DOCUMENT RESUME

ED 253 204 IR 011 480

AUTHOR	Gallagher, Leonard; Salazar, Sandra
TITLE	Report on Approaches to Database Translation. Final Report.
INSTITUTION	National Bureau of Standards (DOC), Washington, D.C. Inst. for Computer Sciences and Technology.
REPORT NO	NBS-SP-500-115
PUB DATE	May 84
NOTE	92p.
AVAILABLE FROM	Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.
PUB TYPE	Information Analyses (070) -- Reports - Evaluative/Feasibility (142)
EDRS PRICE	MF01/PC04 Plus Postage.
DESCRIPTORS	Computers; Cost Effectiveness; Costs; *Databases; *Data Processing; *Information Retrieval; *Information Storage; Information Systems; Models; *Standards
IDENTIFIERS	*Database Management Systems; Database Translation; *Data Conversion

ABSTRACT

This report describes approaches to database translation (i.e., transferring data and data definitions from a source, either a database management system (DBMS) or a batch file, to a target DBMS), and recommends a method for representing the data structures of newly-proposed network and relational data models in a form suitable for database interchange. In the opening chapters, the current state of research in the field of database translation is summarized, various interchange forms that have been proposed as potential standards are examined, and proposed standard data models and other commonly-used data structures are discussed. Additional chapters analyze the results of experiments with database translation between data models--hierarchical or relational to network, hierarchical or network to relational, hierarchical to network, and network to hierarchical--and summarize a cost benefit analysis of a database conversion. Conclusions address the future impact of standardization on database translation, and appendices provide specifications for mapping between data structures and completing the interchange form, together with detailed examples. (LMM)

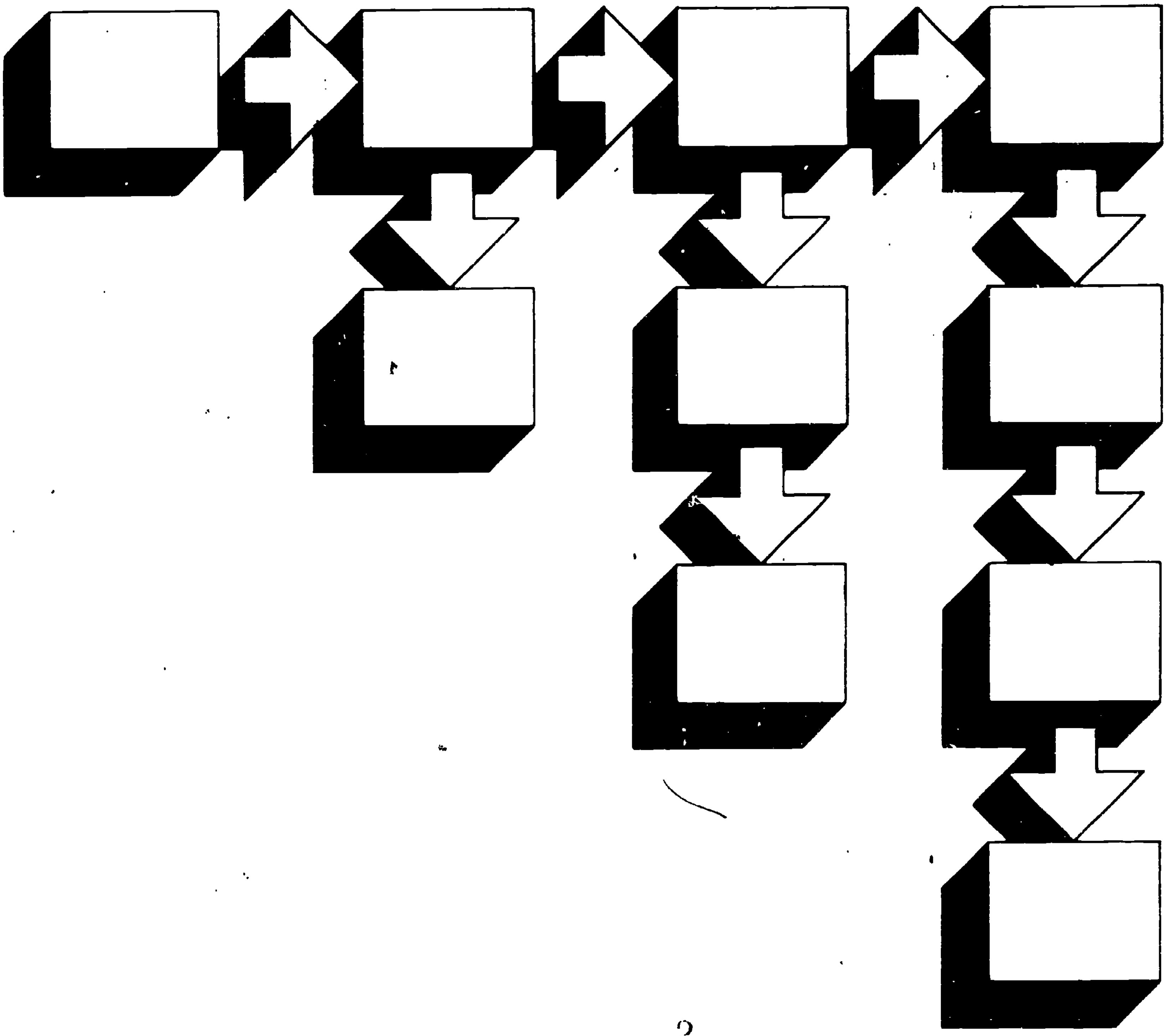
U.S. Department
of Commerce

National Bureau
of Standards

Computer Science and Technology

NBS Special Publication 500-115

Report on Apprcaches to Database Translation



NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] -- Radiation Research --- Chemical Physics -- Analytical Chemistry — Materials Science

THE NATIONAL ENGINEERING LABORATOR. provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] -- Manufacturing Engineering — Building Technology — Fire Research — Chemical Engineering[2]

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted.
mailing address Washington, DC 20234
Some divisions within the center are located at Boulder, CO 80303

Computer Science
and Technology

NBS Special Publication 500-115

Report on Approaches to Database Translation

Leonard Gallagher and Sandra Salazar

Center for Programming Science
and Technology
Institute for Computer Sciences
and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued May 1984

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication

Library of Congress Catalog Card Number: 84-601055

National Bureau of Standards Special Publication 500-115
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-115, 87 pages (May 1984)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1984

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

Page

REPORT ON APPROACHES TO DATABASE TRANSLATION

Leonard J. Gallagher
Sandra B. Salazar

Transporting a database from a source to a target environment has often been an expensive and complex project. In large part this is due to the lack of standards for data models and database interchange forms. This report describes approaches to database translation, discusses candidate interchange forms, and recommends a method for representing the data structures of newly proposed network and relational data models in a form suitable for database interchange. Methods for representing other commonly used database structures in terms of the proposed standard structures show that automated database translation is feasible for most currently installed data models.

A review of various candidate interchange forms shows that the proposed ANSI and ISO Data Descriptive File appears to be the best candidate for character representation of nearly all commonly used database data structures. The form also allows interchange of binary strings in the data fields. Acceptance of standard data models and general database interchange forms could produce substantial benefits to DBMS users in terms of cost savings and increased flexibility. Subsequent vendor supplied, automated tools for reading and writing database structures into standard forms for interchange would make data sharing between non-homogeneous installations a convenient and inexpensive operation.

Key words: ANSI; conversion; data interchange; data models; database; DBMS; data descriptive file; ISO DDF; interchange forms; software standards; translation.

1. Introduction

Because of the dynamic nature of the computer industry, there is an increasing need to transfer data and application programs from one computer environment to another. The entire process of transporting an application system from one environment to another while maintaining the functional requirements of the original system is termed conversion [COLL80]. The conversion process consists of a number of different phases, including planning, data preparation, and testing; but the essence of conversion is the translation phase in which the actual source to target transfer occurs. When a database management system (DBMS) is involved, the conversion process is complicated, due primarily to the fact that the DBMS imposes a structure on the data and on data manipulation. The situation is further complicated since there are no standard DBMS's and consequently few general conversion tools. Each conversion to or from a DBMS tends to be a unique situation.

