

C

UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CHAMPAIGN
ENGINEERING

NOTICE: Return or renew all Library Materials! The Minimum Fee for each Lost Book is \$50.00.

JUL 06 1988

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University. To renew call Telephone Center, 333-8300

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

--	--	--

10. 07
0632
0. 257
of 1,

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

CONFERENCE ROOM

Center for Advanced Computation



UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 257

THE DATA INTERCHANGE FILE:
PROGRESS TOWARD
DESIGN AND IMPLEMENTATION

by

Richard C. Roistacher

February 1978

The Library of the
MAY 25 1978
University of Illinois
at Urbana-Champaign

CAC Document No. 257

The Data Interchange File:
Progress Toward Design and Implementation


by

Richard C. Roistacher

Center for Advanced Computation
University of Illinois
Urbana, IL 61801

Table of Contents

Abstract.....	1
INTRODUCTION.....	1
Card image files.....	2
Self-described files.....	3
SUPPORT FOR A DATA INTERCHANGE FILE.....	4
The CONDUIT conference.....	5
The LEAA Research Support Center.....	5
DESIGN CONSIDERATIONS.....	5
Character format.....	6
Separate dictionary and data files.....	6
Card image dictionaries.....	7
Free format dictionary records.....	7
Machine readable file documentation.....	8
Variable length data records.....	8
Structure definition.....	9
Standards of Good Practice.....	9
AN IMPLEMENTATION OF THE INTERCHANGE FILE.....	10
The Dictionary.....	10
Documentation records.....	10
Variable description record.....	11
Missing data record.....	13
Category label record.....	14
Structure Description techniques.....	14
Rectangular files.....	15
Hierarchical Files.....	15
Network Data Bases.....	18
Relational Data Bases.....	19
Matrices and Tables.....	19
Structure and Record Definition Records.....	20
Record definition record.	20
Structure definition record.	21
Entry definitions.	22
Data Standards and Conventions.....	23
Interchange File Creation and Conversion	23
Manual Creation	23
File Conversion Programs	23
Machine Readable Documentation	24
Codebooks.....	24
Information retrieval keywords.....	24
GLOSSARY.....	25
REFERENCES.....	26



Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

<http://archive.org/details/datainterchange257rois>

The Data Interchange File:
Progress Toward Design and Implementation[1]

Richard C. Roistacher
Center for Advanced Computation
University of Illinois
Urbana, IL 61801

Abstract

Most self described data files are designed for maximum efficiency in processing with a particular data management or analysis system. This paper outlines a design for a Data Interchange File for the transfer and archiving of machine readable data. The primary design criteria of the Interchange file are generality, simplicity, and extendibility. The file will accept rectangular and hierarchical files, matrices and tables of arbitrary dimensionality, as well as data from network and relational data bases. The file is self described and will accept machine readable documentation as archival data. Strategies are also outlined for the technical and organizational implementation of the Interchange file.

INTRODUCTION

An increasing amount of scientific research activity involves the dissemination and secondary analysis of machine readable data files. Some of these data files are produced by individual research projects, some, like the Uniform Crime Reports are produced by organizations in the course of their operations; some, like the National Crime Panel Victimization Data, are produced as part of a special research project; while others, such as the National Election Studies, are produced by ongoing data collection efforts funded by a consortium of data users.

A data collector who is also the data's end user has many options for file construction and documentation. In the limiting case, a producer can maintain data in a completely undocumented deck of punched cards, relying solely on memory or a FORTRAN format statement for

[1] This work was originally supported by Grant 75-NI-99-0077 from the National Institute of Law Enforcement and Criminal Justice, and is currently supported by Grant 77-SS-99-6021 from the National Criminal Justice Information and Statistics Service, Law Enforcement Assistance Administration.

information about the file. Standardized documentation is not always crucial where data are transferred through personal contact between producers and users, although it is not uncommon to discover a colleague who would be happy to share a file, but who has forgotten its format.

An increasing number of files, however, are transferred not by personal contact between producers, but through dissemination by a central archive. National archives such as the Inter-University Consortium for Political and Social Research and the Roper Public Opinion Center receive data from their original collectors, transform files to a standard form, write appropriate descriptions of files, and fill user orders for data and documentation. Even though it is sometimes possible to refer a client's question to the original producer, no archive can afford to omit information transmitted by the producer from its own file; nor can an archive afford to produce documentation which is anything less than a complete summary of the original producer's documentation.

Card image files. The archivist's problem has in some respects been simplified, and in other respects complicated by the development of integrated statistical systems, self-described files, and machine readable documentation. The universal coin of machine readable data exchange is the deck of punched cards or the unlabeled, unblocked card image tape. The most common representation of data in such files is as numeric characters, with missing data indicated either by blanks or by some arbitrary code, such as a field of nines. In some cases, alphabetic characters are used to indicate valid data values, with "-"s and "&"s used to indicate missing data. A few punched card files contain data coded using tabulating machine methods, in which a data item is always represented in a single column. Arbitrary combinations of multiple punches are used when the standard character set has been exhausted.

Several archives, e.g., the Roper Public Opinion Center and the California State Data Program, maintain their data in card image files after converting alphabetic and multiple punched data items to numeric character form. Almost without exception, data archives will continue to produce card image files for export, even where their internal files are maintained in other formats.

Documentation for card image files is most often in the form of printed entries, giving the name, deck and column numbers, missing data values, and where appropriate, category labels for each variable. Some archives have produced machine readable documentation by punching their codebooks onto cards, which can then be stored and

reproduced with the data files to which they refer. Such documentation is both easier to reproduce and more difficult to lose than is paper documentation.

