
CANADA

IDRC-TS14e

# Information Retrieval and Library Management: An Interactive Minicomputer System

Faye A. Daneliuk

The International Development Research Centre is a public corporation created by the Parliament of Canada in 1970 to support research designed to adapt science and technology to the needs of developing countries. The Centre's activity is concentrated in five sectors: agriculture, food and nutrition sciences; health sciences; information sciences; publications; and social sciences. IDRC is financed solely by the Government of Canada; its policies, however, are set by an international Board of Governors. The Centre's headquarters are in Ottawa, Canada. Regional offices are located in Africa, Asia, Latin America, and the Middle East.

©1978 International Development Research Centre
Postal Address: Box 8500, Ottawa, Canada K1G 3H9
Head Office: 60 Queen Street, Ottawa

Daneliuk, F. A.
IDRC IDRC-TS14e
Information retrieval and library management: an interactive minicomputer system. Ottawa, Ont., IDRC, 1978. 16p. : ill.

/ IDRC publication /. Monograph on the mini / computer / -based / information system / at / IDRC / — discusses the background and / feasibility study / of the implementation of the bibliographic information system on a minicomputer, / systems design / and / computer programme / ing work done at IDRC; includes a / bibliography /.

UDC: 002 ISBN: 0-88936-170-3

Microfiche edition available

IDRC-TS14e

# Information Retrieval and Library Management: An Interactive Minicomputer System

**Faye A. Daneliuk**

*Project Manager, Minicomputer Project*
*International Development Research Centre*

*This is a revised version of a paper prepared by the same author, which appeared in "The application of inexpensive minicomputers to information work." AGARD Lecture Series No. 92, ISBN: 92-835-1276-6 (North Atlantic Treaty Organization, Neuilly-sur-Seine, March 1978).*

# Foreword

In the last 10 years, the operating speed and reliability of computers have increased markedly. At the same time, thanks to integrated microelectronic circuits, the physical size and the cost of storage have been reduced by factors ranging from 100 to 1000. Many sophisticated hardware capabilities, a few years ago associated only with expensive large-scale computers, are now being offered as standard items on low-cost minicomputers. Coupled with this the minicomputer manufacturers are now providing fairly elaborate operating system software responding to the diverse needs of the users.

Thus, when new applications requiring automation are being considered, users are turning more and more toward the minicomputer. Governments, institutions, and industries with limited budgets have opted for minicomputers to meet their specific needs. Some of these users have not only proved that minicomputer-based systems are cost-effective but have shown a high level of sophistication in the use of the equipment.

This book describes the International Development Research Centre's minicomputer-based bibliographic information processing system developed in Ottawa over the last two and a half years. The system is now operational on our in-house Hewlett-Packard 3000 Series II minicomputer and provides many automatic procedures for managing IDRC's library needs and permits retrieval from several large data bases. The design of the system draws heavily on the experience with ISIS (Integrated Set of Information Systems), a software package that was developed by the International Labour Office in Geneva to run on IBM 360 / 370 computers. IDRC's minicomputer-based system is compatible with ISIS but is in no way a rewrite of ISIS.

IDRC's system has been designed using "state-of-the-art" data-base architecture and computer technology, and is sufficiently generalized to permit the creation of data bases for many different types of applications.

One of the main reasons IDRC embarked on this project is its involvement and emphasis on international information systems. Many developed and developing countries have expressed interest in acquiring a low-cost computer system to allow them to participate in decentralized international information systems. IDRC's minicomputer system has taken into account the requirements for decentralized systems such as AGRIS and DEVSIS, including the processing of data bases formatted both to UNISIST and non-UNISIST rules and the Selective Dissemination of Information.

A concerted effort to provide a complete package (including full documentation, training, and implementation assistance) is being made, and it is anticipated that the package will be ready for distribution in the near future.

**Sultan Kassum**

*Manager, Computer Systems*
*Information Sciences Division*
*International Development*
*Research Centre*

## Introduction and Background

The International Development Research Centre is a public corporation created in 1970 by an Act of the Canadian Parliament. It is an autonomous body with a 21-member Board of Governors (drawn from several countries) which sets the broad lines of policy and approves individual projects. The Centre's headquarters are in Ottawa, with regional offices in Singapore, Cairo, Bogota, Dakar, and Nairobi.

