source data. Sometimes it is necessary to write custom programs for validating the data and invoking the GDMS population facilities. Database population can be extremely costly in terms of both elapsed time and machine resources. Indeed, for some commercially available GDMS the initial bulk loading facility provides the greatest single limitation on database size.

4.4.5 Application construction. After the database has been defined and populated, specific output requirements are addressed. Construction involves the development of procedures required to produce desired outputs. These procedures may take the form of application programs written in high-level languages such as COBOL, FORTRAN or PL/1 or they may be written entirely in GDMS user language. In either case, "programming" activities including coding, debugging, system testing and documenting are part of application construction. The magnitude of the construction task is dependent on the nature of the application and the GDMS employed. Some applications merely require the establishment of a database that can be queried using a GDMS query language facility. Others have output

requirements that can be satisfied only by developing complex procedures. Because many applications have a range of requirements they employ both ad-hoc and predefined procedures.

4.4.6 Application integration. After its construction, a database application is integrated into the human activities that it was developed to serve. Supporting manual procedures are designed and documented. Personnel are trained to operate and use the application system. The inevitable "bugs" that become apparent during the initial shakedown period must be purged from both manual and automated procedures.

When database management technology is used for the first time, the training of application system users can be a costly and time consuming task. This is especially true when users are expected to interact (possibly for the first time) directly with the computer system. Of course, as personnel become more familiar with the computer and with the GDMS interface they will require less training for each subsequent application system.

4.5 Operation Functions

4.5.1 Data entry. New data must be entered into the database as they become available. For many GDMS based systems, the data entry process is similar to that employed for traditional software systems, involving off-line key stroking and verifying followed by a batch updating process. Other GDMS provide facilities for on-line data entry. On-line data entry is especially useful when the insertion of data points into the database is desirable as soon as they occur. The resources necessary for this task depend on the volume of data and the mode of data entry.

4.5.2 Retrieval. Retrieval activities account for a substantial portion of the resources required for operating a GDMS based application system. Retrieval may be primarily a machine function as in the case of periodic reports generated by COBOL or other high-level language procedures. On the other hand, on-line retrieval using a query answering GDMS can be extremely labor intensive requiring substantial amounts of both personnel and machine resources.

4.5.3 Database Maintenance. A database requires substantial effort to assure that its contents are current and correct. Database maintenance includes the updating of stored data instances to reflect changes. The update capability may be separate from or integrated with the data entry function. Database maintenance is also concerned with guaranteeing the continuing availability of the database. This objective is achieved through the use of restart and recovery procedures and the maintenance of transaction logs and audit trails. Both machine and personnel resources are needed to assure that lost database entries and transactions can be recovered.

4.5.4 Application Maintenance. Application systems change as the organizations they serve change. Manual and computer procedures are modified to reflect changing needs. For GDMS based applications, database restructuring is included in the application maintenance function. Logical and physical database organizations are modified to reflect changing requirements and to improve operational efficiency.

5. COSTING GDMS APPLICATIONS

The next step in the proposed cost evaluation methodology is to identify specific cost factors associated with the life cycle functions described above. To accomplish this objective, characteristics that differentiate and describe costs incurred throughout the GDMS application life cycle are first discussed. Then, a framework and worksheet for determining total life cycle cost for existing or proposed GDMS applications are presented. Finally, some guidelines for applying the proposed cost evaluation methodology appear.

5.1 Cost Characteristics

Total GDMS application cost can be broken down in several ways. Two important classifications are concerned with the recurrence of costs over time and with the dichotomy between personnel and other costs. Each classification is described briefly below.

5.1.1 One-time versus continuing costs. Some costs are incurred only once while others recur, usually periodically, over time. One-time costs are generally incurred prior to the operation phase; consequently, they are sometimes termed "front end" costs. One-time costs include: costs associated with analysis, design and implementation activities; expenditures to purchase hardware and proprietary software products; outlays for training and documentation; and, any other non-recurring expenditures such as those for preparation of physical facilities.

Continuing costs include: yearly or monthly payments for proprietary software; costs for day-to-day hardware usage and software maintenance; and other repetitive costs such as those for supplies. Continuing costs must be evaluated carefully to determine the actual burden that must be carried by a new application. Hardware and proprietary software costs are often step functions; that is, up to a certain activity level there is no increase in cost. For example, basic hardware charges are frequently on a prime-shift basis with extra costs incurred only when usage exceeds eight hours per day. Similarly, GDMS and other proprietary software packages may be priced such that there is no increase in cost unless they are made available on processors other than those for which they were originally procured.

It should be recognized that many costs can not be classified as strictly one-time or continuing, but have both one-time and continuing components. An example of this phenomenon is training. There is both a one-time requirement for training technical personnel in the use of a new GDMS package, and a need for continuing training to enhance personnel skills and to absorb software changes. It is important that the recurrence of costs over time be understood and used when determining total life cycle costs.

5.1.2 Personnel versus other costs. An increasing share of total computer related costs is attributed to personnel. As hardware becomes cheaper and more powerful, this trend is likely to continue. Indeed, the use of tools such as generalized database management systems is often motivated by the desire to substitute machine and software resources for the labor intensive application development activities that occur when traditional software tools are employed. Other non-personnel costs include charges for software and hardware, e.g., computer processor and secondary storage utilization. Costs such as those for supplies and energy, and fees for professional assistance and training also fall in this category.

Computer hardware and support software costs are some of the most important non-personnel expenditures. While total monetary outlays are usually known, allocations of costs to specific users and/or applications are difficult to determine. Most computer system charging algorithms are to a large degree arbitrary and dependent upon installation policy regarding overhead allocation [8,9]. Computer costs can be misleading. For instance, organizations owning their own computer hardware may treat the facility as being essentially "free" when it in fact represents a large investment in capital and personnel. On the other hand, computer time-sharing service charges may seem exhorbitant if one overlooks the support functions included in their costs that do not have to be borne by their customers.

As with the one-time versus continuing cost dichotomy, some costs have both personnel and other non-personnel components. For instance, application construction generally requires substantial amounts of both technical personnel time and machine resources for computer procedure (program) translation, debugging, and system testing. Because in-house personnel costs often have substantially larger overhead factors associated with them than other non-personnel expenditures, it is important that they be recognized when determining total life-cycle costs for GDMS applications.

5.2 Framework for Costing GDMS Applications

Using the life cycle functions and the cost categories described above, a mechanism for determining total life cycle cost for GDMS applications is presented. A worksheet for recording quantities and extending cost factors appears in Figure 3. The worksheet illustrates the various dimensions of total GDMS application cost. It is intended as an example of the proposed cost analysis procedures, not as a definitive statement of cost factors. Worksheet entries are described below.

5.2.1 GDMS application life cycle. The first work sheet column enumerates the life cycle phases and functions described in previous sections of this report. The entries are presented as representative of the type of categories that must be considered. While the life-cycle phases and functions appearing on the illustrative worksheet do describe many GDMS application system development efforts carried out by or known to the authors, they are not the only ones nor are they necessarily the best descriptions for all applications. Indeed, it is expected that life-cycle phase and function categories will be modified as practitioners gain experience with the proposed cost estimation methodology. Organizational and procedural differences should be reflected in the phases and functional descriptions used for estimating total life cycle cost.

5.2.2 Cost factors. Cost factors are enumerated for both personnel and other classifications. The number of man-months is specified for personnel; numbers and descriptions are entered for other cost units, e.g. machine hours, 1000 disk blocks, etc. Unit costs are specified for entries in both the personnel and other categories. Amounts represent extensions of man-months or other units by their corresponding unit costs. Note that for recurring items these amounts represent single period costs only.

5.2.3 One-time or continuing. Costs are identified as either one-time or continuing. For recurring expenses, the number and period (e.g. weekly, monthly, quarterly, etc.) of repetitions is specified.

GDMS APPLICATION LIFE CYCLE	COST FACTORS							ONE TIME VS CONTINUOUS			PV FACTOR	TOTAL COST
Phase/Function	Personnel			Other				one time	continuous			
	no man months	unit cost	amt	description	no units	unit cost	amt		no pds	pd		
PERCEPTION OF NEED High-level recognition of problem GDMS feasibility study												
ANALYSIS AND GENERAL DESIGN Determine information requirements Develop processing specifications												
SYSTEM ENGINEERING Acquisition Documentation and training Installation and testing												
APPLICATION DEVELOPMENT Develop data dictionary/directory Database design Data definition Database population Application construction												
OPERATION Data entry Retrieval Database maintenance Application maintenance												
TOTAL LIFE CYCLE COST												

GENERALIZED DATABASE MANAGEMENT SYSTEMS
APPLICATION COST EVALUATION WORKSHEET

Figure 3

5.2.4 Present Value (PV) factor. Because of inflation and the cost of capital, costs that will be paid in the future cannot be compared with current expenditures. For continuing expenses and for one-time costs that will be incurred in the future, a present value factor is specified. This is a coefficient that converts one or more future payments into current year equivalent amounts. The PV factor should be 1 for a one-time current year expenditure, less than 1 for a single payment in some future year, and greater than 1 but less than 2 for two future payments.

5.2.5 Total cost. Total life cycle costs for each function are determined by multiplying cost amounts by their corresponding present value coefficients. The sum of the resulting products represents a total life cycle cost stated in terms of current year monetary units.

5.3 Guidelines for Applying the Proposed Methodology

The worksheet provides a framework for identifying and quantifying important factors that determine total life cycle cost for GDMS applications. It should be useful as a checklist to assure that pertinent costs are not overlooked.

Any cost evaluation methodology must be applied carefully. Only relevant costs need be considered; sunk costs, that is past expenditures, are not pertinent. For example, costs incurred in the past to acquire, build and/or maintain a particular GDMS are irrelevant; only future costs for using that product versus some other GDMS are relevant. Similarly, costs that are invariant regardless of how the system is implemented should not be considered. Only discretionary costs, those which can be controlled by the relevant decision makers, need be considered.

Finally, it is important to recognize that factors other than cost analyses are instrumental in determining system development approaches. Managerial prejudices, market conditions, and budgetary mechanisms all may impact system development decisions. One contribution of the life cycle approach may be to provide a perspective. Because acquisition costs are only a small part of total cost, an apparently expedient acquisition of a GDMS that is not well matched to system requirements may be more costly than other alternatives.

6. COSTS FOR GDMS VERSUS TRADITIONAL SOFTWARE

The most frequently asked questions pertaining to costs for GDMS-based systems are concerned with their magnitudes relative to those for applications using traditional software. Two factors make these questions almost impossible to answer. Most important is the fact that GDMS based systems perform functions that are not provided when traditional software is used; thus, any direct cost comparisons are for different application products. This is still another demonstration of the close relationship between costs and benefits. A second complicating factor is related to the concept of data independence. The essence of the database approach is the recognition of data as a resource with value that is neither dependent on nor derived from the procedures that reference the data. This concept of separation between data and procedures, termed data independence, is not reflected in traditional software. Consequently, comparisons of GDMS based systems to traditional software applications must be in terms of

specific sets of procedures; any results would appear to be strongly biased in favor of traditional software because one of the most significant contributions of GDMS technology is ignored.

6.1 Cost Relationship Hypotheses

In spite of these barriers and with the caveat that no empirical data will be cited to support the conjectures, we present four hypotheses, labeled H1 - H4, about the relationships among GDMS and traditional software costs. These hypotheses were developed over several years of experience and observation of GDMS application. They are largely intuitive and certainly do not represent a consensus of any group other than the authors.

6.1.1 H1: GDMS costs are concentrated in early life cycle phases. Costs for GDMS based systems are heavier in the the early life cycle phases and lighter in subsequent phases when compared to applications developed using traditional software. This skewed cost curve has been observed for several application systems when their underlying GDMS software was being used for the first time. The concentration of costs in the early life cycle phases decreases rapidly for subsequent applications using an already installed GDMS (See H4 below). Figure 4 graphically illustrated this shift in GDMS based system costs to the earlier life-cycle phases. The relatively higher front-end costs for GDMS applications can be attributed to the complexity of GDMS technology. A GDMS feasibility study is often required. Analysis and design tasks must be pursued in greater depth and often by more highly trained personnel than would be required for traditional software based systems. System engineering includes costly activities related to acquiring, installing and training technical personnel to work with a new GDMS.

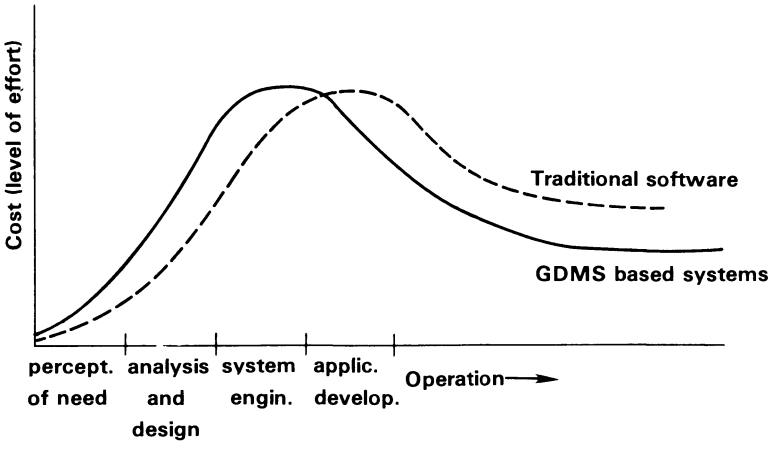
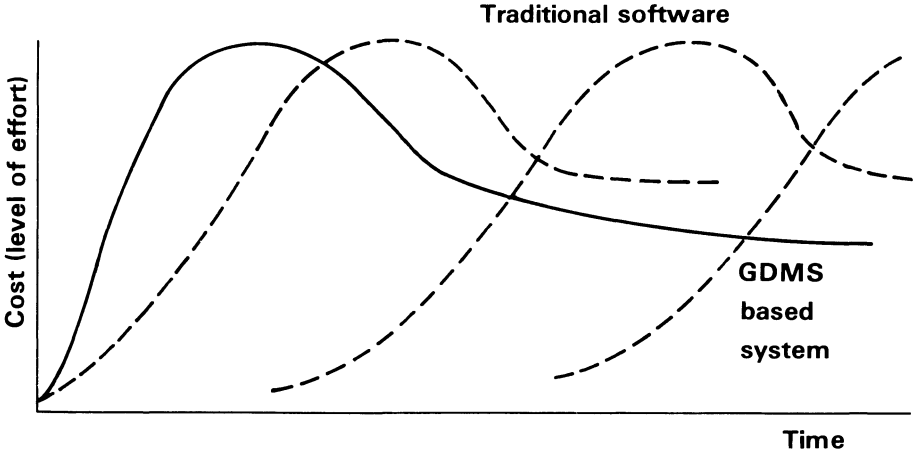


Figure 4

On the other hand, once the GDMS has been acquired and installed, application development often requires less personnel and other resources than traditional software development projects. Application construction, where the bulk of programming for traditional systems occurs, may require little or no effort once the database has been established. This is the case, for example, when a question answering system is established using a GDMS that has its own query language.

6.1.2 H2: GDMS based systems have lower continuing costs. Operation phase costs are lower for GDMS based systems than for those developed using traditional software. This observation, also illustrated in Figure 4, is based on the belief that maintenance and modification activities are more easily accomplished in a database environment than when custom programs must be altered and possibly rewritten. We are emphasizing personnel related operation phase costs and implicitly assuming that non-personnel costs do not differ significantly between the two system development approaches. Some claim that computer hardware and support software costs are substantially greater for GDMS based systems than for alternative software designs. The trend toward cheaper hardware and the increasing portion of total cost attributable to personnel indicate that personnel costs should become increasingly important.

6.1.3 H3: GDMS based systems have an extended life cycle. Because GDMS facilitate maintenance and modification, an application developed using a GDMS should serve an organization longer than a similar system developed in a conventional manner. Traditional software will, after several iterations of modification, fall in disrepair and/or require such major revision that a new system must be developed. GDMS based applications are beter able to respond to changing requirements than other systems. Consequently, use of a GDMS postpones the time when an application has to be rebuilt. Figure 5 graphically illustrates the observation that a single GDMS based application may serve an organization over a time period that would requre several life cycles for systems based on traditional software.

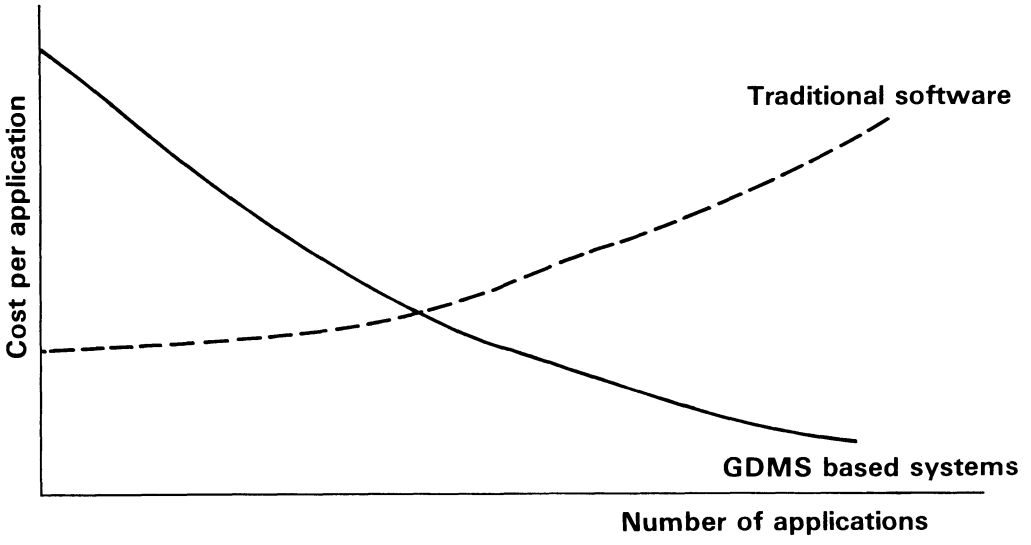


EXTENSION OF LIFE CYCLES OVER TIME FOR
GDMS VS TRADITIONAL SOFTWARE APPLICATIONS

Figure 5

6.1.4 H4: GDMS costs descrease over time. Even the most avid proponents of database technology would not claim cost advantages relative to traditional custom software for a single application requiring the acquisition of a new GDMS. It is the fact that they are generalized that makes GDMS a valuable software development tool. The first application of a particular GDMS may cost more than corresponding traditional software. However, subsequent applications do not have to absorb the substantial front-end costs for specifying, selecting, acquiring and installing the database software. Consequently, total system cost decreases for each subsequent application of a GDMS. Figure 6 illustrates this relationship among GDMS and traditional software costs.

This observation points out the deficiencies inherent in evaluating GDMS costs on an application basis. While each traditional software application system can be viewed as a logically separable entity, GDMS based systems typically span several applications. As noted previously, cost comparisons based on specific sets of procedures (i.e., applications) ignore data independence and the underlying concept of data as an asset; furthermore, cost comparisons on this basis are biased in favor of traditional software systems.



COST PER APPLICATION FOR GDMS VS TRADITIONAL SOFTWARE

Figure 6

6.2 Empirical Results

One reason hypotheses like those presented in the previous section have not been proven or contradicted is the lack of empirical data. Rarely is the same application implemented using both traditional and GDMS software. Although database oriented systems frequently replace traditional software, the new system is generally greatly enhanced, and therefore not comparable to the original. Even in the rare case when directly comparable GDMS and traditional systems are implemented, data are not collected about relative system costs.

The authors wanted to test their GDMS cost hypotheses on an actual application. A National Bureau of Standards (NBS) project for another Federal agency presented a unique opportunity; a system that was initially implemented in COBOL, was reproduced without substantive functional changes using a self-contained query answering type GDMS. Because all work for both implementations was done either by our own staff or by an NBS contractor, we were able to gather comparative cost data. The application characteristics and cost comparison data are described below.

6.2.1 Application Characteristics. The application is a grant analysis and reporting system implemented entirely using the COBOL language. Data describing grant recipients and funded projects are validated, loaded, reformatted, updated, extracted, sorted and displayed in the form of hard copy reports. The system went through several iterations of modification and enhancement to produce reports described by project sponsors.

The prospect of continuing requests for special report outputs provided an incentive for a separate but related effort. The loaded machine readable data was used for populating a database; an interactive query oriented GDMS was employed for this redundant system. Once established, the database facilitated quick response to queries from top level administrators and legislators.

The interactive capability was so well received that an attempt was made to replicate, using the GDMS, all of the outputs produced by the custom COBOL programs. With minor exceptions in the area of printer format, this endeavor was successful. Thus, we had two systems that performed essentially identical functions for which cost data was available.

6.2.2 Comparative Cost Data. Indicators of cost were collected for both systems. Most costs were attributable to either one implementation or the other. In those cases where COBOL facilities served the GDMS system as well, costs were allocated to both implementations. While the resulting figures summarized in Table I are inconclusive, they do show some interesting relationships.

TABLE I: COST COMPARISON OF COBOL VERSUS GDMS IMPLEMENTATION

COST FACTORS	IMPLEMENTATION	
	COBOL	GDMS
TOTAL PERSONNEL TIME	63 MAN-DAYS	69 MAN-DAYS
SECONDARY STORAGE REQUIRED	171K bytes	520K bytes
COMPUTER CHARGES	$8K	$6K

The most surprising finding was that total personnel time in man-days did not differ significantly between the COBOL and GDMS implementations. This can be attributed to the fact that the COBOL validation and load facilities were required by the GDMS implementation as well. Furthermore, although it had been physically installed, this was the first usage of the particular GDMS package.

The size of secondary storage requirements reflects the query answering orientation of the GDMS software. In order to answer queries rapidly, complex secondary indices are maintained by the GDMS; this trade-off between response speed and secondary storage utilization is typical of modern GDMS products.

Finally, computer costs were also surprising. Included in GDMS costs are charges for on-line retrieval, a feature invoked by numerous users. COBOL costs, on the other hand, cover many compilations and test runs and include charges for repetitive data entry, update and report generation cycles.

6.3 Conclusions: GDMS Versus Traditional Software

The lack of empirical cost data about comparable GDMS and traditional software systems precludes definitive general conclusions. Even when data are available, the lack of functional comparability between GDMS based systems and the traditional software applications they replace complicates the cost analysis problem. The hypotheses presented above represent the authors' best intuitive feelings about the relative costs, but they are not proven. Unfortunately, the limited expirical study carried out for this study yielded inconclusive results. The only safe conclusion is that each potential GDMS implementation represents a unique situation that must be evaluated to determine its costs and benefits relative to alternative implementation approaches.

7. SUMMARY

An important prerequisite to the development of a methodology for evaluating costs and benefits associated with the use of generalized database management systems is the identification of pertinent cost factors. The cost identification problem is considered in this paper. A methodology for evaluating GDMS application costs using a life cycle perspective is proposed. The nature of the life cycle is discussed. A preliminary list of functions associated with each of the life cycle phases is presented. Descriptions of cost characteristics and a worksheet for evaluating GDMS application costs also appear. Comparisons among costs for GDMS based systems and those for applications developed using traditional software techniques are considered. Some hypothesized relationships between GDMS and traditional software costs are presented. Empirical data collected for this study is described; the results neither confirm nor disprove the hypothesized relationships. Additional work is required both to better define the cost evaluation methodology and to gather conclusive empirical data about actual system costs.

REFERENCES

1. Selected Literature on Cost Accounting and Cost Control prepared for Automatic Data Processing - A Bibliography for the GAO Task Group Project on Management Guidelines for Cost Accounting and Cost Control for Automatic Data Processing Activities and Systems, Jan 7, 1976.

2. Goldstein, Robert C., Henry H. Seward, and Richard L. Nolan, A Methodology for Evaluating Alternative Technical and Information Management Approaches to Privacy Requirements National Bureau of Standards Technical Note 906, June 1976.

3. Fife, D. W., Computer Software Management: A Primer for Project Management and Quality Control National Bureau of Standards, Special Publication 500-11, July 1977.

4. Benjamin, R. I., Control of the Information System Development Cycle, Wiley-Interscience, a division of John Wiley and Sons, Inc. New York, 1971.

5. Teichroew, D. and E. A. Hershey III, "PSL-PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems" IEEE TRANSACTION ON SOFTWARE ENGINEERING, Jan 1977, Vol SE-3, No. 1 pp 48-48.

6. Sibley, E. H. and H. H. Sayani, "Data Element Dictionaries for the Information Systems Interface", Proceeding of FIRST NATIONAL SYMPOSIUM ON THE MANAGEMENT OF DATA ELEMENTS IN INFORMATION PROCESSING at National Bureau of Standards, COMM 74-10700, Jan 1974, pp 285-304.

7. Leong-Hong, B. and B. Marron, Technical Profile of Seven Data Element Dictionary/Directory Systems National Bureau of Standards Special Publication 500-3, Feb. 1977.

8. Cotton, I. W., "Microeconomics and the Market for Computer Services" COMPUTING SURVEYS, Vol. 7, No. 2, June 1975, pp 95-111.

9. Cotton, I. W., "Cost-Benefit Analysis of Interactive Systems" Proceedings 2nd JERUSALEM CONFERENCE ON INFORMATION TECHNOLOGY, Jerusalem, Israel, July 29 - Aug. 1, 1974, pp 729-46.

APPENDIX - BUDGETING GUIDELINES FOR GDMS APPLICATIONS

The Institute for Computer Sciences and Technology (ICST) of the National Bureau of Standards is charged with providing other U.S. Federal Agencies with technical assistance and consultation to facilitate the efficient use of computer resources. In this role, the authors have gained experience with several GDMS packages and have participated in and observed many Federal system development projects. Based on this experience, budgeting guidelines have been prepared to assist potential GDMS users in estimating life cycle costs. Of course, no amount of experience can be substituted for knowledge of a particular application.

Life cycle costs for GDMS applications are influenced by many variables. Some of the most important determinants of cost throughout the life cycle included:

1. Size and complexity of database
   - number of data item classes
   - complexity of logical structure
   - number of data instances
2. Degree of change from existing processing
3. Level of previous experience with GDMS
4. Pervasiveness of applications within organization
5. Volatility of database
6. Processing mix - query response versus report generation

Because of these and other variables, GDMS application costs can vary over a wide range. Cost estimators must carefully consider each potential application on an individual basis.

Table A-I summarizes cost ranges for each of the life cycle phases based on NBS/ICST experience over the past several years. The reader is cautioned that these data merely reflect NBS experience and are not necessarily applicable to other GDMS applications. Nevertheless, the table does consolidate cost figures from a number of GDMS implementation projects. It can, if used carefully, provide a starting point for developing and evaluating cost estimates.

TABLE A-I: GDMS BUDGETING GUIDELINES

GDMS APPLICATION LIFE CYCLE PHASE	RANGES FOR MAJOR COST FACTORS	COMMENTS
Perception of need	1/2 - 2 person months	More time required when GDMS feasibility study included
Analysis and general design	1 - 6 person months	Extremely dependent on complexity of system. Major organization-wide projects may take many person-years.
System engineering	$60,000 - $150,000 for GDMS software $0 - $20,000 for installation training and documentation 3 - 12 person months for in-house technical personnel	Some GDMS packages bundled with hardware and other software. Others available without charge from universities and US Government Costs smaller for subsequent users of already installed GDMS than for first application.
Application development	1 - 12 person-months	It is not uncommon for large projects to require many person-years.
Operation	1% - 100% of machine resources 1/2 - 3 full time staff over life of system	Major project many require database administration staff of up to 10 people.

An APL Approach to Data Bases

G. A. Martin

Compagnie Internationale de Services
en Informatique, France

ABSTRACT

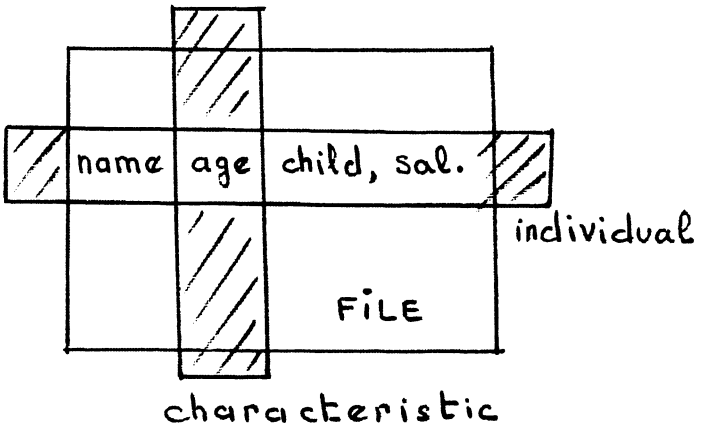
APL (A Programming Language) was developed primarily as a system of notations for describing Data Processing algorithms. It is now a Programming language on many computers but it is also a system and a methodology for programming. Some APL concepts are today fulfilling some of the requirements for tomorrow's Generalized Data Management Systems. This paper will attempt to show how APL can be used either directly in Data Base construction, or as a complement to existing Data Base Management Systems. It is hoped that the controversial way in which some points are made will help the reader to feel adequately the complexity of the problems posed by any integrated approach to data handling. There are no magic solutions.

CONTENTS

1. INTRODUCTION TO THE APL LANGUAGE

Let us consider a set of individuals (employees) with four characteristics: name (up to 20 characters), age, number of children and salary. One may see this set of informations (file) individual by individual (file = sequence of records) or characteristic by characteristic.

In a tabular representation, each individual would be a row (record) and each characteristic a column (file). The first point of view is classical in business and scientific applications (sequential files), while the second is more suitable for retrieval and computations, allowing a global formulation of problems. The representation by characteristics will be called inverted (or dual or transposed) compared to the one by individuals.



Let us define the values of the above characteristics for five individuals by APL expressions directly executable at an APL terminal without previous declarations (the symbol ← is read as ARE or IS):

```
AGES ← 25  32  45  30  50
CHILDREN ← 0  1  2  2  3
SALARIES ← 1800  5500  12000  8000  10000
NAMES ← ',' ⍪ 'JOHN, SMITH, X, Y, Z'
```

AGES, CHILDREN and SALARIES are vectors (lists) of numbers (integers). NAMES is an array (matrix) of characters, each row representing a name (⍪ is an APL function, copied from a public system library, and which transposes a string onto a rectangular array).

The Data (Base) being loaded, one may directly proceed to enquiries and computations

- How many employees?

  It is the number of salaries

  ```
  N ← ⍴ SALARIES
  ```

- What is the sum of salaries?

  It is the reduction (/) by + (sum) of the file

  ```
  S ← +/ SALARIES
  ```

- Mean salary?

  It is the sum of salaries divided by the number of salaries

  ```
  M ← (+/ SALARIES) ÷ ⍴ SALARIES
  ```

The operation MEAN being recognised as usual, one may define a function MEAN with the same syntax as primitive functions, i.e. we extend directly and simply the semantics of APL by adding new user oriented primitives

```
∇ R ← MEAN X
[1] R ← (+/X) ÷ ⍴X ∇
```

One may now directly compute the mean salary by

```
M ← MEAN SALARIES
```

If we recognise that the word OF is simply used as a conjunction linking other words, we may introduce the semantic dummy function OF which corresponds to a transfer of information

```
∇ R ← OF A
[1] R ← A ∇
```

We may now construct our sentence in plain English:

```
M ← MEAN OF SALARIES
```

Changing the names of functions and variables, without changing the semantics, allows us to express this expression in plain French:

```
M ← MOYENNE DES SALAIRES
```

If now we are interested by people more than 30 years old, the expression AGES > 30 will return the value 1 if the individual answers to this description and 0 otherwise. In an example we get the vector 0 1 1 0 1. This vector is said to be logical or binary. The salaries of these individuals will be the result of the reduction by the binary vector (mask) of the file SALARIES:

```
(AGES > 30)      /      SALARIES
condition = mask  ↑      file
                reduction
```

The natural formulation being

```
SALARIES FOR AGES > 30
```

one may invert the previous construction by defining the APL function FOR

```
∇ R ← A FOR B
[1] R ← B/ [1] A ∇
```

where the reduction is done along the first dimension for the case where A is an array (e.g. NAMES); the dimension specification is irrelevant for a vector.

The mean salary of these individuals is simply

```
M ← MEAN OF SALARIES FOR AGES > 30
```

For those people who find the symbol > too mathematical, one may define the function GREATER THAN.

To understand better some APL Data Base Systems, it is interesting to show the direct APL formulation of the above expression:

1. we first compute the mask — MASK ← AGES > 30
2. we check how many individuals — N ← +/ MASK
3. we compute the desired value

M ← (+/ MASK / SALARIES) ÷ N

where MASK/SALARIES defines a restricted access to the file SALARIES.

Binary vectors and arrays (masks) are often used in APL because of the efficiency in processing: a binary number is represented as a single bit of memory and logical operations (AND, OR, NAND, NOR, NOT...) are very fast. On the other hand, it is natural to introduce a separation between search and computation, if only in order to avoid unnecessary computations.

Updating our Data Base will be easy to do by using direct APL expressions or natural language formulations, e.g.

'SALARIES' BECOME (SALARIES + 200) WHEN
(CHILDREN ≥ 3) AND (SALARIES < 10 000)

When an expression has been entered, the user gets an answer immediately: the correct result or an error message. The APL language was designed to run in an interactive environment: APL expressions are interpreted at execution time, and all entities are accessed through descriptors. This principle will be used for APL Data Bases to ensure Data Independence: files (in core or on external devices) will be described by tables and the various tables will be related by common roots (logical names, rather than physical addresses). This is the general approach of Relational Data Bases /1/, and the basic relational operators (JOIN, DIVIDE, ...) are easy to implement in APL.

The interpretative structure of APL is well illustrated by the primitive ⍎ (Execute) which allows dynamic execution of expressions. Combining this with the primitives ⎕CR (canonical representation of a function, i.e. a character form) and ⎕FX (fixing a canonical representation onto a function), it is possible to record on files not only data (variables) but also programmes (functions) so as to record actions. These actions may correspond to 'hooks' in the main programme, initiating predefined action when given situations arise.

It is not our purpose to introduce all the concepts and primitives of APL, even in the context of a Data Base.

It is enough to say that APL contains a rich set of primitives: arithmetic (+, -, x, ÷, circular, exponential ...), relational (=, ≠, <, ≤, ≥ ...), logical (and, or, nand, nor ...), selection (indexing, take, drop ...), structural (grade up and down, transpose, rotate ...). These primitives operate on (rectangular) arrays.

The semantics of APL being so rich, it will be difficult to choose the right way to implement a given system (the same is true of a good GDMS). The most important choice concerns the data structures /2/. In our previous example, we have chosen a representation by characteristics, i.e. by a set of vectors managed in parallel. In some other circumstances (production management, inventory ...) an array representation is more suitable, as in /3/.

Trees and lists are not usual in APL because they need normally to be processed element by element, which may lead to long running time: loops are not natural in APL. However, lists can be very useful in conjunction with recursive functions, and by definition any APL function may be recursive. Such structures are used in graphics (databases of images), system description (hardware configurations, plants, reliability, networks, ...) and so on. But genuinely recursive structures are not so frequent as is supposed by too many database specialists. In any case, they are easy to implement in APL /4/ and seem not to be primitive notions.

In building APL databases, we try to keep in mind the basic virtues of APL: uniformity, brevity, generality, simplicity and familiarity /5/. Starting from good data structures, it is very easy to construct a set of primitive notions (APL functions and variables) directly related to the user (in management, econometry, crystallography, reliability, etc.) instead of forcing the end-user to apply general (EDP) concepts not directly related to his daily problems. The dialogue will be adapted to the user, or personalized: a set of primitives, natural language, prompting, computer aided instruction, ...

Keeping compatibility with APL (syntax and data structures) will allow us directly to use the APL libraries within database systems: plots, graphics, data analysis, linear algebra, ...

Since APL is interpreted, it can be chosen as a machine language. Various APL microcodes are already implemented on minicomputers such as the MCM or the IBM 5100, or on larger size computers like the IBM 370/148. The most recent improvement to APL was the introduction in 1973 of Shared Variables. Shared Variables allow the description of concurrent processes, and provide an interface with resources external to APL. Their use will be illustrated in the following sections.

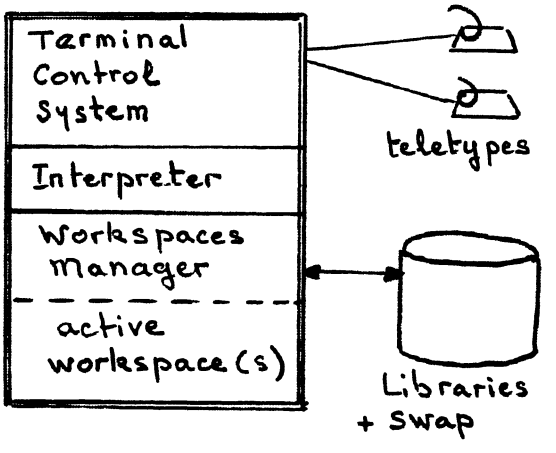
The computers to be built in the near future (gigasystems) will require hardware languages to describe architectures based on array processing and multiple processors. One can imagine that (an extended) APL would provide a solution.

2. INTRODUCTION TO APL SYSTEMS

Let us take as our example the APL-SV system (IBM) used at the French Atomic Energy Commission (CEA). Basically, an APL system is composed of three (logical) parts: a terminal control system, an APL interpreter and a workspace manager.

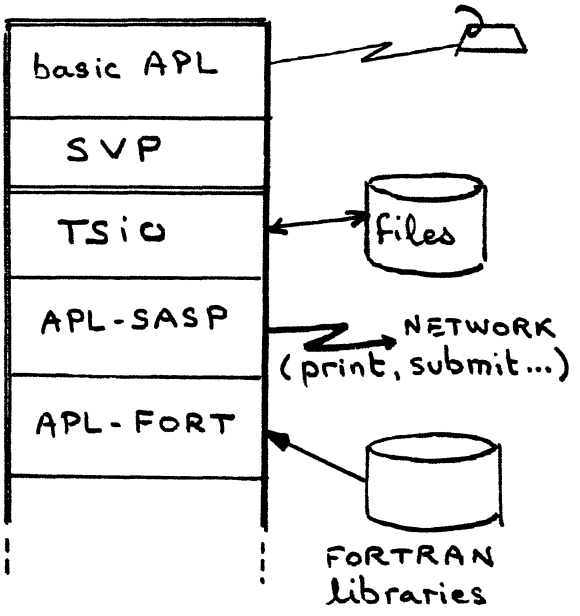
To each active user is attached a workspace containing its data and functions.

Workspaces are organized in (public and private) libraries. Normally, two active users may exchange informations only through system commands (LOAD, COPY, SAVE) outside the control of the APL interpreter (i.e. system commands are not part of the language). The need for external support (files) was recognized very early: a file system, APL*PLUS, was made available in



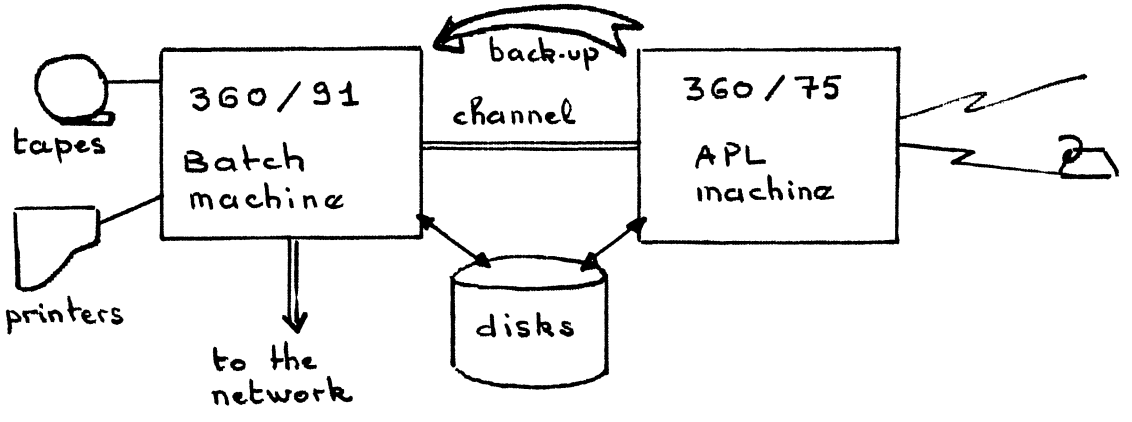
1970 by commercial firms (I.P. Sharp and STSC). In 1973 an important extension of APL was made available: the Shared Variables, a general mechanism to exchange information between two processors, APL users or internal processors. An APL (SV) system may be viewed as consisting of the previous basic components extended by a Shared Variable Processor (SVP), controlling communications between the concurrent processors and a set of internal processors called Auxilliary Processors (AP).

One such auxilliary processor, called TSIO, is provided with APL-SV to assess external files (tapes, disks, printers) or O.S. resources (batch submission).



At the French CEA, we run an APL-SV system on a dedicated machine, and have developed an AP providing an interface with the CEA network (IBM 360/91, 370/168, 2 x 370/158, CDC 7600, 6600, CYBER 173 ... implemented around Europe) which allows the APL user to enter data or produce results at any node of the network. To improve the efficiency of our system, we have also implemented an AP, called APL-FORT, to link directly under APL some FORTRAN routines; this was done mainly to avoid a tremendous effort in converting existing routines: matrix inversion, eigenvalue computations, FFT, data analysis and so on. All these existing facilities will be used in conjunction with databases. New developments are under study for graphics and specific database functions.

One should notice that the APL dedicated machine is directly linked channel-to-channel to another IBM computer (the 360/91) and that they share their disk drives.



With such a configuration it is possible to combine conversational (APL) programmes and batch processing in order to achieve good performance. Of course it is transparent to the end-user, who will simply use APL functions put in public libraries. For example, to submit a FORTRAN job

contained in the file FORT for execution on the CDC 7600, the user will copy the required function from the library 4 APLSASP

```
)COPY 4 APLSASP SUBMIT
```

and enter the command

```
SUBMIT 'FORT, DESTINATION = 7600'
```

Workspace limitations: direct use of files

In the CEA system, the size of workspaces is limited to 80,000 bytes. This is not enough to manage data bases. Coming back to our example of a personnel database, one may define the characteristics of an employee as the result of a read operation from an external file. Suppose we have constructed the file PERSON as a set of records, the first giving the ages, the second the salaries, and so on. Using the file in direct access, one may define the functions AGES and SALARIES by

```
∇R ← AGES
[1] R ← GET 1 ∇
```

```
∇R ← SALARIES
[1] R ← GET 2 ∇
```

All the expressions we have developed(such as MEAN OF SALARIES FOR AGES > 30) remain valid. Within a workspace of 80K it is then possible to handle a large amount of data with the same simplicity as with pure APL data, any information (characteristics) being loaded (read) only when necessary.

Exchange of information between users

Two or more users may exchange information through shared variables or shared files. Although control on shared variables is easy to implement (by using APL primitives), it is not so easy to control the sharing of files, which must be protected against simultaneous read and write operations. In this case, the shared variables can offer a very elegant solution either used as semaphores or as the basis for an APL written auxilliary processor [6] [7].

Another way to control file access is to put in the file a special record which will be dynamically executed during the opening of the file (restriction to specific records, authorisations to users, etc.).

Security and integrity

Any APL application is implicitly a multi-user application. Several protection levels are provided by the system

- a password may be associated to the sign-on number
- a password may be associated to each workspace to protect LOAD and COPY operations
- a function may be locked so that the semantics are unchanged but the code cannot be visualised: for example, we may protect the SALARIES function by testing whether the current user has been authorised to access it

```
     ∇ R ← SALARIES
[1]  R ← ⍳0
[2]  → 0 IF~ ⎕AI [1] ∈ 1224 2048 2100
[3]  R ← GET 2
[4]  ∇
```

where we give an empty answer if the user, known by his accounting number ⎕AI [1], is not a member of a list of authorised users. The function being locked (∇), nobody will know, except the file administrator, what kind of checking is done.

It is then easy to have users with different profiles: one is able to update while another is limited to read specific files or records.

Extending the capabilities of APL for use with Data Bases

Any APL system with shared variables (in particular the IBM systems) is extensible by auxilliary processors. There are several examples of DBMS under APL [8] or in conjunction with APL like the IBM bridge between APL and IMS [9] or the model for Relational Data Bases [10].

Since 1973, several people are working on an extension of APL to arrays of arrays [11] [12] [13] [14], which would make it possible to use the APL primitives either on rectangular arrays or on arrays from which elements may be arrays. Such structures are obviously important for databases, where they may represent trees, lists or hierarchies. For example, an inverted file may be seen as a vector of vectors. One may hope that such extensions will be made available before 1980 (several models are already operational).

Under the new time-sharing system VSPC of IBM, it is possible to have virtual workspaces, i.e. to run large workspaces up to 16 millions of bytes (this possibility was available earlier under VM-CMS). In such a context, a direct APL approach may be better than any DBMS. For example, using virtual workspace is (10 times) faster than using external files. One may also notice that the new IBM access method VSAM will be fully accessible through an auxilliary processor under APL, including (up to 14) alternate indices.

3. APL METHODOLOGY

Building a Database is not just a question of organising data or using a given GDMS. A Database system cannot be dissociated from its use: it is a tool within a problem solving environment and one has to follow rules in its use as with any other function of the related problem. One may recall here the Scientific Problem Solving approach [15] as presented in [2]

(1) Preparation - Phase 1 (Brainstorming)

- Deciding on objectives
- Analysing problems
- Gathering information

(2) Preparation - Phase 2

- Organising information
- Inducting
- Simulation model (evaluation)

(3) Realisation

- Programming (implementation)
- Documentation (in parallel)

with continuous feedback and feedforward.

In a recent Database implementation [16], we were faced with an interesting point of view, classical in chemistry: any plant realisation has to go through a 3 steps evaluation process, similar to the one above.

1. laboratory experiment
2. prototype
3. factory

The 'factory' in this analogy would be a system for information retrieval and data analysis on a very large historical commercial file (50 millions of bytes for 2 years). The approach was the following:

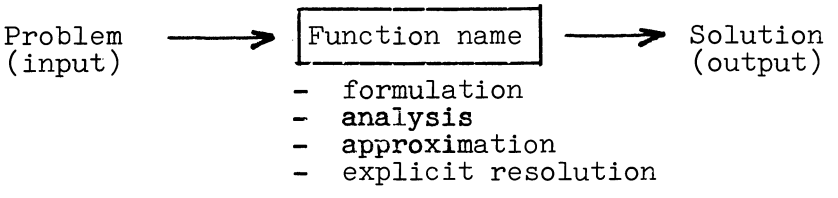
1. Demonstrations of DBMS capabilities on a reduced set of informations. A special APL DBMS was built within a week (using APL libraries), and designed to ensure good performance. A proposal was made to the customer with estimations of exploitation costs.

2. A decision was taken to use this APL solution for a limited period of time (6 months) and a subset of the file of the whole firm. An operational system was implemented within 2 months (for an effort of 3 man-months) with all DBMS facilities: security, integrity and privacy, in a multi-users environment. Our initial cost estimates have proved very accurate. We are now investigating the possibility of interfacing this DBMS with the BMDP statistical package (written in FORTRAN).

3. The decision on full-scale implementation will be taken in October 1977.

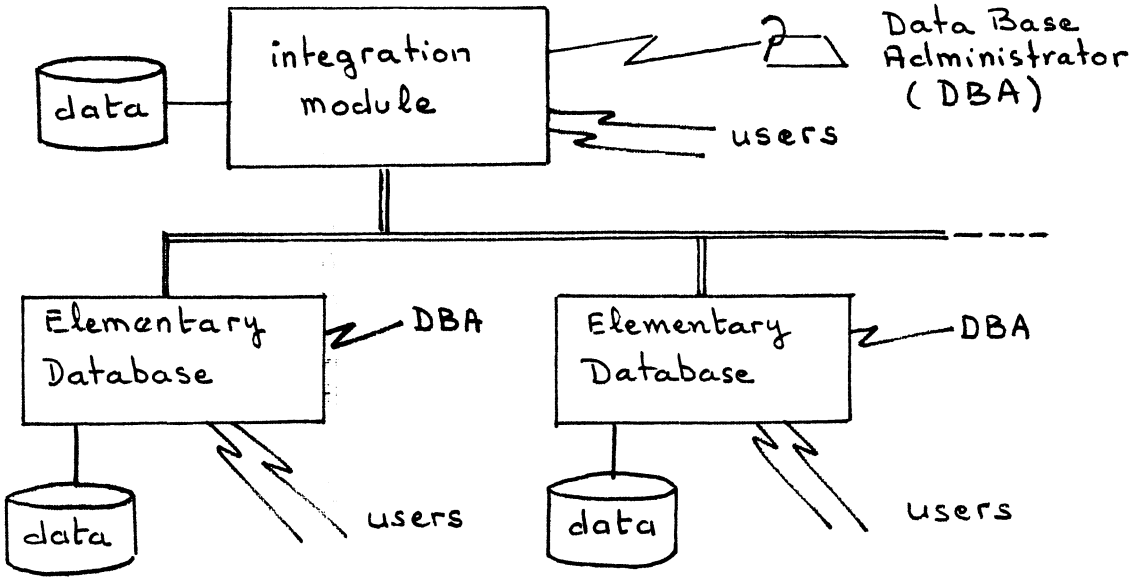
A methodical approach of this kind can help in avoiding many mistakes in choosing and implementing a Data Base Management System.

In Rational Programming, conception, formulation, resolution and interpretation (of results) are four steps of the same Problem Solving process. They will obey the same general rules as above, and will lead to a functional decomposition of any complex problem onto a subset of simpler problems. Conception and Programming will naturally be conducted top-down.



For example, a complex Database will be decomposed onto simpler interconnected Databases. The decomposition will continue until elementary structures (for the GDMS) are reached. The implementation of a given elementary Database will be decided on practical, economical and human criteria: manual or computerized, on local or central computer, real time or not, conversational or not, using APL or another system ...

When deciding the separation onto subsystems, one has also to define the interface (data convention and communication protocol). The final integration will define the Database structure to be implemented.



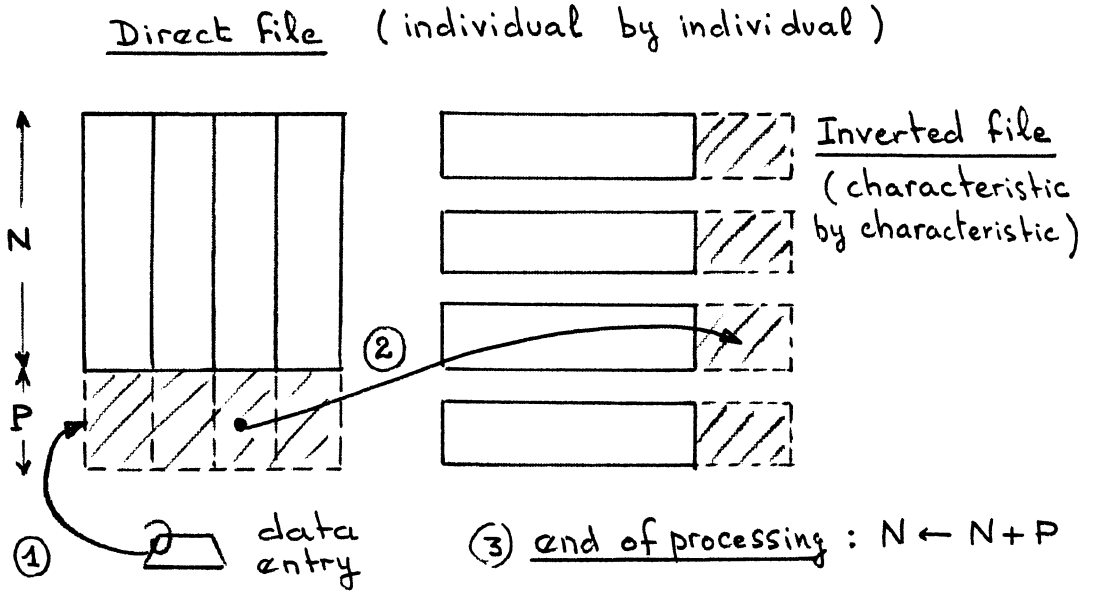
With such a picture in mind, which is the way we implement our APL Databases, it is easy to achieve important requirements for GDMS: security (each DBA may choose his own protection rules), data coherence (each DBA will have the responsibility of his own Database), performances (each Database will be implemented with the best suited techniques). A more detailed discussion will be found in [19]. It is also easy to personalise each elementary Database.

Interconnection between Databases will be achieved by Relational Models in relation with files of actions (APL functions). Indirect access to an elementary Database will generally be made possible through predefined queries (macros) which are the direct result of the various query languages.

4. SOME FEATURES TYPICAL OF APL DATA BASES

4.1. Basic data structures

The elementary Databases mentioned in section 3 will be represented by parallel arrays depending on data types (binary, integer, real, character). Each array will be represented by a direct file (individual by individual) and an inverted file (characteristic by characteristic), the first being used for sequential processing (visualisation of one individual, sorting, lists ...) and the second for information retrieval and global computations.



It is simple to explain the update mechanism:

1. Adding new data

   - the input data are checked (conversational) and added sequentially to the direct file (1)
   - the inverted file is extended characteristic by characteristic (2)
   - the counter N is incremented (3)

   In case of failure, the process may be very simply restarted at the interruption or at a previous level.

2. Removing information

   For a large amount of data, it is not realistic to compress the file each time an individual data element is removed. One will associate to the file a binary vector indicating by 1 if the individual is active (present) and by 0 if passive (removed or not yet created). The next input data element will take the first free space available.

3. Replacing information: this is done by indexing.

In most cases (unless it is justified for economic reasons) we do not need to keep both files, and the data will be represented by the set of inverted files only. The extent area of the sequential files will be retained as a buffer for data entry.

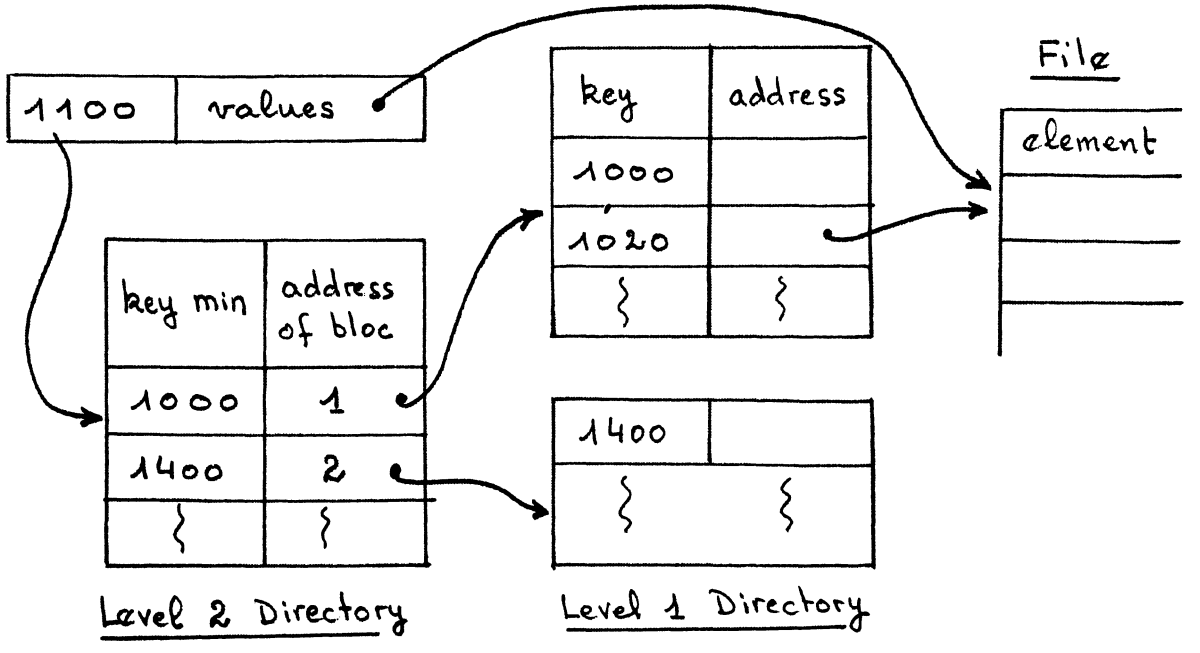
Such an organisation requires only sequential and direct access methods which are directly provided by APL-SV. To improve performance, we have implemented in Saclay a Fixed Block Access Method in which a physical block (set of records) is read into memory and used for input/output operations (with some optimisation). As long as this block contains all the information needed, it is kept in core. Swapping is done when another block is needed. This technique is very efficient when we need to access simultaneously several elements of the file.

4.2. Directories

Keyed access will be done through directories. The physical support may be seen as a set of blocks (e.g. tracks) of a given length. Some blocks will contain the association tables (key, address of element) and are referenced as "level 1 directory". Each block will be ordered by increasing keys. To describe the physical implementation of these blocks, one needs another association table

(key minimum, address of level 1 block)

where "key minimum" is the first key in the corresponding block. Such an association table will be referenced as part of the "level 2 directory". The "key min." are also ordered by increasing values.



One may have as many directory levels as necessary to address all elements in the file. The manipulation of directories is easily done with APL primitives.

Removing or replacing an element or a key is a trivial operation. Adding new elements is also simple: we read the level 1 block concerned and add (catenation) the new couples (key, address). The resulting array is ordered by increasing keys. If the number of rows is smaller than the capacity of a block, the updated block is written back on the disk. Otherwise, we split the block into two (half) blocks, one taking the place of the old one and the second the first place available in the system. The level 2 directory is then updated.

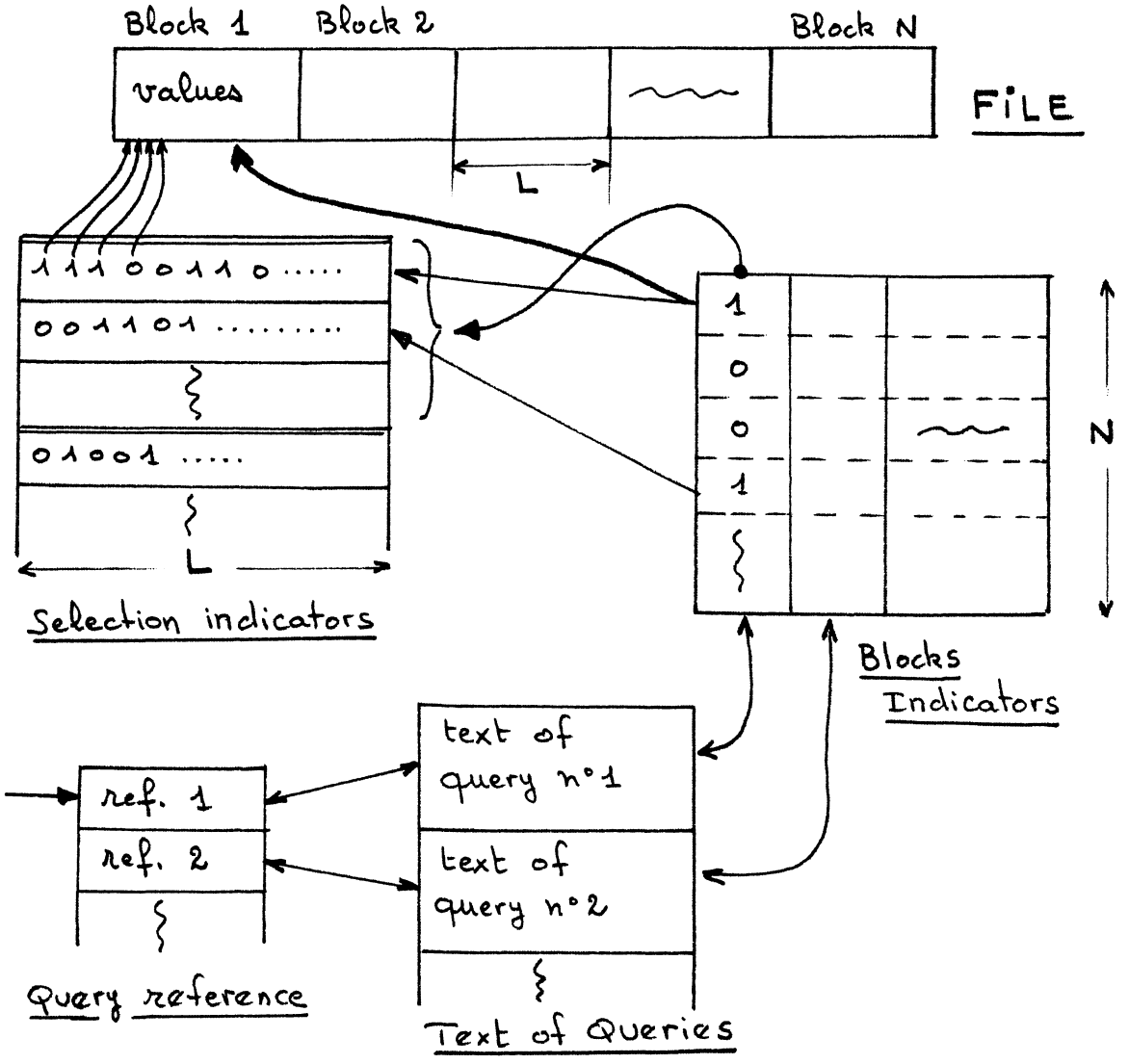
If BUF contains the new couples (key, address) and LEVEL is the sequence of the block numbers for the levels 1, 2, ... directories, the following APL (recursive) function carries out this operation, and offers a more formal description of it:

```
      ∇ BUF UPDATE LEVEL;I;KEYMIN
[1]     →0 IF 0=ρLEVEL                          test if end of process
[2]     BUF←BUF,[1] GET LEVEL[1]                catenation
[3]     BUF←BUF[⍋BUF[;1];]                      sort
[4]     KEYMIN←BUF[1;1],LEVEL[1]                level 2 element
[5]     →UPDT IF CAPACITY≥1↑ρBUF                test if capacity exceeded
[6]     I←⌊0.5×1↑ρBUF
[7]     NEXTBLOC SET(I,0)↓BUF                   write the last elements
[8]     KEYMIN← 2 2 ρKEYMIN,BUF[I+1;1],NEXTBLOC another level 2 element
[9]     NEXTBLOC←NEXTBLOC+1                     new addres of free bloc
[10]    BUF←BUF[⍳I;]                            keep the first element
[11]  UPDT:LEVEL[1] SET BUF                     replace old bloc
[12]    KEYMIN UPDATE 1↓LEVEL                   update next level
      ∇
```

The optimisation of this programme is left as an exercise to the reader (e.g. don't update level 2 if the keymin is not changed ...)

4.3. Binary access tables (queries)

With very large databases /16/, it is possible to avoid scanning all the database by associating to each file (characteristic) a corresponding set of binary indicators.



The file is structured by blocks. To each block is associated a binary value (Block indicator) to show whether the block contains relevant information or not. The set of block indicators related to a given query is represented as a column of the Block Indicators Table or BIT. For a given column (i.e. a query), each 1 will correspond to a binary mask for accessing the information in the related block of the file. Each binary mask will be represented as a row in the Selection Indicators Table or SIT. The number of rows associated to a given query is exactly the number of active blocks (i.e. the number of binary 1s in the corresponding column of BIT). A special record (query number 99999) will be associated with the current active selection. This record is initialized by any selection (query) through a decision tree (definition of the initial domain of a selection). The arrows in our figure are not real pointers because the relative locations of elements in arrays are implicit pointers in APL.

The result of any selection (text and binary mask) may be recorded in that system. Two results may be combined directly by logical operations (AND, OR, NAND, NOR, =, ≠...) to produce a new result without reformulating the original queries. This operational access method /16/ was fully written within one week and has proved to be very efficient.

4.4. Some data reduction techniques

The original purpose of data reduction is obvious: to save storage space and often computer time. For example, 5 characters may be coded as one integer (4 bytes), some statistical series may be represented by their Pearson's coefficients ($\mu, \sigma, \beta_1, \beta_2$) or their first cumulants (Kendall's expansions), a rectangular array may be reduced to its limited singular values decomposition (which is widely used in Data Analysis).

In general, these techniques are transforming a set of data of a given type into another set of data of another type, the transformation being nearly inversible.

An interesting paper /20/ considers the representation of verbal information as unique numbers. When this paper was published (1972), no emphasis was put on distributed computing. Using the proposed techniques allows teleprocessing to be used in a very efficient way by implementing on both sides (emission and reception) common dictionaries instead of full verbs, while relatively small dictionaires are enough for many applications. Such a technique may ensure good data privacy and be implemented cheaply using new technologies. The technique could be very useful for some Databases.

A more promising method of data reduction is the representation of complex data structures by programmes which may be dynamically executed. Let us give a typical example. To compute the reliability factor of an electronic component you may use an analytic model or an approximate table /21/. Let us consider an analytic model for a condenser, equivalent to the two pages of data shown in Fig 1 (Annex).

To compute a given factor λ you have to know 8 parameters and to do the proper computations. If PAR designates a vector with 9 components, the first value giving the condenser type and the 8 others being the computation parameters, the required value may be obtained by

```
LAMBDA ← ⍎ TYPE, ⍕ PAR
```

where TYPE contains the literal 'COND' and ⍕ transforms a numeric vector onto a character string. The result is got by executing (⍎) the constructed executable expression. If now the variable TYPE is used as a key for accessing the proper APL function, one may get the result by

```
LAMBDA ← ⍎ (⎕FX KGET TYPE,⍕ PAR /1/),'_',
                ⍕ 1↓PAR
```

where FX is used for fixing a function represented as a matrix of characters (canonical form). The APL function (model) corresponding to the two preceding pages is simply:

```
     ∇ R←COND19 P;MP;MS;C;ΔT;⎕IO
[1]    ⎕IO←1
[2]    C←(P[4]+1.414×P[5])÷P[6]
[3]    ΔT←10×2⍟1⌈100×(P[5]÷P[6])×(10000÷1⌈P[8])*-0.6
[4]    R←1E¯10×(1+(C÷0.4)*5)××((P[3]+ΔT+273.16)÷380)*20
[5]    MP← 1 1.1 2 4 4 4 15 18 20 ∘.× 3 1 0.3 0.1 0.03
[6]    MS← 0 0.1 0.2 1 1 1 5 6 10
[7]    R←(1E¯9×MS[P[1]])+R×MP[P[1];P[2]]×P[7]
     ∇
```

which takes exactly 384 bytes of disk storage.

4.5. Automatic Generation of Programmes

The most tedious tasks in Data Processing are data input and output. There are several ways to automatize them: Data Definition Languages, Report Generators ...

After writing several similar applications, one may recognise that some operations are similar. One may define generalised programmes (e.g. macros) to assist the user and built-in Programme Libraries (or packages) adapted to a specific range of applications.

Another approach is to use some kind of definition to generate directly the programmes performing the required functions. One may use two types of definition: dialogue [22] and decision tables [23]. With the first approach, the user has to call a programme and the dialogue will be conducted (and assisted) by this programme to produce the required APL code. The code generated may then be used directly or adapted to similar situations. It is interesting to note that the generated programmes are free of bugs. Two "automata" are available: one for entry, the second for reports.

In the second case, one has to define the decision table associated to the required function. An APL written translator will produce code in 3 possible languages: APL, PL/1 or COBOL [24].

A similar approach was used to simulate microprocessors and produce automatically microcode from APL models [25]. This may be useful in the context of distributed Data Bases.

5. CONCLUSIONS

In the author's opinion, APL has proved to be the major contribution to Data Processing of the last decade. It is not the solution to all problems, but it does seem to offer a valid approach to solving day-to-day problems, and an important tool for defining the solution to larger ones.

The usefulness of APL for data management will depend on the user's systems environment as well as on the nature of his problem. The full range of options likely to be of interest is :

- Programming in a high level language such as COBOL, PL/1 or FORTRAN, using the file management facilities of the operating system.
- Programming as above, but using a 'host language' GDMS.
- Programming in APL, using the operating system file manager.
- Use of APL as the host language interfaced to a GDMS (which may or may not itself use APL).
- Use of a self-contained GDMS.

(i) An integrated approach to data handling. Whatever the data handling problem, there are lessons to be learned from the approach to data management represented by GDMS : a collection of related data should be stored in an integrated way, and its structure shown explicitly ; the system should allow flexibility in the use of data for purposes not foreseen when the files were constituted; control of security and integrity must be programmed into the system; data redundancy should be avoided where possible, and in any case controlled by the system.

(ii) Availability of APL. Although APL interpreters are available on many computers (notably IBM and Burroughs) GDMS facilities in APL or linked to it are in practice so far generally available only as part of the APL service offered by commercial networks. They are offered in Europe by several other bureaux as well as CISI, and in North America most notably by STSC (Scientific TimeSharing Corporation).

(iii) Should you use a GDMS at all ? Below a certain threshold of logical complexity, size and requirements for flexibility and simultaneous time-shared access in updating or retrieval, the benefits offered by a GDMS may not be worth their extra cost. Rather than try to accommodate your problems within a closed package (GDMS or other) it may be better to use:

- a good methodology
- a good programming language, and why not the most productive, APL?
- good system support allowing all necessary extensions
- the appropriate program libraries, which may be written, as with the CISI APL system, in a mixture of various languages

and adapt the EDP facilities to your problems. In some cases the use of APL may imply choosing a link to a time-sharing service rather than buying your own smaller computer.

(iv) When do you need a GDMS ? The most obvious need is in a context similar to the business environments for which GDMS have been developed : for a data base which is frequently updated, especially where updating is done 'simultaneously', on-line, by several users. It takes a relatively sophisticated GDMS to provide the privacy protection and data base error protection and recovery facilities ('security and integrity') which this use demands. Their centralised Data Definition facilities are useful in avoiding data inconsistencies and programming errors in a complex data base, or one where structure is often modified.

(v) What are APL's particular advantages, alone or as a host language ?

- Host language flexibility. This question becomes critical when we look at the host language environments available for use of CODASYL systems (such as IDMS) in a scientific environment. COBOL in particular is scarcely a good programming language for scientific applications. Rather than listings, what is required are graphics, data analysis (principal components analysis, correspondence...), or data comparisons (variance analysis, forecasting....), all easier to program in APL than in COBOL.

- Speed of implementation. A comparison extremely flattering to APL can be found in /17/: improvements in programmer productivity are claimed by a factor of 3 relative to PL/1 or 4 relative to COBOL.

  This can be important where non-standard questions are frequently asked of the data base : scientists may be just as impatient as managers, and unless such new questions can be answered quickly the questioner may lose interest in the reply before it is available.

- Performance. As part of a recent congress /18/ a competition was held to compare various DBMS on a Financial Administration problem. Only three of the systems invited agreed to compete : Télésystèmes with COMPOSIT 77, CSS with NOMAD, and CISI using APL with direct access files.

  The total time to solve the problem was the same for the three systems. The time to load and check the data was shorter in APL (with programs specially written for that application) than with the two DBMS (with their Data Definition Languages). The CPU time for executing specific search questions was between 5 and 10 times smaller in APL than with the two GDMS : a strong argument in favour of using APL if the application had to be run on a daily basis.

REFERENCES

/1/ E. F. CODD "A Relational Model of Data for Large Shared Data Banks" - CACM - Vol 13 - n° 6 - June 1970

/2/ P. BRAFFORT and G. MARTIN "Rational Programming: an introduction" - Proceedings of the SEAS Spring Technical Meeting - AALBORG (Denmark) - April 1975

/3/ M. F. O'BRIEN "Inventory Management using APL/SV" - Proceedings of the SEAS Spring Technical Meeting - ST. ANDREWS (Scotland) - April 1974

/4/ H. KATZAN "Representation and manipulation of Data Structures in APL" - ESA microfiches L 2494 (indexed in the NASA system) - 1971

/5/ A. D. FALKOFF and K. E. IVERSON "The Design of APL" - IBM J. Res. Develop. - July 1973

/6/ R. H. LATHWELL "System formulation and APL shared variables" - IBM J. Res. Develop. - July 1973

/7/ S. BARON: "Utilisation des Variables Partagées" - Journées APL de l'AFCET - PARIS (France) - June 1977

/8/ J. VERCAMMEN "APL and Database: Problems and Solutions" - SEAS - GUIDE APL Technical Conference - BRUSSELS (Belgium) - March 1977 - Publications of the Ministry of Economic Affairs

/9/ W. L. ALLISON "APL and IMS, and interactive approach for the user and his Database" - Proceedings of the SEAS STM - ST. ANDREWS (Scotland) - April 1974

/10/ M. BERGEN and others "An Environment for the interactive evaluation of Scientific Data" - Project Report - IBM Heidelberg Sc. Center (Germany) - January 1975

/11/ J. A. BROWN "A Generalization of APL" - PhD Thesis RADC - TR - 73-182 Tech. Report - June 1973

/12/ Z. GHANDOUR and J. MEZEP "General Arrays, Operators and Functions" - IBM J. Res. Develop. - Vol 17 - n° 4 - July 1973

/13/ S. BARON "An Experimental approach to new APL Data Structures" - Proceedings of the SEAS Anniversary Meeting 74 - ZUERICH (Switzerland)

/14/ W. E. GULL and M. A. JENKINS "Recursive Data Structures in APL" - Submitted to CACM - 1977

/15/ K. GREGORY "The Management of Intelligence" - McGraw Hill - 1967

/16/ R. GAUDELAS, D. CARON and G. MARTIN "SVP: Un système d'interrogation en temps réel d'une banque d'informations commerciales" - Journées APL de l'AFCET - Paris (France) - June 1977

/1/ Y. LEBORGNE "APL: A Productive Personal Language" - Proceedings of the SEAS Spring Technical Meeting - AALBORG (Denmark) - April 1975

/18/ JIIA - X - PARIS (France) - June 1977 - Report to be published in September 1977 in 01-Informatique

/19/ J. C. LERALLE and G. MARTIN "The World Nuclear Power Plant Data Base of the French Atomic Energy Commission" - in this report (BERKELEY Meeting - Oct. 1977)

/20/ W. HAGAMEN and others "The representation of verbal information as unique numbers in APL 360 - IBM System Journal - n° 4 - 1972

/21/ RECUEIL DE DONNEES DE FIABILITE - Centre de Fiabilité - C.N.E.T. - LANNION (France)

/22/ D. CARON "Automata in Business Programming" - Proceedings of the SEAS Anniversary Meeting - ZUERICH (Switzerland) - Sept. 1974

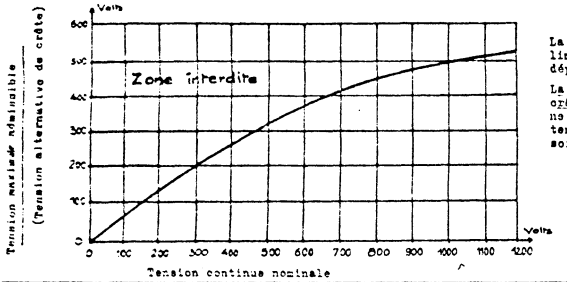
/23/ H. J. MYERS "Compiling Optimized Code from Decision Tables" - IBM J. Res. Develop. Vol. 16 - n° 5 - Sept. 1972

/24/ IBM I.W.P. 5796 PJB "APL Decision Table Processor" - Program Description & Op. Manual - SH 201924

/25/ L.P.A. ROBICHAUD and others "INTERACTIVE tools for Research and Education in Micro-Programming" - EUROMICRO - NICE (France) - June 1975 - and UNIVERSITE LAVAL-QUEBEC.

OBSERVATIONS -	CONDENSATEUR A FILM DIELECTRIQUE (MYLAR) FIABILITE CONTROLEE	CCTU : Néant Modèle : Néant MIL : 14157 B Modèle : CPV

— LIMITE de VALIDITE —



La courbe ci-contre est une limite que l'on ne doit pas dépasser.

La tension alternative de crête + la tension continue ne doit pas dépasser la tension continue nominale soit :

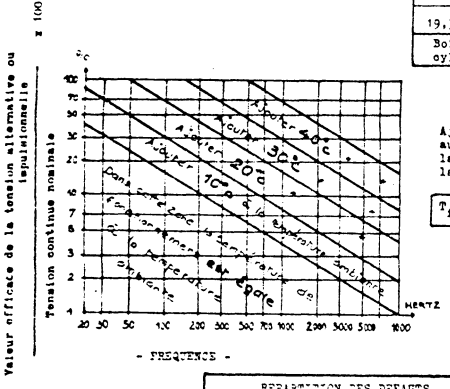
$U_{Max.} + UC < à\ U_{CN}$

Π_{FC}

Π_B

Niveau de fiabilité	Taux de défaillance correspondant	Π_{FC}
L	$50000 \cdot 10^{-9}$/h	3
M	$10000 \cdot 10^{-9}$/h	1
N	$1000 \cdot 10^{-9}$/h	0,3
P	$100 \cdot 10^{-9}$/h	0,1
S	$10 \cdot 10^{-9}$/h	0,03

Dimensions du Boîtier				Π_B
Diamètre		Longueur		
mm	Pouce	mm	Pouce	
7	0,275	19	0,75	0,2
7,9	0,400	22,2	0,875	
10,1	0,400	22,2	0,875	
10,1	0,400	34,9	1,375	0,3
14,3	0,562	41,3	1,625	
17	0,670	41,3	1 625	1
19	0,750	54	2,125	
19,3	0,780	60,3	2,375	
Boîtiers rectangulaires ou cylindriques plus grands				2



Ajouter la température correspondant au domaine considéré de cet abaque à la température ambiante pour obtenir la température de fonctionnement

$T_{fonct.} - T_{amb.} + \Delta T$ → Abaque λ_b

REPARTITION DES DEFAUTS	
Circuit ouvert	: 5 %
Court circuit	: 95 %

FIGURE

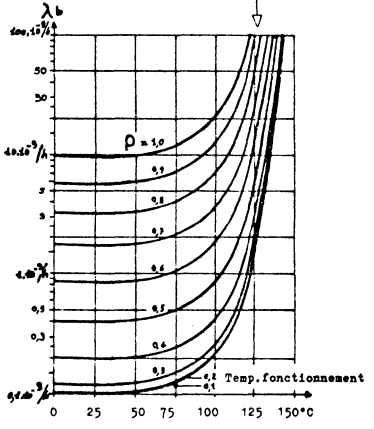
OBSERVATION : -	CONDENSATEUR A FILM DIELECTRIQUE (MYLAR) - Fiabilité contrôlée -	Mil : C 14157 B Modèle CPV

PARAMETRES NECESSAIRES -

- Température de fonctionnement
- Tension appliquée
- Valeur crête de la tension alternative appliquée
- Tension nominale
- Dimension du boîtier
- Niveau de fiabilité
- Environnement

CALCUL DU TAUX DE DEFAILLANCE

$$\lambda = \lambda_b \times \left[\pi_B \times \pi_{FC} \times \pi_E\right] + \Sigma_E$$



λb en fonction de la température de fonctionnement et du facteur de charge ρ

ENVIRONNEMENT	π_E	Σ_E en 10^{-9}/h
Laboratoire	1	0
Satellite (en orbite)	1,1	0,1
Au sol (fixe)	2	0,2
Au sol (portable)	4	1
Au sol (mobile)	4	1
Avion (en habitacle)	4	1
Avion (hors habitacle)	15	5
Satellite (au lancement)	18	6
Missile	20	10

FACTEUR DE CHARGE ρ

$$\rho = \frac{\text{Tension continue appliquée + Valeur crête de la tension alternative appliquée}}{\text{Valeur nominale}}$$

$$\rho = \frac{U_C + \sqrt{2} \times U_{eff.}}{U_N}$$

1

MODELE MATHEMATIQUE de λb

$$\lambda_b = 1 \times 10^{10}\left[\left(\frac{\rho}{0,4}\right)^5 + 1\right] e^{\left(\frac{T}{363}\right)^{20}}$$

T : Temp.de fonctionnement en °K

PART II

GDMS FOR SCIENTIFIC DATA: REQUIREMENTS AND SPECIALIZED SYSTEMS

PARTIE II

SGBD POUR DONNEES SCIENTIFIQUES : BESOINS ET SYSTEMES SPECIALISES

THE STRUCTURE OF R&D INFORMATION -- SOME OBSERVATIONS

A. A. Brooks

Computer Sciences Division
at Oak Ridge National Laboratory
Union Carbide Corporation, Nuclear Division*
Oak Ridge, Tennessee, USA

ABSTRACT

This paper is intended for the potential DBMS user who is not knowledgeable in the area of information and data structures. A simple pictorial but hopefully useful approach is taken which discusses the structure of information from the point of view of the user problem. These structures are then related to some simple considerations of data models. Any reader who wishes to pursue the subject of data models further is referred to Date[1] as a straightforward text and to Chen[2] as a more advanced reference. Knuth[3] is a reference for graphs and trees.