Self-described files. Originally, data files were analyzed with individually written programs designed to no particular standard. Beginning with the Biomedical Data Analysis Program library, (Dixon, et al., 1967), libraries of computer programs with similar control languages and input formats were written at many universities and research centers. In most of these program libraries, and indeed, for many currently used programs, the input data are described with a FORTRAN format statement which is included by the user with the program setup. Such programs and libraries expose the user to the inconvenience and possible error inherent in transcribing codebook information each time a program is used.

Most modern statistical and data management systems use self-described files, which contain program readable documentation. The user of such a system refers to variables by name or number rather than by location in the input record. The analysis program retrieves codebook information from the program readable file description stored with the data. Such systems locate data, provide appropriate handling of missing values, and label both printed and machine readable output with much less user intervention than would be required if a self-described file were not employed.

Even though self-described files make life easier for the user of a particular data management system, they complicate matters for the data archivist, or for the person who wishes to transmit data to someone who uses a different data management system. Most self-described files have been designed to maximize processing efficiency in their "home" systems. In many cases, data are stored in a non-printing internal form, with a high degree of machine and program dependence. Missing data are sometimes represented in program dependent forms which do not fit into the computer's standard set of numeric or character representations. Such files can be called "esoteric," not because they are necessarily incomprehensible, but because they are designed to be read from within a particular system rather than being generally readable. Files which can be interpreted by a simple character dump and which can be read using a FORTRAN-type format statement will be called "exoteric."

A common way of transferring esoteric files is to process them with programs which transform the data into card images and the dictionary into a printed codebook. However, the production of such card-image transfer files

undoes much of the work and nullifies much of the value of building the self-described file in the first place. The recipient of a card image transfer file is either reduced to writing FORTRAN format statements, building a new self-described file from the printed documentation, or using a program which attempts to reconstruct a new self-described file from the printed output. The SPSS WRITE FILEINFO subprogram is an attempt to make the process of degrading and transferring an esoteric file as painless as possible. The SPSS procedure, however, is designed to facilitate the transfer of esoteric SPSS files between SPSS installations on different computers, rather than to make the full self-described file available to other data analysis systems.

Other data analysis systems using esoteric files are SAS, the Statistical Analysis System developed at North Carolina State (Barr and Goodnight, et al., 1975), and IMPRESS (Meyers, et al., 1969). A somewhat less esoteric file structure is used by OSIRIS, (University of Michigan, 1976), which stores an esoteric dictionary separately from its data, which are stored as an exoteric file of fixed or variable length character records.

SUPPORT FOR A DATA INTERCHANGE FILE

An increasing number of data management and analysis systems generate and use self-described data files. The development of new systems should be encouraged, for it fosters a healthy diversity and spirit of innovation. Several considerations render impractical any attempt to standardize a common self-described file for all data analysis systems. Several statistical systems which use esoteric files have been in use for many years, and have been used to produce thousands of self described files. It would be impractical to require the users of such systems to learn and adopt a new file format solely for the sake of standardization with other systems. In addition, the use of a standard file for internal processing might require extensive rewriting of existing systems, with the risk of degrading of their internal processing efficiency. Finally, it would be foolish to attempt to restrict the designers of future statistical systems to the limitations of today's data processing techniques.

A better solution is to design an exoteric file capable of supporting most of the features found in all statistical systems, and designed specifically for the exchange rather than for the processing of machine readable data. Such a file should be designed for simplicity, generality, and extendibility, rather than for data processing efficiency. Designers of statistical systems can accommodate such an

Interchange file by writing procedures which convert their own esoteric files to and from data Interchange files. Thus, a data analysis system's own files can be designed to maximize processing efficiency within the system and, can be transformed to Interchange format for transfer and archival purposes.

The CONDUIT conference. In 1974, CONDUIT, the educational computing consortium, held a conference for the purpose of designing a data Interchange file. The conference laid the technical and political ground work for such a file, but lacked the funding for its further development and implementation.

The LEAA Research Support Center. In 1975 the University of Illinois' Center for Advanced Computation, under a grant from the National Institute of Law Enforcement and Criminal Justice, funded a research and development on techniques for archiving and using machine readable social data. One of the tasks of the new Research Support Center was to define an archiving format for data of interest to criminal justice researchers. The best way to accomplish this task was to continue work on the technical and institutional development of a standard data Interchange file. Accordingly, in January 1976, a conference on the exchange of machine readable data was held at Itasca, Illinois. The conference was attended by representatives of the Center for Advanced Computation; the National Institute for Law Enforcement and Criminal Justice, LEAA; the National Criminal Justice and Statistical Service, LEAA; SPSS, Inc.; the Inter-University Consortium for Political and Social Research, developers of the OSIRIS III data analysis system; the Survey Research Center, University of California, Berkeley; The Institute of Statistics at North Carolina State University, developers of SAS; the National Archives; the Bureau of the Census; and DUALabs.

This is the second in a series of reports resulting from the Itasca Data Interchange Conference. The Interchange file is intended to be used as a standard format for data archives and for the exchange of data between users, regardless of the computing hardware available to them or the data analysis systems they wish to use.

DESIGN CONSIDERATIONS

Several major considerations govern the design and implementation of the data Interchange file. The first consideration is that the file is designed to maximize its utility for data archiving and transmittal rather than for data processing. The file is designed to support arbitrary

data structures and missing data representations. It is designed to support more extensive variable and category labeling than is found in most data analysis systems. However, the data Interchange file is not designed to support all features of all existing statistical systems; in particular its features are not necessarily the union of all features to be found in the systems produced by the participating organizations.