The objectives of the Centre (in the words of the Act) are "to initiate, encourage, support, and conduct research into the problems of the developing regions of the world, and into the means for applying and adapting scientific, technical, and other knowledge to the economic and social advancement of those regions," and "to help developing regions build up their own research capabilities and the innovative skills needed to solve their problems." In order to carry out these objectives, it was empowered to "establish, maintain and operate information and data centres and facilities for research and other activities relevant to its objects" and "initiate and carry out research and technical development, including the establishment and operation of any pilot plan or project, to the point where the appropriate results of such research and development can be applied."

Five program divisions exist: Health Sciences; Agriculture, Food and Nutrition Sciences; Social Sciences and Human Resources; Information Sciences; and Publications. The Information Sciences Division is rather unique as its establishment marked the first instance in which an aid organization created a program division with the specific objective of supporting information projects in developing countries. It is this division that has involved itself with the application of computers to information work.

In 1972 a project that dealt with computerized information systems was approved by the Board of Governors of the Centre. The purposes of the project were fourfold:

(1) to acquire an on-line system that would enable us to computerize our library operations;

(2) to build a machine-readable data base of our own development literature;

(3) to work at an international level with other institutions with a view to the development of a cooperative "network" with a "common" system;

(4) to gain experience that would enable IDRC personnel to aid in the establishment of input / output stations in developing regions.

The system acquired through this project was ISIS (Integrated Set of Information Systems) which had been developed over a period of years by the International Labour Organization in Geneva. ISIS was chosen over a number of other systems, including commercial systems, because: it provided an

interactive mode for data entry and retrieval; it provided considerable batch functions for library management; it was international — at the time, the Mexican Government Office of Information and Labour was installing it, SAFAD (Swedish Agency for Administrative Development) was using it, both the ILO and UNIDO (United Nations Industrial Development Organization) were using it in Geneva, and the Rumanian Government had installed it at the Bucharest Management Centre for Documentalists and Librarians; it provided facilities for the exchange of data bases via magnetic tapes formatted according to ISO 2709; and it gave us immediate access to two data bases containing development literature.

Programed in IBM 360 Assembler language, ISIS required as a minimum a computer installation running under DOS (Disk Operating System) on a 360 machine. For IDRC, this meant installing ISIS at a commercial service bureau because acquiring a 360 facility with telecommunications equipment would have meant a prohibitive outlay of funds. The use of a service bureau is not necessarily an inexpensive approach and the project proposal, recognizing this, had left the division with the future option of renting or buying a computer system (either a machine compatible with the ISIS software or one for which we would design new software). In 1975, after 2.5 years of ISIS operations, it was decided to investigate the possibility of acquiring an in-house computer.

## Acquiring an In-House Computer Facility

In April 1975, Dr Gordon Somerfield was hired as a consultant by the Information Sciences Division to study the feasibility of implementing a bibliographic information system on a minicomputer. The resulting report, which took into account such factors as costs and software development, was positive in its conclusions and justified more intensive investigation on our part.

In the following 8 months a critical evaluation was made of all "stable" minicomputer manufacturers in the market and of their products. The cost of the equipment, although an important factor, was not the only factor taken into consideration. Because we were intending to do our software development in-house, the extensiveness and reliability of the manufacturer's software was studied in great detail. Our end-product (the information system software) was being developed not only for use within IDRC, but also for use in areas where an inexpensive, reliable facility rather than a large-capacity one would be the requirement. This meant that we had to have some assurance that the manufacturer would remain in business for some time. The manufacturer also had to provide some form of service for his equipment in Africa, Southeast Asia, and South and Central America, for we were hoping to make our software available to institutions in those areas. (This latter requirement was not met by any of the minicomputer manufacturers!)

At the same time, we contacted other institutions where information systems were being developed for minicomputers. In some of these institutions, a machine had been selected to run a dedicated system, and our first inclination was to adopt the same procedure. It can be easily understood that if a machine is dedicated to one application then the manufacturer's software need not be overly sophisticated. One can almost accept having to write terminal I / O handlers, file handling systems, and the like if they are to be used in a restricted manner. However, during this period of evaluation, an

on-going dialogue was taking place between Information Sciences and other divisions within the Centre. It was decided that the acquisition of a somewhat more sophisticated computer could be of great benefit to the Centre itself. This led to a narrowing of the field of potential suppliers.