INTRODUCTION

This paper presents a simplified discussion of the structure of R&D information from the view of the information use or user rather than from the view of computer processing. It contains a discussion of

---

*Prime contractor for the Department of Energy.

structure which is logically inherent in the information or which is deemed as a pragmatic "natural" organization by the user. An attempt is made to lead the user through a simple exposition on structures and relate them first to examples and then to the nature of a DBMS which might be required to meet his needs. All examples of uses are taken from experience at the Department of Energy installations in Oak Ridge which include the Oak Ridge National Laboratory and the R&D projects of two high-technology production plants.

In addition to the inherent logical structures and pragmatic organizational structure, the influence of the query language, access methods and analysis algorithms on the structures is also discussed. Since a specific DBMS may process only one information structure, the paper discusses how the more complex structures may be represented as simpler structures likely to be found in a general data base management system (GDBMS). The effect of the use of structures in user processing modules is discussed. A number of secondary considerations such as the influence of structure on data base size and processing times are mentioned.

No effort is made here to deal with the data storage structures except as they may pose problems for the user in retaining and processing his inherent structures. In this we limit ourselves to data base models which are generally available - hierarchical, relational and network.

1.0 Inherent and Pragmatic Structure

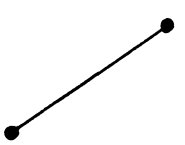
1.1 Inherent structure is the logical associations which exist between actual entities of a problem. It cannot be changed without misstating the problem or at least stating it incompletely. The system chosen to process such a problem must preserve this structure in the sense that all its properties are recoverable as these are essential to the solution of the problem.

1.2 Pragmatic structure is the logical associations created between entities in a problem for the convenience or efficiency of its solution. The pragmatic structure used may be one of several alternatives, but it must enable the system to preserve the inherent structure. Pragmatic structures, due to the nature of computers and other reasons, are often combinations of simpler structures accompanied by rules for reconstruction of the original structure.

1.3 The next section describes structures which may be either inherent or pragmatic.

2.0 Mathematical Structures

Greatly simplified, structure is the pairwise logical association between "things". These things are called entities, and they must have a well-defined membership in a disordered collection, called an entity set. The pairwise logical association or relationship between entities may be nondirectional or directional. Thus, a simple association may be pictorially represented as a line segment connecting two nodes (or points), i.e.



Nondirectional

Directional

Information items known as attribute values may be associated with either entity (node) or with the relationship (segment).

Rather than develop formal definitions of information structures, we shall define and illustrate pictorially a variety of structures which may express the logical associations inherent in the user problem or pragmatic to the organization of his data. We shall speak of nodes and segments rather than entities and relationships.

The most significant attribute of a node is its identification or label; of a segment, its direction if any. Additional attributes of nodes are also referred to as labels while attributes of segments are known as weights. Segments are identified by the labels of the involved nodes with an implied direction where required.

We now define and pictorially illustrate a variety of structures.

2.1 Cyclic Structures

2.1.1 A graph is a set of nodes, S, and a set of pairs of nodes from S, P(S), i.e. segments.

G = <S,P(S)>
S = A,B,C,D,E (nodes)
P(S) = AB,BC,DC,DB,DE,EB,AE (segments)

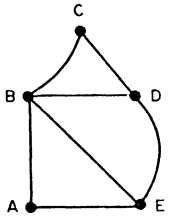
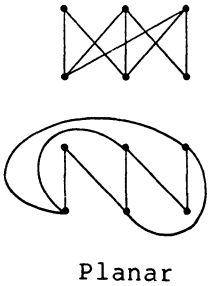
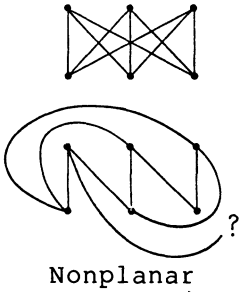


Figure 1

A graph is planar if it can be drawn in a surface topologically equivalent to a plane without intersecting segments, for example:



Planar



Nonplanar

Figure 2

A path is a set of connected segments between two nodes and a circuit is a closed path. A region is an area encompassed within a circuit of a planar graph, i.e. BCDB in Figure 1. For each planar graph, there exists a unique dual graph expressing in its segments the adjacency of the regions of the original graph.

2.1.2 A directed graph or digraph is a graph whose segments have a preferred orientation and is defined by a set of nodes (S) and a binary relation on S, R(S). (A binary relation is a set of ordered pairs of elements (nodes).)

D = <S,R(S)>
S = A,B,C,D,E
R(S) = AB,BC,DC,DB,DE,EB,AE

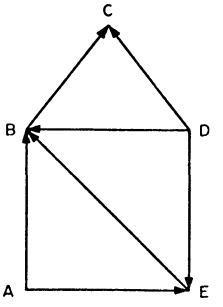


Figure 3

The graph and digraph represent the structure among a set of entities.

2.1.3 The labeled graph or digraph is introduced in order to include the attribute values of the entities.

G = <S,P(S),L(S)>
S = A,B,C
P(S) = AB,BC,CA
L(S) = L(1,A),L(2,A), ... L(n,A)
L(1,B),L(2,B), ...
L(1,C),L(2,C), ...

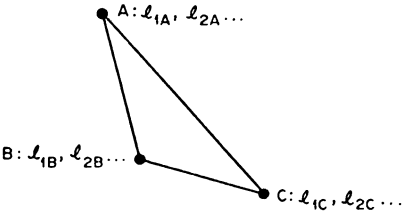


Figure 4

We have illustrated a rather simple set of values as the label and others of arbitrary complexity are permissible. Generally speaking, the labels have logical associations among their own elements, usually one of the simple structures depicted later.

2.1.4 The weighted graph or network is introduced in order to include attribute values of the segments.

```
   N = <S,P(S),W(P)>
   S = A,B,C
P(S) = AB,BC,CA
W(P) = W(1,AB),W(2,AB), ...  W(n,AB)
       W(1,BC),W(2,BC), ...
       W(1,CA),W(2,CA), ...
```

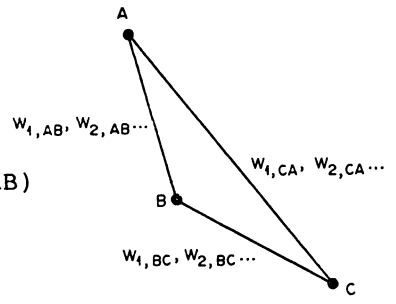


Figure 5

Again we have depicted rather simple weights for each segment, and more complex ones may be required for a specific problem.

2.1.5 Labels and weights may be applied simultaneously to a graph or a directed graph. An example of a labeled, weighted, planar digraph is the following:

```
   D = <S,R(S),L(S),W(R)>
   S = 1,2,3,4
R(S) = 12,23,34,41
L(S) = x(1),y(1),a(1)      W(R) = W,M,P(W),P
       x(2),y(2),a(2)             N,M,P(N),P
       x(3),y(3),a(3)             E,M,P(E),P
       x(4),y(4),a(4)             S,M,P(S),P
```

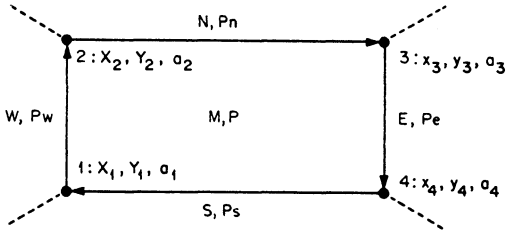


Figure 6

The dashed lines separating quadrants have not been specified in the definition for the purpose of simplicity. The figure could well represent a small geopolitical unit with regions M, N, E, S, and W having populations P, P(N), P(E), P(S) and P(W) with latitude, longitude and altitude at each node. Geopolitical boundaries, geographical boundaries, and linear features such as rivers, roads, canals, and transmission lines have produced the most sophisticated and largest, although planar, structures.

Examples of nonplanar graphs in data management have been production process flows, food chain kinetics, and stereoisomerization of cyclic compounds. These problems generally are not considered for processing by a GDBMS due to their highly specialized nature; but as they become more commonly used, it may be desirable to incorporate management of such data within a more general system.

2.2 Acyclic Structures

Acyclic structures are graphs which have only one path between any two nodes. Their occurence in experimental data is far more common than that of the cyclic structures of which they are a specialized case. We illustrate several below.

2.2.1 A free tree is a nondirected graph having only one path between nodes.

T = <S,P(S)> Unique Path Between Nodes
S = A,B,C,D,E,F
P(S) = AB,BC,DB,FB,AE

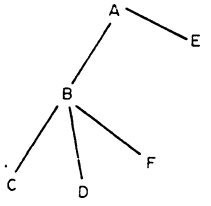


Figure 7

2.2.2 An oriented tree is a free tree with oriented (directed) segments between nodes. Such a tree is shown in Figure 8. It can express "one to many" and "many to one" relationships between nodes.

T = <S,R(S)>
S =A,B,C,D,E
R(S) = AB,BD,BE,CB

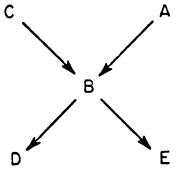


Figure 8a

2.2.3 The rooted oriented tree is an oriented tree with only one incoming path per node and a unique root node which has no incoming paths. It can express only "one to many" relationships which is often a serious limitation for some problems. It is also called a hierarchy and is the most common irregular structure found in data base systems.

T = <S,R(S)> a) Unique Root Node
S = A,B,C,D,E b) One Incoming Path/Node
R(S) = AB,BC,BD,BF,AE

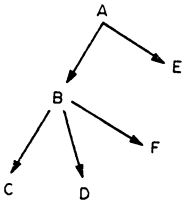


Figure 8b

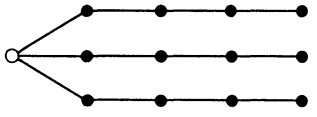
Trees must be labeled and weighted in order to carry attribute values for nodes and segments. The subtrees may be ordered, usually counterclockwise, about the node; and the order may be indicated by a top down left branch first (preorder) traversal sequence of nodes.

The rooted tree structure is accommodated directly or indirectly by most DBMS. Some systems may permit multiple hierarchical descriptions of the same set of nodes which pragmatically may remove the "many to one" restriction.

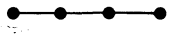
2.3 Regular Structures

Many R&D data bases have very regular structures; namely, vectors and multiply-dimensional arrays. These can be viewed as regular trees.

2.3.1 Array (as regular rooted tree)



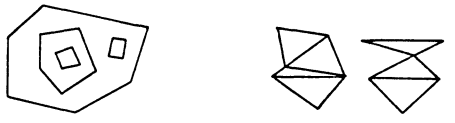
2.3.2 Vector (as a simple rooted tree)



Vectors and arrays are so legion in R&D data that no examples are necessary. It should be sufficient to say that most large computational problems deal with these regular structures.

2.4 Collections of Structures

2.4.1 All of the above structures, graphs, trees, arrays and vectors, may occur as collections of separate (i.e. disjoint) structures which have no connections and no specific order between the isolated subgraphs. Some problems, such as contour lines, do impose some secondary association between disjoint structures. Two examples of disjoint graphs are shown.



Again, these are usually processed by special software systems, not by a DBMS.

2.4.2 Many experimental or observational data bases can be organized into a collection of rooted trees. This occurs frequently when information is uniquely associated with independent conceptual individuals which form a population. Such a collection is termed a forest.

Forest - A Set of Trees

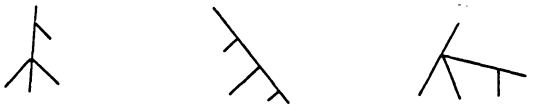
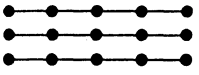


Figure 9

The occurrence of this form is rampant in science, perhaps because it coincides with the statistical view of a population of individuals. Some examples are: field observations, experimental animals, sample analyses, bibliographic files, and card catalogs.

2.4.3 A simpler case is the n-ary relation or collection of vectors.



etc.

This structure is the basis of the relational data base model.

2.4.4 The simplest case is a collection of atomic data elements which is called a set and could be represented without structure as:

Note that a set in general may consist of nonatomic elements and that all the collections shown here are indeed sets.

Our enumeration of structures is at an end. The user should be able to relate the structure of his problem to one or more of those presented. There is opportunity for great complexity. Fortunately, most complex structures can be expressed in equivalent simpler structures which can be used to produce the same results given the necessary algorithm. In fact, the ultimate structure of most computers is linear with some artifice used to represent more complex structures.

3.0 Atomic Data Elements

Atomic data elements are representations of attribute values which have no structure in the sense they cannot be subdivided without loss of meaning. Scientific data management requires several kinds especially if both display and computations are to be carried out. We list without comment several which seem necessary: text strings, numeric strings, machine words (binary integers and floating point), bytes, bits and bit strings. Atomic data elements may be fixed length or variable length and both are desirable if not essential in a scientific data base management system (SDBMS).

4.0 Simple Representation of Complex Structures

It is readily apparent from the definitions of complex structures in terms of simple structure, i.e. associated pairs, that complex structures can be represented by simple structures. It is important, however, that the simple structure be chosen so that the inherent features of the complex structure can be maintained or readily reconstructed from the simple structure.

4.1.1 The basic process for reducing the complexity of a structure is to break a segment and place the name of one node as a label in a data field of the other node. In this manner a graph can be reduced to a tree or collection of trees and a tree to a collection of vectors which form one or more relations. This latter process, called normalization, results in recording the identifying labels of the complete path from root to data node along with all data at the node. This process tends to inflate a data base in size but has desirable attributes with respect to data base update. For a very deep hierarchy (rooted tree), this inflation may be a serious problem for large data bases.

4.1.2 Structure within a data base can exist within a single record, between a few records or a single structure for an entire data base. Often this distinction in the data base model is not made clear. Structure within a single record is almost always expressed as a relative address within memory. Structures between records (which are usually simple structures, i.e. vectors) are usually expressed as device address links to the next logical record. Records stored in single large hierarchies usually use nested directories or indices. The network data base model provides for multiple links to express alternate hierarchical representation but constraints exist upon the owner-member relationships. Thus the user must consider how the structure of his problem maps into the structure of the system used and the algorithms available to "reconstitute" the inherent structure of his problem.

4.2 Processing Algorithms and Queries

4.2.1 The processing algorithms of a DBMS include the processing of queries and usually some other simple standard services. The collected demands of the many autonomous scientific users of a system probably must be satisfied through a user-interface which

enables attachment of user software modules. This may also be required to process special queries which are not readily expressible in the standard query language. In any case, consideration of the ability of the data base system structure to support these extracurricular needs is important if the system is to be satisfactory. For example, if the structure of the problem is a forest of rooted trees and each query demands that this structure be "recreated" from a relational or network model, then this may be more costly than keeping a rooted tree record structure. Specialized computational needs may require the storage of vectors or large arrays within one record. The user interface and module are a solution only if the data base structures are suitable to the user algorithm. Stated another way, a good SDBMS will probably permit vector, array, and rooted trees within a logical record stored within the data base structure.

4.2.2 The visual display of structure (i.e. maps) will require algorithms which reconstruct the complex structure. When such displays involve many "elements", special storage of data may be necessary for efficiency of processing. Again, in addition to the user module, the data may have to be stored in a user-determined structure which can be described for retrieval and referenced easily in the SDBMS for the user. In addition to display, other computational procedures may require referenced user files.

4.2.3 The complex structures of a problem may exist to facilitate structure-oriented retrieval, for example the search for a tree of special characteristics within a forest. One may wish to examine only certain subtrees, ignoring the rest. Such a query may be simple to express with a tree-structured record but complex within a relational data base. The user must be assured that not only is a query possible with simple structures but that it is also reasonable to phrase it in the query system provided by the DBMS.

4.2.4 The query language supported by the DBMS may be low-level such as Boolean expressions, higher level such as predicate calculus, or a "natural" language. Since a language which is natural for one user may not be "natural" for a different type of user, the need for an extensible language should be considered. The examination of the relational model with the various language levels proposed will prove illuminating as an example of a simple basic structure combined with the powerful but awkward predicate calculus as a query language. The potential user should realize that few DBMS have been designed to serve the broad range of scientific users.

4.2.5 Data base management systems usually provide for both real-time processing and batch processing. Real-time processing requires some form of random access to the data base in order to have a suitable response time. Usually some form of index is constructed or threads are drawn through the data base. The user must be assured that whatever process is used, it can support the structure of his data and that essential associations are not lost. When queries cannot be restricted to certain data fields,

then total indexing (i.e. file inversion) may be required. Even so, associations can be destroyed. When real-time response is not absolutely necessary and queries can be "stacked", there is a lot to be said regarding sequential access to a collection of inherent structures. In even small- to medium-sized data bases (10^6 bytes), real-time response will often preclude sequential access; and the user must accept any limitations that the system places on his structure.

5.0 Data Base System Characteristics

DBMS reflect or should reflect the environment which nutured them and the needs of the users in that environment. Therefore, it is not surprising that a SDBMS should be different than the DBMS developed for a different environment. This section lists some facets of the R&D environment, some features of currently-available DBMS and some desirable features of a SDBMS.

5.1 R&D Data Base Environment

The following is a list of some attributes of the R&D data base environment which should affect the structures of a SDBMS.

1. Users are many, autonomous and often small.
2. Sophisticated and simple queries are required and are unpredictable.
3. The same is true for analysis (processing) requirements.
4. Has elaborate inherent structures and variable length data; many vectors, arrays and rooted trees.
5. Has numerous independent collections of data about independent individuals.
6. Requires "research reports" including graphics and plotting.
7. Has users with independent subject-oriented files, some very large (10^{10} bytes) but also many small and ephemeral.

5.2 Currently Available DBMS

Most currently available DBMS were not developed for the scientific community and have the following characteristics:

1. Assume a universe of coordinated users.
2. Assume fixed field data in simple structures.
3. Process simple queries with atomic operands.
4. Have simple analysis queries but no extensive user module interface.
5. Have "fiscal" report generators.
6. Require a staff of surrogates and consultants.

5.3 Some Essential Characteristics of SDBMS

The following is a partial list of characteristics required of a SDBMS:

1. Data Elements - Display and computation-oriented; variable length.
2. Structures - Atomic, vector, array, rooted trees, cyclic structures represented by simple structures.
3. Queries - Boolean, predicate calculus, natural language; structured operands.
4. Analysis - Statistics; FORTRAN, & PL/I interfaces.
5. Mode - Batch and online.
6. Output - "Manuscript" generation, tables, graphics.
7. Model Rationale - Content independent.
8. Data Bases - Large (10^{10} bytes) to small (10^{4}); must manage special files effectively.
9. Data Base Manager - Understands science.
10. Open-Ended - Multiple user schemas.

6.0 Caveats

The user should be reminded that these observations have been made in a large R&D environment and represent the collective needs of many potential users. A single project will often be served by a less comprehensive system. Nonetheless, a system should be sought which will meet the expanded needs of, say, five years in the future.

CAVEAT EMPTOR

REFERENCES

1. Date, C. J., "An Introduction to Data Base Systems," Addison-Westley Publishing Company, London, 1975.

2. Chen, P. C., "The Entity-Relationship Model -- Toward a Unified View of Data," ACM Transactions on Data Base Systems, Volume 1, No. 1, March 1976.

3. Knuth, D. E., "The Art of Computer Programming -- Volume 1," Addison-Westley Publishing Company, London, 1968.

THE CAPABILITIES REQUIRED IN A GENERALIZED DATA BASE MANAGEMENT SYSTEM FOR HANDLING SCIENTIFIC AND TECHNICAL DATA

K. F. Szczesny and W. M. Gersbacher

Information Systems Section
Computer, Information Systems, and Education Department
Battelle Columbus Laboratories
Columbus, Ohio 43201

A. Introduction

A complete specification of the capabilities required in a Generalized Data Base Management System (GDMS) for handling of scientific and technical data is very difficult due to the number of ways in which such a system might be used. Different applications require different data handling capabilities. Indeed, the capabilities of a GDBMS found important in one application may be of little or no importance in another application. As a result, a GDBMS cannot hope to satisfy all requirements for data handling in all situations. A GDBMS can, however, greatly ease the task of data management in most situations and provide a framework on which specific applications can be developed.

The capabilities of a GDBMS outlined in this paper are not intended to be an all inclusive specification for such a system. The intent was to list what was thought by the specialists to be the most important requirements that should be present in a GDBMS. These requirements have been divided into two sections for purposes of presentation. The first section deals with the capabilities required of a GDBMS for handling scientific and technical data. The second part addresses general features often found in a GDBMS which are also required in data applications.

B. Handling Scientific and Technical Data

Scientific and Technical Data present a number of special requirements that are not usually supported by the current data base management systems. In this section we present a number of requirements that should be present in a GDBMS if it is to handle this type of data effectively. Much of the data in a scientific application may be numeric while other types of data may be descriptive. A GDBMS should be able to support both types of data in an optimal manner. Specific requirements for numeric and non-numeric data are given here.

1. Requirement for Numeric Data

(a) Data Element Representation. A GDBMS for scientific data should be able to handle several forms of data. In order to gain a high degree of precision (and compression in storage), it is desirable that the GDBMS support the storage of binary data. Four forms of binary representation should be supported: (1) Fixed Point, (2) Floating Point, (3) Double Precision, and (4) Bit String. In general the system should be flexible enough to allow whatever representation one needs (binary or textual). It should be possible to introduce new data forms (for example, physical constants) and the facilities for handling them as the need arises.

(b) Numeric Data should be easily searched (using data ranges and relational operators) and easily manipulated.

(c) The GDBMS should support the storage of MxN matrics of data values. This must be done efficiently for both dense and sparse matrices. One must be able to address individual elements of the matrix using vector notation (subscript notation), e.g., M(1,2,3). The GDBMS should allow the user to retrieve the entire matrix with one call to the system.

(d) It is very important that the GDBMS be capable of handling very large quantities of data. The GDBMS must provide fast and efficient access

to this large volume of data.

(e) The GDBMS should provide the facility of validating incoming data and perhaps converting external representations to the desired internal representation. This may be accomplished by allowing a "hook" to user written special routines or via host language facilities.

(f) Many complex interrelationships will be present in scientific data. It is important that the GDBMS have flexibility so scientists can experiment with many relationships that seem to be present without high overhead in data base restructuring. The interrelationships among the various data elements should be "soft" in the data base so that many "views" of the data base can be accommodated without restructuring the data base.

(g) A high degree of compression of the data is very important. The use of variable occurence and variable length data elements is very common.

2. Requirements for Non-Numeric Technical Data

(a) The qualitative descriptions and explanations of scientific methods of investigation are often present with scientific data. These descriptions vary a great deal in length. Therefore a GDBMS must support variable length data elements and must allow numeric and non-numeric data elements to be present in the same data base.

(b) Textual descriptions present a number of information representation problems that must be taken care of by the GDBMS. The use of textual descriptions must be carefully controlled so that these descriptions can be used in a precise manner when searching the data base. It is desirable that the GDBMS support many of the capabilities used by information scientists to carefully control the problems that occur in human languages. The system should support validation, classification, and the use of a thesaurus.

(c) Using human languages to communicate technical ideas leads to unavoidable problems that result from differences in education, experience, background, environmental conditioning, and linguistic facility among originators, indexers, retrievers, and potential users of information. Expressed specifically in terms of a functioning information system, the problem is this - "How can the information presented in a physical document be indexed for retrieval so that the identity of its content will not be distorted or obscured by differences in intellectual qualifications and linguistic facility among originators, indexers, searcher, and users?." This problem must be solved to assure that stored information has an acceptable potential of retrievability.

There are imperfections in paths of communication which make it virtually impossible for possessors of knowledge to create written records which will carry to users exactly or even approximately the intended meaning. Imperfections in communication arise from the essential richness and complexity of language, linguistic factors, and human factors. The linguistic factors manifest themselves in the problems of semanatics, generics, and syntactics, while the human

factors pertain to the problem of viewpoint. Effectiveness of retrieval will be directly proportional to the adequacy with which the problems of viewpoint, generics, semantics, and syntactics had been solved.

A technical thesaurus functions as a word-reminder list. It enables indexers: (1) to describe information being indexed by as many terms as appropriate or necessary to provide for the different points of view from which the information in the document might be regarded; (2) to describe information in synonymous or nearly-synonymous terms where appropriate; and (3) to relate narrower concepts to more inclusive concepts on generically high levels. The consistent, conscientious utilization of a thesaurus in indexing will assure that technical information will be stored with maximum potential for retrievability.

(d) The GDBMS should support validation of textual fields by using facilities such as code controls, authority lists, and dictionaries. Very large dictionaries (10,000-20,000 entries) should be accommodated by the system.

(e) The GDBMS should allow one to store a compact code in a data field that can be expanded to a longer textual description when desired. This improves storage efficiency and maintains control of precise textual description of technical data.

(f) The system must be able to handle large volumes of textual data.

(g) Sophisticated indexing and searching facilities should be supported that allow textual fields to be used in a powerful manner for retrieval.

(h) Special symbols should be available in the character set used by GDBMS to represent scientific notation (e.g., greek symbols).

C. General Features Often Found in a GDBMS That are Required to Handle Scientific and Technical Data

1. Portability. Because much scientific work involves collaboration among individuals at several institutions, it is highly desirable that the GDBMS be easily transportable to machines of different vendors.

2. The system should have a very easy to use general purpose interface that allows it to present information in a data base to external programs such as statistics packages and other programs written in high level languages.

3. It is desirable that the system support a host language interface in FORTRAN and other high level languages.

4. The system should be easily used by non-programmer people. The GDBMS must provide software that can be directly used on any data base under system control that allows users to search for and manipulate the data. One should not need to write a program every time the data base is used.

5. The ability to use the GDBMS for interactive access to a data base is desirable.

6. If updates to the data base are made on-line in an interactive mode, then it is very important that adequate logging of transactions be done. One should be able to "roll back" and "roll forward" transactions applied to the data base.

7. Accessing versus Updating. A choice has always to be made between fast retrieval and easy updating. In a scientific data base, there will normally be less updating as compared, for example, with an order-entry program, and emphasis can be placed on retrieval efficiency. Most updates will be additions of new material with some corrections to old data.

8. The ability to search on data elements not specifically organized to facilitate searching (i.e., they are not primary search elements) should be possible.

9. It is desirable that the GDBMS support extensive Report Generation and also that it, perhaps, supports Graphical output.

10. The GDBMS should have a fair degree of tunability. That is, one should be able to adjust the system easily to weight some requirements over others. An example may be to give up updating efficiency to improve retrieval efficiency.

11. One should be able to tailor-make a set of applications to the specific needs of the end-user by using facilities provided in the GDBMS.

12. One should be able to archive unused portions of the data base off-line and bring them back on-line when required.

13. It is desirable that the indexes to large volumes of data could be available on-line with the actual data off-line. One can narrow down a search on-line and retrieve the data off-line in batch.

14. One should be able to permanently store regularly used requests.

15. A capability for restricting access to sensitive data is desirable. The use of encoding and decoding algorithms would be one possible technique for accomplishing this.

16. The system should be designed in a open-ended fashion to allow new features to be added with little difficulty.

REQUIREMENTS FOR THE DESIGN OF A SCIENTIFIC DATA BASE MANAGEMENT SYSTEM

(Derived from Experience with different programmatic data bases at LLL)

by

Viktor E. Hampel and Daniel R. Ries

Data Management Group
Lawrence Livermore Laboratory
Livermore, California, 94550

September 30, 1977

ABSTRACT

First, we discuss probable causes for the absence of a portable, generalized data base management system for *scientific* data at the DOE National Laboratories: The difficulty in assessing the monetary value of accurate up-to-date information and data as a corporate or national resource; the gradual evolution of highly efficient, sole-purpose and installation dependent systems where very large amounts of data are involved; the historical trend of judging the power of computers primarily by their calculational speed; the consequent delegation of data management to a secondary, piggy-back role on the large machines; the difficulty of providing the scientist or engineer with data in his customary notation; the apprehension of the casual user having to learn the peculiarities of home-grown systems; and finally, the absence of a comprehensive body of computer-readable scientific data that is authenticated, in the public domain, and which could act as an incentive for local use.

Second, we identify administrative and technological requirements that seem necessary and desirable for the design of a general ***Scientific Data Management***

• Prepared for the U.S. Energy Research & Development Administration under contract No. W-74050Eng-48

System (SDMS) in support of the newly emerging national programs concerned with energy and the environment: Authorization and funding for ***SDMS*** must clearly come from top DOE management; only then can we hope to reduce the redundant development of special, home-grown systems and their costly maintenance, in support of small but active scientific programs. A ***SDMS*** should, therefore, not be proprietary software. It should be reasonably portable for use on back-end mini-computers and the more powerful machines; it should be self-guided to help the casual user find his way. From a technical point of view, the system should work equally efficiently vectors, matrices, arrays, complex variables, sparse data and text. It should be capable of storing and displaying data with most of the customary scientific notations and attributes in different units of measurement. The user should be able to use the system as a programmable calculator for simple mathematical tasks, and he should be able to extract and transfer data to his model or application program for more difficult calculations. The system should permit common access to a library of inter-related data bases and reference tables. In this sense, the ***SDMS*** should permit expansion to an ***Integrated Information System*** on computer networks with distributed resources. Most importantly, the user should be given a reasonable, English-like command language to start, but he should also have the option to create his own dialects and extensions of the system for his personal and programmatic needs.

1. INTRODUCTION

"Why Another Data Base Management System?"

This question is not new. It has persisted at most AEC/ERDA/DOE Laboratories and computer centers during the past decade. As we are considering a future Generalized Data Management System (GDMS), that might serve the scientific community at large, it seems pertinent that we examine the reasons why an adequate system for generalized scientific data does not exist today. By reviewing the historical evolution of data management in an administrative and technical sense, we should be better prepared not to repeat the mistakes of the past.

The state of the art of business-oriented data management is well documented in the recent literature. Available features and their implementation techniques are summerized by the ***Comparative Data Management Systems*** seminars, given throughout the year by the University of California Extension Division at Los Angeles. At LLL, we have worked with a large number of highly diversified technological and scientific data bases, ranging from material properties, air-pollution data, ecological data, atomic and molecular data, laser parameters, and the ***Table of Isotopes***.[1-4] We have also studied and developed data management systems and computer tools for data validation and display. [5] For the purpose of this report it will suffice that we point out

only those aspects which we believe to be unique and significant for scientific work:

		Additional Scientific Features:
1.	Data types	*Scientific attributes are mandatory*
2.	File Structures	*Vectors, matrices, arrays, etc.*
3.	File Creation	*With scientific validation*
4.	User Interface	*Self-guided, higher user dialects*
3.	Retrieval & Update	*In different units of measurement*
6.	Security	*In the Interest of national defense*
7.	R&D Application Programming	*Control of Input and Output*
8.	Reporting	*Formuulae, scientific notations, etc.*
9.	Computer Communication	*Among evaluators & users of data*
10.	Distributed Resource Sharing	*Among Labs & data evaluation centers*
11.	Access & Transaction Control	*Use of distributed resources*
12.	Integrated Information Systems	*Scientific Reference Data Banks*

Depending upon the particular installation and working environment, one or the other of these features will be emphasized or left out. For the OCTOPUS secure operating system at LLL, for example, it would be added ballast to carry along the overall security provisions necessary somewhere else, although selective access control to individual data fields would be desirable. Of particular importance for our work with scientific data is the ability to annotate data with qualifiers, or attributes, that support the datum. Both, for measured and evaluated properties of materials, the following attributes are significant:

Attributes of Scientific Data

1. Value
2. Uncertainty
3. Units of Measurement
4. Normalization
5. Validity Domain
6. Method of Measurement
7. Conditions and Constraints
8. Type of Data
9. Source of Data
10. Bibliographic Reference
11. Comments
12. Proprietary Status, Classification, etc.

The requirement to store so large a number of attributes per ***datum***, or for a set of data, poses a considerable overhead burden. At LLL, we introduced user-definable dictionaries that link the attributes to a bit or byte pattern, so that each fully annotated ***datum*** may require only 1-2 additional computer words of overhead. Many data sets can be qualified as a group, of course. The relational data base structures are quite suitable for these types of annotations, where needed.

In general, scientists work with multiple data bases and with files of different layout, with reference data, tables of physical and chemical properties, conversion matrices for units of measurement, and libraries of evaluation programs. In addition, engineers and scientists wish to have the right to work with their own best values for some or for groups of data. The researcher expects the capability to create and to use temporary files, or ***scratch-files***, not unlike the yellow notepads used in daily work. The *SDMS* should, therefore, provide a convenient option to extract and to combine whole tables of data, or particular rows and columns. This implies that the *SDMS* should also function as an efficient and versatile hand-calculator. It should lend itself to control the input to calculations, and help with the storage and analysis of a growing volume of calculated results. To the program manager, the essential question might be whether the *SDMS* increases productivity and is cost-effective.

The criteria that any user might apply, consciously or sub-consciously, by ease-of-use, convenience, and the accuracy and quality of results. We examine the above postulated requirements for a *SDMS* in retrospect.

2. HISTORICAL BACKGROUND

At most of the larger National Laboratories, in the United States and abroad, the demands for computer-assisted data management existed in four well-defined areas:

* Business Data Processing
* Searching of bibliographic citations,
* Manipulation of large, specialized numeric files,
* Manipulation of small, diversified numeric files.

2.1 Business Data Processing

Business-, procurement-, and personnel-related files have been managed primarily by commercial software since the early days of electronic data processing. The software packages are tailored to the day-to-day business transactions and may be linked to a particular machine. Only the large installations have found it necessary to develop their own routines for specific administrative tasks. Most

business computer programs operate, even today, in the batch mode and are not readily adaptable to scientific needs.

2.2 Searching of Bibliographic Citations

Computer-assisted searching of the world literature has become big business. It is an essential part of the research activities at National Laboratories. Since 1967, citations to nearly every field of human endeavor have become available on magnetic tape. [6]

This has permitted machine searching of programatically relevant files and has been used to keep the professional staff informed on developments in their field of interest. Ongoing research has been kept up-to-date by subscription to the Selective Dissemination of Information (SDI), while in-depth retrospective searching has been initiated to get new programs started.

The market for this type of information is sufficiently strong and supports a number of large and diversified information services. Here, the big volume of computer-readable, bibliographic citations makes it no longer advantageous to subscribe to these magnetic tapes for the purpose of placing them as aggregate collections on in-house computers. Instead, contracts with vendors of the secondary literature permit browsing and searching by dial-up from remote terminals. [7] This is being done over computer networks, or over regular telephone lines in an interactive manner.

Only at large installations, e.g., the Oak Ridge National Laboratory (ORNL), or the Lawrence Livermore Laboratory (LLL), has it been cost-effective to process the great number of SDI searches with locally developed software in-house. [8] Most of the bibliographic search programs, which are interactive and geared toward large volumes of information, are specifically adapted to the manipulation of descriptive text. They are not suitable for numerical data manipulation.

Federal agencies, and the National Science Foundation in particular, have supported the development of these computer programs over many years. But few are today entirely in the public domain. Often, when Government support was provided after the beginnings of a program had come into existence, or where such support became interrupted in time, the final product was often found to contain intertwined proprietary routines which rendered the final software also proprietary, or incomplete and useless when requested by other agencies. [9] In other cases, where the software may have been in the public domain, hardware/software incompatibilities rendered the transfer and use of the programs difficult or impractical.

2.3 Manipulation of Large, Specialized Numeric Files

Large volumes of numerical data are best manipulated even today with sole-pupose, highly efficient programs. These data are usually required in

support of major programmatic objectives. Data formats and options are dictated by the changing demands in an impromptu manner. Data files may thus become an integral part of the data manipulation programs, which may not even exist in an identifiable stand-alone form and are often simply an option of the model or application program. In most cases, the data are processed in the batch mode. Even for the immense libraries of neutron cross sections, shared among the national and international centers, the corresponding data manipulation programs are specialized and differ from installation to installation. The adaptation of these sole-purpose programs to general-purpose scientific data has seldom been attempted. (Figure-1)

2.4 Manipulation of Small, Diversified Numeric Files

The requirement for the manipulation and aggregation of interdisciplinary data in support of energy and environmental research has brought about in recent years a renewed interest in generalized data management software. However, because of the absence of a strong programmatic funding for the considerable number of relatively small different projects, these needs were usually met by locally-developed systems, some of which acquired in time a considerable degree of flexibility and usefulness: For the ERDA/DOE National Laboratories we note predominantly on CDC and IBM hardware:

BNL	CDC	BDMS [10]
LASL	CDC	GIRLS, MASTER CONTROL [11,12]
LBL	CDC	BDMS, STOFI, SEEDIS [10,13,14]
LLL	CDC	MASTER CONTROL, SDBMS [12,15]
ORNL	IBM	ORCHIS, ORRMIS, RECON [16-18]
PNL	CDC	Remote access to LBL and BNL
SRL	IBM	JOSHUA [19]

These systems and their use have been described in the literature. The list is incomplete. There are certainly other systems at the National Laboratories, especially on mini-computers.

The fully developed, generalized systems like ORCHIS, JOSHUA, and MASTER CONTROL are 6-10 years old, are relatively large in size and could not readily be transferred to other installations; except where the hardware and operating systems were compatible. (MASTER CONTROL was transferred in this manner to LASL, BDMS to BNL, and parts of JOSHUA to ORNL.)

The more recent data management systems, BDMS and SDBMS, now under development at LBL and LLL, still lack many of the attributes that provide power and flexibility to a generalized system. But, they utilize more recent techniques in computer science and are more readily transferable to other machines and operating systems.

2.5 Absence of Coordination and Adequate Funding

The need for generalized management of scientific and technological data was met by ERDA in a decentralized manner. Individual Offices, Divisions, and Branches tried to meet their demands by adapting or creating specialized software for their own programmatic needs. To date, there has been no apparent attempt to unify the information and data management software development effort for ERDA or DOE. Except, there exists a recent ERDA/DOE directive that new developments in this area be made known to DOE Headquarters for review and approval. [20]

The cost of developing a general system has been estimated upwards of $2,000,000 with an overall design and implementation period of more than two years. As mentioned, the small programs could not afford this expense or delay and proceeded to make do with what was available, or what could be written quickly. The cumulative total outlays from all of these individual programmatic efforts may very well have exceeded the single unit cost estimate.

Consequently, nearly all of the systems at the ERDA National Laboratories have evolved gradually, under incremental and minimal funding, and independent of each other. Usually this took place in response to local programmatic requirements that could not be met by the large sole-purpose specialzed systems, nor by the commercially available systems. In recent years, as our search for new sources of energy accelerated and the concern for our environment came into focus, we observed a renewed interest in generalized software to cope with the highly diversified and interdisciplinary activities. But the emphasis had shifted to short-term objectives. Program managers were not prepared to wait and pay the price for development of a new or more versatile system. The old systems were consequently used, and some investments were made to explore whether any of the older systems could be salvaged after all, either by making them portable, or by bringing them up-to-date. [21]

2.6 Lack of Dedicated Disk Storage for General, On-line Data Bases

The computer centers at the National Laboratories are structured foremost to support ongoing applied research. Except for the DOE/RECON bibliographic search facility at ORNL, the centers do not provide access to resources of general interest to the research community. This is on the verge of changing as data storage devices decrease in cost, and as data banks of scientific and technological content come into being.

But even today, at most installations, we seldom find sufficient allocation of direct-access, primary storage that could be used to provide quick, on-line access to data bases. In most cases, the users still have to secure their own files from secondary, off-line storage. This has a discouraging effect and tends to choke operations as data banks are redundantly loaded by users into direct-access storage, without readily available facilities for their sharing and updating.

In addition, the operating systems at BNL, LBL, LASL, and LLL do not support re-entrant code or updatable shared disk files across computers. This has also tended to discourage the development of more generalized data management systems, and may have hindered the effective use of commercial systems where they were considered adequate for business-like technological applications.

The main issue here is one of ***Effort***. What ***effort*** must the interested user exert to find the information or data that he seeks or needs? After all, even central libraries have found it necessary to establish branch libraries on site to bring the printed book closer to the user. Information and data, when not easily accessible or within reach, will seldom be used.

2.7 No Preferential Role for Data Management

Computer centers at the National Laboratories have been oriented primarily, and quite appropriately, to serve the large dominant user groups. General-interest data management systems have, therefore, been relegated to a secondary, piggy-back role on the large powerful machines, even when the data was needed but belonged to a smaller, less significant program. New machines at the National Laboratories have been selected primarily for their calculational speed in support of applied research. Indeed, data management systems usually have had to operate like any other application program, either in the batch mode or, where interactive time-sharing was practiced, without preference to faster I/O requirements.

This situation has led to design improvements in these systems, making them more competitive in the time-sharing environment. Modularity, dynamic-dimensioning, and decreasing core memory were some of the techniques used to elicit faster response. It is probably also true that the many options available on the powerful computers have prompted programmers to write simple and effective data storage and retrieval procedures for particular needs. Developers of such small, self-made systems may have thought at first that their systems could be more responsive because of their small size, but soon discovered that they, too, had to add in time the desirable features of more established systems: ***Report Writers, Arithmetic Manipulation Packages, Recovery Procedures, etc.***, all of which tended to decrease their response time.

When this took place, a trend toward independent, stand-alone mini-computers became apparent which promised to free the users from the overall constraints of the large computer centers, and which were within the budgets of the smaller energy and environmental programs. In some cases this approach proved successful, especially where the small machines were connected to the facilities of the main computer center. However, in the absence of such connection, peripheral equipment in the form of extra storage, tape-drives, graphical display hardware, etc. had to be acquired and duplicated. The maintenance of such mini-computer centers has not been insignificant. The whole

approach was, therefore, successful where, in addition, the purpose and bounds of the mini-system were well defined and enforced.

2.8 Difficulty in Duplicating Scientific Notations

Until quite recently, most computer centers have operated with a minimum of printable characters. Some installations are still in the stone age and print only upper-case letters. Others have extended their capabilities to include the 95-character Federal Standard. [22] But even this capability is not sufficient for physics, chemistry, or mathematics. As a result, researchers have not found in computer-assisted data management an extension of their daily manual working environment. Only where large blocks of simple numeric data were needed as input to machine calculations, were the limitations of impact printers less obvious. It is again only in recent years that these constraints were lifted by the availability of non-impact printers and CRT-driven printing devices like the FR80. And yet, even today, it is difficult at most installations to reproduce chemical formulae with sub- and super-scripts on a screen or on hardcopy terminals. (This report was generated and formatted by computer.)

2.9 Absence of Comprehensive Bodies of Computer-readable, Scientific Data

Most data evaluation today still stops with the publication of a book. This is the case even though the tables of data within a book may have been printed with the assistance of instructions on magnetic tape. The delay between the measurement of a material property and its ultimate publication in readable form may be as long as two years. There does not exist in the United States a National program to establish computer-readable files, or data bases, as counterparts to printed books for direct use by machines.

The Office of Standard Reference Data, NBS/OSRD, offers some of its data on magnetic tape. [23] But, even these tapes are seldom requested, we are told, probably because it is cumbersome to translate the tape formats into local data management systems, and because it may be difficult to reproduce the scientific notations where they have been retained from the type-setting instructions. We conclude, that in order to make data sharing effective, the major tools required to use the data must also be sharable, i.e., in the public domain. Thus, in general, the computer environment has not been very conducive toward the management of general-interest, truly scientific data.

2.10 Could Commercial Systems Fill the Need of the Scientific Community?

The answers to this question have varied. In some cases the technological data bases resemble business data, e.g., those containing air and water quality data, or technological/administrative information. For these cases the commercial systems have certainly been good candidates. Consequently, some of the DOE Labs that did not have adaptable systems acquired commercial systems: MRI's System-2000 was obtained by ANL, BNL, and LASL. (LASL needed a DBMS for the CDC-7600 unclassified machine, operating with a CDC operating system; MASTER CONTROL has been used on the classified machines with LTSS.) SRL purchased ADABAS. Most of their applications were well-defined and routine.

On the other hand, where the data management system was to be integrated with state-of-the-art color graphics, or with data application programs that were themselves under development, or where the resultant data bases and programs had to be shared with a larger group of users at different installations, the acquisition of commercial systems seemed less desirable or simply did not meet the needs of the user. The proprietary nature of the programs, the unavailability of their source files, and the limitations on the distribution of the software after it might have been modified or extended, all made it rather difficult to plan future developments with commercial DBMS products.

More important perhaps was, and still is, the very different computer environment at the National Laboratories, making it quite impractical for commercial systems. As a rule, the Labs have stressed computational power for their programmatic R&D work, as mentioned earlier. This has resulted in acquisitions of the most powerful, up-to-date main frame machines and peripherals. In several cases, e.g., for the CDC-STAR and CRAY computers, the operating systems had to be written or adapted by the Labs. In addition, the application programs that interface necessarily with a data management system were, and always will be, in a continuous state of rapid evolution.

2.11 A Need for Inter-Laboratory Collaboration

Approximately two years ago, when the renewed interest in generalized *SDMS* software came to the forefront of our attention, we proposed to the Division of Biological and Environmental Research of ERDA the formation of an *Inter-Laboratory Working Group for Data Exchange* (*IWGDE*). The initial objectives were to provide a forum where our mutual expertise in the design and use of scientifically-oriented data bases could be shared, and where the functional specifications for a future system that might serve more than one ERDA installation were to be prepared jointly.

This group has indeed been funded now for two years by the Office of Environmental Information Systems (EOIN) of the ERDA Office for Environment, Health, and Safety. EOIN was charged with the difficult task of setting up regional data bases and information services in the country. However, it was felt that the first order of business for this Group of Lab representatives should be the identification of available resources and the definition of data exchange standards among the Laboratories. This has indeed taken place. The *IWGDE* efforts resulted in the formulation of an extension to the present American National Standard for Information Exchange on magnetic tapes, X3L5/506t, for numerical data. [24]

2.12 Professional Disagreement

Data management, a relatively new field of computer science, now employs some 35% of all professional programmers and system analysts. Numerous ***best*** ways have been proposed how to store, access, and use data cost-effectively -- and to the liking of the user. The arguments depend on, and differ with the

characteristics of dissimilar computer environments, their particular operating systems, programmatic objectives, and user needs. There is usually more than one way to do a job ***well***! In addition, since we are dealing with interactive, real-time systems, we note that the techniques and types of the man-machine communication are a matter of personal preference or taste.

As ***The Art of Computer Programming*** is being transformed quickly into a ***Science***, it gives rise to much discourse among designers and users. Solutions have ranged from idealistic ***do-all*** promises to the transplanting of tools borrowed from the world of business data processing. Questions of how best to structure particular data bases, how to make their use efficient and appealing, and how to optimize under dynamic operating conditions have been given many a contested answer. Information and data often receive a personal interpretation and an emotional association by their 'proprietors'. This protectiveness has carried over into the work of system designers.

The discourse is apparent in routine EDP applications, and even more so in the technological and scientific R&D working environment where computers remain subservient to well established fields of science. Major efforts have been expanded in viewing data bases as CODASYL-like networks of information, as hierarchical structures, and more recently as relationships of predicate calculus. This situation is reminiscent of the different missile systems proposed and built during the 1950's when rocketry started to come of age. Today, after a decade of probing in data management, we still do not find concensus on the central and unifying role that data management purports to have in our society, and the essential aspects of good design and implementation techniques.

2.13 Summary of Historic Review

In retrospect, it is probably true that the inaction on the part of administrators to provide determined support for a generalized ***Scientific Data Management System*** (*SDMS*) has been an appropriately cautious response to the uncertainties and imponderables of the past. But some conditions have changed and progress has been made. To bring about a successful *SDMS* design and implementation will depend not only upon the know-how and experience of designers, but also upon the steady support by management and the availability of a conducive computer environment. These requirements have to be treated jointly and are as important, we believe, as any singular proposal of technical excellence.

3. ADMINISTRATIVE AND TECHNICAL REQUIREMENTS

From our historic perspective, we now identify the decisive administrative and technological problems of the past that seem to have inhibited the successful development of general *Scientific Data Management Systems* (*SDMS*) at any of the National Laboratories. We do this in light of today's environment being mindful of future expectations and needs. The crucial problem areas requiring determined action are:

- Management Support
- Demonstration of a Prototype
- Active Participation of Users
- Creation of a Back-end Data Management Machine
- Software, like Data, should be in the Public Domain.
- *SDMS* Design and Development – A Collaborative Project.

3.1 Management Support

There is no doubt that management support and steady, adequate funding are the indispensible condition for any project, and in particular one that transcends organizational boundaries. Had it been present at any of the DOE installations, successful *SDMS* candidates would have emerged sooner and with greater potential. However, for this to take place now, in an environment where Research and Development (R&D) expenditures of the country have been steadily declining, it can probably only be expected after elimination of the credibility gap that seems to surround the *SDMS* issue.

3.2 Successful demonstration of a Limited Prototype

This recommendation is not meant to be a delaying tactic. Administrators have readily supported the in-house development or acquisition of business oriented data processing systems. These systems were the necessary ingredient for a competitive posture of their organization. Project leaders of large national programs, in cognizance of earlier similar applications, also did not hesitate to approve substantial software development needs for their sole-purpose data management needs. The projects could hardly have been carried out otherwise.

To elicit support for a general *SDMS* in support of energy and environmental programs, and for the DOE scientific community, will require a similar determination and support. What we need is the creation of a limited capability system that could excite the interest of potential users and the subsequent endorsement of administrators and top management. It would act as a prototype. For that reason it should ideally serve both groups, be reliable, extensible, and simple in use. It should increase productivity!

Several candidates exist at the National Laboratories and abroad. An effort should be made to eliminate the deleterious circumstances that seem to have handicapped the development of these *SDMSs* at the large computer

installations, to document their performance, and to demonstrate their capabilities of increasing the effectiveness of ongoing research. But here again, without top management support we can not hope to attract capable computer scientists -- or hold them for any length of time.

3.3 An Expanding User Community

As time advances, our expectations of the capabilities for a future *SDMS* seem to increase also. Not only should such a system be capable of handling the complexity of scientific information and data, and display them in the customarily accepted scientific notations, but, in addition, the system should be equally useful to casual users and technical administrators who may have little experience with data management or with computers. This implies that the generalized *SDMS* should be capable of replacing the "Yellow Notebook" of the researcher, be an effective tool for generating input to scientific calculations, and managing the growing volume of calculated results. The *SDMS* should guide the user through a variety of available general reference data and standard analysis routines. In short, it should be an effective tool for computer-supported decision making by scientists, engineers, and technical administrators.

The writing of sole-purpose EDP software requires exact specifications from the sponsor and users with regard to input and output formats. For the design of a general-purpose system, not under the direct sponsorship of a particular program, much responsibility rests with the system designers and implementers to interpret present and future needs of potential users. As a rule, the latter will not have extensive experience with interactive data management systems, and their wishes may be either unrealistic or parochial. But user groups should be involved during the evolution and during the testing of the prototype. It would be desirable that one, two or three major potential users be involved, especially if they have had previous experience in the use of scientific data management by machine. The greatest responsibility, however, rests upon the know-how and experience of the system analysts who structure the system and its capabilities. Users are well versed in their objectives -- the *SDMS* designer and implementer has to understand his own and be practical enough to strike a compromise by careful examination of the present and foreseeable working environment, and the selection of appropriate implementation techniques.

It should also be borne in mind that a general-purpose system will hardly be able to compete with sole-purpose systems, specially adapted to programmatic tasks. System designers should have personal working experience with scientific data. [25]

3.4 The Dedicated Back-end Data Management Machine

It is highly unlikely that generalized management of scientific data at the research-oriented large computer centers of the National Laboratories could suddenly gain a preferred status. Competition for available CPU resources and for direct-access disk storage is simply too great.

Acquisition of future generations of machines will undoubtedly also take into account the management of input and output to calculations with automated and simpler tools than today. As the volume of calculations increases with more powerful machines, our ability to cope with the masses of data must increase also. We recognize a solution to this dilemma and to that of the *SDMS* issue in the trend to employ mini-computers as back-end machines. They can handle most of the I/O and the simpler tasks of data manipulations. Access to the large machines takes place in this concept under control of the mini-computers when needed, or when requested by the user. This is exactly the necessary computer environment for the reliable development of a generalized *Scientific Data Management System*.

This approach is being taken at LLL by at least three unrelated programs: (1) The SHIVA/NOVA laser-induced fusion energy program; (2) The National Uranium Resource Evaluation Program (NURE), and (3) The Integrated Information System (IIS) for the DOE Division of Energy Storage.[26] In each case, a PDP-11/70 machine is to accept data input, provide the user with quick access to needed information and data, execute control over transactions, and communicate with the more powerful computers when high-speed processing or more complex analyses are required.

The conceptual requirements for each of the three applications has been arrived at independently by separate teams. The staff of the Data Management Group at LLL has particular responsibility for the DOE/STOR Integrated Information System. Similar combinations of mini-maxi computers to enhance data management are in preparation at ANL, LASL, LBL, and elsewhere.

The essential advantage of a *Data Management Machine* is its dedicated status. It may alleviate for the user the necessity to learn more complex procedures of a general purpose computer environment. It increases substantially the reliability and response of interactive data access. Of course, since the user may wish to execute some functions on the large machines where his application programs reside, he may find it advantageous as an experienced person to use the *SDMS* there also. The *SDMS* should, therefore, be written for use on large and small machines.

It is still too early to state the degree of success for these solutions. For the SHIVA and NURE projects which deal with relatively routine operations, commercially available operating systems and data management systems were selected. For the DOE/STOR project with non-routine utilization, the UNIX/INGRES combination [27] was chosen as the basis until the *Scientific Data Base Management System* (*SDBMS*), under development by the Data Management Group at LLL, becomes operational later next year.

3.5 Creation of Scientific and Technological Data Bases

Several efforts are under way to demonstrate the usefulness of general-interest numerical data bases. The Nuclear Data Section of the IAEA, for example, has

been mandated at the 1976 meeting at the Culham Laboratories to expand its coordinating function into the Atomic and Molecular data field. National programs of considerable scope have been in progress in Sweden, West-Germany, and Japan. The Data Management Group at LLL is collaborating with the Office of Standard Reference Data (NBS/OSRD) to establish data banks of evaluated properties for materials of interest to DOE/STOR. These data are being prepared for use on the PDP-11/70 machine, as part of the ***Integrated Information System***, with remote access over the ARPAnet and by telephone dial-up.

Legal, monetary, and national constraints on the use of non-defense oriented data can hopefully also be overcome to foster and renew a free spirit of research and progress in the scientific community.[28,31]

3.6 Scientific Notation

The display of scientific notations in the customary, accepted forms of the professional literature is still difficult at many installations, but substantial strides have been made to make it possible. At LLL we have made good use of the Hershey character sets and have added a number of new fonts. Most of the scientific notations are now standard options. By using the FR80 data output machines, we can print formulae and equations on film or directly as camera-ready hard copy. Clearly, the laser physicist studying the polymorphous silicate glasses would like to see printed by the computer

γ-Na_2BeF_4 not GAMMA-NA2BEF4.

Recently, we gained experience in the preparation of rather unusual free-form characters by adding the Hebrew character set to the standard repertory, inclusive of cantillation sentinels. By establishing a macro definition for the keyboard, input could be provided from standard terminals where a sensible correlation was made among characters of the English and Hebrew alphabets. Here it is not important that it was done, but that it could be done within a matter of days. We are applying the lessons learned with free-form characters to the writing of mathematical expressions and formulae. [29]

Similar capabilties are now starting to become available elsewhere. We believe that the difficulty of depicting and displaying scientific notations have now been substantially overcome. The process is still somewhat cumbersome. Physicists and chemists who may have approached only reluctantly a data management system that could not reproduce their customary text and data should now be more receptive to the personal use of a ***Scientific Data Management System***. In the MASTER CONTROL system we introduced the concept of a functional data field that can be used for numerical processing and the integration of scientific text and data. A similar capability should be part of any future ***SDMS***. [12, UCRL-52015]

3.7 ***SDMS***, a Portable System in the Public Domain

The Federal Government does not compete with industry. But there is clearly not

yet enough profit in scientific data management, or industry would have filled the vacuum. Moreover, the DOE family of Laboratories is by far the greatest user of high-power computational equipment. Historically, as mentioned, the most advanced computers have been specified by AEC-ERDA. Operating systems, utility routines and networks are all in a state of rapid flux. This creates a working environment without parallel in the commercial business world. A good example is the extensive use of color graphics and computer-generated movies for the analysis and understanding of measured and calculated phenomena. These graphics programs are experiencing a tighter link to data management, and are in a continuous state of evolution. [30]

A consideration might be to write the exact specifications for a generalized *Scientific Data Management System* in a Request for Proposal (RFP) and to subcontract the writing of the software for the *SDMS* as an extension of a particularly promising business-oriented system to include scientific requirements. In view of the rapidly changing R&D conditions this appears to be difficult, especially since it is desirable to have the *SDMS* operate both on mini-computers and on the more powerful mainframes. More importantly, however, the resultant software would be proprietary and could not be modified by the Labs as is often required on demand and short notice. Such a system could then also probably not be shared with other Labs or installations.

We have also considered a joint research venture with some of the leading industrial software houses, experienced in data management systems. Although the software could perhaps be developed and documented jointly under a mutual research agreement, the resultant software could ultimately be used without legal impediments only in-house and could not be shared with other Labs, according to recent legal counsel on this matter.

Finally, it is conceivable that the new system could be written by a commercial vendor under exact technical specifications from scratch and placed for use in the public domain. To provide enough feed-back during the definition phase would require extensive familiarization of the industrial system analysts and programmers with the DOE scientific computer environment and the needs of its professional community. This is difficult to achieve with an outside organization. When this approach was tried by NASA/Lewis in the early 1970's, the results were not satisfactory. To improve upon this situation, designers and developers should be part of the scientific working environment, just as they were part of the business world for the development of business-oriented systems. We arrive at the conclusion that the involvement of commercial software houses in the writing of a generalized ***Scientific Data Management System*** is questionable to date. Except for routine applications that are similar in nature to business data, where commercial systems have been quite successful, it appears that generalized ***Scientific Data Management Systems*** will probably have to be developed by DOE for DOE.

3.8 *SDMS* Design – A Collaborative Project

The rapid progress of computer hardware and improved communications have stimulated collaboration among the National Laboratories. As software developments increase in cost, we note also a greater reliance upon the sharing of utility routines and computer programs. This is especially apparent between LASL and LLL where the joint use of the LTSS operating system and its extensive library of supporting utility routines has resulted in a considerable savings for the two Labs.

With reference to the ***Interlaboratory Working Group for Data Exchange*** (***IWGDE***), now that a viable data exchange format has been established for magnetic tapes, the concerns about a portable data management system for scientific data is coming again into the forefront of attention. Several of the ***IWGDE*** staff are contributors to the OECD/NEA Specialist Study Group that is preparing to lay the foundation of a generalized system for scientific use. This suggests a modularity of *SDMS* design. One would hope to devise a procedure whereby particular features for a specific installation could be selected from a wish-list to satisfy the necessary and sufficient local requiremments. Also, one would hope that computer scientist with particular expertise in their field of specialty could be engaged to participate in the design and development effort. In a practical sense, features, i.e., modules, needed at some installation should best be developed by in-house professionals, bearing in mind the more general requirements of the overall *SDMS* system. One organization should be assigned responsibility for the project. Authorization and support should come from DOE Headquarters, based upon the foresighted beginning made by the Office of Environmental Information Systems, DOE/EOIN.

SUMMARY

We believe the time has come for top DOE management to deal with the issue of generalized, ***scientific*** data management in a concerted and determined manner. By such action, redundant and costly developments could be channeled to good advantage. Today, the design, development, and implementation of a ***Scientific Data Management System*** (*SDMS*) is less risky than in years past. Significant advances have been made in computer science, the reliability of business data management on a global scale is proven, we are in a position to give to the scientist and engineer his data in acceptable scientific forms, and the cost reductions due to micro-computerization are continuing.

In particular, we recommend for consideration the concept of the back-end mini-computer as a dedicated scientific ***Data Management Machine***. It would not only reduce the I/O burden now carried by the powerful computers, but would also provide for the casual and knowledgeable researcher, or technical administrator, a responsive tool for increased productivity.

The system should be designed, developed, and implemented as a collaborative project, under the technical leadership of one organization. Administrative responsibilities should be kept separate. Contributors could specialize in their field of expertise by writing modules in support of their local requirements. This should find the endorsement of local management. It suggests a top-down, modular, and well structured approach for use on mini-computers and the major main-frame machines.

If we agree that measured and evaluated properties of nature must be unencumbered by legal or monetary constraints to support the national R&D effort in the sciences, then the generalized *Scientific Data Management System* as the primary tool for their effective use must also be in the public domain. Before a major commitment is made, the development of a prototype should be funded. This would permit us to test the practicality of the proposal in a limited sense, and to engender user support.

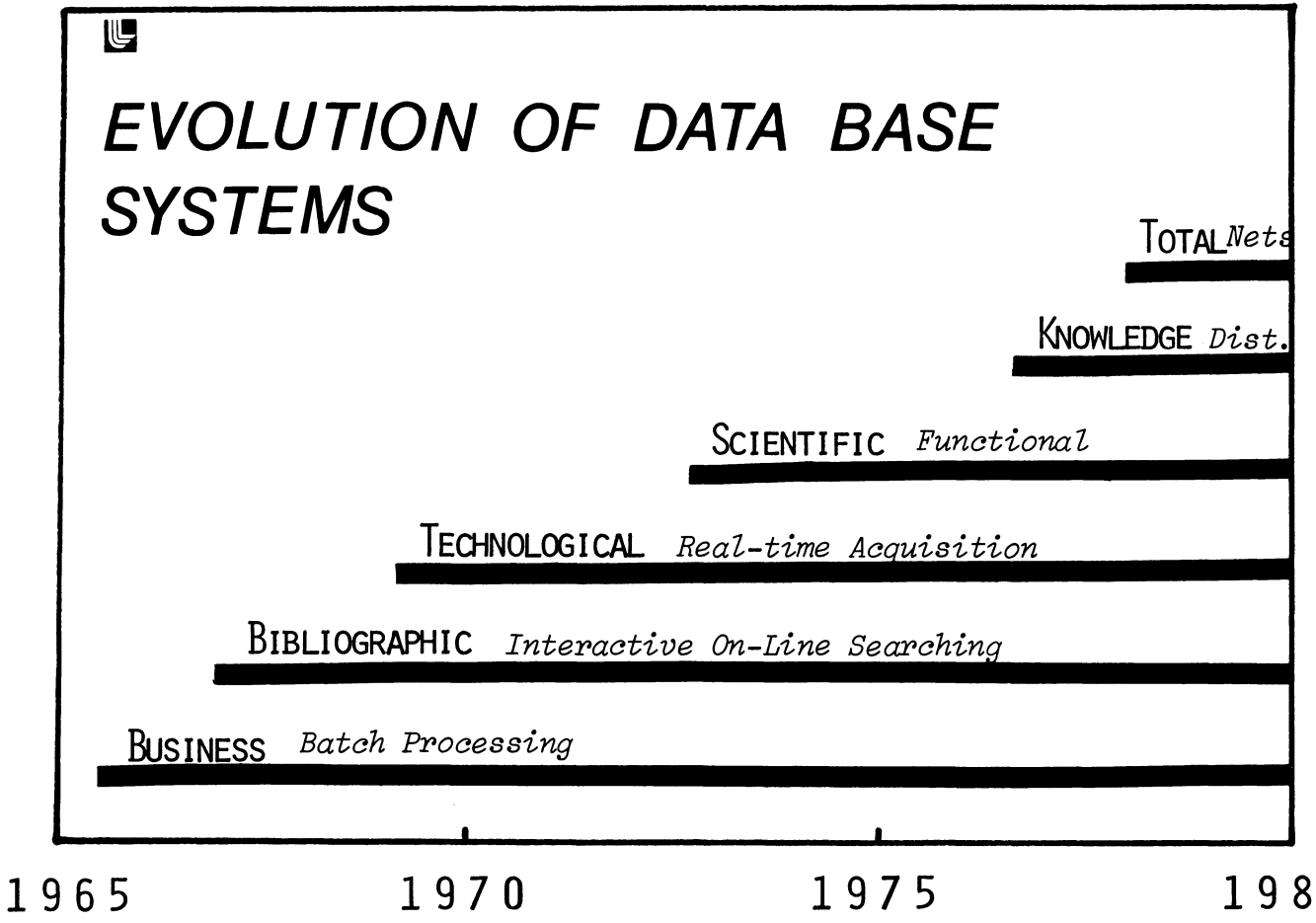


Figure-1: The Evolution of Data Base Management Systems

REFERENCES

[1] V. E. Hampel, ***Problems and Solutions for the Creation and Utilization of Large Interdisciplinary Computerized Data Banks***, UCRL-74685, CODATA Symposium, Freiburg, Germany, 1973.

[2] M. C. MacCracken, G. D. Sauter, ***Development of an Air Pollution Model for the San Francisco Bay Area***, UCRL-51920 and especially UCRL-51537 for the use of MASTER CONTROL as the DBMS, 1974-75.

[3] J. F. Barbieri, ***EPA-San Bernardino National Forest Information System - Final Report***, UCRL-52309, 1977.

[4] E. Henry and V. E. Hampel, ***A Computerized Data Base of the Fundamental Constants of Nature***, UCRL-51969, 1975 E. Henry, ***Computerization of Atomic Level and Transition Data for the First and Second Ionization States of the Elements Hydrogen through Phosphorus***,UCRL-52148, 1976. E. Henry and V. E. Hampel, ***Computerization of Spectroscopic Constants for Selected Heteronuclear Diatomic Molecules***, UCRL-52149, 1976. P. L. Sommerville, L. Wood, and V. E. Hampel, ***Prospective Nuclear Transitions for a Gently-Pumped Gamma-Ray Laser*** (Identified by the use of the MASTER CONTROL data management system), UCRL-76442, 1975.

[5] W. T. Overman, J. E. Ramus, ***VALID, A Routine to Validate Input Data***, UCID-30121, 1975. J. E. Ramus, ***CONCORD, A Word Index Generator for Arbitrary Text Strings***, UCID-30126, 1975. P. R. Keller, R. A. Keir, T. W. Stullich, ***ND, A Program to Analyze Data on Air Pollution in the San Francisco Bay Area***, UCID-30120, 1975.

[6] M. Williams, S. Rose, ***Computer-Readable Bibliographic Data Bases*** - A directory and Data Source Book, ASIS, 1976, Revised 1977.

[7] The Technical Information Department at LLL has ongoing contracts with the following bibliographic information centers: ***Lockheed Information Services, Palo Alto, Ca; System Development Corporation, Santa Monica, Ca.; Bibliographic Retrieval Services, Inco, Scotia, NY; INFOBANK, Parsippany, NJ; Defense Research Development Test and Evaluation on Line System (DDC), Cameron Station, Va; DOE/RECON, Oak Ridge National Laboratory; Medline, National Library of Medicine, Bethesda, Md.***

[8] MASTER CONTROL has been used by the Technical Information Department at LLL since 1970 to process approximately 1,500 interest profiles for the professional staff at LLL against some dozen major computerized bibliographic data bases of the scientific literature.

[9] B. Marron, E. Fong, D. W. Fife, K. Rankin, ***A Study of Six University based Information Systems***, NBS Technical Note 781, 1973; D. W. Fife, K. Rankin, J. C. Walker, B. Marron, ***A Technical Index of Interactive Information Systems***, NBS Technical Note 819, 1974.

[10] D. R. Richards, ***An Overview of BDMS: The Berkeley Database Management System***, Refer to paper in these Proceedings.

[11] W. Draisin, ***CIRLS - A General Information Retrieval and Library System***, Los Alamos Scientific Laboratory.

[12] V. E. Hampel and J. A. Wade, ***MASTER CONTROL - A Unifying Free-Form Data***

Storage and Data Retrieval System for Dissimilar Data Bases, UCRL-71686, 1968, cf. ASIS Proceedings, San Francisco Meeting, 1969. Staff of the Information Research Group, ***MASTER CONTROL User's Manual***, M-066, 1975. R. W. Kuhn, *A Numeric Processor and Text Manipulator for the MASTER CONTROL Data Management System*, UCRL-52015, 1976.

[13] *STOFI - System TO Find Information*, A System of subroutines for manipulating hierarchically related data blocks, for large data files, in the BKY format, Lawrence Berkeley Laboratory.

[14] P. Kreps, *SEEDIS Monitor, A Socio-Economic Environmental Demographic Information System*, Lawrence Berkeley Laboratory, LBL-6440, 1977.

[15] E. B. Birss, S. E. Jones, D. R. Ries, J. W. Yeh, *Scientific Data Base Management at Lawrence Livermore Laboratory: Needs and a Prototype System*, UCRL-80146, 1977, presented in these proceedings.

[16] A. A. Brooks, *ORCHIS - Oak Ridge Computerized Hierarchical Information System*, ORNL-4929, 1973.

[17] R. C. Durfee, *ORRMIS - Oak Ridge Regional Modeling Information System*, ORNL-NSF-EP-73, 1974.

[18] *ERDA/DOE/RECON User's Manual*, TID-4586,1976.

[19] J. R. Hilley, *The JOSHUA System*, DPSTM-500, Savannah River Laboratory, Aiken, S.C.

[20] B. Greenglass, *Interim Procedures for Submission & Review of Information System Proposals*, ERDA Office of Program Management Support, March, 1977.

[21] The ERDA Office of Technical Information and the ERDA Office of Environmental Information Systems have supported several data management information systems at the National Laboratories.

[22] Refer to the *Federal Information Processing Standards Index*, FIPS PUB 12-2

[23] *National Standard Reference Data System*, Publication List, LP-81. 1976, National Bureau of Standards, Washington, D.C.

[24] D. Merrill, D. Austin, *ERDA Interlaboratory Working Group for Data Exchange (IWGDE), Progress Report*, LBL-5329, 1976.

[25] D. R. Ries, *Analyzing User Requirements for Data Management Systems*, UCRL-79440, 1977.

[26] V. E. Hampel, *Decision Making with Interactive Access to Administrative & Technological Data Bases*, UCRL-80353, 1977.

[27] UNIX is an operating system developed by Bell Laboratories for the PDP-11/70 computer. INGRES is a relational data base management system under active development by the Electronics Research Laboratory of the College of Engineering at the University of California in Berkeley.

[28] F. U. Wetzler, *Data Banks for R&D*, Research and Development Journal, June, 1977.

[29] R. W. Kuhn, *Development and Implementation of New Character Fonts at LLL*, UCID-17653, 1977.

[30] *1977 Picture Data Description & Management Workshop*, IEEE, 1977. cf. ACM/SIGGRAPHs.

[31] V. E. Hampel, *The Time for Atomic and Molecular Data Bases is Now*, UCRL-79286, 1977. Proceedings of the IAEA-199.

SCIENTIFIC DATA BASE MANAGEMENT AT LAWRENCE LIVERMORE LABORATORY: NEEDS AND A PROTOTYPE SYSTEM*

by

Edward W. Birss
Stephen E. Jones
Daniel R. Ries
Jeffry W. Yeh

Lawrence Livermore Laboratory
Livermore California

Abstract:

Lawrence Livermore Laboratory (LLL), with such diverse data applications as material compatibility, laser fusion, magnetic fusion, test, equation of state, weather, environmental and demographic data, has an acute need for a Scientific Data Base Management System (SDBMS). The large volume, the numeric values within an epsilon of accuracy, the unknown data relationships, the changing requirements, coupled with the overall goal of extracting new intelligence from the raw data, dictate a database system tailored toward scientific applications. Such an SDBMS should support scientific data types, a relational end user view, an interactive user language, interfaces to graphical and statistical packages, a programming language interface, interfaces to existing facilities, extensibility, portability, and use in a distributed environment.

Addressing these needs, LLL has begun a project to develop a scientific data base management system. A prototype has been implemented which uses a relational algebraic interactive user language. The software consists of a macro processor, a parser, a parse tree generator, a parse tree interpreter, semantic routines, and data base access routines. Currently, the database access routines utilize a CODASYL database system for data storage.

---

*This work was performed under the auspices of the U. S. Energy Research and Development Administration under contract No. W-7405-Eng-48.

Scientific Data Base Management at LLL

INTRODUCTION

Energy research at Lawrence Livermore Laboratory (LLL) increasingly requires management of scientific data by computer. The uniqueness of both the users' requirements and the computing environment dictate the development of a Scientific Data Base Management System (SDBMS). This paper illustrates these needs and provides an overview of the software development effort begun to fulfill them.

SCIENTIFIC DATA CHARACTERISTICS AT LLL

Livermore has large quantities of bulk data, numeric data within an epsilon of accuracy, unknown data relationships, and changing requirements. The magnitude of the amounts of data is illustrated by the Magnetic Fusion Test Facility (MFTF) project. The raw data produced by a half second shot will be on the order of two million bytes. Under maximal operational conditions, it is possible that one shot could occur every five minutes. Computer controlled instruments can generate up to four million ten-bit integers from one expermimental shot of SHIVA's 20 lasers. Much of this data consists of digital representations of photographs of laser beam cross sections. The volume of experimental data exemplified by MFTF and SHIVA dictates that the physicists have efficient access to the collected information.

Epsilon-accurate data elements are exhibited by the National Uranium Resource Evaluation (NURE) project. This program's geologists and physicists gather soil and water samples and test them to find new reserves of uranium. Sample tests generate medium volumes of floating point numbers. To manage the epsilon-accurate data, additional parameters are stored with each real quantity. Some examples are: labels (such as concentration of uranium), units (such as parts per billion (ppb)), and error quantities (such as plus or minus ppb). In working with such epsilon-accurate data, equality comparisons are not meaningful, instead small range comparisons are frequently used.

The NURE project also exhibits the characteristics of unknown data relationships and changing requirements. The goal to find uranium resources is clear enough and the most obvious approach is to look at samples with large uranium concentrations. Unfortunately, the uranium concentration is meaningless unless considered along with other environmental measurements and uranium concentrations in surrounding areas. As more is discovered from data analysis, new tests are added, the data changes and new relationships of significance are discovered between uranium and other elements.

PROCESSING SCIENTIFIC DATA AT LLL

There is a large computer user community at LLL. It consists of over 2000 physicists, chemists, engineers, and other scientists. Of these 2000, only about 300 are computer scientists. Nearly all of these scientists use the main computer facility which consists of four CDC 7600 and two CDC STAR-100 computers. Through a network of about 1000 terminals, the users interactively access any of the large computers. Peripheral devices are shared by the large computers through several networks.

Mini-computer usage at LLL is increasing dramatically. Many of these minis are utilized for data aquisition and reduction, and require data handling software compatible with that on the large computers. The data collected by the minis must often be compared with simulation results generated on the large machines. Thus data interface tools and translators are urgently needed.

With such a large user community, with diverse data including material compatibility, test, equation of state, weather, environmental and demographic data, the ultimate use of the data is seldom known. There are some generalizations of data usage that are of interest. One class of applications involves the collection and analysis of experimental data. Experimental data is analyzed to find out how an experiment worked. The knowledge gained by this analysis is then used to reconfigure the experiment. Comparisons are also made between the experimental results and the expected or theoretical results from analytical or simulation models to gain physics and experimental insight. This type of analysis is usually performed only once, and consequently a specific program is not justified.

On the other end of the spectrum, other types of applications are much more static in nature. Cataloging material properties, for example, is a rather static task. Although relatively static, these databases usually are the type that require periodic standardized reports. These databases last over long periods of time and change little.

GENERAL REQUIREMENTS FOR AN SDBMS

Based on these needs, the following are general requirements for a scientific DBMS at LLL.

SCIENTIFIC DATA TYPES - Since virtually all of the applications at LLL have a scientific orientation, the SDBMS must have data types suitable for storing scientific data. These are typically integers, floating point numbers, and multi-dimensional FORTRAN-like arrays. Additionally, vector data types for new vector processing machines such as the STAR-100 and the CRAY-1, plus character strings, bit strings, and date data types are needed.

RELATIONAL END USER VIEW - The relational data model [1] views data as a set of interrelated relations which may be thought of as tables. The simplicity of this model, its tablular view, and its ability to establish relationships dynamically make the relational approach well suited for scientific applications.

INTERACTIVE USER LANGUAGE - The need to perform one time analysis makes a general, interactive user language system attractive. With such a system, specific programs need not be designed, written, and debugged. Furthermore, a good user language helps one get more useful information out of the database with less effort. A mechanism should be included to enable the user to tailor this interactive language to his application.

INTERFACES TO GRAPHICAL AND STATICAL PACKAGES - Because of the numeric nature of scientific data, graphical, statistical, and numerical

analysis tools are a needed extension to the data manipulation capabilities of a DBMS. Such tools are used by scientists to visualize and analyze their data, and to discover relationships not known a priori.

PROGRAMMING LANGUAGE INTERFACE - Interfaces to programming languages such as FORTRAN, are needed for sophisticated users who, for performance reasons or for a highly tailored user interface, wish to write their own specialized programs that access the database. In addition to permitting special purpose interfaces, a programming language interface will permit adoption of other analysis packages such as graphics, pattern recognition and statistics programs that presently use input data files.

INTERFACES TO EXISTING FACILITIES - Links to existing LLL facilities such as text editors, report writers, secondary and tertiary storage devices, etc, are important in making the system easy to use.

EXTENSIBILITY - The system must be extensible to adapt to changing requirements. Supplementing the language, for example, should not require extensive software modification. Creating sequences of commands should be facilitated.

PORTABILITY - The SDBMS must be able to operate on a wide variety of computer types. It must operate on the CDC 7600 and STAR computers, and on large minicomputers as well. Such portability enhances the ability to share data, but dictates that the code be independent of word size, character set, and other hardware dependencies.

DISTRIBUTED ENVIRONMENT - To be most valuable at LLL, the SDBMS should evolve into a system suitable for use in a distributed environment. Databases on one machine should be accessible and modifiable by users on other machines without user-specific knowledge of the location of the data.

THE PROTOTYPE SDBMS

Two primary objectives guided our implementation of the SDBMS prototype: it must become operational quickly and be easily modified. We wanted it operational quickly to demonstrate SDBMS capabilities and to enable users to suggest improvements. We want to iteratively enhance the user language to ensure that the language has the desired characteristics. To accomplish these objectives, the language will be tested on a limited scale, and enhanced prior to general release. The iterative scheme is effective in demonstrating SDBMS capabilities to users and management, and is instrumental in obtaining support.

Our prototype must be easily modifiable so that the iterative development described above is achievable without extensive software revision, and so that the software maintenence activities are minimized. We thus have adhered to strict and uniform coding standards which stress logical clarity in coding and have documented the code extensively. Our subroutines are designed to be logically independent so that entire portions might be replaced without side effects. We chose standard FORTRAN as an implementation language because it is by far the most widely used language at LLL, and because it is available both on large

scientific computers and on minis. In order to hide word length and character set dependencies, we use functions and subroutines for all bit and character string handling.

We have done both design and implementation in a top-down fashion. The user's view of the system was designed, primarily consisting of the user language. That language was implemented and debugged as the underlying functions were being formulated. We still have temporary structures underlying our system which will be replaced as the system solidifies.

THE PROTOTYPE'S USER LANGUAGE

The SDBMS user language is procedural, based on the relational algebra [2]. It is slightly "lower level" than some of the popular research languages such as QUEL [3] and SEQUEL 2 [4], but we feel it is simpler and more appropriate for our user community. We are not initially addressing the query optimization issues that must be solved for a non-procedural language to achieve acceptable performance.

An SDBMS database consists of a set of TABLEs (relations) which have COLUMNs (attributes) containing values from a DOMAIN of values. For example, a WEIGHT column might contain values from the domain of positive real numbers. We plan to associate integrity assertions with DOMAINs much as does McLeod [5]. DOMAINs and COLUMNs may be of the following data types:

FLOATING	- floating point number.
INTEGER [N]	- fixed length number N bits long (default length is a machine word).
CHAR N	- fixed length character string of length N characters.
TEXT	- variable length character string with leading and trailing spaces deleted and multiple internal spaces reduced to one.
TEXTB	- variable length character string in which blanks are significant.
DATE	- date; can be input and output in a variety of formats but is stored uniformly internally.
BIT	- uninterpretted bit string. primarily for bulk, foreign data.

Additionally, the first three types can also be aggregated into VECTORs and ARRAYs. These types provide for direct interfacing to FORTRAN programs using these structures.

To illustrate our language, consider a portion of a database from the NURE project. A SAMPLE table contains information about each sample taken in the field. Columns are SAMP-ID (a unique identifier), SAMP-TYPE, DATE-COL (date collected), and COMMENTS. Samples, when analyzed, produce many measurements per sample. Table MEASURE contains columns indicating the SAMP-ID of the sample, an ELEMENT found in the sample, its concentration in parts-per-billion (PPB), the ERR of the measurement (plus or minus ppb) and the date when the analysis was performed (DATE-ANAL).

The statements defining these tables are:

```
DEFINE TABLE SAMPLE
  SAMP-ID  CHAR 10,
  SAMP-TYPE CHAR 6,
  DATE-COL  DATE,
  COMMENTS  TEXT;

DEFINE TABLE MEASURE
  SAMP-ID  CHAR 10,
  ANAL-TYPE CHAR 10,
  ELEMENT CHAR 3,
  PPB   FLOATING,
  ERR FLOATING,
  DATE-ANAL DATE;
```

Input is accepted in free format; the spacing in these examples is for clarity only.