The interests of simplicity dictate that the Interchange File support only the minimal number of program readable features. However, it can carry arbitrary textual information in its documentation records. Documentation records can be marked to indicate that they contain information which is not part of the interchange standard, but which is readable to a particular data management system. For example, the documentation records of an Interchange file produced with IMPRESS may carry information on the "standard dichotomy" of each variable. Such information may be read as text by users of systems which do not use such a standard recode, but may be read back into a receiving IMPRESS system. The Interchange standard includes a method for marking documentation records to indicate the presence of program readable information. However, the format of such information is the concern of those responsible for the design of the particular data management system.

Several characteristics are basic to the design and implementation of the Interchange data file.

Character format. Interchange files will be transmitted entirely in character form. It remains to be decided whether both ASCII and EBCDIC will both be allowed as character formats. If there is to be a single character set then obviously it will have to be the American standard ASCII. However, if the capability of almost all machines to understand IBM is granted, then either character set can be allowed. Both the dictionary and the data file will be composed entirely of printing ASCII (or EBCDIC) characters.

Separate dictionary and data files. Each Interchange data set will be transmitted as two separate files, a dictionary and a data file. Thus, it will be possible to separate the dictionary from the data without the use of special programming facilities, and to read and operate on the data either with or without the mediation of the dictionary. One of the major reasons why most self-described files are esoteric is that dictionary and data are written into a single file. Special programming, intrinsic to a particular data management or analysis system, is required to read and interpret the dictionary, and to

determine where the dictionary ends and the data begins. Only by storing dictionary and data in two separate files can the need for special programming be eliminated.

Card image dictionaries. Most users of machine readable data have facilities for creating new data variables, facilities which may range from a ten line FORTRAN program to an extensive recode language. However, it cannot be assumed that all users of machine readable data will have access to similarly powerful facilities for modifying and editing dictionaries. Therefore, it seems wisest to maintain the Interchange dictionary in the form of eighty-column card image character records with five-digit sequence numbers in columns 76-80. Although it is hoped that more elegant facilities will be available, in the last resort a user should be able to produce and edit Interchange dictionaries with tools no more complicated than a set of card listing and punching routines and a key punch.

Free format dictionary records. All dictionary records will have a type identifier in column 1, a variable number in columns 2-6, a file identification in columns 73-75, and a sequence number in columns 76-80. Wherever possible, columns 7-72 will be used to record dictionary information in free format. The use of a free format for recording missing data information, variable, and category labels allows both greater ease of file creation, and greater flexibility in adapting to the characteristics of new statistical systems as they appear on the scene. Free format is obviously easier for someone who must create a dictionary by hand, and is irrelevant to a dictionary creating program. Free format dictionary information is probably easier reading for the person who must interpret the dictionary manually.

It is sometimes argued that the reading and processing of free format information requires unduly sophisticated software and inordinate extra expense. Such criticisms are simply no longer valid. Any reasonable computer system has available the software to do simple parsing of text. The additional expense entailed by the one-time use of parsing programs in converting an Interchange file is negligible in comparison to other expenses incurred in obtaining and processing the file. The length of a dictionary is determined by the width of its data file i.e., the number of variables and their range of codes. Extremely large and expensive files usually contain large numbers of cases as well as large numbers of variables. Thus, dictionary processing expenses for files with large numbers of cases are relatively small in relation to data processing expenses; dictionary processing expenses for one-time conversion from Interchange to some other self-described

file format will be negligible in comparison with other expenses incurred in acquiring and processing the data.

Machine readable file documentation. A direct consequence of recording information in free format wherever possible is that extensive file documentation can be produced and transmitted in machine readable form. Each Interchange dictionary will contain a set of documentation records giving technical information on the file. Such information should include the file's name and creation date, the file's logical structure, the system on which the file was originally created, the character set and collating order used in the file, global treatment of blanks, and other technical processing information. The description should also include an abstract of the file and the study which produced it, as well as any notes the producer wishes to pass on to future users.

There are two main reasons why such documentation need not be program readable. First, it is impossible to tell what information producers of Interchange files will want to include in their file description records. Second, most of the information contained in the header records will probably require human intervention in any case. Thus, the interests of flexibility dictate that basic information on such things as a file's name and creation date be acted on by the user, rather than interpreted by a computing system.

Since the contents of documentation records will be transparent to the programs which read and write Interchange dictionaries, there is no reason why they cannot be used to document individual variables and parts of the dictionary. The inclusion of a variable number field on documentation records, in addition to the sequence number, allows unique placement in the dictionary file. Since the documentation text will be in whatever format the producer wishes, all that can be guaranteed is that conversion programs will print all documentation records. However, this does not preclude users from including program readable information beyond that required by the Interchange format.

Variable length data records. Interchange data records will be of variable length in order to support files with more than one type of record. A number of questions regarding data record formats remain to be answered. Although the canonical structure of Interchange data sets is hierarchical, many, if not most, Interchange data sets will be rectangular. Should rectangular data sets be allowed to use fixed length records, or should all Interchange files, regardless of their structure, use variable length records?

Another unresolved question concerns the labeling of

Interchange data sets stored on tape. If an IBM tape file is assumed, then it would be expected that IBM labeled, format VB files would be acceptable. If, however, the ANSI tape format is to be used, should the canonical Interchange data format be an ANSI labeled, fixed-length record, blocked file for the dictionary; and an ANSI labeled, variable-length record, blocked file for the data? The problem of users' computers not being able to handle such labeling or deblocking can be solved by writing the labeling and deblocking routines into the programs which import and export Interchange files.

Structure definition. The Interchange dictionary will carry not only a description of each separate type of data record, but also a program readable description of the file's logical structure. A user should be able to obtain a rectangular file based on the lowest level of analysis; e.g., a file in which data from the record on a household has been duplicated and appended to the record of each person in the household.