In early 1976, a project proposal dealing with both the acquisition of the minicomputer system and software development, was presented to the Board of Governors by the Information Sciences Division. The project, as approved by the Board, specified three major reasons for a minicomputer:

(1) to reduce operational costs of running ISIS at IDRC (this was a significant factor — running costs at a service bureau can be extremely high);

(2) to define an optimum cost-benefit minicomputer installation that could be offered, complete with programs, for AGRIS (Agricultural Information System of the Food and Agricultural Organization of the United Nations) / DEVSIS (Development Sciences Information System) / ISIS activities at national centres in developing countries;

(3) to provide a basis for a Canadian input / output centre to a future international network for development information (DEVSIS).

The three-member "computer group" within the Information Sciences Division drew up a tender that stressed three characteristics: (1) the power of the operating system; (2) the reliability and availability of the manufacturer's software; and (3) the potential of the machine to handle the job mix. The same group also designed and conducted benchmark tests that emphasized the three critical specifications in the tender. Hewlett-Packard won the tender with their 3000 system. Although the HP-3000 had some shortcomings, which are only now being corrected (specifically, labeled tape processing and support for private disk volumes), some fine hardware features (stack architecture) and their sound, integrated operating system (MPE — Multiprogramming Executive) helped them to win the tender. In August 1976 our equipment was delivered and development began on the software.

## Developing a System

Once the computer was selected, thoughts turned to the system design. In March 1976 a final decision had yet to be made that we were indeed going to develop new software for our information system. Other alternatives did exist. We could simply recode the ISIS programs for the HP-3000 or we could adopt the data-base management package (IMAGE) developed by Hewlett-Packard for the 3000. The first alternative, although providing a quick and easy solution to a system-design problem, would have proven unrealistic because it could not have taken advantage of the special features of the HP-3000. The second alternative demanded more consideration but was finally rejected for reasons familiar to those who have worked with bibliographic systems: (1) no capability for handling true variable-length records; (2) no capability for handling variable length, variably occurring fields; (3) no capability for handling subfields; (4) no capability for supporting keys embedded in text; and (5) no capability for handling long descriptive abstracts. A new design was definitely called for!

As a first step in designing the system, a number of guiding principles were adopted:

(1) general applicability — the system should be as general purpose as possible;

(2) modularity — the system should be totally modular to promote ease of maintenance and extension;

(3) independence — the applications functions should be independent of the data-base management functions;

(4) user considerations — (a) the system should be flexible in that it should be capable of handling data in almost any physical form; (b) the system should be simple to understand so that it could be implemented and used with minimum effort; (c) a user-attractive language should be provided so that users could really be users; and (d) the system should be able to provide a wide variety of outputs;

(5) mission orientation — (a) the system should have the capacity to accept outputs from other information systems; (b) the system should be viable within a small organization; and (c) the system should be compatible with other international systems, specifically ISIS and AGRIS;

(6) cost-effectiveness — the basic system should be in operation by December 1977 so that we could dissociate ourselves from the service bureau where we had been running ISIS.

## Theoretical Foundations

In designing a system, a set of guiding principles, though very important, is not sufficient. Also required is a theoretical framework around which to build the system. This framework provides a coherence that otherwise would be difficult to realize. Careful study was given to the three prominent data-base management theories: the CODASYL (network) approach; the hierarchical approach; and the relational approach. It was finally decided to employ the relational approach in the system's design because this model of data was seen to have a number of inherent advantages not shared by other models.

The relational model is based on the mathematical theory of relations. Although there is a mathematical definition of relations (Date 1975; Codd 1970) one can think of a relation as being a collection of unique "flat" records made up of any number of elementary data items. This implies that one can think of a relation as a two-dimensional table in which: (1) no two rows are identical; (2) at every row / column position within the table there is only one value, not a set of values (i.e. repeating groups are not allowed); (3) columns are homogeneous; and (4) non-key fields are functionally dependent on the primary key. Given such a structure a generalized set of relational operators are defined for the manipulation of the data at both the domain (or field) level and the relation ("file") level. These operators are defined in the relational algebra (Codd 1972) and the domain algebra (Merrett 1976) (Codd 1971, 1972 also defined a relational calculus). An operation executed using relational algebra is one that takes one or more relations as its operands and produces a relation as a result; domain algebra does the same thing with domains.