Most manipulation is done with the relational assignment statement, in which one or more tables (relations) are operated upon yielding a new table. Queries to the database are realized by creating a result table containing the answer(s). We will illustrate a few commands, including the three basic table operations.

1. PROJECT - This operation copies desired columns from a table into a new table.

```
TEMP1 = SAMPLE PROJ SAMP-ID SAMP-TYPE;
```

   The result is a new table named TEMP1 with columns SAMP-ID and SAMP-TYPE from the SAMPLE table. It may be printed by

```
PRINT TEMP1;
```

2. SELECT - Desired rows may be selected from one table creating a new table based on the value of an arbitrarily complex boolean function.

```
TEMP2 = MEASURE WHERE ELEMENT = "CL" OR
     (ELEMENT = "U" AND PPB > 500);
```

   TEMP2 is created with all the columns of MEASURE but contains only rows for chlorine measurements or uranium measurements with concentrations of greater than 500 parts per billion.

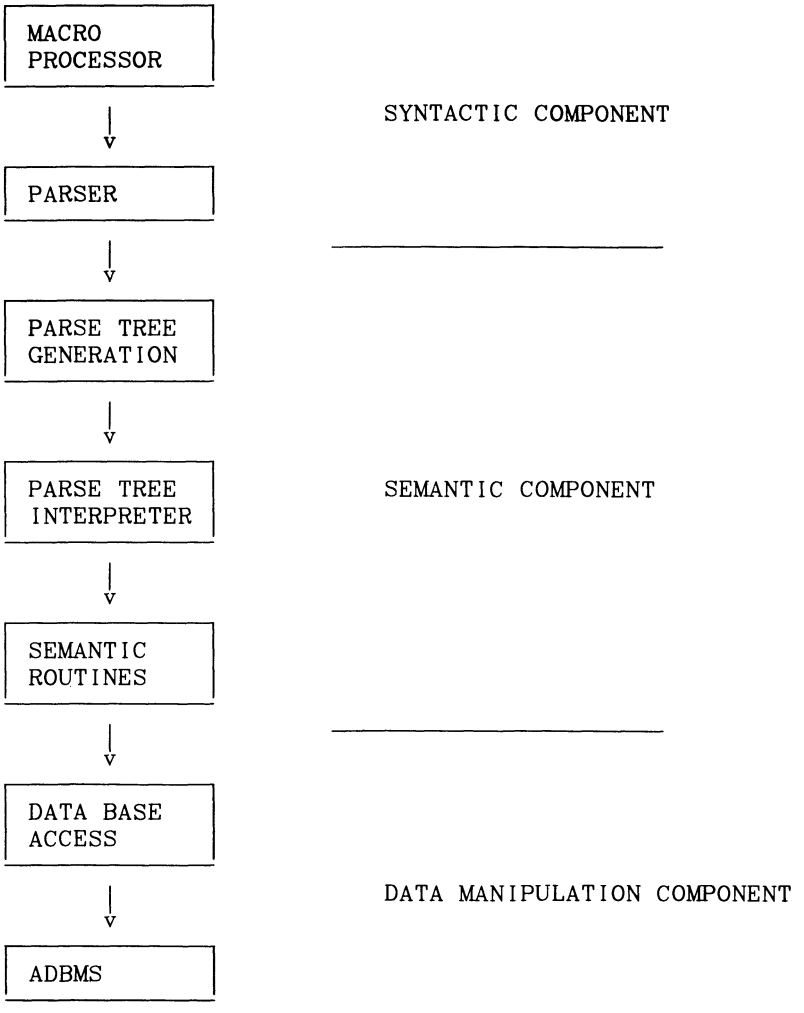
3. JOIN - Tables may be combined based on equal values in designated columns.

```
TEMP3 = (JOIN MEASURE WITH SAMPLE ON SAMP-ID)
       PROJ SAMP-TYPE ELEMENT;
```

   This statement combines the rows from SAMPLE with those from MEASURE where the SAMP-ID columns in each match. It then projects only SAMP-TYPE and ELEMENT causing TEMP3 to be a table showing all the elements found in each sample type.

ARCHITECTURE AND EXECUTION

The architecture of the prototype may be viewed as having seven logical levels:



A distributed macro processor which operates on lexical tokens sits between the parser's lexical scanner and the parser itself. The parser is an LALR type, based on the work of DeReemer [6]. It is table driven, the tables being produced by a separate grammar analyzer. Thus changes to the language require no changes to the parser (although code to process the commands must be written). Having the parser independent of the language greatly eases language modification. The macro processor and parser are quite logically separate from the rest of the program, having been previously developed for another project.

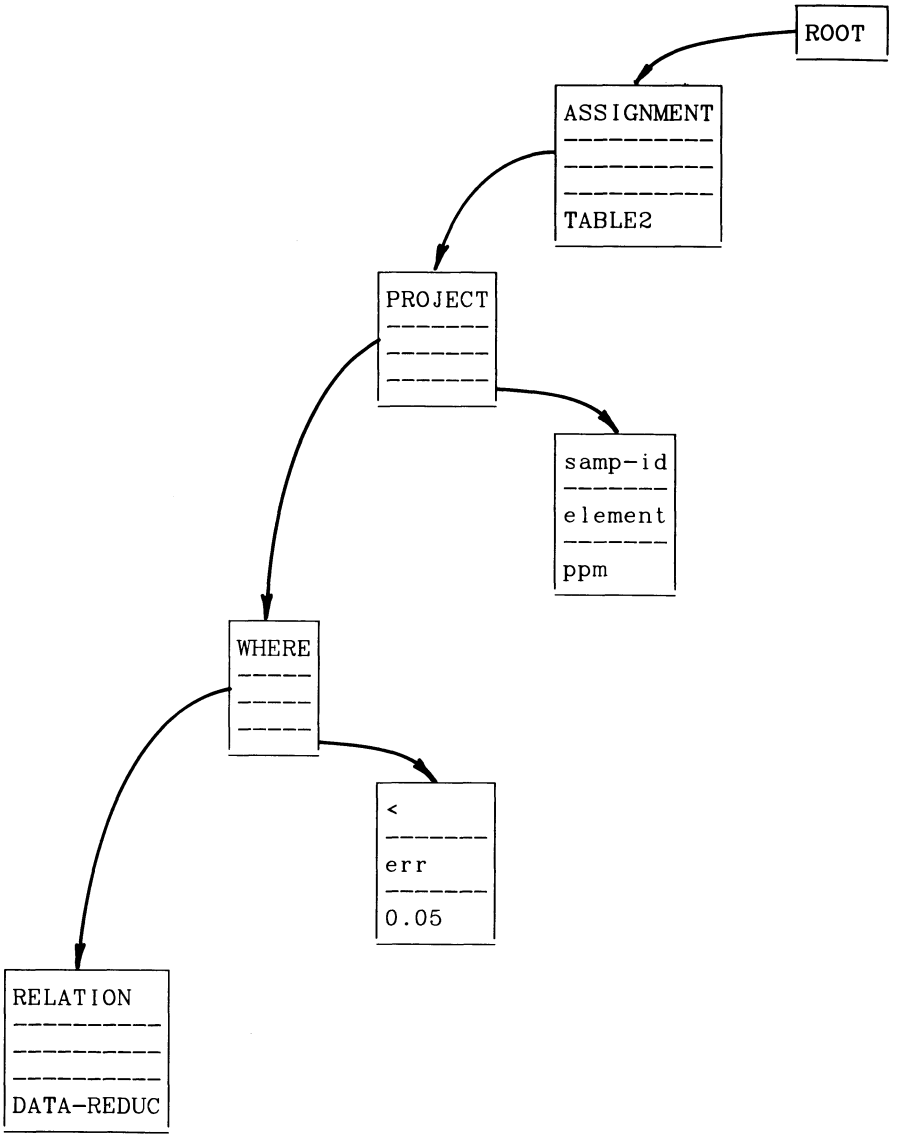
As the statements are parsed a tree structure of the command is generated. Execution of the command occurs as this tree is traversed and pruned. This tree is used as an intermediate structure of the command:

* to assure syntactic correctness of the entire command before it processed.
* to store information that will be used repeatedly in the execution the command.
* to allow global knowledge of the command's meaning thus allowing optimized processing of the request.

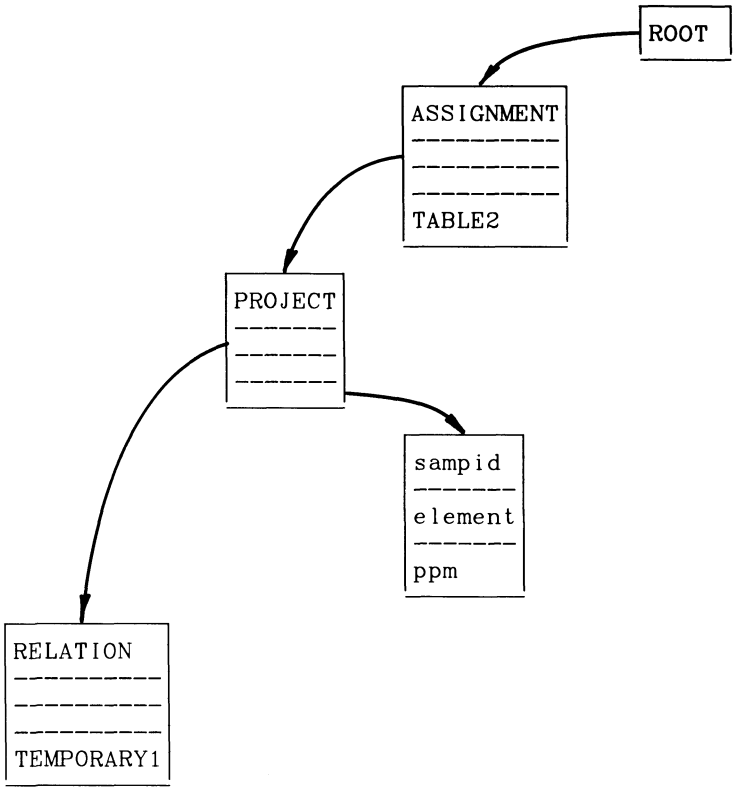
Nodes in the tree usually represent relations or operations which yield a relation. Additional information is stored in auxilliary nodes which are attached to the tree. During a postorder traversal of the tree, visiting nodes triggers actions which result in that node and its subtrees being reduced to a terminal node (relation).
Example:

```
TABLE2 = (MEASURE WHERE ERR < 0.05) PROJ ELEMENT PPB;
```

When this statement is parsed, the following tree is constructed: (The nodes with capital letters are tree nodes, the others are auxilliary information).



Traversal of the tree causes first the evaluation of the WHERE node resulting in a temporary relation created from MEASURE. The tree would then look like this:



Visiting the PROJECT node similarly yields another temporary relation which in turn is renamed TABLE2 when the ASSIGNMENT node is visited.

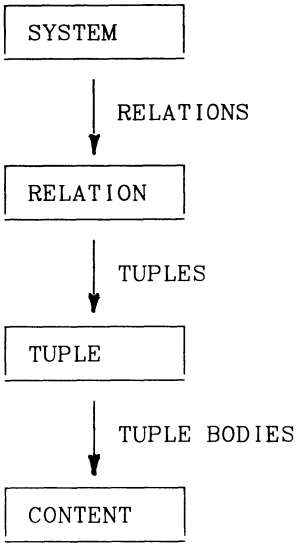
The semantic routines are those which perform the action specified in the language (select, project, join, print, etc.). They are independent of the tree traversal just described; they do "atomic" functions given straightforward FORTRAN arguments as input. They typically have one or two relations and some auxilliary information as input and produce a temporary relation as output. Thus these routines perform the basic relational functions while the command tree manipulation routines have the global awareness of the command processing.

The relational access routines form the interface between the semantic routines and the data. They are logical rather than physical in scope. Functions include open a relation, get a tuple (row), store a tuple, etc. The semantic routines have no knowledge of the underlying data structures or I/O mechanisms.

The access routines are currently built upon a CODASYL [7] type DBMS called ADBMS [8]. We used ADBMS in this way primarily to speed development. This was much faster than writing our own I/O subsystem and thus allows earlier development of the upper level code. We plan to

replace the CODASYL system with our own access methods at a later date. The system is simplified by having working data, system data, and user data all stored as relations and accessed by the same routines. The data description components of the SDBMS are stored in the database as system-owned relations. Similar to INGRES [9], these include a relation that describes relations in the database, a relation that describes attributes and a relation that describes domains.

The ADBMS schema is a very simple structure that models a relation:



Hence an SDBMS data base consists of a set of relations which in turn consists of a set of tuples. A tuple consists of a set of tuple content records, a mechanism for implementing tuples of various lengths. This simple schema has the advantage that there is little reliance on the ADBMS functions, thus replacement is eased. It is necessary to achieve dynamic characteristics in a relational system that are unavailable in CODASYL systems (such as creating new relations at runtime).

The disadvantages of using ADBMS are size and performance penalties. We are certainly not using most of its capabilities and are paying for unneeded functions.

CURRENT STATUS AND FUTURE PLANS

We started in April of 1977 and have had two to three people working on the prototype and are encouraged with the results so far. Currently (September 1977) most of the prototype's commands are implemented, JOIN being the notable exception. It runs in 45K (decimal) words. It should be smaller when the code is overlayed and ADBMS is replaced.

Our future plans are divided into short-term and long-term plans. In the short-term we want to complete the user interface. This task

involves adding graphics and statistics interfaces, views and derived attributes, tertiary storage management, and perhaps a report-writer. Views in this context are an application program's expectations of data stored in the data base, and a derived attribute is an item derived from other attributes stored in the same relation. In addition to completing the user interface, we also intend to enhance the capabilities and performance of the SDBMS. We plan to replace the CODASYL base understructure and develop our own access methods. A variety of access methods are invisioned, including direct, indexed, and sequential. We may also develop access methods that have flexibile clustering mechanisms, for example, the ability to store tuples of one relation physically close to tuples of other relations (where keys of the tuples are similar).

Our long-term plans include developing a programming language interface, query optimization, performance optimization base on usage patterns, integrity constraints, and data base administration utilities. We also plan to convert the system to large minicomputers and use the system in a distributed environment.

REFERENCES

[1] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," CACM 13,6 (June 1970) pp. 377-397.

[2] C. J. Date, An Introduction to Database Systems (Addison Wesley, Reading, Mass. 1975).

[3] G. D. Held et. al., "INGRES -- A Relational Database System," Proc. 1975 NCC, pp. 409-416.

[4] D. D. Chamberlin et. al., "Sequel 2: A Unified Approach to Data Definition, Manipulation, and Control," IBM Journal of Research and Development (November 1976) pp. 560-575.

[5] D. J. McLeod, "High Level Definition of Abstract Domains in a Relational Data Base System," Computer Languages 2(1977) pp. 61,73.

[6] F. L. DeReemer, "Simple LR(k) Grammars," CACM 14,7 (July 1971) pp. 453-459.

[7] CODASYL Data Base Task Group Report, April 1971, ACM, New York.

[8] E. W. Birss, ADBMS User's Guide, LLL report UCID 17417, February 1977.

[9] M. Stonebraker et. al., "The Design and Implementation of INGRES," ACM Transactions on Database Systems 1,3 (September 1976) pp. 189-222.

LBL-7201

AN OVERVIEW OF BDMS: THE BERKELEY DATABASE MANAGEMENT SYSTEM*

David R. Richards
Lawrence Berkeley Laboratory
University of California
Berkeley, California

ABSTRACT

The design and implementation of BDMS is outlined with emphasis on those features of particular importance to the management of complex scientific data having significant numerical content. These include binary internal representation of integer and single or double precision real data element types, ability to handle vector values of numeric data elements as well as arbitrary length character and bit strings, and the provision of an extensive set of "hooks" for the attachment of user-supplied processing routines.

INTRODUCTION

BDMS, the Berkeley Database Management System[1], is a hierarchical database management system whose design was heavily influenced by the requirements of scientific data management. It has its origin in the joint development of a Particle Physics Data System (PPDS)[2,3] by the Berkeley Particle Data Group (PDG) and the Caltech Data Compilation Group.

The original conception of the PPDS software called for a separate systen to manage each of the databases: a document system, a reaction data system, a particle properties system, a vocabulary control system, etc. In fact, by mid-1973 a prototype of each of these systems had been developed. It rapidly became clear, however, that the task of modifying, extending, and even maintaining these specialized systems was beyond available resources. There was ample reason to expect that the requirements for these systems would never stop changing; the nature of particle physics data had changed significantly over the past decade as the field evolved and was expected to continue changing.

Searching for a different approach, we realized that it should be possible to use a single database management system that would be general enough to manage all the databases and serve as the basis for the specialized software that would still be necessary. The broad spectrum of size, growth rate, volatility, and complexity encompassed by these databases and the complexity of their intended uses demanded capabilities not provided by any database management system then in existence that could be run on our hardware (CDC 6400, 6600, 7600). Thus, we were led to develop our own system.

We are now painfully aware of the very large commitment of time and resources necessary for a project of this magnitude. In fact, our naivete at that time was probably a major factor contributing to the eventual attainment of our goals, since if we had realized how difficult it would be, we might never have embarked on the project, instead simply reducing our expectations.

*This work was done with support from the U.S Department of Energy.

There is, on the other hand, a clear advantage in local control of system development. When some new capability is needed, it is possible to make a decision as to whether it is a general facility that is best built into the database system once and for all, or whether it deserves only to be put into the application software. With a system supplied by a commerical vendor, some very complicated and unattractive kludges may be necessary to add a capability, since the only option is building it outside of the existing system.

In the rest of this paper, the structure of a BDMS database will be defined and the facilities of the system (version 2.1) outlined. Some comments will be made concerning those aspects of the implementation that might affect its suitability for a proposed application. Finally, our plans for future development will be outlined.

STRUCTURE OF A BDMS DATABASE

A BDMS database is structured into records, the units in which data pass between the system and disk storage. Normally, a record will have some significance to the user, e.g. a record in a bibliographic database would be a description of a single document, but this is not always necessary or desirable.

The individual data items within a record are called data elements; they are the smallest units of data with any meaning to the system, although an individual data element might have some internal structure known to an application program. A data element has a unique name and is normally referenced by name (or a synonym). There is essentially no limit on the number of data elements that may be defined for a database.

A hierarchical structure may be imposed on the information within a record when a database is defined. This means that some data elements are declared to be subordinate to other, parent, data elements. Those data elements for which no parent is declared are called record-level data elements; it is often useful to consider the record itself to be their parent. There is essentially no limit on the number of levels which may be defined in the record structure.

Within a given record, each record-level data element may occur once, several times, or not at all. Likewise, each occurrence of a data element at any level in the hierarchy may have linked to it one, several, or no occurrences of each of its subordinate data elements. There is essentially no limit on the number of times any data element may occur in a single record.

Data elements are classified into six types according to the values they can assume. INTEGER, REAL, DOUBLE, CHAR, BIT, and NODE. Integer or real (floating point) data elements may be scalars (single numbers) or arbitrary length vectors (i.e. an ordered set of numbers, which are the components of the vector). Real data elements may be single or double precision. Character or bit strings may be of any length with no limit beyond that imposed by run-time memory restrictions. Any of the foregoing data element types may occur with a null value. Pure node data elements carry no value; they may be used to link together subordinate data elements in the record hierarchy or as flags.

Any data element, regardless of type, may serve as a node in the hierarchical record structure. In general, if one of a group of related data elements may occur only once in each occurrence of the group, it should be made the parent of the rest of the data elements. However, if all of the data elements in such a group may occur multiply, they all must

be linked to a pure node parent, since no single occurrence of any one of them can serve as the parent of the rest.

Any data element may be declared to be a record key. The system will then maintain an index for that data element to allow efficient retrieval. In an index, key values have a fixed length that is declared in the database definition; data element values are truncated or padded as necessary when they are put into an index.

It is possible to declare a data element to be virtual, which means that it will be recognized in input data and appear in the record buffer, accessible to user-supplied processing routines, but will be discarded when the record is stored in the database. This is useful if the input contains data elements that one does not wish to store in the database at all. One might also wish to store some data element values only in the indices, where they point to the record, without permanently allocating space for them in the record itself. This is often the case if a key value is constructed from the values of one or more data elements by a user-supplied processing routine (discussed below). In this case, a virtual data element is defined and is further specified to be a key. It is then used to hold the key values temporarily; when the record is stored, these values will be indexed before the virtual data element is discarded.

The system assigns a record ID to each record as it is created. This guarantees that each record has a unique identifier by which it can be selectively retrieved even if none of its data elements is defined to be a key so that no indices are maintained. The ID is displayed whenever a record is listed by the system.

BDMS FACILITIES

BDMS comprises a database definition compiler, a database executive, and several utility programs.

The database definition compiler is used to create a new database. It accepts a description of the logical record structure expressed in a database definition language (DDL) and generates tables describing the database. These tables then drive the rest of the system when that database is in use.

The database executive is a self-contained system providing a user interface to database maintenance and retrieval facilities through a high-level language. It has two major subsystems: an editor and a query language processor.

The editor permits a user to enter data into a new record or modify an existing one by appending, inserting, replacing, or deleting data element occurrences by means of free-format editing commands. A string substitution facility is provided for modifying the values of character string data elements. Any change to the database made via the executive is immediately effective and reflected in the indices.

The BDMS query language permits a user to search a database for those records satisfying an arbitrarily complex condition on key (indexed) data element values. The condition is constructed as a Boolean combination of key value specifications, including inequalities and ranges. Furthermore, it is possible to search for records having an occurrence of a specified data element regardless of value, or for those having an occurrence of the data element with a null value. Truncated value specification for character string keys may be used to search for those records having an occurrence of the data element beginning in a particular way. A par-

ticular record ID or range of record IDs may be included in a query explicitly. The result of a query is the set of records that satisfy it. Any of them may be listed at the user's terminal, printed, dumped, modified, or deleted. Existing sets may be combined with other sets and still further conditions through the query language.

The utility programs provide for efficient initial database loading, full database dump, data file garbage collection, and rebuilding of indices for more efficient query processing and disk space utilization.

Except for the utilities, which are batch programs, the system may be used in either batch or interactive mode by simply linking it to the proper set of low-level I/O routines. The user language is identical in either mode. BDMS also may be invoked procedurally from a user-written FORTRAN program by means of a small and carefully chosen set of subroutine calls.

The database executive incorporates exits to user-supplied processing routines to allow input data validation, and data transformation on input, output, during the creation of index entries, and the processing of queries. In addition, the user may supply routines that are called just before a record is stored or just after it is fetched from the disk. The store processor routine may perform more complex data validation involving correlation of several data element values and may generate additional data element values, e.g., keyed virtual data elements as discussed above. The fetch processor routine is primarily useful to rematerialize virtual data elements when it is desirable to make them visible to a user or application program.

These "subroutine hooks" provide a powerful facility for tailoring the database executive for specific databases and applications without requiring the user to write a completely new "front end." They are particularly important to the management of scientific databases since even input data validation is likely to be too complex to be handled by the simple facilities (e.g. within a specified range or explicit list of values) typically provided by a commercially-oriented database management system.

IMPLEMENTATION

A BDMS database is divided into three system disk files: the data file, which contains the database definition and data records, the directory file, which contains the physical storage addresses of data records, and the inversion file, on which reside indices for key data elements.

On the disk, the hierarchical structure of a data record is represented by unidirectional pointers linking together the data element occurrences. There is no storage overhead associated with data elements that do not occur at all, either in the record or a particular occurrence of their parent. When a record is brought into the record buffer, it is restructured to facilitate access--the pointers are made bidirectional and relocated relative to the start of the work area, and entry pointers are allocated for all missing data elements to allow insertion. When the record is converted back into its disk-resident form prior to storage, all garbage resulting from updating activity is automatically eliminated.

If a modified record is no longer than it was prior to modification, it is stored back in its original location on the data file and any unused space following it is flagged as deleted for the data file garbage collection utility. If the modified record is longer than it was, it is written at the end of the data file, its directory file entry is updated, and the original form of the record is flagged as deleted.

Numeric data element values are stored in the database in the internal binary format of the machine being used. This decision was based upon the assumption that much of the use of a scientifically-oriented database management system involves access to the data by analysis software requiring numeric values in internal binary form. They must be converted into this form from their character representation only when initially input rather than every time they are read by an analysis program, as would be the case if they were stored in character form.

Paged, multilevel tree-structured indices are maintained by the system for data elements declared to be keys. These indices are updated automatically whenever a new or modified record is stored in the database or a record is deleted. A "leaf" entry in an index tree is a unique value for the indexed data element followed by a list of record IDs for all records containing an occurrence of that data element with that value. The query language processor performs Boolean operations by merging these lists of records.

A single access to the directory file, with an entry address calculated from a record ID, provides the disk address and length of that record on the data file. This level of indirection was provided so that it is not necessary to update all index entries for a record that has been made larger by an update and has to be relocated; the only index entries that need to be changed are those for data elements whose values were actually modified. Likewise, when records are moved by the data file garbage collection utility, only their directory file entries need to be changed; all index entries remain correct.

The BKY operating system, under which BDMS is run at LBL, makes no provision for re-entrant code or updatable shared disk files, so BDMS was built as a single user system and does not support updatable shared databases. However, BKY does allow shared access to a read-only ("public") disk file, so multiple users, each with his own copy of the database system, can retrieve information from a common disk-resident database.

Likewise, BKY does not support permanent disk files, so elaborate crash recovery machinery in BDMS was not deemed necessary. Normally, a database is stored on tape, staged to a disk for use, and then staged back to another tape if it has been modified. One can then "roll back" to a previous state of the database provided that a sufficient number of tapes is used in the storage cycle.

The only situation in which a system crash can be a major annoyance is when an interactive user has made extensive changes to a database immediately before the crash. To save such a user from having to re-key all those changes, the system records all user input on an audit file. If this file is maintained in such a way that it survives the crash (tape would be virtually foolproof), it can then be processed as batch input against the original version of the database. The audit file can also serve as a record of update activity if it is preserved as a part of normal operating procedure.

Since each user either uses a read-only public file or has his own copy of the database, we did not feel it was necessary to design an elaborate security mechanism to prevent unauthorized update. We are considering the addition of a password scheme, access control at the data element level, and possibly even selective encryption to ensure privacy of sensitive information.

One of the design goals was easy transportibility. This was achieved by a) writing the major part of the system in a relatively machine-

independent subset of FORTRAN IV, and b) careful modularization to isolate machine and operating system dependence in a few low-level interface routines that can be recoded easily for another system. Versions of BDMS exist now for CDC 6600 and 7600 computer running BKY, and IBM 360 series machines. An experimental 7600 version has been run under SCOPE and we are currently working on a PDP-11 version for RSX and IAS systems.

FUTURE PLANS

Current plans for development of BDMS fall into two major areas: extension to allow multiple record types ("multi-file databases"), and enhancement of the query facility to handle intra-record and non-key data element qualification. A consequence of providing multiple record types will be the ability to modify easily the definition of an existing database. The ability to formulate queries involving intra-record qualification will allow retrieval conditions to be specified in terms of the relative position of data element occurrences in the record hierarchy. We are also considering the addition of multidimensional array data element types as well as the access protection machinery mentioned in the previous section.

As BDMS has evolved, many people have contributed ideas and assisted with programming. They include Tricia Coffeen, Paul Chan, Geoffrey Fox, Marge Hutchinson, Silvia Sorell, Paul Stevens, Gill Ringland, Alan Rittenberg, Deane Merrill, Tom Lasinski, Tom Trippe, Vicky White, and George Yost. Over the course of development, support has been provided by the National Science Foundation and the National Bureau of Standards, in addition to that of the U.S. Department of Energy (in its previous incarnations as the U.S. Atomic Energy Commission and later the U.S. Energy Research and Development Administration).

REFERENCES

1. Richards, D.R., BDMS User's Manual, LBL-4683 (Revision 1); BDMS Programmer's Manual, LBL-4684; BDMS Implementation Manual, LBL-4685.

2. Berkeley Particle Data Group, Particle Physics Data System Document File, PDG-3100; Particle Physics Data System Reaction-Data File, PDG-3200.

3. Stevens, P.R., and Rittenberg, A., The Particle Data Group: Using a GDMS to Solve Data Handling Problems in Particle Physics, CALT-68-622, contribution to this study.

EXTENSIONS TO THE JOSHUA GDMS TO SUPPORT ENVIRONMENTAL SCIENCE AND ANALYSIS DATA HANDLING REQUIREMENTS*

J. E. Suich and H. C. Honeck
Savannah River Laboratory
E. I. du Pont de Nemours and Company
Aiken, South Carolina, U.S.A.

SUMMARY

For the past ten years, a generalized data management system (GDMS) called JOSHUA has been in use at the Savannah River Laboratory. Originally designed and implemented to support nuclear reactor physics and safety computational applications, the system is now also supporting environmental science modeling and impact assessment.

Extensions to the original system are being developed to meet new data handling requirements, which include more general owner-member record relationships occurring in geographically encoded data sets, unstructured (relational) inquiry capability, cartographic analysis and display, and offsite data exchange.

This paper discusses the need for these capabilities, places them in perspective as generic scientific data management activities, and presents the planned context-free extensions to the basic JOSHUA GDMS.

APPLICATIONS OVERVIEW

In recent years, there has been considerable discussion of the application of data base management techniques to environmental data.[1] At the Savannah River Laboratory (SRL), the concern has centered around the Environmental Transport Division's (ETD) computer applications[2] which naturally involve the JOSHUA data management system.[3] In this manuscript, the JOSHUA system is the given basis for data management, and the emphasis will be on necessary extensions to support current and planned ETD applications. Clearly, not all applications demand new data management or even new systems extensions, but may be supported by new applications programs.

---

* The information contained in this article was developed during the course of work under Contract No. AT(07-2)-1 with the United States Energy Research and Development Administration.

In this manuscript, these teleological distinctions will not be much in evidence because at the present level of conceptualization one cannot distinguish the exact degree of generality an implementation will possess. However, the areas of most general need for new systematic capabilities probably shade from context-free data management (data base exchange), through context-free data manipulation services (information retrieval), to context-dependent data manipulation services (geocoded data management), and, finally, context-free data analysis (statistical analysis) for developing applications based on generic functions (geographical data manipulation and display). All such extensions will here be considered to be systems because they are not typical of programming done for any specific end-use and are sufficiently involved with the JOSHUA systems routines to require development and maintenance by Computer Sciences Section (CSS) personnel.

The JOSHUA data management facilities were developed for reactor physics applications. Subsequent use of these facilities for environmental science applications is straightforward where there is a computational similarity to the original application. However, the JOSHUA facilities require systems extensions to accommodate distinctively different computational features.

One case in point is the JOSHUA emergency and routine environmental impact assessment handler (JEREMIAH) subsystem, currently being developed under native JOSHUA capabilities. This computational system for advective-diffusive atmospheric transport and dose effects is implemented in exactly the same coupled module/data base approach as that used for nuclear reactor physics and engineering applications.[3] Taken as a system applicable to any geographical region, JEREMIAH would founder on JOSHUA's lack of facilities to deal with geographically encoded data. Taken as a system specific to the Savannah River Project site, however, the small scale of the region, the levelness of the terrain, and the fixed map locale have permitted the geographical problem to be made subordinate to JOSHUA's non-geographical, indexed (x,y) list handling capability.[4,5] This requirement for geographical data manipulation, analysis, and mapping is a generic one for ETD's applications, and constitutes the highest priority need for JOSHUA data management extension.

A second ETD application which illustrates the need for extended data management capability is the environmental impact study for the USERDA alternative fuel cycle technology study (AFCT). Because of the scope of this study, data bases and models at several ERDA and NOAA laboratories and a variety of computer systems have been used. The JOSHUA system contemplated an offsite data base exchange only with the National Neutron Cross Section Center (NNCSC), and there only for a single, clearly defined, cross-section data set which could be defined to the system (in fact, this data set structure and definition was the system model). Thus, JOSHUA inherently incorporates the mechanism for definition of the necessary AFCT data sets, but provides no means for data set interchange beyond the NNCSC specific tape load routines. The ERDA Interlaboratory Working Group for Data Exchange (IWGDE) has been developing standards[6,7] for data base exchange and PL/1-FORTRAN interface routines to implement those standards. The JOSHUA data set definition facility now needs to be extended to permit input from, and output to, data sets in the exchange standard format. Recognition should be given to differences among SRL and other data centers in the use of formats, nomenclature, geocodes, and data structures. Much of the process of translation can be automated, and the present manpower intensive situation can be significantly improved.

The process of data archiving, retrieval, and evaluation was presumed to have been done at the NNCSC, and additional support by the JOSHUA data management system was not contemplated. Several aspects of ETD's computing activities fall in this category, including ecological experimental studies and assessment simulation studies. Because data evaluation is a relatively unstructured activity (it generally means whatever the individual evaluator does), it is difficult to support these activities at other than a very basic, context-free level. However, three such generic activities in data evaluation are evident in ETD programs: unstructured inquiry, statistical analysis, and data display.

The JOSHUA system provides for FORTRAN retrieval of an entire data record, given its complete name, and simply catalogues the FORTRAN I/O list to format the data record for terminal users. In addition, FORTRAN callable routines and terminal commands are available to retrieve record names based on their unique names or their hierarchical tree structure. These routines and commands have been a very powerful and useful set of facilities for retrieving information for those applications in which inquiry keys can be thought out in advance and the inquiry response structured as hierarchically organized lists of names. This retrieval ability will be referred to here as structured inquiry to connote both the implied high degree of name and data organization, and the predefined problem-solving which that structure supports. The unstructured inquiry, which characterizes data evaluation, violates both notions of structure. The data are often organized by experimental observation or acquisition number, with no predefined record name inquiry keys; and the inquiries likely to arise during evaluation are not predefined, but are themselves data-dependent.

Unstructured information retrieval can readily be supported by a relational set of data bases, in which what is placed under control of the inquirer is the definition of relations among the data elements themselves. In this inquiry mode, the inquirer uses names as usual to select records for a subsequent search through the record itself to find data elements obeying the defined relational expression. All such groups of elements then constitute a sequential set on which an inquiry is simply answered by elementary operations--count, average, variance, sort, list, etc. To implement these operations under JOSHUA, commands must be provided for record selection and retrieval of data elements within a record which satisfy the inquiry criteria. These commands are commonly implemented as arithmetic and Boolean logical operations on the data elements encountered during record retrieval.[8,9]

The requirement for statistical analysis of experimental or observational data is basic to both data evaluation and to ecosystems modeling (e.g., Reference 10) which are generally what the soft environmental sciences involve. The attractiveness of satellite packages, e.g., the Savannah River Project deer file (SASS) and the forest energy balance model (CSMP), is that the satellite packages offer an extensive set of statistical and time series analyses coupled with crude data management to extract a relational set and operate on it through a user-oriented command language. Until JOSHUA offers comparable facilities, the environmental data base at SRL will remain fragmented and not supportive of integrated environmental model development.

The third general area where basic help can be given to evaluators is in computer graphics. The cartographic capability[9] mentioned earlier is probably the most useful addition to present systems and has been actively sought by ETD for the past two years. This capability, when associated with thematic data, will enable the user to treat the graphics

terminal as a map-like view of his data base, in contrast to today's numerical listing capability. Contouring, vector fields, and other typical graphic styles of data display should be implemented under the system as automatic display features.

DATA MANAGEMENT OVERVIEW

In this manuscript, the spectrum of data processing activities are segmented into three categories (Figure 1).

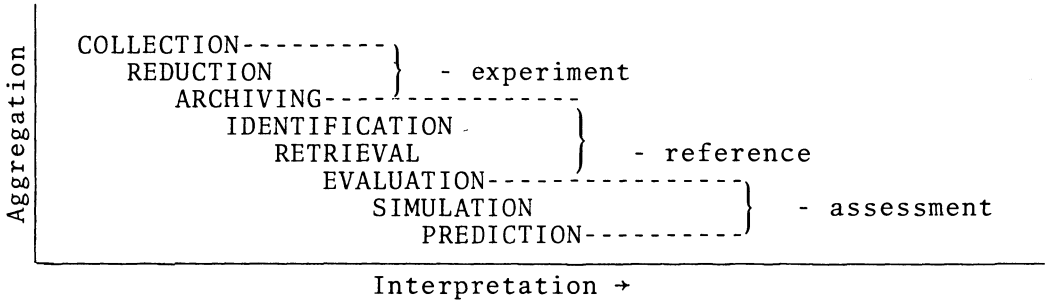


Figure 1. Data Processing Categories

The systems involved in managing experimental data are at the lowest level of data aggregation and interpretation. Such systems are least susceptible to generalization; their function is dictated by the experimental equipment, data quality control, and so on. It is probably unavoidable that each experimental project must develop appropriate data management approaches to deliver its end-product to an archive.

At the other end of the spectrum are the assessment simulation-modeling systems. Characterized by mathematical models, these are typically FORTRAN computer programs which have best-estimate (reference) data sets as input. These data sets are heavily interpreted aggregations of experimental data. As in the experimental information systems regime, data management is directed by the end product; however, effective record cataloging is provided by JOSHUA, at least for simple grids.

The systems involved in developing and distributing reference data sets provide the link between massive, raw-data sets and the highly aggregated, best-estimate, data sets used by models. At SRL, this type of application has not been well supported and, consequently, examples of this activity involve *ad hoc* use of various programming languages, e.g., MARK-IV, SASS, CSMP, and FORTRAN.

A second classification scheme will be used to relate data management facilities to application types (Figure 2).

In translating Figure 1 into Figure 2, interpretation was replaced with retrieval index structure, and aggregation was replaced with inquiry structure. Generally a strong correlation exists between these concepts; that is, the more that data have been subjected to the evaluative and interpretive process, the more, in general, these data have been subjected to correlative and classification schemes which produce names, indices, and structures which organize the data into forms suitable for retrieval

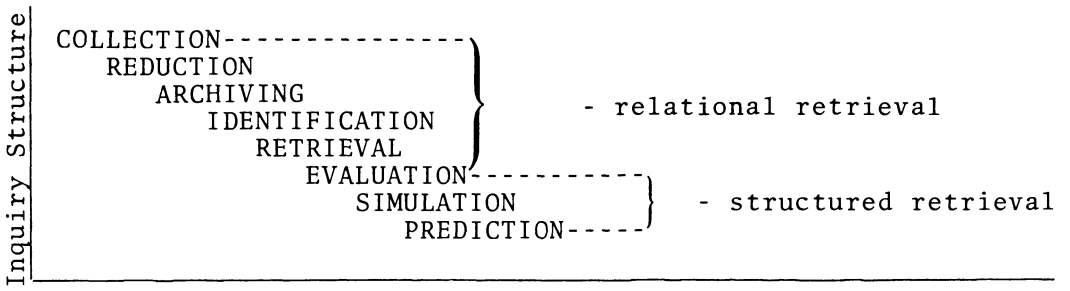


Figure 2. Informational Retrieval Types

(at least for the purposes for which it was organized in the first place). Again, as data become more highly aggregated, the original diffuse nature of the inquiries to the data base become more structured along the lines of the aggregation processes themselves.

The more primitive relational inquiries are therefore of a relatively unstructured type (precise nature of the inquiry not known in advance), being based at inquiry time on the relationships between the (typically) disaggregated data themselves.

The structured type of retrieval is characterized by reliance on the data-group name and its internally associated data elements to respond to the inquiry. However, raw observational data are seldom so structured, but rather are stored as sequential data sets of records containing all variables for an individual observation, necessitating a search through many records to elicit all occurrences of the desired relationship. For this reason, JOSHUA data management does not provide any systematic facility for unstructured inquiry, and the aforementioned ETD applications falling in the experiment and evaluation categories are implemented in an *ad hoc* way using a variety of programming languages which address this type of inquiry with more or less sophistication. Clearly, if JOSHUA supported relational inquiry, there would be less incentive to go to other languages, and data captured for these purposes would then be more readily available for assessment modeling.

A third distinction between application and information retrieval types can readily be made on the basis of identification of geographical data, as shown in Figure 3.

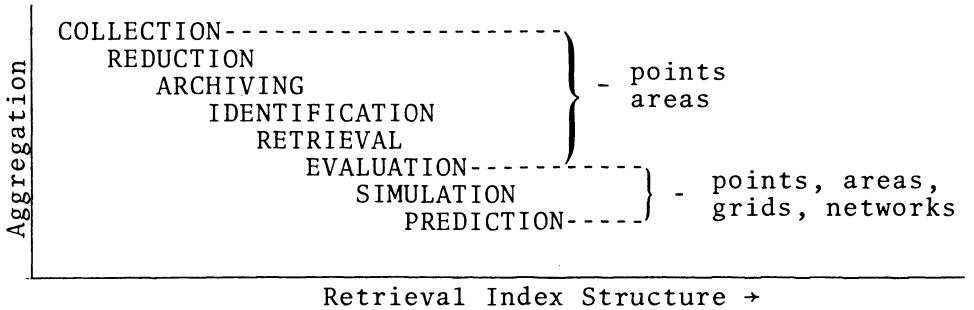


Figure 3. Geographical Coding Structures

Here, the association of the experiment and evaluation systems with data recorded at points (normally latitude and longitude) and for areas (specially defined tracts such as census districts, or convenient political boundaries such as counties) is an obvious one. For evaluation and assessment, some data are most compactly stored as the measured characteristic of some derived area as, for example, soil type or geologic formation boundaries. Finally, the assessment and analytical application systems normally require some data to be interpolated onto a regular grid or network for further modeling as, for example, in transport and diffusion models of the atmosphere or surface water systems.

Even within systems directly oriented to the manipulation, analysis, and display (mapping) of geographical data, a need exists to treat these various geographical indexing schemes as a proper data management function, because of the need of the geographical package to perform basis file mode conversions internally for the sake of efficiency of various algorithms, and to achieve comparability of various related data sets.

NEW JOSHUA DATA MANAGEMENT FACILITIES

The data manager in JOSHUA is called the basic named access method (BNAM). BNAM has been running successfully for nearly nine years, and has well-served the nuclear reactor numerical data handling needs at the Savannah River Plant. In view of environmental science and analysis requirements, and the desire to use JOSHUA on non-IBM computers, a new version of BNAM is needed, and is now being designed.

JOSHUA can be used on most large IBM 360 or IBM 370 class computers. JOSHUA has been installed on an IBM 360/75 at Idaho Nuclear Engineering Laboratory (INEL) and on an IBM 360/91 at Oak Ridge National Laboratory (ORNL). The system was installed at ORNL to support a fast reactor safety data base called SACRD very successfully.[11]

BNAM provides the facilities for managing record names and storage locations. The contents (data elements) of a data record are not known to BNAM. In the eight years since BNAM was developed, many advances have been made in data manager technology, and many of these advances could be incorporated into a new BNAM.

The extended relational capabilities required to meet the needs of the environmental sciences can be accommodated under this newer technology, as well as the inter-ERDA laboratory transport-ability requirement. The most economical approach, therefore, appears to be the development of a new JOSHUA generalized data management system (GDMS) which is upward-compatible with the current version.

One component of a GDMS is used to manage the names of data records. This manager is called the JOSHUA System Name (JSN) manager. Another component is used to manage physical disk storage space. This manager is called the JOSHUA System Storage (JSS) manager. A third component is used to manage the establishment of relationships (owner/member sets) between data records. This latter component is called the JOSHUA System Relationship (JSR) manager. The implementation of the GDMS using these conceptual managers can be visualized in Figure 4.

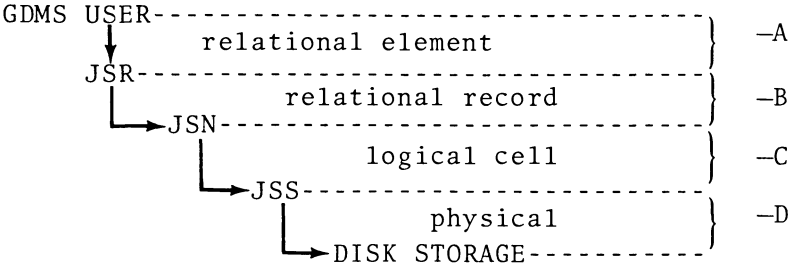


Figure 4. GDMS Components and Interfaces

Information is translated several times between disk storage and the GDMS user. A translation maps one view of the data onto another view. Four views are illustrated in Figure 4, *viz*:

A – Relational at the element level

B – Relational at the record level

C – Logical at the cell level

D – Physical at the page level

The GDMS user should have access to to system with any or all of these views. BNAM today roughly corresponds to the combination of name and storage managers (JSN and JSS). Thus the JOSHUA user has access only to Views B and D in Figure 4.

The relational view (at the record level) has five components; i.e., application data set, data record, data name, record alias, and relation.

Application data sets are the logical subdivision of the data base. They correspond to JOSHUA job, user, and standard data sets and are named with a single qualifier. These data sets may be specified in a hierarchical sequence, so that each data set in the sequence is viewed as a modification of the next higher data set in the sequence.

Data records are contiguous collections of data elements. However, the data elements are not seen in this view. The data record is simply viewed as an arbitrary (but specified) length byte string.

Each data record has a single unique record name. The name consists of up to 16 alphanumeric qualifiers. The record names form a hierarchy with the qualifiers as the nodes. The hierarchy may have several root nodes, that is, it may be segmented into several application data sets. A data record may be associated with any node of the hierarchy, but any node need not have an associated data record. If a terminal node does not have an associated record, the data are, in effect, the node qualifier.

A record may be given one or more alternate names. Each is called an alias. A name in the hierarchy must be either a record name or an alias, but not both. A record name may have several aliases, but an alias must be specific for a single record name.

Aliases establish relationships between data records supplementing those inherent in the hierarchical record names. These alias relationships are managed directly by the GDMS user for such applications as

reordering the record name hierarchy to establish a new search sequence, sharing the same data elements between different collections, and maintaining inverted lists.

The relational view at the element level managed by JSR provides system-maintained linkages between record names and/or alias names based upon specified set relations on the data element values. These relationships establish a network of pointers which can be variously interpreted, and include a full directed graph, owner/member chains, and inverted lists. These linkages are established and maintained by JSR dynamically as data records are being catalogued by JSN.

The current version of JOSHUA provides only the views discussed above as application data set and record name. The additional views are, of course, generally useful GDMS concepts; but they are particularly useful in meeting the needs of geographical data handling, which requires maintenance of topological networks and directed graphs, and the association of thematic data with the geographical-basis reference data sets. With the facilities provided by JSR, much of the algorithmic specification of geographical data relationships will be automatically provided by the system, and need not be hard-coded into applications packages.

REFERENCES

1. Proceedings of the Conference on Computer Support of Environmental Science and Analysis, 9-11 July 1975, Albuquerque, New Mexico, USERDA Document CONF-750706.

2. T. V. Crawford and C. E. Bailey, "Environmental Science and Computations--A Modular Data-Based Systems Approach," in USERDA Document CONF-750706.

3. H. C. Honeck, "The JOSHUA System," USERDA Report DP-1380, E. I. du Pont de Nemours & Company, Savannah River Laboratory, Aiken, South Carolina (1975).

4. P. R. Coleman, R. C. Durfee, and R. G. Edwards, "Application of a Hierarchical Polygonal Structure in Spatial Analysis and Cartographic Display," presented at the Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, 16-21 October 1977, Cambridge, Massachusetts.

5. R. G. Edwards, R. C. Durfee, and P. R. Coleman, "Definition of a Hierarchical Polygonal Data Structure and the Associated Conversion of a Geographic Base File from Boundary Segment Format," presented at the Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, 16-21 October 1977, Cambridge, Massachusetts.

6. A. A. Brooks, "Draft Proposal: American National Standard Specifications for an Information Interchange Data Descriptive File," X3L5/589F (corrected), available from the author, Post Office Box X, Oak Ridge, Tennessee 37830.

7. D. Merrill and D. Austin, "ERDA Interlaboratory Working Group for Data Exchange (IWGDE)," USERDA Report LBL-5329 (1976).

8. J. B. Fried, "BASIS On-Line Retrieval and Analysis of Large Numeric Data Bases," in Proceedings of the Fifth Biennial International CODATA Conference on the Generation, Compilation, Evaluation, and Dissemination of Data for Science and Technology, July 1976, Boulder, Colorado, Pergamon Press, New York (1976).

9. Session VI-C "Data Access and Manipulation Languages" in Proceedings of the Berkeley Workshop on Data Management and Computer Networks, 25-26 May 1976, Berkeley, California, USERDA Report LBL-5315 (1976).

10. B. C. Patten, et al., "Total Ecosystem Model for a Cove in Lake Texoma," in "Systems Analysis and Simulation in Ecology," Vol. 3, B. C. Patten, ed., Academic Press, New York (1975).

11. N. M. Greene, G. F. Flanagan, and H. Alter, "Central Computerized Data Base for LMFBR Safety Codes," in Proceedings of the Fifth Biennial International CODATA Conference on the Generation, Compilation, Evaluation, and Dissemination of Data for Science and Technology, July 1976, Boulder, Colorado, Pergamon Press, New York (1976).

THE MANIPULATION OF SCIENTIFIC DATA FOR NUCLEAR ENERGY CALCULATIONS

K R Montgomery
United Kingdom Atomic Energy Authority
England

THE MANIPULATION OF SCIENTIFIC DATA FOR NUCLEAR ENERGY CALCULATIONS

INTRODUCTION

1. The UKAEA are currently operating a prototype fast reactor (PFR) at Dounreay, North Scotland. When the reactor is operating at full power, fuel burn-up rates in the inner core will be of the order of $1\frac{1}{2}\%$ per reactor cycle of 6 weeks. The reactor contains 78 core region fuelled sub-assemblies (325 fuel pins per standard sub-assembly) and 42 breeder sub-assemblies (85 breeder fuel pins per sub-assembly) in the breeder zone. There are some 20 non-standard sub-assemblies in the core region.

2. Fuel management strategy is vital to the efficient operation of this reactor, the strategy falling into three broad groups of planning, operational and post-operational activities. In brief, the principle objectives of each stage are as follows:-

(i) Planning. To provide sets of workable and economic forward core loadings allowing fuel enrichment to be defined for replacement sub-assembly orders. To provide advanced data of predicted sub-assembly performances in respect of reactivity worths, heat ratings and maximum temperature requirements (for pre-gagging the sub-assembly coolant flow).

(ii) Operational. To provide on demand fuel burn-up rates, power distributions, control rods worths, breeding gain, neutron balances and off-centre reaction rates.

(iii) Post Operational. To provide accurate reaction data for fuel element performance analysis, and to generate a permanent data store of fluxes, spectra and neutron group cross-sections in the PFR fuel data bank.

3. The above is merely a brief outline of the objectives of the fuel management calculational requirements. Although there are many other aspects of calculational processes in the nuclear energy field that require the processing of large data sets, this paper will be restricted to an outline of the fuel management calculations as being typical of the processes involved. The unusual nature of the associated problems, the ideal requirements of the system to be adopted, and the present technique for their solution will be presented in outline form. Some of the essential requirements of a scientific data base to support such calculations will be defined.

PROGRAMMING OBJECTIVES

4. Tasks in nuclear fuel management have three distinct characteristics, namely

(i) large amounts of data require computerised manipulation and storage

(ii) the computer programs that perform the calculations involve the iterative execution of several programs, each requiring data generated by the others in addition to user-input data.

(iii) program coding is necessarily divided between dispersed groups of programmers in different fields of expertise.

5. These characteristics give rise to two well-defined requirements of a system that will operate successfully under the above constraints, namely:

(i) a data storage and retrieval function possessing considerable data independance, controlled by a management suite and permitting volatile and non-volatile files.

(ii) inter-code compatibility that will permit dispersed programmers to "plug-in" modules coded by other programmers to perform a particular set of the total algorithm.

COMPATIBLE OPEN SHOP MODULAR OPERATING SYSTEM (COSMOS)

6. Many of the PFR nuclear energy calculations are performed within the COSMOS system, which is a general system of FORTRAN 4 coding handling a scientific database and a compatible code scheme and is devised to operate within the above constraints. For a detailed technical description of the system the paper by Brissenden[1] should be consulted; the scope of the present paper is limited to a brief outline of some of the features of the COSMOS system.

7. COSMOS has been in use since 1971 and its supporting database now contains some 300 Mbytes of information. It is a modular code scheme under which sets of programs designed to use common datafiles and programming conventions operate. Data handling is independent of the processing algorithms and communication facilities are provided between programs. Standard datafile accession routines for read/write options upon well ordered datafiles are built into the programs. COSMOS is integral with the computer system control in that the embedded accession sub-routines will locate and transfer datafiles from the database to programmer specified core storage. A program that performs a single part of a multi-function task is regarded as a 'module' of the program suite. Data generated by modules may be written to the database for subsequent modification within the total program execution or reserved for use by other programs at a much later date. A single database holds all the common data required by extant COSMOS modules.

8. COSMOS is installed on the ICL 4-70 computer at the Atomic Energy Establishment at Winfrith, Dorset, England. It is coupled by data links to Risley (Cheshire, England) and the Prototype Fast Reactor (Dounreay, North Scotland). Modules are available for a range of calculations including Thermal Performance, Irradiation Swelling, Shielding and Fuel Management.

THE COSMOS DATABASE

9. The datafiles are stored on EDS60 (Exchangeable disc system, 60 Mbytes) discs which are called the COSMOS Interface. The Interface is subdivided into many small libraries that give the user some independence in the storage of his own data. On average, each user tends to operate within about four libraries. The datafiles are generalised Fortran arrays of numeric data and the Nests are integer strings which describe the datablock structures and their parametric relationships. The datafiles are identified by Labels which normally have five components.

DATABLOCKS AND NESTS

10. The common understanding between all users of the definition of each datablock structure and the ability of all programs to interpret it guarantees the compatibility of each individually written program. This definition is given in the Nest linked to the datablock. To clarify the point, an example is given of a possible and conveniently simple datablock and its Nest, but it is not necessary for the user to be aware of each datablock/Nest definition since interpretive modules, such as NLOOK, are provided that unpick the datablock structure from the Nest for the user.

11. In several reactor physics codes the composition of the reactor is input in terms of the densities of various nuclides in various regions. A table of COMPOSITION showing the densities of 8 nuclides in 3 regions might be as follows:-

Nuclide -	1	2	3	4	5	6	7	8
Region								
1	.076	.134	-	-	-	-	-	-
2	-	-	.244	.101	.176	.040	-	-
3	-	-	-	-	-	-	.204	.011

The linear datablock could be written:

.076,.134,0,0,0,0,0,0,0,0,.244,.101,.176,.040,0,0,0,0,0,0,0,0,.204,.011

and structured by the integer string 3,8 (3 regions, 8 nuclides). Writing the COMPOSITION table in columnar form as follows:

Nuclide	Region	Density
1	1	.076
2	1	.134
3	2	.244
4	2	.101
5	2	.176
6	2	.040
7	3	.204
8	3	.011

Here, the column of data 'Density', is the datablock and the relationship columns 'Nuclide' and 'Region' is the Nest. The Nest descriptor could be 3,8,11222233 for the linear string datablock. Noting further that successive nuclides are always in the same or next higher region, the datablock relationships could be expressed by a Nest 3,8,242.

12. The numerical representation of all likely conventions have been formalised in COSMOS. The Nest integers always appear in the pattern defined by the convention and include the convention code number. The integers are never mixed with the datablock itself. Nest interpretation routines are bound into every COSMOS program. The names of the connected entitles gives the Nest Display name, eg (REGION(NUCLIDE)) and the input to the program would state that the datablock COMPOSITION is described and structured by the Nest whose display Name is (REGION(NUCLIDE).

LABELS

13. The datafiles are held on close packed variable length datablocks and catalogued by their Labels, and each dataset catalogue is a datafile in the COSMOS directory handled by accession routines. The directory consists of an index of Datafile names linked to an accession number, and physical address pointers also linked to accession numbers. The 5 dimensions of the Label are:

(i) Display Name: the datafile descriptor and up to 190 characters for the Nest components.
(ii) Version Number: a variable to augment the Display Name.
(iii) Program Name: to enter the datafile to the database.
(iv) Run Number: to record the individual computer runs of the same program.
(v) Library Number: to identify the area of the database holding the datafile.

The parameters are punched on the label cards in the above sequence (with '*' as the default character). The labels are specified by the user for all data, whether input from source, from the Interface, or output to the Interface.

14. This brief outline of COSMOS has necessarily omitted several of the outstanding features of the system, such as Tracing Facilities, Database Integrity, Sequential Files and Management Control. Reference must however be made to the COSMOS FORTRAN IV program WORKSHOP that is used to service the database as its standard utility. The program can amend and store "decks" of cards on COSMOS files, and submit these "decks" to the job stream for execution.

15. In nuclear physics calculations there are many examples of tasks that involve running a sequence of jobs using a number if distinct programs,that are linked by later jobs in the sequence. If the overall task can be programmed for consecutive operation without user intervention the risk of input error is reduced. The workshop program 'EASY' can perform this function by issuing directives to the workshop to cause control to pass to a set of Fortran compiled statements.

WORKSHOP

16. Workshop is a Fortran based program combining the card editing facilities and logic of simple Fortran. It supports a range of commands that handle the COSMOS system and has roll-in/roll-out facilities to enable it to roll-out into the database when not in use. Here it may remain until it resumes further execution at a later time. A workshop program can submit a program for execution, roll itself out into the database and resume execution on the completion of a set task. In this way, a workshop program can proceed intermittently over several days or weeks. It can be used for the automatic submission of reactor physics calculations. It can be used for the management of user resources such as opening new files, archiving old ones, deleting old libraries and renewing magnetic tapes, assigning run times and print-out. It interacts with the computer operators whilst rolled out from the database and can take remedial action in the event of a system crash. It can cycle indefinitely through the computer whilst performing database service tasks.

FUEL MANAGEMENT CALCULATIONS

17. A typical requirement would be to predict the parametric changes consequent upon a change to the reactor core loading. A sequence of calculational processes would be undertaken under the control of the Workshop COSMOS program EASY using modules for the solution of each stage of the process and would input data currently held on the COSMOS Interface in conjunction with the new data associated with the core change. A probable set of steps would be as follows: For the new core (1) source input data for the new core loading, (2) compute mean core burn-up, (3) computer macroscopic power cross-sections, (4) compute neutron flux distribution, (5) compute power distribution; for the burnt-up core (6) compute nuclide composition after a specified time period, (7) compute macroscopic power cross-sections, (8) compute final neutron flux distribution, a and finally (9) display all computed information from the task in reactor zone plan format.

18. A detailed statement of the Fuel Management processes has been published by Wardleworth and Wheeler (2). The following short description is only intended to indicate the interactive nature of the individual modules of the task and the inter-dependence of the Interface datafiles and the module generated datafiles.

19. The compositions of the sub-assemblies are source input from Design Reports that detail the volume fractions of each constituent material for each sub-assembly type.

Their nuclide compositions are amplified through the modules SACOMP and RSSHIELD from the 37 group neutron cross-section library, the compositions of the individual materials and the fissile loading from the PFR Fuels Databank. These interactions create a sub-assembly data store on the Interface of nuclide densities and neutron cross-sections still tied to individual sub-assemblies for each sub-assembly type in the new core. The projected reactor loading is then input to relate new individual sub-assemblies to locations in the reactor. The reactor model is then constructed, using a definition of zone, group and mesh structure and weighting fluxes. The compositions and cross-sections are smeared and condensed to form the macroscopic cross-sections required for the neutron diffusion theory program (TIGAR) which computes the neutron fluxes. Using these computed fluxes, the sub-assemblies from the data store are 'burnt-up' as required and returned to the store which will ultimately contain compositions at each stage of burn-up. It can be seen that many individual programs are involved and the input data at each stage is changing in a dynamic manner. The burden of the task is greatly reduced by the Workshop program.

CONCLUSIONS

20. This paper has attempted to give an account of a practical application of scientific calculation in the nuclear energy field and to demonstrate the necessity of a supporting database structure and a compatible modular code scheme. The COSMOS databank differs from the General Database Management System IDMS in that it is a dynamic area in which any user may operate. Provided that the user conforms to set conventions, he may write datafiles to and read them from the database. By labelling the files and storing the labels in the Directory, these files are available for other users to employ. The files are related in the sense that they have been created within a chosen model and have used particular generations of files that are current on the database. The user name, program name, version number and run number all uniquely identify the file and its points of origin. In effect, all users are their own database administrators, but the files that they have generated conform to the administration rules of the database. The files appear unrelated in that pointer navigational paths between files are not used; the files labels are used through their labels to point to generically related files. It is possibly superior to IDMS in possessing the additional attribute of interaction with the system and the system operators, and offers not only data independence but also program independence. Through its complexity it tends to possess a poor user image which is being improved through the continuing development of the system.

ACKNOWLEDGEMENTS

21. The author is greatly indebted to D Wardleworth for permission to extract certain parts from his unpublished Committee Paper, and to the publications of Wardleworth & Wheeler and R J Brissenden.

REFERENCES

22. (1) "The Compatible Open Shop Operating System COSMOS", R J Brissenden. American Nuclear Society "Topical Meeting on Computational Methods in Nuclear Engineering", Charleston, South Carolina, April 15-17, 1975.

(2) "Reactor Physics Calculational Methods in Support of the Prototype Fast Reactor", D Wardlesworth & R C Wheeler. Journal of the British Nuclear Energy Society. 1974, 13(4) pp 383-390.

GENERALIZED DATA MANAGEMENT SYSTEMS AND SCIENTIFIC INFORMATION

Report of a specialist study

SYSTÈMES DE GESTION DE BASES DE DONNÉES ET INFORMATION SCIENTIFIQUE

Rapport d'étude de spécialistes

Published by/Édité par
OECD NUCLEAR ENERGY AGENCY
AGENCE DE L'OCDE POUR L'ÉNERGIE NUCLÉAIRE
38 bd. Suchet, 75016 Paris France
1978

NEA WORKING GROUP ON NUCLEAR ENERGY INFORMATION
GROUPE DE TRAVAIL DE L'AEN SUR L'INFORMATION
DANS LE DOMAINE DE L'ÉNERGIE NUCLÉAIRE

GENERALIZED DATA MANAGEMENT SYSTEMS AND SCIENTIFIC INFORMATION

SYSTÈMES DE GESTION DE BASES DE DONNÉES ET INFORMATION SCIENTIFIQUE

report of the specialist study on computer software
rapport d'étude de spécialistes sur le logiciel d'ordinateur

The use of Generalized Data Management Systems
for handling Scientific Information

L'utilisation de systèmes de bases de données généralisés pour
le traitement de la documentation et des données scientifiques

jointly organized by/organisé conjointement par
OECD NUCLEAR ENERGY AGENCY
AGENCE DE L'OCDE POUR L'ÉNERGIE NUCLÉAIRE
and/et
UNITED STATES DEPARTMENT OF ENERGY

in cooperation with/en coopération avec
U.S. NATIONAL BUREAU OF STANDARDS

Chairman/Président
A. SHOSHANI, LAWRENCE BERKELEY LABORATORY

Secretary and Editor
N. TUBBS, OECD NUCLEAR ENERGY AGENCY

published by/édité par
OECD NUCLEAR ENERGY AGENCY
AGENCE DE L'OCDE POUR L'ÉNERGIE NUCLÉAIRE
38 bd. Suchet, 75016 Paris France

CONTENTS

TABLE DES MATIERES

PART III

GDMS APPLICATIONS FOR HANDLING SCIENTIFIC DATA

PARTIE III

APPLICATIONS DES SGBD A LA MANIPULATION DE DONNEES SCIENTIFIQUES

A. Experience of GDMS use

Les SGBD à l'usage

B. Forward planning for GDMS use, and potential GDMS applications

Planification pour l'introduction de SGBD, et applications potentielles de SGBD

STATUS OF DATA BASE MANAGEMENT SYSTEMS AT ARGONNE NATIONAL LABORATORY

P. M. Fuja
A. J. Lindeman
Argonne National Laboratory
Argonne, Illinois U.S.A.

ABSTRACT

Argonne National Laboratory has been using the System 2000 data base management system for the past two years. It has been used for technical as well as administrative applications. This paper describes some of the experience gained relating to advantages and disadvantages of data base management systems as well as of System 2000 in particular.

Argonne National Laboratory acquired System 2000 from MRI Systems Corporation in 1975 and since that time it has been used for a wide variety of applications. The purpose of this paper is twofold: (1) to review some of our experiences with System 2000 and (2) to discuss recent DBMS technology advancements.

In the Spring of 1975, several people felt the need for a data base management system and began defining management and system objectives in preparation for obtaining one. Several approaches were considered in obtaining a DBMS. First, where do you get one? Some national laboratories have been fortunate in receiving funding for designing and implementing their own software in the DBM area. ANL felt that we were getting into the DBM area rather late and in this case, a software development was unnecessary since there were commercial systems available. A second approach was to obtain a "public" DBMS from another laboratory or possibly a university. Although this approach was considered, we found that some would not work on our computer, some didn't have the needed features, and others had little or no documentation. Ultimately, we decided to analyze and select from commercially-available systems. This choice offered several advantages and some disadvantages:

> There were a number of DBMS packages available for IBM hardware; therefore, we had a wide choice.
>
> Companies selling systems generally support their products by fixing bugs, providing documentation, and usually improving the package.
>
> A substantive argument often raised against commercial systems is that the source code is usually not available. Therefore, no enhancements suited to the local needs are possible. This can be an advantage, however, in maintaining compatability with other installations having the same package.

After deciding to buy a commercial system, it was necessary to specify desired capabilities and then establish an evaluation methodology. Some of the desired capabilities were:

an English-like query language
accommodate complex data structures
efficient update and retrieval
provide interfaces with high level programming languages
report writer facility

System 2000 was selected in the Summer of 1975 over several other products as the DBMS that was most compatible with ANL requirements.

To give the reader some impressions on what has been accomplished with data base systems at ANL, several applications are briefly described.

1. WENDS

The WENDS system is a hierarchical data base where detailed information on energy R&D on a world-wide basis is stored for easy access by scientists, engineers, and project or program managers. In addition to detailed technical data, material on economic, social, and political conditions for all nations covered in the program will provide additional perspective to users.

This application supports a hierarchical data base where elementary components are units of information called screens. A screen of information is defined as 22 lines of text with 70 characters per line so that it fits within the bounds of most video terminals. The characters displayed within a screen are from the standard ASCII 96 character set which includes both upper and lower case letters. All retrievals and updates for the prototype system are implemented with the System 2000 Natural Language.

WENDS is constructed on three logic systems: 1. A geographic tree, in which specific information about the energy picture in any given nation can be quickly assessed, along with whatever specific data is desired on particular technologies; 2. A technology tree, which will allow users to obtain data on any given technology and the status of projects, regardless of where they are taking place; and 3. Tutorial data and discussions for background. It is anticipated that data will cover all countries affecting or related to the energy situation in the U.S. They will encompass all major energy technologies on an unclassified basis.

2. INSITE

Interactive Nuclear Site data base was developed under contract for the Nuclear Regulator Commission. It contains site parameters on all commercial nuclear power plants, proposed, under construction, or operating in the U.S. The data is to be accessed from state agencies and utilities across the country and has been made available on a commercial time sharing service network. Extensive use was made of the PL/I language rather than the System 2000 Natural Language to load data and to create conversational retrieval programs.

3. RADIATION EXPOSURE SYSTEM

The Occupational Health and Safety Division of ANL is responsible for maintaining current records for those personnel that may be exposed to radiation. To better accommodate these record-keeping responsibilities, a System 2000 data base of pertinent exposure data has been implemented. In fact, the global query facilities and all report writing features that are part of this system are provided by System 2000.

Each employee and visitor who has an assigned film badge or dosimeter is included in the data base. Additional descriptive information, such as payroll number, birth date, service date, and location is also included. All of the monthly exposure details (rover,

beta, gamma, xray, and neutron exposures) are retained for all sections in which the employee or visitor is monitored.

4. SPECIAL MATERIALS

ANL is developing a new Special Materials Information System for processing and analyzing all data on special nuclear materials (SNM) to provide current (daily) records of the quantities, movements, and locations of SNM in ANL custody. This system includes the capability of company book values with measurement values and statistical analysis of the significance of all differences.

The System 2000 data base is organized by physical control area with supporting data consisting of batch definitions, starting balances, transactions, and physical inventories.

5. FERMILAB

The Fermi Laboratory currently utilizes the ANL central computer facility for its administrative data processing and has implemented several data base applications:

Financial Information System
Capital Assets System
Experiment Inventory for the Accelerator
File of Experimentors and Institutions

Some of this data had previously been kept on other management systems but System 2000 gives much more flexibility and provides for easier update than previously realized.

EXPECTED VERSUS ACTUAL USE OF SYSTEM 2000

For the purpose of this paper, it is desirable that we class our data base applications into two categories--administrative and technical or scientific. A simple, broad definition of an administrative data base is an information system which we use to run our organization and support our paperwork. The technical data bases include the data we use to fulfill our contracts or do our research. Obviously, some data cannot be correctly classified into either of these categories. Some contract data bases are really administrative in nature or could be called management data bases.

Most literature and educational material in the data base field has been directed toward administrative applications. The information needs in many administrative functions fit quite naturally into the criteria for which a data base implementation is appropriate. Some of these are: (1) a fairly well defined set of data which may need to be accessed in different ways (if separate files were kept, there would be considerable redundancy), (2) constant updating is needed, (3) protection or privacy of data is needed, and (4) there are defined relationships between data items. It appears that the DBMS concept is a response to the needs of commercial or administrative computer uses. In fact, the administrative users at ANL had a good idea what they needed in a DBMS and the effort required to implement a data base application.

In contrast, technical users have had some difficulty in gaining the proper perspective. However, the DOE Laboratories have been directing their efforts into fields where the data problems are massive, and some new type of data management techniques seems necessary. At ANL, there have been many trial and error attempts to build useful data bases. Many of these attempts have succeeded but some have failed. It is our desire to attempt to classify some of the reasons for success and failure so that others may benefit from our experience.

Differences in technical and administrative data bases occur in the character of the data, the probable applications, and the methods or characteristics of the people implementing or using the data base. We may consider each of these separately.

In System 2000, the available data types are integer, decimal, character, text, date, and money. The last three of these are rarely needed in scientific data but floating point, vector, and matrix representations are needed. Presently, few DBM systems handle these data representations. Another tendency with scientific data is that fewer updates are needed. Often, a set of data will be completely replaced rather than partially updated. A third difference in the data is that privacy is seldom an issue but accuracy is very important.

Applications also tend to be different in administrative and technical or scientific data base systems. Administrative applications usually need a small amount of arithmetic capability and extensive report generating or information retrieval. Technical applications may require extensive calculations, statistics routines or graphical output generation as well as report generation.

The last area of difference is in the people and their ways of thinking. In the research mode, it is difficult to know what applications may be desirable from a data base. This tends to complicate the proper structuring of the data. Researchers tend also to be quite autonomous and the idea of a data base administrator with broad authority is many times unworkable. Lack of centeralized authority, however, implies an increased need for education of users as well as a high level of technical consultation being available.

In general, technical uses of a data base management system may require a system with broader capabilities than are needed for administrative applications. However, there are many technical users who can benefit from the DBM systems now available even with current limitations.

The following list of questions are given for consideration in deciding whether a given application should be done via a DBMS. They are not exhaustive but may uncover some possible difficulties.

1. Is the data going to be used long enough to warrant constructing a data base? Can it be updated or does the entire set of data need to be replaced?

2. Is there a variety of uses for the data? In some cases, a data base may be appropriate for only one application but it is more effective if many applications can use the data. A related question is: Is the application too simple for the amount of data to be loaded?

3. Can an intelligent structure be defined for the data? Is it too simple or too complicated?

4. Is it in a form to be conveniently loaded into the data base? Is a conversion to such a form worthwhile?

5. What other functions are needed for the data (statistics, graphics, extensive calculations), and are the functions available within the DBMS? System 2000 provides several functions, high level language routines can be written, or records can be generated for the graphics or statistics packages through list or report generator facilities. Interactive use of such packages is not practical except through PL/I.

6. How much external calculations are needed? Calculations can certainly be done efficiently with high level languages as everyone recognizes but data base structures do not lend themselves to matrix manipulation, for example.

Several of the advantages and disadvantages contained in Sytem 2000 have been mentioned throughout this discussion. There is always something that one finds a

software package unable to do but this is an extensive system with broad capabilities for doing administrative and technical data base management. It is a particular advantage that System 2000 is available on IBM, CDC, and Univac hardware and that it has been acquired by several DOE Laboratories.

System 2000 is designed to operate only with hierarchical data. In the administrative work at Argonne, it is sometimes more convenient to utilize other data structures, particularly network structures.

Although System 2000 has proved to be inadequate for some needs, it certainly has many features that make it very useful. Data bases are accessible in either a batch or an interactive mode. It has a particularly good natural query language in which the string and function definition features give extensive capabilities. The report writer has limitations since disjoint portions of the hierarchical structure cannot be accessed in the same report. Since System 2000 interfaces to Cobol, Fortran, and PL/1, many reports are written in a higher level language. A capability not available that would be desirable would be to access more than one data base in the natural language. This feature is available in PL/I.

Because of the flexibility of System 2000, the efficiency can vary widely. In large applications, therefore, attention to optimum data structuring and tuning is necessary. The availability of technical consultants with in-depth knowledge of the system is then critical. As mentioned previously, this has been a problem at ANL which has only recently been alleviated.

CURRENT ANL PLANNING ACTIVITIES

The complexity of the administrative structure at ANL gives rise to complex data structures in an overall administrative data base or what might be termed a laboratory-wide data base environment for administrative information systems. System 2000 is not adequate for such an environment because it is limited to hierarchial data structures and because of the limited capacity to access more than one data base. Therefore, ANL has initiated a planning study to plan for and acquire software for the establishment of laboratory-wide data base environment for administrative information systems. ANL has recognized that the ultimate success of such an environment is critically dependent upon long-range planning.

Why is a lab-wide administrative environment needed? Historically, most data bases have been installed to achieve efficient and flexible management and maintenance of data. However, commercial data base management systems have developed to the point that recent improvements in the area of data accessibility are dramatic. Current demands, such as the proposed DOE Uniform Contractor Reporting Guidelines, require effective, flexible, and timely information retrieval abilities.

We see as one of the main goals of this planning effort the determination of the most appropriate DBMS environment for the Laboratory; that is, one that has the optimum balance of accessibility features, efficient organization, and growth potential. One DBMS may be outstanding in data accessibility, another may be more in line with "where things are going," and yet another may be most efficient. The state of the DBMS technologies will be thoroughly studied within the context of ANL requirements. Part of the planning effort, then, is that the information objectives and requirements of key Laboratory administrative areas are to be identified.

There are a number of software and hardware technologies that can be utilized to accomplish these objectives. One alternative is to keep the current file-oriented application systems, but augment them by additional report writer software. This approach appears inadequate, but will be examined on the basis of its low cost and quick implementation.

In addition to this file-oriented approach, there are at least three forms of data base management technologies to consider: centralized, distributed, and back-end. Currently, the centralized DBMS is the most common: all software, including the TP monitor, is on the central computer and data storage is on attached disk drives. A distributed DBMS is where the data base resides on a number of computers and those computers and data base management systems communicate with each other.

The newer back-end DBMS concept involves off-loading the data base management function from the host CPU to a dedicated minicomputer, or "back-end" processor. ANL has only recently been introduced to this approach for practical applications and it may be a potentially effective alternative for both high-volume centralized environments and also distributed processing networks.

The function of the back-end is to provide data base management services on behalf of the host. The motivations generally given for development of back-ends include: higher performance/cost ratios than with a centralized CPU in executing data base-oriented tasks; increased independence between main frames and secondary storage; enhanced data base security, integrity and availability; and more effective sharing of data among multiple hosts. When the technology develops, the extent to which a given back-end system will satisfy those objectives will depend greatly on key design issues (such as word length, address space, instruction set characteristics, and multiprocessing support) and the degree of integration of the back-end hardware and software architectures.

Both MRI Systems Corporation and the Cullinane Corporation have asserted that improved host throughput accounts for the major cost advantage of back-end DBMS. In fact, both Cullinane and MRI are currently developing prototype back-end DBMS technologies. The Cullinane design currently employs an IBM 370/158 as the host with a DEC PDP 11/70 minicomputer serving as the back-end processor. MRI Systems Corporation is developing the back-end DBMS technology using an Interdata 832 minicomputer. Both organizations claim that these products will be available in approximately one year.

The current state-of-the-art for DBMS technologies reminds the authors of the stories of the state-of-the-art of constructing bridges at the beginning of the century. At that time, engineers estimated the load that the structure must bear, tried to make sure that enough steel and structural support would be provided to hold the load--and hoped. Not until fairly recently have we been able to design structures to withstand wind, water, and earthquakes. It seems that we are building our data bases now much like we built our bridges at the turn of the century. We are now evolving to a better design methodology.

THE PARTICLE DATA GROUP: USING A GDMS TO SOLVE DATA HANDLING PROBLEMS IN PARTICLE PHYSICS

Paul R. Stevens*

Physics Department

California Institute of Technology, Pasadena, California 91125

and

Alan Rittenberg

Particle Data Group**

University of California

Lawrence Berkeley Laboratory, Berkeley, California 94720

ABSTRACT

We examine data handling needs and problems in particle physics and look at three different approaches at resolving these problems with emphasis on the efforts of the Particle Data Group and on the role of GDMS in the solutions.

I. INTRODUCTION

In 1977, worldwide, there are approximately 7000 high energy, or particle, physicists (plus graduate students). Particle physics annual research budgets total to about $600,000,000 and some 300 experiments are under way at 16 major accelerator centers. Approximately 150 of these experiments are completed each year and the experimental data they produce are reported in approximately 1,500 preprints, reports, journal articles or theses. The data fall into two general categories: first, data on properties of elementary particles and resonances and, second, data on reactions.

In the area of particle properties, for the past 20 years the Particle Data Group, PDG, has, by itself, been able to handle all of the data and to satisfy the needs of all users world-wide.

In the area of particle reactions, PDG has been able to compile only a rather limited subset of the data. Although other groups have also compiled reaction data,

---

*Supported by U. S. Energy Research and Development Administration under Contract EY76-C-03-0068.

**The Berkeley Particle Data Center is jointly supported by the U. S. Energy Research and Development Administration, the Office of Standard Reference Data of the National Bureau of Standards, and the National Science Foundation.

notably the CERN-HERA group in Geneva (see Dr. Moorhead's contribution to this study) and, more recently, groups cooperating with ZAED (see the talk of Drs. Bau and Behrens), the situation at present is not satisfying: in many areas no compilations exist at all, in others the compilations are not up-to-date, and in many cases the compilations are difficult either to obtain or to use.

PDG, CERN-HERA, and ZAED are each currently trying to improve the situation but are using very different approaches. The availability, or lack thereof, of a suitable generalized database management system, GDMS, has influenced the approach each group has taken and the range of problems it is trying to resolve.

In the rest of this paper we look at these various approaches, especially that of PDG, and pay particular note of the role of GDMS. We also argue that scientific GDMS be able to handle multidimensional arrays and give an example of how PDG has implemented and is using such a capability.

II. THE NATURE OF THE DATA AND THE NEEDS OF USERS

Particle properties data and reaction data have very different characteristics and are used very differently.

Particle properties (masses, lifetimes, decays, spins, and so on) play an important role in all areas of particle physics, and in other areas, such as nuclear physics and astrophysics, as well. Each year about 1000 new results are reported in approximately 300 papers. Most particle physicists are only interested in averages and few of the widely used ones change in a given year.

PDG has developed a system for handling the particle properties data which consists of a great deal of specialist attention and a simple card-oriented, sequential file manipulated by a number of special-purpose computer programs. Biannually, annually, or biennially, depending on the state of the field, PDG publishes the data and printed averages and distributes them to most particle physicists (1). Because of the stability of the averages and the relatively small volume of data, this simple system has been able to serve both users and compilers very successfully for 20 years (2). This system does not constitute a GDMS. On the basis of this operation alone, PDG could not justify conversion to a GDMS.

Particle reaction data are characterized by a very large number of possible measurements because the number of distinct reactions is very large, collisions between any two particles can result in the production of particles, and because the cross section for a given reaction can be a function of many variables, the number depending on what particles are scattered and on what particles are produced. Also, the description of reaction data is very complex, first, because there is no standardization in the choice of variables and units and, second, because combinations of cross sections may be measured and reported (i.e. ratios, products, sums, differences involving the same reaction or different reactions). These complications are inherent in the nature of particle physics because the field is highly research oriented and constantly changing. Different choices of the independent variables may be preferred in different theoretical models and the preferred choice may change as understanding of the data changes.

As experimental techniques improve, not only do data rates keep growing but data on more and more different reactions are measured, and for specific reactions cross sections are measured for larger ranges of the independent variables and with finer steps. Most new measurements reported in a given year are either entirely new, completely supersede existing measurements, or cover different ranges of the independent variables.