Standards of Good Practice

Since the Interchange file is designed to maximize flexibility, it will support many options which are not presently in use, and some options which probably should never be used. The final specification of the Interchange data file should implicitly support some rules of good practice, rules whose violation can be supported by the Interchange data structure, but which should be discouraged. While a full list of such rules is probably infinitely long, some rules come immediately to mind.

Although the Interchange data set will store alphabetic data, variables should be stored in numeric form wherever possible. In particular, missing data information should be numeric rather than alphabetic. In some cases it may be necessary to have a variable whose value represents some attribute of a particular value of another variable. For example, variable one may have the value "1" in observations in which the value of variable two is an estimate, while variable one has the value "0" in observations where the true value of variable two has been obtained. Such cases are rare and should be made as rare as possible, since they invariably require some recoding before the file is usable. In most cases identical information can be transmitted using ordinary missing data conventions.

Creators of Interchange data sets should also be sparing in the number of exact-match missing data codes used in the file. Most users receiving a file having, for

instance, eight exact-match missing data codes per variable will be forced to spend considerable time collapsing such codes into a more manageable number. File producers should also be discouraged from using mid-range missing data codes. A variable ranging from 0 to 9 should be given a missing data code outside this range, even if a value within the range is unused.

AN IMPLEMENTATION OF THE INTERCHANGE FILE

The Dictionary

All dictionary records are 80 character records containing a type identifier in column 1, a variable number in columns 2-6, a file identifier in columns 73-75, and a sequence number in columns 76-80. The format of columns 8-72 is different for each type of dictionary record.

The sort order for dictionary records should be by variable number and then by sequence number. Such a sort order will allow intervals to be left in the original sequence numbers for the insertion of new records.

Documentation records. Documentation records contain free format text giving information about the file as a whole, about sections of the file, and about individual variables and responses. Documentation records dealing with the file as a whole should probably have a variable number of 00000. Since the contents of a documentation record is transparent to the Interchange format, the variable number of a documentation record can indicate the number of the variable to which the record applies, or can be used as an indicator of the record's position in the file.

It seems reasonable that some users would make certain documentation records readable by some receiving programs. For example, a data analysis system which produced value labels longer than twenty characters long might write shortened labels for the Interchange value label records, while inserting the original longer labels in documentation records. The original labels can be marked so that they are automatically recovered if the Interchange file were converted back to its original type. These documentation records would in no sense be a replacement for the Interchange value label records, but would serve as additional documentation for most users, and as a way of allowing some users automatically to recover the original labeling.

Column 7 of the documentation record is used to indicate the presence of text which is program readable. A

blank in column 7 indicates that the record carries no program readable information. A printing character in column indicates that the record carries information which is program readable to some data management or analysis system. The file's producer must indicate in the documentation which characters are used to mark data for which systems.

This strategy has been chosen in order to minimize the number of elements which require standardization. Designers of data systems may develop their own conventions for the transfer of program readable information, but these conventions are transparent to the Interchange standard. There is little to be gained by establishing any common list of data system identifiers to use in column 7 of the documentation record. It is easy for a user to look in the file documentation, while it is difficult to maintain and update a list of standard identifiers.

Variable description record. The variable description record stores a variable's number, name, location and width, label, level in the file, and whether the variable is to be considered a number or a character string. Since the recording of missing data values may require more space than will be available on the variable description record, all missing data information will be placed on a separate dictionary record.

The format for the variable description record is:

Column	Information
1	Record type: V
2-6	Variable number. The variable number will be the basic data identifier in the Interchange file.
7-9	Relation number. A relation is a set of variables referring to the same type of observation. A relation could also be called a record type. Examples of relations are information about a single household, or information on a single individual in one wave of a panel study. Rectangular files will have only one relation.
10-17	Variable name. This field is designed to carry variable identifiers generated by systems which refer to variables by alphabetic names. The field can also be used to hold an OSIRIS reference number, which also serves as a variable name independent of the ordering of variables

in the file.

18-19 Record number.

This field allows the support of data on cards or other unit records. Interchange dictionaries and data files stored on disk and tape will usually have only one record per observation. However, when a file is stored on cards, this field will indicate the the sequence order of the card carrying the variable. A blank in this field should probably be allowed to indicate that the observation is stored on one and only one record.

20-24 Location.

This field records the location of the leftmost character in the variable, counted from the left edge of the record. The count includes all linking information required to associate a record with records in other groups, but does not include the binary length field which is part of the format VB record. Thus, the location field will give an accurate account of the variable's position once the record has been deblocked.

25-28 Width.

This field records the width of the variable in characters.

29 Field type.

This field is a "0" for numeric data, a "1" for purely alphabetic data, and a "2" for variables which are numeric in some observations, and alphabetic in other observations. The latter case can occur in systems which generate alphabetic missing data codes for numeric variables.

30-31 Number of decimal places.

This field has two columns to accommodate very large numbers, and numbers with a negative number of decimal places. The latter case can occur where income is being stored in hundreds of dollars. Since all data are in character, rather than in binary form, E-format should be allowed for the representation of very large and very small numbers.

32 Spare place for later expansion.

33-72 Variable label.

The maximum length of a variable label is forty characters. An interpolated set of documentation records may be used to extend the variable labeling information, but only the forty characters on the

variable description record will be considered program readable for Interchange purposes.

73-75 File identification.

76-80 Sequence number.

Missing data record. The missing data record contains missing data specifications for the variable in columns 7-72. Since missing data specifications vary widely among systems, it seems best to allow the greatest possible flexibility in the specification of missing data. The most general way of specifying missing data would be as a Boolean expression describing which numbers and character strings will be used to represent missing values.