Three basic operators exist within relational algebra: join; project; and storage. Join is by far the most powerful command available in the algebra; basically, it permits you to "put together" two or more relations using a domain as a bond. For example, the natural join (which may be thought of as a generalization of a Boolean AND) of two relations that each have a common domain will result in a relation in which are contained only those tuples that had identically matching values in the common domain. For example, if you had the two relations ACRO and CORP:

| ACRO | ACRONYM | FULL NAME |
|---|---|---|
| | ACE | American Council on Education |
| | ALA | American Library Association |
| | AACC | American Association of Cereal Chemists |

| CORP | CORPORATE ACRONYM | CORPORATE CODE |
|---|---|---|
| | CIDA | 000484 |
| | ACE | 004310 |
| | ALA | 000062 |
| | ECLA | 000573 |

and you executed the natural join of these two relations on the acronym domain the resulting relation would be:

| ACRO_CORP | ACRONYM | FULL NAME | CORPORATE CODE |
|---|---|---|---|
| | ACE | American Council on Education | 004310 |
| | ALA | American Library Association | 000062 |

Five other joins have been defined and can be investigated in the literature (Codd 1972; Merrett 1976).

Projection is an operation that enables one to generate a new relation that is composed of one or more columns of the table that was the original relation. For example, the projection of our newly created ACRO_CORP relation on full name and corporate code would result in the relation:

| FULL NAME | CORPORATE CODE |
|---|---|
| American Council on Education | 004310 |
| American Library Association | 000062 |

Storage (Date 1975) is a name applied to those operations that effect the insertion and deletion of tuples (or "records"). To insert a tuple into a relation one uses the set union operation. This union produces a set that contains all elements of the original sets. Deletion, on the other hand, uses the set

difference operation. The difference will produce a set in which are contained all those elements of the first set that are not contained in the second set.

Domain algebra was, in its practical form, introduced by Merrett in 1976. It allows "domains to be the operands and the results of the usual arithmetic, logical and string operators." The operand domains and the resulting domains may be (and usually will be) virtual — they exist only as a defined result that can be actualized on output commands. The operators of domain algebra may be vertical or horizontal. Horizontal operators work on one or more domains, a row at a time; whereas, the vertical operators work on a column at a time, across all rows. For example, the total cost of all books ordered by a library would be the result of a vertical operation; whereas, the number of days taken by a supplier to fill an order would be the result of a horizontal operation (date book received minus date book ordered).

Within the confines of the relational model, the basic "data containers" are the relations. An individual making use of the data base (call this person a user) interfaces with these relations through a data submodel. A data submodel enables the user to see his own view of the data contained within the data base. The submodel is defined using a data definition, and is usually the result of operations (on a relation or set of relations) that were executed by the data-base management routines at the request of an application program. Although all three theoretical approaches to data-base management provide some facility for the "redefinition" of data, only the relational model provides a uniform interface for accessing the data at all levels.

## A Practical Realization

In any implementation of a data-base system, the definition of data is critical — if data are not well defined, they cannot be used. The data must have a structural definition, and their relationship to other data in the system must be understood before they can be used. This information is essential to the end-users of the data base, to the data-base management system itself, and to the interface between the two. We therefore have three "views" of the data: an "external" view that is seen by the user; an "internal" view that is seen by the operating system; and an "intermediate" view that relates the external and the internal views to one another. Within the IDRC's system a data-definition facility was implemented to deal with this problem.

Four operational levels have been identified for the IDRC data-base system:

(1) end-user — the end-user of our system may be a researcher, a librarian, or a casual user;

(2) system-manager — this person is akin to the data-base administrator who is so often discussed in the literature;

(3) programmer — this is the applications programmer who writes the code that serves as the interface between the end-user and the data-base management system;

(4) data-base management system — this level sits just above the operating system. All physical access to the data base is controlled by the routines at this level.

Let us start with the end-users. The user will define a data submodel by providing to the system manager the following information: (1) the name by which the submodel is to be known; (2) a description of the fields of which the submodel is comprised — this description includes the length of the field, the names of the field, an indication as to whether the field is repeatable, field type (numeric or character), and an indication of whether the field could be hierarchically processed, among other things; (3) a specification of a default display format; and (4) an indication of whether any "fast" access paths are required for this submodel, and the fields on which they would be realized.

This information is reprocessed by the system manager to produce an intermediate view of the data for the system. Questions answered at this stage are: (1) Does this data exist elswhere? (2) Is the data volatile? (3) Are "fast" access paths justified for the submodel? (4) Will access be read only, or read-write? (The system manager must take care in defining this intermediate view because the independence of the data within the system must always be maintained.) At this point the data-definition processor is invoked and the system manager enters the definition into the system.

At some time later when the user wishes to access the submodel, the data-base management system processes the data definition and presents to the application program, the user view, and to itself, the internal view. (Keep in mind that many different relations may be used to represent the data submodel as defined by the user. These relations in themselves may be actual or virtual, and may or may not represent a single physical file!) The internal view is always in terms of relations and domains. Figure 1 illustrates the logical path from the end-user to the physical data base.