The number of potential users for reaction data is smaller than for particle properties data but their needs are more demanding and varied. Some are interested in comparing many reactions at the same values of the independent variables; others in

studying a few reactions as a function of all the independent variables. For some, a few sample points are sufficient; for others all existing data are necessary. Some require data many times a year; others seldom. In almost all cases, users want the data presented to them in units and variables that they select, regardless of how the data were reported.

III. EARLY COMPILATION EFFORTS

Early reaction-data compilations fell far short of being able to handle all data and of servicing all user needs. In 1969 PDG and the CERN-HERA group in Geneva began to publish selected subsets of reaction data. Since the late 1960's numerous other groups have also compiled subsets of reaction data but have done so primarily for their own research needs and few have published or distributed their compilations. The problems with these efforts were: they did not cover the field; each was done in its own format and each was handled by separate special-purpose programs; those published as printed reports quickly became out of date; those available on tape often required the user to write his or her own data extraction programs.

In 1975 the CERN-HERA group upgraded its system by adopting a relatively simple GDMS available at CERN called TABLOID. By limiting the scope of its compilations to reaction cross sections with all variables integrated out, CERN-HERA has been able to keep its compilations up to date. By using TABLOID it has allowed users direct access to the data and greatly facilitated the tasks of update, sorting, and report generation. The CERN-HERA effort is described in Dr. Moorhead's contribution to this study.

In 1974 the Federal Republic of Germany established the Zentralstelle fur Atomkernenergie - Dokumentation (ZAED) whose mission, in the area of particle physics, was to coordinate compilation of reaction data and to encourage and facilitate the publication of printed compilations. ZAED helps compilers prepare their work for publication and publishes and distributes the printed reports (see the talk of Drs. Bau and Behrens). As a result some compilations have been published which might otherwise have remained private and difficult to access. So far, however, ZAED has made no effort at completeness and, though its compilations are accessible, for many purposes they are still not easy to use since ZAED relies on the format and programs of the different compilers which vary from compilation to compilation.

IV. THE PDG SOLUTION

PDG has taken the point of view that most reaction data should be compiled and that most user needs should be met (clearly some data are of too limited interest to warrant compilation and some requests are too rare to justify setting-up general procedures or too difficult to service economically).

In 1971 PDG realized that the proliferation of many independent, special-purpose compilations would not satisfy user needs in the long run and so began a comprehensive review of its own operations and the data compilation needs of particle physics in general. As a result of this study, PDG decided to build a single system which could handle all data and data-related bibliographic information for particle physics.

In 1973 PDG decided that, to maintain the various databases of this unified system, it needed a GDMS. Furthermore, it decided to design and implement its own. The GDMS which resulted, the Berkeley Database Management System (BDMS), is described by Dr. Richards in another paper submitted to this study.

In 1975 a prototype of BDMS became operational and PDG began implementing its new BDMS-centered system of databases and operating procedures. At the same time PDG turned over primary development responsibility for BDMS to the Computer Sciences and Applied Mathematics Group of the Lawrence Berkeley Laboratory both because BDMS promised to be useful to other laboratory groups as a general scientific database management system and because its development was turning out to require greater resources than PDG had available.

The development of BDMS and of the BDMS-centered system of databases and operating procedures have been intricately related. Initially the needs of PDG and the users dictated features that BDMS should have. As BDMS became a more and more general system, features were added which could be of potential use to PDG and any scientific data compilation effort. Armed with greater capabilities in BDMS, PDG soon found ways to refine and extend its system which, in turn, led to the discovery of new features BDMS should have. Both BDMS and PDG's system have benefited immensely from this interplay, but an unfortunate consequence has been that both systems have taken much longer to develop than anticipated. Even now, in 1977, BDMS is still far from fully developed and PDG's BDMS-centered system is at least a year away from being fully operational. However, a version of BDMS is operational and has many powerful features; all major components of PDG's system are designed and are in various stages of being implemented.

PDG's new unified system contains three major databases: the document file, the reaction-data file, and the particle-properties file. Each file has its own hierarchical intra-record structure and its own encoding language. Each language has some data elements which must be encoded in a controlled vocabulary, for example particle names, and some which must be written in a rigidly-defined syntax, for example reaction names. To a degree the files are interrelated in that some data elements appear in all three, for example particle names, and no matter where they appear they must be encoded in the same way. The controlled vocabularies and some syntax definitions are kept in auxiliary databases.

BDMS handles general database management operations for all databases: retrieval, update, storing, etc. Features of BDMS of special significance to PDG's application are: random access and update of individual records; batch and interactive capabilities with common command language; extensive retrieval facilities; hierarchical intra-record structure; support of character bit strings, integer or real (single or double precision) vectors; exits to user-supplied routines; and modular design and, for the most part, machine-independent FORTRAN IV coding (see Dr. Richards' talk for more details).

In addition PDG has interfaced to BDMS extensive special-purpose software for input data verification, syntax checking, controlled vocabulary checking, data transformation, and special-purpose report generating. For the document-file, the simplest of the major files, the data elements are still sufficiently complex that the amount of special-purpose software is comparable to that of BDMS itself. For the more complex reaction-data file which is still under development, the relative amount of special-purpose software is expected to be much greater.

Most anticipated user queries can be handled by the BDMS retrieval features coupled with special PDG-supplied extensions. The BDMS query language (3) includes Boolean and relational operators, nested parentheses in search expressions, truncated and range search. Any data element value or quantity derived from it may be defined as a key. PDG-supplied software transforms some data element values before the keys are constructed, for example to ensure uniform units for the keys. PDG is also writing software to produce output in special formats most suitable to a given user's needs or to interface to graphics.

The mere existence of PDG's system which is capable of handling all particle physics data is not sufficient to ensure that all data will be compiled. In fact, the volume of old uncompiled data, the production rate of new data and the amount of expertise necessary to compile all types of data are just too high for PDG to hope to do all steady-state encoding and catch up on the backlog by itself. With its unified encoding language, the powerful capabilities of the BDMS-plus-PDG software, and the transportability of the system, PDG has begun to recruit collaborators. The most likely candidates are the same groups that have produced or are producing the special-purpose compilations.

The document file plays a central role in PDG's effort to coordinate the efforts of the various groups which actually compile the data. It contains bibliographic information and experimental descriptions for all papers reporting new experimental data

and keeps track of the encoding status of all data. This database is now operational and is being kept up to date. Users can alert themselves to the existence of newly published or preprinted data as soon as they become available and can simultaneously determine whether they are encoded, in process, or not to be encoded. The Crystalographic Data Centre in Cambridge also uses a document database to coordinate the activities of its center, see Kennard et al. (4).

At present, initial work on the reaction-data file is being carried out by a collaboration whose members are located in Berkeley and Pasadena, U.S.A.; Rutherford Laboratory and Durham, England; and Glasgow, Scotland. This geographically diverse collaboration is absolutely necessary because no one location has members with the required expertise in all the classes of reaction data being compiled nor with all the knowledge and skills in systems development. The unified encoding language and the single software system implemented on different computers have been crucial in making this collaboration work. So far emphasis has been on developing and refining the encoding language, software system and collaborative procedures, even while a useful and very large, though far from complete, database is being established. During this development phase, close communication between all collaborators is essential. This collaboration could not have gotten off the ground without an initial visit by all members to Berkeley where the main systems development work is being carried out and could not have been sustained without day-to-day communication via the ARPA-Network.

PDG has not yet begun conversion of its particle properties file to BDMS since its pre-BDMS operation is so successful. Given that it now has a GDMS, PDG does plan to convert these operations eventually.

V. A DATA MODEL FOR TABULAR DATA

When it designed BDMS, PDG felt that a sufficiently accurate model of the intrinsic logical structure of the data could be constructed using a hierarchical intra-record structure together with, among other things, data elements which were real vectors. However, the most common way in which data are presented in the literature is in the form of tables with column headings and row labels giving the names and values of the independent variables and the body of the table containing the data points. PDG could find no satisfactory way to model these tables and the associated labels using just the associations implied by a hierarchical structure. Instead, PDG designed a very efficient and accurate representation of tabular data in terms of a multi-dimensional array model. In PDG's implementation the data and labels are still stored and input in the same way as hierarchically associated quantities but special PDG add-on software applies and interprets additional associations which are nonhierarchical.

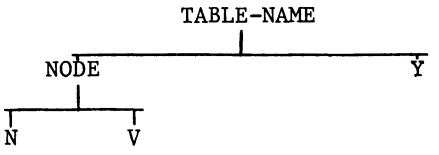
We feel the ability to input and manipulate tabular data conveniently and efficiently should be part of any scientific GDMS and so we briefly outline our particular method.

Consider the following table:

		S,U	s,u			
		W	w_1		w_2	
X1	X2	X3 \ Z	z_1	z_2	z_1	z_2
x_{11}	x_{12}	x_{13}	y_1	y_2	y_3	y_4
x_{21}	x_{22}	x_{23}	y_5	·	·	y_8
x_{31}	x_{32}	x_{33}	y_9	·	·	·
x_{41}	x_{42}	x_{43}	y_{13}	·	·	y_{16}

In this table the capital letters represent the names (and units) of all the independent variables which are needed to characterize the data points. The corresponding lower case letters represent the values of these independent variables. Finally, the y_i's represent the data points (and errors). The association of independent variables and data points is obvious from the tabular representation.

Now, consider the following fragment of a hierarchical record structure:



where N contains the names (and units) of the independent variables, V their values, Y the data points (and errors), and NODE just links related names to values. A possible encoding language is:

TABLE-NAME=TABLE;

NODE; N=Z; V=z_1;z_2;

NODE; N=W; V=w_1;w_2;

NODE; N=S;U; V=s;u;

NODE; N=X1;X2;X3; V=x_{11};x_{12};x_{13};x_{21};...x_{43};

Y=y_1;y_2;y_3;y_4;y_5.....y_{16};

It is very convenient to regard Y as a multidimensional array with the following structure: $\bar{Y}$(n(Z),n(W),n(S,U),n(X1,X2,X3)) where n(Z) is the number of distinct values of the variable Z (2 in the example), n(W) is similar, n(S,U) is the number of distinct pairs of s,u values (1), and finally, n(X1,X2,X3) is the number of distinct (X1,X2,X3)-triplet values (4). Z is the fastest running variable, the (X1,X2,X3)-triplet is the slowest. This array representation of the table makes it very easy to formulate and visualize any data selection or manipulation operation: for example, selecting data points with certain values of the independent variables amounts to projecting out rows, columns, or planes from the table, or equivalently from the array $\bar{Y}$ (the example should be regarded as a 3-dimensional table with the (X1,X2,X3)-axis down, the Z-axis across, and the W-axis into the paper with the right half of the table as the second sheet). This model resembles the relational model and shares its conceptual simplicity.

The convenience of using this model and its efficiency are related to the very precise way in which it models the intrinsic logical structure of the data. Consider a dependent variable y which is a function of n independent variables. Then y is a function in an n-dimensional space and the data points y_i may be regarded as lattice points in that space. Finally, a typical table will be a projection on 2, perhaps 3, dimensions of that n-dimensional lattice.

As an aside, we remark that PDG has experienced great difficulties in converting many non-BDMS files to BDMS. First, PDG had to overcome such tedious, yet straightforward, problems as translating free format quantities into controlled vocabularies, allowing for different encoding standards, correcting errors, removing duplication, and finding omissions. However, a much more difficult task was translating compilations where the data model used was not defined precisely enough and, as a result, there were ambiguities in what the actual associations between elements were. Such

cases were quite common for descriptive information in compilations using sequential card-oriented formats and could only be resolved by looking at the source publications.

VI. CONCLUSIONS

The Particle Data Group has designed and is now implementing a system of databases, encoding languages, database management software, and operating procedures that it hopes will resolve most problems with data compilation and dissemination in particle physics. PDG's operation requires both extensive general database management capabilities as well as numerous capabilities particular to its application. There is no doubt that most of the general capabilities could be used in other scientific data handling operations and that, likewise, most of the general capabilities developed for other applications could be used productively by PDG. Since the development of a GDMS is so very difficult and time-consuming and at the same time so essential to operations such as PDG's, the highest priority should be given to prompt development of a GDMS suitable for most scientific applications and this development should be coordinated at the national, or perhaps even international, level. BDMS and some of the other systems described at this conference have many of the capabilities required of such a system.

An especially important characteristic of any GDMS is how well it can model the intrinsic logical structure of the data being handled. In this regard PDG found it necessary for the GDMS to be able to handle multidimensional arrays. Since such a capability is not yet in BDMS, PDG has implemented a scheme through special add-on software.

ACKNOWLEDGMENTS

PDG's system has been built by the ideas and hard work of many people without whose contributions we would have had nothing to report. The design and implementation of BDMS has been almost entirely the work of David Richards. Those who planned, developed, and tested the software and procedures particular to PDG include B. Armstrong, T. Coffeen, R. Crawford, F. Gault, C. Horne, M. Hutchinson, R. Kelly, T. Lasinski, B. Read, R. Roberts, T. Trippe, F. Uchiyama, V. White, and G. Yost. Finally, a great deal of motivation and inspiration, especially in the early stages of the project, was provided by Geoffrey C. Fox and Arthur Rosenfeld.

REFERENCES

1. Particle Data Group, Rev. Mod. Phys. 48, S1 (1976).
2. A. H. Rosenfeld, Ann. Rev. Nucl. Sci. 25, 555 (1975).
3. BDMS User's Manual, D. R. Richards, LBL-4683 (Revision 1) unpublished.
4. O. Kennard et al., Chemistry in Britain 12, 213 (1975).

USE OF A GDMS FOR HIGH-ENERGY REACTION DATA

W.G. Moorhead
CERN, Geneva, Switzerland

1. ABSTRACT

At CERN, data on high-energy reactions is being compiled using a Generalized Data Management System. The GDMS is a stand-alone system designed for administrative and engineering applications. The Data Base at present contains about 20,000 cross-section values, each linked to a description of

a) the corresponding reaction, and

b) the publication from which the value was derived.

The immediate objective is to produce the widely circulated Compilation Reports, and the standard Report Generator of the GDMS is being used for this. Direct retrieval is also possible.

2. The GDMS

A generalized data base management system called TABLOID was implemented for the CDC 6000 Series Computers under SCOPE 3.4, primarily to meet some of the needs arising in the construction of the 300 GeV accelerator. From an external point of view, TABLOID is a self-contained GDMS by means of which Data Bases of a fairly general type may be defined, and procedures for update, retrieval, sorting and report generation specified, all in a (rather primitive) high level language. Internally, TABLOID interfaces with the CDC SCOPE Indexed Sequential (SIS) file organization module for storage and access of data on disks.

In TABLOID, a Data Base consists essentially of an SIS file containing variable length records composed of items, sub-items, and repeating items to one level only, the organization within a record being defined by a schema. Some of the items are designated in the schema to be the components of the unique Data Base Key used to access the records. The schema and the user-defined procedures written in the TABLOID language are kept in source form in a separate word-addressable file.

In addition, a second SIS file containing any nymber of "dictionaries" may be attached. Each dictionary has its own schema defining its record format and key, in the same way as the main Data Base. The key is generally an abbreviation which may occur many times as an item in the main Data Base, and the purpose of dictionaries is thus to avoid the reading in and storing of excessive amounts of redundant information.

TABLOID is now used for about a dozen Data Bases relevant to the installation of cables, magnets and other equipment in the accelerator tunnels and auxiliary buildings, as well as for other applications.

One of the limitations of TABLOID is that it is used entirely in batch mode and no interactive facilities are envisaged for it, though batch jobs may be submitted via an interactive terminal.

3. THE HIGH-ENERGY REACTION ANALYSIS APPLICATION

TABLOID is being used to produce compilations of cross-sections for certain classes of high-energy reactions. There is a main Data Base file containing at present about 20,000 records where each record represents one data point obtained from a publication. A data point is a cross-section and its errors for a particular reaction, with a specified energy of incident particles, and possibly with specified decay modes of secondary particles. A reaction is of the form

$$I + T \rightarrow A + B + C \ldots \ ,$$

where I is an incident particle; T is a target particle and A, B, C, etc., are secondary particles, e.g.,

$$PI + P \rightarrow P + RHO + PI^{-} + PI^{o}$$

The cross-section may, in fact, correspond only to a channel in which one (or more) of the secondary particles decay into other particles, which may in turn decay, etc. For example, a reaction can take the more general form :

$$\begin{aligned} I + T \rightarrow A + B + C \ldots \\ \hookrightarrow D + E \ldots \\ \hookrightarrow F + G + \ldots \end{aligned}$$

The Data Base key of the main file consists of the following items :

i) an eight character code, such as A 100000, for the reaction;

ii) the energy of the incident particle in MeV/c;

iii) a four character code, such as S102, for the publication reference;

iv) the decay mode and particle number of the first secondary particle listed as decaying.

The records in the main file are thus kept in ascending order of these values, and it is possible to retrieve rapidly by reaction code plus energy range.

There are five dictionaries of which the three most important are :

i) a Reaction Dictionary containing fuller details of about 3500 reactions referred to by the eight character code in the main file. The incoming and outgoing particles are recorded in this dictionary, together with a threshold energy for the reaction;

ii) a Particle Dictionary containing for each particle its mass and for each decay mode the decay products, together with the branching ratio and its error. The particle name which is used as key component in this dictionary has 10 characters, and may have occurred as a secondary in the Reaction Dictionary or as a decay product in the Particle Dictionary itself. There are about 6000 entries in this dictionary at present;

iii) a Reference Dictionary containing expanded (but still coded) descriptions of about 1500 published papers referred to by the four character reference code in the main file. An example of a coded description of a publication is :

GRARD,PL59B,409-75

which means a paper by Grard et al., starting at page 409 of Phys. Letters, Volume 59B (1975).

Procedures for updating the main data file and the dictionaries have been provided for the end user, together with simple retrieval facilities for sampling parts of the files. Pre-processing programs had to be provided to convert the variable field input cards preferred by the user to the fixed field cards required by TABLOID.

An example of part of a page of the compilation report generated by TABLOID is shown in Fig. 1. Much calculation and dictionary searching can be performed in correcting the cross-section and its errors to take account of the decay modes, possibly to several levels. A table of contents is also produced, and output is generated for a post-processor program which makes histograms and graphical plots.

APP

	S	K.ENERGY	PLAB	CROSS SECTION	ERROR +	ERROR -	REFERENCE	FOOT-NOTES
..... REACTION 1 ..... APP = (K*+890/K*-890) (K-/K+)								
	2.149	0.585	1.2000	0.21000	0.02000		BARLOW,NCA50,701-67	
(K*+890/) = KL (PI+/PI-)	2.149	0.585	1.2000	0.07000	0.00700		BARLOW,NCA50,701-67	
		CORRECTED ----		0.07000	0.00700			
(K*+890/) = KS (PI+/PI-)	2.149	0.585	1.2000	0.08500	0.00700		DUBOC,NPB46,429-72	
		CORRECTED ----		0.08500	0.00700			
	2.158	0.604	1.2240	0.06390	0.01510		HANDLER,NPB110,173-76	
	2.169	0.629	1.2560	0.07580	0.01540		HANDLER,NPB110,173-76	
	2.179	0.654	1.2860	0.02260	0.01100		HANDLER,NPB110,173-76	
	2.188	0.676	1.3130	0.05900	0.01310		HANDLER,NPB110,173-76	
	2.201	0.706	1.3500	0.06670	0.01470		HANDLER,NPB110,173-76	
	2.378	1.136	1.8500	0.08900	0.01800		CHAPMAN,NPB42,1-72	Y
THRESHOLD	3.519	0.000	0.0000				9 DATA POINTS LISTED	
..... REACTION 2 ..... APP = (K*+890/K*-890) (K-/K+) PI0								
	2.149	0.585	1.2000	0.24700	0.04000		BARLOW,NCA50,701-67	
(K*+890/) = KS (PI+/PI-)	2.149	0.585	1.2000	0.06520	0.00760		DUBOC,NPB46,429-72	
		CORRECTED ----		0.06520	0.00760			
	2.158	0.604	1.2240	0.06260	0.02930		HANDLER,NPB110,173-76	
	2.169	0.629	1.2560	0.05850	0.02440		HANDLER,NPB110,173-76	
	2.179	0.654	1.2860	0.06230	0.02930		HANDLER,NPB110,173-76	
	2.188	0.676	1.3130	0.06450	0.02350		HANDLER,NPB110,173-76	
	2.201	0.706	1.3500	0.04000	0.02350		HANDLER,NPB110,173-76	
THRESHOLD	5.189	0.889	1.5677				7 DATA POINTS LISTED	
..... REACTION 3 ..... APP = (K*+890/K*-890) (K-/K+) (PI+/PI-)								
	2.158	0.604	1.2240	0.02640	0.01690		HANDLER,NPB110,173-76	
	2.169	0.629	1.2560	0.03010	0.01900		HANDLER,NPB110,173-76	
	2.179	0.654	1.2860	0.03170	0.01960		HANDLER,NPB110,173-76	
	2.188	0.676	1.3130	0.01340	0.01450		HANDLER,NPB110,173-76	
	2.201	0.706	1.3500	0.01700	0.01810		HANDLER,NPB110,173-76	
	2.378	1.136	1.8500	0.38100	0.06100		CHAPMAN,NPB42,1-72	Y
							6 DATA POINTS LISTED	
..... REACTION 4 ..... APP = (K*+890/K*-890) (K-/K+) (2PI+/2PI-)								
(K*+890/) = KS (PI+/PI-)	2.149	0.585	1.2000	0.00950	0.00420		DUBOC,NPB46,429-72	
		CORRECTED ----		0.00950	0.00420			
..... REACTION 5 ..... APP = (K*+890/K*-890) K0 (PI-/PI+)								
	2.149	0.585	1.2000	0.57900	0.08000		BARLOW,NCA50,701-67	
	2.158	0.604	1.2240	0.05140	0.01460		HANDLER,NPB110,173-76	
	2.169	0.629	1.2560	0.14440	0.03510		HANDLER,NPB110,173-76	
	2.179	0.654	1.2860	0.09740	0.05100		HANDLER,NPB110,173-76	
	2.188	0.676	1.3130	0.08710	0.03310		HANDLER,NPB110,173-76	
	2.201	0.706	1.3500	0.07470	0.01890		HANDLER,NPB110,173-76	
THRESHOLD	3.519	0.000	0.0000				6 DATA POINTS LISTED	
..... REACTION 6 ..... APP = (K*+890/K*-890) KS (PI-/PI+)								
(K*+890/) = KS (PI+/PI-)	2.149	0.585	1.2000	0.03390	0.00680		DUBOC,NPB46,429-72	
		CORRECTED ----		0.03390	0.00680			
	2.158	0.604	1.2240	0.02860	0.02850		HANDLER,NPB110,173-76	
	2.169	0.629	1.2560	0.00480	0.01960		HANDLER,NPB110,173-76	
	2.179	0.654	1.2860	0.03750	0.02570		HANDLER,NPB110,173-76	
	2.188	0.676	1.3130	0.01230	0.02030		HANDLER,NPB110,173-76	
	2.201	0.706	1.3500	0.00990	0.02200		HANDLER,NPB110,173-76	
							6 DATA POINTS LISTED	
..... REACTION 7 ..... APP = (K*+890/K*-890) (K*-890/K*+890)								
(K*+890/) = KS (PI+/PI-)	2.149	0.585	1.2000	0.52400	0.00670		DUBOC,NPB46,429-72	
		CORRECTED ----		0.52400	0.00670			
(K*+890/) = KS (PI+/PI-) & (K*-890/) = KS (PI+/PI-)(NC)								
	2.149	0.585	1.2000	0.07030	0.00620		DUBOC,NPB46,429-72	
(K*+890/) = KS (PI+/PI-) & (K*-890/) = KS PI-(NC)								
	2.149	0.585	1.2000	0.05240	0.00670		DUBOC,NPB46,429-72	
..... REACTION 8 ..... APP = (K*+890/K*-890) K*0890 (PI-/PI+)								
(K*+890/) = KS (PI+/PI-) & K*0890 = KS PI0								
	2.149	0.585	1.2000	0.00200	0.00310		DUBOC,NPB46,429-72	
		CORRECTED ----		0.01200	0.01861			
(K*+890/) = KS (PI+/PI-) & K*0890 = (K+/K-) (PI-/PI+)								
	2.149	0.585	1.2000	0.01860	0.00380		DUBOC,NPB46,429-72	
		CORRECTED ----		0.02793	0.00571			
	2.158	0.604	1.2240	0.02000	0.01320		HANDLER,NPB110,173-76	
	2.169	0.629	1.2560	0.02200	0.01580		HANDLER,NPB110,173-76	

NOTES Y=BEAM MOMENTUM IS CENTRAL VALUE. SEE ORIGINAL PAPER

FIG. 1: FRAGMENT OF A COMPILATION REPORT

THE WORLD NUCLEAR POWER PLANT DATA BASE OF THE FRENCH ATOMIC ENERGY COMMISSION

J.C. Leralle - GIDE - DPg - Commissariat à l'Energie Atomique
G.A. Martin - Service APL - CISI

ABSTRACT

The expansion in the construction of nuclear plant for energy production has brought with it a need for data on the characteristics and performance of nuclear power plant, to be used for forward planning and other studies by governments, public and private enterprises in the nuclear field, and financial institutions.

The Data Base described in this paper is an implementation of a GDMS written in APL, and carries technical and economic data on nuclear power installations worldwide.

CONTENTS

1. INTRODUCTION

In 1974, the Economic Information and Documentation Group of the Program Department within the French Atomic Energy Commission (CEA - DPg - GIDE) asked CISI, its EDP subsidiary, to develop a system for handling information about the nuclear power plants of the world.

As a government body, the CEA has to answer various questions about the nuclear market, which may condition political and economic decisions. The increasing number of queries made it necessary to rationalize this activity. Although the data are available, they are often hard to extract from the literature, or simply not published, and it seemed appropriate to store these data on computer media. While the data themselves do not change rapidly, the decision to go Data Base was justified by the wide variety in type and the increasing number of queries.

A first attempt, using a conventional GDMS, proved ineffective, and was followed in 1976 by a feasibility study in order to choose between APL and SYSTEM 2000. While SYSTEM 2000 seemed well adapted to the data structures to be represented, the choice of APL was motivated by the need to allow for unpredicted changes in the system, and for complex computations on the data carried within it.

In June 1977, all basic software for the system was available (data structures, access methods, reports, retrievals, etc....), but the data themselves were still incomplete, because of the need for meticulous checking. Some further delay was due to new demands on the system from the users : new facilities can be implemented rather easily, which incites the users to ask for more.

The present system is a subset of a more general Data Management System LGI [1], written entirely in IBM APL-SV [2] and in which the concepts of the paper 'An APL approach to Data Bases' are fully implemented. LGI is used by CISI for several data bases, in particular Sea Water Desalination Plant (this base will later be linked to the Nuclear Power Plant data), Plutonium Needles (a real-time data base for the Fontenay-aux-Roses production centre), computer systems in the CEA (there are 540 systems with more than 4K of memory) and some small commercial data bases.

2. DATA BASE ORGANISATION

2.1 Data structures

Some criteria will have a single value for each plant (e.g. name, country, etc...) while some others will have multiple values. The first will be represented by vectors of numbers or arrays of characters (e.g. names) where each vector (or array) is a logical record in a file. This set of vectors constitute level 1 of our data base.

There are two types of multiple value criteria : the history of power changes (normally 2 or 3 changes during the life of the plant, with a maximum of 9) and the history of electricity production (one finds one set of data per time interval, normally for each month, though the periodicity may vary between countries. A plant may have a life of 20 years. These two sets of data are normally described as Repeating Groups.

Power changes are few, so that one may represent each (multiple value) record as a matrix, in which a row carries one set of information for all the plants : the first row is the set of the first (initial)

powers, etc... If we associate a relative pointer (number of the row) to this array structure, one may consider the level 2 information as similar to the level 1 information : access will be restricted to a single value of the pointer at any given time and the number of occurrences will be introduced as a direct (level 1) criterion to know whether a plant may be selected or not for a given value.

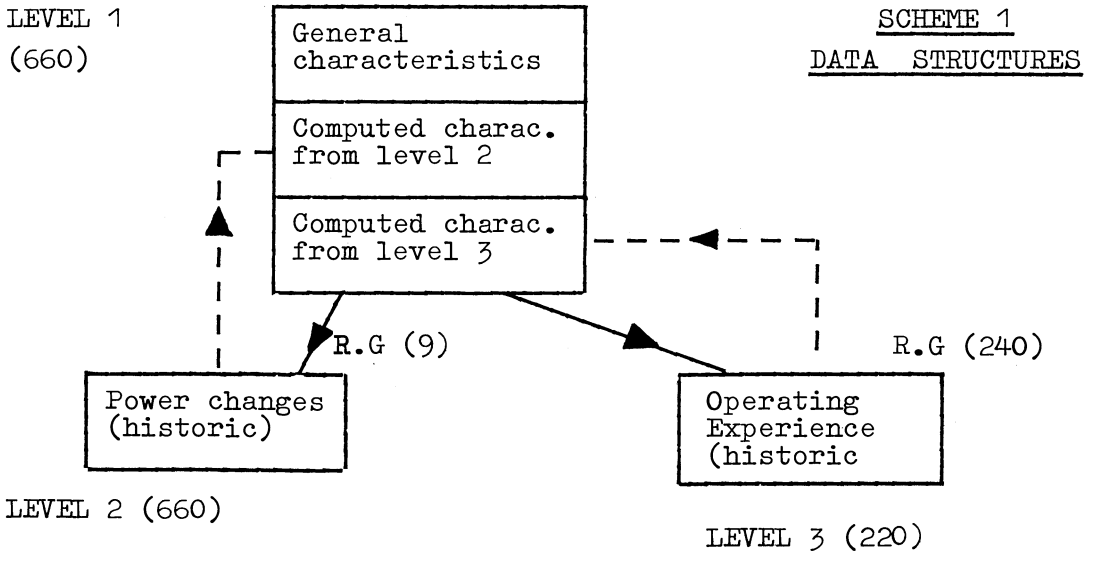
The set of data on the electricity production of a plant is known as the Operating Experience of that plant. It will constitute our level 3 information. A special access method (historic) has been developed to manage this information properly (see [1]). It will be possible to scan the history of the plants from the beginning (e.g. the first 10 years of a set of plants), from the end (e.g. the last 5 years of the life of the plants) or from any random point (e.g. the 2 years after initial criticality or the 2 years before the date of cancelling). To increase the performance, some statistics will be automatically recomputed each time the level 3 data is updated (e.g. the production in the most recent period or for the past 12 months or for the current calendar year). In France, the PEON commission has defined an Energy Utilisation Factor k :

$$k = \frac{100 \times \text{gross electrical generation}}{\text{gross electrical power} \times \text{related period}}$$

For a set of plant units, one may compute the average of the k's or estimate a parameter k in function of each individual period and gross electrical power.

The data structures of level 3 are defined but not yet loaded. The initial loading will be done in September 1977 and will take a few months (there are 220 active generating plants!).

The data structures may be summarized as follows :



National "Reference Plants" (having characteristics defined by the PEON commission) are also recorded, for comparison with a given reactor or class of reactors such as PWR.

2.2 System architecture

The APL functions are organized in 4 workspaces : each workspace may be loaded independently.

2.2.1 Workspace DEFINITION

This workspace is under control of the Data Base Administrator (DBA) and contains functions for :

- file creation and deletion
- initial data base loading (in batch mode)
- updating the various tables
- data base administration (various access tables)
- program library maintenance.

The data structures of the present base were defined directly in APL. A Data Definition Language is under consideration, but the need for one is not immediately evident, since the APL coding of the present structure took only one day's work. It may be preferable to employ the APL support specialists to code data structures rather than construct a DDL which will consume equal coding effort but may result in less efficient data structures.

So as to leave maximum memory in the SELECTION workspace, user-defined functions are stored in an overlay file which is maintained under the control of the DBA, using the DEFINITION workspace.

We may request a list of active keywords and tables (Fig. 1 & 2: the display can be output in either French or English).

2.2.2 Workspace SELECTION

All search operations are run in the SELECTION workspace using the query language described in Section 3. Access to this workspace is in general open only to the DBA : the operations performed are :

- EDITION on high-speed printer or typewriter terminal

- MAJ (mise à jour = update). Currently only the DBA is authorised to update this data base, and updates are performed interactively. Multiple-user updating could be quickly implemented using shared files and a shared variables lock/unlock control mechanism

- BILAN produces balance sheet listings of power plant performance (level 2 data). These may be in standard form (balances by country and type of reactor over a given period) or follow the user's selection criteria

- LISTE for a given keyword prints out all values within the required subset

- VALEURS prints the complete set of values for a keyword.

In addition, system commands are available for routing output within the CISI network, dumping tables from the data base, query optimisation and query storage on the QUESTIONS file.

2.2.3 Workspace DUMP

This workspace contains two main functions :

- dump of the full data base on a line printer (tables and formatted images for each plant)

- selective dump of a set of keywords on punched cards for large updates. Some information is pre-punched on the card (e.g. the internal address) to facilitate the updating operations.

2.2.4 Workspace QUESTIONS

In data bases using the LGI [1] software, this is the normal workspace for external users other than those authorized to use the SELECTION facilities. Any query may be saved from a SELECTION session. The optimized (compiled) search code is kept as well as the initial source for further visualization and use.

For technical and political reasons external users may not at present access directly this nuclear power plant data base. Queries are now handled by the DBA, but client access to the data base is under consideration.

2.3 Technical characteristics of the system

The system uses the standard 80 Kbyte workspaces of the APL-SV Release 2 system [2] installed in CISI ; some 40 Kbytes remain free in a workspace for the user's current operations. Data are recorded on OS files, while APL functions are called for use in overlay from a library file. Some 350 Kbytes of disc space is sufficient to store the characteristics of up to 900 power plants. The present population of the base is 660 plants (with data in levels 1 and 2) of which about 220 are producing electricity (level 3 data). System storage capacity can be extended by reorganizing the disc space allocations. The cost of execution for each search expression in the SELECTION workspace is printed with the reply, and an accounting routine for cost survey and analysis is available in the APL public library.

3. THE QUERY LANGUAGE

The query language used is a subset of that used in LGI, and allows for execution, normally in the SELECTION workspace, of the various standard operations EDITION, LISTE, MAJ, etc. Four types of expression are accepted :

- Logical expressions using arguments connected by ET (and), OU (or), SAUF (and not), PAS (not). These arguments may be arithmetic expressions involving keywords (keyword = xxx etc.) or the results of previous selections.

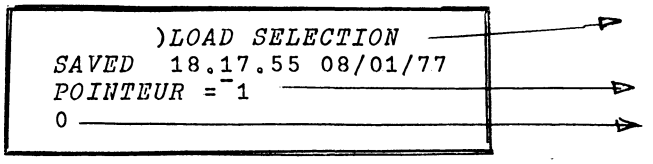
- Computational expressions yielding a value rather than acting as a selection mask.

- Directly executed APL expressions.

- System management commands.

3.1 Typical selection sessions

The conversation below (with user commands outlined) is an example. Line numbering is done by the system.



loading the workspace

relative address for level 2 file opening completed

(actions automatically taken when loading the workspace)

```
SELECTION
```

```
[1] PAYS=US
...TROUVES : 282
COUT :       1.97
```

(number of items answering (the question

the cost is automatically given in French francs)

```
[2] ET FILIERE=PWR
...TROUVES : 164
COUT :       2.53
```

```
[3] ET CONSTRUCTEUR=WEST
...TROUVES : 0
COUT :       2.97
```

these conditions are linked to the previous condition

```
[4] (LIGNE 2)
...TROUVES : 164
COUT :       1.34
```

come back to the selected subset at line 2

```
[5] LISTE CONSTRUCTEUR
...VALEUR :
ALCO
B∧W
C.E.
MARTIN
ORNL
WEST.
```

the result is now a value

set of exhaustive values

the value WEST is unknown, hence the result in /37

```
COUT :       2.02
```

```
[5] ET CONSTRUCTEUR=WEST.
...TROUVES : 90
COUT :       2.86
```

this condition is combined with the last active one, i.e. line /47

```
[6] (LIGNE 2)ET CONSTRUCTEUR=ALCO
...TROUVES : 2
COUT :       3.64
```

activation of a new selection

```
[7] )REFERENCES
 1004 PM2-A-(GROENLAND)     US
 1002 SM-1-A(GREELY)        US
COUT :       3.18
```

system command (like in APL)

```
[7] ∇
```

end of SELECTION

```
COUT DE LA SELECTION :       22.28
```

TOTAL COST

⇒ Control is now returned to APL

User queries are free of the constraints imposed by some information retrieval systems. The results of a selection may be further refined in succeeding expressions /line 2/ or recalled to become the active selection /lines 3 and 4 below/ which may be useful when a selection is empty, as in the question 'LISTE CONSTRUCTEUR' on line 3 below : this represents the intersection of the first 3 search lines.
Selection lines giving an explicit result, indicated by the message ... VALEUR are in fact not normally retained as part of the selection, but may be kept following a system command.

It will prove cheaper to use complex selection expressions, as shown in this search dialogue, than a series of simple conditions (5 francs for 3 conditions combined, rather than 7 francs as in the first example). Typical comparison operators are =, ≠, ∈ (belongs to a list), ⊂(lies between given limits), <, ≤, >, ≥, and this set may easily be extended.

SELECTION examples of complex expressions

```
[1] (PAYS=US)ET(FILIERE=PWR)ET (CONSTRUCTEUR=WEST.)
...TROUVES : 90
COUT :       5.35

[2] (PAYS=US)ET(FILIEREєPWR,PHWR)SAUF(CONSTRUCTEURєWEST.,ORNL)
...TROUVES : 73
COUT :       6.60

[3] LISTE FILIERE
...VALEUR :
PWR

COUT :       1.78

[3] LISTE CONSTRUCTEUR
...VALEUR :
ALCO
B∧W
C.E.
MARTIN

COUT :       1.96
```

these functions are using the last active mask, i.e. line /2/

The SELECTION function may be called under SELECTION to indicate that the last active mask in the workspace has to be used.

```
[3] SELECTION
...TROUVES : 73
COUT :       2.67

[4] ET CONSTRUCTEUR=B∧W
...TROUVES : 33
COUT :       4.76
```

the active mask is the result of the line 2

intersection with a new condition

which is useful for the parameterisation of the questions recorded.

3.2 Listings and Balance Sheets

One may continue with a listing of the selected subset (33 plants) just by entering EDITION :

```
[5] EDITION
LARGEUR :37
MOT_CLEF : CONSTRUCTEUR
LARGEUR :50
MOT_CLEF : SITE
LARGEUR :55
MOT_CLEF : TYPE
LARGEUR :64
MOT_CLEF : COMPAGNIE
LARGEUR :74
MOT_CLEF : ▽
PHASE DE TRI :
MOT_CLEF : SITE
MOT_CLEF : COMPAGNIE
MOT_CLEF : ▽
RUPTURE
MOT_CLEF : SITE
MOT_CLEF : ▽
DECOUPAGE ? : ▽
```

the subset is small enough (33). One may edit a short report.

1. Select the keywords. The system answers with the print width (LARGEUR). Our present report will have a width of 74 characters.

2. Sort phase

3. Changes in the sequence (one blank line will be inserted)

4. To allow automatic page formatting

{ End of data entry symbol. Default options are taken if it is the only answer.

A pause is now made to allow paper positioning, for example before a new page :

N°	NOM	PAYS	FILIERE	CONSTRUCTEUR	SITE	TYPE	COMPAGNIE
1142	ARKANSAS-ONE-1	US	PWR	B∧W	1	COMMERC.	APL
1101	INDIAN-POINT-1	US	PWR	B∧W	1	COMMERC.	CON_ED
1175	MIDLAND-1	US	PWR	B∧W	1	COMMERC.	CPC
1176	MIDLAND-2	US	PWR	B∧W	1	COMMERC.	CPC
1265	GREENWOOD-2	US	PWR	B∧W	1	COMMERC.	DETR.ED
1266	GREENWOOD-3	US	PWR	B∧W	1	COMMERC.	DETR.ED
1127	OCONEE-1	US	PWR	B∧W	1	COMMERC.	DUKE
1128	OCONEE-2	US	PWR	B∧W	1	COMMERC.	DUKE
1143	OCONEE-3	US	PWR	B∧W	1	COMMERC.	DUKE
1136	THREE-MILE-ISLAND-1	US	PWR	B∧W	1	[illegible]	
1158	THREE-MILE-ISLAND-2	US	PWR	B∧W	[illegible]		
1349	ERIE-1	US	[illegible]			COMMERC.	VEP
1350	[illegible]				1	COMMERC.	VEP
	[illegible]			B∧W	1	COMMERC.	VEP
	[illegible]-1-HANFORD-1	US	PWR	B∧W	1	COMMERC.	WPPSS
1243	WPPSS-4-HANFORD-3	US	PWR	B∧W	1	COMMERC.	WPPSS
1159	CRYSTAL-RIVER-3	US	PWR	B∧W	2	COMMERC.	FPC
1309	SOUTH-RIVER-1	US	PWR	B∧W	4	COMMERC.	CP∧L
1310	SOUTH-RIVER-2	US	PWR	B∧W	4	COMMERC.	CP∧L
1311	SOUTH-RIVER-3	US	PWR	B∧W	4	COMMERC.	CP∧L

COUT : 16.22

Using the appropriate system command will permit this report to be kept as part of the selection (saved as a question) either in a fixed context (output keywords and sort conditions) or leaving the user to choose the sort order and the breaking conditions. It is possible to use direct keywords (as above) or computed keywords by entering arithmetic expressions (e.g. COUPLAGE-TRAVAUX) with an appropriate title.

One may now get a power balance (for the selected units) by calling the function BILAN :

```
[5] BILAN 19770101
RUPTURE PAYS ? : NO
RUPTURE FILIERE ? : NO

CENTRALES INSTALLEES
                        19157.0    6689.0    6382.0   8

CENTRALES EN EXPLOITATION
                        16385.0    5729.0    5476.0   7

CENTRALES EN CONSTRUCTION
                        29836.0   10251.0    9729.0  10

CENTRALES EN COMMANDE
                        51006.0   17644.0   16794.0  15

CENTRALES TOTALES
                        99999.0   34584.0   32905.0  33
COUT :      19.40

[5] V

COUT DE LA SELECTION :        26.10
```

PTH

PBRUT

PNET

number

is the sum of 1, 3, 4

3.3 Direct use of user defined functions

The BILAN function was specially written for the N.P.P. Data Base. There are several other specific functions. For example, one may compute the average period between the start of construction (TRAVAUX) and the first production of electricity (COUPLAGE), i.e. the mean time to become operational :

```
      SELECTION

[1]   SELECTION ET(COUPLAGE≠0)ET (TRAVAUX≠0)
...TROUVES : 8
COUT :       4.60

[2]   MOYENNE COUPLAGE MOINS TRAVAUX
...VALEUR :    .
2253.25 ──────────────────────────────→ value in days

COUT :       2.57

[2]   ANNEE MOYENNE COUPLAGE MOINS TRAVAUX
...VALEUR :
6 2 1 ──────────────────────────────→ value in year,
                                      month, day
COUT :       2.95                     (standard year)

[2]   ∇

COUT DE LA SELECTION :       10.71
```

It is easy to install new functions from public libraries. For example, to implement a histogram facility, we define a HISTOGRAMME function with the appropriate specifications (as explained in [1]) and we copy the required functions as explained in the STATPACK Reference Manual [5]. By a)SAVE command, the new functions are kept as part of our DBMS.

```
      )LOAD SELECTION
SAVED  15.14.17 08/02/77
POINTEUR = 1
0

      )COPY 42 STAT1 DESCRIPTION STATISTICS HISTOGRAM MEDIAN MODE
SAVED    9.28.45 09/17/76
      )COPY 42 STAT1 GEOMETRIC HARMONIC QUADRATIC CUMULATIVE
SAVED    9.28.45 09/17/76

      )SAVE
  15.16.47 08/02/77 SELECTION
```

```
      ∇HISTOGRAMME[□]∇

    ∇ R←PAS HISTOGRAMME V;□IO;□PP
[1]   □IO←1
[2]   □PP←6
[3]   V←MASK/V
[4]   ⍎(PAS<0)/'V←(V≠0)/V'
[5]   (|PAS) DESCRIPTION V
[6]   R←CC←10
[7]   2 1 ρ' '
    ∇
```

user interface between the Database and the functions copied from the public library.

The HISTOGRAMME function may now be used directly under SELECTION. It has taken less than one hour to find the appropriate functions, to write the interface and to test it under SELECTION. If this new capability is recognized as important, one may ask the Data Base Administrator to include this set of functions in the data base library for dynamic call, which will reduce demands on the user's workspace. The use of the HISTOGRAMME function is shown below :

```
      SELECTION

[1] PAYS=US
...TROUVES : 282
COUT :      1.98

[2] ¯10 HISTOGRAMME COUPLAGE MOINS TRAVAUX

A DESCRIPTIVE ANALYSIS ON THE INPUT DATA VECTOR YIELDS THE FOLLOWING
MAXIMUM 3656
MINIMUM 30
AVERAGE 1741.71
STD.DEV 707.624
RANGE   3626
NO. OBS 96

ENTER UPPER AND LOWER LIMITS IN VECTOR FORM.
IF THE COMPUTED MAX AND MIN ARE DESIRED ,ENTER 0
□:
      3683 30                                          one will get
FREQUENCY HISTOGRAM ; EACH STAR = 1 PERCENT            a repartition
                                                       per year
                                                       (365.25 days)

      LOWER      UPPER        NO.     REL.FREQUENCY IN PERCENT
      30.00     395.30          3     ***
     395.30     760.60          6     ******
     760.60    1125.90          7     *******
    1125.90    1491.20         21     **********************
    1491.20    1856.50         17     ******************
    1856.50    2221.80         19     ********************
    2221.80    2587.10         13     **************
    2587.10    2952.40          5     *****
    2952.40    3317.70          4     ****
    3317.70    3683.00          1     *

THE FOLLOWING ARE PERCENT AND NUMBER OUTSIDE INDICATED LIMITS
      30.00        .00          0
    3683.00        .00          0

THE MEDIAN IS          1745
THE MODE IS            1384.65
THE GEOMETRIC MEAN IS  1511.62
THE HARMONIC MEAN IS   876.099
THE QUADRATIC MEAN IS  1878.58
```

```
CUMULATIVE FREQUENCY HISTOGRAM
      LOWER        UPPER         NO.      REL. CUMULATIVE FREQ. IN PERCENT
      30.00       395.30           3     *
     395.30       760.60           9       *
     760.60      1125.90          16           *
    1125.90      1491.20          37                 *
    1491.20      1856.50          54                       *
    1856.50      2221.80          73                             *
    2221.80      2587.10          86                                   *
    2587.10      2952.40          91                                     *
    2952.40      3317.70          95                                       *
    3317.70      3683.00          96                                       *

COUT :      10.81

[2] ▽
```

We may be surprised by the distribution of the delays between the start of construction and the production of electricity. One month of delay seems to indicate an error in our data. Calling SELECTION, we submit the proper query and EDITION will give us more details. It is now easy for the Data Base Administrator to find what may be wrong (see the listing below).

This example illustrates the power of an interactive DBMS.

3.4 Interactive updating

Calling the MAJ (update) function under SELECTION will allow interactive and context oriented updating.

```
SELECTION

[1] CONSTRUCTEUR=WEST
...TROUVES : 0
COUT :       2.18

[2] CONSTRUCTEUR=WEST.
...TROUVES : 121
COUT :       2.25

[3] MAJ CONSTRUCTEUR
POINTEUR = 1
NOMBRE DE VALEURS A METTRE A JOUR 121
CONTROLE ? : N
ENTREZ LES NOUVELLES VALEURS (MEME ORDRE QUE PRECEDENTS
▽: WEST
CONTROLE ? : N
COUT :       3.71

[3] CONSTRUCTEUR=WEST
...TROUVES : 121
COUT :       2.15

[4] ▽

COUT DE LA SELECTION :       10.71
```

this name is wrong (0 values are found.)

One needs to remove the period. Let us call the update function (MAJ)

All the names have been replaced in a single operation.

The updating could be done element by element as well.

```
SELECTION

[1] (PAYS=US)ET(COUPLAGE≠0)ET(TRAVAUX≠0)ET(395≥COUPLAGE MOINS TRAVAUX)
...TROUVES : 3
COUT :       7.82

[2] EDITION
LARGEUR :37
MOT_CLEF : CONSTRUCTEUR
LARGEUR :50
MOT_CLEF : TRAVAUX
LARGEUR :62
MOT_CLEF : DIVERGENCE
LARGEUR :74
MOT_CLEF : COUPLAGE
LARGEUR :85
MOT_CLEF : CALCUL
NOM DU RESULTAT : DELAI OP.
CALCUL : JOUR COUPLAGE-TRAVAUX
LARGEUR :95
MOT_CLEF : PTH
LARGEUR :105
MOT_CLEF : PBRUT
LARGEUR :114
MOT_CLEF : ∇
PHASE DE TRI :
MOT_CLEF : ∇
RUPTURE
MOT_CLEF : ∇
DECOUPAGE ? : ∇
```

it is the number of units found in the first cell.

we ask for a computed field

which will be expressed in days

no sort

no sequence modification

N°	NOM	PAYS	FILIERE	CONSTRUCTEUR	TRAVAUX	DIVERGENCE
1016	BORAX-3	US	BWR	ANL	1955 6 1*	1955 6 9
1017	BORAX-4	US	BWR	ANL	1956 11 1*	1956 12 1*
1004	PM2-A-(GROENLAND)	US	PWR	ALCO	1960 7 1*	1960 10 2

```
COUT :      13.68

[2] ∇

COUT DE LA SELECTION :      22.11
```

COUPLAGE	DELAI OP.	PTH	PBRUT
1955 7 1	30	15.5	2.4
1957 4 1	151	15.5	2.4
1960 11 12	134	10	1.9

see 3.3

3.5 Multi-user data base

As a final example, let us see an APL session for a user who has been permitted a restricted use of the data base : relatively few keywords are authorized and the access is restricted to non military power plants.

```
     )1001                                          logon (sign on)
OPR:  CISI-SIA2 APL SERVICE
027) 17.06.16 08/02/77 COURS

   . A P L .                                        APL is active

     )LOAD 1450 SELECTION                           Let us load the work-
SAVED  16.01.12 08/02/77                            space from the DB Admin-
POINTEUR = 1                                        istrator Library. The files are
0                                                   automatically open

     SELECTION

[1]  PAYS=US                                        We found previously 282
...TROUVES : 276                                    nuclear power plants in
COUT :       1.91                                   the U.S. There are in
                                                    fact 6 military plants,
[2]  LISTE TYPE                                     TYPE 4 (military) is not
...VALEUR :                                         available for this user.
PROTOTYP
COMMERC.                                            The keywords are also
EXPERIM.                                            limited to the authorized
                                                    list as loaded during the
COUT :       2.15                                   file opening operation.

[2]  MOTSCLEFS                                      This user will never know
...VALEUR :                                         which possible keywords
NUMERO                                              are forbidden to him.
NOM
PAYS
FILIERE
BOUCLE
TYPE
FROID
COMMANDE
CONTRAT
CHCOMB
DIVERGENCE
COUPLAGE

COUT :       .81

[2]  ∇

COUT DE LA SELECTION :        5.33

     )OFF                                           log off (sign off)
027  17.08.02 08/02/77 COU
CONNECTED     0.01.46  TO DATE     0.31.47
CPU TIME      0.00.02  TO DATE     0.00.10          )running statistics
                                                    )provided by the system.
```

4. CONCLUDING REMARKS

The initial GDMS implementation for this nuclear power plant data was abandoned in 1974. The present project was started in October 1976, and development costs so far are 1 man-year for system design and 3 man-months for APL coding.

The main problem proved to be data loading and validation, for which some interesting features were developed. Development is continuing on graphics capabilities and cross- and contingency tables for data analysis and economic studies. The system is expected to be fully operational at the end of 1977.

Further extensions which may be envisaged are :

- Safety information
- Inclusion of research reactors
- Nuclear waste data for optimisation of the management of waste storage areas
- Linkage with other nuclear plant data bases
- Linkage with economic data bases, to compare predictions and reality in the energy field.

Some lessons which may be drawn from our experience are :

- The quality of data is critical to the usefulness of the data base. A few good data are better than many poorly checked values.
- The system should be extensible, and should be written in a high-level programming language. It should be possible to make extensions rapidly, without upsetting the existing data base. APL offers a highly cost-effective solution.
- It should be possible to personalize data input and output. Graphics are certainly the most valuable tool for data representation.

5. REFERENCES

[1] G. HERVY and G. MARTIN : "Manuel d'Utilisation du Système de Bases de Données LGI"
Rapport technique CISI - PARIS (France)

[2] APL - Manuel de Référence - IBM manual GHF2-0056-0
APL-SV - Manuel de Référence - IBM manual SHF2-0080-0
(These manuals are available in English)

[3] G. MARTIN : "An APL Approach to Databases"
in this issue -

[4] G. HERVY : "Manuel d'Utilisation de la Base de Données des Centrales Nucléaires dans le Monde"
En preparation - Manuel CEA - PARIS (France)

[5] APL Statistical Library - Program Description and Operations Manual - IBM manual SH20-1841-0

DESCRIPTION EN FRANCAIS ← language selection

```
)LOAD DEFINITION
SAVED  11.38.52 08/01/77
```

← loading the workspace

↓ date and time of the last SAVE operation

LISTE DES MOTS-CLEFS POSSIBLES

N°	MOT-CLEF	SIGNIFICATION (LIBELLE)
1	NUMERO	REFERENCE INTERNE DE LA CENTRALE (N° DE SEQUENCE)
2	NOM	NOM
3	PAYS	PAYS
4	FILIERE	FILIERE
5	BOUCLE	NOMBRE DE BOUCLES
6	PTH	PUISSANCE THERMIQUE (NIVEAU 2)
7	PBRUT	PUISSANCE ELECTRIQUE BRUTE (NIVEAU 2)
8	PNET	PUISSANCE ELECTRIQUE NETTE (NIVEAU 2)
9	PDATE	DATE D'APLICATION DU CHANGEMENT DE PUISSANCE (NIVEAU 2)
10	GROUPAGE	NOMBRE DE CHANGEMENT DE PUISSANCE (NIVEAU 2)
11	TYPE	CARACTERE DE LA CENTRALE (COMMERCIALE, MILITAIRE ....)
12	FROID	MODE DE REFROIDISSEMENT DU CONDENSEUR PRINCIPAL
13	SITE	SITUATION GEOGRAPHIQUE
14	COMMANDE	DATE DE COMMANDE
15	CONTRAT	DATE DE CONTRAT
16	PREVISION	DATE DE PREVISION INITIALE DE MISE EN EXPLOITATION COMMERCIALE
17	CONSTRUCTEUR	CONSTRUCTEUR DE LA CHAUDIERE NUCLEAIRE
18	COMPAGNIE	COMPAGNIE D'ELECTRICITE PROPRIETAIRE
19	TURBO	FOURNISSEUR DU TURBOALTERNATEUR
20	CUVE	FOURNISSEUR DE LA CUVE OU DE L'ENCEINTE PRIMAIRE
21	GV	FOURNISSEUR DU GENERATEUR DE VAPEUR
22	ECHANGEUR	FOURNISSEUR DES ECHANGEURS DE CHALEUR
23	ARCHI	ARCHITECTE INDUSTRIEL
24	DDEPC	DATE DE DEMANDE DU PERMIS DE CONSTRUIRE
25	PCONST	DATE D'OBTENTION DU PERMIS DE CONSTRUIRE
26	TRAVAUX	DATE DE DEBUT DES TRAVAUX
27	DDEPEXP	DATE DE DEMANDE DU PERMIS D'EXPLOITATION
28	PEXPP	DATE D'OBTENTION DU PERMIS D'EXPLOITATION
29	CHCOMB	DATE DE DEBUT DE CHARGEMENT DU COMBUSTIBLE
30	DIVERGENCE	DATE DE DIVERGENCE INITIALE
31	COUPLAGE	DATE DE PREMIER COUPLAGE AU RESEAU
32	EXPLOITATION	DATE EFFECTIVE D'EXPLOITATION COMMERCIALE
33	PMAX	PUISSANCE MAXIMALE CONTRACTUELLE ATTEINTE
34	FIN	DATE DE MISE HORS SERVICE OU D'ABANDON

FIGURE 1

DESCRIPTION IN ENGLISH

LIST OF POSSIBLE KEYWORDS

N°	KEYWORD	MEANING (LABEL)
1	NUMERO	N.P.P. INTERNAL REFERENCE (SEQUENCE NUMBER)
2	NOM	NAME
3	PAYS	COUNTRY
4	FILIERE	TYPE OF REACTOR
5	BOUCLE	NUMBER OF LOOPS
6	PTH	THERMAL POWER (LEVEL 2)
7	PBRUT	GROSS ELECTRICAL POWER (LEVEL 2)
8	PNET	NET ELECTRICAL POWER (LEVEL 2)
9	PDATE	EFFECTIVE DATE OF POWER CHANGE (LEVEL 2)
10	GROUPAGE	NUMBER OF POWER CHANGES (LEVEL 2)
11	TYPE	NATURE OF THE POWER PLANT (COMMERCIAL, MILITARY ....)
12	FROID	MAIN CONDENSOR COOLING METHOD
13	SITE	GEOGRAPHICAL LOCATION
14	COMMANDE	DATE OF ORDER
15	CONTRAT	DATE OF CONTRACT
16	PREVISION	ORIGINAL DATE OF COMMISSIONING
17	CONSTRUCTEUR	N.S.S.S. CONSTRUCTEUR
18	COMPAGNIE	UTILITY
19	TURBO	DELIVERY CONTRACTANT FOR TURBINE GENERATOR
20	CUVE	DELIVERY CONTRACTANT FOR REACTOR VESSEL
21	GV	DELIVERY CONTRACTANT FOR STEAM GENERATOR
22	ECHANGEUR	DELIVERY CONTRACTANT FOR HEAT EXCHANGER
23	ARCHI	ARCHITECT ENGINEER
24	DDEPC	DATE OF CONSTRUCTION PERMIT APPLICATION
25	PCONST	DATE OF CONSTRUCTION PERMIT ISSUANCE
26	TRAVAUX	DATE OF CONSTRUCTION START
27	DDEPEXP	DATE OF OPERATING LICENCE APPLICATION
28	PEXPP	DATE OF OPERATING LICENCE ISSUANCE
29	CHCOMB	DATE OF FUEL LOADING BEGINNING
30	DIVERGENCE	DATE OF INITIAL CRITICALITY
31	COUPLAGE	DATE OF INITIAL ELECTRICITY
32	EXPLOITATION	DATE OF COMMERCIAL OPERATION
33	PMAX	DESIGN ELECTRICAL GROSS POWER REACHED
34	FIN	DATE OF CANCELLING OR COMPLETED OPERATIONS

FIGURE 2

DESCRIPTORS OF POSSIBLE KEYWORDS

N°	KEYWORD	DATA BASE	FILE NAME	RECORD	INDEX IN TABLES (HOOKS) TEC/TEL	TTA	TEL	TAE	TEQ	NUMBER OF BLOCKS
1	NUMERO	1	CENTR	26	0	0	0	0	0	1
2	NOM	1	CENTR	5	-6	0	1	0	1	2
3	PAYS	1	CENTR	7	-4	0	0	0	0	1
4	FILIERE	1	CENTR	8	-2	0	0	0	0	1
5	BOUCLE	1	CENTR	21	0	0	0	0	0	1
6	PTH	1	CENTR	2	7	0	2	0	0	1
7	PBRUT	1	CENTR	3	7	0	2	0	0	1
8	PNET	1	CENTR	4	7	0	2	0	0	1
9	PDATE	1	CENTR	1	1	0	4	0	0	1
10	GROUPAGE	1	CENTR	27	0	0	0	0	0	1
11	TYPE	1	CENTR	24	5	0	0	0	0	1
12	FROID	1	CENTR	20	0	0	0	0	0	1
13	SITE	1	CENTR	25	0	0	0	0	0	1
14	COMMANDE	1	CENTR	13	1	0	3	0	0	1
15	CONTRAT	1	CENTR	23	1	0	3	0	0	1
16	PREVISION	1	CENTR	22	1	0	3	0	0	1
17	CONSTRUCTEUR	1	CENTR	9	-2	0	0	0	0	1
18	COMPAGNIE	1	CENTR	10	-2	0	0	0	0	1
19	TURBO	1	CENTR	11	-2	0	0	0	0	1
20	CUVE	1	CENTR	37	-2	0	0	0	0	1
21	GV	1	CENTR	38	-2	0	0	0	0	1
22	ECHANGEUR	0		0	0	0	0	0	0	0 ← not active
23	ARCHI	1	CENTR	12	-2	0	0	0	0	1
24	DDEPC	1	CENTR	32	1	0	3	0	0	1
25	PCONST	1	CENTR	33	1	0	3	0	0	1
26	TRAVAUX	1	CENTR	14	1	0	3	0	0	1
27	DDEPEXP	1	CENTR	34	1	0	3	0	0	1
28	PEXPP	1	CENTR	36	1	0	3	0	0	1
29	CHCOMB	1	CENTR	35	1	0	3	0	0	1
30	DIVERGENCE	1	CENTR	15	1	0	3	0	0	1
31	COUPLAGE	1	CENTR	16	1	0	3	0	0	1
32	EXPLOITATION	1	CENTR	17	1	0	3	0	0	1
33	PMAX	1	CENTR	18	1	0	3	0	0	1
34	FIN	1	CENTR	19	1	0	3	0	0	1

if 0 : no action
if >0 : argument is numeric
if <0 : argument is litteral

not used in this application

DESCRIPTION OF THE TABLES USED IN THE DATABASE

DESCRIPTION OF FILES TABLE

N°	FILE	BASE	USAGE
0		0	EMPTY FILE (KEYWORD NOT ACTIVE)
1	CENTR	1	NUCLEAR POWER PLANT DATABASE (LEVELS 1+2)

CODING/DECODING FUNCTIONS FOR INPUT/OUTPUT

N°	TEC	TED	DESCRIPTION OF ACTION
1	CAD	CDATE	CONVERSION OF CALENDAR DATES (DAY, MONTH, YEAR) TO JULIAN DATES (INTEGERS)
2	COD	DCOD	STRING OF 10 CHARACTERS ↔ 1 REAL NUMBER (8 BYTES)
3	COD1	DCOD1	STRING OF 5 CHARACTERS ↔ 1 INTEGER (4 BYTES)
4	PAYC	PAYD	EXTERNAL COUNTRY NAME ↔ INTERNAL REFERENCE IN A TABLE (VALIDITY CHECK)
5	RIEN	TXPD	NO CODING (RIEN). PLAIN VALUE (CHARACTERS) RESTITUTED FOR EDITION.
6	RIEN		NO CODING/DECODING (NAMES ARE NOT ENCODED)
7	PUIC	PUID	FLOATING NUMBER (X DECIMALS) ↔ INTEGER (MULTIPLICATION BY 10 POWER X)

FUNCTIONS TO BE APPLIED DURING A READ OPERATION

N°	TEL	DESCRIPTION OF ACTION
1	NOML	CATENATION OF THE 2 PARTS OF NAME FOR EDITION
2	CAR3	EXTRACTION OF THE CURRENT ROW (POWER) IN THE LEVEL 2 RECORD
3	MODU	SPECIAL TREATEMENT FOR NEGATIVE (INCOMPLETE) DATES
4	CARD	SAME FOR LEVEL 2 DATES

FUNCTIONS APPLIED TO OPERATIONS

N°	TEQ	DESCRIPTION OF ACTION
1	NOMO	REPLACEMENT OF OPERATION = BY THE APL EQUIVALENT FOR ARRAYS (^.=)

LABORATORY ANIMAL DATA BANK - ENVIRONMENTAL HUSBANDRY FACTORS, HEMATOLOGY, AND CLINICAL CHEMISTRY FILES*

Kang Hsu

Information Systems Section
Computer, Information Systems, and Education Department
Battelle Columbus Laboratories
Columbus, Ohio 43201

ABSTRACT

BASIS (Battelle Automated Search Information System) is introduced along with a detailed description of the environmental husbandry factors, hematology, and clinical chemistry files of the LADB (Laboratory Animal Data Bank). This paper has attempted to show that LADB involves a great deal of data manipulation, data base management, and owncode interface software. LADB illustrates how a generalized data management system called BASIS can be utilized to handle both scientific and technical information processing tasks. This sophisticated yet easy-to-use LADB system has the potential to bring some dramatic impact in the research area of animal science.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the assistance and suggestions from members of Battelle LADB research team including Dr. William J. Clarke, Dr. Charles R. Claydon, Albert R. Fish, Dr. Willard Gersbacher, Dr. Hugh H. Harroff, Victor A. Kean, Jr., Robert T. Niehoff, Lyn Sander, Richard C. Simon (LADB Project Manager), Kenneth F. Szczesny, Dr. Daryl C. Thake, and Dr. Ralph E. Thomas.

---

* This work was supported by Contract NO1-LM-5-4747 of the National Library of Medicine and the Department of Health, Education and Welfare Committee to Coordinate Toxicology and Related Programs.

INTRODUCTION

Currently the Information Systems Section of the Battelle Columbus Laboratories is conducting a research project to design and implement the Laboratory Animal Data Bank (LADB) sponsored by the National Library of Medicine and the Department of Health,Education and Welfare Committee toCoordinate Toxicology and Related Programs. The system design of the LADB is based on BASIS (Battelle Automated Search Information System) which is a completely user-oriented, information storage, retrieval, and analysis system [1]. LADB represents a successful application of BASIS in the area of handling scientific and technical information tasks. The ultimate goal of this project is to design and implement LADB which is an on-line and easy-to-use laboratory animal data base comprising laboratory control animal data supplied by numerous research organizations with a wide range of environment conditions. LADB is accessible via computer terminals allowing biomedical scientists, researchers, breeders, and managers of animal laboratories to search, retrieve, analyze, and statistically manipulate control animal information. There are several objectives of the LADB. Among them are:

- To provide comparison data with animals being tested.
- To assist researchers in making relational selections of the best species and strain of laboratory animal for a specific biomedical experiments.
- To provide comparisons of laboratory data with consideration given to origin, strain, environment and husbandry conditions, and test methods used.
- To establish more accurate baseline values considering similarities of colonies with regard to the factors outlined above.
- To determine which factors most significantly affect results for a variety of test parameters.
- To establish incidences of spontaneous disease conditions and pathologic lesions in various strain and species of laboratory control animals and determine how these are influenced by the factors outlined above.
- To assist in monitoring test results more efficiently by comparing the research data with the information stored in the LADB.
- To assist in designing experimental protocols.

The kinds of laboratory control animal data which are collected, screened, and stored in the LADB are data on animal species/strain, environmental and husbandry factors, physical characteristics, hematology, clinical chemistry, and pathology. The kinds of data described in this first paper are hematology, clinical chemistry, and environmental and husbandry factors. In addition to describing LADB, this paper also emphasizes the fact that the capabilities provided by BASIS can be efficiently utilized in handling scientific and technical information.

BASIS OVERVIEW

BASIS is a completely user-orientied, interactive information storage, retrieval, and analysis system. Operational since 1970, the storage and retrieval module has been designed to allow users to search large files of textual or numerical information by index terms or data values and rapidly retrieve information

satisfying the search criterion [1]. BASIS supports sophisticated computational, owncode, profile, monitor, on-line sort, thesaurus control, range search, statistical analysis, interactive graphics, tabular reports, and generalized data base creations and maintenance capabilities. BASIS thereby fulfills the requirements crucial to GDMS (Generalized Data Management System) [3-5]. The system has been utilized to create and maintain over one hundred data bases that differ widely in size, complexity, and scope. These data bases comprise a national network for scientific and technical information [9], and cover such areas as materials and metals, medical and cancer research [6-8], social and economic, management, and library science [12]. BASIS is both economically viable and completely user oriented. No programming experience is needed to use the system. Response time to the entry of individual search parameters averages only a few seconds and output can be secured both on-line and off-line.

Since the BASIS software is written in a high level language (FORTRAN), it is adaptable to other computer systems. BASIS is currently operational on the following computers; CDC 6000 series, UNIVAC 1100 series [10,11], XEROX Sigma 9 [12], DEC 10, DEC 20, and IBM 360/370 series. Utilizing experience gained in the implementation of BASIS on these computer systems, there should be no major technical problems to implement BASIS on other third generation time-shared computer system. An organization may choose to implement their data bases on Battelle's computer or acquire BASIS to operate on their own computer.

BASIS ARCHITECTURE

BASIS is a completely modular software system [2,13,18]. Each module has a number of submodules responsible for specialized functions. The BASIS system is simialr in many respects to other on-line storage and retrieval systems, but it provides a wide range of additional capabilities including:

(1) Combined TEXTUAL and NUMERIC DATA retrieval and analysis

(2) On-line DATA MANIPULATION and STATISTICAL ANALYSIS

(3) On-line SORTING

(4) On-line REPORT GENERATOR

(5) On-line THESAURUS

(6) Complete system interaction MONITORING

(7) User search and save procedures - PROFILE

(8) INVERTED FILE or SEQUENTIAL FILE searching

(9) Ability to execute external programs (OWNCODE) from BASIS

(10) User-oriented RETRIEVAL AIDS

(11) Extremely fast retrieval for SMALL and LARGE files

(12) Sophisticated file CREATION and MAINTENANCE packages (including the ability to easily update very large files).

All of the above features are fully integrated, tested, and operational in a real-world production environment.

BASIS FILE ORGANIZATION

"File organization" is used to describe the manner in which a file is logically structured. The file organizations designed for BASIS are meant to match the processing requirements of the BASIS system to the capabilities of the structures used. In addition to the commonly used sequential and random file organizations, there are two more sophisticated file organizations designed and utilized by BASIS. They are symbolic keyed (SK) file and numeric keyed (NK) file which provide efficient and comprehensive capabilities for the BASIS user to directly access a record in a file at random [13]. Both SK and NK files access methods provide for a machine independent method for creating and maintaining large files of randomly stored variable size records that can be retrieved by logical sequential position, and appropriate keys (textual string key for SK files or numeric keys for NK files). The variable length records are stored in fixed length data blocks. Usually each block will contain several records. The records are scattered across the data blocks wherever they fit. A record space index is used to locate available space in the data blocks, and a new data block is only created when it is impossible for a new record to be placed in any existing data block. Old records may be replaced by different size new records. SK and NK files use the same kind of internal key index structure, index block splitting scheme, data blocks, record space index, and the file space allocation method. These carefully designed file organizations help make BASIS a generalized data management system.

LADB DESIGN

A conceptual system plan (Figure 1) for handling hematology and clinical chemistry data in LADB is presented here. This plan includes six major aspects: data specification and collection, the IADB (Individual Animal Data Base), document set generation and data summarization, LADB (Laboratory Animal Data Bank), structured search mode, and a statistical interface program. Prior to describing each of these six major aspects, it is important to describe the main reasons for summarizing individual animal information stored in IADB. Theoretically the IADB is suitable for searching. For example, a scientist could search IADB for hemoglobin on nine-month old female beagle dogs, via standard BASIS search mode, send the retrieved data to an interface program and perform appropriate statistical analysis. This approach does work yet it has drawbacks. The major problems of maintaining IADB on-line are:

- Since each data element of every IADB record contains a single value for each individual control animal, a great deal of disk storage is required to handle thousands of individual animal records.

- More computer time is required to access each stored individual animal data value to perform statistical calculations.

To overcome these drawbacks, it was decided by both NLM and Battelle to build LADB (Laboratory Animal Data Bank) as a summarized version of IADB (Individual Animal Data Base). In addition to the standard BASIS search mode, LADB is also equipped with a menu-driven structured search mode which provides many comprehensive yet easy-to-use search, retrieval, and analysis capabilities for the LADB user. The combination of LADB and structured search mode reduces dramatically the cost of

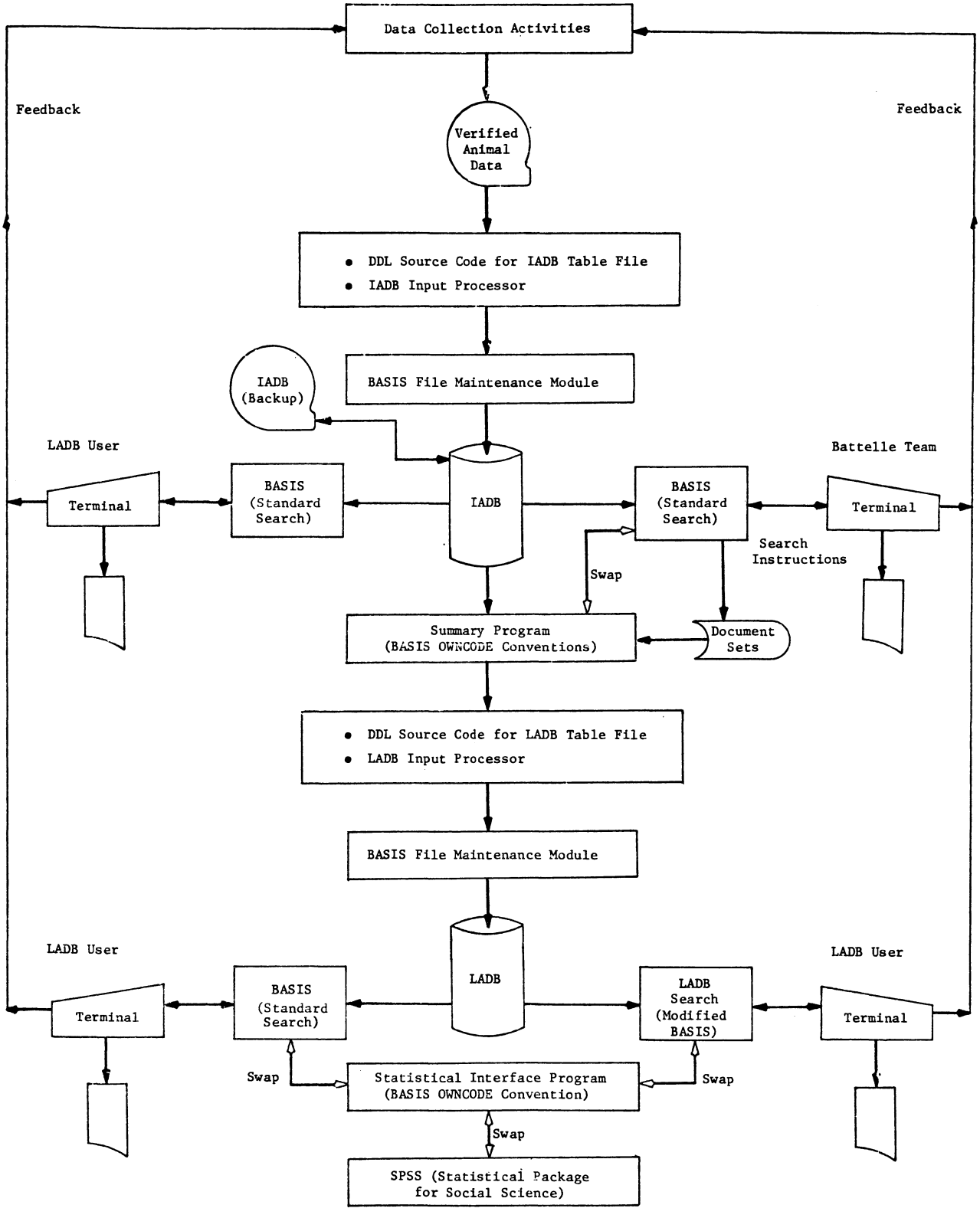


FIGURE 1. SYSTEM PLAN FOR HANDLING HEMATOLOGY AND CLINICAL CHEMISTRY DATA

using and maintaining LADB in terms of training time, disk space, and computer time.

LADB DATA SPECIFICATIONS AND COLLECTION

The primary goal of LADB is to collect pertinent data on certain strains of laboratory control animals, screen, summarize these data, and then build up an on-line and easy-to-use information system for use by any research scientist having need for such data. The quality of the data which are collected, screened, summarized, and stored into LADB plays a critical role in the acceptance of LADB by the biomedical community. The very first task involved in data collection is to work out the data element specifications. Each LADB data element is carefully defined and specified in terms of both technical and system specifications. The technical specification for a LADB data element includes element name, synonym(s) or abbreviation, major data group and subgroup classification, general description, acceptance criteria, screen methods, and certain data attributes such as units of measurement, number of significant digits, and preferred measuring method. The system specification for each data element defines both search and display mnemonics, display label, field or data element number, and whether the data element is searchable, displayable, or manipulatable statistically in various level of data files. With these well defined data element specifications, the next task is to design various data collection forms for recording the individual animal data. These Battelle designed data element specifications and LADB data collection forms can be found in the LADB Data Collection Manual [15]. Battelle has full responsibility to make contacts with potential data sources, evaluate the acceptability of data sources, identify and define the animal colonies, set up the conditions for data collection, define the mandatory data elements, establish procedures for data collection, actually collect the data, and screen the collected data. The derivation of acceptance criteria was carefully undertaken but not without some difficulties. A realistic approach was required because if these criteria were too stringent, very little data would qualify for LADB and if the criteria were too loose, LADB would be meaningless. These acceptance criteria represent a blend of required procedures, accepted procedures, and good laboratory practice, tempered with Battelle's years of research experience in biomedical and life sciences.

INDIVIDUAL ANIMAL DATA BASE

The IADB (Individual Animal Data Base) consists of four logical files which are the environmental and husbandry factors file, the hematology and clinical chemistry file, strain file, and the pathology file. As indicated in the Introduction, this paper only describes the first two logical files.

An animal colony is defined as a group of animals of the same strain, the same source laboratory, the same supplier (or breeder), the same microbial barrier, and maintained under the same or similar, control, environmental, and husbandry factors. In order to identify a unique animal colony, the environmental and husbandry factors logical file has more than a hundred data elements. There are numerous data elements which have been defined and collected in the hematology and clinical chemistry logical files. All of these hematology and clinical chemistry data elements are numeric. Each set of data has to be submitted by its data source together with its mandatory data elements. Where the data source uses units of measure different from LADB standard units, the "non-standard" units and their conversion factors must be recorded. For the irreversible cases, IADB treats them as separate elements [15]. A free form keying format has been designed to key and

verify all the individual animal data. A unique data collection form code is always assigned to each type of data.

The IADB is described to BASIS by using the BASIS Data Description Language (DDL) Compiler which creates a data base description for BASIS. The information from the data collection forms is keyed and then processed by IADB input processor (FORTRAN program) that presents the data to BASIS standard interface routines. BASIS then updates the data base via its data base maintenance facilities (Figure 1).

Data screening and quality control are a very significant part of the IADB input process (Figure 2). Using the facilities provided with BASIS, the IADB input processor plays a key role for handling data screening and quality control task for IADB. It checks for the presence of mandatory data elements, certain attributes, expected formats and data element specifications. It also performs the range check on numeric data elements and spelling checks on textual data elements via various internal dictionaries. Error messages are issued by the IADB input processor where expected conditions and data standards are not met. While the IADB input processor is producing appropriate transactions for all the good data and issuing error messages for all the questionable data, it also generates a record transaction file with all the input data in it. All the questionable data rejected by the IADB input processor are printed out for subsequent review. Values falling outside established ranges and misspelled text are checked for data transcription errors, decimal point problems, and keying mistakes. If none of these problems exist but the value is still outside the established normal range, Battelle veterinarians examine the data and decide whether this questionable data should be rejected, or accepted. Occasionally, the data source is recontacted to see if there is an explanation for this data value. Normal ranges can be adjusted if data suggest a trend toward a larger normal range. A standard editing program can be used to access the data file to correct records. Corrected data will be sent back to the IADB input processor for normal processing. In addition this error-free data file will be kept for future purposes.