The required resources and accrued costs associated with database translation are closely related to the dissimilarities of the data models of the source and target environments, the availability of automated translation aids, and the experience of the conversion personnel. The following situations are prime factors affecting the costs of database translation:

* There are no general-purpose database translation aids (e.g. documented software packages or "turnkey" products) currently available that have effective applicability to multiple translation environments. Many agencies are required to design, develop, test, and document the aids needed for a particular conversion.

* Translation aids that do exist are usually tailored to very specific source-and-target combinations.

* The lack of translation aids forces many database conversions to depend on specially developed, one-time aids constructed in-house or under contract.

* Translation experience among agency personnel is often acquired through "live-and-learn", on-the-job conversion projects. In-house learning experiences are often very expensive--especially so in this instance because database translation can be very complex.

- Translation frequently requires detailed expertise with the data handling conventions of two systems--the source system and the target system. In most data processing shops, the degree of competence is high for any one given system but diminishes as the scope extends beyond more than one.

- DBMS and database translation standards are not available. There is no standard DBMS that may be incorporated to circumvent the costs and complexities associated with database translation. Also, there are no standard "unload" and "load" methodologies adhered to by the various database management packages on today's market.

The focus of this report is restricted to one area of the translation phase of conversion in a database environment, namely database translation. We deal with transferring data and data definitions from a source, which can be either a batch file system or a DBMS, to a target DBMS. We will not consider application program translation, which deals with expressing and converting operations on the data. Although not explicitly addressed here, a general purpose database translation approach may be useful in a distributed database environment involving heterogeneous DBMS's.

The purposes of this report are:

- To describe database translation and problems associated with it.

- To present and evaluate alternative approaches to database translation.

- To describe the data structures of proposed standard DBMS data models.

- To describe a recommended approach to database translation based upon a standard data interchange format.

- To present results of database translation studies undertaken by the Institute for Computer Sciences and Technology (ICST).

Chapter 2 describes the database translation process and summarizes the current state of research in this area. Following that, we discuss in chapter 3 the various interchange forms that have been proposed as potential standards.

Specifications for mapping between data structures and one of the interchange forms are given in Appendix A, and detailed examples appear in Appendix B. Since efficient and cost-effective translation is dependent upon standard structures and methods, chapter 4 discusses proposed standard data models and other commonly used data structures. Chapters 5 and 6 report on related research undertaken by ICST. The results of intermodel database translation experiments, using two different interchange forms, are analyzed in chapter 5. Chapter 6 summarizes a cost/benefit analysis of database conversion performed under contract to NBS. This report concludes with a look toward the future impact of standardization on database translation.

2. The Database Translation Process

2.1 A General Model of Data Translation

A generalized procedure for performing data translation from a source database or set of files to a target database or set of files is illustrated in Figure 2.1. Essentially, three programs--a writer, a mapper, and a reader--are involved. Data, formatted according to an interchange form, is passed between these system components in files.

An <u>interchange form</u> (ICF) specifies the format for the transfer of data occurrences between DBMS's or between a file-based system and a DBMS, as well as the format for the transfer of the rules for the correspondence between the ICF structures and the data occurrence structures of the source or target data model. For example, the form might specify that the schema (i.e. the description of the data) is transferred as a text string, followed by the data occurrences.

A <u>writer</u> reads the source database or set of files and writes ICF files onto some medium (such as disk, magnetic tape, computer network). The structures within the ICF files correspond one-to-one to the structures within the source database. The writer is driven by the source database's schema.

A mapper reads ICF files that were produced from the source database and writes corresponding ICF files for later transfer to the target database. A mapper is driven by the source and target database schemas, as well as by a table or other structure that indicates the correspondence between the underlying data structures of each schema.

A reader reads ICF files and writes the target database. The structures within the ICF files should resemble the structures of the target database. The reader is driven by the target database schema.

If the source and target data models are identical, then the mapper and one set of ICF files can be omitted, as one set of ICF files is both the output of the writer and the input of the reader.

2.2 Required Functions

Several functions that are required for database translation can be abstracted from the general procedure outlined above. Standardization of the functions, and their provision by DBMS vendors, would facilitate the translation process. These required functions are:

1. To read a source database or set of files and write ICF files whose structures correspond to those of the source data model.

2. To read ICF files whose structures correspond to a specific data model and write a target database of that same data model.

3. To read ICF files whose structures correspond to one data model and write ICF files whose structures correspond to a different data model. (This function is not relevant if the source and target data models are identical.)

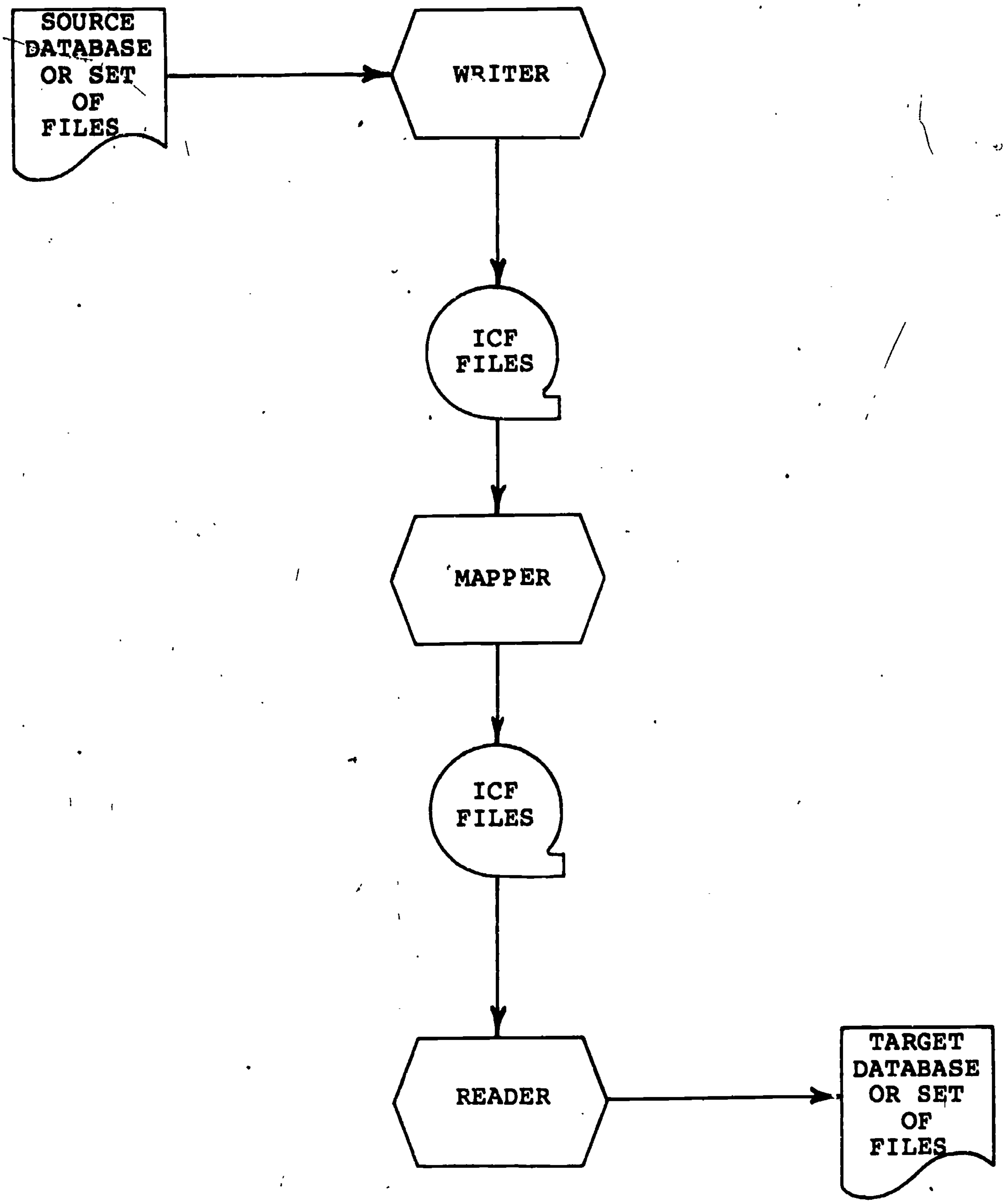

SOURCE DATABASE OR SET OF FILES
WRITER
ICF FILES
MAPPER
ICF FILES
READER
TARGET DATABASE OR SET OF FILES

Figure 2.1 General Procedure for Database Translation

2.3 The Value of a Generalized Approach

In order to make database translation an efficient and cost-effective process, general purpose conversion aids, that is, documented software packages that perform the functions described above, must replace the use of specially developed, one-time programs. Acceptance of a small number of data models and a standard interchange form would hasten vendor development of writers, readers, and mappers.