In this instance, the full Boolean format can be abbreviated. The "or" connective can be implied by a simple sequence of values. The statement, "If V is missing, then V equals 7 or V equals 8 or V equals 9," is well defined by "7 8 9". "If V is missing" is implied by the missing data record itself. The phrase "[or] V equals" can be used as the default meaning of a delimiter. If a fuller representation is desired, the missing data example above can be rendered as "7 OR 8 OR 9". If the "OR" default is used, then the necessary connectives are "AND", "(", ")", and "'". The relational operators, "LT", "LE", "EQ", "NE", "GE", and "GT" complete the set of primitives needed to form a missing data language. (It is not clear that "NE" has any real use in such a language.)

Examples of missing data codes in most systems are easy to express in this language. Some examples are:

```
SPSS      "77 88 99"  
OSIRIS    "99 GE 77"  
PICKLE    "LT 10 GT 90"  
SAS       ".A .B .C .D .E .F .G .H"
```

Things are relatively simple when missing data codes are entirely numerical. However, there is some question of whether ranges of character strings ought to be allowed in missing data expressions. To do so implies that there is a common collating order. If this is the case, then the ASCII collating order could be specified as the order underlying such expressions as "(GE .A AND LE .Z)", which would neatly express all of the SAS internal missing data codes. The use of collating order ranges for alphabetic missing data codes is attractive, but may violate the primary criterion of relentless simplicity which underlies the design of the Interchange file.

One way to express a set of universal missing data codes might be to specify all missing data codes for the file with a single set of missing data records with a variable number of 00000. Thus a SAS data set in Interchange format would have a single set of missing data records spelling out the 26 SAS missing data codes. This single set of records would apply to the entire file. The description of global blank treatment follows immediately from this procedure. Global treatment of blanks as missing data is indicated by a missing data record with a variable number of 00000 which carries a blank between primes. The file standard should probably state that variables with local missing data declarations are exempted from any global missing data declaration.

Category label record. The category label record contains a value for a categorical or discrete variable, a label of up to 20 characters, and an optional frequency count for the category. It seems simplest to have a single category on a card, with the first character string in the field interpreted as the code value, the second string interpreted as the label, and the third string interpreted as the frequency count. This set of conventions will allow the correct interpretation of,

2 FEMALE

and of

2 FEMALE 500

if blanks are allowed as string delimiters. However, additional delimiters are required for the proper interpretation of such label data as

3 FATHER'S HOUSE 405

It seems best to require that all labels containing blanks be enclosed in primes so that frequencies can be added without having to reformat the record.

Structure Description techniques

The Interchange dictionary describes each of the several types of records contained in the data file and the structural relation between record types. The structure definition facilities of the Interchange file will support rectangular files, hierarchical files, generalized network data bases of the CODASYL type, and relational data bases.

Rectangular files

Rectangular data files contain only a single type of record, and thus require no explicit structure definition. Where an interchange file is being used to store data from a DBMS or a set of files, a rectangular file may be stored as a single relation.

Hierarchical Files

While the Interchange file will store data bases of arbitrary structure and complexity, most complex data structures cannot be processed without being reformatted and restructured onto direct access devices. Such restructuring is the province of the receiving data base management system. rather than the Interchange file or its supporting software. All that the Interchange file can do is to store a description of the original structure along with the data records. The receiving data base management system must then create its own internal data structures from the stored description.

However, it is possible to store tree structured data in a form which may be processed directly from a sequential Interchange file. Since most users will be using sequential processing systems, it is advisable to provide an actual data structure for tree structured files within the Interchange file, as well as a structure description.

While data base management systems could reconstruct a tree structure from the structure description alone, most sequential processing systems would require a set of pointers on each data record. It thus makes sense to provide information which will allow the sorting of the file into meaningful order. The most robust linkage method is to give each record a complete set of upward pointers.

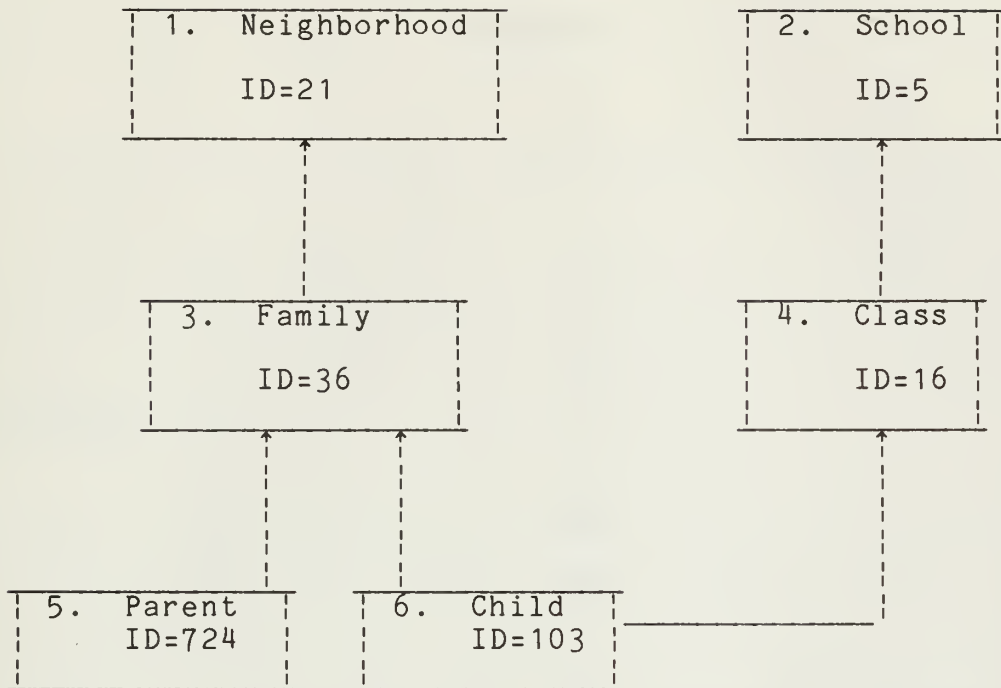


Figure 1: A hypothetical data structure.