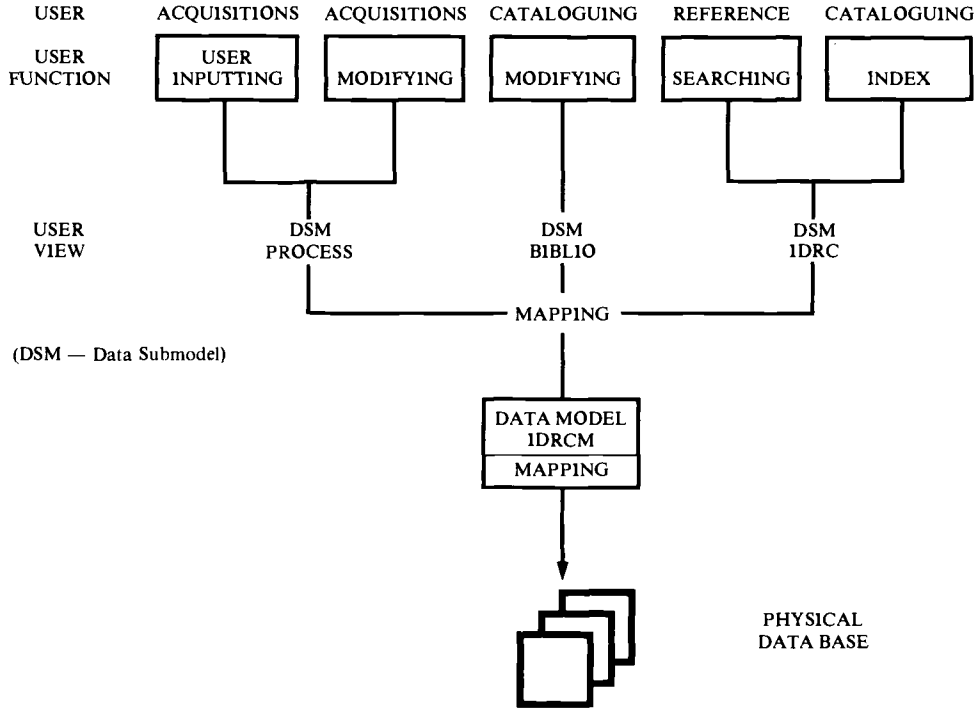


*Fig. 1. The logical path from the end-user to the physical data base.*

Having provided a tool for dealing with the various forms of data, we may now look at the functions provided at each level for handling the data. At this point, it is necessary to introduce four concepts. They are in order of use:

(1) processor — an application routine that implements a particular function.

(2) process — the unique execution of a program by a particular user at a particular time.

(3) stream — this term is, to the best of my knowledge, unique to the HP-3000 system. Streaming spools batch jobs (or data) during either interactive sessions or jobs. The spooled job is then scheduled by the operating system, and runs independently of the session or job that streamed it; control returns to the "streamer."

(4) restrict — to select a subset of a relation on the basis of certain criteria, the criteria usually being the values held by particular domains. To search for all those records in a data base in which the affiliation is the Ford Foundation is to do a restriction of the data base on affiliation.

The functions at each operational level will be discussed.

## End-User

For the end-user, an attempt was made to provide a simple yet powerful language that would make it possible to get full benefit from the system. The user's requirements were divided into two categories: data entry and data retrieval. Our experience with an automated system had made us aware of problems that were often overlooked; for example, record numbers can be a nuisance, duplicate records can cause a lot of headaches in an environment where gift material can form as large a part of a collection as can purchased material, standardization of data items causes retrieval problems, determining costs can present a tedious manual task, keeping track of bibliographic levels is often useful. It was, therefore, decided to implement seven major application processors to handle the user needs.

For data entry, we have:

(1) the INPUT processor, which permits the user to enter new records into the system. Depending on the structure of the data to be input, an internal sequence number (ISN) may or may not be required for that record. If one is required then it will be generated automatically by the data-base management routines. The processor is then told whether bibliographic levels are necessary for this user view; if they are, prompting will be done accordingly. Through the data definition facility this processor can also be instructed to automatically check for duplicate records and to validate the contents of data fields against standard authorities. If the new record is a duplicate, the user will be informed and will be given the option of continuing to enter that record or starting on another; if a data field is not validated, the user again will be informed and will be able to reenter the field (having optionally scanned the authority at this point) having found a valid replacement, or to reenter the field having first been passed to a son process where the user was permitted to generate an authority entry that would validate the field contents just entered.