SUMMARIZATION SCHEME AND LABORATORY ANIMAL DATA BANK

The Battelle-designed scheme used to summarize hematology and clinical chemistry data is a two-stage scheme (Figure 3). The first stage requires searching IADB via the standard search mode provided by BASIS. For hematology and clinical chemistry data, the search instructions are various unique combinations of animal colony, sex, age range (or body weight range for wild caught animals with unknown ages), observation date range, and the logical file (or subfile) type. The logical file (or subfile) search term SUBFILE:CH retrieves all the hematology and clinical chemistry records out of the IADB. The animal colony search term (for example COL:00022) defines the environmental and husbandry factors associated with the animals. The animal sex search term can be either SEX:MALE or SEX:FEMALE. Age ranges search terms (or the body weight ranges) are chosen based on the age categories defined by Battelle. There is a unique age category number assigned to each age range (or body weight range). This age category number is the age information stored in a summary record. The age related index term generated in the summary record level is the age category number concatenated with the corresponding age range (or body weight range). The observation date range search term divides the time interval in a yearly base. For example, the observation date range search term OBS:720101/721231 retrieves all the laboratory animal data observed in 1972 out of the IADB. The basic reason to construct the search instructions by using the unique combination of logical file type, animal colony, sex, age range (or body

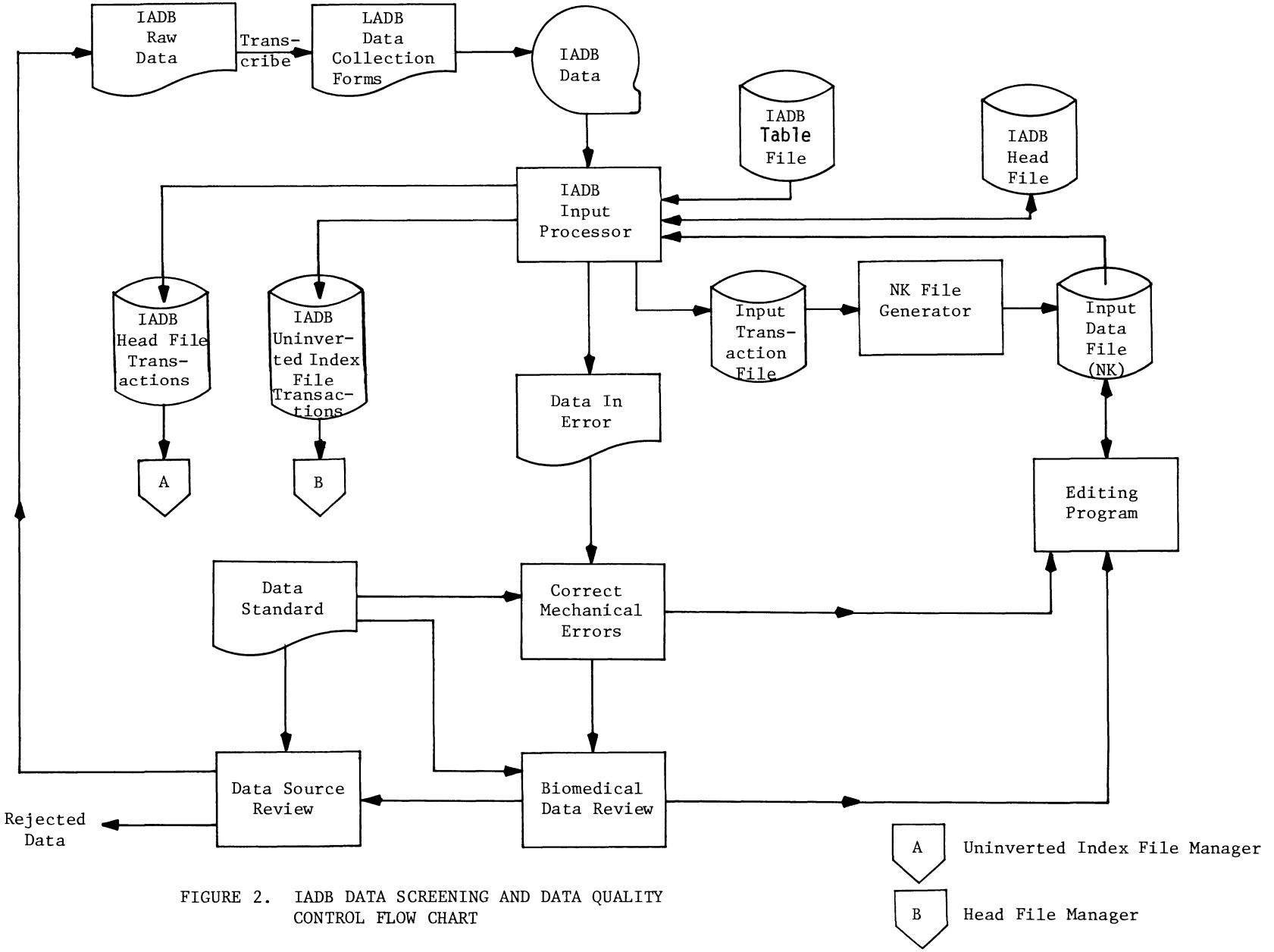


FIGURE 2. IADB DATA SCREENING AND DATA QUALITY CONTROL FLOW CHART

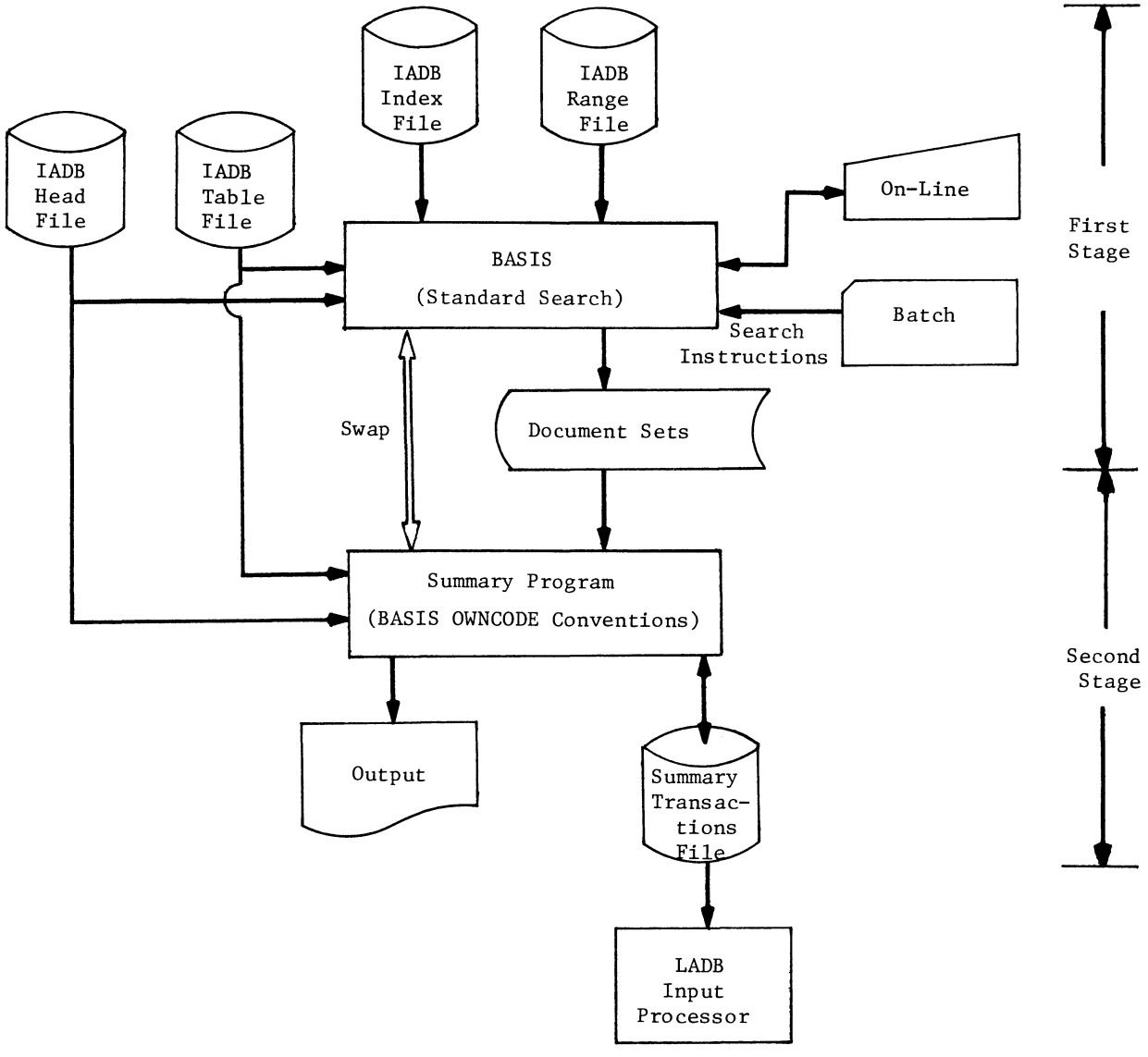


FIGURE 3. SUMMARIZATION SCHEME

weight range), and observation date range is because the document sets defined by these unique combinations are the appropriate data which can be used to generate the baseline values. In other words, each hematology and clinical chemistry data element in a summary record represents a group of data for the set of individual control animals with the same sex, category, observation year, logical file type, and environmental and husbandry factors. The product of this first stage effort is essentially to generate a series of document sets which will be processed by the summary program to generate the summary records.

The summarization scheme currently utilized in the second stage to create the summary records is the one commonly used by statisticians to summarize the raw data into frequency distributions [16]. For the purpose of minimizing the grouping error, the number of class intervals has been set to sixteen. The detailed procedure to create a summary record can be outlined as follows.

Step 1. Access the IADB to retrieve a set of individual animal records with identical colony, sex, age range, test year, and subfile type.

Step 2. For each numerical data field, find out its actual minimum (AMIN) and actual maximum (AMAX) data values, the unique animal count (UAC), and data field number (NFLD).

Step 3. Perform the necessary calculations to determine range of the values (r), the size of class interval width (w), and all the seventeen class boundaries (b_i, i = 1,17) for the sixteen class intervals (h = 16) according to the order of the following relevant formulas:

Let AMIN = actual minimum data value (see Step 2)
AMAX = actual maximum data value (see Step 2)
w = class interval width
h = 16 = number of class intervals
r = range of the data value
CMIN = calculated minimum data value
CMAX = calculated maximum data value
b_i = the i-th class boundary

then:

$w = (AMAX-AMIN)/h$

$r = AMAX - AMIN + w$

$w = r/h$ (w is redefined)

$CMIN = AMIN - w/2$

$CMAX = AMAX + w/2$

$w = (CMAX-CMIN)/h$ (w is again redefined)

$CMIN = CMIN - w/2$ (CMIN is redefined)

$CMAX = CMAX + w/2$ (CMAX is redefined)

$b_1 = CMIN$

$b_i = b_1 + (i - 1) * w$ (where i = 2, 3, ..., 16)

Step 4. Access the data values of this data field within the domain of the retrieved IADB records to find out the category frequencies (f_i , for the i-th class interval) via a binary search algorithm.

Step 5. Write out this portion of summary record according to the following format (each data element is separated by semicolon)

NFLD;CMIN;CMAX;UAC;w;f_1;f_2;...;f_{16};

Step 6. Repeat Step 2 through Step 5 for all the other data fields required to be summarized for this retrieved IADB record set defined by Step 1.

Step 7. Repeat Step 1 through Step 6 for the other requested combination of colony of animal, sex, age range, test, and subfile type to generate other sets of summary records as required.

This summary program will generate a summary transaction file which is the input file to the LADB input processor. Figure 4 is a schematic representation to illustrate the file relation between IADB and LADB. Based on the fact that a set of IADB records is often replaced by a single summary record, the disk space needed to maintain LADB as an on-line data bank is much less than IADB. The disk storage data included in Table 2 indicates that the amount of disk space saving is up to 86% [14]. As it stands now, LADB contains data on 10 animal species/strains broken down into 71 distinct colonies which include 11,000 unique laboratory control animals.

STATISTICAL INTERFACE MODULE AND STATISTICAL ANALYSIS

One of the major objectives of LADB is to provide the baseline data in an easy-to-use manner. For this reason, a BASIS OWNCODE module program has been designed and implemented to interface BASIS and the Statistical Package for Social Science (SPSS) [17]. For a requested baseline data field, this interface module will access a set of LADB data records retrieved by the user via the structured search to build up the following files: (a) A SPSS input file which consists of its category frequencies (f_i) and its associated middle point value (X_i) for the requested data field, and (b) A set of SPSS control statements file which consists of the requested data field name and unit, statistic procedure name, desired variables, selected statistic options, and other necessary SPSS control statements.

With these SPSS input and SPSS control statements files and the on-line version of SPSS executed by the statistical interface module, the user will obtain the statistical analysis results automatically (Figure 5). Currently this statistical interface module provides capabilities to handle three SPSS procedures which are FREQUENCIES, BREAKDOWN, and CROSSTABS [17]. The FREQUENCIES procedure is used to produce basic statistical information and an associated histogram (Appendix A). BREAKDOWN is used to make a breakdown results which may show the genetic drift. Finally, the CROSSTABS procedure can be used to generate a tabular report for the variables selected by the user. It is important to emphasize again that the BASIS OWNCODE module really makes BASIS an open-end generalized data management system which is very significant for handling the scientific and technical information tasks.

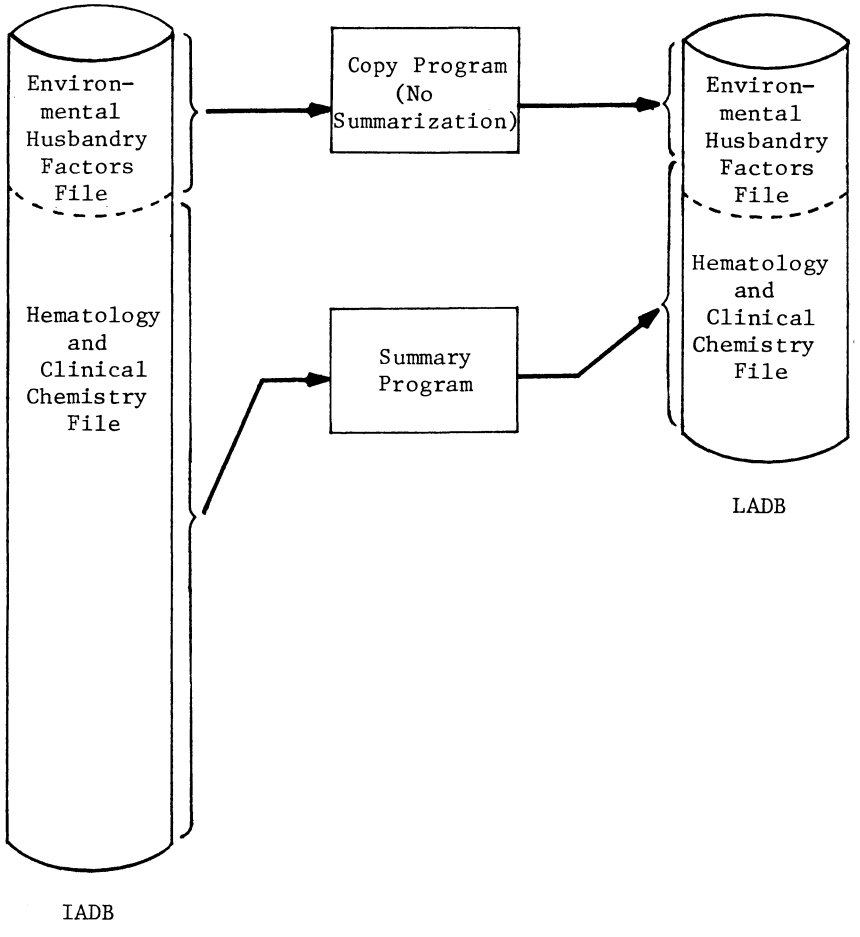


FIGURE 4. FILE RELATION BETWEEN IADB AND LADB

TABLE 2. DISK STORAGE STATISTICS

File Type	IADB (bytes)	LADB (bytes)
Head File	5,900,160	1,021,440
Inverted Index File	2,786,560	898,560
Range File	924,800	--
Table File	57,600	57,600
Uninverted Index File	9,507,840	673,280
Total	19,176,960	2,650,880

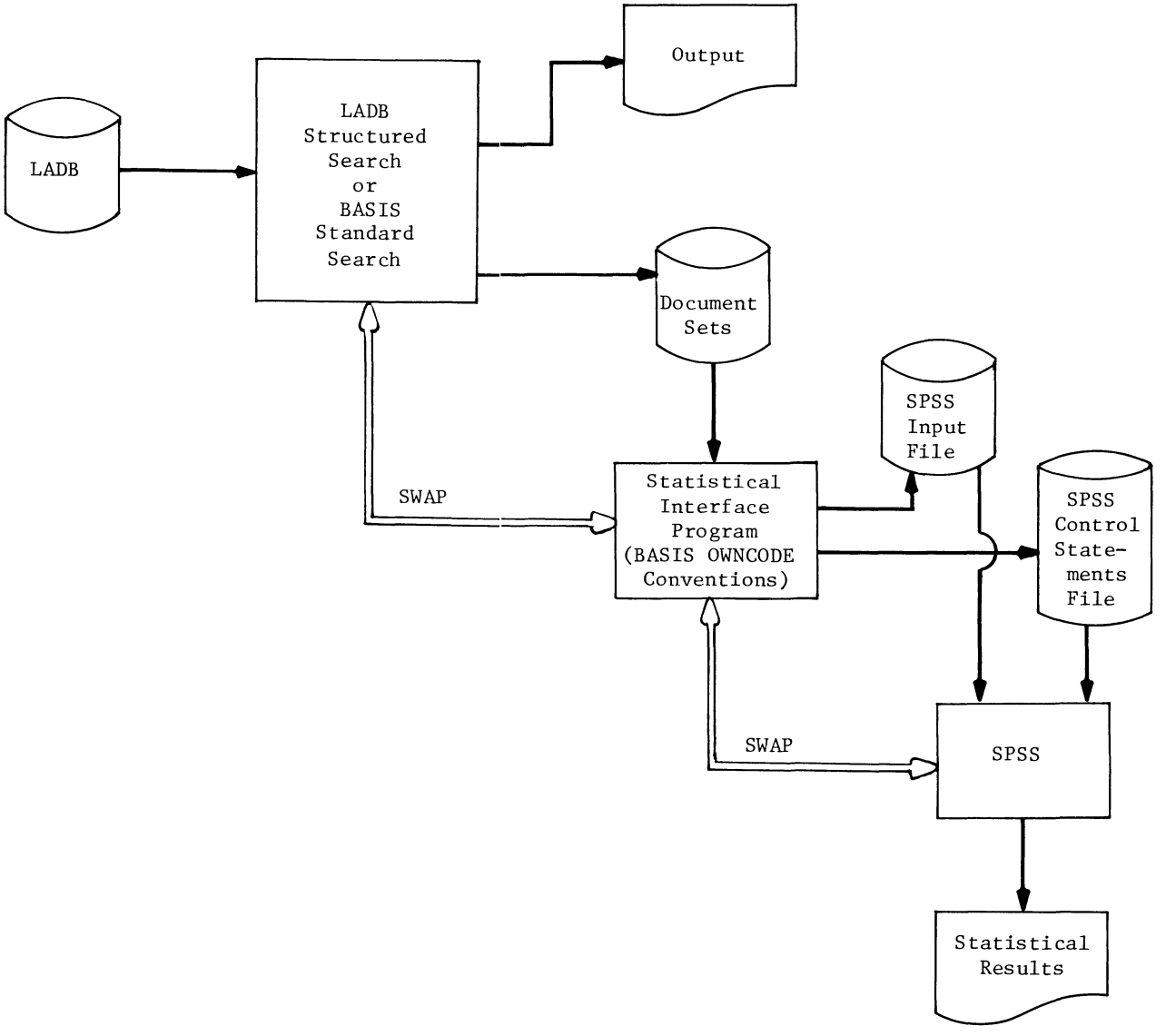


FIGURE 5. STATISTICAL INTERFACE FLOW CHART

STRUCTURED SEARCH

A menu-driven structured search has been designed and implemented to ensure that the LADB is extremely user oriented (Figure 6). The standard BASIS search mode was modified to have tree type menus built in and provides the structured search capabilities. The difference is that modified BASIS handles internally all the logical combinations based on the set and sequence of menus the user has chosen. All the menus used by LADB can be classified as one of the following types:

- The basic LADB service selection menu is used to choose one of the LADB basic services including:

    1. SEARCH (Build a profile)
    2. USE A PREVIOUSLY SAVED PROFILE
    3. SIGN OFF (Stop LADB)
    4. BUILD A WORK SET (Your Data)
    5. STATISTICS
    6. PRINT REPORTS

- The strain(s) or specie(s) selection menu is used to choose which strain(s) or specie(s) the user wants.

- The logical file selection menu is used to choose one of the logical files a user is interested in.

- The class(es) selection menu is used to choose the class(es) the users desire. Each class may contain a set of data field categories or a group of closely related data fields.

- The category(ies) selection menu is used to choose the category(ies) the user needs. Each category contains a group of closely related data fields.

- The data field search term(s) selection menu points to all the search terms associated with a specific data field. The user may 1) enter any stem term to bring in a list of search terms with the same stem for further selection via a "LOOK" option, or 2) ask the LADB to list a set of search terms associated with this specific data field for further selection via "LIST" option, or 3) leave this data file as it was chosen and go on to the next data field via a "LEAVE" option. For each data set selected, the LADB system will provide prompting to allow the user to make selections.

- The control flow selection menu is used to choose what the user is going to do next at each logical end of a search activity. The user may choose to either run immediate search for results, save the data elements of the search as a profile for future search, or go back to the class(es) selection menu to perform further selections. In case the user choose to run immediate search for result, the LADB will inform the user how many colony records were retrieved and ask the user to either make the profile more general/specific, or save the profile for future usage, or go back to the basic LADB service selection menu to select one of the basic services.

- The save profile menu provides the capabilities to save the current search profile for future usage.

- The previously saved profile selection menu provides the capabilities of letting the user retrieve one of the previously saved profiles.

- The work set construction menu allows the user to enter his own data. The

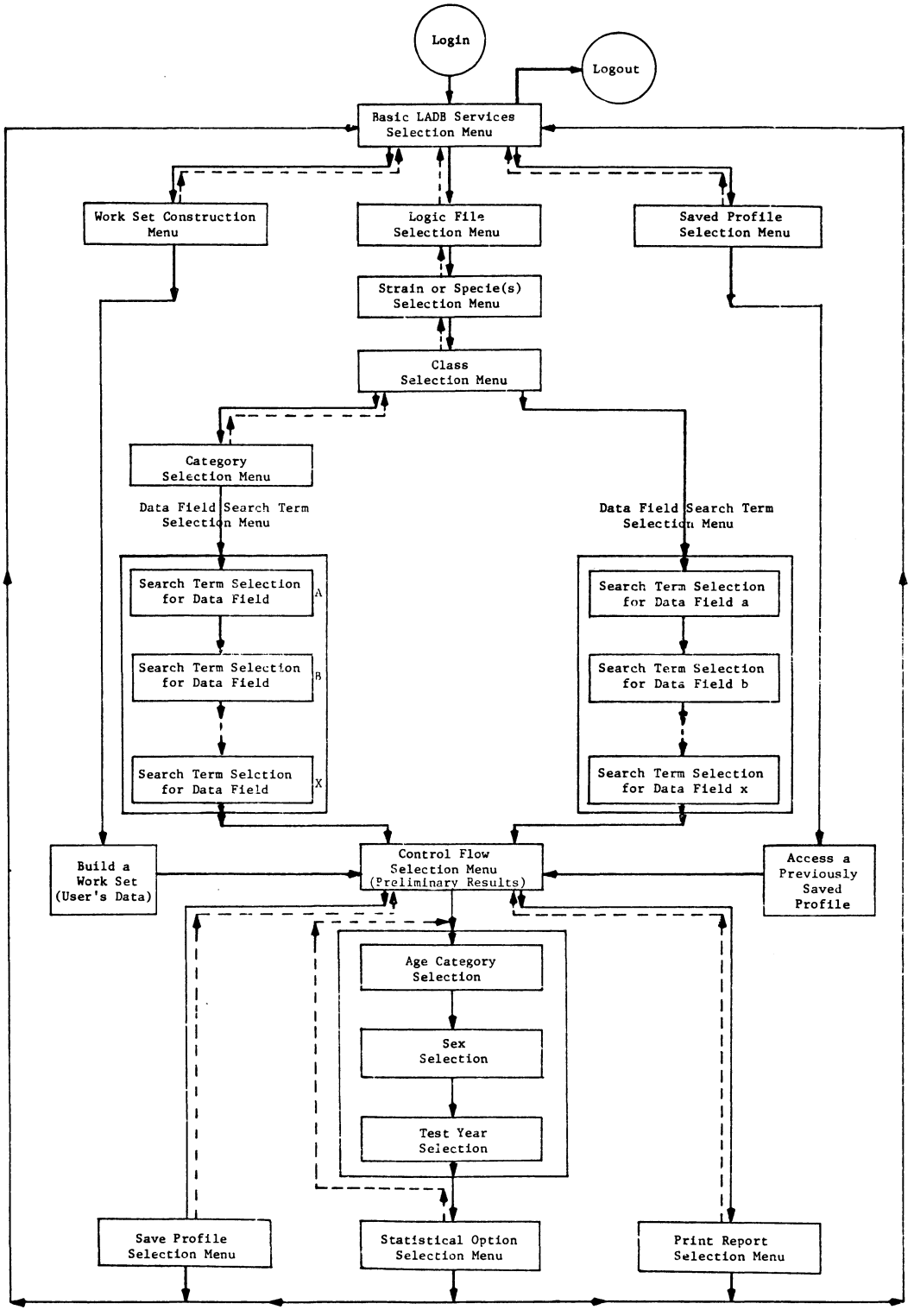


FIGURE 6. LOGICAL FLOW OF LADB STRUCTURED SEARCH

LADB system will process this set of data and offer a compatible statistical analysis.

- o The statistical option selection menu is used to define the necessary information for performing requested statistical analysis. Options include data field, procedure name, and variable(s).

- o The print reports selection menu is used to choose the contents of tabular reports.

INFORMATION CONTENT OF LABORATORY NUMERICAL DATA

The distribution of a set of laboratory numerical data can sometimes be described by Gaussians, or normal distribution (De Moivre, 1733). It is the theoretical distribution of the relative frequency of a large number of observations made under the same experimental conditions. This normal distribution has three distinct characteristics which are the major information content of any set of laboratory numerical data. The first characteristics is the central tendency (or signal) represented commonly by arithmetic mean. The second characteristic is the measurement of variability (or noise) represented by standard deviation. The third characteristic is the range represented by confidence interval.

The theory of normal distribution was developed from a mathemetical theory of errors. In the laboratory the researcher can not afford to make a large number of observations; as a result the researcher does not know the true populatoion mean and the true standard deviation by using the sample standard deviation. When the researcher does this, one should use the "Student's t-distribution" [19] which is independent of true standard deviation and is dependent only on the sample size. The t-distribution is flatter than the normal distribution but approaches it as the sample size increases, becoming identical to the normal distribution as the sample size approaches infinity. For practical purposes, researchers sometimes use the normal distribution for sample sizes more than 30. It is seen that Student's t-distribution with 30 degrees of freedom has characteristics approximately equal those of the normal distribution. With more than 30 observations, this means the information content of a laboratory numerical data set is characterized by its central tendency (arithmetic mean), variability (standard deviation), and range (confidence interval), regardless of whether the t-distribution or normal distribution is used. The information content loss of the laboratory numerical data due to the summarization scheme can be studied by comparing its major information content for those prior to and after the summarization. A comparative study has been carried out for the same set of statistical analyses with respect to the similar sets of animal data retrieved separately from IADB and LADB. The results included in Table 3 indicates that the amount of information content loss due to the aforementioned summarization scheme is insignificant. More comparative studies between IADDB and LADB will be found elsewhere [14].

CONCLUSION

LADB data has periodic measurements of the source data elements. The data is collected by colony under a strict protocol and includes environmental and husbandry factors. In addition to the immediate usefulness of this data, in the future it may be possible to measure the "strain drift" which is the slow change in animal characteristics arising from genetic noise and biased breeding selection. By collecting LADB-type data on colony foundation stock and comparing it to the LADB

TABLE 3. INFORMATION CONTENT COMPARISON DATA

Data Field	No. of Cases	Mean	Standard Deviation	95% Confidence Interval		Data Base
				Start	End	
Red Blood Cell Count	1000	7.100	0.743	7.054	7.145	IADB
		(7.108)	(0.761)	(7.061)	(7.156)	LADB
White Blood Cell Count	1008	10.115	3.117	9.922	10.308	IADB
		(10.214)	(3.076)	(10.022)	(10.406)	LADB
Hemoglobin	1006	16.682	1.808	16.570	16.794	IADB
		(16.790)	(1.847)	(16.675)	(16.904)	LADB
Serum Glutamil Oxaloacetic	868	27.586	9.854	25.930	28.243	IADB
		(27.786)	(10.035)	(27.125)	(28.467)	LADB
Total Protein	323	6.138	0.582	6.074	6.202	IADB
		(6.198)	(0.618)	(6.129)	(6.268)	LADB
Creatinine	859	0.819	0.159	0.808	0.830	IADB
		(0.833)	(0.166)	(0.822)	(0.844)	LADB

statistical norms for control animal populations drawn from the same colony, it should be possible to select foundation stock which will reinforce the probability of central tendency over time and to thereby limit genetic drift. This interactive process should also make it possible to statistically engineer the LADB data and ultimately to produce laboratory animals with certifiably stablized and predictable base lines suitable for particular lines of research in animal related research fields.

This paper has attempted to show that LADB involves considerable text processing and data manipulation. LADB illustrates how the generalized data management system called BASIS can be utilized to handle the textual and numeric information tasks. The easy to use LADB system could make a dramatic impact in the research fields of animal science. LADB will be even better if all the scientists and research organizations across the country will submit their own data to LADB and let it be shared by other scientists in the same research fields.

REFERENCES

1. Fried, J. B., "BASIS-70 Interface", Interactive Bibliographic Search: The User/Computer Interface., D. E. Walker, ed., Montvale, N.J. AFIPS Press, 1971. pp. 143-157.

2. "BASIS On-line Retrieval and Analysis System Manual", Information Systems Section, Battelle Columbus Laboratories, Columbus, Ohio, 1976.

3. Hsu, K., "What BASIS Can Do For Implementing a General Purpose Computerized Chinese/English Language Information Retrieval and Data Analysis System in Taiwan, Republic of China", Proceedings of International Computer Symposium, Vol. II, pp. 458-475, 1975.

4. Fite, D. W., Rankin, K., Fong, E., Walker, J. C., and Marron, B. A., "A Technical Index of Interactive Information Systems", Washington, D. C.: National Bureau of Standards, February, 1974 (NBS Technical Note 819).

5. Martin, T. H., "A Feature Analysis of Interactive Retrieval Systems", Stanford, California: A Report (Report No. SU-COMM-ICR-74-1) of the Institute for Communication Research, Stanford University, September, 1974.

6. Simon, R. C., and Kovacs, G. J., "The Historical Toxicology Information/Data System - HISTOX, Drug Information Journal, pp. 144-153, 1975.

7. Litterst, C. L., Mimnaugh, E. G., Gram, T. E., Guarino, A. M., Simon, R. C., "Acute Toxicity of Substrates of the Mixed Function Oxidase System in Normal and Phenobarbital-Pretreated Mice", Journal of Toxicology and Environmental Health, Vol. 1, pp. 39-46, 1975.

8. Levy, A. F., Simon, R. C., Beerman, T. H., Folk, R. M., "Scheduling of Toxicology Protocol Systems", Computers and Biomedical Research, Vol. 10, pp. 139-151, 1977.

9. Claydon, C. R. and Klette, I. J., "New Techniques for Weapons Systems Cost Analysis", Proceeding, Military Operations Research Society, June, 1974, pp. 1-5.

10. 小久保孝二, 小沢正恕, 上野 満, 神田雄一, 新谷 聡, 小林正彦, 岩泉洋一, 平松豊, "加工データ心力江関する調査研究", 技術時報, 技術研究時, 機械振興協会, 第10巻, 第6号, pp. 211-226, Nov. 1974.

11. 小久保孝二, 新谷 聡, "加工情報検索システムの調査・開発", 技研時報, 技術研究時, 機械振興協会, 第10巻, 第6号, pp. 227-242, Nov. 1974.

12. Krohn, R. E., Fish, A. R., Hsu, K., "Ohio College Library Center Subject Search", in progress.

13. "BASIS Data Base Construction and Maintenance", Information System Section, Battelle Columbus Laboratories, Columbus, Ohio, 1976.

14. Hsu, K., "Laboratory Animal Data Bank," Master's Thesis, Computer and Information Science Department, The Ohio State University, in progress.

15. "LADB Data Collection Manual," Information System Section, Battelle Columbus Laboratories, Columbus, Ohio, 1976.

16. Snedecor, G. W., Cochran, W. G., "Statistical Method", 6th Edition, The Iowa State University Press, Ames, Iowa, 1967.

17. Nie, N. H., Hull, C. H., Jenkins, J. G., Steinbrenner, K., Bent, D. H., "Statistical Package for the Social Science," 2nd Edition, McGraw-Hill Inc., 1975.

18. Fried, J. B., "BASIS On-line Retrieval and Analysis of Large Numeric Data Bases," Proceedings of Fifth Biennial International CODATA Conference, pp. 541-547, 1976.

19. "Student", Biometrika, 6:1, 1908.

APPENDIX A. EXAMPLE SEARCH (BASE-LINE FILE)

Question in Mind:

"What is the distribution of platelets data of 10 month old female beagle dogs?"

After LOGIN, the logical menu-selection procedures are as following:

1. The menu-selection activities* included in the first page of this example illustrates the search steps which make the data available to LADB.

2. The menu-selection activities* included in the second page of this example illustrate the search steps which select only the 10 month old female beagle dogs out of the 8 colony records.

3. The menu-selection activities* included in the third page of this example illustrate the search steps which select the "STATISTICS + HISTOGRAM" option of the "DISTRIBUTIONS" procedure to analyze the retrieved platelets data of 10 month old female beagle dogs.

4. The latter part of the third page, the fourth page and the upper part of the fifth page contain the statistical data and the frequency distribution of the selected data set. It is the standard output provided by the FREQUENCIES procedure of SPSS.

5. The menu-selection activities* included in the fifth page of this example illustrate the automatic LOGOUT procedure provided by LADB.

---

* All user entries are underlined in the search example.

```
CHOOSE BASIC LADB SERVICE BY NUMBER:
  1. SEARCH (BUILD A PROFILE)
  2. USE A PREVIOUSLY SAVED PROFILE
  3. SIGN OFF (STOP LADB)
     FOR HELP CALL (202)-785-8414
   / 1
CHOOSE FILE BY NUMBER:
  1. BASE-LINE FILES
  2. STRAIN DESCRIPTIONS
  3. HUSBANDRY DESCRIPTIONS (NOT AVAILABLE)
  4. PROTOCOL DESCRIPTIONS (NOT AVAILABLE)
  5. ANALYTICAL PROCEDURES DESCRIPTIONS (NOT AVAILABLE)
  6. RETURN TO "CHOOSE BASIC LADB SERVICE"
     FOR HELP CALL (202)-785-8414
   / 1
CHOOSE SPECIES BY NUMBER:
  1. DOG (BEAGLE ONLY)
  2. HAMSTER (SYRIAN GOLDEN ONLY)
  3. MONKEY (RHESUS ONLY)
  4. MOUSE
  5. RAT
  6. RETURN TO "CHOOSE FILE"
   / 1
CHOOSE CLASS(ES) OF BASE-LINE DATA BY NUMBER: (E.G. 1,4,5)
      ** INDIVIDUAL DESCRIPTORS **
  1. ENTER DATA ELEMENT ABBREVIATION(S) FROM LADB SERVICE CARD
      ** GROUPED DESCRIPTORS **
  2. COLONY ID
  3. PROTOCOL/HUSBANDRY FACTORS
      ** DATA DESCRIPTORS **
  4. HEMATOLOGY
  5. CLINICAL CHEMISTRY
  6. GROWTH AND LIFESPAN
  7. PATHOLOGY

  8. RETURN TO LAST CHOICE
   / 4
CHOOSE TYPE OF HEMATOLOGY PARAMETER BY NUMBER: (E.G. 1,2,3)
  1. NON-SEGMENTED NEUTROPHILS
  2. SEGMENTED NEUTROPHILS
  3. LYMPHOCYTES
  4. MONOCYTES
  5. EOSINOPHILS
  6. BASOPHILS
  7. HEMATOCRIT
  8. HEMAGLOBIN
  9. MEAN CORPUSCULAR HEMAGLOBIN
 10. MEAN CORPUSCULAR HEMAGLOBIN CONCENTRATION
 11. MEAN CORPUSCULAR VOLUME
 12. NUCLEATED RED BLOOD CELLS
 13. PLATLETS
 14. RED BLOOD CELL COUNT
 15. RETICULOCYTES
 16. WHITE BLOOD CELL COUNT
 17. RETURN TO "CHOOSE CLASS OF BASE-LINE DATA"
   / 13
```

```
YOU HAVE SELECTED DATA ELEMENTS OF INTEREST
CHOOSE YOUR NEXT STEP BY NUMBER:
  1. RUN IMMEDIATE SEARCH FOR RESULTS
  2. SAVE DATA ELEMENTS OF THIS PROFILE FOR FUTURE SEARCH
  3. RETURN TO "CHOOSE CLASS(ES) OF BASE-LINE DATA"
   / 1

YOUR SEARCH HAS PRODUCED THE FOLLOWING PRELIMINARY RESULT:
     8 COLONY RECORDS
CHOOSE YOUR NEXT STEP BY NUMBER:
  1. MAKE PROFILE MORE GENERAL/SPECIFIC
  2. SAVE PROFILE FOR FUTURE USAGE
  3. RETURN TO "CHOOSE BASIC LADB SERVICE" (REPORTS, STATISTICS, ETC.)
   / 3
CHOOSE BASIC LADB SERVICE BY NUMBER:
  1. SEARCH (BUILD A PROFILE)
  2. USE A PREVIOUSLY SAVED PROFILE
  3. SIGN OFF (STOP LADB)
     FOR HELP CALL (202)-785-8414
  4. BUILD A WORK SET (YOUR DATA)
  5. STATISTICS
  6. PRINT REPORTS
   / 5
DATA AVAILABLE FOR STATISTICAL ANALYSES ARE GROUPED
BY COMBINATIONS OF AGE (OR AGE GROUPINGS), TEST YEAR, AND SEX
FOR EACH DATA ELEMENT (E.G. HCT, BUN, WBC) WITHIN A COLONY.

THERE ARE 132  COMBINATIONS OF THESE IDENTIFIERS FOR YOUR SEARCH
DO YOU WANT TO SELECT GROUPS BY AGE, SEX, OR TEST YEAR?
ENTER YES OR NO
   / YES
FOR ** SEX **
CHOOSE 1-LOOK, 2-LIST, 3-LEAVE
   / 2

CHOOSE ITEM(S) FOR ** SEX **
  1. CHOOSE ALL
  2. FEMALE (66)
  3. MALE (66)
  4. CHOOSE NONE
END OF TERMS FOR ** SEX **
   / 2
FOR ** SEX **
CHOOSE 1-LOOK, 2-LIST, 3-LEAVE
   / 3
FOR ** AGE CATEGORY **
CHOOSE 1-LOOK, 2-LIST, 3-LEAVE
   / 2

CHOOSE ITEM(S) FOR ** AGE CATEGORY **
  1. CHOOSE ALL
  2. 03(  2.00 - 2.99 MO) (1)
  3. 04(  3.00 - 3.99 MO) (2)
  4. 05(  4.00 - 4.99 MO) (1)
  5. 06(  5.00 - 5.99 MO) (1)
  6. 07(  6.00 - 6.99 MO) (3)
  7. 08(  7.00 - 7.99 MO) (3)
  8. 09(  8.00 - 8.99 MO) (8)
  9. 10(  9.00 - 9.99 MO) (13)
 10. CHOOSE NONE
   / 9
```

```
MORE TERMS EXIST FOR ** AGE CATEGORY **
WOULD YOU LIKE TO SEE THEM?
ENTER YES OR NO
   / NO
FOR ** AGE CATEGORY **
CHOOSE 1-LOOK, 2-LIST, 3-LEAVE
   / 3
FOR ** OBSERVATION YEAR **
CHOOSE 1-LOOK, 2-LIST, 3-LEAVE
   / 3

THERE ARE 13   COMBINATIONS OF THESE IDENTIFIERS FOR YOUR SEARCH
DO YOU WANT TO SELECT GROUPS BY AGE, SEX, OR TEST YEAR?
ENTER YES OR NO
   / NO
CHOOSE ONE DATA ELEMENT BY NUMBER
  1. PLATELETS (10**3 PER CU MM)
  2. ENTER A DATA ELEMENT ABBREVIATION
  3. RETURN TO "SELECT DATA GROUPINGS"
  4. RETURN TO "CHOOSE BASIC LADB SERVICE"
   / 1
CHOOSE LADB STATISTICAL SERVICE BY NUMBER:
  1. DISTRIBUTIONS (PRINTED CURVE AND STATISTICS,ETC.)
  2. CROSSTABULATION (DATA ELEMENT VS SEX,AGE,ETC.)
  3. BREAKDOWN (COLONY(S) BY SEX,AGE,ETC.)
  4. T-TEST (COMPARE DATA ELEMENT BY SEX,AGE,ETC.) (NOT AVAILABLE)
  5. RETURN TO "SELECT DATA GROUPINGS"
  6. RETURN TO "CHOOSE BASIC LADB SERVICE"
   / 1
CHOOSE DISTRIBUTIONS OPTION BY NUMBER:
  1. STATISTICS
  2. TABLE(DETAILS)+STATISTICS
  3. STATISTICS+HISTOGRAM
  4. STATISTICS+TABLE(CONDENSED)+HISTOGRAM
   / 3
SPSS/ONLINE V4.0

- - - FREQUENCIES - - -

END OF FILE ON FILE INDATA
AFTER READING    123 CASES FROM SUBFILE NONAME

PLAT        PLATELETS IN 10**3 PER CU MM

MEAN         283.102      STD ERR       6.212      MEDIAN      272.169
MODE         406.744      STD DEV      82.652      VARIANCE   6831.297
KURTOSIS        .613      SKEWNESS       .719      RANGE       451.231
MINIMUM      116.000      MAXIMUM     567.231      SUM       50109.130
C.V. PCT      29.195      .95 C.I.    270.842            TO     295.363

VALID CASES      177      MISSING CASES     0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
- - - FREQUENCIES - - -

PLAT      PLATELETS IN 10**3 PER CU MM


     CODE
           I
  116.000 ** (      1)
           I
           I
  146.079 ******** (      7)
           I
           I
  176.159 ************** (     13)
           I
           I
  206.238 ********************** (     21)
           I
           I
  236.317 ************************** (     25)
           I
           I
  266.397 ********************************** (     34)
           I
           I
  296.476 ************************ (     22)
           I
           I
  326.555 ***************** (     15)
           I
           I
  356.635 *********** (     10)
           I
           I
  386.714 ************ (     11)
           I
           I
  416.793 ********* (      8)
           I
           I
  446.873 ******* (      6)
           I
           I
  476.952 ** (      1)
           I
           I
  507.031 ** (      1)
           I
           I
  537.110 * (      0)
           I
           I
  567.190 *** (      2)
           I
           I
           I.........I.........I.........I.........I.........I
           0         10        20        30        40        50
           FREQUENCY
```

```
VALID CASES     177      MISSING CASES     0

THESE STATISTICS ARE BASED ON NO MORE THAN
100   INDIVIDUAL ANIMALS.  THIS IS THE ACTUAL
NUMBER OF INDIVIDUAL ANIMALS IF ONLY ONE
AGE CATEGORY WAS CHOSEN.

=========================================================================

CHOOSE YOUR NEXT STEP BY NUMBER:
  1. SAME DATA ELEMENT WITH ANOTHER STATISTICAL PROCEDURE
  2. ANOTHER DATA ELEMENT
  3. RETURN TO "SELECT DATA GROUPINGS"
  4. RETURN TO "CHOOSE BASIC LADB SERVICE"
   / 4
CHOOSE BASIC LADB SERVICE BY NUMBER:
  1. SEARCH (BUILD A PROFILE)
  2. USE A PREVIOUSLY SAVED PROFILE
  3. SIGN OFF (STOP LADB)
     FOR HELP CALL (202)-785-8414
  4. BUILD A WORK SET (YOUR DATA)
  5. STATISTICS
  6. PRINT REPORTS
   / 3
YOU HAVE COMPLETED A LADB SERVICE.

IF YOU WOULD LIKE TO SUBMIT YOUR
DATA FOR LADB ACCEPTANCE, PLEASE CONTACT
(202)-785-8414 OR WRITE TO:
             LADB
             BATTELLE WASHINGTON OPERATIONS
             2030 M ST. N. W.
             WASHINGTON, D.C.  20036

CONNECT TIME 0 HRS.  7 MIN.

GOODBYE
```

THE USE OF TOTAL AT THE NETHERLANDS ENERGY RESEARCH FOUNDATION ECN

H.M. Rietveld, Netherlands Energy Research Foundation ECN, Petten (N.H.)

TOTAL data base management system.

The general data base management system TOTAL is implemented on the CDC 6600 computer at the ECN research centre at Petten (N.H.). It is a network structure data base which provides a direct linkage between files. There are two types of files: master and variable. A master file can be independent and its records can be accessed directly by respective control keys. A variable file is dependent and must be attached to a master file. Its records are chained in groups. Each of these records is, in turn, chained to a unique master record in a related master file. This chaining provides the access paths to the variable records. A master file can be linked to more than one variable file and a variable file to more than one master file. Thus multiple access paths can exist between multiple master files and a single variable file. This interlinkage of files permits the construction of a network structure data base. TOTAL lends itself well for the handling of problems involving data sets which can be differently interrelated. The first step is to define the contents of the records of the required master and variable files. Secondly, the links between these files should be established. These steps define the data base which can be accessed by means of special application programs. In Petten, all application programs are written in FORTRAN and the interface with the data base consists of a call to the subroutine DATBAS, with the proper parameters defining the required functions to be executed. It should be stated here that the use of this subroutine in FORTRAN programs is rather cumbersome. This is mainly due to the fact that FORTRAN does not have the character type variable, as does COBOL. This means that for each CALL DATBAS the required character string has to be newly constructed. Until now, in Petten, TOTAL has been used in two widely different applications of which a short description will be given.

Library automation.

The first is the use of TOTAL for library automation. An obvious reason to do this is to reduce the labour involved in the mainly routine type transactions common in a library, thereby enabling the staff to handle a larger volume of work. In view of the fact that the library budget is, as usually, limited, this exercise should not cost too much. The availability of a GDBMS helps to ease significantly the required programming effort. In effect, without it, the automation of a small library does not pay. The schematic of this integrated acquisition, cataloging and circulation control system is shown in fig. 1. The squares indicate master files with their appropriate names and control keys. Circles constitute variable files. The files can be described generally as follows:

PERS - master file containing records with names and other data of persons allowed to use the library. The control key consists of the person's registration number,

BOOK - master file containing short descriptions of books (or other items) on order. The book number is the control key,

SUPP - master file containing the names, addresses and other data on publishers and booksellers with their code number as control key,

BUDG - master file containing the budget amounts and spendings of the different departments in the institute. The budget number is also the control key,

CATL - master file with records containing a full bibliographic description of books present at the library. The book number is the control key,

LOAN - variable file with lending transactions,

RESN - variable file with reservations for books on loan,

ORDR - orders for books placed with the booksellers.

The application programs operating on this data base have already partly been written and enable one to control all the common operations in the library, thereby ensuring a better management and a more efficient use of the library. Almost all programs are written for interactive terminal use.

Fuel burn-up computation.

The initial fuel load of a nuclear reactor consists of a mixture of nuclides. When the reactor is operating this fuel is subjected to a neutron flux. As a consequence neutrons are absorbed by the nuclides leading to either fission or capture. In addition radioactive isotopes formed by these processes can decay to other nuclides. In nuldimensional calculations the spatially variable neutron flux is averaged over the reactor core.

This flux is then split up in a number of energy groups. For each energy group a set of appropriate cross-sections is supplied. The whole process of mutations in the nuclear fuel can be described in terms of depletion chains. These specify (1) how each nuclide is formed (radioactive decay or capture) from previous nuclides in the chain, (2) wether or not the nuclide is a direct product of the fission process, and (3) how the nuclide is destroyed (radioactive decay and/or absorption). The differential equations describing the burn-up processes in a given chain are of the following two types:

I $\frac{dN_i}{dt} = \bar{Y}_i + \alpha_i \beta_i N_{i-1} - (\lambda_i + A_i) N_i$

II $\frac{dN_i}{dt} = \bar{Y}_i$

$\bar{Y}_i = \sum_k y_{ik} S_k$ = total yield of nuclide i

y_{ik} = yield of nuclide i after fission of nuclide k

S_k = fission rate of nuclide k

N_i = atomic density of nuclide i in this chain

A_i = total absorption cross-section of nuclide i,

α_i = partition to parallel branches in a chain (isometric states of a nuclide). For first nuclide in a chain $\alpha_i = 0$, for all other normal cases $\alpha_i = 1$

λ_i = decay constant of nuclide i

$\alpha_i = \lambda_i$ or C_i

C_i = total capture cross-section of nuclide i

Analytical solutions to these differential equations can be found and are used in this program. The schematic of the data base is shown in fig. 2. The contents of the files are as follows:

ISOS - master file containing 1200 records. Each record consists of a control key equal to the isotope number plus a yield index number k and the energy in Joules released on fission. The isotope number is calculated according to Zx1000+A, where Z is the atomic number and A the mass number. When no fission is possible, k≠0.

KETS - master file of 180 chains. Each chain comprises a control key (=chain number), the number of isotopes in the chain and the type of burn-up process in the chain.

SETS - master file of 60 sets of chains. The set number is the control key. Each record further contains the number of chains in the set and the type of isotopes in the chain i.e. fission products only or fissionable nuclides. In each burn-up calculation normally one of each type of sets is used.

YIND - a stand alone master file containing 36 records with control key=yield index number k. The record contains also the corresponding isotope number. Only fission nuclides, i.e. isotopes with k=0 in file ISOS, are stored.

SKET - variable file consisting of 255 records, each with two control keys: a set number and a corresponding chain number.

DATA variable file consisting of 3000 records, each with two control keys: a chain number and a corresponding isotope number and further containing the following data for this isotope:
λ_i, α_i, y_{ik} (k=1, -----, 30), oc and cp.
oc indicates wether the nuclide has occurred before in another chain: cp indicates the coupling, capture or radioactive decay, to the parent nuclide. The subscript k in y_{ik} is the yield index number.

Two application programs in FORTRAN have been written for this data base. One calculates the atomic densities N_i for a certain neutron flux and after a certain time interval, in addition to other related quantities. The other is used to update an existing data base or create a new base.

Conclusion.

The use of the TOTAL data base management system has some definite advantages such as the ease of programming complicated interconnected file structures. It also proves to be efficient in core occupation, less than 2000_8 words, as well as execution time. Disadvantages are the cumbersome implementation of the subroutine DATBAS in FORTRAN application programs and the absence in the CDC implementation of such possibilities as a report writer, a data dictionary and sorting and merging routines. However, it can be said that, overall, TOTAL is a handsome system to implement a data base, especially with regard to its cost.

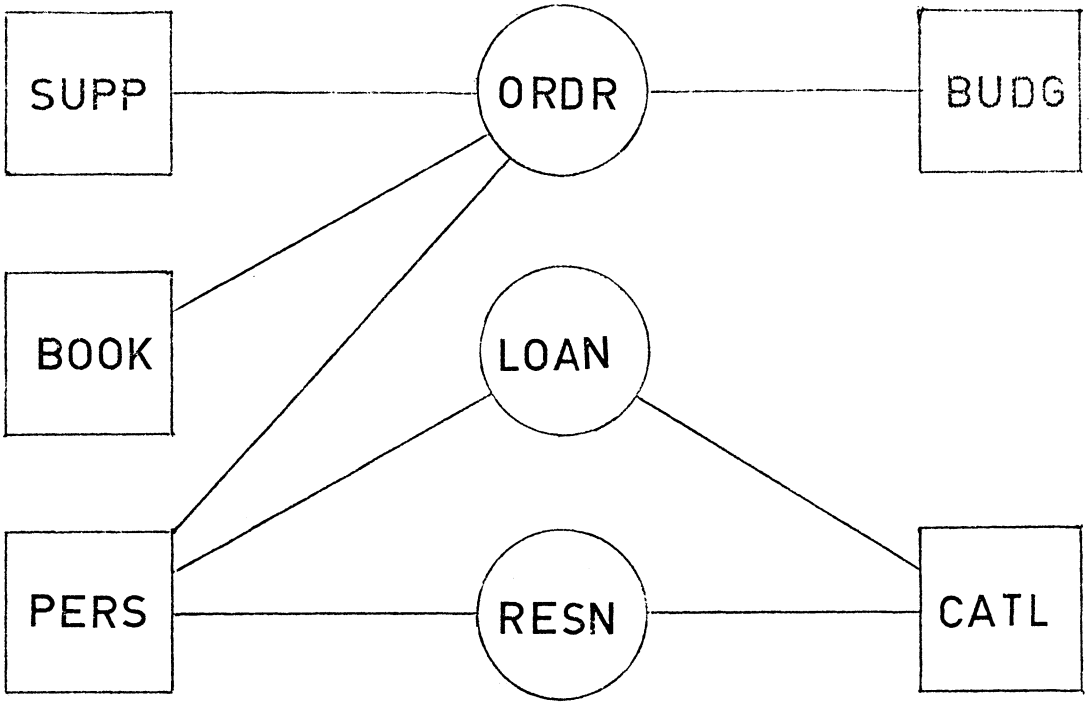


fig. 1. Schematic of an integrated library acquisition, cataloging and circulation control system.

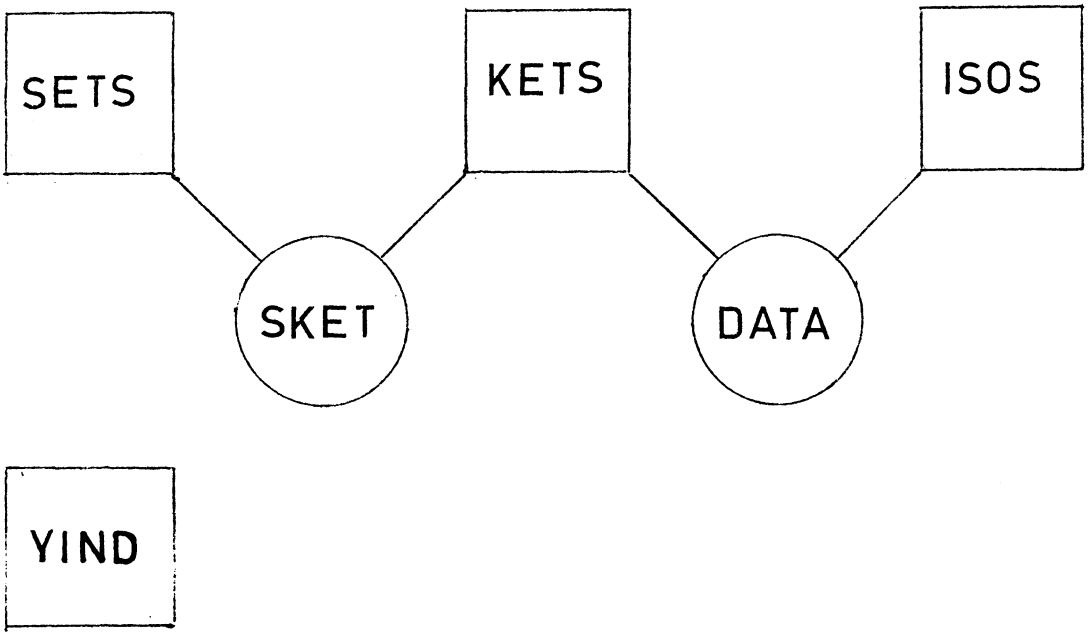


fig. 2. Schematic of a fuel burn-up computation system.

USE OF DBMS-10 FOR STORAGE AND RETRIEVAL OF* EVALUATED NUCLEAR DATA FILES

C. L. Dunford
National Nuclear Data Center
Brookhaven National Laboratory
Upton, New York 11973 U.S.A.

Abstract

The use of a data base management system (DBMS) for storage of, and retrieval from, the many scientific data bases maintained by the National Nuclear Data Center is currently being investigated. It would appear that a commercially available DBMS package would save the Center considerable money and manpower when adding new data files to our library and in the long-term maintenance of our current data files.

Current DBMS technology and experience with our internal DBMS system suggests an inherent inefficiency in processing large data networks where significant portions are accessed in a sequential manner. Such a file is the Evaluated Nuclear Data File (ENDF/B) which contains many large data tables, each one normally accessed in a sequential manner.

After gaining some experience and success in small applications of the commercially available DBMS package, DBMS-10, on the Center's DECsystem-10 computer, it was decided to select one of our large data bases as a test case before making a final decision on the implementation of DBMS-10 for all our data bases. The obvious approach is to utilize the DBMS to index a random access file. In this way one is able to increase the storage and retrieval efficiency at the one-time cost of additional programming effort.

*Research carried out under the auspices of the U.S. Energy Research and Development Administration under Contract No. EY-76-C-02-0016

I. Introduction

Brookhaven National Laboratory, Upton, New York operates on behalf of the Energy Research and Development Administration, a nuclear data center known as the National Nuclear Data Center. This center is charged with the responsibility for providing information on nuclear reactions and nuclear structure and decay in the area of low and intermediate energy physics. In order to fulfill this charter, the center is active in the compilation, storage, retrieval and dissemination of a wide variety of data types in all forms available through current technology.

Until 1976, the center was known as the National Neutron Cross Section Center with responsibilities limited to the area of low energy neutron physics. As such, the center maintained three separate data bases, one for bibliography, one for experimental data and one for evaluated data. Although small in number the individual data bases were large, ranging from 5 million words to 20 million words. Since 1969 the center has had a dedicated computer system to support its responsibilities. This computer is a DECsystem-10 with 30 million words of disk storage, 144K memory and auxiliary devices such as a paper plotter, card reader, and magnetic tape drives. A PDP-15 computer and interactive graphics facility is joined with this system. The computer is operated in time-sharing mode and is heavily used by the center's 20 professionals and 8 technical and clerical staff.

In the past the center staff has developed its own data storage and retrieval systems by essentially developing a data base management system for each application. These ranged from a fairly sophisticated system for the large, loosely formatted experimental data file to a very simple system for the rigidly structured evaluated data file. This ability to design and construct each data base was possible because each computerized data system was developed as the need arose and as the programming manpower was made available. But in 1976, the center's responsibility was increased when new areas of nuclear physics were added. With the additional responsibility came an additional four data bases, which increased our data base contents by about 300 percent.

It was quickly realized that we could not implement four new data bases and all the necessary processing programs with the modest staff increase allocated to the data base management function. Therefore a commercially available DBMS software package was purchased. After comparing the DBMS systems available for the DECsystem-10, DBMS-10 was purchased. It is a system which closely follows the CODASYL specifications. We have implemented DBMS-10, version 3, on our computer using FORTRAN as the host language. Our goal was to minimize the programming effort involved in developing storage and retrieval systems for our new data bases and to reduce the time required to make the new data bases operational.

II. Description of the Application

In order to explore the capabilities and limitations of DBMS with respect to our large scientific data bases several smaller applications have been developed. I will describe in this paper some of our experiences with DBMS-10. Our system for handling evaluated neutron data will be described in some detail. Interesting results from other applications will also be described.

The evaluated nuclear data file consists of small amounts of text and organizational data and large amounts of sequentially accessed data tables.

The information is rigidly organized and structured since the format of the data contained in the file was designed about 11 years ago to interface directly into large processing codes used in nuclear power reactor physics design programs. In Fig. 1, the original sequential organization of the data is illustrated. The core of its design is the section which contains a data table describing a particular reaction and function for a single material. In the simplest case this table consists of a maximum of 5000 numerical pairs (energy and cross section) ordered by increasing energy.

Our old file management system used a random access technique to access a material. No processing programs were directly linked to the random access file. An intermediate retrieval step to produce a sequential file was required. At the present time the evaluated data file is undergoing major revisions. Therefore it was decided to build a DBMS which could automatically handle much of the bookkeeping operations, provide access to the fundamental unit of information and be interfaced with most of the processing programs easily.

III. System Design

One of the crucial problems to be resolved before implementing a DBMS to handle the evaluated data base was developing a strategy to provide for efficient storage and retrieval of the large data tables. We believe that the present implementation of DBMS is prohibitively expensive if used for storage and retrieval of a large number of records retrieved in a sequential manner. Therefore, at the expense of some additional programming effort, it was decided to use DBMS for the indexing and associated bookkeeping while the data itself is stored using random access data files.

The data base management programs were designed, programmed, and debugged with about a 3 man-week effort. The data base is currently accessed by three programs: a file update program, a retrieval program, and an index program. All are designed to operate in an interactive mode using one of the terminals with access to our DECsystem-10. Processing programs are now being interfaced to the new system.

The schema used is illustrated in Fig. 2. You will note that we have made frequent use of SORTED sets. This is done in order to quickly produce output in a useful order. Such a decision is not without cost but we have found that the penalty is acceptable as long as the number of records in a set occurrence is not large (up to 100 records). Most retrievals can then be simply done by following a set pointer chain. The next version of the DEC FORTRAN system will contain a fast sort routine. We will then investigate the possible advantages of not using SORTED sets. Retrievals would be done by retrieving the keys to records satisfying the retrieval criteria along with any information required by the sorting specifications. These would be sorted by the FORTAN sort routine and then the retrieval done by sequentially processing the sorted key file.

Some records have duplicate information stored in them. Three records, the MAT record, the Z-A record, and the MAT-SECTION record contain the variable NLIB, the library number and MAT, the material number. This kind of duplication eliminates the need to do multiple lookups following linking relationships to construct a single output record. We have found that a schema employing many record types of short length is very inefficient for our retrieval purposes.

Sets and their relationship were created to handle the majority of the envisioned retrievals. The schema is designed to provide access to the file on three of the four levels illustrated in Fig. 1. Only access at the "file" level has been elimated. By entering at the MAT record level we can retrieve the data for an entire material, while entrance at the TAPE record level enables us to retrieve an entire tape. Since dual access keys are not permitted we have added Z-A records to the file. These records permit accessing a meterial by its alternate identification, namely its nuclear charge number (Z) and its mass number (A). Normally a material is identified by its material number (MAT).

Since external random access files are used to store the data we have included records to describe these external files. This record gives the file name, the number of records used, and a pointer to the next available record. MAT records are linked to occurrences of DA-FILE records in order to permit the system to open the proper random access file for retrieval.

IV. Special Program Features

ENDLIB is the library generation and maintenance program whose main function is to take an input sequential file and replace or add material sections one at a time in the data base. The DBMS index is updated and the data tables stored in a random access file. All sections for a material are stored in the same random access file. Several small materials may be clustered in the same random access file while large materials are stored in their own file. The link between the DBMS index and a material section is the random access record number of the first record in the section's data table.

When a section is replaced, the new version of the section is stored at the end of the appropriate random access file and the record pointer updated in the DBMS index. With sufficient updating activity, the record utilization efficiency in the random access file is greatly diminished. When this efficiency falls below 70 percent, the file is copied over thus automatically eliminating unused records.

ENDRET is the library retrieval program. Retrievals can be done by specifying a TAPE thereby retrieving all materials assigned to that TAPE. Retrievals can also be done by material (giving either its material number or its nuclear identification (Z,A)) or by material section. The program automatically expands the compacted storage format into the standard file format.

INDED is a program designed to permit various miscellaneous operations on the DBMS index. Two elementary, but vital functions are modules to dump the contents of the DBMS file in a fixed format sequential file and another module to recreate the DBMS file from this sequential file. In this way backup and cleanup functions are easily accomplished.

A module exists which permits the user to modify the DBMS file contents. Current implementations include creation and modification of TAPES, reassignment of material numbers, and revision of section or material status codes. A last module allows one to print on the terminal the contents of any tape or material.

V. Other Operating Experiences

In operating other DBMS applications we have noticed that data base loading times would fluctuate widely while the actual CPU time required remained relatively constant. One significant factor was the configu-

ration of our disk storage and the job mix in the machine. All six 5 million word disk drives are on one channel. Heavy IO activity from other jobs often dramatically increased loading times. Another factor was competition for positioning the disk head. Load times were decreased when the DBMS file was on a private disk structure with little or no head position competition as opposed to a public disk structure where program swapping as well as other user input/output were causing competition for head positioning.

Load times are also heavily affected by whether the data are to be loaded into a file in some specific sort order. However, we did find that presorting the input file did minimize both the number of page retrievals and the elapsed time. We would assume that the sorting algorithm used in DBMS-10 can take advantage of the fact that an input file is presorted. Presorting seems to be a good procedure whenever a record is sorted in only one order with the DBMS file. Savings should also result if more than one sort order is required, through a judicious selection of one of the sort orders for presorting.

The DEC FORTRAN-10 system has extremely good file manipulation capabilities which may not be available on other FORTRAN systems. Such capabilities are essential for selecting and accessing the different random access files required in the above application in an interactive mode. In another application it was desirable to have many small DBMS data bases which could be processed by a single computer program, where the proper file for processing is to be selected interactively. The current version of DBMS-10 requires that the DBMS file have a name predetermined by the schema. It is very inconvenient to have to regenerate the schema for each different file to be accessed. This obstacle was overcome by using the RENAME feature of the file closing statement (CLOSE). The selected file is renamed to name.DBS before being opened by the DBMS system and restored to its original name after the DBMS CLOSE.

VI. Summary

We have investigated the use of a commercially available DBMS, DBMS-10 for use in constructing and maintaining scientific data bases. Two applications have been implemented with relative ease and are now in full production use. The additional storage and operation overhead has been within acceptable bounds in relation to the savings in the cost of development and maintenance for the data management system. In the next phase of development we intend to implement two large scientific data bases based on the results of the experiments described in this paper.

REFERENCES

1. D. Garber et al., "Data Formats and Procedures for the Evaluated Nuclear Data File, ENDF", BNL-NCS-50496, Oct. 1975

2. "Data Base Management System (DBMS-10) Administrator's Procedures Manual", DEC-10-AAPMA-B-D, Digital Equipment Corporation, Maynard, Mass., May 1975

3. "Data Base Management System (DBMS-10) Programmer's Procedures Manual", DEC-10-APPMA-B-D, Digital Equipment Corporation, Maynard, Mass. May 1975

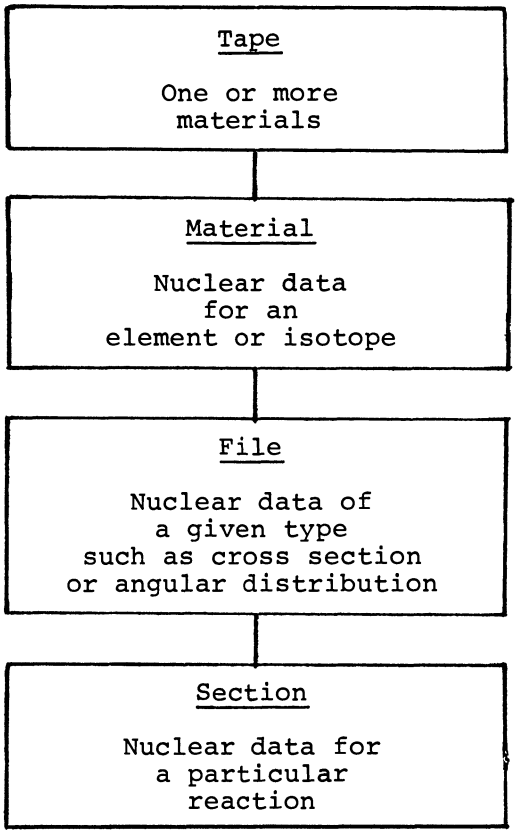


Fig. 1. Structure of ENDF (Evaluated Nuclear Data File)

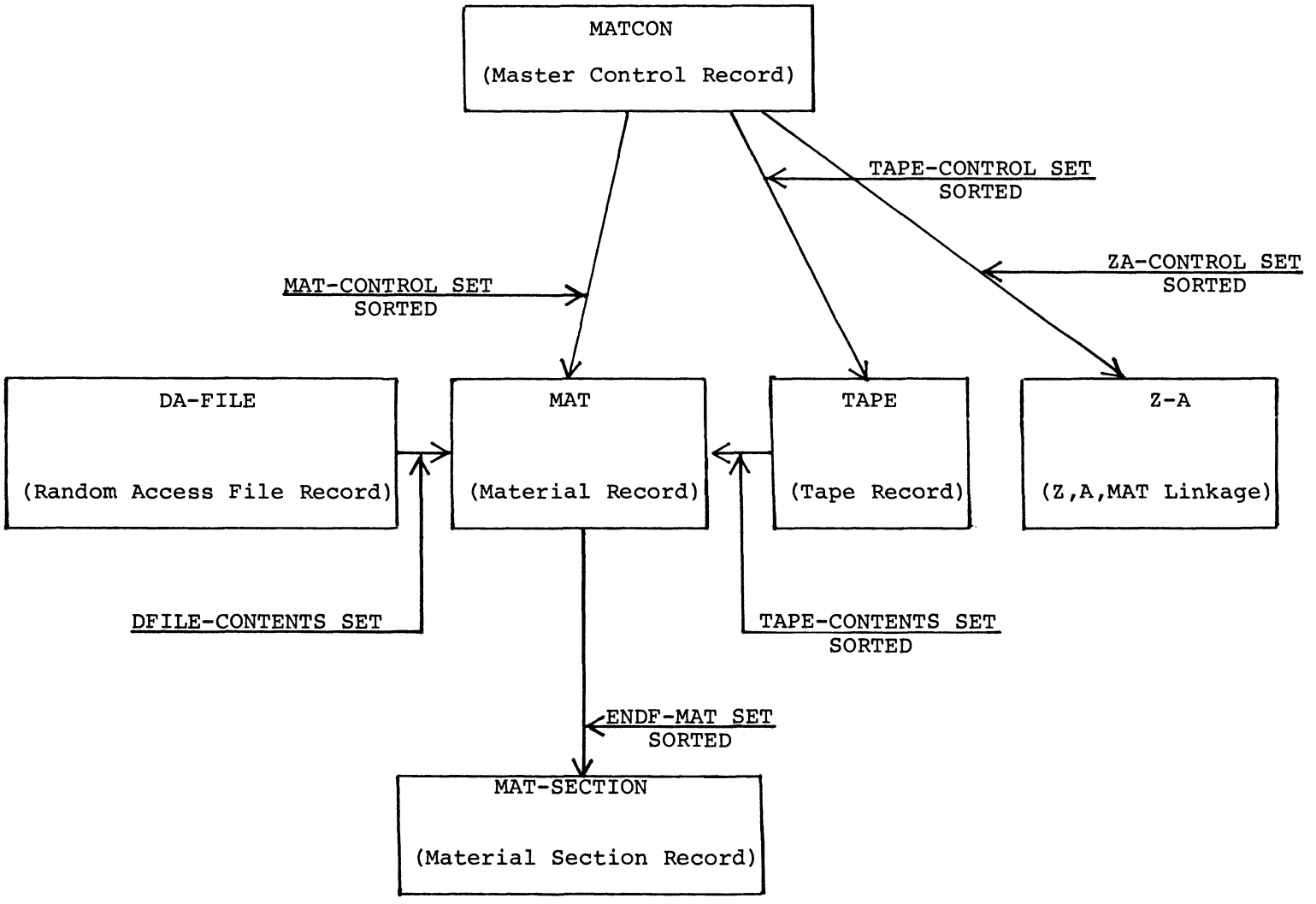


Fig. 2. Schema for DBMS system for ENDF (Evaluated Nuclear Data File)

A LARGE DATA BASE ON A SMALL COMPUTER

Neutron Physics data and bibliography under IDMS

A. Schofield
L. Pellegrino
OECD/NEA Neutron Data Compilation Centre, Saclay, France

N. Tubbs
OECD Nuclear Energy Agency, Paris

1. Introduction

The OECD/NEA Neutron Data Compilation Centre (CCDN) works with three other regional data centres to provide world coverage for the compilation and distribution to users of numerical data and other information on neutron-induced nuclear reactions. The 'four-centre network' and the characteristics of the data files exchanged within it are discussed in detail in the paper by Mrs. Attree of IAEA elsewhere in this report [1]. In respect of GDMS use in these centres, it is important that their four major information projects are implicitly (if not explicitly) linked, in that these files refer to successive stages in the elaboration of neutron cross-section data and leading up to the presentation of evaluated files of numerical data in standard format: 'best values' for use in reactor computations.

The work reported in this paper concerns the transfer of three associated files to an IDMS data base: the CINDA bibliographic index to neutron physics publications (now 140,000 records or some 17 Mbytes), the cumulated EXFOR exchange tapes used for maintaining parallel data collections at all four centres (2 million records, or 35 Mbytes when packed on disc) and the CCDN's internal data storage and retrieval system NEUDADA (2.65 million records, 54 Mbytes packed). With associated dictionaries and inter-file conversion tables the corresponding IDMS data base will be about 160 Mbytes. The main characteristics of the three files are shown in Section III, 1 below.

2. The Decision to Use a GDMS

It is proposed to replace the two NEA data centres, CCDN and the Computer Program Library (CPL) located in Italy, by a single NEA Data Bank in Saclay. In evaluating the computer requirements of the Data Bank a conflict became apparent between the benefits in speed and convenience of replacing the CCDN's IBM 370/125 computer by remote links to an IBM 370/168 (in any case essential for the program testing work of the CPL) and the high overhead cost of disc storage for the neutron

physics data. It was decided to overcome this difficulty by installing a 'heavy mini-computer' in the Data Bank for use as a remote job entry station to the IBM 370/168 and other large computers on the Saclay site, and as a data base carrier. The equipment provisionally chosen is a PDP 11/70 with some 350 Mbytes of disc storage: however, there are other heavy minis on the market which might be expected to perform well in the same role.

The transfer of CCDN files to a GDMS had been under discussion for some time. The Centre had over the years produced its own data handling systems for CINDA and NEUDADA using the Indexed Sequential Access Method (ISAM), but was conscious of their limitations to the extent that the question of using GDMS was being discussed, but for a future time several years ahead. The decision to change computers precipitated the choice: by transferring control of the contents of the CCDN files from the home-made programs to a GDMS on the 370/125, and later 'lifting off' the data base on to replacement hardware but using the same GDMS, only the 'physics-dependent' programs need be rewritten for the new computer thus shortening the changeover. Besides its reputedly good performance in disc input/output, the PDP 11/70 offered a choice of IDMS and TOTAL data management systems, both compatible with the 370/125. IDMS was preferred because it was adopted by DEC for the PDP 11, and because it can represent directly hierarchical as well as network data structures.

3. Structuring the CCDN Data Base

1. Inherent and system-dependent structure of the data at CCDN

It now seems most convenient in handling numerical neutron data to consider as the basic unit of information the measurement of a cross-section for one element or isotope, in a particular laboratory. One measurement will generate a number of data points (as incident neutron energy is varied, for example) which will be grouped in a data table for storage purposes. Several authors will work in a group to do a number of similar measurements, usually reported in several progress reports and a journal or conference paper. The unit of data compilation is on the other hand the paper or report through which the compiler learns about this work, and which usually gives results for several measurements. Each one of these measurements may have a different history of measurement and extended remeasurement over the several years' lifetime of that particular experimental set-up. These issues have been clarified only slowly, and largely in connection with successive developments of CINDA [2], the CINDA-based IAEA data index DASTAR and the design of EXFOR.

An unfortunate corollary is that some criteria essential in uniquely defining a given measurement are coded differently in each file, to the extent that conversion between files may require more information than is explicitly available in either, and which must be generated indirectly from other 'clues' in the file, or added manually, usually as tables to be consulted by conversion programs. It is this apparent incompatibility between historically different 'views' of the same information which has caused much difficulty in separate but co-ordinated operation of the three files, and is responsible for much of the complexity of the schema and the data base loading programs now being written. Examples of incompatibility are:

- Different nomenclature in each file for nuclear reaction cross-sections. More or less aggregation of similar cross-sections into one 'Quantity' has created a many-to-many correspondence

between 'quantities' in CINDA and NEUDADA files.

- Bibliographic references are coded differently in each file.

- Manual intervention may be needed to steer conversion of EXFOR data tables (a 'compiler view' of the data) to produce the 'FORTRAN view' required for input to customer programs and represented by NEUDADA 'Calculation' output. This steerage information will be generated once-for-all and stored in the data base as conversion tables or set linkages.

Data is identified within each of the three files by a mixture of externally significant criteria (target nuclide, laboratory of measurement) and arbitrary, system-dependent data items such as 'work no.' (EXFOR) or 'Experiment block no.' (CINDA). Different subsets of these quantities may be sufficient uniquely to define a measurement, and it is in fact defined differently in each one of the present files, which then in turn reflect this definition in the design of their storage programs at CCDN:

CINDA	ISAM key items: (Most significant first)	Laboratory, Nuclide, Quantity, Experiment block no., serial no.
	Non-key items:	Neutron energy, Bibliographic Reference, Type of work, Coder, date of Entry, cross-reference to EXFOR work/subwork no.
NEUDADA	ISAM key items:	Nuclide, Reference, Tag (arbitrary measurement identifier)
	Inverted indices on:	Laboratory, Reference, Quantity
	Non-key items:	Neutron energy
EXFOR	Direct access:	Work no., subwork no.
	Sequential scan of CCDN index:	Laboratory, Reference, Nuclide, Quantity

The data items underlined in the table above are externally significant criteria frequently specified in searching the present files, and design of the schema must allow good access both to these and to such system-dependent criteria as it is necessary to keep because they are frozen into externally agreed formats (EXFOR work no.) and/or the existing data files to be transferred to the data base.

2. Construction of the initial "SCHEMA"

One of the authors (A.S.) spent some two months analysing the files and data handling programs which make up CCDN's current investment in the three interrelated projects described in the introduction, and a further two weeks working together with a consultant from SEMA Informatique (the French agents for IDMS) to draw up the data base schema (the current version is shown in Fig. 1). The notation is explained in ref. [3], while the operational context of the data base can be seen in Fig. 2, which will be easier to follow after reading the explanation of international neutron data activities in ref. [1]. The dotted line shows the limit of data base working: most operations outside it will be run on the IBM 370/168 computer for which the PDP 11/70 will serve as a remote batch entry station.

3. Data base loading strategy

In one calendar year since IDMS was installed on the IBM 370/125, some two man-years of effort have been put into preparing the data, defining the data base, and writing loading programs. Of this time, about half has been devoted to 'upgrading' (eliminating inconsistencies and completing) the data files. The loading strategy shown in Fig. 3 is that which would leave fewest set pointers unresolved during loading due to the absence of set owners to whom members must be linked. It can in fact be broken down into sections which can be loaded in any order and linked by the appropriate sets afterwards at relatively small extra cost.

A fast load utility will soon be released for IDMS, at least on the 370/125; it is not yet announced for the PDP 11/70. This utility promises important gains in loading performance, but cannot absolve the user of the need to prepare his data for loading, not only by cleaning it but also by supplying preformatted input data containing all the information needed to resolve set linkages. As far as possible, CCDN's homemade loading programs have been written so that they can be transformed into updating programs for future use. As the fast load utility is at present, the whole data base must be loaded at once. The size of some of the intermediate data base workfiles to be sorted during this operation, and the difficulty with a small team in getting all the information together on time for loading, may in any case make the more gradual approach of Fig. 3 preferable for the initial loading process.

4. Data Base Performance at the CCDN

The decision to 'go data base' at CCDN was based in general on the high-level advantages of GDMS and brought forward by the choice of a PDP 11/70 as data base carrier for the NEA Data Bank. The question of physical performance had been considered only in very general terms: after all, the programs in use on the 370/125 were basically those transferred from an earlier 360/30 computer used on a one-shift rental agreement. Surely the 11/70, designed to optimise disc input/output, and well spoken of by the users we contacted, would perform at least as well as a 360/30? In view of the low level of effort available for preliminary work on the data base, pending approval of the Data Bank proposal, CCDN preferred to start work directly on preparing real neutron data to load into the schema of Fig. 1.

Then the troubles began. Measurement on a small number of CINDA records (corresponding data items are dispersed over the left-hand part of the schema) yielded a retrieval time of about 6 secs/record from the first version of the schema, or 200 hours to reconstitute the CINDA file from the data base. The whole file must be read at least twice a year when the CINDA bibliography is issued in book form. Loading times for numerical data into the TABDAT record type (three-quarters of the way across the schema, to the right) were 2.5 seconds per data point, or of the order of 1000 hours to load the EXFOR data tables point by point.

Solutions could be found to both these problems with some help from SEMA: in the original schema all data redundancy had been eliminated from the CINDA subschema, but in order to reconstitute the CINDA record it was necessary to follow three successive sets from, say, the ZAQCIN entry point, then to complete the skeleton record CINDA thus recovered by adding the key information first from ZAQCIN and then by following an 'owner' pointer back to the corresponding DICLAB entry. This section of the schema has been modified and in particular all information needed to produce the tape used in printing the CINDA book is now available at the

lowest hierarchical level in the CINDA record. Data points were re-aggregated into variable length records representing either a data table or a 2Kbyte block of data points, whichever is shorter. These data blocks require about the same loading time as a single data point.