For example, consider Figure 1, which represents a file having six types of records in three hierarchical levels. Every record in this Interchange file will carry six identification numbers, one for each type of record. Each record will carry the identification variable of all records under which it can be structured. Thus the record of a child will carry numbers identifying the records of its family, neighborhood, class, and school. The child's record will carry missing data in the field carrying a "parent" pointer, since children are not subordinated to parents.

It should be noted that the structure in Figure 1 is not a tree, but a lattice, a structure with more than one root. The file may be sorted into either of two trees. One tree consists of the "neighborhood", "family", "parent" and "child" structure. The other tree consists of the "school", "class", "child" structure. While either of these trees may be created by an appropriate sort of the file, only one may be processed in any one pass of the file.

		Pointer #						
		0	1	2	3	4	5	6
Record Type	1	21	-	--	--	---	---	
	2	--	5	--	--	---	---	
	3	21	-	36	--	---	---	
	4	--	5	--	16	---	---	
	5	21	-	36	--	724	---	
	6	21	5	36	16	---	103	

Figure 2: Pointer array for the data structure in Figure 1. Pointer 0 is the record type.

All possible linkages may be represented by such sets of upward pointers. The assignment of levels does not always imply that records at a lower level are disaggregations of records at a higher level. In some cases the specification of levels is simply to resolve which record is pointing at which.

Although the set of pointers in Figure 2 is implied by the linkages in Figure 1, it would be difficult to infer those linkages solely from analysis of the pointers, since to do so would require reading the entire file. The structure can be inferred from the information that parents and children never point to each other but both point to families, and that families point to neighborhoods, while neighborhoods do not point anywhere.

The structure definition, however, should be provided explicitly, so that both the user and the file importer know what to do with the data. Thus the Interchange dictionary should have an explicit structure definition record as well as a set of record definitions giving pointer information. The person creating an Interchange file must choose an identification variable for each type of record. People should, in general, be discouraged from creating records without identification variables. If no variable is

appropriate as an identifier, then the file exporter should supply an arbitrary sequence number for each record. The sequence number need not be unique within the file, but only within the level at which the record enters the structure.

For example, the records of children in Figure 1 may carry unique identification numbers, but if they do not, a simple sequence number within each family will suffice. Each of the N types of records in a tree structure will be prefixed by N+1 identification variables, consisting of a record type identifier and N pointers, one for each type of record in the file. Where two records are connected by more than one link, then more than one pointer will be required. Hopefully, people will be sparing in their use of multiple identification fields.

It should be noted that at least one pointer on each record is a simple duplication of one of the variables in the record. The duplication is justified in order that the syntax of the pointer variables be completely under the control of the file exporter, and thus absolutely canonical in the Interchange format. The user may use almost anything as a missing data indicator in the data portion of the record, but when that identifier is copied into the pointer section, it will be subject to conventions specified in the Interchange data standard. Such conventions should require that pointers have only numeric values, that there be no blanks in pointers, and that a particular type of missing data indicator be used. (See Tsichritizis and Lochovsky, 1976, for a fuller exposition of hierarchical data bases.)

Network Data Bases

A network data base is a generalization of the hierarchical file. In the hierarchical file, records are related to each other in only one way, by being subordinate or superordinate to each other. In the example above, children are subordinate to families, who are subordinate to neighborhoods.

Suppose that the file contained a set of records each of which held information on a hobby, and that each "child" record contained a variable called "favorite hobby". This variable is a link between the child and information about his or her favorite hobby. Thus, the "hobby" and the "child" records need share no sorting information, but their linkage must still be documented. The linkage is documented in the structure definition by naming it and by indicating the variables which link the two records. However, there is no way in which hobby records can be sorted into the hierarchy of neighborhood, family, and child. (See Taylor

and Frank, 1976, for a fuller exposition of network data bases.)

Relational Data Bases

Relational data bases are constructed from rectangular files which have unique identification variables. Relational data bases have advantages in being uniquely simple and general. Their present disadvantage is that there are at present relatively few implementations of such data base systems. Each data structure in a relational data base is a rectangular file with a unique identification variable. The relational data base is simple because it uses no system of pointers. These data structures or "relations" as they are called in data base management terminology, are linked by a process of sorting and merging on the basis of identification numbers.

As long as the proper conventions of uniqueness of identification variable are maintained, the interchange file will support relational data bases in the form of heterogeneous records with no explicit hierarchical relation. Each relation a relational data base is simply assigned a relation number in the interchange file. (For a fuller exposition of relational data bases, see Chamberlin, 1976, and Teitel, 1977.)

Matrices and Tables

The interchange file stores matrices and multidimensional tables in straightforward fashion. A matrix or two dimensional table may be stored as a rectangular file. A table of higher dimensionality may be stored in either "row" or "cell" fashion. An n-dimensional table stored in "row" fashion is treated as an n-1 level hierarchical file with a single record type. Each record in a "row" file is a vector from the table, labeled with its coordinates in the table.

A 3 x 4 x 10 table could be stored as a file of 12 records, each of which contained 10 variables. Each or the records would be treated as the third level of a hierarchical file, and would have pointers indicating the row and plane of the table to which it belongs. Such a file could easily be sorted by row and plane. However, a sort by column would have to be performed by reformatting the records.