An example of user and vendor interest in a generalized approach to data exchange is the Initial Graphics Exchange Specification (IGES) [KELL81, SMIT83]. IGES is a communication file structure for data produced on and used by Computer-aided Design and Computer-aided Manufacturing (CAD/CAM) systems. This structure provides a common basis for the automatic interchange of data between interactive graphics design-drafting systems, for the transfer of data to and from external application programs, and for the archiving of the data. The IGES project is an organized effort of both government and industry on a national level to resolve interface problems by introducing a set of specifications for standard data structures and formats. That users and vendors alike recognize the value of this response to the data exchange problem is demonstrated by the rapid acceptance of IGES as an American National Standard [ANSI81].

In a similar manner, standardization of data models, of an ICF, and of inter-model mappings would alleviate some of the difficulties involved in developing general purpose software for database translation. Standardization of an ICF would greatly facilitate specification of inter-model mappings, because the mappings could be defined in terms of the ICF structures. Standardization of an ICF implies standardization of data models, because ICF structures are to be defined along with rules for their correspondence with data structures of specific data models. As discussed in chapter 4, proposals for a network model and a relational model are currently under consideration by ANSC X3H2.

3. Proposed Interchange Forms

The interchange of data between different database systems can be facilitated by the use of a common interchange form. This chapter discusses several candidates to serve this purpose: the Structured Data Interchange Form proposed by James P. Fry and associates at the University of Michigan, the Data Descriptive File advanced by ANSI technical committee X3L5 and ISO TC97/SC15, the data extraction and restructuring system designed by Robert W. Taylor and researchers at IBM Research Laboratory, a data interchange form used by several personal computer software vendors, a specification for application data interchange under development within the British Standards Institution, and the standard format data unit project under development by NASA and other national space agencies.

Each of these candidates has characteristics that make it potentially suitable as a database interchange form, although some are much more general or flexible than others. One of these candidates does stand out, however, in that it is already a draft international standard for data interchange. For this reason we focus on the Data Descriptive File in chapter 4 and Appendix A of this report to show how it can be used to represent the structures common to most existing data models.

3.1 The Structured Data Interchange Form

The Structured Data Interchange Form (SDICF) [FRY81] was designed specifically for database interchange and supports the three principal types of database management systems--network, relational, and hierarchical. With several refinements, it could be sufficiently general to support the exchange of other kinds of structured data, that is, data in which the relationships between entities and attributes are explicitly described through a data definition facility. The data interchange file consists of both the data occurrences or values to be transferred, and a description of their logical and physical characteristics and the relationships within the data. Structured data existing in the source environment is translated into the format defined by the SDICF. The final step, left to the user, is to translate the SDICF into the format required by the target.

An SDICF file is composed of two sections: a description section and a data section. The description section describes the logical structure of the database, includes selected storage structure information, and partially defines the format of the data section. The data section contains attribute values with additional information to allow reconstruction of the database in the target environment.

The description section contains one control record and up to six types of description units, depending on the data model associated with the data being described. The contents of these units are summarized below:

* The control record includes a schema identifier, which is the mechanism to associate a data section with the correct description section.

* A domain unit is required to define each domain of a relational database.

* An attribute unit is required to describe each elementary piece of data accessible by the DBMS (that is, what is termed a field, data item, element, etc., depending on the DBMS).

* An aggregate unit describes an aggregate, a named collection of attribute values within an entity. An aggregate corresponds to a CODASYL data aggregate.

* An area unit specifies a logical storage area for record types for a CODASYL-like network database.

* An entity unit defines what are termed record types, relations, segments, etc., depending on the data model. In the SDICF representation, an entity is a named collection of attributes and aggregates. An entity unit associates a name and an identifier with the entity, defines the components (attributes and aggregates), and specifies the primary key and secondary indexes. To handle CODASYL-like DBMS's, the entity unit can include an area clause to specify the area(s) in which instances of the entity may be stored and a location mode clause to specify access information. An association clause indicates the association(s) in which the entity participates as either a member or an owner (in the sense of the network or hierarchical models).

- An association unit declares an association, a named collection of entities in which one entity serves as the owner and one or more entities are members. Associations are used to represent set-types in the CODASYL model and parent-child relationships in the hierarchical model, and to provide a structuring mechanism where none explicitly exists, as in the relational model.

The data section of an SDICF file contains one control section followed by any number of data units. The control record has the same specifications as the control record of the description section. The data units contain the actual data from the database being interchanged. Each data unit contains the data of one entity instance (e.g. relation, record occurrence) plus information needed to maintain the database structure. This information consists of entity, instance, area, attribute, and association identifiers, as well as association pointers that link owner and member instances of an association.

The constructs supported by the SDICF are thus a composite of the constructs of the network, hierarchical, and relational models. Mapping from a source database to an SDICF file is relatively direct, with little or no restructuring. If the interchange is intra-model, the mapping from the SDICF file to the target database is also straight-forward. Inter-model translation is considerably more complicated, as the SDICF to target mapping involves restructuring at the definition level. Much of the "implicit semantics" of a database are not captured in the SDICF and must be reconstructed in the target environment.

3.2 The EXPRESS Form

The Data Extraction, Processing and Restructuring System (EXPRESS) [TAYL82] was developed by IBM Research in the late 1970's as a research prototype to study the data translation problem. EXPRESS is based on a standard interchange form, with the added notion of high level operations on forms.

The system follows the general translation model described in chapter 2, with the components being referred to as the Reader, Restructurer, and Writer, instead of the Writer, Mapper, and Reader, respectively. The Reader is a custom program, written in the DEFINE data description language, that reads the file to be converted and creates an output file in internal form. The Restructurer is a custom program, written in the CONVERT restructuring language, that produces one or more restructured files, also in internal form. The Writer, also a DEFINE language program, maps the restructured internal form files to the target database. The system also employs a file description catalog to communicate internal form data structure descriptors between the components.

EXPRESS is based on a methodology that combines a data description capability--the "form" concept, with a data manipulation capability--a set of restructuring operators over forms. The data manipulation capability of EXPRESS is claimed to produce a much more powerful tool than a form that has merely a data description capability. The data manipulation feature aids in restructuring in the case of inter-model mapping, eliminating the need for the user to write a one-shot program in a conventional language.

The form itself is a series of one or more hierarchical "sections" of data. A section is a hierarchical record that includes all dependents at all levels, corresponding to a database record in some hierarchical data models. The form can represent databases of different data models. For example, a flat file can be considered as a series of root-only hierarchies, and in many cases a network can be decomposed into a family of hierarchies.

The high-level restructuring language CONVERT consists of data mapping and data validation operators on forms. Among these operators are SELECT, which selects sections based on boolean qualification, projects fields, and eliminates subtrees; SLICE, which flattens a hierarchical branch to a flat file; and GRAFT, which joins two hierarchies at the root. These operators produce forms that correspond to the structures of the target system.

Unlike the SDICF and DDF interchange forms, the EXPRESS form was designed in conjunction with the mapper or restructurer component of the translation system. Thus, the operations on the form were considered when developing the structure of the form. The requirement to produce a sufficiently general and easy to use manipulation language for restructuring in turn restricted the complexity of the form. An interchange form that would handle more general structures than the hierarchical structures supported by the EXPRESS form would require more complex data description and data manipulation languages.