Having learned the hard way that complex structure is expensive in data base performance, we tried and failed to get 'typical' performance figures for the IDMS operations, sufficient to allow us to calculate the order of magnitude of running times for different CCDN applications, in particular on the PDP 11/70 as DBMS-11. The manufacturers could give us only global performance targets. We spoke to some IDMS users who found performance 'adequate' but had not made measurements. SEMA was therefore commissioned to load a reasonable amount of data into their standard "Hoes, rakes and shovels/customer, salesman, order" IDMS test data base (Fig. 4), and to help run benchmark performance tests on an 11/70 and on CCDN's 370/125. In parallel, work continues in preparing data and loading programs for the CCDN data base, in order to test and adjust our own schema, and to show before a final decision is taken on the hardware and software for the Data Bank that CCDN operations can be adequately carried on a GDMS.

Table I shows some of the results of the benchmark measurements. Table II shows some comparative execution times for current CCDN programs, compared with results extrapolated from measurements on a partially loaded data base. These figures should be seen only for what they are: an attempt to demonstrate that GDMS performance will be at least acceptable for CCDN operations, both now on the 370/125 and later in the Data Bank using an 11/70 as data base carrier. The 11/70 performs all data base operations tested faster than the 370/125, with the exception of 'sequential read within area' for which the two computers perform equally well. We deduce from this that if a CCDN data base can be made to run acceptably on the 370/125, performance will also be acceptable on the 11/70. Data base loading on the 370/125 can be speeded up by running the computer in mono-user mode, without spooling.

5. Conclusions

The lessons we have learned, and our tentative conclusions after a year's work in preparation for loading a 160 Mbyte complex data base on a small computer are:

- There is an inevitable conflict between the desire to simplify the structure of an integrated data base by redesigning working methods around it, and the need to preserve compatibility with data suppliers and users outside the CCDN. The three different views (CINDA, EXFOR and NEUDADA) of what is eventually the same physics data are frozen into such links and cannot be changed in the short term.

  We can identify a paradox here. Data centres with a large stock of data (and experience in handling it) are very likely to be faced with historical problems of this kind when installing a data base. New data compilations will in time surely acquire similar incoherences of structure as they expand to include data not foreseen in the original system design, whether as a result of errors due to lack of knowledge of the data, or as a result of changed circumstances. Either way, data base structures in real data centres will tend to be more complicated than the theoretical optimum.

- Performance tests at CCDN have shown a very strong dependence of running times on the detailed design of the IDMS schema: estimates

of performance have changed by two orders of magnitude in a year's development work. Our second paradox is this: almost no detailed information on data base performance is publicly available on which to base a preliminary data base design, although errors in design can degrade performance sufficiently to make use of the data base infeasible.

- It is too early to talk of 'tuning' the CCDN data base. However, we have already found it necessary to reduce the number of sets scanned in CINDA retrieval, and to provide for sequential scanning of the whole or large parts of this 'file' by reintroducing (controlled) redundancy into the bibliographic part of the schema. As an extension of this, the use of 'sorted sets' will be replaced by external utility sorting of data retrievals.

- The 'typical' data base does not exist (perhaps one reason why manufacturers are reluctant to state performance figures), but these CCDN applications may given an idea of the size and type of data base which can be mounted on a small computer and still give acceptable performance. Data base loading times are now thought likely to give the most difficulty, as much because all linked files are to be loaded in a single series of operations as because of longer running times compared to current data files. Retrieval performance is expected overall to be comparable with current programs.

- As much effort may be absorbed by upgrading existing data files in preparation for loading as in definition and programming work on the data base.

References

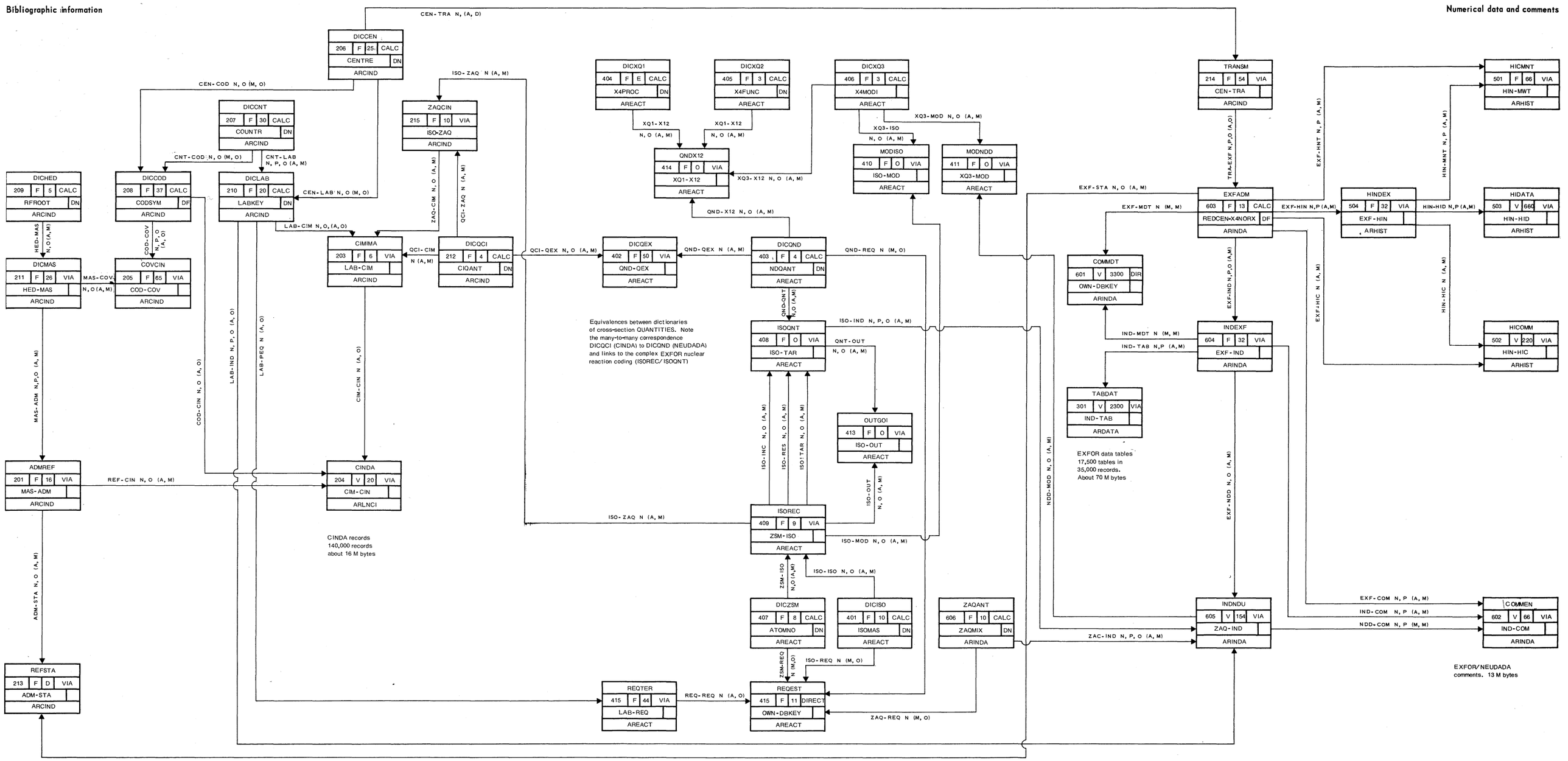
1. "Problems of a nuclear data centre in an international network" Ms. P. Attree, IAEA Nuclear Data Section: in this report.

2. "The CINDA neutron data index: an illustration of complementarity between mission-oriented and specialised information systems" N. Tubbs, OECD/NEA, in proceedings of the IAEA Symposium "Information Systems: their interconnection and compatibility" (Varna, Bulgaria 1974, IAEA-SM-89 paper 44).

3. "IDMS: Concepts and Facilities", Section 4. Cullinane Corp., 20 William St., Wellesley, Mass. 02181, U.S.A.

Figure 1

CCDN DATA BASE SCHEMA AS OF NOVEMBER 1977



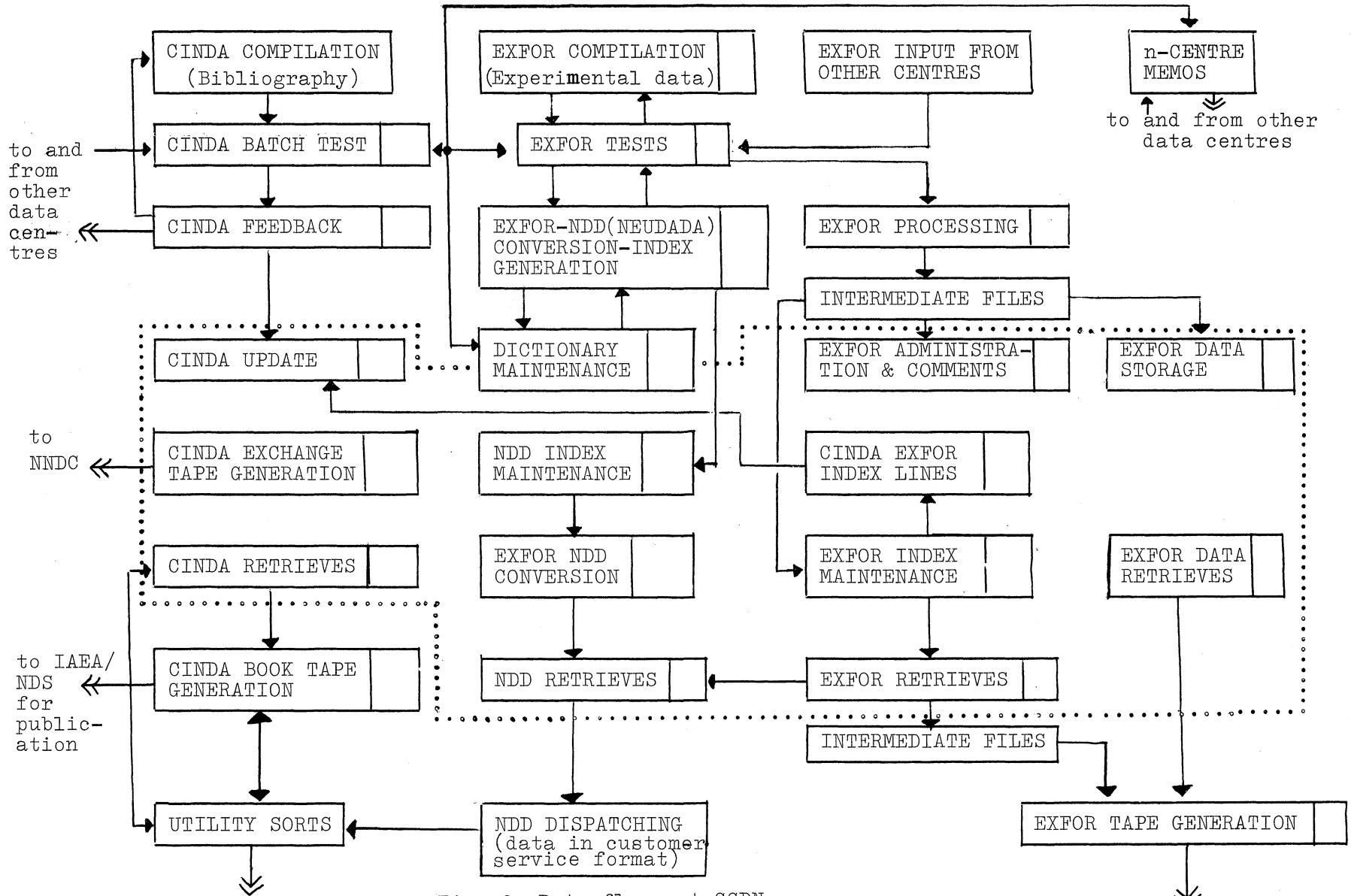


Fig. 2. Data flows at CCDN

The dotted boundary separates operations using the data base from others.

1. LOADING OPERATIONS

C	CINDA
QCN	NEUDADA - CINDA QUANTITY CORRESPONDENCE
IX	EXFOR INDEX
COV	CINDA COVERAGE FILE
XCO	EXFOR COMMENTS
DA	EXFOR DATA TABLES
RW	CINDA REFERENCE - EXFOR WORK CORRESPONDENCE
IN	NEUDADA-INFOR INDEX
NCØ	NEUDADA COMMENTS
QX	EXFOR ISO-QUANT FIELDS
IQ	REACTION ISOTOPE-QUANTITY CORRESPONDENCE
R	REQUEST FILE

2. LOADING STRATEGY

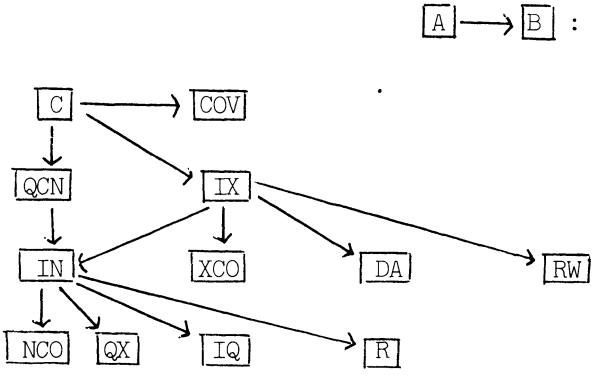


Fig. 3. Loading strategy for the CCDN data base

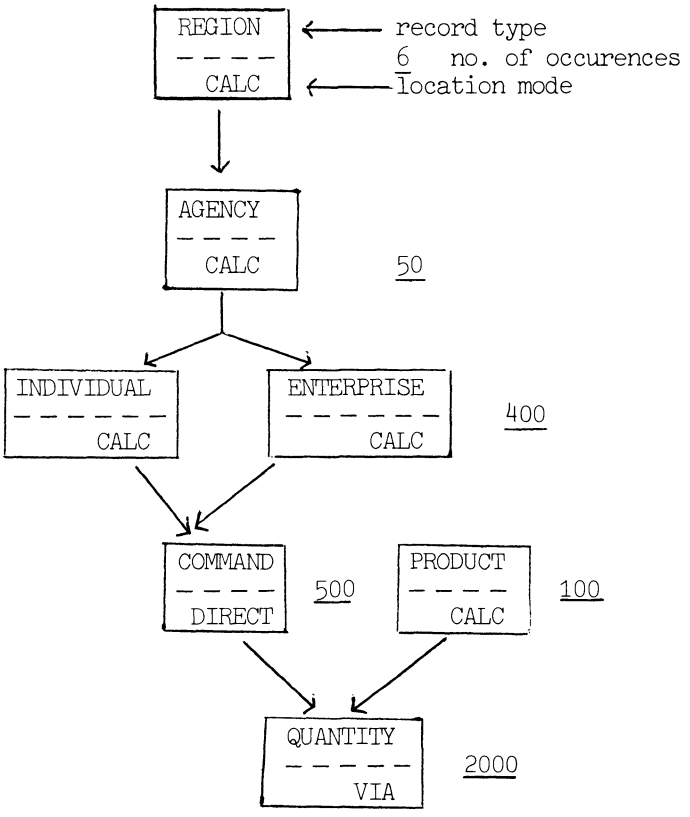


Fig. 4. Test data base for performance benchmark

Table I

Performance tests with a 3000 record data base /Fig. 47

	IBM 370/125 128 Kb (virtual storage)		PDP 11/70 256 Kb (64 Kb work spaces)	
Storage times	**Elapsed**	**CPU**	**Elapsed**	**CPU**
Data base loading	53mn 15s	8mn 43s	6mn	0mn 34s
Store isolated CALC record	36s		0.06s	
Store CALC record in set (e.g. AGENCY)	1s.74		0.10s	
Store VIA record with CALC owner	1s		0.06s	
Store VIA record with 2 CALC owners	1s		0.10s	
Retrieval times	**Elapsed**	**CPU**	**Elapsed**	**CPU**
Sequential scan of CALC records, follow through two sets: (e.g. INDIVIDU COMMAND QUANTITY)	7mn 21s 5 transactions/sec	3mn 11s	5mn 7 transactions/sec	45.34s
Find CALC record from symbolic key (e.g. PRODUCT)	25s 4 trans/sec	6s	15s 6.6 trans/sec	0.46s
CALC record by key then VIA record (e.g. PRODUCT QUANTITY)	49s 5 trans/sec	9s	21s 11.7 trans/sec	4s
CALC record by key then follow 2 sets (e.g. INDIVIDU COMMAND QUANTITY)	52s 15.6 trans/sec	19s	41s 19 trans/sec	7s
Extract a given record type by sequential sweep of area (e.g. QUANTITY)	1mn 32 trans/sec	38s	1mn 10s 28 trans/sec	12.20s

During these tests, both computers ran with a 'single-user' but with the spooling system in. The very large disparity in loading performance between the 370/125 and the 11/70 may be reduced if the spooling system is not leaded on the 370/125.

Table II

IBM 370/125: Estimates by extrapolation of DBMS performance, compared to current programs

Data base loading

1. EXFOR data sets, blocked up to 100 data points/DB record.

   'TABDAT' 2 m data points, 50/block, 2.5 secs/block ± 30 hrs

   (spooling on)

   Sort and load NEUDADA ISAM file, generate inverted indices

   (2.65 m records, packed and heavily blocked ± 15 hrs)

2. CINDA records, stored and set linked singly to retrieval keys

   'CINDA' 150,000 records plus retrieval keys ± 24 hrs

   (spooling off)

   Sort and load CINDA ISAM file ± 1.2 hrs

Data base interrogation (spooling on)

3. Typical experimental data search yielding 10,000 data points

   Data base (~1 block/sec via index, 50 secs/block) ± 3 mins

   NEUDADA (50 records/sec, using inverted indices) ± 3 mins

4. CINDA retrieval of 500 entries

   Data base (access from CALC keys VIA 2 sets) ± 8 mins

   CINDA ISAM file: Laboratory/Country specified (60%) ± 1 min

   Others (sequential scan ~40%) ± 30 mins

5. Sequential scan of CINDA file to produce book printing tape

   Data base (scan DB area, write in book format) (spooling off) ± 1.75 hrs

   CINDA ISAM file (write in book format) ± 45 mins

DATABANK FOR THE PROTOTYPE FAST REACTOR

K R Montgomery
United Kingdom Atomic Energy Authority
England

Introduction

This databank is a large set of files controlled by a management suite of COBOL data handling programs. The data, data dictionary and data directory are stored on exchangeable disc packs in the Random Access Index Sequential mode on an ICL 4/72 computer at the UKAEA headquarters at Risley.

It is a databank for the UKAEA Prototype Fast Reactor which stores data pertaining to the entire composition of the in-core components, concentrating particularly upon the fissile fuelled driver assemblies and the many experimental assemblies of both fuel pin and cladding/structural materials.

At this time, the reactor is not yet operating at full power. This should be achieved in 1977 and the databank will then contain reactor operating history and reactor physics data, and also post-irradiation examination data for the discharged units.

The databank will then represent a complete data compilation of in-reactor performance of fully documented materials. The current size of the databank is of the order of 100 megabytes and is expected to increase to approximately 200 megabytes over the next 2 years.

The ICL 4/72 computer will be phased out during 1977 and replaced by an ICL 2980 computer. The Databank Management System adopted by ICL is IDMS (Integrated Database Management System), and it will be available on the 2980 computer. Initially, the databank will be moved on to the 2980 in its current form, but IDMS potential will be probed by moving selected sub-sets of the databank into this management system. We hope that eventually IDMS will enhance the databank from batch mode to transaction mode processing. The expected growth of the databank will increase data retrieval and file reorganisation times, and it is hoped that IDMS may ease this problem.

The Filing System

There are 150 separate files. Each file contains fixed length records of data items for a common entity. For example, the Steel Ingot file contains chemical analysis records for each of the 1000 steel ingots used by the project.

The records consist of an entity key (eg steel ingot number) and a set of individual data items (eg analysis data) common to the entity keys. The records may also contain lists of data items that are entity keys for related records in other files (eg lists of steel turbine batches manufactured from the steel ingot of that record). These are regarded as forward pointers. Sub-key items in the records are also present as backward pointers (see section "File Navigation").

The entity key occupies the first 12 bytes of every record, and the records are arranged in ascending order of entity key value within each file.

The 150 files are sub-divided into 9 groups of files. Each group of files have similar but not equal, record lengths:-

File Group	Maximum Record Length (Bytes)	Number of Files in Each File Group
1	888	1
2	60	19
3	66	4
4	96	19
5	152	29
6	344	31
7	968	28
8	2760	12
9	1600	7
		150

The grouping of the files is chosen for maximum usage of the disc storage capacity and without regard for logical relationships between the files.

When a file is allocated to a file group, the 12 byte entity keys are extended to 14 bytes; the first 2 bytes being used for the file group sub-group number.

File Group	Sub-Group	File Number	File Title	Record Length (Bytes)	Entity Key of First Record in each file	Number of Records
3	02	056	Empty Fuel Pin	65	02100000------	60000
3	03	065	Reworked Pin	61	03177565------	250
3	05	135	Components	49	05ABC760302---	300
3	06	134	DMSA Load	56	06XYZ003------	50

The above system preserves the identity of each file within the file group, all keys within a file group are unique (duplicate keys in files are not permitted), and all the records in a file group are maintained in ascending order by the record key values.

In order to recover a specified record from a particular file, the file number and the 12 byte key to the record is specified. A file table in the database is interrogated for the correlations file number to file group and file sub-group. For example, to recover the record with key 177987 from file 065 (Reworked fuel pin file), 065 is located in file table which returns file group 3 sub-group 03. File group 3 is then searched, randomly or sequentially, for the record with the key 03177987.

The file groups are arranged in vertical sets of tracks (cylinders) on the disc packs in order to minimise radial head movements during the selection of the appropriate sub-group.

File Navigation

There is a hierarchic relationship between the files. A typical descending hierarchy of a small sub-set of files is shown in figure 1.

Navigation down the file hierarchy is shown (fig 1) to be provided at the record level. The data in each record in a file is preceded by the entity key value which are in turn the entity keys to related records in the files lower down the hierarchy.

Data Types

The following data forms are stored in this databank:-

A	Alphanumeric, 1 character = 1 byte
N	Numeric, 1 number = 1 byte
H	Half-word Integer ($< 2^{15}$) = 2 bytes
I	Full-word Integer ($> 2^{15} < 2^{31}$) = 4 bytes
F	Floating Point (E ± 64) = 4 bytes

Decimal points and units are inserted at output time. Sub-routines are provided for the conversion of numeric characters to binary integers for calculational work in applications programs.

Data Input

The data originator completes pre-formatted data collection forms. This is punched on to 80 column cards to provide the data set for the INPUT program. The program inspects the identifier characters on the data cards which reference the Card Layout File records stored in the bank. This in turn directs the data to the appropriate files and records for updating. If a part record already exists for the quoted entity key, that record is updated; if the entity key does not yet exist, a new record is created. The entity key value is checked for correct format before record creation or update by reference to a Keys-Format file. The data values are tested for validity and specification limits and suitable Error Messages are returned on hard copy to the data originator. The system software indexes the new records sequentially into the files or the file overflow areas.

Data Retrieval

Data retrieval is controlled by the databank access module. The module is written in COBOL occupying 8.5 k bytes of core plus 60 k bytes required by the ISAM routines. The principal entry point in the module is DBREC; this and other entry points are compatible with FORTRAN. A typical FORTRAN call statement with the DBREC arguments is:-

CALL DBREC (IFILE, IOP, ISTAT, IREC, IKEY) Applications programs must provide a 3 k byte area to hold the retrieved record (IREC), and a 12 byte area for the record key (IKEY). The selection mode is specified by a digit for random or sequential (IØP), the required file by the file number (IFILE), and the routine will return a status report such as good or bad read (ISTAT) to the program. The routine is used by application programmers for manipulating selected data from the databank.

Selected records may be obtained from the bank by a standard program DBREAD for the inspection of file records by non-specialist programmers. The program will only recover data from specified files and not from associated linked records. However, a standard program SEARCH is available that will provide an automatic file walk and data recovery for all linked records across the file hierarchy for a given starting entity key. The program uses the file directory and requires no knowledge of filing structures by the user.

A data manipulation language is provided as an instruction set in the program DBLOOK. This permits the selective retrieval of data items from linked records across the file structure. It does not require expertise in program procedures but does demand an intimate knowledge of file linkages and data item locations.

File Maintenance

A complete suite of housekeeping programs is available to the databank administrator for maintaining directories, dictionaries and the ordering of the database records. These include the normal DUMP, REORGANISE, DELETE, AMEND, JOURNALISE AND ARCHIVE features.

The Scope of the Current Databank

The following remarks can only give a very brief indication of the scope of the databank in its current form. For any one of the 250 sub-assemblies at present in the reactor core or intended for replacement purposes, the databank will provide information on its significant components. Several thousand data values provide information on its significant components. Several thousand data values can be returned for each unit, covering manufacturing and inspection data (including metrology and material analysis). Some 50,000 fuel pins are recorded for which 400 fuel batches, 1500 breeder batches, 1500 component batches and 1500 pin tubing batches are associated. The steel derivatives (pin tubing and components) are associated with 500 ingots from 400 casts. Data for the 250 carriers (hexagon tubes) linked to some 2000 internal component batches and 4000 fuel pin support grids are readily available. The simplest search question for the bank of 'report all' for 6 sub-units in one hexagon carrier returned 120,000 lines of printed data output. Permutations of possible questions based upon the several thousand individual data types in the bank for the large units (sub-assemblies) and the sub-units (clusters) are almost infinite and are answerable within minutes on the computer; questions such as "report the vanadium content of the fuel pin tube from any one of 50,000 pins, give the in-core disposition of a particular (say suspect) fuel batch, give tabulated detailed pin dimension data, give metrology data for a set of fuel pin spacer grids found in a specified cluster sub-unit, give fuel enrichment and O/M ratio for fuel in a pin without quoting the fuel batch number, give the pre-irradiation weight of a fuel pin", and so on. The monumental task of recovering such answers from dispersed paper records is obvious.

Conclusions

The PFR fuels databank is a well organised high activity filing system and contains at present detailed information upon the individual elements of PFR fissile and non-fissile components. In its present form it is an ordered collection of data that is being used as a data source for the solution of thermal and nuclide/neutron interaction calculations. The bank will eventually contain data upon fissile and non-fissile component performance under PFR irradiation conditions upon which detailed performance analysis will be assessed and from which the optimisation of CFR will be more readily achieved.

The ICL 2980 computer complete with the database management system package IDMS is now installed at the UKAEA Headquarters (Risley) and is currently undergoing pre-acceptance trials. The computer should be available to users in July 1977.

With the databank at its current size of 100 M bytes, there are several operational criticisms:-

(i) File reorganisation time is becoming excessive. The whole bank required 74 minutes for copying (dump). The reorganisation times for the individual file groups are as follows:-

01 - 8 mins;	02 - 9 mins;	03 - 13 mins;
04 - 14 mins;	05 - 46 mins;	06 - 27 mins;
07 - 43 mins;	08 - 20 mins;	09 - 39 mins,

totalling 219 minutes.

This time consumption arises because reorganisation takes place at the file group level; a large proportion of the files within the file group do not require reorganisation.

(ii) Data retrieval times can also be large, not only because files to be retrieved may lie deep in the hierarchy but also because of excessive radial head movements on the disc packs. This arises from the storage of logically unrelated files (having only similar record lengths in common) in vertical cylinders of tracks.

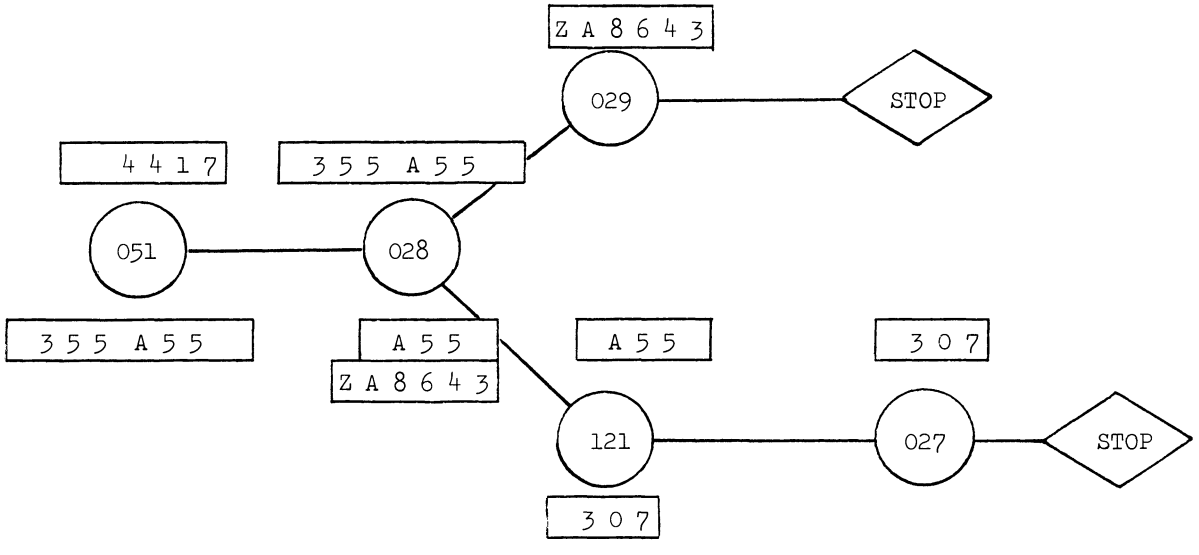
(iii) It follows from (ii) that interactive transaction processing is not a viable proposition with the current databank structure constraining users to batch mode processing.

(iv) The current amendment/deletion system does not provide total automatic removal of redundant records. A proportion of disconnected records result.

(v) Application programmers often require detailed knowledge of the structure of large areas of the databank and the navigational paths between the records.

Although it is unlikely that the whole databank will be restructured into IDMS, selected sub-sets of related files will be re-ordered into this database management system. In this manner, we will probe the potential of IDMS. The page dump method is expected shortly on IDMS (area/realm dump only is currently available) and this will reduce copying and reorganisation times, (i). Enhanced retrieval ability must follow from the clustering structure of related records by the IDMS command 'VIA', and transaction processing facility should follow from faster retrieval times, (ii) and (iii). The automatic fade-out of disconnected records provided by IDMS will reduce redundancy, (iv). The provision of sub-schemes to application programmers will remove the burden of file structure and navigational path knowledge from the user, (v). It is hoped that the result of a 3 month exercise in this field will be available by October 1977.

FILE	KEYNAME	SUBKEY (1)	(1) GO TO FILE	SUBKEY (2)	(2) GO TO FILE
051	PIN NO. 4417	FUEL BATCH 355 A55	028	LAST OF 5 BYTES OF FUEL BATCH A55	121
028	FUEL BATCH 355 A55	MET LAB SERIAL NO. ZA8643	029		
029	MET LAB SERIAL NO. ZA8643	STOP			
121	LAST OF 5 BYTES OF FUEL BATCH A55	GRANULATION BATCH 307	027		
027	GRANULATION BATCH 307	SOURCE POWDER BATCH C00H206	023		
023	SOURCE POWDER BATCH C00H206	STOP			



THE CARBIDE BREEDER FUEL FILE HIERARCHY AND FILE WALK

DESIGN OF A SOLAR HEATING AND COOLING DATA CENTER

D. Deutsch

Institute for Computer Sciences and Technology
National Bureau of Standards*
Washington, DC 20234 U.S.A.

The National Bureau of Standards designed and operates a Data Center serving the Federal Solar Energy Research, Development and Demonstration Program. The design effort included thorough consideration of the applicability of generalized database management software. The functional requirements for the Data Center and the factors influencing the database decision are described. A modified database approach is presented and reasons for its adoption by the Data Center are discussed.

Key words: Database management; data center; GDMS; software selection; solar energy; system design.

1. BACKGROUND

The Congress of the United States in 1974 enacted legislation establishing an interagency task force for carrying out a five year Federal Solar Energy Research, Development and Demonstration program. As part of that program the Institute for Computer Sciences and Technology (ICST) of the National Bureau of Standards (NBS) was charged with development of an operational Data Center for receipt, maintenance and distribution of technical and non-technical data pertaining to the program. The requirements analysis and design activities carried out during the fourteen months ending October 1977 were concerned with the selection of hardware and software tools, and the preparation of an integrated system design for serving the many diverse data providers and Data Center users; one major design consideration was the potential applicability of generalized database management software.

---

*NBS/ICST Data Center development activities were supported by the NBS Center for Building Technology under interagency agreements IAA-H-38-76 with the U. S. Department of Housing and Urban Development, and E-49-1-3800 with the U. S. Department of Energy (formerly Energy Research and Development Administration). Development of this report was supported in part by the U. S. Department of Energy under Interagency Agreement No. EA-77-A 01-6010, Task No. A050-TI.

The NBS Solar Heating and Cooling Data Center receives inputs from several sources and provides information and processing services to a number of institutional users. The Center maintains numeric, alphanumeric and graphical data that is accessible via on-line as well as batch processes. The source database will grow from the modest four million characters available in the last half of 1977 to an estimated fifty to one-hundred million characters in 1980.

2. FUNCTIONAL REQUIREMENTS

The functional requirements for the Solar Heating and Cooling Data Center are perceived as falling into three broad areas. First is the receipt and maintenance of machine-readable source data in a form suitable for satisfying all output requirements, either directly or through additional processing steps. A second area is the production of printed reports and compendia both for project participants and for a variety of other users. Finally, there are less predictable ad hoc and presently undefined requirements that will be specified throughout the five-year life of the project. Figure 1 illustrates this three-part conceptual view.

2.1 Data Receipt and Maintenance

The Solar Heating and Cooling Data Center provides a central location for the receipt, storage, processing, and reduction of data collected from solar demonstration projects. Machine-readable data is received from data-formatting and encoding contractors as well as from other data collection and processing facilities.

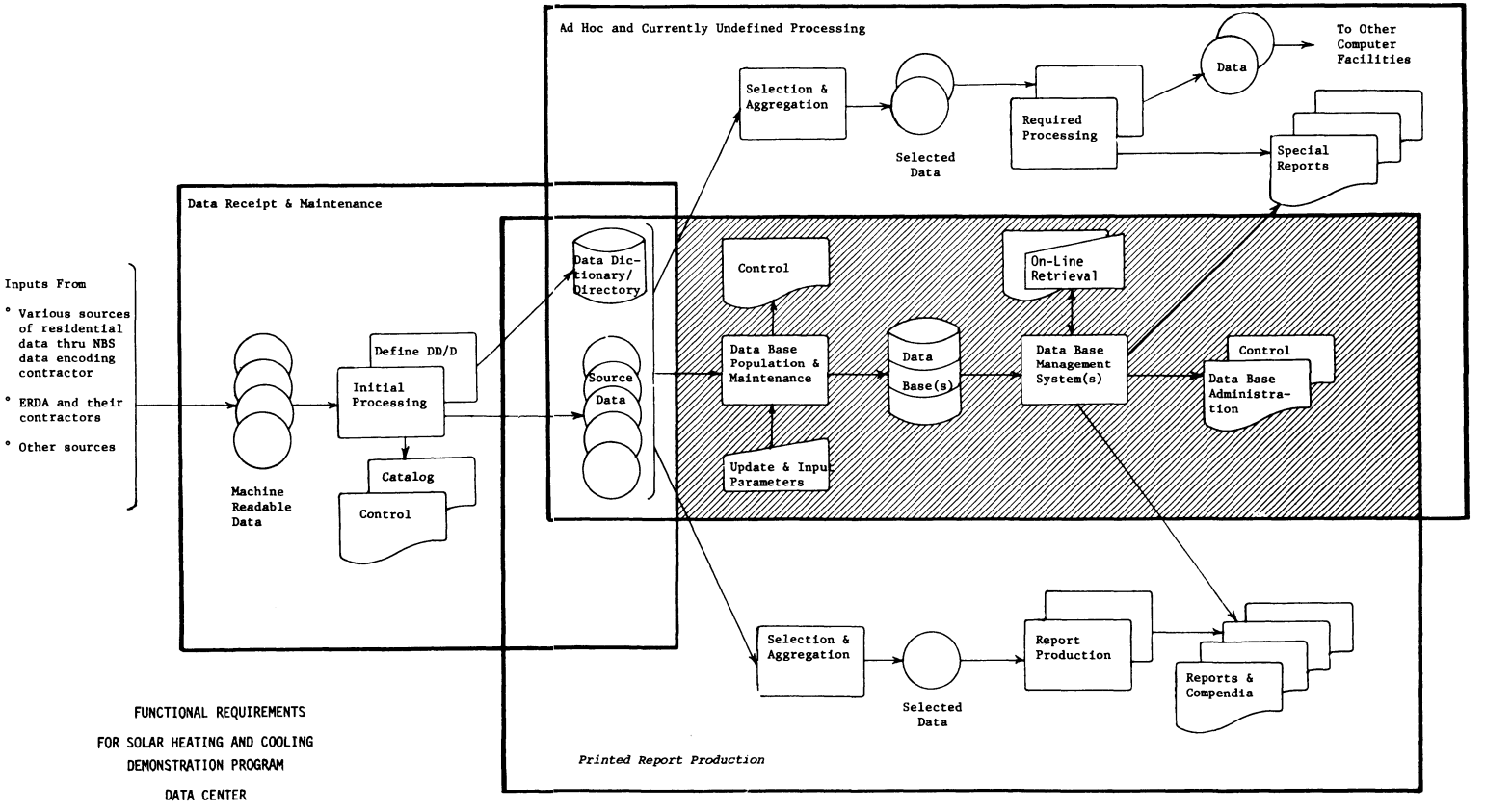
Data received at the NBS installation is cataloged and, if necessary, edited for accuracy prior to insertion into the database. All pertinent source data are retained, and a catalog of available data is published periodically for potential Data Center users by this NBS data repository. This approach helps assure the security, integrity and wide distribution of the data throughout the life of the project.

2.2 Production of Printed Reports

A major continuing function of the Data Center is the printing of summaries of the data using various selection criteria and levels of aggregation. Printed reports and compendia are produced both periodically and upon request. Their timely production is facilitated through the availability of generalized software capabilities such as report generators, sort packages, statistical programs, file manangement systems, and data management tools.

2.3 Ad Hoc and/or Curently Undefined Processing

In addition to the more well-defined and predictable requirements for data receipt and maintenance and production of printed reports, the Data Center satisfies various other functional requirements as they become known. The general approach followed is to forecast as accurately as possible additional requirements that will be placed on the Data Center. These requirements are then matched against the array of basic processing and data management tools available on or obtainable for the NBS computer



FUNCTIONAL REQUIREMENTS
FOR SOLAR HEATING AND COOLING
DEMONSTRATION PROGRAM
DATA CENTER

Figure 1

facility. In the event that the NBS facility does not appear to be able to satisfy some class of requirements, alternatives are identified. So far all unanticipated information needs have been handled in-house with tools that were already in the NBS software inventory.

3. FACTORS INFLUENCING THE APPLICABILITY OF GDMS

The potential applicability of Generalized Database Management Systems (GDMS) to the Solar Heating and Cooling Data Center was influenced by several factors including those listed below.

Complex Structure - Major applications exploit structural relationships among collected data items; specifically, the analysis of failure data using a component tree for organizing both system descriptions and maintenance data is planned.

Generation of Printed Reports - A primary function of the Data Center is to produce hard-copy listings and summarizations of the data received.

On-line Queries - Portions of the database must be immediately accessible for responding to on-line queries.

Limited Technical Staff - NBS/ICST, primarily a research organization, does not have the personnel to staff a heavy coding effort such as that associated with the development of custom programs. Two full-time employees of the Data Center, a Database Administrator and a mid-level programmer/analyst, interface with data providers and users, monitor contractors and operate the Data Center. After the development effort is completed ICST will serve only in an advisory capacity; responsibility for day-to-day operations will belong to the Data Center staff.

Lagging Requirements Definition - Initially, data collection activities progressed faster and further than the definition of user requirements. Consequently, data were being collected before many users were identified and their requirements defined.

Diverse User Access Requirements - The diverse nature of the user community requires that access to the database be provided via several high-level languages.

Post Processing - Post processing of data extracted from the database using various statistical and other tools is necessary.

Privacy Constraints - The Privacy act of 1974 imposes stringent requirements for limiting access to information that could threaten individual privacy. Consequently, security and integrity constraints were a major consideration in the design and acquisition of software for the Data Center.

Use of Existing Computer Facility - There was a strong predisposition on the part of funding agencies and Data Center users to utilize the NBS computer facility. This resource currently includes a Univac 1108 processor and related peripherals.

4. EVALUATION OF DESIGN CONSTRAINTS

The contradictory nature of design constraints made the decision of whether to "go database" a difficult one. The factors listed above both suggest and discourage the application of generalized database management software. On the positive side, the availability of data prior to determination of user requirements indicates a need for the flexibility promised by GDMS packages. The complex structural relationships and privacy requirements also point to the use of database technology.

On the other hand, the generation of many of the desired printed reports can be accomplished using software tools other than GDMS; report generators and file management systems are generally both easier to use and less expensive than database manangement software. Also on the negative side is the question of whether GDMS available in the marketplace can completely satisfy the requirements for post-processing and access via multiple high-level languages. Finally, the list of available GDMS for the NBS Computer configuration is limited.

Two design constraints were not not seen as clearly positive or negative with regard to the use of GDMS software. The requirement for answering on-line queries could be satisfied by some database management systems, but not by others. Other, less costly and less complex, software packages could provide on-line retrieval capability. The limited availability of technical personnel also did not clearly indicate whether a GDMS should be used; a staff that is not large enough to develop a custom software system, might not be of sufficient size for building and/or supporting a complex database. While the development could be done by an outside contractor, project management wanted the day-to-day operation and administration of the database to be an in-house function.

Because no clear conclusions regarding the use of GDMS followed from an analysis of Data Center functions and design constraints, a modified database approach was chosen; GDMS software is being applied where it is clearly desirable. In addition, an array of software tools is available for augmenting the capabilities of the GDMS and for satisfying processing requirements not particularly suited to database solutions. The modified database approach for developing the Solar Heating and Cooling Data Center is described in the following section.

5. ROLE OF GENERALIZED DATABASE MANAGEMENT SYSTEMS

The three part conceptual overview of Data Center functions appearing in figure 1 also illustrates the planned cross-functional utilization of generalized database management systems. The shaded area in the figure identifies the functions planned for GDMS based implementation and the utilization of these generalized software packages for both generation of printed reports and ad hoc processing. The use of a computerized data dictionary/directory for supporting the receipt and maintenance of data as well as database design and administration activities is also depicted.

Because of the range and uncertainty of requirements -- rich structure, on-line query, generation of hard-copy reports, response to ad hoc requests -- a hierarchy of software tools including multiple generalized database management packages are available for implementing Data Center applications. In addition to conventional software tools such as language

translators, report generators and statistical packages, two types of generalized database management systems are used by the Data Center. While the bulk of report generation requirements are satisfied using a generalized report generator, the production of complex reports and the retention and analysis of data that is richly structured (as in the case of the component tree and failure data) is accomplished using a CODASYL type GDMS. This type of data management tool uses a high-level programming language as a host; that is, it is invoked by imbedding references to the database in application programs written in COBOL or other high-level languages.

A second class of database management system is used when selected subsets of the data maintained by the center are identified as objects for on-line retrieval. Two on-line query systems have been acquired. One, developed by a U.S. Federal Agency, provides a capability for establishing a database in a short period of time that can respond to on-line requests. It is not suited for maintaining large volumes of data nor does it handle complex structures efficiently, however. The second system is a commercial package that provides a query language interface for the CODASYL database management system. While not as flexible as a self-contained query-response system, this alternative is attractive because it allows retrieval from databases established for other (e.g., complex report generation) purposes.

The link between the file oriented receipt and maintenance function and the database(s) serving Data Center users is provided by a comprehensive data dictionary/directory. This automated compendium has entries for every data item and aggregate. Entry of data item definitions using software developed by NBS has been underway for several months but is not complete; it is viewed as a continuing task. The data dictionary grows and changes as the amount of data received increases, databases are established and applications are developed.

6. SUMMARY

The Solar Heating and Cooling Data Center maintains a large database serving a diverse community of users. Because the application of GDMS technology is not clearly preferable to other less complex and less costly alternatives for all aspects of Data Center operation, a modified database approach is being used. While generalized database management software is applied wherever requirements demand, other software tools are also used. To date, this approach has provided the desired flexibility with a minimum of design and development costs.

SDI-PROGRAMS FOR SMALL COMPUTERS USING THE INIS-DATABASE

A. Nevyjel, Österreichische Studiengesellschaft
für Atomenergie Ges.m.b.H.

Scientific data handling activities in Austria are in general restricted to using existing databases and participating in international information systems. As in other smaller countries, the requirements in manpower and computer equipment would be too large to allow for producing databases of our own. Furthermore, rather modest computer installations impose the use of "simple tape storage systems", very similar to the Geneva INFOL system, discussed earlier in this report (1).

In planning a national automatic information retrieval system using the tape services of international documentation organizations as a database the question arises "What program philosophy and file organizations are practicable and advantageous for the computer equipment available?". There exist some information retrieval programs developed by the computer manufacturers, but the layout of most of these programs is for very large computer installations, since the file organizations used are very expensive. For smaller computers it is more practicable to use the magnetic tape received from an international organization directly as a database for information retrieval. That means that the file of documents stored on tape is scanned sequentially in the search run, while random access storage on disk is used only for temporary work files. In this way the programs can manage with a minimum of one disk drive with abcut 7 million byte storage capacity.

On the other hand the response time in such file searches is relatively slow, since a complete file scan is needed before any information can be retrieved. So the main problem in using such a program philosophy is to accelerate the working process of the computer. A high speed performance can be reached, if a machine readable thesaurus is used and if besides the descriptors the descriptor numbers are also included in the records on tape. This implies the restriction that only descriptors recognized by the thesaurus can be used as keywords in the search profiles currently maintained by the Austrian information service.

This program philosophy is described in a report of the Austrian Research Centre Seibersdorf (2) and was also discussed in a paper presented at the International Symposium on Information Systems in Varna 1974, Bulgaria (3). The so-called direct file organization in batch processing procedures implies that there should be a large amount of profiles processed together using a rather small database. So SDI is a form of information service well suited to these programs. The monthly INIS output tapes (about 6000 items/month) are used as a database for the SDI.

For the query formulation we have a standardized internal format, which enables the user to submit one or more profiles for one or more problems. The profiles are formed as Boolean expressions of keywords. These profiles are exclusively decisive for the search run. But the user has the possibility to include also the free text formulation and other remarks for his problems in the submitted deck of cards. This information is then printed out on the first page of the user's listing of results and gives an additional identification of the listings.

At present the programs are used to provide a monthly SDI-service for about 500 profiles, to about 130 users, using the monthly INIS outputtape as a database.

Our information centre in Seibersdorf services all universities of Austria and some industries with the INIS data. So about 50% of the users are from universities, 5% from industry and 45% in-house users. In future it is planned to extend the information available to include some international data services (e.g. RECON) by terminal, so as to allow for retrospective searches and for extension of the subject scope covered.

With regard to a GDMS containing numerical data, no development in the longer term is envisaged in the Austrian Nuclear Research Center, since the manpower needed for an effort of this kind could not be justified. However, great interest exists in implementing a suitable system developed elsewhere, or if it is more appropriate from the technical and economical standpoint a link to an established computer centre should be envisaged. International work to find the best solution for the customer is strongly recommended by the Austrian delegate to the NEA-NINF working group.

References

1) Moorhead, G. and Tubbs, N.
An introduction to Generalized Data Management Systems (in this report)

2) Nevyjel, A.
SDI-SGAE, SDI-programs for small computers, program description 1974, available from NEA-CPL Computer Program Library.

3) Nevyjel, A.
Problems of automatic information retrieval at the Austrian Research Centre Seibersdorf.
Proceedings of the International Symposium on Information Systems, their interconnection and compatibility, Varna 1974, Bulgaria
IAEA-SM-189/20, pp. 61-69

SCIENTIFIC DATA HANDLING, NEEDS AND PROBLEMS AT THE ZENTRALSTELLE FÜR ATOMKERNENERGIE-DOKUMENTATION (ZAED)

W. Bau and H. Behrens
Zentralstelle für Atomkernenergie-Dokumentation (ZAED)
Federal Republic of Germany

I. Introduction

In 1974 the Federal Government passed the "Programme of the Federal Authority for the Promotion of Information and Documentation" (IuD Programme) which intends 16 Specialised Information Systems to be established, the core of each one to be a Specialised Information Center. The Zentralstelle für Atomkernenergie-Dokumentation (ZAED) together with the Zentralstelle für Luft- und Raumfahrtdokumentation und -information, the Physikalische Berichte and the Zentralblatt für Didaktik der Mathematik will be the center of the Specialised Information System 4, which will cover the fields of energy, physics and mathematics. It will be located at the Karlsruhe Nuclear Research Center.

As it belongs to the tasks of Specialised Information Centers not only to look after literature documentation and information but also to extend this service quite explicitely to data the ZAED has been commissioned, in anticipation of the establishment of the Specialised Information Centre, by the Federal Ministry for Research and Technology to substantially improve the situation in the field of physics data by establishing a data information system. For this purpose, data compilations are regularly to be published - and at the same time to be recorded on magnetic tape - in a number of physics subfields, and to be kept up to date. Moreover, as complete as possible, a list of existing data compilations in the world is to be compiled in order to be able to give information on these and to facilitate the search for data. Another task will be the collection and distribution of existing data compilations for the Federal Republic of Germany. This applies in particular to data compilations on magnetic tape, for which ZAED, at a later date the Specialised Information Center of course, will act as a distribution center. In order to avoid duplication of work in compiling physics data, ZAED will also have the task of coordinating to a certain degree the

activities in this field. At the same time, close international cooperation is envisaged.

II. Requirements for a physical data bank

a) Definition of physical data
It has to be made very clear, that whenever the term "data" is used, we are thinking of physical data, e.g. density, temperatures, melting points, cross sections, etc.
This is important, as in computer terminologie the term "data" has been applied to too many things which are quite foreign to the matter which is under discussion here.

b) It is our aim not only to publish physical data in printed form, but at the same time also to establish a data file in machine-readable form. The latter point is of importance for small- and large-scale data compilations alike as only with the aid of a computer a quick and uncomplicated update procedure can be achieved.
In addition, in the case of large-scale data compilations it is inescapable to have the magnetic tape version in order to be able to retrieve and select, and also in order to be in a position to handle the data in a practical way.
However, as the user continuously makes bigger demands on retrieval, demands which cannot be satisfied with an ordinary data file, e.g. to establish logical connections between various data, to retrieve numerical values within a given interval, to resort the data according to new criteria, etc., it is consequently necessary to extend the file to a data bank.

c) Building a physical data bank, in our opinion, should entail the following:

- To store matrices (data tables) with m columns and n rows (m and n to stand for any specific number), whereby the elements of these matrices can contain numerical as well as alphamerical information. Thus, it will be characteristic for a data bank to have a lot of matrices of this type.

- It should not only be possible to sort rows and columns according to new criteria within one matrix, but also to combine certain rows and/or columns from different matrices into a new one.

- With references to the update procedure it is necessary to be able to replace the elements of a matrix by new ones in a unique way. Furthermore, it will also be necessary on the one hand to increase the number of rows and columns of the matrix, and on the other to add new matrices. The first point (replacing elements) is of particular importance for data banks which contain evaluated data only; the latter points are particularly relevant as far as data banks containing experimental data are concerned.

- A data bank, besides these matrices, would also include bibliographic items, e.g. in the case of a data bank with experimental data, the bibliographic part would contain

bibliographic data of the literature from which the data originates.
Last but not least, it should be mentioned that the ZAED will apply the International Nuclear Information System (INIS) for this bibliographic part.

- Another part which a data bank must have is reserved for the necessary information relevant to the data, e.g. name of material, chemical formulae, material composition, data type, method of determination.
As in the case of the bibliographic part, the information part, too, has to be related to the data in question.
Some of the information will be standardized by the use of key words.

III. Concept

a) As already mentioned in the introduction, ZAED intends to operate as a distribution center for existing data files and data banks, which should run on the ZAED-own computer Siemens System 7.755-J.
Existing data files and banks, however, have been built or have accomplished the requirements for a data bank outlined under chapter II in completely different fashions. This fact entails that the format and structure of the parts described above, are not compatible with each other, and this is the reason why ZAED uses in each case the programmes belonging to the data file or bank in question. It goes without saying, that these programmes are as different from each other as the data files or banks they belong to. However, compatibility of these programmes, in our opinion, does not seam feasible in the near future.

b) Another of our targets, also mentioned in the introduction, is to develop and to establish data files and data banks in fields of physics not yet covered. At the moment we are in the process of establishing a data compilation on superconductivity which is to grow gradually into a data bank for this field. We realize the requirements for a data bank, as described under chapter II for this particular project as follows:

Information part
Name of material or trade name
Chemical formulae
Material composition
Material description
etc.

Bibliographic part
Title
Author (s)
Literature reference
etc.

Data part
This part contains the matrices with the data as described under II, c), for example: Difference of entropy in normal states and in superconducting states as a function of temperature. Information belonging to this matrices, such as description of the quantity measured, other parameters, validity range, data type, measuring method, etc., are - according to our system - recorded under the information part.

PROBLEMS OF A NUCLEAR DATA CENTRE IN AN INTERNATIONAL NETWORK

P.M. Attree, IAEA Nuclear Data Section, Vienna

1. Introduction

This paper presents the environment within which the Nuclear Data Section (NDS) of the International Atomic Energy Agency (IAEA) operates, the systems which currently exist, and examines possible NDS use of the ADABAS system when it becomes available for general use within the IAEA at the end of 1977.

2. Computer environment of the Nuclear Data Section

Following a recent upgrade, the IAEA is serviced by an IBM 370/158 with 3 Mbytes central memory, six 3330 and three 3350 disc units, as well as the usual tape and card handling equipment. Further significant upgrades are anticipated in 1978 and 1979. Regarding software, the usual programming languages are available, the change to MVS operating system was made recently and, most significantly, ADABAS was installed during 1977 and will be available for general use at the end of the year.

NDS is located in an annex, so that its work must pass through an RJE station for inclusion in the batch stream of the 370/158. Currently, NDS uses about 4% of the capacity of the 370/158, but the load is expected to increase with the widening scope of the Section's activities. The systems in operation at the moment were developed and programmed at NDS, using PL/I and, to a lesser extent, FORTRAN.

3. NDS participation in the international exchange of nuclear data

For neutron-induced nuclear reactions NDS is one of the nodes in a 4-centre data exchange network. The data centres are located in:

Brookhaven, USA	National Nuclear Data Center (NNDC),
Obninsk, USSR	Centr po Jadonym Dannym (CJD),
Saclay, France	OECD/NEA Neutron Data Compilation Centre (CCDN),
Vienna, Austria	IAEA Nuclear Data Section (NDS).

For experimental neutron data the world is divided into four areas: each centre compiles data produced in its area and transmits the information to the other three centres. Thus the data files should be identical at each of the four centres. Each centre services requests for information from its own area. This data is exchanged in the EXFOR format, developed in common by these four centres. Besides simplifying this four-way exchange of data, an additional aim of EXFOR is to standardize the content of compilations by means of a combination of controlled keywords with associated codes, which can be used for retrieval purposes, and free text explanation.

It is important to realize that although each centre may maintain different systems internally, it is essential that the structure of EXFOR is not violated for exchange purposes. This is clearly a major constraint when considering revisions to internal systems. A short guide to EXFOR is given in Appendix I.

In order to maintain the consistency of the information in EXFOR, a series of dictionaries are maintained. These are used to control the codes which are permitted with each keyword. It is the duty of NDS to maintain these dictionaries, and regularly to send up-dates to the other centres.

The structure of EXFOR was designed in such a way that the scope of information compiled could fairly readily be extended. This has recently been done to include charged particle data, and the network has been enlarged by three other centres. However, in this case, the Karlsruhe Charged Particle Group is responsible for collating the compilations and transmitting the complete master file (at the moment rather small) to the remaining centres at regular intervals. It must be foreseen that in the future both the scope of information compiled and the number of centres involved will expand.

Each of the four nuclear data centres is responsible for servicing requests from its area for evaluated data. The content and format of the evaluated data files are controlled by the originators of these files. There are three or four major files, all having different formats which are updated intermittently by the originators and used for selective retrievals for users. There are also specialized evaluations, which are usually received from evaluators and transmitted to users and other centres in toto.

Another area of inter-centre co-operation is CINDA. The master file is maintained by CCDN, Saclay, but the other centres provide input to the system. NDS is responsible for the production of the CINDA book. NDS also receives the complete master file about four times per year which is used for in-house operations. Although it is possible to generate computer links between CINDA and the EXFOR data, these are not maintained at NDS except in the form of a number of common dictionaries.

The final area of co-operation between the four nuclear data centres is WRENDA - World request list for nuclear data, a list of measurements or evaluations which are requested to be made, and not to be confused with requests for data from existing files. This is a low level activity at NDS which peaks around the publication date of the list. The master file is maintained by NDS and input is received from the other centres. There is no direct overlap between WRENDA and the other systems, the WRENDA tables and files being almost completely independent, with the exception of some of the EXFOR dictionaries.

4. Current operations at NDS

4.1 Dictionary maintenance for EXFOR

The master file, which is kept on tape, contains about 6000 logical records of 88 characters each. The file is updated as needed, usually once or twice per month. Immediately following each update three ISAM files are created, from the tape, on disc. These are used extensively for checking purposes and also to provide code-expansions for edited listings. The dictionary master file is transmitted to the other centres every three months, at which time listings are also produced for data-centre physicists.

4.2 Libraries in EXFOR format

All master files are kept on tape. The regular neutron reaction experimental data library resides on 4 high density tapes, split according to the code of the centre from which the data originated.

Area	Approximate number of records
1 - NNDC	830,000
2 - CCDN	810,000
3 - NDS	120,000
4 - CJD	110,000
Total	1.87 million

Each record is 80 characters in length.

In addition there are two sub-master files, containing data from the period before the 4-centre network existed, with a total of about 700,000 records. The additions to the regular master files are roughly constant each year. In 1976 this amounted to an increase in file size of about 25%.

There are two additional master files in the same format; one containing evaluated data not yet included in any other library (about 6,000 records) and the other containing the charged particle data (about 12,000 records).

The master files are each updated about once every three months upon receipt of a transmission tape containing new and revised entries. Data compiled at NDS is kept on tape as a separate file (LIMBO) until it has been thoroughly checked. It is then transmitted to the other centres and added to the master file.

When data are added to the master files the index is updated. The index is a very important part of the system. Data-elements which may be required for retrieval purposes are extracted from the EXFOR entries and stored in the index master file in a standardized form. At retrieval time, the index is matched against the request; when an equivalence is found, the index record points to the required sub-entry on the master file by its accession number. Listings of the index are also used extensively by the compilers at NDS. The current index-master file resides on tape; it contains about 40,000 logical records each of 200 characters.

The flow of data in the EXFOR system, as it currently exists at NDS, is shown schematically in Appendix II. The EXFOR libraries are accessed about three times per month for requests from users in the NDS service area; the index is accessed far more frequently. Output from the files is in the form of either edited listings (see examples in Appendix I) or in standard format, usually on magnetic tape. Both are accompanied by an index-listing of the data retrieved.

4.3 Evaluated data libraries

The number of data libraries originating outside NDS, but held for distribution to requesters in the NDS service area, has grown over the years from 4 in 1970 to 14 in 1974 to 32 in 1976. It should be noted that in addition to evaluated data libraries, this figure includes specialized compilations of experimental data. Many of them are simply copied in toto and sent to users on request. Several, however, are large libraries in one or the other standard evaluated data format. From these libraries selective retrievals are made on the basis of reaction plus isotope or data-set number. No computerized index is kept of these libraries, but the contents of each is published in CINDU-11, which is updated regularly. The standard evaluated data libraries are accessed several times per month to satisfy requests from users in the NDS area. All libraries are stored on tape. The output is either in the original format (usually on tape) or as edited listings.

5. Future plans of the NDS data centre

As is probably the case in most data-centres, the various systems in operation at NDS have grown to meet immediate needs and to satisfy the commitments within the four-centre network. This has often meant ad hoc patching as the environments have changed and in particular as extensions to EXFOR have been introduced. This type of growth has been useful in so far as it has given us considerable experience in operating a data centre without massive expenditure on systems which may then have been difficult to modify as requirements changed. The NDS EXFOR system, for example, uses some 20 independently compiled programs, and can easily be reconfigured as changes are required.

However, the time has now come to integrate and improve the NDS systems, without discarding all that exists at the moment. In particular we need to improve in-house operation by:

- co-ordinating the dictionaries and various tables which are used by the different systems, keeping in mind that these will grow as the scope of EXFOR is extended to other types of data;
- extending the EXFOR index to include other retrieval-fields;
- including the contents of the standard evaluated data files in this index;
- automating all book-keeping associated with user requests.

We need to improve the services to our user community by:

- providing data in a variety of computation formats better suited for input to calculation programs;
- providing graphical plots of the data, if requested.

Keeping in mind limited man-power and limited budget and noting that ADABAS will be available at IAEA, we must decide whether to 'go data-base' and if so, to what extent.

The data index is the nucleus of the data-centre operation. It is also the area where modifications and extensions are most needed. We have therefore recently decided to experiment with ADABAS to see if it satisfies our requirements. In particular we need to investigate the query language and output capabilities, to ensure that these are adequate without having to write a great deal of host language software. Our first tentative attempt at data definition indicates that out of 43 data-fields, 21 need to be defined as 'descriptors' (that is fields which can be used as search criteria). The highly complex structure of the nuclear reaction coding apparently requires 14 descriptor fields in order to enable querying to a sufficient depth of detail. Many requests for data specify not only the nuclear reaction of interest but also an energy range of the incident particle. We must therefore investigate carefully how to handle the problem in ADABAS of searching on floating-point data.

If this experiment proves satisfactory we will load the EXFOR index into the data-base, adding at a later date the index to the major evaluated data files and the book-keeping files. We do not anticipate loading our actual data files into the data-base because of the large volume and the comparitively infrequent access. Queries to the index will give the accession numbers of the data-sets required, which will then be retrieved from the master file tapes.

Initially all our operations will be in batch mode. This will be true in the long term for updating the index because it involves bulk changes and additions at rather infrequent intervals. On the other hand, querying the index in the interactive mode is foreseen in order to facilitate the work of the physicists at the data centre.

A new project has recently been discussed in NDS, namely a compilation of isotope decay properties. This is a potential GDMS application, however its implementation will depend upon the suitability of ADABAS for handling scientific data.

SHORT GUIDE TO EXFOR

EXFOR - a computerized EXchange FORmat - presents in a convenient compact form experimental numerical data as well as physical information necessary to understand the experiment and interpret the data. Keywords and codes make the information computer intelligible. The structure of EXFOR is briefly described in the following.

Each EXFOR "entry" consists of two or more "subentries". The first subentry of an entry contains information which is common to all the following subentries of that entry. Each subentry may include two types of information: descriptive text information and numerical data. Each item of descriptive text information is identified by keywords such as TITLE, STANDARD, ISO-QUANT, which may exhibit a code within parenthesis, such as (GELI), (SCIN) for the keyword DETECTOR or (TOF), (COINC) for the keyword METHOD. The meaning of most keywords is self-explanatory. The meaning of most codes is given in the free text following the code. Of particular importance is the keyword "ISO-QUANT". Under this keyword are coded the "isotope and quantity" or, in other words, the reaction and parameter measured.

EXFOR information is available in two formats:

- the "standard format" primarily designed for the international exchange of data in computer processable form, and
- the "edited format" in which coded information and data tables are edited in an easily legible form.

The EXFOR structure, the standard and edited formats are illustrated in example 1.

There are several categories of numerical data:

- In the DATA TABLE the numerical data of the quantity defined above under ISO-QUANT are given under DATA (or RATIO) together with the columns of independent variables, errors, etc.
- Constant numerical values which are common to the entire data table of a given subentry, are given in the CONSTANT PARAMETERS (also called COMMON in the standard format) section.
- Constant numerical values which are common to all subentries of a given entry, are given in the CONSTANT PARAMETERS (resp. COMMON) section of the first subentry of that entry.

All numerical data are defined by Data-heading keywords (e.g. DATA, EN = incident neutron energy, STAND = standard) and by Data-unit keywords (e.g. EV, MB).

Some data tables may have a more complex structure, for example there may be several ISO-QUANT per subentry; in this case each ISO-QUANT is connected to its pertinent column in the DATA TABLE by means of a "pointer", as illustrated in example 2. More generally a pointer can be used to connect related pieces of information (see example 3).

EXFOR ENTRY 30282.

"EDITED" LISTING

"STANDARD" LISTING

```
NUCLEAR DATA SECTION, INTERNATIONAL ATOMIC ENERGY AGENCY, VIENNA.   ACCESSION NUMBER EXFOR 30282.

BIBLIOGRAPHY, EXPERIMENTAL DESCRIPTION, EXPLANATIONS
     TITLE      ACTIVATION CROSS-SECTIONS OF PT-198 WITH FAST NEUTRONS
     AUTHOR     (I.URAY,A.SZALAY)
     INSTITUTE  ATOMKI, DEBRECEN, HUNGARY
                (3HUNDEB)
     EXP-YEAR   (73)
     REFERENCE  ATOMKI(ATOMMAG KUT.INTEZ.)KOZLEM., 15, (3), 161 (MAR.1973) :  FULL INFORMATION.
                PROG.REPT. INDC(HUN)-11, 14 (SEP.1973) :  ABSTRACT ONLY
     SAMPLE     MEASUREMENTS WERE MADE SIMULTANEOUSLY WITH NATURAL AND
                ENRICHED PT SAMPLES (87.68 PERCENT PT-198).
     STANDARD   ----------------------------------------------------------------
                | 78-PT-198      PARTIAL (N,2N) CS POPULATING A METASTABLE     |
                |                STATE OF THE RESIDUAL NUCLEUS                 |
                ----------------------------------------------------------------
                (78-PT-198,N2N,,MS) NUMERICAL VALUE FROM P.H.WHITMARTER,
                NUCL.PHYS.A 158(1970)77, BRANCHING RATIOS AND CONVER-
                SION COEFFICIENT WERE TAKEN FROM M.B.LEWIS,NUCL.DATA
                SHEETS B7(1972) 129.
     HALF-LIFE  (ML1,78-PT-197-M) THE HALF-LIFE OF THE STANDARD 346
                KEV ISOMERIC TRANSITION WAS MEASURED BY AUTHOR.
     FACILITY   240 KV NEUTRON GENERATOR OF ATOMKI
     N-SOURCE   (D-T)  T(D,N)ALPHA REACTION. THE NEUTRON YIELD WAS
                ABOUT 5E10**9 NEUTRONS/SEC.
     METHOD     (ACT&V) ACTIVATION METHOD
     DETECTOR   (GEL,I) 12 CM3 GE(LI) DETECTOR WITH A RESOLUTION (FWHM)
                OF 3.5 KEV AT 661 KEV. ENERGY CALIBRATION WAS PERFORMED
                WITH AIEA STANDARD SOURCES.
     STATUS     DATA TAKEN FROM ATOMKI KOEZLEMENYEK 15(1973)161.
     HISTORY    (741125C) CA.

CONSTANT PARAMETERS
     EN         = 19.0          MEV
     EN-RSL     = 0.4           MEV
     STAND      = 894.          MB                  STAND DEFINED ABOVE UNDER STANDARD.
     STAND-ERR  = 101.          MB
     ML1        = 64.4          MIN
     ML1-ERR    = 0.8           MIN

THE ABOVE INFORMATION APPLIES TO ALL SUB-ACCESSION NUMBERS STARTING WITH 30282.
```

FIRST SUBENTRY 30282.001

INFORMATION COMMON TO THE ENTIRE ENTRY

KEYWORDS

CODES

CONSTANT PARAMETERS TO ALL SUBENTRIES IN ENTRY 30282

```
SUBENT      30282001     741205                                  30282001000001
BIB             14          23                                   30282001000002
TITLE      ACTIVATION CROSS-SECTIONS OF PT-198 WITH FAST NEUTRONS 30282001000003
AUTHOR     (I.URAY,A.SZALAY)                                     30282001000004
INSTITUTE  (3HUNDEB)                                             30282001000005
EXP-YEAR   (73)                                                  30282001000006
REFERENCE  (J,AK,15,(3),161,7303) FULL INFORMATION.              30282001000007
           (P,INDC(HUN)-11,14,7309)  ABSTRACT ONLY               30282001000008
SAMPLE     MEASUREMENTS WERE MADE SIMULTANEOUSLY WITH NATURAL AND 30282001000009
           ENRICHED PT SAMPLES (87.68 PERCENT PT-198).           30282001000010
STANDARD   (78-PT-198,N2N,,MS) NUMERICAL VALUE FROM P.H.WHITMARTEN, 30282001000011
           NUCL.PHYS.A 158(1970)77, BRANCHING RATIOS AND CONVER- 30282001000012
           SION COEFFICIENT WERE TAKEN FROM M.B.LEWIS,NUCL.DATA  30282001000013
           SHEETS B7(1972) 129.                                  30282001000014
HALF-LIFE  (ML1,78-PT-197-M) THE HALF-LIFE OF THE STANDARD 346   30282001000015
           KEV ISOMERIC TRANSITION WAS MEASURED BY AUTHOR.       30282001000016
FACILITY   240 KV NEUTRON GENERATOR OF ATOMKI                    30282001000017
N-SOURCE   (D-T)  T(D,N)ALPHA REACTION. THE NEUTRON YIELD WAS    30282001000018
           ABOUT 5E10**9 NEUTRONS/SEC.                           30282001000019
METHOD     (ACT&V) ACTIVATION METHOD                             30282001000020
DETECTOR   (GEL,I) 12 CM3 GE(LI) DETECTOR WITH A RESOLUTION (FWHM) 30282001000021
           OF 3.5 KEV AT 661 KEV. ENERGY CALIBRATION WAS PERFORMED30282001000022
           WITH AIEA STANDARD SOURCES.                           30282001000023
STATUS     DATA TAKEN FROM ATOMKI KOEZLEMENYEK 15(1973)161.      30282001000024
HISTORY    (741125C) CA.                                         30282001000025
ENDBIB          23                                               30282001000026
COMMON           6          3                                    30282001000027
EN         EN-RSL     STAND      STAND-ERR   ML1      ML1-ERR   30282001000028
MEV        MEV        MB         MB          MIN      MIN       30282001000029
19.0       0.4        894.       101.        64.4     0.8       30282001000030
ENDCOMMON       3                                                30282001000031
ENDSUBENT      30                                                30282001999999
```

SECOND SUBENTRY 30282.002

CONSTANT PARAMETERS VALID FOR SUBENTRY 30282.002 ONLY.

"DATA" DEFINED UNDER ISO-QUANT OF SUBENTRY 30282.002

```
BIBLIOGRAPHY, EXPERIMENTAL DESCRIPTION, EXPLANATIONS   SUB-ACCESSION NUMBER EXFOR 30282.002
     ISO-QUANT  ----------------------------------------------------------------
                | 78-PT-198      PART,,(N,GAMMA) CS TO METASTABLE STATE        |
                ----------------------------------------------------------------
                (78-PT-198,NG,,MS)
     HALF-LIFE  (ML1,78-PT-199-M) MEASURED BY AUTHOR
     PART-DET   (DG) THE TOTAL INTERNAL CONVERSION COEFFICIENT OF THE
                392 KEV TRANSITION IS ABOUT 0.14 (FROM M.A.WAHLGREN,
                W.W.MEINKE, PHYS.REV.115(1959)191)

CONSTANT PARAMETERS
     ML1        = 13.3          SEC
     ML1-ERR    = 0.2           SEC

DATA TABLE
     DATA DEFINED ABOVE UNDER ISO-QUANT
      DATA          DATA-ERR
      MB            MB
   1  0.5           0.4

SUBENT      30282002     741205                                  30282002000001
BIB             3           5                                    30282002000002
ISO-QUANT  (78-PT-198,NG,,MS)                                    30282002000003
HALF-LIFE  (ML1,78-PT-199-M) MEASURED BY AUTHOR                  30282002000004
PART-DET   (DG) THE TOTAL INTERNAL CONVERSION COEFFICIENT OF THE 30282002000005
           392 KEV TRANSITION IS ABOUT 0.14 (FROM M.A.WAHLGREN, 30282002000006
           W.W.MEINKE, PHYS.REV.115(1959)191)                    30282002000007
ENDBIB          5                                                30282002000008
COMMON          2           3                                    30282002000009
ML1        ML1-ERR                                               30282002000010
SEC        SEC                                                   30282002000011
13.3       0.2                                                   30282002000012
ENDCOMMON       3                                                30282002000013
DATA            2           1                                    30282002000014
DATA       DATA-ERR                                              30282002000015
MB         MB                                                    30282002000016
0.9        0.4                                                   30282002000017
ENDDATA         3                                                30282002000018
ENDSUBENT      17                                                30282002999999
```

THIRD SUBENTRY 30282.003

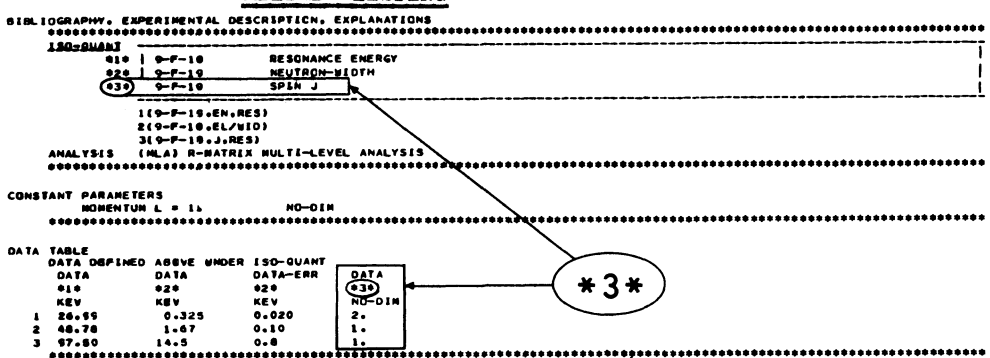
CONSTANT PARAMETERS VALID FOR SUBENTRY 30282.003 ONLY.

"DATA" DEFINED UNDER ISO-QUANT OF SUBENTRY 30282.003.

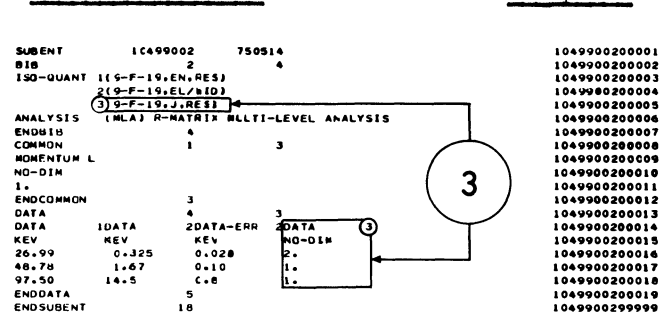
```
BIBLIOGRAPHY, EXPERIMENTAL DESCRIPTION, EXPLANATIONS   SUB-ACCESSION NUMBER EXFOR 30282.003
     ISO-QUANT  ----------------------------------------------------------------
                | 78-PT-198      (N,GAMMA) CROSS-SECTION                       |
                ----------------------------------------------------------------
                (78-PT-198,NG)
     HALF-LIFE  (ML1,78-PT-199) GIVEN BY COMPILER
     PART-DET   (DG) THE 316, 493 AND 542 KEV TRANSITIONS WERE MEASURED
                FOR THE DETERMINATION OF THE TOTAL (N,GAMMA) CROSS-SEC-
                TION.

CONSTANT PARAMETERS
     ML1        = 30.           MIN

DATA TABLE
     DATA DEFINED ABOVE UNDER ISO-QUANT
      DATA          DATA-ERR
      MB            MB
   1  12.           2.

SUBENT      30282003     741205                                  30282003000001
BIB             3           5                                    30282003000002
ISO-QUANT  (78-PT-198,NG)                                        30282003000003
HALF-LIFE  (ML1,78-PT-199) GIVEN BY COMPILER                     30282003000004
PART-DET   (DG) THE 316, 493 AND 542 KEV TRANSITIONS WERE MEASURED30282003000005
           FOR THE DETERMINATION OF THE TOTAL (N,GAMMA) CROSS-SEC-30282003000006
           TION.                                                 30282003000007
ENDBIB          5                                                30282003000008
COMMON          1           3                                    30282003000009
ML1        ML1-ERR                                               30282003000010
MIN        MIN                                                   30282003000011
30.                                                              30282003000012
ENDCOMMON       3                                                30282003000013
DATA            2           1                                    30282003000014
DATA       DATA-ERR                                              30282003000015
MB         MB                                                    30282003000016
12.        2.                                                    30282003000017
ENDDATA         3                                                30282003000018
ENDSUBENT      17                                                30282003999999
```

Example 2

"EDITED" LISTING

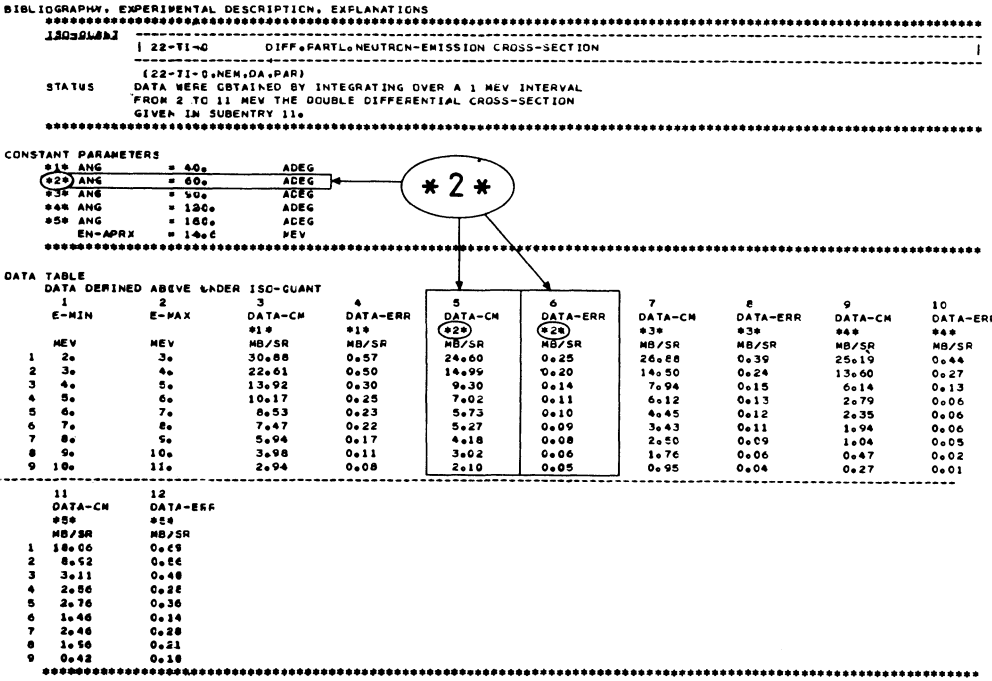


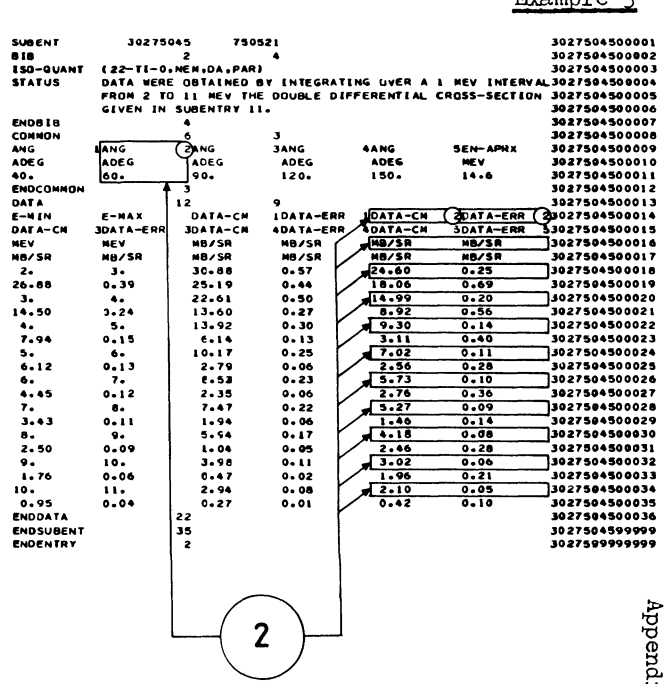
"STANDARD" LISTING



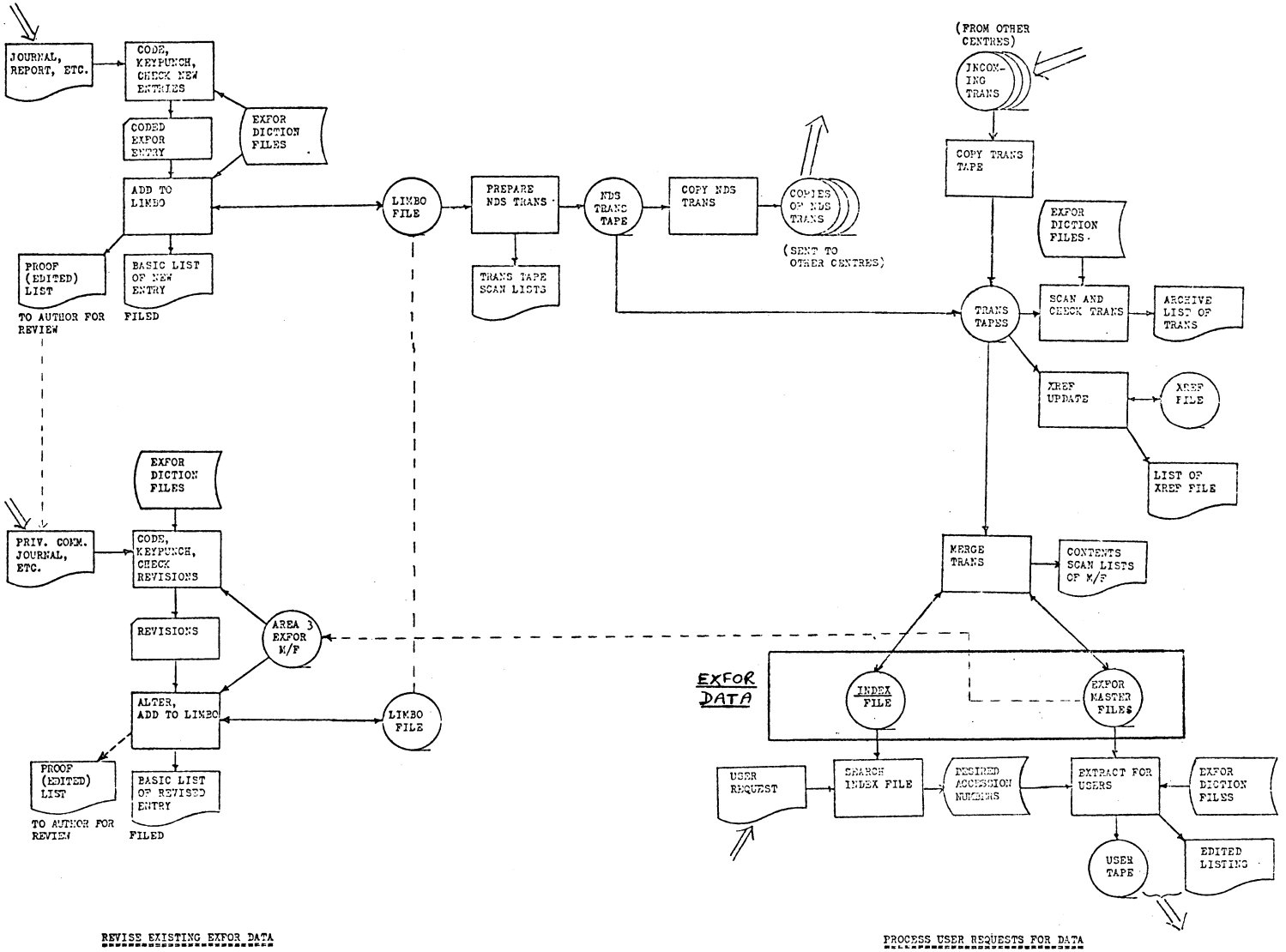
Pointers link related pieces of numerical and/or text information. In this example a pointer (e.g. 3) links an ISO-QUANT with its corresponding data column.