(2) the MODIFY processor, which permits the user to make changes to records within the system. The modifications operate on fields and can be any one of: CHANGE — to change data within a field, ADD — to add a new field, DELETE — to delete a field, TRANSFER — to transfer one field to another, REPLACE — to replace an old field with a new one. If the record on which the user wishes to operate is not accessible by ISN, or if the user does not have the ISN, then the user may do a search within the processor to find the record in question. If more than one record is found to fulfill the query specifications, then the user may select the one that is actually required. This processor also works at a global level; when global processing is specified the processor will stream a copy of itself to run in a lower priority queue, thus leaving the user free to run the terminal for whatever else is desired, including the making of changes to individual records, within MODIFY.

(3) the RELEASE / DELETE processor, which permits the user with appropriate security clearance within the system to release records and to delete records. The deletion of records is self-explanatory; at the time that this book was written, deletions were logical rather than physical. The purpose of releasing is not so obvious, hence an explanation is in order. A record may be released in either of two ways. The first release flags the record as being one to which changes can no longer be made. This feature is particularly useful in a situation in which the person doing the terminal work is not the person filling out worksheets or creating the record; it signifies that the record is clean, and it prevents modifications from being made in error. The second type of release will undo the first release, i.e. it will free the record for further modifications.

For data retrieval, we have:

(1) the QUERY processor, which enables the user to search data bases interactively or in "batch." This processor has facilities for both Boolean retrieval and free-text retrieval. Retrieval may be done on the full data base or on a portion of it, depending again on the user's view of the data base. An interesting feature of QUERY is that it will provide for multilingual thesaurus-aided retrieval. This feature is based on the existence of a true, stable thesaurus. The IDRC has made use of the macrothesaurus published by the Organization for Economic Co-operation and Development (OECD) in Paris in 1972. This thesaurus, whose development was funded in part by IDRC, exists in many different languages (among them, English, French, Spanish, German, Portuguese), and relates descriptors to one another through the concepts of broader term, narrower term, related term, any term, use term, use for term, and facet number. At IDRC, descriptive abstracts for documents are written by using descriptors, selected from the macrothesaurus, as embedded keywords. The macrothesaurus exists in machine-readable form at IDRC in three languages: French, English, and Spanish. QUERY permits one to search in any one of the three languages as a base language, and will translate the descriptor (automatically OR'ing the translations to the original descriptor entered) into the other languages. For our users at the Centre this proves particularly useful because many of the documents in our collection are in languages other than English. Furthermore QUERY will enable one to search using the structural relations in the thesaurus (for example, BT development aid, will result in a search being done on development aid and its broader terms, in all available languages — if the translation facility has not been disabled by the user) to any level desired, and to display the structural

relations in the base language. Other features include both browsing (sorted or unsorted), and root searching capabilities.

(2) the INDEX processor, which enables a user to generate outputs in any wild and fantastic sequences that may be desired. INDEX works on the output of a query or on a user view to produce, essentially, sort keys that will be used to produce ordered outputs from the inputs. This processor is one of the most powerful in the system and, like MODIFY and QUERY, can stream itself to a lower priority queue. It will accept specifications interactively or from a redefined file of specifications, and will also initiate the print job which prints it.

(3) the COMPUTE processor, which enables a user to perform arithmetic operations on data, and to produce reports based on the results of arithmetic operations.

The PRINT processor does all the printing for the system. PRINT will print the outputs of INDEX and QUERY, and will print any user view defined for the data base. PRINT is very flexible: printing can be done on special forms or plain paper; it can be tabular or columnar; it can be with or without diacritical characters, page numbers, and the like; it can be output to a terminal (hard copy or CRT), a line printer, or other device. The printing specifications can be predefined and saved, or defined at the time of the run, through the use of a "chatty" dialogue within PRINT itself. This dialogue obtains from the user all the information required to run a print job, including the field formatting specifications. This same facility allows a user to modify an existing specification. Like the other processors, it is streamable.

With respect to the end-user, data may be stored in both upper and lower case, and encoding for diacriticals may be embedded with the data (the encoding is recognized by the relevant processors). The working character set of the system is 7-bit ASCII. An arbitrary working maximum record length (data portion) of 2048 characters has been set; it appears to be more than sufficient for current needs. Simultaneous access to the data bases is a feature of our data-base system because of the file locking-unlocking capabilities provided by the file system of MPE. The lock is granular — it is at the relation (file) level as opposed to the record level. (Studies of current literature, and our own experiences, show that this granularity is not detrimental to performance.)