The internal representation of a form is as a series of fields and subtrees. A field is either an integer, floating point number, or fixed or variable length character string. Subtrees are represented by integer offsets and are connected by pointers. The section is encoded in top-to-bottom, left-to-right order. Large sections may be segmented into records which are identified by subsequence numbers.

Possible advantages of the EXPRESS system include the following:

* The form is specifically suited to non-relational data, as it was developed for the purpose of translating a hierarchical database.

* The encoded form does not pass information about the source environment to the target; it is assumed that the target system does not usually need to know the detailed structure of the source environment. If the detailed encoding characteristics of the data are necessary, the recommended solution is to transmit, along with the form description, a data description which can be used in compiling the writer program.

* Hierarchically encoded forms preserve a physical clustering of data that is useful in systems of all data models.

3.3 The Data Descriptive File

The Data Descriptive File (DDF) for information interchange is a draft international standard under development by ANSC X3L5 and ISO TC97/SC15 that establishes media and machine independent formats for interchanging information between computing systems. The proposed ISO Standard [ISO83a] specifies:

* media-independent file and data record descriptions for information interchange.

* the description of data elements, vectors, arrays, and hierarchies containing character strings, bit strings, and numeric forms.

* a data descriptive file comprised of a data descriptive record and accompanying data records that enable information interchange to occur with mimimal specific external description.

* a special data descriptive record that describes the characteristics of each data field within the corresponding data records.

* three levels of interchange depending on the complexity of the allowed structure.

With minor changes, the DDF can be used as the interchange form to transport network, hierarchical, or relational databases. Essentially, each structure of the data model (eg. record type, set type, relation) is transformed into a sequence of characters forming a single DDF. The complete database may then be transported as a sequence of such files on any medium capable of representing, logically, a sequence of characters.

A Data Descriptive File consists of a Data Descriptive Record (DDR) followed by a sequence of Data Records (DR). The DDR and DR records have the same structure, consisting of "leader", "directory", and "data" portions. The "leader" is a sequence of 24 characters that gives the total record length in characters, codes for the level and type of the record, and information for reading the "directory" portion. The "directory" establishes integer "tags" that correspond to fields in the "data" portion of the record and gives starting positions and lengths for all such fields. For an interchange file consisting solely of fixed-length records containing only fixed-length data fields in which the DR's have identical leader and directory values, the leader and directory of the first DR apply to all subsequent DR's. The leader and directory of the subsequent DR's may be omitted.

The "data descriptive area" of the DDR contains a "data descriptive field" for each of the "user data fields". Each data descriptive field associates a data name or a reserved word with each tag. The "data fields" of the "user data area" of the DR contain the user information to be interchanged. Each data field is an instance of the user data structure and data types defined by the DDR data descriptive field with the corresponding field tag. Data names in the DDR correspond to data values in the DR if and only if they have identical tags.

The proposed standard provides for three implementation levels from which the users may choose depending on the complexity of their data structures. Level 1 supports multiple fields containing simple, unstructured character strings. Level 2 supports level 1 and also processes multiple fields containing structured user data comprising a variety of data types. The third level supports level 2 and hierarchical data structures.

The Data Descriptive File form is a general form for transferring data and does not require extensive modifications to handle different data models. One of the proposed changes to the X3L5 specifications to support database interchange is the use of several unassigned tags in the DDR to indicate such information as record names of owner and member record types and the kind of structure (eg. record type, set type) represented by the DDF. Appendix A consists of a specification for mapping database data structures to the DDF structure.

3.4 The Data Interchange Form

In recent years several commercial software houses dealing with application systems on personal computers have agreed to a common Data Interchange Form (DIF) for interchanging simple, two-dimensional tables of information [KALI81]. Like the Data Descriptive File (DDF) described above, every DIF file has two parts: a "header section" containing descriptive information about the table, and a "data section" containing the data values. The DIF format is much simpler than the DDF format in that it does not attempt to handle variable length records, variable occurrences of data values, sequences, array structures, bit field data, or nested repeating groups. It handles only two data types - numeric values that may be signed integers, decimals, or scientific notation, and string values that may be alphanumeric characters with no control characters or quotation marks. The numeric values are always represented in digital form.

The basic structure of a DIF file is a "line" of information. A line is purposely very simple so that it can be handled by nearly all applications, even those that only read records of 40 or fewer characters. A new line in the DIF specification is very analogous to a field terminator in the DDF specification. A line usually consists of just a single data value, and in no cases does it consist of more than two numeric values separated by a comma.

The header section of a DIF file consists of a number of required "header items" - one to specify the table name, one to specify the number of columns, one to specify the number of rows, one to specify the name of each column, and one to declare the end of the header section and the beginning of the data section. DIF also specifies optional header items, including formats for inserting comments and physical size information. Each header item consists of three lines of information - the first line identifies the type of header item, and the second and third lines specify the item value and give supporting information to identify if the item refers to the whole table or to a specific numbered column.

The data section of a DIF file consists of a number of "data items", each consisting of two lines of information. A sequence of such data items specifies the data values of each row of the table. The first line of each data item identifies the data item type (e.g. string, numeric, or special) and specifies the data value if the item is numeric; the second line specifies the data value if the item is a string. Both lines are always required - the line that does not carry the data value has a NULL value. The "special" data items are used to declare the beginning and end of each tuple and the beginning and end of the data section itself.

A DIF Clearinghouse [cf. KALI81] coordinates and distributes information about DIF and commercially available programs that use DIF. Its primary application is to transfer application data to and from a commercial spreadsheet available on a wide range of personal computers.

3.5 The DIAL Interchange Form

The structure and representation of data for interchange at the application level (DIAL) [BSI82] is a draft proposal, subject to further development, for interchanging machine-readable, application data in the United Kingdom. It consists of a grammar for representing data elements, groups of data elements, and messages.

Each data element represents a value whose definition, representation, identification, and meaning have been agreed to by the interchanging parties. Examples include numeric values, identification and classification codes, financial amounts, names and addresses, or goods and services. A data element consists of two parts, a generic part to identify the concept and an attribute to represent the value of that occurrence. Each data element agreed to for data interchange is assigned a unique 4-character identifier that serves to identify occurrences of that element in the interchange form.

A group is used to link logically related data elements. Groups facilitate the representation of nested and repeated data values. Each group has a definition that is agreed to in advance by the interchanging parties. The definition consists of a list of data elements that may be present in that group and a unique 3-character identification tag for the group.

A message is the basic unit of data interchange. It may represent a document or form (e.g. invoice), or any other object of interchange (e.g. a database) that is self-contained and can be communicated and processed independently. Each message has a definition that is agreed to in advance by the interchanging parties. The definition consists of a list of data groups that may be present and a unique 6-character identification tag.

Messages are formatted as a sequence of characters from a pre-defined character set consisting of uppercase letters of the alphabet, digits, and special symbols for delimiters and separators. A message header uniquely labels the message and identifies the message tag and version number that defines its contents. Facilities for routing the message between the interchanging parties are not part of the specification.

3.6 SFDU Interchange Project

The standard format data unit (SFDU) project was initiated by the National Aeronautics and Space Administratio (NASA) to facilitate the exchange of space research data using a variety of communication media. An SFDU is a collection of data that conforms to standard specifications for formatting and labeling. A NASA proposal of a conceptual framework for development of an SFDU guideline [CCSDS83] was accepted by Panel 2, Standard Data Interchange Structures, of the Consultative Committee for Space Data Systems (CCSDS) in May 1983. The CCSDS is a consortium of space agencies from a number of countries including Brazil, France, Germany, Japan, the UK, and the USA.

An SFDU consists of four parts:

- a globally interpretable icentifier, called a primary label,

- specific additional identifiers or descriptors of the data, called Supplementary Labels,

- the data contents, encoded in a form suitable for communication, and

- a capability for data to be appended to an existing SFDU or for an existing SFDU to be combined with others in a batch.

A format for the primary label was tentatively approved by Panel 2 of CCSDS in November, 1983--development of the other parts is continuing. Papers presented at a 1981 worhshop on self-describing data structures [NASA82] give approaches that may be appropriate to the other parts.