Example 3





In this example a pointer links an angle and the corresponding differential cross-section. Note that tables with more than 6 columns which are tedious to decipher in "standard" format, are clearly presented in the "edited" listing.


COMPILE NEW EXFOR DATA
TRANSMIT NEW, REVISED EXFOR DATA
PROCESS INCOMING AND NDS TRANS TAPES
JOURNAL, REPORT, ETC.
CODE, KEYPUNCH, CHECK NEW ENTRIES
CODED EXFOR ENTRY
EXFOR DICTION FILES
ADD TO LIMBO
PROOF (EDITED) LIST
TO AUTHOR FOR REVIEW
BASIC LIST OF NEW ENTRY
FILED
LIMBO FILE
PREPARE NDS TRANS
TRANS TAPE SCAN LISTS
NDS TRANS TAPE
COPY NDS TRANS
COPIES OF NDS TRANS
(SENT TO OTHER CENTRES)
(FROM OTHER CENTRES)
INCOMING TRANS
COPY TRANS TAPE
EXFOR DICTION FILES
TRANS TAPES
SCAN AND CHECK TRANS
ARCHIVE LIST OF TRANS
XREF UPDATE
XREF FILE
LIST OF XREF FILE
MERGE TRANS
CONTENTS SCAN LISTS OF M/F
EXFOR DATA
INDEX FILE
EXFOR MASTER FILES
USER REQUEST
SEARCH INDEX FILE
DESIRED ACCESSION NUMBERS
EXTRACT FOR USERS
EXFOR DICTION FILES
USER TAPE
EDITED LISTING
EXFOR DICTION FILES
PRIV. COMM. JOURNAL, ETC.
CODE, KEYPUNCH, CHECK REVISIONS
REVISIONS
AREA 3 EXFOR M/F
ALTER, ADD TO LIMBO
LIMBO FILE
PROOF (EDITED) LIST
TO AUTHOR FOR REVIEW
BASIC LIST OF REVISED ENTRY
FILED
REVISE EXISTING EXFOR DATA
PROCESS USER REQUESTS FOR DATA

THE NEA COMPUTER PROGRAM LIBRARY: A POSSIBLE GDMS APPLICATION

W. Schuler, NEA Computer Program Library, Ispra, Italy

Abstract NEA Computer Program Library maintains a series of eleven sequential computer files, used for linked applications in managing their stock of computer codes for nuclear reactor calculations, storing index and program abstract information, and admininstering their service to requesters. The high data redundancy between the files suggests that a data base approach would be valid and this paper suggests a possible 'schema' for a CODASYL GDMS.

1. Introduction: The NEA Computer Program Library

Since 1964, the Nuclear Energy Agency of OECD has operated in Ispra, Italy, a Computer Program Library (CPL) which as its principal assignment collects and disseminates computer programs in the area of nuclear reactor design. To inform users of the programs available from the collection, descriptive catalogues going into different levels of detail are prepared and published periodically and distributed to all users of the service. In addition, CPL is engaged in a number of supplementary activities, such as the organisation of specialist meetings on selected topics of computer code application and a Service on Experience of Code Utilisation. The services of CPL are open to all member countries of NEA, including the United States and Canada. By a special arrangement with the International Atomic Energy Agency in Vienna, non-NEA countries may also participate. Institutions interested in the service, e.g. research centres or commercial firms may apply for nomination and be registered as member establishments. The CPL is supported by government contributions. No fees are charged to member establishments.

A proposal has been made to transfer the services now carried out by the Library to a new NEA Data Bank in Saclay, France, where a CODASYL GDMS would be implemented (initially IDMS on an IBM 370/125 computer, and later the very similar DBMS-11 system on a DEC PDP 11/70.

2. CPL Operations

This presentation will deal only with CPL activity in computer program collection, storage and redistribution. The physical unit that is managed by CPL is the "program package". The complete package consists of the program source deck, mostly in Fortran, input data and corresponding output of one or more typical problems, plus if necessary such additional data as cross section libraries, and documentation. In general, the package should be complete to such an extent as to render the program

operable as much as possible independent of any special computer environment.

The program package undergoes a series of operational phases from acquisition by CPL to redistribution to a requester:

(a) The program is offered by a member establishment to CPL.

(b) On the basis of documentation supplied by the author, CPL decides if the offered program lies within the subject scope of CPL and if it may be of general interest. It may then request the program for inclusion in the collection. In practice, CPL frequently takes own initiatives by requesting programs from their authors the existence of which it had known through other sources than through a direct offer by the author.

(c) The program arrives at CPL.

(d) In general, programs are only redistributed after they have successfully been tested. This testing which is normally performed by CPL staff, consists in a check for completeness of the package according to the definition given above and a re-run of the typical cases.

(e) The tested program package is stored in a standard form on a master-tape.

(f) The program may be requested by a member establishment. In case that it has not yet been tested, it will be tested upon arrival of a first request for it.

(g) Tested and requested programs are then dispatched, as tape copies from the master-tape. Documentation is also included.

For administrative purposes, each of these phases is recorded on one or more computer files.

For the descriptive catalogues, an abstract consisting of 18 standard abstract items and an 80-byte short description are also prepared for each program and stored on tape. From these tapes, abstract folders and KWIC index booklets are prepared twice each year and mailed to all members. They are the main reference catalogues of the program collection.

3. Bookkeeping and Publication files

Both the customer information services of CPL and its informal administration have progressively been more computerised so that there now exists the typical situation where a total of eleven sequential tape files have to be updated periodically. The files, moreover, contain a considerable amount of redundant data, i.e. information stored identically several times on different files. Table I illustrates schematically this redundancy. The first two columns lists record types which will be used later in the data base schema proposed. Columns 3 to 13 represent the files used in the present system. At the intersections marked "x", the same data exist on a CPL file and a data base record. Where more than one "x" appears on the same line, data are redundant.

Abstracts file. This file contains abstract descriptions of each program in a standard format under 18 subject items, e.g.

- Nature of Physical Problem Solved

- Method of Solution

- Hardware requirements

- Programming Language,

etc.

Within each item paragraph free-format text is used. The file is updated twice per year and edited and published as "Nuclear Program Abstracts" folders.

KWIC file. For each program, an 80-byte description which includes program-name, abstract-number and subject category under which the program falls, is stored on this file. A KWIC Index is prepared from this file once or twice each year and published in booklet form.

Index file. This file contains an index to announcements or notes on programs, given in the bulletin "NEWS from CPL" which is published four times each year.

Program file. Each 80-byte record of this file describes the physical contents and availability status (arrived, tested, etc.) for one program package. It should be noted that one program abstract may cover more than one program package. Different packages of the same program normally represent versions written for and/or running on different computers. The file is updated every week.

Request file. All requests for programs made by member establishments are recorded on this file for the time that the request remains pending. The file is updated every week.

Dispatch file. When a program requested has been mailed, the corresponding entry is transferred from the request to the dispatch file. The file is updated every week.

Obsolete programs file. With the mutual agreement of program authors and users, CPL from time to time decides to delete old programs from the "active" Program file. In order that these programs shall remain traceable they are transferred to this file. The file is updated at most once every year.

Installations file. This file contains all member establishment abbreviation codes, liaison officer names and addresses and in addition special mailing instructions such as number of copies to be sent to a particular member for a particular publication. The file serves to produce address stickers and to calculate postage fees. It is normally updated before bulk dispatches to all members are planned, i.e. approximately ten times per year.

Computing Facilities file. Here details about the computer equipment available to each member establishment are stored. The format is similar to that of the Abstracts file. At present, no publication is prepared from this file. It serves only for internal consultation by the Library.

Tape File. The file contains for all "archive" tapes - tapes containing original program material - the shelf numbers where they can be found. The file is used to assign cupboard space and to remove tapes no longer used from the archive. The file is updated every month.

Dispatched tape file. Here numbers and destinations of all tapes used for dispatches are stored. It serves to recover CPL proprietary tapes sent out. The file is updated every month.

4. Size of CPL Operations

I would now like to give some figures which reflect the size of the CPL services. These figures will then serve to estimate access frequencies to various data items of the data base which I will then propose.

- The latest published catalogue lists about 1100 program packages which are currently available.
- To this figure, about 100 to 150 new programs have to be added each year.
- About 100 programs are also tested each year.
- In 1976, about 750 complete packages were mailed and in addition about 250 program reports.
- The user community of CPL now comprises about 350 registered establishments.

5. Integrated Data Base Management System

In order to avoid the disadvantages of the file system described, the schema of Fig. 1 for a data base is proposed. Its logical structure is such as is supported by the system IDMS of Cullinane Corporation, and the nomenclature used is that of this system. The data base itself can to a large extent be generated from the data now available on tape files.

The following conventions have been used when drawing up the schema:

- A square box represents a record type.
- Within a record type,

  the first line gives the record-name

  the second line contains the record-id and

  the location-mode: CALC - the record location is calculated from the data-item within the record as given on the third line by a hash algorithm

  VIA - all record occurrences are stored near the owner record occurrence within the set given on the third line.

- A pair of record types connected by an arrow represents a "set". The arrow points from the owner to the member of the set.

Table II lists and describes all record types and estimates their occurrences within the data base, their total lengths and the number of accesses to them per month. For each record type, data items are described. Records which contain free-format data have variable

length. For these records, therefore, total lengths are maximum lengths.

Within the schema of Fig. 1 we note a principal logical division between the record type PROGRAM on the left and PACKAGE on the right side. One occurrence of PROGRAM represents one abstract of a program. In fact, abstract items (indicated by bracketed numbers) 3, 4, 5, 6, 7, 14, 15, 18 appear as member records in sets with PROGRAM as owner and are also stored VIA these sets. PACKAGE on the other side represents one program version as a physical unit in the whole collection. Consequently it is stored as a member within the set linking it to PROGRAM which allows for more than one PACKAGE occurrence for one PROGRAM (-Abstract) occurrence. Abstract items 8, 9, 10, 11, 13, 16 which may be specific to one particular program version are linked to PACKAGE and stored near their PACKAGE occurrence.

The scope of the CPL program collection is formally subdivided into a number of subject categories, which are the occurrences of the record type CATEGORY in the schema. CATEGORY is linked as set owner with PROGRAM. This means that a specific category, e.g. "Reactor Safety Analysis", is an owner occurrence of all PROGRAMs falling into the category "Reactor Safety Analysis".

In a similar manner, records ORIG-COMP (original computer), TEST-COMP, ORIG-LANG, and TEST-LANG are linked as set owners with PACKAGE.

To further illustrate the logical structure of the schema Fig. 1 , I will now discuss several practical examples.

1. Store a new package REQUEST.

   (a) Locate the PACKAGE occurrence containing the name of the program package requested.

   (b) Locate the requester INSTALLATION.

   (c) Store the new REQUEST record giving a request number and date.

In IDMS, since memberships have been defined OPTIONAL AUTOMATIC (OA), the REQUEST occurrence just stored is automatically linked with the PACKAGE and INSTALLATION occurrences located before.

   (d) If further specification of the material requested is necessary, all required occurrences of DOCUMENT and MATERIAL associated with the PACKAGE are located and in this case manually connected to the REQUEST occurrence.

   (e) If the request requires special comments, they may be stored in REQ-COM.

2. Record a package dispatch

   (a) Locate the REQUEST record in question by its request-no.

   (b) Insert into this REQUEST record, date and tape-no. of the dispatch.

3. Find all open requests for one requester INSTALLATION.

   (a) Locate the INSTALLATION occurrence by its abbreviation code.

(b) Find within the INSTALLATION-REQUEST set all member occurrences of REQUEST which have the located INSTALLATION as owner occurrence.

(c) To find the name of the package requested, locate likewise for each REQUEST the member occurrence of PACKAGE.

(d) Find for each REQUEST possible occurrences of DOC, MATERIAL and REQ-COM.

6. Conclusion

The introduction of an integrated data base management system offers a number of obvious advantages over the present file-based system:

- The maintenance and editing of the Program Abstracts file in the present form is a time-consuming task requiring one to two man-months per year of senior staff. The updating of this file has therefore been mainly restricted to the insertion of new abstracts for new programs. If abstracts data are integrated into a data base they will automatically be brought up to date whenever changes to package descriptions have to be made. The preparation of the publication itself could be reduced to a simple report generation from the data base. In this way, the value of the publication as a general reference manual could be considerably enhanced. At the same time a saving in manpower may be possible.

- In general, the reduction or even complete elimination of data redundancy would render system maintenance easier, more efficient and less error-prone. At this point it should be noted that due to the staff structure and resources of CPL, most of the staff have to devote part of their time to some kind of file maintenance. This constitutes a burdensome overhead to other assignments and could no doubt be considerably reduced with the introduction of an easily manageable integrated system.

On the other hand, it is clear that the rather moderate demands on such a system, in particular as far as disk storage and access times are concerned, would probably not be enough to justify the introduction of highly sophisticated software only for the tasks which I have outlined above. The whole proposal should therefore be seen in a context where a data management system is likely to be already available for other more demanding projects.

The most important data missing from the schema presented are the CPL 'master files' of program packages tested and available for distribution. The volume of source code stored on tape is large and, once tested, a program may be considered simply as a block of text to be copied for the benefit of requesters. In view of their large volume and apparent lack of informal structure, the CPL master files would at least initially continue to be stored on tape, outside the data base.

Table I

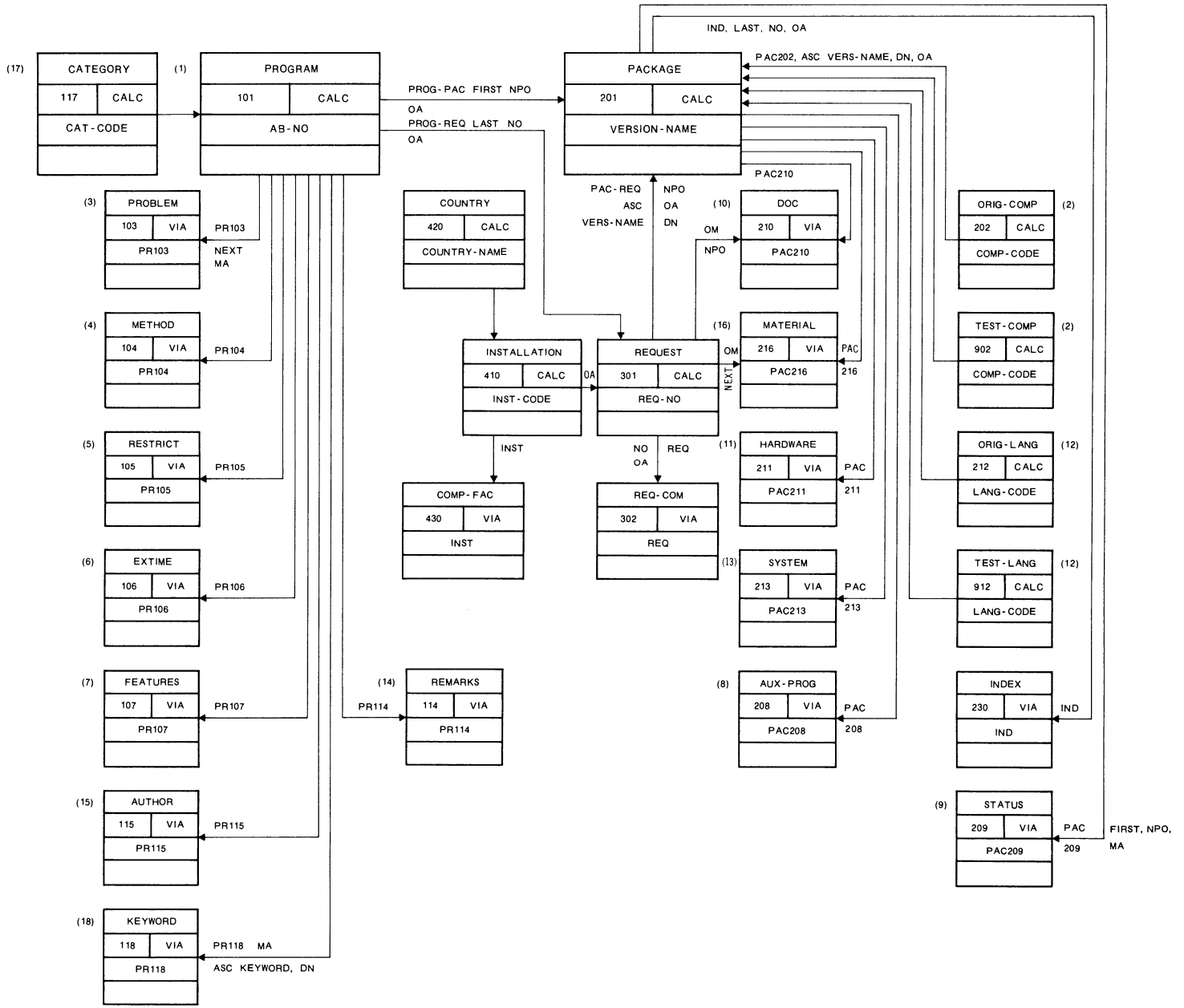
DATA REDUNDANCY BETWEEN CPL FILES

Rec Id	Record-name	Abstr.	Prog.	Req.	Disp.	Obs.	KWIC	Index	Inst.	Comp.	Tape	Disp-tape
101	PROGRAM	x	x	x	x	x	x	x			x	
103	PROBLEM	x										
104	METHOD	x										
105	RESTRICT	x										
106	EXTIME	x										
107	FEATURES	x										
114	REMARKS	x										
115	AUTHORS	x										
118	KEYWORDS	x										
117	CATEGORY	x	x				x	x				
201	PACKAGE		x	x	x	x		x			x	
202	ORIG-COMP	x	x	x	x	x						
902	TEST-COMP	x	x	x	x	x						
208	AUX-PROG	x	x				x	x				
209	STATUS	x	x			x		x			x	
210	DOC	x										
211	HARDWARE	x										
212	ORIG-LANG	x	x	x	x	x					x	
912	TEST-LANG	x	x	x	x	x					x	
213	SYSTEM	x										
214	REMARKS	x										
216	MATERIAL	x	x			x					x	
301	REQUEST			x	x							x
410	INSTALL.		x	x	x	x	x		x	x		x
420	COUNTRY								x	x		
430	COMP-FAC									x		

x *Redundant information*

Figure 1

PRELIMINARY IDMS SCHEMA



REC ID	record-name	loc mode	data item	description	length (bytes)	total (kbyte)	occ	ac /mth
100	MASTER	CALC		publications record			1	
			NEADTE	date of last NEA abstracts	6			
			NEATYPE	type of issue: 1 - update 2 - complete	4			
			NEALAST	last abstract no. published	5			
			USDTE	date of last US abstracts	6			
			USTYPE	type of issue	4			
			ULAST	last USCC abstract no. publ.	5			
			CLAST	last RSIC abstract no, publ.	5			
			KWICDTE	date of last KWIC issue	6			
			NEWSDTE	date of last NEWS issue	6			
					47	.05		
101	PROGRAM	CALC AB-NO		logical program unit			1500	15
			PROG-NAME	name of computer program	14			
			AB-NO	abstract no.	5			
			KWIC	KWIC title	65			
					84	6		
103	PROBLEM	VIA PR103		Nature of physical problem solved			1500	15
			PROB	(free-format description)	2000	3000		
104	METHOD	VIA PR104		Method of solution			1500	15
			METH	(free-format description)	1500	2250		
105	RESTRICT	VIA PR105		Restrictions on the complexity of the problem			1500	15
			RESTR	(free-format description)	1500	2250		
106	EXTIME	VIA PR106		Typical running time			1500	15
			EXTME	(free-format description)	1000	1500		
107	FEATURES	VIA PR107		Unusual features of the program or restrictions			200	15
			FEAT	(free-format description)	1000	200		
115	AUTHOR	VIA PR115		Name and establishment of author			3000	15
			NAME	Author name	50			
			ADDRESS	Address	200			
					250	750		
118	KEYWORD	VIA PR118		Descriptor			500	15
			KWORD	Keyword	50	25		
201	PACKAGE	CALC VERSION-NAME		Physical code package description			2000	400
			VERSION-NAME	Name of program package	14	28		

TABLE II

REC ID	record-name	loc mode	data item	description	Length (bytes)	total (kbyte)	occ	ac /mth
214	REMARKS	VIA PAC214		Any other programming or operating information or restrictions			500	15
			OP	(free-format description)	1000	500		
216	MATERIAL	VIA PAC216		Material available			8000	300
			MAT	material code: S - source deck D - test case data L - test case printout P - test case punch T - library data BCD C - library data binary M - load module X - auxiliary program(s) R - report W - working description	1	8		
229	STATCOM	VIA ST		Remarks on availability status			500	20
			COM209	(free-format text)	500	250		
239	DOCOM	VIA DC		Remarks on documentation			1000	100
			COM210	(free-format text)	500	500		
249	MATCOM	VIA MA		Remarks on available material			800	300
			COM219	(free-format text)	500	400		
301	REQUEST	CALC REQ-NO		program request record			5000	200
			REQ-NO	Request sequence no.	4			
			REQ-DTE	Date of request	6			
			DISP-DTE	Date of dispatch	6			
			DISP-TAPE	Tape # mailed	16	80		
302	REQ-COM	VIA REQ		Remarks on request			500	200
			COM301	(free-format text)	500	250		
410	INSTALLATION	CALC INST-CODE		Member establishment			300	20
			INST-CODE	Member establishment code name	10			
			LIAISON-OFFICER	Name of Liaison Officer	50			
			ADDRESS	Mailing address of L.O.	200			
			SPECIAL	Flag for IAEA or other special status	4			
			MAIL	Special mailing details	4			
					268	80		

TABLE II (Contd.)

REC ID	record-name	loc mode	data item	description	Length (bytes)	total (kbyte)	occ	ac /mth
(202	COMPUTER)	VIA PAC202		Computer on which program is operable			2000	40
			COMP1	Computer for which the program is designed	10			
			COMP2	other computer on which the program is running	10			
					20	40		
208	AUX-PROG	VIA PAC208		Related and auxiliary programs			500	20
			NAME	Program name	14			
			DESCR	Program description	486			
					500	250		
209	STATUS	VIA PAC209		availability status of code			4000	100
			STATDTE	date of status change	6			
			STATCODE	status code: O - offered D - requested A - arrived P - prepared T - tested S - test suspended X - not to be tested	1			
			TAPE	Tape no.	4			
			BOX	place where tape is stored	4			
			TCODE	code letter of tester				
			ICODE	installation code of testing establishment	10			
					26	104		
210	DOC	VIA PAC210		Documentation record			4000	200
			REF	Reference no. of document	20			
			AUTHOR	Author name	50			
			DOCSTAT	availability status: D - requested T - distributed if STATCODE =T X - distributed if STATCODE ≠T A - available but not distr. N - not distributed	1			
			COPIES	no. of copies on stock	4			
			DOCDTE	date of status change	6			
					81	324		
211	HARDWARE	VIA PAC211		Machine requirements			1500	15
			HARD	(free-format description)	500	750		
(212	LANG)	VIA PAC212		Programming language used			2000	30
			LANG1	language of original program	10			
			LANG2	language of adaptation	10			
					20	40		
213	SYSTEM	VIA PAC213		Operating system or monitor under which program is executed			2000	30
			SYST	(free-format description)	500	1000		

TABLE II (Contd.)

REC ID	record-name	loc mode	data item	description	length (bytes)	total (kbyte)	occ	ac /mth
420	COUNTRY	CALC		Member country			50	1
		COUNTRY-NAME						
			COUNTRY-NAME	Country name	46			
			SPEC	Flag for IAEA or special stats.	4			
					50	3		
430	COMP-FAC	VIA INST		Computing Facilities			50	100
			TEXT	(Standard-format description)	8000	400		
117	CATEGORY	CALC CAT		Subject category			30	20
			CAT	Category code letter	1			
			CAT-TITLE	Category title	200			
			CAT-DESC	Category specification	500			
					701	21		
230	INDEX	VIA IND		NEWS from CPL Index			3000	100
			NEWS-NO	Issue no. of NEWS from CPL	6	18		

TOTAL STORAGE REQUIREMENT FOR DATA BASE: = 14269 kbytes

TABLE II (Contd.)

COMPUTERIZED DATA HANDLING IN THE ENVIRONMENTAL CHEMICALS DATA AND INFORMATION NETWORK

*J.H. Petrie, J. Powell, W.G. Town**
Commission of the European Communities
Joint Research Centre - Ispra Establishment
21020 Ispra (Va) - Italy

Introduction

ECDIN (Environmental Chemicals Data and Information Network) is a pilot project to study the feasibility of setting up an information network for chemical substances and their effects on the environment. The project will constitute a valuable contribution to the European Communities EMIN (Environmental Management Information Network). ECDIN is a research project of the European Communities; work is being done both by the Joint Research Centre of the EC and by institutions in the EC member states under contract within the framework of the Environmental Research Programme. Data being collected by these institutions are being brought together at the EC Joint Research Centre in Ispra, Italy.

The aim of the project is to establish a data bank containing the information required for decision making in environmental management. In addition to establishing which data elements are necessary for environmental impact assessment and control, consideration has been given to the chemical compounds to be included. In our view, effective control of environmental chemicals depends not only on the monitoring at known environmental stresses, but also on the systematic collection and organization of

- a) all chemicals manufactured in large quantities,
- b) all toxic chemicals which are manufactured,
- c) all metabolites and degradation products of compounds in a) and b) and by products resulting from their manufacture.

It is estimated that in an operational system these criteria could lead to a file of about 30,000 compounds. However, in the pilot phase we have limited the file to 5000 compounds.

SIMAS information retrieval system

From the start of the project it was decided to set up a computerized system using the SIMAS information retrieval system which was developed at JRC-Ispra. Although it was already clear in 1972 that SIMAS was not an ideal system for ECDIN, it had the advantage that, as it was a local system, limited improvements to the system for the ECDIN application were possible.

SIMAS was originally designed for the library of computer programs of EUROCOPI (European Computer Programs Institute). SIMAS allows the system designer to set up a number of **classes** each containing **objects** which may be **catalogued**. In addition, **keywords** and **searchable identifiers** may be assigned to the 'objects'.

---

* *To whom all queries should be addressed*

In the ECDIN implementation we chose to use one 'class' in which the 'objects' were chemical compounds as it was not possible to search across classes. One implication of this decision was that the logical record of the ECDIN file would have a hierarchical structure with the chemical compound as the root of the tree. The problems arising from this record structure will be considered later in greater detail.

To enable the data stored in SIMAS to become retrievable, it was necessary to develop a thesaurus for ECDIN. The SIMAS system contains procedures for thesaurus construction and maintenance. The thesaurus is organized in a number of **broad keyword groups** each of which may contain a number of **narrow keyword groups.** Each 'narrow keyword group' may in turn contain a hierarchy of keywords. One improvement of the SIMAS system introduced for the ECDIN implementation was the ability to associate numerical values with keywords and to search them with operators such as: **equals, less than, greater than, less than or equal to, greater than or equal to, between** (for ranges), **error** (to allow for uncertainty in data). Further the units in which the original measurement was made, could be input and automatically translated into a standard unit. This facility was also made available to the searcher.

ECDIN input format

In addition to imposing a hierarchical data structure on ECDIN, the SIMAS system imposes other severe constraints. Firstly, in SIMAS we could not easily represent in fine enough detail the data structure which we felt to be appropriate to ECDIN. Secondly, the facilities for data management in SIMAS (like most other information retrieval systems) were minimal. In order to change one digit in the data record for a compound, it was necessary to reload the whole display file for the compound. Similarly in order to change or add one keyword, it was necessary to reload all keywords for a compound.

As a result it was decided from the outset to develop an ECDIN input format which would have the following advantages:

- a) the ECDIN data bank was not too dependent on SIMAS and the change to a new system would be facilitated,
- b) there would be more flexibility to represent the data as it should be stored,
- c) the design of the format could be improved as a greater understanding of the inherent data structure was obtained.

However, the ECDIN input format was of necessity still conditioned by the constraints of the SIMAS system and we were forced to adapt a hierarchical data structure for the ECDIN input format record.

Data structure of the ECDIN record

The data elements considered appropriate to the aims of ECDIN were organized into ten **categories**, each of which was divided into **fields** and, where necessary, **subfields.** The ten data categories are listed in Fig. 1 and the field structures of two of these categories are shown in Figs. 2 and 3. In some cases a field may occur once only (e.g. preferred systematic name in category 1) while in other cases the field may be repeated (e.g. trade names in category 1). For some fields it has been necessary to introduce a repeating group of subfields. For example, a chemical compound may have many producers and for each producer we may wish to record the following data elements:

- company name
- plant location
- plant capacity
- process used
- merchant capacity
- material source

Each of these data elements becomes a subfield in a group of subfields which describes a producer and the group may be repeated as many times as there are producers for the compound (see Fig. 4). Often the hierarchical structure inadequately represents the true data structure. There may be a need to refer from one field to another, as is the case with chemical processes in the example above or it may be necessary to refer to other compound records.

In certain fields (mostly in categories 8 and 9) a free text **condensate** is used to present an abstract of the state-of-the-art for the compound in the field. Several bibliographic references may be used in preparing the condensate and these are listed after the condensate (see Fig. 5). Each element of the bibliographic reference is tagged as a subfield. The concept of the 'condensate' was introduced partly to overcome the space limitations

1. Identification of the chemical (IDN)
2. Chemical structure information (CSI)
3. Physical and chemical properties (PCP)
4. Chemical analysis data and methods (CAD)
5. Supply - production and trade (SPL)
6. Transport, packing, handling, storage and hazards (TPH)
7. Use and disposal (USE)
8. Dispersion and transformation in the environment (DTE)
9. Effects of the chemical on the environment (incl. toxicity) (TOX)
10. Regulatory data (CRR)

Fig. 1 : ECDIN data categories

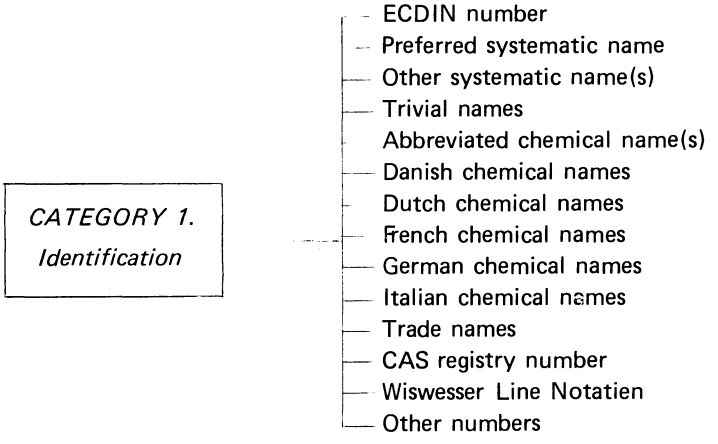


Fig. 2 : Field structure of category 1

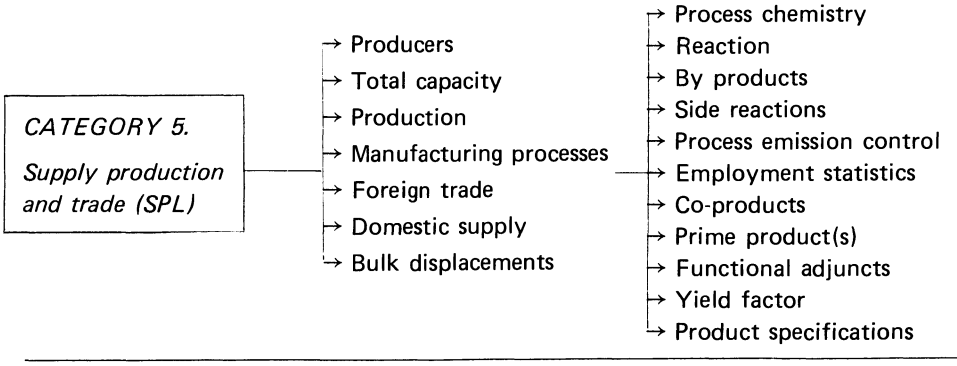


Fig. 3 : Field structure for category 5

Only the data for the field "manufacturing processes" are expanded. Data for other fields are similarly divided.

Company name	:	Montedison
Plant location	:	Porto Marghera, I-30100
Plant capacity	:	7 KT
Process used	:	No. 1 (Cross reference)
Plant location	:	Villadossola, I-28039
Plant capacity	:	7 KT
Process used	:	No. 1 (Cross reference)
.		.
.		.
Company name	:	BASF
Plant location	:	Ludwigshafen, D-6700
Plant capacity	:	20 KT
.		.
.		.

Fig. 4 : Example of data : chemical producers

CONDENSATE

Primary clinical pathological changes are formation of methaemoglobin with resulting cyanosis at 10 - 15% conversion and anoxemia at about 30% conversion . . . . . failure and death (1). Absorption through skin is frequently main route of entry . . . . . several hours exposure (2).

References

1. International Labour Office. Encyclopaedia of Occupational Safety and Health. ILO, Geneva, 1971 (2 Vol.), p. 98

2. etc.

Fig. 5 : Example of data for category 9 field: "Effect on Man"

of the SIMAS field. In the future, a structured data representation will be introduced in some of these fields and the 'condensate' will be reserved for comment on or evaluation of the data.

Present data management system

As a consequence of adapting the ECDIN input format, it was necessary to develop a conversion program which would reformat ECDIN data into SIMAS input format. During the conversion process much of the fine structure of the record is lost. Since data management in SIMAS is difficult and since, in any case, we would have lost the fine structure of the record in SIMAS, it was also necessary to develop file maintenance routines for the ECDIN input format.

It is not necessary here to describe in detail the file maintenance system but the following list of system elements will show that it is non-trivial:

- sorting and merging of files in ECDIN input format
- error detection routines
- editing routines
- file statistics
- selection of data by compound
- selection of data by data field
- creation of file subsets according to characteristics of data records
- maintenance routines of ECDIN compound registry file (an authority file)

The maintenance of two sets of files is wasteful of time and effort. Furthermore, the file conversion and retrieval file updating are expensive and complex procedures and as a result the retrieval file is updated at infrequent intervals.

Summary of reasons for changing to DBMS

The advantages to be gained from a change to DBMS are of two types:

- elimination of the disadvantages of SIMAS,
- improvements in retrieval and data management offered by DBMS.

One of the chief disadvantages of SIMAS is the imposition of a hierarchical data structure. This results in redundancy of information (e.g. producers, processes, bibliographic references) and as a result of this redundancy it is difficult to ensure that exactly the same form of, for example, a producer's name is stored everywhere in the data base. Redundant storage of information can also lead to increased updating since, for example, when a company name changes, all records in the data file containing the name must also be changed. Furthermore, the hierarchical structure distorts the true data structure causing problems of cross referencing as already mentioned.

Even though the association of values with keywords was an improvement in SIMAS, this feature is still inadequate to deal with all relationships which should become searchable. Consider, for example, the concept of 'production in a region'. Here there are three values or attributes related to this concept, namely:

- region or country
- year of production
- quantity of production

As SIMAS allows the association of only one value with a concept, we are forced to multiply the number of keywords used to represent this relationship in SIMAS. For example, we could do this with keywords of the following type:

- production in Italy in 1977
- production in Italy in 1976, etc.
- production in France in 1977, etc.

This is clearly inadequate for a data bank such as ECDIN. The positive gains to be expected of a DBMS are:

- better representation of the ECDIN data relationships in a network structure
- more efficient data management
- greater flexibility to change data structures
- dynamic hierarchy definition.

An idea of the complexity of these data relationships is shown in Fig. 6.

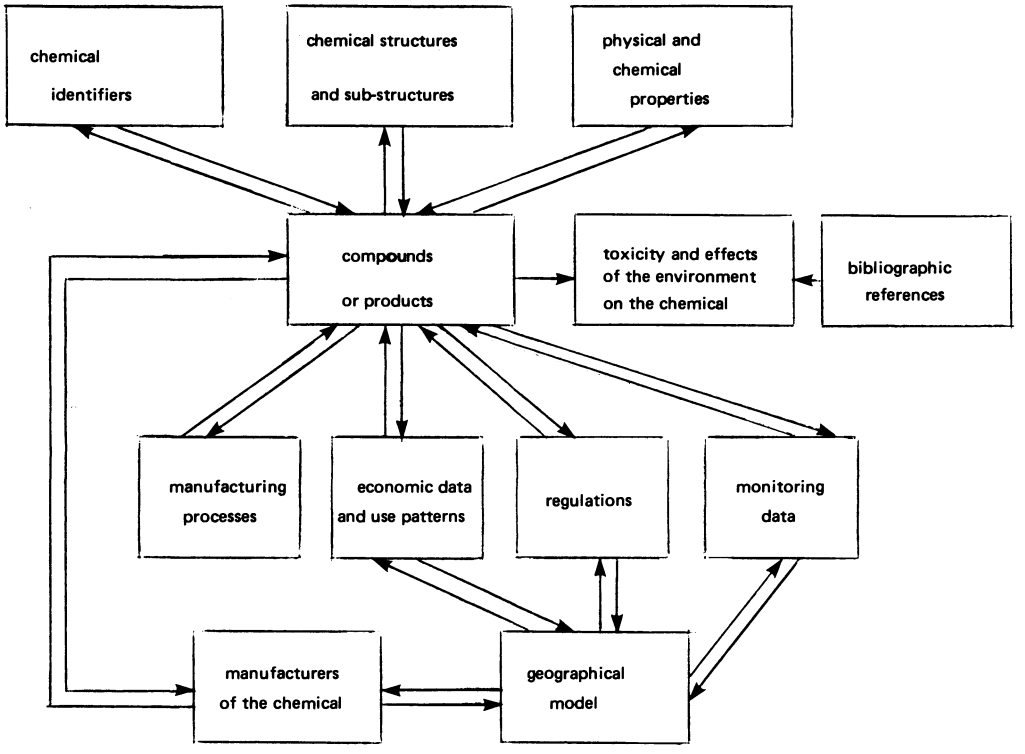


Fig. 6 : A model of ECDIN using a network data structure

With such a model of ECDIN we would be able to answer more easily questions directed to the manufacturers, to chemical processes, to administrative regions or eventually even to hydrographic basins. Clearly, as before, questions directed to the chemical compound will be of importance. Obviously, relationships between compounds also exist and the above diagram should not be regarded as exhaustive.

Study of software needs for JRC-Ispra data banks

In view of the obvious inadequacies of SIMAS for ECDIN and the need to replace SIMAS with a new system, a study contract was awarded to an external consultant who was given the task of examining the six data banks which are proposed or operating in the JRC-Ispra and, in the light of the available commercial software, of making recommendations for an eventual replacement for SIMAS. As a result of the first part of the study, a class of DBMS software (inverted file systems) was recommended as a replacement of SIMAS. In the second part of the study, three of the JRC-Ispra data banks (including ECDIN) were examined in turn with each of three DBMS packages (ADABAS, INQUIRE, SYSTEM 2000) to determine the problems likely to arise. The choice between ADABAS and INQUIRE was difficult but on the grounds of easier extendability, the former system was chosen. Trials with ADABAS at JRC-Ispra should begin in the last quarter of 1977.

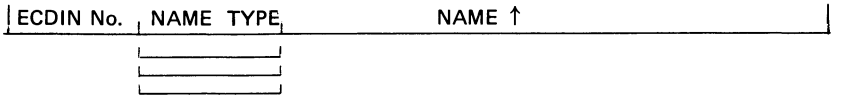
Conversion of ECDIN to ADABAS

The existence of the ECDIN input format should facilitate the conversion of ECDIN to ADABAS. A study of the problems involved for certain parts of the data base has already begun and a number of computer programs has been written to facilitate the conversion process. To ensure a realistic test of ADABAS, it was decided to select data according to the following criteria:

- data required to be searchable but which are not searchable at present,
- data which are not adequately searchable at present,
- data which would test the features of ADACOM (a new command language available with ADABAS),
- data having a close association which would enable realistic questions to be asked.

Accordingly, the initial conversion will be made on chemical names, producers and chemical processes.

Consideration of the characteristics of the data in ECDIN category 1 (identification) and the limitations of ADABAS system has led to the proposal of the following record structure for chemical names:

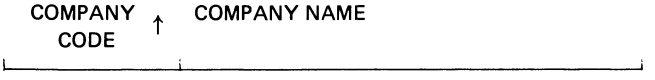


i.e. one record for each name with a multiple field for name type. The following requirements are satisfied by this structure:

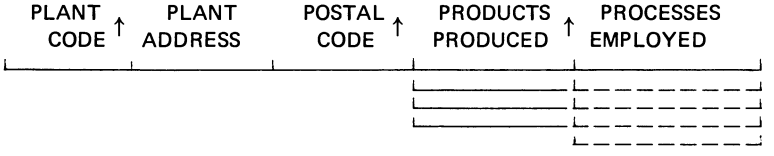
- 1) to be able to search for chemical compounds using all chemical names present in the data base,
- 2) to be able to identify chemical names by name type,
- 3) to link chemical names with other files containing data which relate to chemical compounds (i.e. production, chemical processes, etc.)

The data in ECDIN input format on chemical producers is stored redundantly in each compound record and, as a result, the company names and plant locations must be standardarized before we can create separate company and plant files. Computer programs have been written to sort the existing data by company name, plant location and ECDIN number and to convert various non-standard forms of company name and plant addresses into a standard representation. These programs can be easily modified to produce files for direct input to ADABAS having the following record structures:

Company file record structure



Plant file record structure



The **plant code** is a combination of the owning company code and an "idiot" number. The company code portion of the plant code could be used for direct coupling of the Plant and Company Files.

The **products produced** are the ECDIN numbers of the compounds produced at that plant.

The following search requirements may be satisfied with these two record structures:

- 1) to search for where a chemical compound is produced,
- 2) to search for what companies produce a chemical compound,
- 3) to search for what chemical processes a plant employs,
- 4) to search for what chemicals are produced in a certain country/region/area,
- 5) to link chemical producers with other files containing data which relate to chemicals and their production (i.e. chemical names, chemical processes).

The existing data in ECDIN input format on chemical processes also requires standardization but it is inadequately indexed in terms of cross references to other compounds. Rather than to convert this data by program, a new format has been designed and the data are being manually recoded into this new format. As the volume of this file was small, it was felt that programmed conversion was not justified in this case. However, a selected data printout is being used to facilitate recoding. Once this file entered into ADABAS it should be easier to eliminate redundancy and standardize process descriptions. The record structure developed for this file is shown in Fig. 7.

Hence, in many cases direct conversion from the ECDIN input format has proved feasible and even when this is not the case, the conversion process may still be greatly facilitated.

Acknowledgements

The authors would like to thank the many organisations and individuals who have contributed to the ECDIN project.

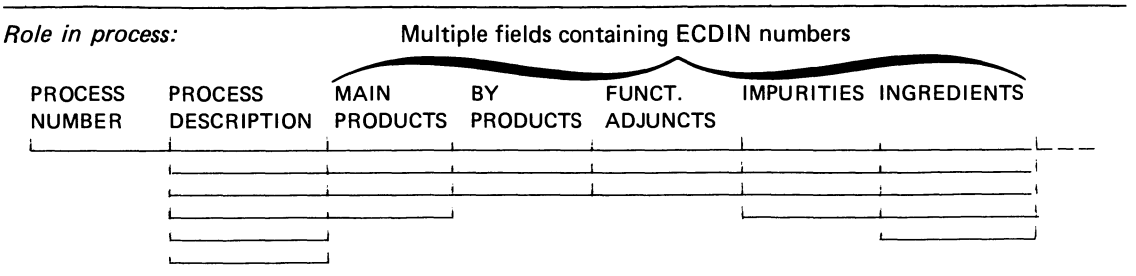


Fig. 7 : Record structure for chemical processes

PART IV
THE DIRECTION OF GDMS DEVELOPMENT

PARTIE IV
LE DEVELOPPEMENT DES SGBD DANS L'AVENIR

THE RATIONALE OF A STANDARD INTERCHANGE FORMAT

A. A. Brooks

Computer Sciences Division
at Oak Ridge National Laboratory
Union Carbide Corporation, Nuclear Division*
Oak Ridge, Tennessee, USA

ABSTRACT

The proposed draft, *American National Standard Specifications for a Data Descriptive File for Information Interchange*, which is in development by the ANSI/X3L5 committee is described. The standard is suited for R&D data interchange in an ASCII character mode.

The interchange of information between n dissimilar information systems can be accomplished by the construction of n(n-1) *ad hoc* interfaces between pairs of systems or by the construction of 2n interfaces between the systems and a suitable common interchange system commonly called an interchange format. The latter approach was adopted in 1976 after considerable debate by the ERDA Inter-Laboratory Working Group for Data Exchange. An interchange format was devised and reported[1] in that year and implemented in part during 1977. During the latter part of 1976 the American National Standards Institute Subcommittee on Labels and File Structure (X3L5) assumed further work in its project on Interchangeable ASCII Data Files and is currently working on a proposed draft standard.[2] This paper discusses the properties of such an interchange file.

It is assumed that the dissimilar systems have equivalent processing algorithms and sufficient capability for the expression of the inherent logical structure and form of the information; otherwise the act of interchange would be an exercise in futility. We set forth some necessary characteristics of the interchange format:

1. It must be capable of accepting from any system the atomic data elements to be interchanged in a form accessible to all systems.

---

*Prime contractor for the Energy Research and Development Administration

2. It must be capable of expressing the logical structure inherent in the information and convenient to the pragmatic organization of the information.

3. It must admit the construction of automated interfaces to all of the n-dissimilar systems, particularly to data base and file management systems.

4. It must offer processing efficiency as a desirable rather than an essential feature since the act of interchange is an occasional occurrence by comparison to internal processing.

5. It must be acceptable to a wide variety of users and must be extendable as new needs and data forms arise.

An examination of the wide spectrum of information to be interchanged reveals the following:

1. The atomic data elements are usually logically expressible as alphanumeric characters.

2. The most sophisticated inherent logical structures found are graphs, directed graphs and networks (in the mathematical sense); but these structures are represented by simpler structures and associated algorithms or by highly system-dependent pointers or access methods.

3. The vast majority of the data can be represented in structures no more complicated than rooted trees and often in simple vectors and/or regular arrays.

4. There exist several standard content-oriented interchange systems[3] for textual information all conceptually based on ISO 2709[4] and that the concepts therein can be generalized to media- and content-independent standard.

5. The most significant interchange media today is magnetic tape but is rapidly changing to other surfaces and to transmission.

6. Complex logical structures may be defined in terms of simple structures and are expressible in them (for example, a digraph is a set of elements and a relation on the set). In these simpler structures logical linkages are expressed as information values to be used in the formation of logical associations.

A consideration of the above objectives and observations led to the following conclusions:

1. The standard should be media-independent; i.e., it should specify the format of logical records which, when written on a specific media, can conform to the standards for that media.

2. The atomic data elements should be alphanumeric strings expressing text or numeric strings expressing numbers.

3. The atomic data elements should optionally be aggregated into vectors or arrays placed into fields which can further optionally be aggregated into a hierarchical (i.e. rooted tree) structure to comprise a logical record.

4. The logical records would be repetitive occurrences of the above structure whose description should accompany the interchange on the same media. The information records and the descriptive record should each constitute a separate file.

5. The format should attempt to include, insofar as reasonable, the existing standard systems of the ISO 2709 family as special cases.

6. ANSI X3.4-1968[5] should be used for control fields between systems which do not support the same character code set. The code extension techniques of X3.41-1974[6] should be adopted for data fields but not control fields.

The interchange standard can be described briefly by the following.

1. The standard draws upon the concepts of ISO 2709 for the logical record format composed of:

   a) A leader containing controls.

   b) A directory of field tags, printers and field lengths.

   c) Variable length data fields containing data elements.

2. An interchange is comprised of a pair of files:

   a) A data descriptive file which describes the data file giving i) a data base name and optional generic hierarchical structure information and ii) a field-wise description of each data field including field names, optional subfield names, data type, structure and format information as well as character code set.

   b) A data file composed of repetitive logical records comprised of data fields which are an instance of their description in the descriptive file.

   c) Multiple file pairs are permitted.

3. The data types permitted are text and the three numeric representations of X3.42-1975[7]; i.e., implicit point, explicit point and explicit point scaled.

4. The structures permitted as data elements are a) atomic, b) vector, and c) multiply-dimensional arrays of the allowed data types or mixtures thereof.

5. The data subfields are defined by formats or delimiters.

6. Any rooted tree structure of the fields within the logical record is optionally described by a preorder traversal sequence of the data tags in the directory.

7. A G1 extended character code set may be declared for the entire file or for each data field and further escape sequences are permitted.

8. An externally defined subsystem which conforms to the leader, directory and variable data field requirements of ISO 2709 can be declared in lieu of complete data field descriptions in the data descriptive file.

9. Three implementation levels are defined plus an extended character code set implementation.

10. The standard is designed for extension to new sets of data field descriptions and data elements.

Complex structures such as cyclic graphs may be fragmented into trees and interchanged by replacing "address pointers" with logical pointers. Acyclic digraphs such as rooted tree structures and simpler structures such as vectors, arrays, or relations can be transmitted directly. Relations can be transmitted as a set of vectors. Where very short, regular structures such as vectors, arrays, or relations are involved, a packing into pseudo-logical records may be desirable to reduce overhead.

The discussion by Date[8] of the equivalence of forms between the most prominent data base models suggests ways they can be converted into suitable interchange structures. These techniques presume the capability of the receiving system to house and reconstruct the structure. In short, the exchange format is a tool which like most sophisticated tools will require some reflection before use.

A Level 1 implementation for tape has been programmed for both IBM and CDC equipment at eight ERDA installations. The implementation is for a magnetic-tape environment and draws an ANSI X3.27-1977[9] as the tape label and file structure standard. The implementations provide interfaces for text streams from local DBMS (S2000 and ORCHIS) and "card image" input. File pairs of arbitrary level can be prepared by forming appropriate text strings. Documentation is being issued and implementation of "FORTRAN environment" input stream software for vectors and arrays is scheduled for next year. Automation of more complex forms is planned but presumes a DBMS equivalent and will be installation-dependent.

Copies of the current working draft of the proposed standard in microfiche form (48x) are available to interested parties from the author.

ACKNOWLEDGMENTS

The author wishes to acknowledge the partial support of implementation by the Savannah River Laboratory — Light Water Reactor Program and the personal efforts of C. Benkovitz, BNL; B. McNeely, ORNL; and R. Wiley, LASL, as well as the many contributions of the IWGDE members and X3L5 members to the concepts of the standard.

REFERENCES

1. Merril, Dean (ed.), *Annual Report of the Inter-Laboratory Working Group for Data Exchange,* Lawrence Berkeley Laboratory. (IWGDE has representation for Argonne National Laboratory, Brookhaven National Laboratory, Los Alamos Scientific Laboratory, Lawrence Berkeley Laboratory, Lawrence Livermore Laboratory, Oak Ridge National Laboratory, Pacific Northwest Laboratory and Savannah River Laboratory).

2. Brooks, A. A. (ed.), Draft Proposed: *American National Standard Specifications for a Data Descriptive File for Information Interchange*, ANSI-X3L5/646F, September 26, 1977.

3. Martin, M. D., *Reference Manual for Machine Readable Bibliographic Descriptions,* UNISIST SC.74/WS/20, UNESCO, Paris, 1974.

   INIS: Magnetic Tape Specifications and Record Format, IAEA-INIS-9 (Rev. 1); International Atomic Energy Agency, Vienna, 1971.

   American National Standard for Bibliographic Information Interchange on Magnetic Tape, Z39.2-1971; ANSI, New York City, 1971.

   UNIMARC - Universal MARC Format; International Federation of Library Associations and Institutes; London, 1977 (draft version).

   AEC/TIC Magnetic Tape Format for Bibliographic Citations and Indexing, TID-4581-R2; Technical Information Center, Oak Ridge, Tennessee, 1974.

4. *Documentation Format for Bibliographic Information Interchange on Magnetic Tape*, ISO 2709-1973(E); International Organization for Standardization, Paris, 1973 (available from ANSI).

5. *American National Standard Code for Information Exchange*, X3.4-1968; American National Standards Institute (ANSI), New York City, 1968.

6. *American National Standard Code Extensions Techniques for Use with a 7-Bit Coded Character Set of American National Standard Code for Information Exchange*, X3.41-1974; ANSI, New York City, 1974.

7. *American National Standards Representation of Numeric Values in Character Strings for Information Interchange*, X3.42-1975; ANSI, New York City, 1975.

8. Date, C. J., *An Introduction to Data Base Systems*, Addison-Westley Publishing Company, Reading, Massachusetts, 1975.

9. *American National Standard for Magnetic Tape Labels and File Structure for Information Interchange*, X3.27-1977; ANSI, New York City, 1977.

FUTURE DIRECTIONS IN GDMS DEVELOPMENT AND DATABASE CONVERSION

A. Shoshani
Lawrence Berkeley Laboratory
Berkeley, California

The development of Data Management techniques has progressed significantly over the last 10 - 15 years. The main reason to the advances in this area is the tremendous improvement in hardware cost and performance. It became possible to store large amounts of data on random access devices (such as discs), and to afford the overhead of using generalized techniques in order to save special purpose software development costs. In turn, the easier and more efficient was the process of accessing and maintaining data, the more data was collected and relied on for daily operations. The current stage of this process is that more and more users who are not computer specialists need to use computers for their data management needs, and the amounts of data needed to be managed is getting larger and larger. It is with this picture in mind that future development is presented here. We will discuss Hardware Development trends for making data management a more efficient process, Software Development trends for making GDMSs more useful and easy to use by different types of users, Distributed Databases to allow the distribution of data over a computer network, and Database Conversion to provide software tools for moving data from one computer environment to another.

1. HARDWARE DEVELOPMENT TRENDS

The most significant impact expected to take place is as a result of storage hardware development. In addition, improvements in cost effectiveness of hardware logic will influence the Data Management areas. We will concentrate here on specialized machines for data management, on hardware for large databases, and on "Back-End Machines" which will perform data management functions exclusively.

1.1 Specialized Machines for Data Management

Data Management can be thought of to a large degree as the process of associating data. For example, a search for all records that satisfy a certain criteria (e.g. SEX = MALE and SALARY < 20000) requires the association of the values in records with the values in the criteria. Using a general purpose serial machine is quite inefficient for this type of operation. Therefore indexing techniques, "hashing" techniques and the like have to be used. However, these techniques introduce more storage inefficiency and data management overhead.

It has been realized in the past that an "associative processor" would be more efficient for data management functions. In the example above the entire search could be done in one hardware (super) operation if we have all the data in an "associative memory." However, the cost of such an associative memory and processor are too high to be practical.

The introduction of conceptual models (discussed in more detail in Section 2 below), and especially the "relational" model served to emphasize the usefulness of a large scale

associative processor for data management. With the reduction in hardware costs there are several attempts of constructing such machines (e.g. [1,2,3]), by having specialized logic circuits associated directly with a large scale storage device (for example, adding this logic as part of read/write heads of disc units). One can expect this kind of work to become cost effective in the future, thus having essentially very efficient special purpose machines for accessing and manipulating databases.

1.2 Hardware for Large Databases

Large databases in the order of 10^9-10^{10} characters are now a reality. Data can be generated very fast, for example, in a scientific experiment or a large scale survey (e.g. census). The major problems are in storing them in a useful way. Devices such as tapes are not adequate for these situations, since one needs hundreds of tapes to hold this amount of data. In addition, the search of the data is very slow and expensive.

There are a few so-called "mass storage devices" in existence today that can store and access large amounts of data. However, they did not proliferate mainly because of high cost. These devices rely on some mechanical mechanism to get the data. Some examples are: the IBM 1360 photo-digital storage system, the IBM 3850 cartridge tape device or the CALCOMP automatic tape library. Some of the devices require long maintenance procedures and some are too slow for many applications. It is still the case that large databases are managed inefficiently using large discs or tapes. The management overhead is large, because the system needs to decide what parts of the database should be kept at what level of storage devices, and manage swapping large amounts of data between primary and secondary storage devices.

However, there is some hope that in the future reliable cost-effective mass storage devices would become a reality. Devices such as the video disk, bubble memories, and electron-beam access method (EBAM), show much promise of success. It is conceivable that dedicated mini-computers will be connected to these devices for the purpose of data management and data transfer.

1.3 Back-End Machines

Large scale scientific computers (e.g. CDC 7600) are not designed to perform efficiently data management functions (their capabilities are wasted while waiting for I/O operations. Therefore the idea of using a dedicated "back-end" machine to perform data management functions has been pursued, while complex analysis of the data (e.g. statistical analysis) after the data are retrieved is done at the main machine. In addition, as the number of data management applications grow, it seems more justified to have such a dedicated back-end machine to perform data management functions. With the current technology and the size of databases growing, it seems more and more attractive to have an I/O-oriented mini-computer configuration as the back-end machine. Of course, when Data Management machines (discussed in 1.1) become a reality they could serve as the back-end machine.

Back-end machines require the design of software for operating system, data management and communication functions that will operate efficiently in this specialized environment. Computer networking, process-to-process communication and data management techniques have advanced over the last several years to such a degree, that there is little doubt that efficient cost effective back-end machines can be designed.

2. SOFTWARE DEVELOPMENT TRENDS

Much of the work over the last few years in the data management field has concentrated on developing techniques that will make GDMSs more user oriented. The concept of "data independence" was introduced to emphasize that users do not need to be exposed to the details of physical organization of a database, but only to the logical relationships between data elements. Similarly, the ability to provide the user with a "non-procedural"

query language is considered advantageous to a user. Roughly speaking a non-procedural language gives the user the capabilities to describe what he wants to retrieve (search) or modify without the need to specify how the system is to get the data. Attention to proper user interfaces is coming about because more and more people who are not computer specialists need to use data, and they need a functional view of data management that is simple and easy to learn. This can be compared to driving an automatic car, where the burden of shifting gears is left to some automatic device that is part of the machine. The following discussion of data management software development is motivated by the need for user oriented software.

2.1 Conceptual Views and User Views

The conceptual view of a database is a model that describes the logical relationships of data elements in the database. A conceptual view usually starts with the entities which describe the data, then the attributes of these entities and then the relationships (associations) between these entities. Entities are characterized by an independent existence while attributes and relationships do not. Quite often, it is hard to draw the distinction between entities and attributes, because they reflect the way in which the database is intended to be used.

Different conceptual models that have been developed over the last few years (e.g. relational [4], entity-relationship [5], entity set [6]) emphasize different aspects of logical structures, but they all share the characteristic of being independent of physical database organization. Systems based on these models are still experimental, and commercial GDMSs (including COSASYL-based systems [7]) reflect in varying degrees a dependency of the logical model on the physical organization. Usually, this is done in order to provide more control to the user, so that he can write programs that process the data efficiently. At the same time it places the burden on him to write complex and detailed commands for effecting the retrieval or manipulation of the data.

The trend today is clearly oriented in the diretion of alleviating the user from knowing and controlling the details of "navigating" through the database. Instead, software development is progressing with the goal of performing optimal searches automatically, based on the analysis of the query and knowledge of the physical structures. To perform this process properly the system must also maintain statistical information about the data. All this adds to overhead costs that introduce some inefficiency. However, this cost may be worthwhile when considering the much simplified task of the user.

In addition to freeing the user from the need to know about details of the physical organization of the data, there is a possibility of providing him with tools to define his own view of the database. The user view may not coincide with the conceptual model, or include only a subset of it. Such techniques are only in their infancy, the difficulty being the need to map from user views to conceptual views dynamically. We can expect the ability to define user views to be a standard part of future GDMSs.

One of the data models that was most influential in the trend discussed above is the "relational" data model. This model introduced by Codd [4] (many additional references can be found in the March 1976 issue of Computing Surveys, Volume 8, Number 7), is based on representing data into "relations" which are essentially tables or matrices. The columns of the tables represent data elements of fields and rows represent specific instances for all fields. For example a relation called "employee" could have columns called "name", "salary", "age", and "department." A row representing a certain employee could contain for example "Jack Jones, 15000, 37, Research" corresponding to his name, salary, age and department. This example represents no more than a "flat file." However, in the relational model the user can use a language to relate any two relations for the purpose of query or modification of the data. For example if there was another relation on departments having columns containing "name" and "manager," then one would ask about all employees that earn more than 15000 and work for a certain manager, by writing an appropriate command linking the two relations. In this model, the user does not need to be aware of the physical structure implementing the relations and can access the data

involving any number of relations. This model is attractive because of its simplicity and the ability to perform complex associations using a powerful query and modification language. Other models are also based on the "independence" from physical implementation, but emphasize different aspects of the semantics (meaning) of the database.

2.2 Multiple User Interfaces

Experience with data models indicates that different users prefer to use different conceptual models for the same database depending on the application needs they have. Some databases can be modeled, for example, quite naturally as hierarchies while others require richer models, such as the relational model. However, trying to represent a hierarchical database as a set of relations can introduce redundancy of data. At the same time trying to represent a relational database as a hierarchy may prohibit some relationships and introduce other unnatural relationships.

To accomodate multiple models for users one needs an underlying conceptual model that is rich enough to accomodate the different models. This in itself is not a difficult task. The difficulty is in transforming dynamically queries based on one model structure to queries in the conceptual model without introducing inefficiencies and inconsistencies. Research is still in progress in this area, and we can expect to have an efficient solution in the future.

2.3 Interfaces to Other Software Packages

With current GDMSs, after a user retrieves the data, he often needs to go through extensive reformatting process in order to input it to another program, such as a graphics package or a statistical package. This process is often difficult and cumbersome.

We should expect that in the future GDMSs will be able to interface directly to other software packages, thus alleviating the user from the task of reformatting and initiating these packages. Some activities defining and adopting standard forms for data streams are already taking place (for example [8]).

2.4 Flexibility of Physical Organization

Another area that should improve with future systems, is the flexibility to have a large spectrum of physical organizations available in a GDMS. Furthermore, it should be possible to change the physical organization without affecting the application programs. In current systems the number of possible choices of physical structures is rather limited and non-flexible.

In future systems physical organization of data will clearly be affected by hardware advances, but no matter how much the cost-effectiveness improves, there will always be a need for organizing and managing database as they grow in size. The ultimate solution will be the development of techniques for reorganizing the physical structure of a database automatically as the use of the database changes in time.

3. DISTRIBUTED DATABASE SYSTEMS

Recent advancements in computer networking technology bring about the potential of using computer systems in new cooperative ways. One of the most exciting and promising areas is that of distributed data bases. There are many reasons for the need to have data distributed, but the most prevalent is the organization of data according to their functionality, thus allowing for local applications to be performed efficiently, while still permitting global operationg to take place. For example, consider several hos-

pitals being put on a computer network. Most of the processing needed will be done locally for every hospital, but some global operations, such as statistics, summaries, or search for an appropriate donor, could be performed over the network involving several or all of the hospitals. Similar situations and applications can be imagined in enterprises such as banking, inventory management, libraries, and research facilities. Another need for distributed data might arise when very large databases exist. In that case, data can be distributed over several facilities for the purpose of parallel access to the data.

Distributed databases impose several problems that are beyond the technology of computer network communication. In order to achieve distributed database systems, it is necessary to smooth out the differences between the (possibly) disparate data management systems (DMSs) that manipulate data on the network. It is also necessary to interface these systems in such a way that a user will be unaware of the fact that he is dealing with different systems across a network. Finally, when a user deals with data that is physically distributed over several systems, he should be able to think and refer to it as a single database.

Some of the properties that are desired in distributed database systems are discussed below.

a. Allow different DMSs to exist on the network. Different DMSs exist, because they offer different cost effective features suitable for certain applications. For example, a system designed for fast retrieval requires index mechanisms which tend to slow down updating.

b. Allow data existing on different systems to be shared. Mechanisms for correlating existing databases are necessary if databases need to be physically distributed.

c. Allow evolutionary integration of the DMSs. When a new DMS is added to the network or replaces an existing DMS, it should cause a minimum of disruption to the network.

d. Fail-soft properties. A distributed database system should allow for a degraded service in case of a local failure.

e. The additional cost for achieving distributed database systems in terms of response-time and implementation should be small relative to the cost of database management and insignificant relative to the benefits achieved.

There are several problems that need to be solved before distributed databases can be used effectively. The most important ones are concurrency, integrity and distributed control. Concurrency is the problem of permitting multiple users to access and modify a distributed database, without interfering with the consistency of each other's results. Integrity is the problem of maintaining a valid content of the distributed database in view of the non-synchronous nature of computer network communication. Distributed control is the problem of devising algorithms that do not require control and directories for the access of distributed databases to be located in one central location. Centralization of control is not desirable because of traffic jams and reliability in the case that the central node fails. Extensive research is already taking place in these areas, and we should expect solutions to these problems to prevail in the future.

4. DATABASE CONVERSION

Although it has been recognized that generalized database conversion tools are quite useful, the development and proliferation of such tools have not taken place at a large scale. There are only a handful of limited attempts in the industry, and a few more ambitious projects at universities and research institutions (for example [9,10]).

Conversion tools are necessary for any enterprise dealing with data. The ability to reorganize a database easily after its creation and to introduce dynamic improvements are as critical as the initial decisions in the database structure. It is unrealistic to expect in most cases, that the initial specification of a database would always be correct, or that the use of the database would not change in time. It is also naive to assume that once a data management system is selected (hardware and software) there will not be a need to move databases to other systems because of technological advances or new application needs. The lack of conversion tools is stifling to an enterprise because its databases stay fixed and stagnant. Distributed databases also require conversion tools. In order to facilitate distributed databases, it might be necessary to transfer databases from one application environment to another across computer network nodes. In addition, if a dynamic distributed data management is to exist, it is necessary that data organized by existing application systems can be converted and transferred into new more advanced systems. These are some of the reasons for the search of generalized tools for database conversion and transfer.

A database conversion process can be thought of as taking place in stages. First, the source data needs to be read from its physical environment into a data stream (called the "unload" process), then the data stream is reformatted into a standard form, then a restructuring process (where a logical reorganization of the database) takes place because of changes required in the logical view of the data or a change in conceptual models from the source to the target system. The result of the restructuring process is a standard form and then the process reverses itself. Reformatting takes place into a data stream acceptable by the target "load" process, followed by the load process itself to generate the target physical structure.

The strategy for future development should include a consideration for ease of use of conversion tools, and the modularity of these tools. The best hope in having usable conversion tools is by simplifying the conversion process to the point that a non-database expert can use it. For this reason there is a need for unload and load tools that eliminate the need to have knowledge of physical organization of databases. This can be achieved by designing future GDMSs to include them as standard facilities. Similarly, the reformatting process should be made transparent as a result of using a standard form. The entire conversion process should be stated only in logical data structure terms. The methodology for the restructurer should allow for logical database descriptions in dissimilar models for the source and the target. In this way each description will be as close as necessary to the DMS it is associated with, and efficient restructuring could be performed minimizing the overhead. Also, error detection facility and automatic conversion checks should be provided by the system.

Over the last several years, the state-of-the-art have advanced enough to give hope for generalized tools. Within the next five years we can expect more conversion systems to become operational, but they would not be completely generalized. We can expect to have a standard form developed and agreed upon. It will probably take longer before manufacturers will see the benefit of adopting a standard form and provide load and unload facilities using it. However, we can expect them to provide some conversion tools to convert databases from other systems to their own. It will probably take as much as ten years before a generalized converter would be available commercially, and manufactuers adhering to a standard form. Another area of concern is the application program conversion that is required as a result of database conversion. This is a difficult technical problem even within one data model, and still requires much research. It is hard to expect that a generalized solution for this problem will be achieved within the next five years.

REFERENCES

1. Ozkarahan, E. A., Schuster, S. A., and Smith, K. C., "RAP - An Associative Processor for Data Base Management," Proceedings of the National Computer Conference, 1975, pp. 370 - 387.

2. Su, Stanly Y. W. and Lipovski, G. J., "CASSM: A Cellular System for Very Large Data Bases," Proceedings of the International Conference on Very Large Data Bases, Farmingham, Massachusetts, September 1975, pp. 456 - 472.

3. Lin, C. S., Smith, D. P. C., and Smith, J. M., "The Design of Rotating Associative Memory for Relational Data Base Applications," ACM Transactions on Data Base Systems, Vol. 1, No. 1, November 1976, pp. 53 - 65.

4. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks, " Communications ACM 13, pp. 377 - 387, 1970.

5. Chen, P. P.-S., "The Entity-Relationship Model: Toward A Unified View of Data," ACM Transactions on Database Systems 1, pp. 9 - 36, 1976.

6. Senko, M. B., Altman, E. B., Astrahan, M. M. and Fehder, P. L., "Data Structures and Accessing in Data-Base Systems, " IBM Systems Journal 12, pp. 30 - 93, 1973.

7. CODASYL Data Base Task Group Report, April 1971, ACM, New York.

8. Brooks, A. A., "The Rationale of a Standard Interchange Format," in this report.

9. Birss, E. W., and Fry, J. P., "Generalized Software for Translating Data," Proceeding of the 1976 National Computer Conference, AFIPS press, Montvale, New Jersey, 1976, pp. 889 - 899.

10. Shu, N. C. et al, "Express: A Data Extraction, Processing and Restructuring System," ACM Transactions on Database Systems 2, 2, June 1977.

E P I L O G U E : THE COMPOSITION OF THE STUDY, AND ITS CONCLUSIONS

THE COMPOSITION OF THE STUDY

This study on the use of Generalized Data Management Systems has brought together thirty-seven participants from twenty-two organisations, in its two meetings held in Europe and in the United States. The organizers put their best efforts towards recruiting a well-mixed group of scientific information specialists, scientists constrained by their work to maintain large data collections, data base systems specialists, and managers in the field of scientific information.

The relatively high proportion of computer-oriented participants reflects the novelty of the data base approach in scientific information work and in science itself : very few of the GDMS applications discussed were more than two years old, and many of the case studies discussed in Section III of the report refer to pilot developments and feasibility studies. Most scientific applications which are up and running are still too modest to test the real capacity of the data management systems which carry them.

Within these limitations, contributions cover a very wide range of topics in the general field of scientific information handling : numerical data compilations (whether carried out by data centres or 'data-handling scientists'), bibliographic indices and library administration. The computer systems contributors are engaged both in GDMS systems development and as data base administrators. Two 'systems' papers cover a highly integrated approach to scientific calculations, so far limited to nuclear technology : the modular codes used for a very wide range of reactor calculations, each module accessing a common data store maintained by a purpose-built data base management system.

CONCLUSIONS

During the second meeting in Berkeley, the study group attempted to draw conclusions as to the value of the data management approach to scientific information. Generalized Data Management Systems were agreed to be useful for a wide range of scientific applications, and we could identify some criteria which would help users in deciding whether or not to take a data management approach to their own information handling problems.

Criteria for GDMS use

An integrated approach to data base construction, using GDMS software, is likely to be very worthwhile under one or more of these conditions :

- Where a data base needs to be shared between, and perhaps modified by, several users.
- Where reasonably complex logical interrelations between data exist and are to be made explicit, perhaps by linking several separate collections of data files.
- For data bases in the size range from a few million characters to a few hundred million.
- If user queries cannot be predicted when the data base is established, or may change with time.
- Where the organization does not have the manpower, time or expertise to develop a special-purpose system and to maintain it, or where a GDMS is already available within the organization.
- Where data users are not computer programmers and do not want to be drawn into programming.

The effect of data base size

For data bases somewhere below one million characters in size, and depending on the use made of the data, the demands made on a GDMS may be relatively trivial, so that home-made storage and retrieval programs could give good results more cheaply. Where a GDMS is already available, and users are familiar with it, some of them prefer to use it for even the simplest application, since in this way they can use the full range of GDMS facilities without the trouble and expense of writing the underlying data handling programs.

The advantages of current GDMS are most apparent for data bases between a few million characters and a few hundred million. In the 'grey area' beginning around 3×10^8 and extending up beyond 10^9 characters, storage costs on disc or other high-speed memory devices become appreciable, and the performance of systems in which external memory is reached through the operating system's file manager may no longer be adequate.

For very large data bases, a few billion characters or more, a dedicated computer and corresponding investment in special-purpose software is likely to be necessary, and justifiable by the cost of acquiring the data. A data base of this size probably contains 'raw' data from interrelated experimental measurements, and the flexible, high-level user view which GDMS are aiming for is surely essential if such masses of data are to be adequately assimilated. While very large semi-fast storage devices have been developed, correspondingly efficient access software is not yet available, and data management systems must be able to work efficiently under the constraint that most of the data will at any given time be 'backed off' from fast memory on to slower devices (currently tapes).

Choosing data management software

The meeting agreed that it was fruitless to propose any water-tight definition of a GDMS, although the management of logical structure within the data base, providing user access through a high-level access language, seemed promising as a test differentiating GDMS from file managers and information retrieval systems. Many potential users are not interested to know whether or not a given system is a GDMS : what they want is software which works for them.

However, in discussing systems requirements, the point was re-repeatedly made that the data base and its management software must be open-ended to permit developments not foreseen when the project was first planned. It is not certain that money will be saved by using a GDMS rather than writing special-purpose programs, but users will gain flexibility and so be able to use data in the long term better than they otherwise could. Project managers would be wise to choose data management software offering more flexibility than they think they are likely to need.

It is in this light that readers may find it useful to study the papers in Section II about scientific GDMS, and requirements for scientific data handling : there is a surprising degree of overlap between these independent analyses, and readers may discover that they want some of these features too. Chapter 2 gives a list of data handling software, which is necessarily incomplete, but covers a range of packages from full GDMS to proprietary file management systems, information retrieval systems designed primarily for text searching, and report generators. It offers a possible starting point in deciding which packages to review for a specific scientific data or information project.

Very large or very small data bases may be better served with special-purpose programs. For smaller data bases, Chapter 4 presents an approach using one particular very high level programming language, APL. The author makes the point that an apparently complex collection of data may often in practice be resolved into a number of smaller, simpler and nearly independent data bases which may be administered more cheaply using special-purpose programs. Well-written structured programs in a high-level language should then be easy to extend as the need arises. In his view, the need for data protection (security and integrity in shared data bases, for example) as provided by many GDMS is as important as logical complexity in deciding whether or not to use a GDMS package.

Limitations of currently available GDMS

Although a fair proportion of the requirements for scientific GDMS presented in Section II are not filled by the systems presently available to the public, it seems clear that for a given scientific data project one or more systems can be found which will do most of what its programmers want, and so help towards their goal of a satisfactory overall system. Three very prominent limitations are :

- It is not yet possible to offer a fully general representation of data structures without sacrificing performance. Most commercially available systems offer a more limited data model, which may force the user to adapt his applications to the constraints of the system.

- In particular, commercially available systems do not support some of the structures inherent to scientific data, such as vectors and arrays. Most such systems do not recognize

numerical data types such as floating point : these data are treated as characters and must be interpreted outside the GDMS. It would also seem natural to store data tables and text strings in variable length records at appropriate points in the logical structure of the data base. This approach would for scientific data give rise to extreme variations in record length, over one or two orders of magnitude, and cannot easily be implemented in current systems.

- Systems presently on the market do not provide full data restructuring facilities. Fundamental restructuring of a data base whose use has changed may be expensive both in programmer time and computer running time.

The cost of using GDMS

Chapter III develops a methodology for identifying the costs of mounting a GDMS application, and compares these with the cost of special-purpose software in one particular project. One can conclude in general that where GDMS installation is charged to the project, costs are brought forward in the project life-span as compared to special-purpose programs, but initial costs will be of the same order. However :

- The second and subsequent applications using the GDMS will be cheaper (the GDMS costs are already written off).

- Applications using GDMS will be more flexible, and so have a longer useful life.

- The continuing maintenance costs will be lower using GDMS.

Besides initial purchase of the GDMS package, allowance must be made for training, maintenance charges for the GDMS, the cost of likely increases in disc storage required and on-line use of the computer, and the computer time used. The main benefits of GDMS lie in the better use that can be made of the data base and of computer personnel rather than in their longer-term financial advantage. Further study of GDMS costs and benefits relative to straightforward programming is needed, but is limited precisely because these applications are rarely comparable : GDMS programmers will be tempted to use the system at full stretch, and do more with the data.

GDMS performance

This study turned up only very limited information about data base performance. Prospective users would surely like to compare running times of a particular application using GDMS or special-purpose programs, particularly for large data bases on small computers where poor performance could make GDMS impracticable. When users have eliminated data base design errors (later seen as 'obvious'), are there non-linear effects which will degrade performance as the data base is fully populated ? Most GDMS applications are new, with small data bases, rather than conversions of existing data files, so that comparisons are again difficult.

The NEA Neutron Data Compilation Centre (CCDN) is the GDMS user with the smallest computer in this group, an IBM 370/125 with 128 Kbytes main memory, later to be replaced by a DEC PDP11/70. Tests on reasonably large sections of their projected data base (using IDMS, a CODASYL GDMS) suggest that, for a well-tuned logical 'schema', retrieval performance will be comparable on the small IBM computer with that of current ISAM file-based programs. Data loading without the fast load utility may absorb up to 60 hours running time for the whole 160 Mbyte data base,

but this is considered acceptable for a one-off operation that can be interrupted at will. A limited benchmark comparison suggests that overall performance of IDMS functions, especially loading, will be faster on the PDP 11/70.

While these tentative conclusions can be extrapolated only with great caution to other data bases and other computers, CCDN experience suggests that the GDMS approach is viable even with very modest hardware.

Future development in GDMS

We hope that software producers will recognize the important place likely to be taken by GDMS in handling scientific information, and take into account in their systems design the requirements discussed in Section II. The spectrum of applications to be covered by the next generation of GDMS is so diverse that it seems unlikely that a single system can handle all of them well. A modular approach to GDMS may prove more satisfactory.