## System Manager

At this level a number of processors for maintaining and extending data bases have been provided:

(1) the ISOCONV processor, which produces and accepts tapes in ISO 2709 format. This is the processor that is used to load data bases received from other organizations. It was made as general as possible, with facilities for special processing exits, so that we could accept data from as many organizations as possible, and could use it for conversions;

(2) the data definition processor, which is used to create, modify, and delete all forms of data definitions (user views, system views, and intermediate views);

(3) the RENUMBER processor, which permits the replacement of ISN's

with other ISN's. This processor is particularly useful in the production of printed bibliographies;

(4) the INVERT processor, which does batch inversions on domains to provide new rapid access paths to a data base;

(5) the "garbage collector," which is periodically run to recover unused and free space within the data bases, and to generate backups of the physical files of which the data base is comprised;

(6) various initialization processors.

Also available to the system manager are all the facilities provided by the manufacturer. In the case of the HP-3000 we are fortunate to have many.

## Programmer

The applications programmer is the person who writes the code to implement the end-user and the system-manager processors. Available to this individual are a set of calls to the data-base management system. The functions, which were made an intrinsic part of the data-base management routines, make it very easy for the programmer to do things such as permit queries within the MODIFY processor. The functions available to the programmer for the writing of applications processors are procedures which will:

(1) perform syntax analyses of dialogues;

(2) actualize virtual domains according to the rules of domain algebra. For the application programmer, this call is made for arithmetic processing;

(3) perform restrictions on the data base, and can be used wherever restrictions are desirable. For example, QUERY is a sophisticated interface between the user and the restriction module; INPUT uses it in checking for duplicates; MODIFY uses it to process user queries;

(4) project new relations from existing relations;

(5) join existing relations to produce new relations;

(6) write user records to the data base;

(7) read records from a data base;

(8) validate the authenticity of data elements (this is used during INPUT, for example);

(9) provide the programmer with the capability of reading, writing, deleting, and updating fields within user records;

(10) make data bases available for processing, and give to the application the appropriate user view with which to work (i.e. open the data base);

(11) close a data base;

(12) format records into displayable forms. This procedure is a major

component of PRINT. It is also used by QUERY and MODIFY;

(13) generate an ISN for a new record entering the system if an ISN is required;

(14) extract keys from data elements. Keys may be defined as words, phrases, thesaurus terms, etc., and can be generated in many different forms.

Again, available to the programmer are functions for which provision was made by the manufacturer, for example, SORT / MERGE, and TRANSLATE from EBCDIC and ASCII.

## Data-Base Management System

At this level are all those procedures that are our implementation of the relational theory. The procedures include the fourteen listed above, in addition to three others whose functions are: (1) to process generated keys; (2) to do the actual work involved in releasing and deleting records; (3) to massage data definitions to produce the user view for the application program and the system view for the data-base management system (this is called by (10) above).

Although many of the procedures at this level are activated by applications programs, they are also activated by each other to perform many different functions. One reason for this is that at the data-base management level, all the routines work with relations and domains, be they virtual or actual. To produce a user record to pass back to the caller, GETUPLE (the procedure invoked by the application to read a record) may have to invoke JOIN in order to put together tuples from the constituent relations. Symmetrically, using PROJECT, AUGMENT may have to split the user records into $n$ $n$-tuples belonging to different relations. At times, GETUPLE may have to use PROJECT to provide a user record.

As explained earlier, it is possible to define user views that are comprised of virtual domains. At the time the user tries to access this view, the procedure that opens the relations recognizes that this is a virtual relation, and activates a procedure that will actualize this relation for the user. All other processing of this relation is identical to that for other relations.

A comment should be made on access. Relational theory does not provide for nonrelational access paths to a data base, but we have done so. Fast access paths are provided by means of inverted files. Inversion is done on keys extracted from data fields that have been defined for this type of processing by the system manager. The inverted file for a key that is part of an existing data base may be updated at three different times: when the record is first written to the data base (in this case AUGMENT looks after the calling of the key routines to do the key update); when the record is rewritten if the data field containing the key has been modified in any way (in this case the field manipulation procedure looks after it); or when a record is released or deleted (here the data-base procedure that actually releases or deletes records does the work of calling the key routines). The three different times are actually two cases: on-line or immediate update and update at release time only. If a record is deleted then all references to it in all inverted files are immediately deleted. A storage technique known as B-trees (Knuth 1973) was used to implement the

inverted files; compacted bit maps were used to store postings and to implement the Boolean logic.