One feature of the SFDU that distinguishes it from the SDICF, DDF, and DIAL forms described above is that it is proposed as a binary interchange form rather than a character string form. Although the DDF has provisions for interchanging bit string data fields, its labels and identifiers are all character based. The SFDU as originally proposed would have binary codes for all labels and would be targeted, primarily, for transporting binary field data.

3.7 Choice of an Interchange Form

The SDICF and EXPRESS forms are concerned with the logical representation of specific database structures--they are less concerned with physical implementation details such as character string representations for elementary data types, self-description of the linear representation, and the use of length declarations and delimiters. This may not be a disadvantage if they are used in a specific vendor environment, but application in a multiple vendor environment would likely require additional specification in this area. The DDF and DIF forms emphasize details of the physical layout, but only for a single file. They do not specify how integrated database structures should be represented in terms of the file structure. Also, without additional specification the DIF form would not be able to handle network or hierarchical database structures. The DIAL form is intended for application-specific messages whose generic structure is pre-defined and known to both sending and receiving parties. It would not accommodate general database structures without much additional specification. The SFDU project has not yet matured and has no agreed upon formats for representing user data structures.

We conclude that each of the candidate interchange forms is incomplete and cannot be used for general database translation without additional specification. Some forms require specific implementation details to describe how the data structures are laid out in linear form; others require additional specification for mapping specific database data structures to the candidate interchange form. We believe that at least in the near term any successful interchange form will have to have the same general flexibility that the DDF has for declaring the relative position of specific data fields in the overall linear representation of the database. This feature would allow a file reader to handle variable length data structures in an optimal manner. Thus we focus our efforts in chapter 4 and in Appendix A to provide the additional specification required for using the DDF to represent arbitrary database structures.

We show in chapter 5 that no definitive argument can be advanced at this time to support one of the forms SDICF or DDF over the other, based on such considerations as completeness of specification, efficiency, or ease of use. Thus, use of the DDF form is favored because it is already near acceptance as an ANSI and ISO standard. Much development time and effort can be saved by using this existing and familiar form whenever possible. Use of the DDF, together with the database representation rules specified in this report, provides a complete specification for representing proposed standard database structures in a linear form that can be transported among heterogeneous database installations. Acceptance of such a standard interchange form by the user community would make convenient vendor support highly probable and would ensure more rapid development of additional conversion tools and capabilities, with the result being greatly decreased expenses for database translation and increased flexibility to users.

The DDF interchange form was never intended to be a highly efficient representation for use in real-time data interchange. Acceptance of the DDF as one vehicle for database translation, especially for archival purposes or for occasional data exchange, should not stifle development of other forms that may be more suitable or offer greater efficiency for specific applications or in special environments. In particular, work on standard forms for data interchange in open systems interconnection (OSI) [e.g. ISO83b, NBS83] will continue within Federal, ANSI, and ISO standardization committees.

4. Proposed Standard Data Models

4.1 Data Models and Database Management Systems

A data model is a collection of data structures together with operations that manipulate the data structures for the purpose of storing, querying, or processing the structure contents. A data model may also include integrity constraints defined over the data structures, or it may include access control facilities or mechanisms for defining various external user views of the database. Some data models provide physical storage structures and physical access methods as part of the data model, but usually a data model is limited to the data structures and operations that are available to an end user and may be accessed from an application program.

A database management system (DBMS) is a general purpose, application independent, software package used in association with computer hardware to facilitate the entry, storage, processing, sharing, and retrieval of data from a database. The portion of a DBMS that deals directly with the processing of the data structures of a data model is sometimes referred to as the database control system. A DBMS supports a data model and is an implementation of that data model. Some database management systems may support multiple data models by providing different user interfaces to the database. A DBMS provides for transformation of the logical data structures of a data model to the physical storage structures of a particular hardware environment. The DBMS goes beyond the data model in that it must provide for communication with the operating system of the host computer, as well as interface with programming languages or associated software systems such as data dictionaries, report writers, statistical packages, and libraries of special data processing functions. A DBMS also provides concurrency control, backup, and restart, as well as dumping and loading facilities for all databases under its control.

Since database translation is concerned with transporting databases rather than database application programs, this document focuses on the data structures of data models and is not concerned with different data manipulation operations that may be used to distinguish different data models. This allows us to make certain simplifications in partitioning data models by structure, since many data models have nearly identical structures but widely varying operations. This chapter presents an overview of the data structures associated with data models that have been mentioned extensively in the literature, including network, hierarchical, and relational data models, as well as "entity-relationship" structures and structures associated with various "value-based" approaches to database management. The network and relational data models are given principal attention because they are candidates for standardization by the American National Standards Institute (ANSI) [X3H283a, X3H283b]. The data structures of most other data models may be described in terms of network or relational structures. Database interchange involving non-standard data models would then require one extra step to define the data in terms of standard structures before interchange. The following sections describe how this might be accomplished.

4.2 Data Types For Database Interchange

Every data model has its favorite collection of data types. A data type is the definition of a set of values that can be represented in a data model. A value is primitive in that it has no logical subdivision within the data model. Values are the basis of definition for the other data structures of a data model. Most data models have a character string data type and at least one numeric data type. Some models make a distinction between fixed-length and variable-length strings, or between exact and approximate numeric values. Most numeric types may be defined with different degrees of numeric precision. Other favorite data types include: calendar date, time-of-day, zip code, sex code, Boolean, money, complex numbers, long strings of text, enumeration types, or various forms of pointers and identifiers. For database interchange, each of these types may be represented as a sequence of characters or bits. American National Standard X3.42 [ANSI75] gives a standard character string representation of explicit and implicit decimal point and scaled and unscaled representations of numeric values. Other syntactic representations for elementary data types [e.g. CCITT83, NBS83] are under consideration by IEEE, ISO, and other standards developing bodies.

The ANSI database committee has defined a set of data types derived from the established ANSI standard programming languages. Instead of trying to describe all such data types we will focus on three elementary types that are common to a number of languages. These include character strings, fixed point numbers, and floating point numbers. Subsequent paragraphs present definitions of these common data types and Appendix A shows how they might map to ANSI X3.42 representations in a standard database interchange form.

A character string is a finite sequence of characters taken from some well-defined character set. Character sets for databases may include: the human readable "graphic characters" as specified by ANSI X3.4 [ANSI77], the complete set of 128 characters as specified by ANSI X3.4, various international character sets as specified by ISO standards, or application-specific character sets such as videotex or special graphics symbols. Each character string has a fixed length, a positive integer associated with the string that describes the number of characters or number of bits in that string. Strings may be of variable length, but logically they are padded with blank characters when used in comparisons or interchange.

Numbers are values that have normal mathematical properties. They are defined as real numbers with decimal base. Fixed point numbers are assumed to be "exact" values. All fixed point numbers have an associated precision and scale. The precision is a positive integer that specifies the number of significant decimal digits. The scale is a signed integer that indicates the position of the decimal point, Floating point numbers are assumed to be "approximate" values. Floating point numbers consist of a mantissa and an exponent. The mantissa is a fixed point number. The exponent is an integer that specifies the value of the number to be the value of its mantissa multiplied by 10 to the power of the exponent. Every floating point number has a precision. The precision is a positive integer that specifies the precision of the mantissa. The scale factor of the mantissa is non-negative and less than or equal to the precision.

The following sections describe specific data models. We assume that the data types just described are the elementary data types of these data models. For database interchange, all other data types would first be described in terms of one of these. Appendix A specifies representations of the two numeric types in terms of ANSI standard numeric representation forms [ANSI75], and shows how all elementary types could be described using the ANSI/ISO DDF [ISO83a] as a database interchange form.