An important stumbling block in the use of GDMS by non-programmers is the failure of current systems to achieve full data independence. Future development should concentrate on providing users with a logical model (data structures and manipulation language) independent of the physical representation of the data base. The continuing fall in hardware component costs may make it possible to absorb the performance overhead necessary to provide full data independence. People time, rather than computer time, will become the major cost factor in data processing, and future GDMS design should continue striving to save it.

APPENDIX

A SELECTION OF REFERENCES TO GDMS LITERATURE

N. Tubbs, OECD Nuclear Energy Agency

This general reference list is intended to supplement the more specialised tests included with some of the papers in this report. Comprehensive bibliographies on Data Base Management are hard to find, perhaps because many of the introductory and GDMS overview publications come from small or little known publishers : the present list is deliberately short, and is limited to

- Review books and articles about GDMS, particularly introductions to the subject.
- Comparisons between different GDMS.
- A limited selection of 'technical overview' articles and documents, for only a few systems, and chosen on the basis of readability.

French and Germany references are included besides English, and there is even one in Japanese. We recognize that this list is still very incomplete. The notes about individual publications reflect only the author's personal opinion.

I. BOOKS and Monographs

1. The March 1976 issue of ACM Computing Surveys is devoted entirely to Data Base Management Systems, and gives a very good introductory treatment. See 10 below if you read French.

2a. Data Base Management Systems L.J. Cohen

b. Systems descriptions of :
IMS/VS TOTAL ADABAS SYSTEM 2000 IDMS

c. Data Base Systems : a practical reference I.R. Palmer

d. Monographs, No. 1 : "Access Mechanisms and Data Structure Support in DBMS" R.M. Curtice
gives a very clear presentation in sixty pages.

All published by QED Information Sciences
P.O. Box 181
141 Linden Street
Wellesley, Mass. 02181

3. Selection and Acquisition of Data Base Management Systems

A Report of the CODASYL Systems Committee, March 1976

(ACM order dept. P.O. Box 12105, Church St. Station, New York, NY 10249 # 12)

A 'black box' approach tells you all about what they do, and how to choose and buy them, but not really what they are.

4. An Introduction to Data Base Systems

C.J. Date, Addison-Wesley, 1975

ISBN 0-201-14452-2

Textbook, introduces relational data base ideas and gives an extensive treatment of IMS. The treatment of CODASYL systems seemed less good to me.

5. Computer Data Base Organization

J. Martin, Prentice-Hall, 1975

6. Organisation de données (Cours MIAG, première année)

C. Delobel, Grenoble 1975, in French

7. Infotech International State of the Art Reports

Infotech Internat. Ltd.
Nicholson House
MAIDENHEAD SL6 1LD
Berkshire (England)

- DATA BASE SYSTEMS (1975)
- ON-LINE DATA BASES (1977)

8. DATENBANKORGANISATION (in German)

H. Merten, Verlag Rudolf Mueller, Köln 1972
ISBN 3-481-35496-7

The presentation is highly systematic. As it is an enunciation of design principles rather than a treatment of specific software it may be useful in spite of its age.

9. Gesellschaft für Mathematik u. Datenverarbeitung (GMD)
BONN, Federal Republic of Germany

Reports in German of the Institut für Informationssystemen

- Kurzbeschreibung von Information Storage und Retrieval Systemen
- Datenbanksysteme - Erfahrungsberichte

10. Guide Pratique des Bases de Données

Tricot (ed), 1976, in French. ISBN 2-901001-02-5

Editions d'Informatique
82 rue Lauriston
75116 PARIS

Very complete and well written, probably one of the best books available on GDMS. A theoretical section covering logical structures and their physical implementation is followed by information and user comments (including performance figures) on most systems on the French market. Good bibliography.

11. Data Base Management Systems

Tsichritzis and Lochovsky, Academic Press 1977
ISBN 0-12-701740-2

Clear and pleasant presentation, leading in to GDMS from the general context of data handling and the need for multiple views of data, via general data models and particular implementations. Chapters on IMS, S2K, IDMS, TOTAL and ADABAS. Many worked examples and reader exercises.

II. Reports, articles, etc. (introductions to GDMS or systems comparisons)

1. Six Data Base Management Systems : Feature analysis and user experiences.
NBS Technical Note 887

This and many other U.S. Govt. publications can be bought from :

National Technical Information Service
U.S. Dept. of Commerce
5285 Port Royal Road
Springfield, VA 22161

2. Joho Shori (Information Processing Society of Japan)

Vol. 17, No. 10, October 1976 is a special issue on GDMS in Japanese

3. Data Base Management : What's it all about ?

A very good general introduction.
DEC Internal Report DEC-00-XDBMA-A-D

4. Datapro A buyer's guide to DBMS

and Datapro reports :

70E-010-61a	Software
70E-491-01	IMS DL/1 DOS/VS
70E-132-01	TOTAL
70E-757-01	ADABAS
70E-762-01	Sys 2K
70E-282-02	IDMS

Data Research Corp.
1805 Underwood Blvd.
Delran, NJ 08075

The feature comparison in these reports might be of little use to readers who do not yet understand DBMS.

5. User ratings of software packages Datamation

Articles published in December of 1975 and 1976 include ratings of GDMS.

6. EDP Analyzer February 1974, Vol. 12 No. 2

The Current Status of Data Management

7. Elements of Data Management Systems

G.G. Dodd, ACM Computing Surveys Vol. 1, No. 2, June 1969

This pre-GDMS article gives a very good exposition of the basic physical file access methods underlying GDMS. The high-level features of the various GDMS are sometimes more, but usually less, transparent implementations of these storage strategies.

8. Eigenschaften von Datenbanksystemen - ein Vergleich

M. Plesch, J. Griese : Angewandte Informatik 11, 489 (1972)

The characteristics of a number of GDMS, and the data structures which may be represented, are reviewed. The explanation of data structures is clear, but clearly the systems have evolved since the article was written. In German.

9. Performance Assessment of Data Management Systems

A SCICON (London, U.K.) report on a series of performance prediction program packages, currently available for IDMS (in IBM and ICL versions) and planned for TOTAL, ADABAS and DMS 1100.

10. Présentation et Analyse de SGBD commercialisés en France

Bazillon and Benci, 1975, in French

IRIA, Domaine de Voluceau
Rocquencourt
78150 Le Chesnay, France

Relatively superficial presentation of the characteristics of the main systems available in France. Most useful as a reference guide for readers already familiar with GDMS structure.

III. GDMS Technical Overview Documentation

Many such publications are glossy and uninformative, or less glossy and still hard to understand unless you already know the system. Fuller documentation is of course available from software vendors, and juxtaposed or comparative introductions to some better-known systems can be found in several of the publications listed in Sections I and II. The limited list below references some presentations which may help the reader form a clearer view in his own mind of how a given system works.

1. System 2000 — General Information Manual (MRI Systems Corp.) Concepts et Possibilités (CAP-SOGETI, in French).

2. TOTAL — Reference manuals are supplied by Cincom Systems for TOTAL as implemented on a number of different computers. The introductory section is very clearly written : like TOTAL itself, the manual is relatively compact. A brief systems summary "The TOTAL Data Base Management System" is available in French from their Paris office.

3. IDMS — Concepts and Facilities manual, available from Cullinane Corp., Wellesley, Mass. Clear and with abundant diagrams and worked examples on data base structure ; from a simple start it goes quite deeply into system working and data base design. Valuable to anyone interested in CODASYL systems. A short overview is available in French from SEMA Informatique.

4. IMS — See the books by Date (Part I, 4) or Tsichritzis (Part I, 11) or attend a one-day IBM course on IMS. See also IBM Systems Journal.

5. ADABAS — See the chapter on ADABAS in Tsichritzis (Part I,11) or contact Software AG., Darmstadt, Germany.

These five systems are also covered by QED Information Sciences systems descriptions (Part I, 2b and 2d).

Relational Systems

6. INGRES — A good overall description of this relational GDMS is given in "The Design and Implementation of INGRES" (Stonebraker et al). ACM Trans. on DB Systems Vol.1, p. 189 (1976). Also issued as U. California, Berkeley, ERL-M577.

7. System R — IBM's major relational DB project. See, for example, "System R : a Relational Approach to DB Management". (Astrakan et al, ACM Trans. on DB Systems, June 1976).

Scientific Data Management Systems

8. BASIS — Capabilities Description is available from Battelle Columbus Laboratories. Clearly written and comprehensive.

9. BDMS — Berkeley Data-Base Management System Users Manual (LBL-4683 and later versions). See the paper in this report.

10. ORCHIS — Oak Ridge Computerized Hierarchical Information System. See for example ORNL-4929 (July 1973).

11. Master Control — The introduction to the Users' Manual of this LLL system gives an overview.

12. OMNIDATA — A users' manual will be issued as NBS Handbook 123 by the Office of Standard Reference Data.

COSMOS (UKAEA), JOSHUA (Savannah River Laboratory) and RSYST (University of Stuttgart) are examples of modular systems for nuclear reactors calculations, with more or less generalized facilities for handling structured data incorporated as part of the system. See for example the report on JOSHUA (DP-1380, April 1975) and the papers on COSMOS and JOSHUA in this report.

PART V
FRENCH TRANSLATIONS

PARTIE V
TRADUCTIONS FRANCAISES

INTRODUCTION GENERALE

Ce rapport vise tant les scientifiques, tels physiciens, chimistes, biologistes, etc., que le personnel administratif. Il devrait stimuler une prise de conscience parmi les scientifiques de toutes disciplines, en ce qui concerne l'utilisation des Systèmes de Gestion de Bases de Données (SGBD) pour le stockage, la manipulation et les recherches de leurs données collectées qui doivent souvent être partagées. Le rapport devrait également intéresser les fonctionnaires et programmeurs qui seront appelés à prendre des décisions sur la gestion de données scientifiques (numériques ou non). De plus, le rapport a pour objet d'identifier les caractéristiques spécifiques que devrait posséder un SGBD pour supporter les données scientifiques, notamment la gamme de représentations et de fonctions spéciales pour manipulation de données.

Nous espérons que la présentation du rapport sera utile pour ceux qui n'ont pas de connaissance préalable sur les SGBD. Un traitement des conceptes de base et de la terminologie SGBD, est suivi par une discussion des besoins en systèmes SGBD pour les données scientifiques, illustrée par des études de cas. Le lecteur qui a déjà de l'expérience en SGBD pourra bénéficier des conseils sur les systèmes, ainsi que des fonctions SGBD utiles pour la manipulation des données scientifiques. En particulier, ce rapport s'efforce :

a. d'exposer clairement ce que sont les SGBD, dans quel cas ils peuvent être utiles et quel est le matériel informatique nécessaire ;

b. de présenter une liste des performances nécessaires des systèmes généralisés de gestion de bases de données devant servir à la manipulation des données scientifiques. Un tel inventaire, présente à un moment où l'on investit un effort considérable dans le perfectionnement du logiciel SGBD, pourrait influencer les spécifications de cette troisième génération de systèmes de gestion des données ;

c. de comparer les SGBD avec d'autres possibilités, telles que les logiciels élaborés sur place, l'APL et les systèmes de gestion de fichiers ;

d. de montrer par des études de cas d'une grande diversité d'applications actuelles ou possibles de SGBD à des données scientifiques (dans différents domaines, au caractère plus ou moins numérique) ce qui est impliqué lors de l'utilisation des SGBD et quels avantages peuvent en résulter ;

e. de suivre l'orientation des travaux de perfectionnement des SGBD : systèmes de base de données relationnels, traitement des données distribuées et conversion de bases de données.

En tant que système, un SGBD offre certains outils logiques d'application générale, permettant de structurer une base de données, de charger les données et de les modifier, et d'organiser la base de données de façon à favoriser des recherches efficaces, avec mise en page des données. Un Système de gestion de données dit "généralisé" comporte un langage orienté vers l'utilisateur pour la commande des différentes fonctions, permettant ainsi de définir une quelconque nouvelle base de données et sa structure interne, de retrouver et de modifier les données sans qu'il y ait besoin de développer un logiciel (programmes) propre à chaque nouvelle base de données. D'un récent examen des SGBD nous citons les fonctions principales de ces systèmes [17] :

- Faciliter l'accès par une communauté hétérogène d'utilisateurs à une collection intégrée de données ;
- Maintenir la qualité et l'intégrité des données ;
- Garantir la confidentialité par des mesures de sécurité prises à l'intérieur même du système
- Permettre un contrôle centralisé de la base, condition nécessaire d'une gestion efficaces des données.

Du point de vue de l'utilisateur, le SGBD devra offrir :

- Une indépendance réciproque données-logiciel (ainsi il ne sera pas nécessaire de modifier les programmes d'application lors d'un changement de données ou de leur structure)
- Les langages et autres programmes utilitaires nécessaires à l'exécution de toute la gamme des fonctions de gestion de données : définition des structures logiques, chargement et mise à jour des données, recherche de données d'après des critères définis, sortie des données avec mise en page. Suivant le mode d'opération et les besoins de l'application, ces fonctions s'exécuteront soit en ligne, soit en traitement par lots.
- Répresentation interne et accès explicits tant des données numériques que de celles en format "caractères".

Ces points seront discutés plus en détail dans la première partie du rapport.

L'utilisation d'un SGBD peut apporter de nombreux avantages, mais pour beaucoup de scientifiques et autres utilisateurs non spécialisés en informatique (et qui n'ont pas le temps de se recycler en programmation) l'avantage principal est sa disponibilité immédiate. Si leurs besoins en manipulation de données sont relativement simples, ceux-ci peuvent éventuellement être satisfaits sans programmation supplémentaire en exploitant les possibilités du langage d'interrogation/utilitaire d'édition (Report Writer) d'un SGBD bien choisi. L'utilisateur ayant des applications SGBD plus compliquées pourra écrire ses programmes d'applications en un "langage-hôte" à haut niveau (tel FORTRAN ou COBOL) pour les relier à la base de données par des commandes spéciales, employant le Langage de Manipulation de Données du SGBD et intégrées dans les programmes d'application. Ainsi l'effort de programmation disponible peut être dirigé vers la solution de ses problèmes propres.

Le logiciel SGBD permettant nécessairement une mise en oeuvre très générale, son utilisation dans une application donnée peut donner

des performances moindres, comparées à un programme écrit dans ce but précis. En général, cette perte de performances est largement compensée par des économies de temps et de coût dans le développement du logiciel, par la disponibilité des données pour de nombreuses applications, et par les protections intégrité/sécurité des données, ainsi que par la facilité de la mise à jour et de la manipulation des données. Comme l'accès en temps partagé aux ordinateurs à partir de différents terminaux, et comme l'utilisation de langages de programmation évolués à la place de l'assembleur, la gestion des bases de données est le dernier en date des compromis dans lesquels la commodité d'utilisation est augmentée, au prix d'une augmentation de la puissance du système requis. Les coûts de ce "sacrifice" sont initialement élevés, mais le matériel et le logiciel ont historiquement évolué de telle façon que le prix de cette commodité s'est réduit.

Il semble évident que les données scientifiques peuvent avoir des caractéristiques différentes des données "commerciales", et le but de cette étude est d'explorer ces différences, et d'identifier les systèmes ayant des caractéristiques bien adaptées à la manipulation des données scientifiques. Ainsi, les participants à l'étude ont été choisis tant dans la communauté scientifique que dans le domaine de l'informatique. Le rapport comprend en particulier des études de cas d'applications des SGBD aux données scientifiques.

Le rapport présente d'abord la gamme des solutions et techniques SGBD, et leur terminologie. Ce chapitre examine les SGBD au niveau fonctionnel, sans entrer dans le détail de leur fonctionnement interne. Le chapitre suivant passe en revue certaines caractéristiques des systèmes disponibles (pour la plupart commerciaux) et offre des bases de comparaison. Dans la partie suivante sont discutés les besoins en capacités spéciales pour la manipulation des données scientifiques; plusieurs SGBD conçus spéciquement pour la manipulation des données scientifiques sont présentés. Les sections suivantes présentent un certain nombre d'applications réussies des SGBD dans le domaine scientifique, ainsi que diverses applications potentielles actuellement en considération. La dernière partie comporte une discussion des directions d'avenir dans le développement des SGBD et autres logiciels accessoires, pour que le lecteur puisse prendre en considération leur conséquences possibles pour son environnement informatique. En conclusion (dans l'"Epilogue") nous résumons les points de vue des participants sur les deux grandes questions de l'étude : quand utiliser un SGBD, et et quel résultat attendre de son utilisation.

REFERENCE

/1/ James P. Fry and Edgar H. Sibley, "Evolution of Data-Base Management Systems," Computing Surveys, Vol. 8, No. 1, March 1976, (le numéro entier est consacré à la gestion des Bases de Données)

INTRODUCTION AUX SYSTEMES DE BASES DE DONNEES GENERALISES

G. Moorhead, CERN, Genève
N. Tubbs, OCDE/AEN, Paris

I. QU'EST-CE QU'UN SYSTEME DE BASES DE DONNEES ?

1. Traitement de données scientifiques par ordinateur

Les programmes scientifiques et d'ingénierie traitent des données provenant de diverses sources. Le processus dans lequel des données brutes produites dans des expériences sont améliorées en vue de leur utilisation finale dans le domaine de la technologie, comprend typiquement trois phases. En premier lieu les données brutes sont saisies sur un équipement expérimental ; elles sont ensuite analysées par les expérimentateurs en vue de leur publication. Au cours de la phase finale, dite d'"évaluation", les résultats expérimentaux sont examinés, comparés et regroupés en un ensemble de données final, recommandé, qui peut être utilisé en tant que données pour des calculs relatifs à de nombreuses applications technologiques différentes ; on parle alors de données "évaluées". Une application scientifique peut demander des données initiales dans l'une quelconque de ces phases ; elles seront introduites à l'aide d'un ou plusieurs équipements de types variés. Ces données ont pu être initialement perforées sur des cartes ou sur un ruban perforé, frappées au clavier d'un terminal conversationnel ou saisies directement sur une expérience et enregistrées sur bande magnétique. A une certaine étape, les données aboutissent dans les fichiers d'un ordinateur et un programme d'analyse lira ces fichiers pour fournir un ensemble de résultats présentés sous la forme d'histogrammes, de graphes, de valeurs calculées ou de simples tableaux imprimés.

Tous les grands ordinateurs disposent d'un programme de gestion de fichiers qui fait partie de leur système d'exploitation et qui permet au programmeur de déclarer des fichiers logiques identifiés par un nom, et de spécifier sur quels dispositifs physiques de mémoire ils doivent résider. Le système organisera l'espace physique, y écrira des données ou les y lira lorsqu'un programme d'utilisateur le lui demandera. Un programme d'analyse typique sera rédigé en FORTRAN, et la disposition des données dans les fichiers sera précisée par les instructions FORMAT qui y figurent. Cette technique ne présente aucune difficulté lorsque les fichiers ne sont créés qu'une seule fois puis utilisés dans un seul programme, avec éventuellement de légères variantes.

Toutefois il se rencontre de nombreuses situations dans lesquelles plusieurs programmes différents doivent pouvoir accéder à des fichiers logiquement interreliés. La complexité sera plus grande encore si ces

fichiers sont fréquemment mis à jour ; ils peuvent également faire l'objet de modifications indépendantes par plusieurs utilisateurs autorisés.

C'est dans de tels cas que la solution des systèmes de bases de données généralisés (SBDG) peut être d'une grande utilité, et il existe actuellement de nombreux exemples de son emploi fructueux dans des applications scientifiques ou d'ingénierie. Pour l'essentiel, un SBDG offre un langage évolué pour la description et la manipulation de données, qui constitue ainsi un complément par rapport aux langages de programmation tels que le FORTRAN (surtout conçu pour la réalisation des calculs), le COBOL ou le PL/1, et qui permet de maintenir une base de données intégrée et cohérente, en vue de son emploi dans de nombreuses applications différentes.

2. Le rôle d'un SBDG

Un système de base de données généralisé se présente à l'utilisateur sous la forme d'un interface logiciel entre les programmes d'une part, et le système d'exploitation et l'équipement de mémoire externe d'autre part, lors de tout accès à un ensemble de données intégré, soumis à un contrôle central et partagé par un certain nombre d'utilisateurs. Cet ensemble est appelé une base de données. Le système fournit des ressources pour la définition de la structure physique de la base de données et des relations logiques internes, pour le chargement et la modification des données, pour la protection de la base de données contre une détérioration accidentelle ou des accès non autorisés, et pour une recherche de données efficace. De bons systèmes à usage spécialisé peuvent offrir un grand nombre de ces ressources. Un système de base de données est "généralisé" lorsqu'il fournit un langage de commande orienté utilisateur pour toutes ces différentes fonctions, qui est applicable à toute nouvelle base de données, indépendamment de son organisation interne ; il supprime donc la nécessité de rédiger de nouveaux programmes de manipulation de données pour chaque nouvelle base de données.

Lors de la lecture d'un fichier par un programme d'analyse classique, on lit habituellement les divers "enregistrements" un à un ; un enregistrement est un ensemble d'"articles" contenant des informations logiquement associées telles que les mesures effectuées en un certain endroit et/ou à un certain instant, au cours d'une recherche scientifique. En général, un fichier ne contient que des enregistrements d'une même forme, qui peuvent, par exemple, être lus à l'aide d'instructions FORTRAN d'un même type. Les fichiers qui seront intégrés dans une base de données peuvent présenter un nombre égal de différents types d'enregistrements ; chacun d'eux peut contenir à son tour des éléments de données ayant des caractéristiques formelles différentes (nombre entier, nombre en virgule flottante, données alphanumériques ...). Un SBDG offre des ressources pour manipuler ces enregistrements et ces données soit individuellement, soit par groupes, en utilisant des ordres de manipulation de données ne dépendant que de la structure logique de la base de données. La répartition des enregistrements entre divers fichiers et la disposition réelle des enregistrements et des fichiers sur les dispositifs physiques de stockage sont pratiquement invisibles pour l'utilisateur.

Il faut remarquer qu'un programme-produit fournissant la majeure partie des possibilités fondamentales d'un SBDG qui sont décrites ci-dessous peut être lui-même rédigé dans un langage plus évolué tel que le FORTRAN, en utilisant l'interface standard pour la manipulation des

fichiers qui est contenu dans ce langage. En règle générale, le développement d'un tel programme-produit n'exige que quelques années-homme, mais ce programme peut comporter des insuffisances en matière de généralité, fiabilité et efficacité.

II. LES POSSIBILITES OFFERTES PAR UN SBDG

3. Indépendance des programmes des utilisateurs par rapport aux données

Un SBDG permet à un utilisateur de faire référence et d'extraire directement des articles isolés présents dans un enregistrement, en les désignant par leur nom, sans qu'il soit nécessaire de déclarer la structure de l'enregistrement dans le programme de l'utilisateur. La structure de la base de données (structure des enregistrements, noms des articles contenant les données et relations liant les différents types d'enregistrements) est déclarée indépendamment des programmes individuels, pour toutes les applications. Cette déclaration intervient dans une phase initiale appelée définition des données.

Dès qu'une base de données existe et qu'elle contient des données, un utilisateur particulier n'a qu'à se préoccuper des noms des articles qui l'intéressent, et un programme d'application référence seulement les articles dont il a besoin. Cette possibilité est loin d'être négligeable, comme on peut le constater dans une application réelle d'un SBDG à la recherche océanographique, dans laquelle chaque enregistrement ne contient pas moins de 73 articles allant de la latitude et la longitude aux pourcentages des différents minéraux dans les échantillons qui ont été étudiés (x). Naturellement, la même information aurait pu être divisée entre plusieurs fichiers qui se recouvrent et qui comprendraient des enregistrements plus courts, mais une telle division impliquerait la nécessité d'une gestion des relations entre les enregistrements correspondants rencontrés dans les différents fichiers, gestion qui devrait être réalisée par programme et par inclusion d'informations de chaînage. Cela constitue en soi une fonction très importante d'un SBDG.

Le fait que toute information explicite sur la structure physique et logique des données se trouve extraite des programmes des utilisateurs pour résider dans un "schéma" central, auquel on a accès par l'intermédiaire du SBDG, confère à ces mêmes programmes un certain degré d'"indépendance par rapport aux données". Dans ce rapport d'étude, d'autres articles montreront que l'indépendance par rapport aux données est loin d'être complète dans la plupart des systèmes actuellement disponibles.

4. Possibilités de mise à jour

Le système de base de données offre des possibilités de mise à jour pour le stockage, la modification ou la suppression de données dans la base de données. Lorsque cette base est mise à jour, soit en y insérant des enregistrements totalement nouveaux soit en modifiant les valeurs des articles contenus dans les enregistrements, le système vérifie et convertit automatiquement les données conformément aux caractéristiques formelles de l'article, qui sont habituellement indiquées par l'utilisateur lorsque cet article a été défini en vue de son inclusion dans le schéma de la base de données. Un article peut être du type nombre entier, donnée alphanumérique (par exemple un texte alphanumérique), date, nombre réel,

---

Note (x) Cette base de données est utilisée par le Centre National pour l'Exploitation des Océans (CNEXO), au Centre Océanographique de Bretagne, à Brest (France)

etc. A titre d'exemple de validation notons que "1A34" ne serait pas accepté à la place de 1234 comme valeur d'un article appelé LENGTH (longueur) et décrit comme nombre entier. Certains systèmes permettent également à l'utilisateur de spécifier une gamme ou un ensemble de valeurs acceptables pour un certain article, ou même des conditions de validation encore plus générales.

La recherche rapide d'articles contenant des données, en opérant soit à partir d'un fichier classique, soit à partir d'une base de données, exige habituellement que ces données soient stockées physiquement sur un dispositif de mémoire à accès aléatoire (actuellement des disques). Dans la programmation des opérations de mise à jour des données pour des fichiers à accès aléatoire, l'un des problèmes les plus difficiles consiste à garantir la récupération d'un fichier au cas où il se trouve détérioré au cours de la mise à jour, à la suite d'une erreur de programme ou d'une panne de l'ordinateur. Dans un SBDG permettant à plusieurs utilisateurs de travailler en parallèle lors de la mise à jour de la base de données, il est important que non seulement les erreurs dues à l'un d'eux puissent être corrigées, mais également que cette opération se réalise sans effacer pour autant le travail qui a été effectué entre temps par les autres utilisateurs. De nombreux SBDG offrent des possibilités de récupération de la base de données à la suite d'un incident affectant l'un des fichiers qui la constituent ou de l'inclusion de données non valides. Pour ce faire, on peut réaliser périodiquement des copies de la totalité de la base, complétées par l'enregistrement de toutes les transactions de mise à jour.

5. Recherche de données

La fonction la plus importante d'un SBDG est de permettre la recherche de données selon certains critères. Sous sa forme la plus simple, cette recherche peut consister en l'extraction, dans un programme d'application, d'un enregistrement isolé contenant des valeurs spécifiées dans des articles déterminés qui ont été déclarés comme articles-clé, par exemple l'article PARTICLE=PROTON et l'article PLAB=1260. Une variante de cette opération, qui reste facile à mettre en oeuvre, consiste à rechercher tous les enregistrements correspondant à une gamme de valeurs des articles-clé, par exemple PARTICLE=PROTON et PLAB compris entre 12000 et 13000 ; les divers enregistrements sont alors fournis au programme d'application sur sa demande. Toutefois certains systèmes limitent à un le nombre des articles-clé possibles; il peut s'agir alors simplement d'un numéro d'identification pour l'enregistrement.

Il est facile de comprendre qu'un fichier puisse être organisé de telle sorte que la recherche des enregistrements, en utilisant des clés déclarées, soit efficace. En fait, des organisations de fichiers des types "séquentiel indexé" ou "accès aléatoire" sont prévues à cet effet dans la plupart des systèmes d'exploitation. Toutefois l'avantage d'un SBDG tient au fait que l'utilisateur a simplement à indiquer quels sont ceux de ses articles identifiés par un nom, qui doivent être utilisés comme clés ; le SBDG prend alors en charge un important volume d'opérations de gestion afin de créer les fichiers et d'organiser la mise à jour et la recherche.

De plus, un SBDG offre la possibilité de poser facilement des questions d'une nature qui a pu être envisagée ou non lors de la création de la base de données. Pour cela, l'une des solutions évidentes, qui n'est cependant pas la plus facile, consiste à passer en revue tous les enregistrements d'un type donné, en examinant les valeurs de certains articles. Les articles demandés seraient obtenus en utilisant des ordres

CALL transmis au SBDG, et les tests seraient effectués en employant les instructions IF du "langage hôte", tel que le FORTRAN, à partir duquel sont lancés les ordres CALL. La plupart des SBDG permettent de créer des enregistrements de différents types pouvant être scrutés simultanément, ce qui permet un travail équivalent à une recherche sur plusieurs fichiers. Certains systèmes présentent une autre caractéristique appelée recherche en format libre, consistant en un examen d'un texte alphanumérique caractère par caractère, en vue de repérer une sous-chaîne particulière qu'il contient.

Les SBDG les plus perfectionnés fournissent en plus un langage d'interrogation évolué permettant d'exprimer des critères de recherche de manière naturelle. Ce même langage peut être utilisé pour les mises à jour. Les conditions de recherche consistent habituellement en de simples prédicats dans lesquels des articles sont comparés à une constante (par exemple : PARTICLE=PROTON) ou liés ensemble par des opérateurs logiques comme dans l'expression ci-après : (PARTICLE=PROTON OR PARTICLE= ANTIPROTON) AND PLAB GE 12000 AND PLAB LE 13000. En plus de la recherche d'enregistrements isolés, le SBDG offre souvent des fonctions statistiques fondamentales telles que les moyennes, variances et analyses de régression ; il peut même présenter les résultats d'une recherche sous la forme d'histogrammes.

La confidentialité des données peut être assurée en limitant l'accès des utilisateurs à certains fichiers (ou types d'enregistrements) ou à un sous-ensemble logique de la base de données qui est appelé "subschema" dans certains SBDG. Le degré de discrimination entre les utilisateurs qui peut être imposé par ces "verrous de confidentialité" dépend du système ; plusieurs systèmes peuvent permettre à l'utilisateur d'accéder à des articles de données spécifiés, mais seulement dans un type donné d'enregistrement.

Quelquefois il est permis essentiellement à l'utilisateur de déclarer en premier lieu le type de questions qu'il pense poser fréquemment ; le système crée alors des "voies d'accès" en vue d'une recherche plus rapide, au détriment du temps de mise à jour et de l'espace nécessaire pour le stockage. Toutefois, la possibilité de poser des questions imprévues, sans avoir à rédiger et à tester un programme spécial, reste une des caractéristiques les plus agréables d'un SBDG.

Pour des applications commerciales il est très important de pouvoir présenter les données fournies par la recherche sous la forme de rapports mis en page et triés, avec titres et sous-titres, notes de bas de page, résumés, totaux partiels, etc. Le fait que dans un SBDG standard on puisse disposer de "report generators" (éditeurs) très élaborés, de tel ou tel type, peut être de peu d'intérêt dans la majorité des applications scientifiques, mais ces programmes peuvent s'avérer extrêmement utiles lorsque des listes imprimées sont exigées. Malheureusement une sortie de résultats sous forme de graphiques qui est évidemment souhaitable pour des applications scientifiques n'est habituellement pas disponible avec les éditeurs, mais il est clair qu'un langage hôte interfacé à la fois avec un SBDG et un programme-produit infographique offre cette possibilité au prix d'un certain codage supplémentaire.

6. Contrôle de la redondance entre les données

Une base de données regroupe sous forme plus ou moins intégrée les données qui autrement seraient dispersées sur un certain nombre de fichiers se recouvrant partiellement. Dans le cas des fichiers, leur

recouvrement logique se réalise par enregistrement des mêmes valeurs pour plusieurs articles de données dans deux ou plusieurs fichiers. A l'intérieur de l'ensemble des données considéré comme une entité définie, et stocké dans une base de données intégrée, on peut estimer que ces données répétées sont redondantes. L'information structurelle exprimée dans les fichiers qui se recouvrent par la répétition des valeurs de certaines données peut maintenant être gérée par le SBDG et apparaître sous la forme de pointeurs d'adresse ou d'indices croisés, transparents pour l'utilisateur.

L'une des raisons d'éviter l'inclusion d'une information redondante dans la base de données tient au gaspillage d'espace de stockage qui en résulte. Une autre raison encore plus importante (du fait que de nombreuses bases de données occupent davantage d'espace que les fichiers qu'elles remplacent) est que la redondance qui n'est pas contrôlée par le SBDG lui-même peut entraîner des erreurs qui détériorent gravement la base de données. Au niveau des articles de données, la redondance peut être réduite en limitant le nombre d'occurrences des enregistrements d'un article donné dans la base de données ; l'information structurelle contenue dans le système qui lie les différents types d'enregistrements se référant à l'article en question remplacera la répétition de ce même article. Au niveau des articles, une autre méthode consiste à stocker seulement une fois tout long texte qui apparaît de nombreuses fois en tant que valeur d'un article alphanumérique, ou tout ensemble de valeurs de différents articles qui sont toujours associées à une valeur particulière d'un autre article, par exemple les propriétés d'un composé chimique. Pour l'utilisateur, la caractéristique importante tient au fait qu'il n'a besoin de conserver qu'en une seule place les données décrivant complètement par exemple une particule ou un composé chimique qui est identifié ailleurs dans la base de données à l'aide d'un nom ou d'un code court. Une réduction de la quantité des données stockées peut entraîner une amélioration de la qualité.

III. STRUCTURES DES DONNEES

Les possibilités fondamentales offertes par la majorité des SBDG ont été maintenant passées en revue. Dans les descriptions des actuels SBDG on insiste habituellement beaucoup sur le type de structure logique que l'utilisateur est autorisé à employer pour stocker ses données. Dans la pratique courante cette structure logique est inextricablement liée à la structure physique ou aux méthodes d'accès utilisées. Les structures logiques offertes par divers SBDG peuvent être suffisamment différentes pour que leurs caractéristiques influencent le choix en faveur de tel système plutôt que de tel autre, pour une classe particulière d'applications ; toutefois ces différences ne sont pas habituellement assez importantes pour qu'elles entraînent une invalidation générale de l'emploi d'un SBDG donné.

7. Structure hiérarchique à l'intérieur d'un enregistrement individuel

La hiérarchie intra-enregistrement est un type courant de structure logique dans laquelle les articles composant un enregistrement sont organisés suivant une structure hiérarchique ou arborescente. Elle peut se limiter à une arborescence à deux niveaux dans laquelle un article du premier niveau peut comprendre un nombre indéfini de sous-articles du second niveau, en étant en quelque sorte analogue à un vecteur. Plus généralement, certains systèmes prévoient une hiérarchie à plusieurs niveaux dans laquelle des sous-articles eux-mêmes peuvent être formés de plusieurs sous-articles, et ainsi de suite. Par exemple, une particule isolée peut se désintégrer en plusieurs particules, et l'une

quelconque d'entre elles peut à son tour se désintégrer en d'autres particules, etc. Naturellement le système gère tous les pointeurs internes qui sont nécessaires pour mettre en oeuvre une telle hiérarchie et la recherche de l'utilisateur à travers l'arborescence s'effectue selon une procédure adaptée à l'application. Historiquement, des structures intra-enregistrement de ce type formaient la base des premiers SBDG car les enregistrements devaient être stockés dans un fichier séquentiel sur bande magnétique. La principale difficulté tenait au fait que le fichier devait être copié en totalité lorsque l'on procédait à une mise à jour (ce qui était nécessaire de toute manière pour des fichiers sur bande magnétique), alors que la recherche elle-même pouvait également exiger la lecture de l'ensemble du fichier.

Comme exemples de systèmes utilisant des hiérarchies intra-enregistrement on peut citer le système INFOL, déjà ancien ; mais qui reste en usage pour des applications scientifiques au CERN, et qui a été initialement rédigé par T.W. Olle vers 1965, ainsi que le système ORCHIS d'Oak Ridge. La figure 1 présente une hiérarchie intra-enregistrement. Dans la liste figurant à la fin de ce rapport on trouvera des références aux divers systèmes discutés.

8. Hiérarchie des enregistrements

Du fait que la diminution des coûts permettait de stocker de grandes bases de données sur des mémoires de masse, tels que les disques ou les tambours, présentant des caractéristiques d'accès semi-aléatoire, la recherche à partir d'une structure hiérarchique intra-enregistrement devint plus efficace car, à condition de savoir où regarder, la découverte d'un enregistrement isolé n'impliquait plus une recherche à travers la totalité du fichier. Il devint possible de permettre au SBDG de manipuler plusieurs fichiers constituant alors une base de données unique, et pour l'utilisateur, le temps d'accès à un article quelconque présent dans l'un quelconque de ces fichiers s'exprima alors en millisecondes. L'utilisation du stockage sur disque entraîna le développement de structures plus puissantes tenant compte de plus en plus des exigences des utilisateurs plutôt que des contraintes de la machine.

Une structure hiérarchique étendue, telle que celle qui est mise en oeuvre dans le SYSTEM 2000 et qui est présentée sur la figure 2, peut être complétée par des fichiers inversés permettant un accès direct à des articles de données spécifiés, par l'intermédiaire de l'adresse sur le disque qui se trouve dans ces fichiers, au lieu d'avoir à suivre l'arborescence depuis le bas jusqu'au sommet. Dans la hiérarchie logique, les différents niveaux sont représentés par des ensembles d'enregistrements distincts (appelés "repeating groups") au lieu de correspondre à des articles dans un seul enregistrement. Une telle structure hiérarchique ne permet pas de chaînages directs entre des articles ou des enregistrements appartenant à des hiérarchies différentes.

9. Structures en réseau

Des structures logiques en réseau se basent sur la notion d'"ensembles" interconnectés, comprenant chacun un enregistrement principal ("owner record") et un ou plusieurs enregistrements secondaires ("member records"). Un ensemble peut se présenter sous la forme d'une liste circulaire chaînée, et la recherche peut s'effectuer suivant une séquence logique en entrant au niveau de l'enregistrement principal et en scrutant successivement les enregistrements secondaires jusqu'à ce que l'enregistrement recherché soit identifié.

Un enregistrement donné peut être à la fois principal dans un ensemble et secondaire dans un autre, et dans ce cas il est possible de représenter directement une hiérarchie en utilisant seulement des ensembles, mais ils ne peuvent être à la fois principal et secondaire. Bien qu'un plus grand nombre de niveaux puissent être représentés indirectement, la représentation directe se trouve ainsi limitée à une hiérarchie à deux niveaux. Des systèmes conformes aux spécifications du CODASYL (voir paragraphe 13, ci-après) ne comportent pas cette restriction et ils peuvent représenter des hiérarchies complètes.

La plus grande puissance des réseaux par rapport aux structures hiérarchiques tient à la possibilité d'associer un type d'enregistrement à presque n'importe quel autre. Au lieu d'adapter la structure de la base de données à l'ensemble limité des associations permises par une structure hiérarchique, l'utilisateur peut commencer par définir les types d'enregistrements intéressants pour son application puis exprimer dans le schéma de la base de données toutes les associations qui existent entre ces mêmes enregistrements. Une arborescence pure peut être considérée comme le cas le plus simple d'un réseau, dans lequel un enregistrement secondaire ne peut appartenir qu'à un seul ensemble.

10. Bases de données relationnelles

Le modèle relationnel développé ultérieurement et mis en oeuvre jusqu'à maintenant dans un certain nombre de systèmes expérimentaux représente une approche davantage orientée utilisateur, pour les structures de bases de données. Dans ce modèle, une base de données est considérée comme un ensemble de relations n-aires de tables homogènes dont chaque ligne est analogue à un enregistrement contenant n articles, mais aucun d'eux ne peut avoir des occurrences multiples. Lors de la définition des relations, la cohérence et la non-redondance peuvent être garanties à condition de suivre un ensemble de règles formelles assurant que toutes les relations soient en "troisième forme normale".

Il existe une algèbre fermée des opérations appelées union, projection, etc., qui peuvent être effectuées sur des relations, et l'interrogation d'une base de données relationnelle consiste à appliquer cette algèbre. Lors de la définition des relations, l'utilisateur n'a pas à spécifier des voies d'accès, bien que de telles voies soient en fait utilisées lors de la réalisation d'unions et qu'elles ne puissent être totalement ignorées. Aucun SBDG relationnel n'est encore disponible sur le marché. La figure 4 montre schématiquement comment une hiérarchie peut être représentée à l'aide d'un ensemble de tables de relations dans lesquelles sont stockées les données.

11. Stockage physique des données

Il convient de préciser que les structures logiques qui peuvent être exprimées par un SBDG n'ont pas nécessairement une incidence directe sur le mode de stockage sur disque des enregistrements de données. Par exemple, dans des systèmes basés sur des structures en réseau, les enregistrements auxquels on accède par l'intermédiaire de leurs clés logiques sont souvent répartis sur le disque avec des adresses apparemment aléatoires, dans le cadre d'un fichier défini par le système. Ces adresses sur le disque sont souvent obtenues à partir des clés logiques par un algorithme d'adressage dispersé ("hashing"), et le but de cette procédure est d'assurer une distribution uniforme des données dans l'espace disponible sur le disque. Dans le SYSTEM 2000, l'information structurelle qui constitue la hiérarchie est stockée dans des "tables d'adresses hiérarchiques" (index) qui sont séparées des enregistrements de données

eux-mêmes ; ces derniers sont stockés sur le disque dans un ordre proche de celui suivant lequel ils ont été chargés. Dans des systèmes en réseau les ensembles et la structure hiérarchique sont habituellement exprimés à l'aide de pointeurs d'adresses ; ces ensembles ou ces hiérarchies peuvent très bien être physiquement mélangés et se présenter sous forme désordonnée, par rapport à l'ordre logique qui est exprimé par les pointeurs créés par le système.

Bien que l'implantation physique d'une base de données puisse ne pas être très analogue aux structures logiques qu'elle représente, les caractéristiques du système peuvent cependant être considérablement affectées par le mode d'"adaptation" du stockage des données afin de correspondre à la structure logique et aux voies d'accès les plus courantes des programmes des utilisateurs à travers la base de données. Inversement, ces mêmes caractéristiques peuvent être détériorées si la base de données est représentée directement sous la forme de l'aspect logique propre à l'utilisateur ; par ailleurs, les caractéristiques d'un programme particulier seront en tout cas détériorées lorsque la structure de la base de données aura été conçue en vue de satisfaire à des exigences élevées pour des applications multiples.

IV. NOMENCLATURE

Nous présenterons maintenant quelques autres caractéristiques des SBDG actuels afin d'introduire certains termes qui seront utilisés ultérieurement dans ce rapport.

Les innovations concernant les structures de base de données qui sont devenues possibles à la suite de l'emploi du stockage sur disque ont été discutées aux paragraphes 8 à 11, ci-dessus. On peut atteindre directement les données stockées dans des fichiers à accès aléatoire sur disque en obtenant l'emplacement sur le disque (c'est-à-dire l'"adresse") par l'intermédiaire d'un algorithme d'"adressage dispersé" ou à partir d'un index (un index est un répertoire dans lequel une valeur-clé logique peut être examinée pour découvrir les emplacements physiques des enregistrements dans lesquels cette valeur-clé est présente et qui peut être mis en oeuvre à l'intérieur du système selon diverses modalités, y compris l'adressage dispersé). Dans un contexte de base de données, des pointeurs se réfèrent à des adresses créées par le SBDG et incluses dans les enregistrements de la base de données afin de fournir directement l'emplacement physique des enregistrements adjacents dans la structure logique. Des pointeurs étaient déjà utilisés dans les hiérarchies intra-enregistrements des premiers SBDG, avec stockage sur bande magnétique ; ils constituent le support principal pour la mise en oeuvre des réseaux du CODASYL.

12. Un exemple de l'un des premiers SBDG : le système INFOL

La plupart des caractéristiques importantes des actuels SBDG se retrouvent dans les versions les plus récentes du système simple INFOL qui a été mentionné au paragraphe 7. INFOL est un système entièrement non procédural possédant son propre langage d'interrogation et de mise à jour. Il se différencie de la plupart des systèmes ultérieurs par le fait que l'accès ne peut se réaliser à l'aide d'ordres CALL figurant dans un langage hôte tel que le COBOL ou le FORTRAN. Ces langages des utilisateurs permettent de formuler des ordres à un "niveau évolué" d'abstraction logique ; ils se distinguent ainsi des langages "peu évolués" (tel que le langage assembleur) qui sont beaucoup plus proches du langage machine de l'ordinateur utilisé. Bien que la version originale d'INFOL ait été rédigée en code d'assemblage pour des ordinateurs CDC 3600/3800, ce

système a acquis une importante portabilité après avoir été rédigé à nouveau en FORTRAN standard. Une autre nouvelle caractéristique concerne la possibilité de l'utiliser en mode conversationnel (c'est-à-dire à partir d'un terminal équipé d'un clavier) aussi bien qu'en mode différé (c'est-à-dire en utilisant un lecteur de cartes et une imprimante par lignes).

INFOL a été conçu pour le stockage sur bande magnétique et il possède une structure de fichier séquentielle, avec hiérarchie intra-enregistrement. Les données sont décrites par l'utilisateur dans une phase de définition en utilisant ce qui serait maintenant appelé un langage de description de données (LDD). Dans ce cas la description des données consiste seulement en des descriptions logiques des articles à l'intérieur d'un enregistrement ; elle est stockée dans le fichier en étant séparée des données elles-mêmes. Dans l'enregistrement il ne peut y avoir qu'un seul article ayant le statut d'article-clé ; dans le fichier les enregistrements sont rangés d'après la valeur de cet article. Un article peut être déclaré multiple (il est plus généralement appelé répété), c'est-à-dire qu'il peut présenter un nombre indéfini d'occurrences.

Le chargement initial des données dans une base de données vide est appelé peuplement de cette base. Dans le cas du système INFOL cela peut être réalisé dans la phase de mise à jour dans laquelle des données déjà chargées peuvent également être modifiées ou supprimées.

Les articles sont vérifiés avant leur insertion. Comme exemple de validation automatique dans INFOL, notons qu'une date est vérifiée pour constater s'il s'agit d'une date calendaire correcte, en tenant même compte des années bissextiles. Du fait que la mise à jour se réalise par copie de la totalité du fichier séquentiel, on dispose habituellement d'un double de sécurité et ainsi l'intégrité est facilement garantie, au détriment de l'efficacité.

La recherche s'effectue au cours d'une phase d'interrogation en utilisant un langage d'interrogation dans lequel des critères de recherche peuvent être spécifiés, comme cela a déjà été décrit précédemment. Comme un enregistrement INFOL ne possède qu'un article-clé et que de toute manière il s'agit d'un fichier séquentiel, toute interrogation est virtuellement une question imprévue, ce qui implique une scrutation de tous les enregistrements constituant le fichier. Lorsque les enregistrements recherchés ont été découverts, ils peuvent être affichés partiellement ou complètement, selon des modalités faciles à définir, en utilisant un éditeur de puissance limitée qui n'offre pas la totalité de la gamme des facilités de mise en page que possèdent des éditeurs plus élaborés.

Finalement INFOL permet une restructuration par modification de la description des articles existants ou par addition de nouveaux articles. Ces opérations sont facilitées par le fait qu'après toute modification des données ou de la description, la base de données est copiée en totalité.

13. Les systèmes du CODASYL

Le rapport 1971 du Groupe de Travail sur les bases de données du CODASYL a défini des normes pour un SBDG devant être avant tout utilisé avec le COBOL comme langage hôte. Il est proposé d'utiliser une structure en réseau (voir paragraphe 9) pour modéliser les relations entre les enregistrements dans la base de données. Des relations hiérarchiques ne constituent qu'un cas particulier simple d'une chaîne d'ensembles en réseau ; elles peuvent donc également être représentées.

Le rapport fournit une syntaxe détaillée pour un langage de définition de données (LDD) et pour un langage de manipulation de données (LMD) en vue de leur inclusion dans le langage COBOL. Une description de base de données est appelée un schéma et l'aspect du schéma qui est nécessaire à un utilisateur programmant en COBOL, ou qu'il est autorisé à détenir, est appelé "subschema" (sous-schéma). Dans le LDD, l'utilisateur dispose d'un choix de méthodes d'accès pour rechercher des enregistrements dans des ensembles ou pour atteindre un type donné d'enregistrement ; de ce fait, l'utilisateur peut donc tenir compte de l'efficacité pour des recherches prévues. Lorsqu'il utilise le LMD, le programmeur travaillant en COBOL navigue en quelque sorte à travers le réseau, en passant d'un ensemble à un autre.

Aucun langage d'interrogation n'est proposé dans le rapport du CODASYL, mais par contre la question de l'interaction entre de multiples utilisateurs est traitée un peu superficiellement. Des règles sont prescrites pour éviter les conflits qui peuvent apparaître lorsque deux utilisateurs tentent de mettre à jour simultanément des données identiques ou étroitement liées.

De même, la sécurité à tous les niveaux est garantie pour restreindre l'accès à l'information contenue dans la base de données.

Parmi les systèmes CODASYL bien connus on peut citer l'IDMS (Cullinane Co pour les ordinateurs IBM, ICL et autres), le DMS 1100 (Univac), les DBMS-10 et -11 (DEC) et l'IDS-II (Honeywell).

14. Futurs développements des SBDG

Les SBDG relationnels ont été discutés au paragraphe 10 et c'est ce modèle de données qui actuellement retient le plus l'attention dans le développement de nouveaux systèmes. Indépendamment du modèle de données utilisé, il a été considéré comme important d'établir un cadre général de normes pour les futures conceptions des SBDG. Les propositions de 1975 de l'ANSI SPARC qui ont été très largement acceptées, prévoient une séparation nette entre : a) le schéma externe par l'intermédiaire duquel les programmes des utilisateurs ont accès aux données, et qui peut être des types relationnel, ou en réseau, ou correspondre à tout mode d'interface qui semble plus naturel à l'utilisateur, b) le schéma conceptuel qui contient la structure intrinsèque de l'entreprise modélisée, et c) le schéma interne qui contrôle le stockage physique dans la base de données. Le schéma interne peut être adapté, par exemple en ajoutant ou en supprimant des voies d'accès afin de correspondre à l'usage actuel de la base de données. EDMS (rédigé par CDC, Bruxelles et par certaines universités) est un système qui commence à se rapprocher de cette conception.

On peut penser que les futurs SBDG disposeront d'interfaces pour les utilisateurs permettant d'effectuer des interrogations dans un langage plus ou moins naturel ou dans un langage formel puissant, selon la préférence de l'utilisateur. Ces bases de données elles-mêmes pourront être réparties entre plusieurs ordinateurs intégrés dans un réseau de communications.

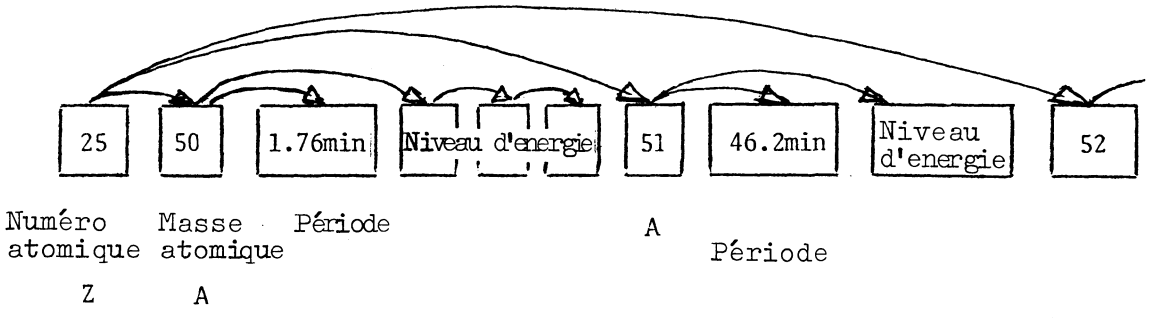


Fig. 1: Une hiérarchie intra-enregistrement à trois niveaux avec un unregistrement par élément. L'exemple présente une partie d'un enregistrement pour le manganèse, Z=25. Les flèches montrent la structure logique.

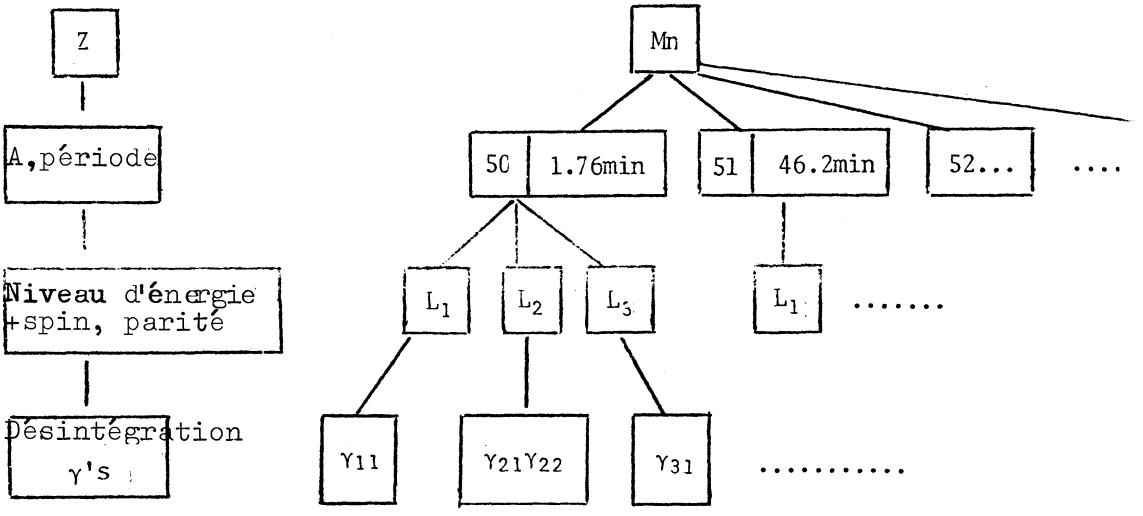
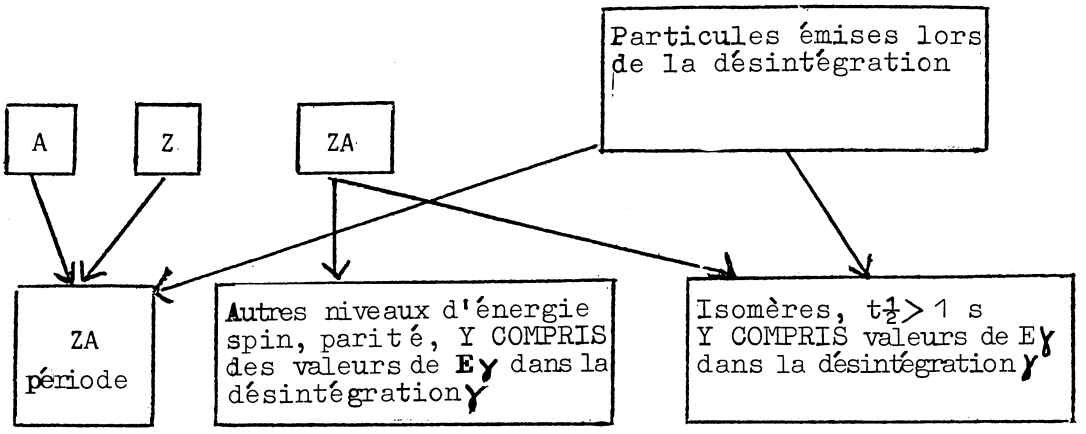
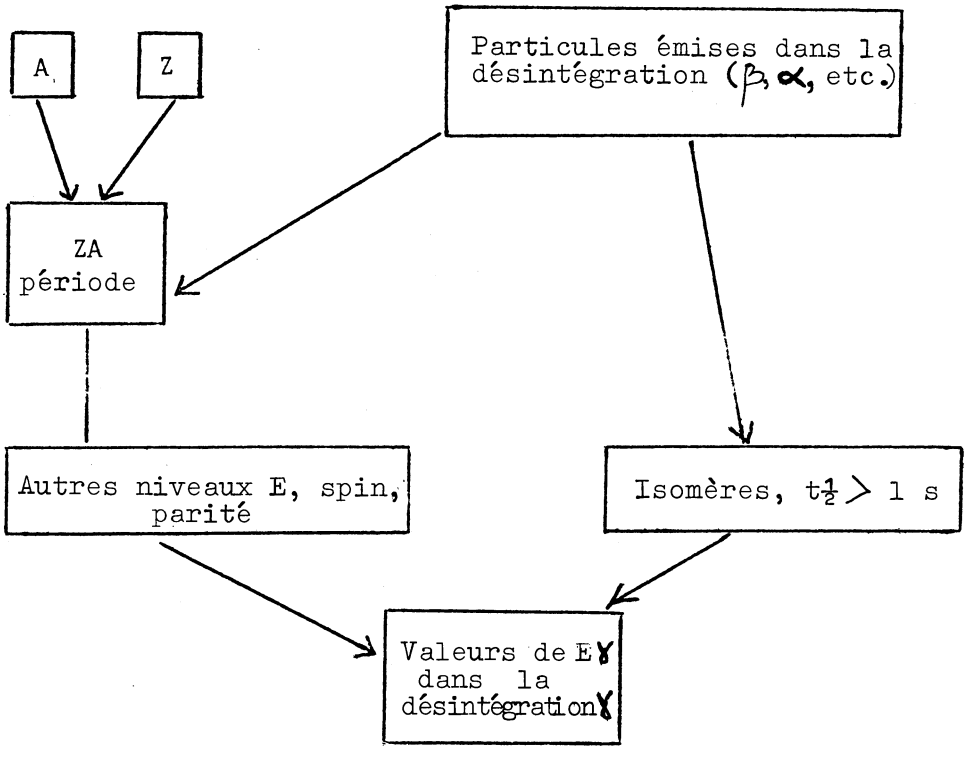


Fig. 2: Une structure hiérarchique complète présentant d'autres détails de la structure nucléaire. Sur la gauche, chaque cadre définit un groupe répété, et sur la droite, l'exemple montre comment une branche de l'article Z de la hiérarchie peut être peuplée.



a) Exemple d'une structure en réseau limitée



b) Exemple d'une structure complète en réseau

Fig. 3: Exemples de structure en réseau présentant d'autres détails de la structure nucléaire.

Domaine	Z	A	Période	...
Tuple 1	25	50	1.76 min	...
2	25	51	46.2 min	...
3	25	52	...	...
...	...	...		

Domaine	Z	A	Niveau d'E	Spin	Parité
Tuple 1	25	50	E_1	...	...
2	25	50	E_2	...	...
3	25	51	E_3	...	...
4	25	51	...	...	...
...	...	...			

Domaine	Z	A	Niveau d'E	Gamma	...
Tuple 1	25	50	E_1	γ_{11}	...
2	25	50	E_2	γ_{21}	...
3	25	50	E_2	γ_{22}	...
4	25	50	E_3	γ_{31}	...
...	...	...			

Fig. 4: Eléments de relations représentant des détails de la structure nucléaire correspondant à la fig. 2. Dans une base de données réele, les paramètres E_i et γ_{ij} seraient des valeurs effectives.

CONSIDERATIONS FINALES : L'ETUDE ET SES CONCLUSIONS

ORGANISATION DE L'ETUDE

Cette étude relative à l'utilisation des Systèmes de gestion de bases de données (SGBD) a permis à trente-sept participants appartenant à vingt-deux organismes de se réunir à deux reprises en Europe et aux Etats-Unis. Les organisateurs de ces réunions ont fait tout leur possible pour rassembler un groupe bien équilibré de spécialistes de l'information scientifique, de chercheurs astreints par leur travail à entretenir d'importantes collections de données, de spécialistes des systèmes de bases de données et de responsables de la gestion dans le domaine de la documentation et des données scientifiques.

La proportion relativement élevée de participants, dont les activités sont axées sur l'informatique, reflète la nouveauté que constitue l'utilisation des bases de données dans les travaux de documentation scientifique et dans les sciences elles-mêmes. Parmi les applications des Systèmes de gestion de bases de données, qui ont été examinées, très peu remontaient à plus de deux ans et nombre des études de cas considérées dans la troisième partie du rapport, ont trait à des travaux pilotes de mise au point et à des études de faisabilité. La plupart des applications scientifiques, qui sont au point et fonctionnent depuis un certain temps, sont encore trop modestes pour qu'on puisse tester la capacité réelle des systèmes de gestion de données sur lesquels elles s'appuient.

Dans ces limites, les contributions couvrent un très large éventail de sujets dans le domaine général du traitement de l'information scientifique : compilations de données numériques (qu'elles soient exécutées par des centres de données ou par des "chercheurs manipulateurs de données"), index bibliographiques et administration de bibliothèques. Les informaticiens, auteurs de contributions, s'occupent à la fois de la mise au point de Systèmes de gestion de bases de données et de l'administration de bases de données. Deux rapports sur des "systèmes", traitent d'une méthode hautement intégrée utilisée pour aborder des calculs scientifiques, mais limitée jusqu'à présent à la technologie nucléaire. Il s'agit de programmes modulaires employés pour une très large gamme de calculs de réacteurs, chaque module permettant d'accéder à un stock commun de données conservé par un système de gestion de bases de données spécialement conçu à cet effet.

CONCLUSIONS

Au cours de la deuxième réunion, tenue à Berkeley, le groupe d'étude s'est efforcé de tirer des conclusions quant à l'intérêt que présente la méthode de gestion intégrée de données pour la documentation et les

données scientifiques. On s'accorde à considérer que les Systèmes de gestion de bases de données sont intéressants pour un large éventail d'applications scientifiques et on a pu définir certains critères susceptibles d'aider les utilisateurs à décider s'ils doivent ou non adopter un tel logiciel pour résoudre leurs propres problèmes de traitement de l'information.

Critères applicables à l'utilisation des SGBD

Une approche intégrée de l'établissement des bases de données, utilisant un logiciel constitué par un SGBD, est susceptible d'être très intéressante lorsque sont remplies une ou plusieurs des conditions suivantes :

- lorsqu'une base de données doit être partagée entre plusieurs utilisateurs et éventuellement modifiée par ces derniers ;

- lorsqu'il existe entre les données des interrelations logiques raisonnablement complexes, qui doivent être rendues explicites en reliant éventuellement plusieurs collections distinctes de fichiers de données ;

- dans le cas de bases de données, dont les dimensions sont comprises entre quelques millions et quelques centaines de millions de caractères ;

- s'il n'est pas possible de prévoir les demandes des utilisateurs au moment où la base de données est établie ou si ces demandes peuvent évoluer avec le temps ;

- lorsque l'organisation ne possède ni le personnel, ni le temps, ni les connaissances nécessaires pour mettre au point un système spécialisé et pour en assurer l'entretien, ou si un SGBD est déjà disponible à l'intérieur de l'organisation ;

- si les utilisateurs des données ne sont pas des programmeurs d'ordinateurs et ne souhaitent pas être entraînés à exécuter des travaux de programmation.

L'effet des dimensions de la base de données

En ce qui concerne les bases de données, dont les dimensions se situent en-dessous d'un million de caractères et selon l'usage qui est fait de ces données, les besoins qu'un SGBD peut être appelé à satisfaire, peuvent être relativement insignifiants, de sorte que des programmes de mise en mémoire et d'extraction conçus sur place pourraient donner de bons résultats à meilleur compte. Lorsqu'on dispose déjà d'un SGBD et que les utilisateurs sont familiarisés avec son utilisation, certains d'entre eux préfèreront l'utiliser même pour l'application la plus simple car de cette manière, ils pourront exploiter toute la gamme des moyens offerts par leur SGBD sans avoir à prendre la peine et à encourir le coût de rédiger les programmes sous-jacents de traitement des données.

Les avantages des actuels SGBD sont les plus manifestes dans le cas des bases de données de quelques millions à quelques centaines de millions de caractères. Dans les zones intermédiaires, allant de 3×10^8 caractères et s'étendant au-delà de 10^9 caractères, les coûts de mise en mémoire sur disques ou sur d'autres dispositifs de stockage à grande vitesse, deviennent appréciables et la vitesse de fonctionnement des systèmes dans lesquels on accède à la mémoire externe par l'intermédiaire du système de gestion de fichiers du système d'exploitation peut ne plus être adéquate.

En ce qui concerne les très grandes bases de données, de quelques milliards de caractères ou davantage, un ordinateur spécialisé et l'investissement correspondant en logiciel spécial sont susceptibles d'être nécessaires et de se justifier, vu le coût de la saisie des données. Une base de données de cette importance contient probablement des données "brutes", provenant de mesures expérimentales présentant des interrelations et la grande latitude dans le choix de l'aspect macroscopique que les SGBD ont pour but d'offrir à l'utilisateur constitue certainement un facteur essentiel, si l'on veut que de telles masses de données soient convenablement assimilées. Alors que de très grands dispositifs de stockage à accès semi-rapide ont été mis au point, le logiciel d'accès présentant une efficacité correspondante n'est pas encore disponible et le système de gestion de données doit être néanmoins capable de fonctionner de façon efficace, la contrainte étant que la plupart des données se trouverait à un moment donné "renvoyées" de la mémoire rapide sur des dispositifs plus lents (habituellement des bandes).

Choix du logiciel de gestion de données

Les participants à la réunion sont convenus qu'il est stérile de proposer une quelconque définition rigide d'un SGBD, bien que la gestion d'une structure logique à l'intérieur de la base de données, permettant l'accès de l'utilisateur par l'intermédiaire d'un langage évolué, semble prometteuse en tant que critère en vue de différencier les SGBD des systèmes de gestion de fichiers et des systèmes d'extraction de l'information. De nombreux utilisateurs potentiels ne se préoccupent pas de savoir si un système donné est ou non un SGBD : ce qu'ils veulent, c'est un logiciel qui fasse leur travail.

Cependant, lors des échanges de vues sur les besoins en matière de systèmes, les participants ont, à de nombreuses reprises, insisté sur le fait que la base de données et son logiciel de gestion doivent être extensibles de manière à permettre des développements qui n'étaient pas envisagés lorsque le projet a d'abord été préparé. Il n'est pas certain que l'on réalisera des économies en utilisant un SGBD plutôt qu'en rédigeant des programmes spécialisés, mais les utilisateurs disposeront de davantage de souplesse et seront ainsi capables d'utiliser des données à long terme mieux qu'ils ne le pourraient autrement. Les chefs de projet seraient avisés de choisir un logiciel de gestion de données qui offre davantage de souplesse que ce qu'ils jugent susceptible de leur être nécessaire.

C'est dans ce contexte que les lecteurs peuvent estimer utile d'étudier les rapports présentés dans la deuxième partie relative aux SGBD scientifiques et aux spécifications applicables au traitement des données scientifiques : il est surprenant de constater combien certaines de ces analyses indépendantes se recoupent et des lecteurs peuvent aussi estimer que certaines de ces caractéristiques leur seront également utiles. Le Chapitre 2 donne une liste de logiciels de traitement des données, qui est nécessairement incomplète, mais qui couvre une gamme de programmes-produits allant de SGBD complets à des systèmes de gestion de fichiers commercialisés indépendamment des constructeurs, des systèmes d'extraction de l'information principalement conçus pour la recherche dans le texte et des programmes d'édition (Report Generators). Il offre un point de départ possible pour décider quels programmes-produits il convient d'examiner en vue d'un projet spécifique de documentation et de données scientifiques.

De très grandes ou de très petites bases de données peuvent être mieux desservies par des programmes spécialisés. Dans le cas de très petites bases de données, le Chapitre 4 présente une méthode utilisant un langage de programmation particulier très évolué, l'APL. L'auteur démontre qu'une collection de données apparemment complexe peut souvent, dans la pratique, être décomposée en un certain nombre de bases de données plus petites, plus simples et pratiquement indépendantes, qui peuvent être administrées à meilleur compte à l'aide de programmes spécialisés. Des programmes structurés, convenablement rédigés dans un langage évolué, devraient alors pouvoir être aisément développés à mesure que le besoin s'en fait sentir. Selon l'auteur de ce rapport, la nécessité de protéger les données (d'assurer la sécurité et l'intégrité dans des bases de données partagées, par exemple) qui est prévue dans le cas de nombreux SGBD, est tout aussi importante que la complexité logique, lorsqu'on décide s'il y a ou non lieu d'utiliser un Système de gestion de bases de données.

Limites des SGBD actuellement disponibles

Bien que les spécifications relatives aux SGBD scientifiques, présentées dans la deuxième partie, ne soient pas, pour une bonne part, satisfaites par les systèmes actuellement disponibles sur le marché, il semble manifeste que dans le cas d'un projet déterminé de données scientifiques, l'on puisse trouver un ou plusieurs systèmes qui exécuteront la plupart des tâches souhaitées par les programmeurs chargés du projet et les aideront ainsi à parvenir à l'objectif visé, qui est d'établir un système global satisfaisant. Il existe trois limites très évidentes, à savoir :

- Il n'est pas encore possible d'offrir une représentation parfaitement générale des structures de données sans sacrifier la performance. La plupart des systèmes disponibles dans le commerce offrent un modèle de données plus limité, qui peut obliger l'utilisateur à adapter ses applications aux contraintes du système.

- En particulier, les systèmes disponibles dans le commerce n'admettent pas certaines des structures inhérentes aux données scientifiques, telles que les vecteurs et les rangées. La plupart de ces systèmes ne reconnaissent pas des types de données numériques telles que la virgule flottante : ces données sont traitées comme des caractères et doivent être interprétées en dehors du SGBD. Il semblerait également naturel de stocker des tableaux de données et des suites de textes dans les enregistrements de longueur variable en des points appropriés de la structure logique de la base de données. Dans le cas des données scientifiques, cette méthode donnerait lieu à des variations extrêmes de la longueur des enregistrements (de un à deux ordres de grandeur) et ne peut être aisément mise en œuvre dans les systèmes actuels.

- Les systèmes actuellement disponibles sur le marché ne prévoient pas de moyens de restructurer complètement les données. La restructuration fondamentale d'une base de données, dont l'utilisation a été modifiée, peut être coûteuse en temps de programmeur et en temps d'exploitation de l'ordinateur.

Le coût d'utilisation d'un SGBD

Le Chapitre 3 expose une méthodologie permettant de déterminer les coûts de mise en place d'une application de SGBD et de les comparer au coût du logiciel spécialisé dans le cadre d'un projet particulier. On peut conclure en général que, lorsque la mise en place d'un SGBD est à la charge du projet, les coûts répartis sur toute la durée du projet sont moindres si on les compare à ceux des programmes spécialisés, mais les mises de fonds initiales seront du même ordre. Toutefois :

- La deuxième application utilisant le SGBD et celles qui suivent, seront moins coûteuses (les coûts du SGBD étant déjà amortis).

- Les applications utilisant les SGBD seront plus souples et auront donc une durée de vie utile plus longue.

- Les coûts réguliers de maintenance seront inférieurs si l'on utilise un SGBD.

Mise à part l'acquisition initiale du programme-produit de gestion de bases des données, il faut tenir compte des frais de formation et d'entretien relatifs aux SGBD, du coût supplémentaire probable du stockage sur disques qui est nécessaire et de l'utilisation en ligne de l'ordinateur, ainsi que du temps-machine requis. Les principaux avantages du SGBD tiennent à ce qu'il permet une meilleure utilisation de la base de données et du personnel informaticien plutôt qu'à des avantages financiers à long terme. Il est nécessaire de poursuivre l'étude des coûts et avantages des SGBD par rapport à la programmation simple, mais cette étude présentera des limites, précisément du fait que ces applications sont rarement comparables : les programmeurs de SGBD seront tentés d'utiliser le système à plein et de faire davantage avec les données disponibles.

Performance des SGBD

La présente étude n'a permis d'obtenir que des renseignements très limités sur les performances des bases de données. De futurs utilisateurs souhaiteront certainement comparer les temps d'exploitation d'une application particulière utilisant soit un SGBD, soit des programmes spécialisés, en particulier dans le cas de grandes bases de données sur de petits ordinateurs, où des performances médiocres pourraient rendre le SGBD impraticable. Lorsque les utilisateurs ont éliminé les erreurs de conception de la base de données (considérées ultérieurement comme étant "évidentes"), existe-t-il des effets non linéaires, qui entraîneront une dégradation des performances à mesure que la base de données se remplira complètement ? La plupart des applications de SGBD sont nouvelles, portant sur de petites bases de données, plutôt que sur des conversions de fichiers existants, de sorte que là encore les comparaisons sont difficiles.

Le Centre de Compilation de Données Neutroniques (CCDN) de l'AEN est, parmi les utilisateurs de SGBD, celui qui est doté du plus petit ordinateur, à savoir l'IBM 370/125, ayant une mémoire centrale de 128K octets, lequel doit ultérieurement être remplacé par un DEC PDP 11/70. Des essais portant sur des fractions raisonnablement importantes de la base de données projetée (à l'aide du logiciel IDMS, qui est un système généralisé de gestion des données conforme aux spécifications CODASYL), semblent indiquer que dans le cas d'un "schéma" logique bien conçu, les performances d'extraction obtenues sur le petit ordinateur IBM, seront comparables à celles des programmes existants utilisant des fichiers en séquentiel indexé. Le chargement des données, sans l'utilitaire de

chargement rapide, peut exiger jusqu'à 60 heures de temps d'exploitation pour l'ensemble de la base de données, mais cela est considéré comme acceptable pour une opération unique, qui peut être interrompue à volonté. Une comparaison limitée des performances semble indiquer que l'ensemble de l'exécution des fonctions IDMS, notamment le chargement, sera plus rapide sur le PDP 11/70.

Alors que ces conclusions provisoires ne peuvent être extrapolées qu'avec une extrême prudence à d'autres bases de données et à d'autres ordinateurs, il ressort de l'expérience pratique acquise par le CCDN, que la méthode des SGBD est viable même avec un matériel très modeste.

Evolution future des SGBD

Il est à espérer que les producteurs de logiciels reconnaîtront la place importante que les SGBD sont appelés à prendre dans le traitement de l'information scientifique et tiendront compte, dans leur conception des systèmes, des spécifications étudiées à la deuxième partie du rapport. La gamme des applications que la prochaine génération de SGBD devra couvrir, est si variée qu'il semble peu probable qu'un seul système soit capable de les traiter toutes convenablement. Une approche modulaire des SGBD peut s'avérer plus satisfaisante.

Un obstacle important, auquel se heurte l'utilisation des SGBD par des non-programmateurs, tient au fait que les systèmes actuels ne permettent pas de parvenir à une indépendance totale à l'égard des données. On devrait s'attacher à l'avenir à élaborer, à l'intention des utilisateurs, un modèle logique (structures des données et langage de manipulation) indépendant de la représentation physique de la base de données. La baisse continue des coûts des composants du matériel peut rendre supportables les frais supplémentaires d'exploitation nécessaires pour assurer cette indépendance totale à l'égard des données. Le temps du personnel, bien plus que le temps-machine, deviendra le principal facteur de coût dans le traitement des données et dans la conception des futures SGBD, on devrait continuer à s'efforcer de l'économiser.

AUTHOR INDEX, WITH ADDRESSES OF PARTICIPANTS

REPERTOIRE DES AUTEURS, AVEC LES ADRESSES DES PARTICIPANTS

Attree, Ms. P. (Chapter 24)	International Atomic Energy Agency Nuclear Data Section Kärntnerring 11, 1011 Vienna Austria
Bau, Dr. W	See Behrens (Chapter 23)
Behrens, Dr. H. (Chapter 23)	Zentralstelle für AtomkernEnergie Dokumentation (ZAED), Kernforschungs- zentrum Karlsruhe, 7514 Eggenstein- Leopoldshafen, F.R. Germany
Birss, Dr. E. (Chapter 8)	Lawrence Livermore Laboratory Box 808, Livermore, California 94550 U.S.A.
Brooks, Dr. A. (Chapters 5 and 27)	Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830 U.S.A.
Collica, Dr. J	See Deutsch (Chapter 3)
Deutsch, Dr. D. (Chapters 2, 3 and 21)	National Bureau of Standards Washington, D.C. 20234 U.S.A.
Dunford, Dr. C. (Chapter 18)	National Nuclear Data Centre Brookhaven National Laboratory Upton, N.Y. 11973 U.S.A.
Fong, Ms. E	See Deutsch (Chapters 2 and 3)
Fuja, Ms. P. (Chapter 12)	Argonne National Laboratory Argonne, Illinois 60439 U.S.A.
Gersbacher, Dr. W. (Chapter 6)	Battelle Memorial Institute Information Systems Section Battelle Columbus Laboratories Columbus, Ohio 43201 U.S.A.
Gottschalk, Dr. C.	Office of Technical Information Department of Energy Washington, D.C. 20545 U.S.A.
Haire, Ms. G.	Lawrence Berkeley Laboratory Information Research Group Berkeley, California 94720 U.S.A.
Hampel, Dr. V. (Chapter 7)	Lawrence Livermore Laboratory Box 808,Livermore, California 94550
Hilsenrath, Dr. J.	National Bureau of Standards Washington, D.C. 20234

Honeck, Dr. H. (Chapter 10)	Savannah River Laboratory Aiken, South Carolina 29801 U.S.A.
Hsu, Dr. K. (Chapter 16)	Battelle Memorial Institute Information Systems Section Battelle Columbus Laboratories Columbus, Ohio 43201 U.S.A.
Hughes, Mr. T.	Library of Congress Science and Technology Division Washington, D.C. 20540 U.S.A.
Johnston, Dr. P.	OECD Nuclear Energy Agency Neutron Data Compilation Centre (CCDN) B.P. No. 9, 91190 Gif-sur-Yvette France
Jones, Dr. E. (Chapter 8)	Lawrence Livermore Laboratory Box 808, Livermore, California 94550 U.S.A.
Knoll, Dr. D.	National Oceanographic Data Centre 2001 Wisconsin Avenue Washington, D.C. 20235
Leralle, Mr. J.	See Martin (Chapter 15)
Lindeman, Mr. A.	See Fuja (Chapter 12)
Martin, Mr. G. (Chapters 4 and 15)	Compagnie Internationale de Services en Informatique (CISI) B.P. No. 24 91190 Gif-sur-Yvette France
Montgomery, Mr. K. (Chapters 11 and 20)	United Kingdom Atomic Energy Authority Risley, Warrington WA3 6AT U.S.A.
Moorhead, Dr. G. (Chapters 1 and 14)	CERN Laboratoire I DD-Division 1211 Geneva 23 Switzerland
Nevyjel, Dr. A. (Chapter 22)	Österreichische Studiengesellschaft für Atomenergie, GmbH, Lenaugasse 10, 1082 Vienna Austria
Pellegrino, Mr. L.	See Schofield, Tubbs (Chapter 19)
Perschke, Dr. S.	Commission of the European Communities Joint Research Centre 21020 Ispra (Varese) Italy
Petrie, Dr. J. (Chapter 26)	Commission of the European Communities Joint Research Centre 21020 Ispra (Varese) Italy
Powell, Mr. J. (Chapter 26)	Commission of the European Communities Joint Research Centre 21020 Ispra (Varese) Italy

Richards, Dr. D. (Chapter 9)	Lawrence Berkeley Laboratory Computer Science Berkeley, California 94720 U.S.A.
Ries, Dr. D. (Chapters 7 and 8)	Lawrence Livermore Laboratory Box 808, Livermore, California 94550 U.S.A.
Rietveld, Dr. H. (Chapter 17)	Netherlands Energy Research Foundation Dept. of Scientific and Technical Information, Petten (N.H.) The Netherlands
Rittenberg, Dr. A.	See Stevens (Chapter 13)
Robinson, Ms. J.	Lawrence Berkeley Laboratory Information Research Group Berkeley, California 94720 U.S.A.
Schofield, Dr. A. (Chapter 19)	OECD Nuclear Energy Agency Neutron Data Compilation Centre (CCDN) B.P. No. 9, 91190 Gif-sur-Yvette France
Schuler, Dr. W. (Chapter 25)	OECD Nuclear Energy Agency Computer Program Library C.P. No. 15, 21027 Ispra (Varese) Italy
Shoshani, Dr. A. (Chapter 28)	Lawrence Berkeley Laboratory Computer Science Berkeley, California 94720 U.S.A.
Stevens, Dr. P. (Chapter 13)	California Institute of Technology Physics Department 356-48 Pasadena, California 91125 U.S.A.
Suich, Dr. J. (Chapter 10)	Savannah River Laboratory Aiken, South Carolina 29801 U.S.A.
Szczesny, Dr. K. (Chapter 6)	Battelle Memorial Institute Information Systems Section Battelle Columbus Laboratories Columbus, Ohio 43201
Town, Dr. W.	See Petrie, Powell, (Chapter 26)
Tubbs, Dr. N. (Chapters 1, 19 and appendix)	OECD Nuclear Energy Agency 38 Boulevard Suchet 75016 Paris France
Yamamoto, Dr. T. (Annex to Chapter 2)	University of Tokyo, Computer Centre 2-11-16 Yayoi, Bunkyo-Ku, Tokyo 113 Japan
Yeb, Mr. J.	See Birss, Jones, Ries, (Chapter 8)