A seemingly more clumsy, but preferable way of storing a table is in "cell" form, in which each cell of the table

is labeled with its coordinates. The coordinates are in canonical form as pointers rather than variables, and are thus provided by the exporting program, rather than by the user. Although a "cell" file might appear quite bulky in relation to the original table, it provides the advantages of being totally open in format and of being possible to manipulate without reformatting records.

Since even huge tables are relatively small in relation to huge files, "cell" files should not be particularly expensive or clumsy to handle. Since these files will be written and read by table producing and handling systems, the disaggregation and labeling of "cell" files will be invisible to the average user.

Structure and Record Definition Records

Record definition record. The record definition records constitute a dictionary for the rectangular subfile formed by the type identifier and vector of pointers. The format for the record description record is:

Column	Information
1	Record type: R.
2-6	Variable number. The variable number has no direct application to the record definition record but is used solely for sequencing in the file. Thus, any variable number less than the smallest variable number can be used. Users should probably be encouraged to number variables in a way which helps identify their record type, such as having variables in record type 5 begin with 501.
7-9	Relation number. Several options are available for the formatting of relation numbers. One option is that the data Interchange specification require that the first three columns of every data record be a record type identifier. The second alternative is that the location of the record type identifier be inferred as everything ahead of the first pointer field. Thus, if the pointer for the lowest record type begins in column three, columns one and two of the record are assumed to be the record type. A third, and perhaps the most suitable alternative, is that the definition of record type zero indicate the location of the record type indicator.

10-11 Level.

12-31 Name.

32-36 Pointer location.

37 Pointer width.

38-46 Pointer missing data value.

47-55 Pointer inappropriate value.

It has been suggested that separate missing data and inappropriate codes are not needed for pointers. A missing data code in a pointer to a higher level can be interpreted as actual missing data, while a missing data code in a pointer to the same level or to a lower level can be inferred as inappropriate. It has not yet been decided whether or not inappropriateness should be explicit or inferred.

56-60 Pointer variable number.

This field indicates the variable number in the record type which has been used as the pointer. A variable number of zero indicates that the file exporter produced an arbitrary sequence number for the record.

61-65 Number of variables in the record.

66-70 Aggregate record length.

These two fields would be helpful in allowing the importing program to allocate work space for reformatting the file. However, they may require two passes through the file to create the dictionary and might be omitted from the record definition record. Further discussion of whether or not to include them is necessary.

73-75 File identification.

76-80 Sequence number.

Structure definition record. The structure definition provides an explicit indication of the links between record types. The structure definition consists of a set of free format expressions indicating the equivalence between pointers in different relations. The format for the structure description record is:

Column	Information
1	Record type: S.
2-6	Variable number. The variable number has no direct application to the record definition record but is used solely for sequencing in the file. Thus, any variable number less than the smallest variable number can be used.
7-72	Structure definition expressions. Free format expressions showing the logical structure of the file and the variables linking different relations.
73-75	File identification
76-80	Sequence number.

A suggested syntax for structure description expressions is:

<name>:<rectype>(<var#>)=<rectype>(<var#>)

In the case of hierarchical records, it would be nice to require that the direction of the expression go from lower level to higher level in order that the hierarchy in the file be inferrable without reference to the level numbers contained in the record definition records. Hierarchical files do not need explicit names for pointer relationships. However, names are necessary to clarify the relations between the records of a generalized network data base. The structure of the file in Figure 1 could be indicated (using arbitrary variable numbers within record types) as:

5(2)=3(1) 6(2)=3(1) HOME_ROOM:6(3)=4(1)
3(2)=1(1) 4(2)=2(1)

This structure definition allows both the user and the importing program to recover the original structure of the file.

Entry definitions. Following the OSIRIS convention, an entry is defined as the rectangularized file actually read and analyzed by a program. The OSIRIS structured file carries with it a default entry definition which is used in the absence of any specification by the user. There is some question as to whether the Interchange file should carry a default entry definition with its dictionary. If the importing system uses a hierarchical file, then the importer could simply transform the Interchange file into an esoteric

hierarchical file. However, it can be expected that many importing systems will not support hierarchical files, and that the file must therefore be rectangularized. Perhaps the most reasonable course is to include a verbal summary of some entry definition and leave the actual construction of the entry to the user and the file importer.

Data Standards and Conventions

Perhaps the only restriction on the data is that they be in the form of printing ASCII or EBCDIC characters. While there are many rules of good practice regarding the choice of coding schemes and the layout of data, most of these rules have no effect on the syntax of the Interchange file.

Interchange File Creation and Conversion

Manual Creation

Proper design of the Interchange dictionary will allow many Interchange files to be constructed without the use of special programs. Rectangular files will require the addition of an observation identifier, something which should probably be there in any case. Once such a data file has been produced, a valid Interchange dictionary can be produced by hand.

File Conversion Programs

In order for the Interchange file to succeed, statistical systems must have facilities for converting their own esoteric files to and from Interchange format. File importers will probably need special care in their design, since they must be capable of correcting the file producers' deviations from good practice. Importers will probably require not only extensive recoding techniques for converting such things as missing data codes, but also reasonably powerful text editing techniques in systems which will not support the long labels of the Interchange dictionary. In the long run, it would be far better to increase the labeling capabilities of other systems to the SPSS standard, than to degrade one of that system's most pleasant and useful features. The design of an Interchange file importer for each statistical system is a problem whose difficulty should not be minimized. Hopefully, much of the work of civilizing files which violate rules of good practice will be done by data archives.