4.3 The Network Model

The network approach to database management originated in the 1960's and has evolved over the years to become accepted as one of the major approaches for managing structured data in a database. In 1971, the CODASYL Data Base Task Group produced functional and language specifications [DBTG71] for incorporating network database structures in the COBOL programming language. In subsequent Journals of Development, CODASYL DDLC produced language specifications for defining data structures and CODASYL COBOL produced language specifications for COBOL DML operations. In May 1978, the American National Standards Committee X3 formed a new X3 technical committee, X3H2 Database, to develop an ANSI standard for database data definition. In 1981, the X3H2 program of work was extended to include development of both structures and operations for the network model. Specifications for the logical data structures and basic operations of a draft proposed American national standard network database language (NDL) [X3H283a] have been completed by this committee. This proposed standard provides functional capabilities for designing, accessing, maintaining, controlling, and protecting the database. The following network model data structure definitions are derived from the X3H2 specifications.

The network data model contains two basic data structures: the record and the set. The record is the basic unit of data manipulation. Records are stored, erased, found, modified, and connected and disconnected from other records. The set is the basic unit of navigation. The set is a structure that maintains inter-record relationships. A user is able to move from one record to another along logical set access paths defined by the database schema.

A record is a collection of data components. A component is either a data item or an array. A data item consists of a single value; an array is a sequence of data items. Each array has a fixed dimension that is a positive integer. Each positive integer less than or equal to the dimension determines a direction for that array. Each direction has an extent that is also a positive integer. For example, in a two-dimensional array, the extent in the first direction determines the number of rows and the extent in the second direction determines the number of columns. The number of data items that occur in an array is the product of the extent integers of that array. A data item within an array is referenced by a multi-dimensional subscript. An implicit row-major ordering of array items establishes a unique correspondence between a data item referenced by a subscript and its sequential position in the array. For example, in a two-dimensional array with three rows and four columns, items 1-4 occupy the first row, items 5-8 the second row, and items 9-12 the third row. The subscript (2,3) thus references the seventh sequential position.

All records in a database are partitioned according to record type. A record type is the definition of a named collection of record occurrences all having the same component characteristics. A record type defines the components of each record occurrence of the record type and declares a record name for the record type and a component name for each component. Each record of the database is an occurrence of exactly one record type and consists of exactly the data items defined by that record type.

A set is a collection of related records. It is an occurrence of a set type, which is the definition of a named collection of set occurrences all having the same characteristics. The declaration of a set type specifies a set name for the set type and specifies the owner and member record types that are associated with the set type. The set establishes a relationship among its component records that must be maintained by the DBMS. One record from each set is designated as the owner record of that set. Any other record in the set is a member record. The owner record of each set of a given set type is a record occurrence from its owner record type. Each record from the owner record type is the owner record of exactly one set occurrence of that set type. A record occurrence of a member record type of a given set type is a member record of at most one set of that set type. The member records of each set of a set type are maintained in a sequential order determined by the schema ordering criteria for that set type.

The network model supports two special set types: singular and recursive. A singular set type has SYSTEM declared as its owner record type. SYSTEM may be thought of as a special record type consisting of exactly one record, the system record, having no data items. Hence there is only one set occurance of a singular set type, and it always allows direct access to member records without first navigating through an owner record. A recursive set type has the same record type declared both as the owner record type and as a member record type; thus it allows convenient representation of hierarchical relationships, such as manager/employee, among records of the same record type.

All records must be distinguishable by the DBMS, including those that participate in the same data structures and have the same values for each component data item. For this reason, each record is associated with a unique record identifier called a database key. The database key is a conceptual object used to maintain position in the database; it is not directly available to an end user. The action initiated by any database statement is dependent upon the database keys that occur as values of special cursors maintained by the DBMS in a session state for each database session. For database interchange, an external representation of database keys is needed to identify both records and their participation in set relationships. We assume the existence of a facility within the DBMS to make the external representation of each database key available as a fixed-length string of characters. As described in Appendix A, the occurrences of each set type may then be represented in a data descriptive file using the character string database key representations.

A network database definition may include specification of certain integrity constraints on the data items, records, and sets of the database. For example, the length of strings and the precision of numbers are important constraints that should persist with the data itself. Additional integrity constraints may be derived from the following: a check condition is an expression that must be satisfied by the values of a record when it is stored in the database or inserted as a member record in a set, a default value is a value assumed by component occurrences in the absence of a specific value supplied by a user, and a unique constraint is a specification that no two records may occur in the database with identical values for specified components.

A set type description also includes specific integrit declarations. A set ordering specifies whether the logical ordering of member records in a set is sorted, first, last, next, prior, or system default. If order is sorted, then a key declaration specifies the data items that determine the order key. A set insertion declaration specifies whether the insertion of a record as a member of a set is automatic, structural, or manual. If insertion is structural, then values for the structural data items determine how an owner record is selected from the database. A set retention declaration specifies whether the retention of a member record in a set is fixed, mandatory, or optional.

All of the above declarations are integrity constraints often considered to be part of the database structure that should be preserved during database interchange. These declarations could be exchanged via the standard character string syntax of the network model Schema Definition Language [X3H283a], or they could be represented using the same data interchange forms that represent the data occurrences. The first alternative is generally preferable; but if that is not suitable, then appendix A shows a proposed format for the latter alternative.

4.4 The Relational Model

The relational approach to database management began in 1970 with the publication of E.F. Codd's paper [CODD70] proposing basic structures and algebraic operations for a relational data model. This paper spawned a flurry of interest in various high level query and manipulation languages based on a predicate calculus. By the late 1970's both Quel [STON76] and Sequel [ASTR75] had achieved popularity as very powerful yet user friendly data manipulation languages. In October 1982, the ANSI database committee was charged with an additional project to standardize an interface to the relational model. Specifications for the logical data structures and basic operations of a draft proposed American national standard relational database language (RDL) [X3H283b] have been completed by this committee. The ANSI data manipulation language is a close derivative of Sequel. The proposed standard provides functional capabilities for defining, querying, protecting, and altering relational databases, as well as functions to support modification of structure and constraint definitions. The following relational model data structure definitions are derived from the X3H2 specifications.

The primary data structure of the relational model is a table, which is defined in terms of rows and columns. A column is an unordered collection of values of the same elementary data type, and a column entry is a single value of that data type. A value from a column is the smallest unit of data that may be selected from a table and the smallest unit of data that can be modified in a table. Unlike the network model, the relational model does not support arrays of values. A row is a non-empty sequence of values, and it is the smallest unit of data that can be stored into a table or erased from a table. The table itself is an unordered collection of rows that are not necessarily distinct.

Each table is associated with a table definition that defines the table name and table characteristics as well as the column name and column characteristics of each column of that table. Every row of the same table has the same cardinality and contains a value for every column of that table. The relational model supports the notion of null values. The null value is a special value that is comparable with any value, but is distinct from all non-null values. The null value is assumed for a column position in a row whenever no non-null value is specifically assigned.

Tables may be base tables, derived tables, or viewed tables. A base table has a persistent storage representation and a persistent table description. A derived table is a temporary table derived from one or more base tables during the execution of a database statement. A viewed table is a derived table that has a persistent description. A viewed table provides subschema views of the database to external users. All three types of tables may be the objects of database statements in the relational data manipulation language, and may be the objects of database interchange.

All rows in a table must be distinguishable by the database control system, including those rows in the same table that have identical values for each column. For this reason, relational implementations must rely on some kind of unique row identifier, be it physical placement in a file or use of logical indicators or pointers, to distinguish rows. However, the relational model is different from the network model in that row identifiers are not necessary to maintain position in the database or to represent inter-record relationships. Hence, it is never necessary to require that record identifiers be known to end users or that they be representable in database interchange forms.

Even with the above difference in row identifiers, and with the absence of arrays and the addition of null values, a relational table is still essentially identical to the record occurrences of a network model record type. For the purposes of database translation, a table may be represented in an interchange form as a record type population. We do not state specific rules in Appendix A for representing tables via the DDF interchange form; instead we assume that the record type rules apply equally to tables, with the exception that database keys can be ignored and a special null value representation is needed for null character strings and numbers.