## Comments on Implementation

The system as implemented did not require modifications to the manufacturer-provided software. In fact, some of the facilities that have been implemented were possible only because of the manufacturer's software and hardware. The HP-3000 is a machine that boasts a stack architecture. This has meant that all our code is reentrant and recursive. The system software is device-independent so our software is device-independent; processes can be easily spawned through simple calls to the operating system; user processes can spawn, as processes, even manufacturer processors; all terminal handling is done by the manufacturer's software (unlike ISIS where terminal I/O handlers had to be written using the PIOCS — physical input-output control system — provided by IBM); processors that we have written can run in any queue, thus they can be streamed, they can stream themselves, and they can stream one another — this is a very important feature because processors that are normally used in interactive mode can be run in "batch" mode if the type of work to be done warrants it; access security is provided by the operating system. It is important to note that the facilities of which we made use are basic facilities that are inherent in the operating system software.

All code for the system (application routines and data-base management routines) was written in SPL, the systems programing language of the HP-3000. This language is high-level ALGOL-like language, which is also the assembler language on the system. Thus it provides one with high-level structured statements such as DO...UNTIL and WHILE...DO, yet it also has instructions for bit manipulations. It is doubtful whether our programing efforts would have been as productive if we had had to write in some other language.

Out team consisted of two persons for 12 months, and three persons for an additional 7 months. During the first 12 months the design of the system was completed, and 13 of the 17 data-base management routines were completed. During the next 7 months, we wrote three of the four remaining data-base routines and the user processors. System development began in June 1976. In October 1977 our first users were phased in (the data entry processors and the print processors were complete) and at the time of writing, we are in the final stages of phasing in our other users. The INDEX processor is complete and the QUERY processor is in the shake-down stages. Needless to say, all the system-manager processors are operational as well. Our progress is proof that a small, devoted development team is as effective, or even more effective, than a larger group.

The data base of the Centre contains some 26 000 references. We also hold the data base of the ILO, which runs to over 60 000 references, the data base of the FAO, which is in the order of 40 000 references, and two smaller data bases, which together are in the order of 10 000 references. Until we can afford to acquire more mass storage, we handle the problem of on-line access through scheduling — the non-IDRC data bases are available at certain fixed times during a week or on advance user request.

Our basic system is operational, but our work is by no means complete. Soon, the procedure that implements the domain algebra must be tested and

implemented. By July 1978, we expect to have implemented an SDI (selective dissemination of information) processor, and to have completed documentation of the system. We also hope to write a procedure that will use a magnetic tape unit as a virtual mass-storage unit for the processing of large data bases, and to implement a photocomposition interface in PRINT.

It is hoped that this system will prove to be attractive to institutions that require a bibliographic data-base system, but cannot afford expensive equipment and have no means of sharing a larger computer facility. It is a logical outgrowth of our experience with ISIS and will, hopefully, help to enlarge the common network that began with ISIS.

## References


CODASYL 1971. CODASYL data base task group report. New York, ACM, April 1971.

Codd, E. F. 1970. A relational model of data for large shared data banks. CACM, 13(6), 377-397.

1971. A data base sublanguage founded on the relational calculus. In Proceedings of 1971 ACM-SIGFIDET Workshop. New York, ACM

1972. Relational completeness of data base sublanguages. In Rustin, R., ed., Database systems, Englewood Cliffs, N.J., Prentice Hall.

Date, C. J. 1975. An introduction to database systems. Reading, MA, Addison-Wesley, Publ. Co. Inc.

Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L. 1976. The notions of consistency and predicate locks in a database system. CACM, 19(11), 624-633.

International Organization for Standardization. 1973. ISO-2709 Documentation — Format for bibliographic information interchange on magnetic tape.

Knuth, D. E. 1973. The art of computer programming, volume 3: sorting and searching. Englewood Cliffs, N.J., Addison-Wesley, Publ. Co. Inc.

Merrett, T. H. 1976. MRDS — an algebraic relational database system. In Proceedings of the Canadian Computer Conference, Montreal, May 1976, 102-124.

Organization for Economic Co-operation and Development. 1972. Macrothesaurus: a basic list of economic and social development terms. OECD, Paris.

Ries, D. R., and Stonebraker, M. 1977. Effects of locking granularity in a database management system. ACM Transactions on Database Systems, 2(3), 233-246.