The task of designing a file exporter seems somewhat simpler than that of designing an importer. The Interchange dictionary can be written from the system's esoteric dictionary, and the pointer section of the data records written without much difficulty.

Machine Readable Documentation

Codebooks. At present, the development of machine readable code books considerably lags the present state of computer text processing. Most code books are simple transcriptions of paper code books to punched card for easy transmission with the data. The OSIRIS code book, the most highly developed of machine readable code books, is basically a primitive form of document processor manuscript. OSIRIS code books are laborious to prepare and difficult to edit. Few users employ the subsetting facilities of the OSIRIS system, while even fewer ever edit, expand, or create new OSIRIS code books.

The full data Interchange file should probably include a machine readable code book. Code book information can be carried on the documentation records in literal form, and these records can even be subsetted as the file is broken into subsets. However, transmission and storage of code book information in literal form loses most of the flexibility afforded by computer document processor systems. Code book information stored as a document processor manuscript can be easily edited, subsetted and modified. In addition the document processor will provide such features as automatic resolution of table and variable numbers and an automatic table of contents and cross reference.

Future work on the data Interchange file should include the selection of a document processing language. In the meantime, documentation on the Interchange file should probably be stored in literal form.

Information retrieval keywords. Program readable documentation can be supplemented with key information so that the interchange dictionary can be read directly into an information retrieval system.

Keywords could be appended to the file description and to each variable, using documentation records marked as containing program readable text. The dictionary could then be read into an information retrieval system such as SPIRES (the Stanford Public Information Retrieval System) which would reformat the documentation into internal structures referenced by the keywords.

GLOSSARY

This glossary is intended to clarify certain terms which are used in new or unusual ways in this paper. It is not meant to be in any sense a complete glossary of terms relating to the Interchange standard.

Cell file. A file each of whose records consists of a single cell from a table and the indices of the cell in the table. Several tables with the similar structures can be stored in a single cell file.

Data set. A file or set of files containing complete information on a set of self-described data. An SPSS data set consists of one file, while an OSIRIS data set can consist of two or three files.

Dictionary. A program readable set of information describing a machine readable data file.

Entry. The data vector created from a hierarchical file which is actually read and analyzed by a statistical program.

Esoteric file. A file which cannot be interpreted with simple printed dumps and read by simple FORTRAN style format statements. Esoteric files must be read by specially designed software. SAS and SPSS files are both esoteric.

Exoteric file. A file which can be interpreted with character format dumps and which requires only a simple format statement for interpretation. Card image files are exoteric.

Exporter. A program or subprogram built into a data analysis system which generates Interchange data sets from the system's native data set.

File. A set of machine readable data organized as a unit with respect to a computer system. A file need not be coterminous with a data set. For example, several SAS data sets can occupy a single IBM file, an SPSS data set is coterminous with an IBM file, while an OSIRIS data set requires two IBM files.

Importer. A program or subprogram built into a data management and analysis system for converting Interchange data sets into the system's native data format.

Interchange data set. A dictionary file and data file constructed according to the standards outlined in this

paper and agreed on by the working group.

Literal text. Text which is printed exactly as it is stored on the machine readable medium without reformatting.

Machine readable. Information stored on punched cards or magnetic media which can be interpreted by a computer. Machine readable data, e.g., literal text, is not necessarily in a form which can be interpreted by processing programs and should be distinguished from program readable data.

Pointer. The vector of identification variables prefixed to each Interchange format data record.

Program readable. Machine readable data in a form suitable for interpretation and processing by a computer program. For example a set of keywords punched on cards are both machine readable and program readable, while a comment statement is merely machine readable.

Row file. A file each of whose records consists of a row of a table and the indices of the row in the table. A row file of a two dimensional table is an ordinary rectangular file.

REFERENCES

Barr, J., and Goodnight, J. SAS progress report. Paper presented at the SAS Users' Group meeting, Orlando, FL, January, 1976.

Buhler, R. The P-STAT system. Pp. 283-286 in Proceedings of Computer Science and Statistics: 7th Annual Symposium on the Interface, Iowa State University, 1973.

Chamberlin, D. G. Relational data base management systems. Computing Surveys. (8),43-66.

Dixon, W. J. (Ed.) BMD Biomedical Computer Programs. Berkeley, CA: University of California, 1975.

Meyers, E. D. Jr. Project IMPRESS: Time-sharing in the social sciences. AFIPS - Conference Proceedings. (34), 673-680, 1969.

University of Michigan. OSIRIS III: Volume I, system and program description. Ann Arbor, MI: Author, 1976.

Nie, N. H., Hull, C. H., Jenkins, J. G., Steinbrenner, K., and Bent, D. SPSS: Statistical Package for the Social

Sciences. New York, McGraw-Hill, 1975.

Roistacher, R. C. The data interchange file: A first report. Center for Advanced Computation. (CAC Document No. 207), 1976.

Spires 1970-1971 Annual Report. SPIRES BALLOTS, Stanford University, Stanford, CA, December 1971.

Design of SPIRES II. SPIRES BALLOTS, Stanford University, Stanford, CA, July 1971.

Taylor, R. W. and Frank, R. L. CODASYL data base management systems. Computing Surveys. (8), 67-104, 1976.

Teitel, R. F. Data base concepts for social science computing. Proceedings of Computer Science and Statistics: 9th Annual Symposium on the Interface. Harvard University, 1976.

Tsichritizis, D. C., and Lochovsky, F. H. Hierarchical data-base management. Computing Surveys. (8), 105-124, 1976.



UNIVERSITY OF ILLINOIS-URBANA

510.84/L63C CD01
CAC DOCUMENTS/URBANA
256-258 1977-78



3 0112 007264101