Some implementations or user installations of the relational model include definitions for the primary key and secondary key concepts. A primary key is declared for a single table; it consists of a column whose values uniquely identify a row of the table. A secondary key is also declared for a single table; it relates back to some existing primary key in a different table. The secondary key specifies a column of its table that assumes values comparable to values of its associated primary key. The secondary key value of each row in a secondary key table identifies a unique row in the primary key table. In the absence of specific data structures to represent inter-record relationships, primary and secondary keys are often used to maintain logical connections between tables.

In the ANSI X3H2 relational specifications, the notions of primary and secondary keys are not explicitly defined. Instead, a primary key may be assumed whenever a table unique constraint is specified over a single column, and a secondary key may be assumed whenever a table referential constraint is specified. A referential constraint requires that the "secondary key" of each row of the referencing table have a value that is identical to the "primary key" of some row of the referenced table. A referential constraint has some of the same features of the network model set type in that modify and delete statements may cascade from the primary key row to secondary key rows.

A relational database definition may include specification of certain integrity constraints on the rows and columns of tables in the database. As with the network model, this includes specification of length of strings and precision of numbers, as well as declaration of default values for columns and check conditions for rows and columns. Relational table constraints also include the unique and referential constraints defined in the preceding paragraph. Such integrity constraints are often considered to be part of the database structure and thus subject to preservation during database translation. These declarations could be exchanged via the standard character string syntax of the create table statement in the relational database language. This would be the most straight-forward method for interchanging integrity constraints provided that the target system is capable of parsing the syntax.

Integrity constraints could also be interchanged by using system catalogue tables. Most relational database management systems maintain a schema database, consisting of catalogue tables accessible to end users, that contains all of the structure and integrity information about the database. At the present time there is no standard format for these tables and each implementation defines its own. If system catalogue tables exist, they can be interchanged in the same manner as regular data tables, thus giving a convenient method for interchanging integrity constraints and other schema information. If neither of these interchange methods are suitable, then relational integrity constraints could be interchanged by using the data descriptive files described in Appendix A. Since the only constraint in the relational model that is truly different from those in the network model is the referential constraint, it would only be necessary to add one new representation rule in the interchange form for specifying the referenced and referencing columns.

4.5 Hierarchical Models

The hierarchical approach to database management derives from a generalization of the repeating group structures found in many programming languages. Hierarchical systems evolved independently in the 1960's so that currently there are many different versions in the marketplace. Most such systems support data structures similar to the repeating group capabilities in COBOL, but they vary widely in their approach to data access and data manipulation. Some systems are very navigational in that they support movement from one record to another: up, down, and across the underlying tree. Other systems make extensive use of indexes and algebraic operations on sets of record identifiers to isolate desired portions of the database. For the purposes of database interchange, it is possible to ignore these significant differences in data access methodology and concentrate instead on the underlying tree-like data structures common to most such systems.

The basic structures of hierarchical models may be viewed as a subset of the network model data structures defined above. The main structure is a node (sometimes called segment or component) that is essentially equivalent to a network model record type. Nodes are connected one to another in a parent-child relationship very much like the owner to member record relationship in a network model set type. The major restriction is that no child node may have more than one parent node associated with it. This restriction simplifies data definition to the point that it is not necessary to define path names (i.e. network model set names) for the link between owner and child nodes; each such path is uniquely identified once the owner and child nodes are known. Other restrictions are that direct access to a child node similar to a network model singular set, or circular node connections as in the network model recursive set, are usually not allowed. These restrictions on data organization limit the flexibility for defining highly integrated databases, but provide certain capabilities for operational efficiency and retrieval flexibility.

For database interchange of hierarchical databases, one may assume that node populations are transported in the same manner as are network model record type populations and that parent-child relationships are transported as network model set type populations. For naming purposes, the set type representing such a parent-child relationship has its set name and member record name equal to the name of the child node, and has its owner record name equal to the name of the parent node. The retrieval order of child node occurrences in the hierarchical specification determines the order of member records in each set. If retrieval order is not important, then member record order is assumed to be system default. This method for transporting hierarchical databases assumes the existence of a facility for generating unique identifiers for each node occurrence that can be used in the same way that database keys are used in transporting network databases.

Another method for transporting hierarchical databases is to assume an equivalent representation of the database using the structures of the relational data model. This method would require the specification of a secondary key, as specified in the relational model above, into each child node. Each secondary key would refer back to a primary key in the parent node. In this manner, each hierarchical parent node, child node, and parent-child relationship could be represented as two tables: one for the parent node with the primary key clearly identified, and one for the child node with the secondary key clearly identified. If order of the child node occurrences is critical to the hierarchical database semantics, then the relational table representing the child node must also contain a column for the ordinal position of that node in its parent-child relationship.

A third alternative for transporting hierarchical databases might be to make direct use of the hierarchical capabilities in the data descriptive file (DDF) interchange form [ISO83a]. An hierarchical structure can be mapped into the DDF interchange form using a single logical record for each tree occurrence. Under this approach, a specific data tree is first described by a binary relation in a data descriptive record; then a preorder traversal of each specific tree is contained in a data record. The proposed DDF standard contains an appendix that presents the details of this representation technique. Very large tree occurrences could be broken into smaller ones provided that any parent-child relationships lost by this subdivision are replaced by primary and secondary keys as described above.

The integrity constraints of a hierarchical database are specified differently by each implementation of a hierarchical data model. In most cases it will be possible to restate the integrity constraints using the same syntax as specified for the proposed network or relational standards. Integrity constraints could then be interchanged as a parsable character string in the same way that network or relational constraints are interchanged. In other cases, the integrity constraints may be described in English text and passed as a message along with the data.

4.6 Entity-Relationship Structures

The entity-relationship approach for describing a database became popular in 1976 with the publication of an article by P.P. Chen [CHEN76]. Two conferences [ER79,ER81] have been devoted to the logical design and application of databases defined using this approach. The model proposed originally did not include specification of any data manipulation operations; instead it focused on the specification of entity types and the relationships among entity types. Later authors [e.g. JOHN82, SHIP81] have specified operations to make it a complete data model. In most application environments, the entity and relationship specifications are used only to define the logical organization of the database. This data organization is then implemented by using the specific data structures of a network, hierarchical, or relational database management system.

The entity structure of this model may be considered as equivalent to a relational table or a network record type. In most cases only elementary data items are considered, so the array structure of the network model would not apply. As in the case of both record types and tables, one must assume the existence of unique entity identifiers to distinguish among entities that have identical data values. In most cases, such identifiers are not externally visible or accessible. The relationship structure of this model is a generalization of the network model set type. Instead of representing only one-to-many associations between one owner entity type and one or more member entity types as does the network set, the relationships may be many-to-many among any number of participating entity types.

For database interchange, it is necessary to assume that some facility exists for the external representation of entity identifiers. Then each entity type could be interchanged either as a network model record type, with the entity identifier represented directly as a database key, or as a relational model table, with the entity identifier incorporated into a primary key column. Each relationship structure could then be represented as an additional record type or table; this new structure would have one component or column for each entity type participating in the relationship. Each record or row would consist of the entity identifiers that determine one of the associations among individual entities. Integrity constraints could be restated in terms of network or relational model integrity constraints, or they could be described in English text.

4.7 Other Structures

Most database structures can be represented in terms of the specific structures defined above. As a last resort, even the most complex data structures can be represented as a collection of binary relationships, which in turn can be interchanged as tables with two columns. In this section we describe how some of the more common data structures that have not been defined above, but yet appear in current commercial products, are represented in terms of the above structures.

Many database management systems provide for the storage and manipulation of variable repeating items. As an example, an employee record may contain a variable repeating item for the names of family members or for a listing of multiple telephone numbers. Such record structures are not directly representable in either the network model or the relational model. However, there are several ways that such repeating items can be represented in either model. If there is some known upper limit on the number of occurrences of the repeating item, then it can be defined as a one-dimensional array in the network model with a default value (e.g. blanks) defined for unused positions. In the DDF interchange form used in Appendix A there is an alternative for listing such occurrences without wasting unused space.

If there is no known upper limit, or if the number of occurrences varies widely among records, then the repeating item could be defined as a separate record type or table. In the network model it would be connected to the owner record via a set, and in the relational model it would be associated with the owner record by using a secondary key and a referential integrity constraint in the auxilary table. In many implementations, such a declaration would be recognized by the system as a variable repeating item, and the repeating values would be stored next to the owner record in a physically optimal way.

Early specifications of the CODASYL network model included definition of nested repeating groups. Since CODASYL specifications assumed a COBOL programming language interface, repeating groups were defined by nested level numbers and OCCURS clauses just as in a COBOL record. A multidimensional table was defined using nested level numbers with an OCCURS clause at each level. Such repeating groups were deleted from the ANSI network model specifications because each repeating group is easily representable in terms of arrays or in terms of additional record types and set types. As an example, consider a nested record structure as follows:

```
RECORD NAME IS  R
 01  A  PIC X(10)
 01  B  OCCURS 5 TIMES
     02  C  PIC X(10)
     02  D  OCCURS 2 TIMES  PIC X(10)
```

This structure could be represented as a single network model record type with A as an elementary item and C and D as arrays. The array C would be one-dimensional, occurring 5 times, and the array D would be two-dimensional, occurring 5 by 2 times. Any references derived from the data name B would be lost to the data model, as would the association between occurrences of C and the first direction of D. Explanations of such associations could be carried along as comments.

A second representation of the above record structure could be through the definition of additional record types and set types. This approach is particularly suitable if data components B and D occur a variable number of times instead of a fixed number. For example, A could be an elementary item in R, C an elementary item in S, and D an elementary item in T, where R, S, and T are different record types. The names of record types S and T would be derived from data components B and D respectively. Record type R would be the owner of a set type with S as a member, and S would be the owner of a set type with T as a member.

The same situation could be described in the relational model using three tables logically associated by the appropriate primary and secondary keys and referential integrity constraints. For example, the tables R (A,...), S (A,C,...), and T (A,C,D,...) would convey essentially equivalent information with the following restrictions: column A in table R and both column A and column C in table S would have to be declared unique; column A of table S would have to reference column A of table R; and columns A and C of table T would have to reference columns A and C of table S. If the columns defined above were all of the record, then table T alone would suffice. We have added R and S to account for the cases where there are other components at both the 01 and the 02 levels. These components would constitute the other columns in tables R and S, respectively.

The CODASYL specification of the network model also includes definitions for additional structures such as areas and record order keys. An area is a collection of records together with a sequential ordering over the records. This structure still exists in many commercial products, usually as the logical grouping of record type populations stored together on the same physical file or storage device. Areas were deleted from the ANSI database definition because each area is logically equivalent to a singular set type with multiple member record types. The equivalence can be demonstrated by defining exactly one additional singular set type to represent each area. The area name becomes the set name of the new set type, each record type having records contained in the area becomes a member record type of the singular set type, and the sequential order of records in the area determines the record order of member records in the singular set.

A record order key is the declaration in a record type that record occurrences shall be ordered on given data items in a specified way. Record order keys were deleted from the ANSI network model specification because each such key is logically equivalent to a singular set type defined over that record type. The record order declaration becomes the order declaration for the member records in that set. For database interchange, areas and record order keys can be interchanged as singular sets using any interchange form suitable for set types of the network model.

5. Tests of Two Proposed Interchange Forms

As part of the process to select an interchange form, ICST has conducted tests using two of the proposed ICFs. The tests performed to date are preliminary, involving only the interchange of data, not of schemas. The SDICF and the DDF forms were the two ICFs used in the experiments. The rest of this chapter gives an overview of the tests, then discusses some details, results, and conclusions.

5.1 Overview of Tests

This section describes the tests' purposes and general procedures.

5.1.1 Purposes.

The tests had the following purposes:

- To conduct an initial, incomplete test of the adequacy of each interchange form, and to find any problems in the specifications.

- To compare the amount of computer time required to read/parse/write the two forms.

- To compare the amount of space required by the two forms.

- To compare the ease of use of the two forms. This includes the amount of human time required to construct programs to read/parse/write the two forms.

5.1.2 General Procedures.

The following general procedures were used for the tests:

1. For each DBMS used, a schema was constructed and data was loaded for a predefined database, described below.

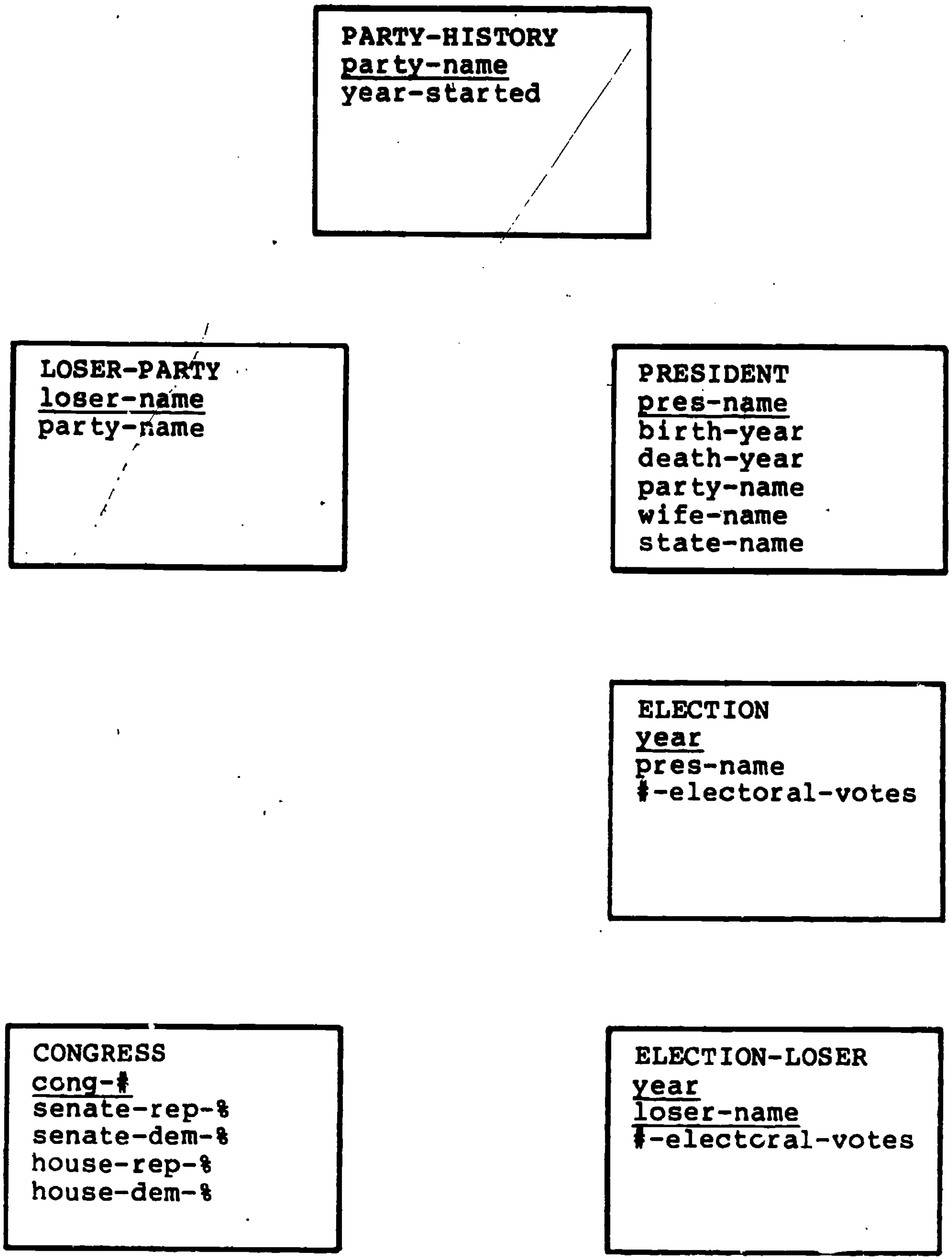
2. For each interchange form, and for each DBMS used, interchange files that corresponded to the original database were first written, stored on disk, then read back in to construct a new database. The new database was compared to the criginal.

3. For each interchange form, and for each ordered pair of different DBMS's used, interchange files that corresponded to the source database were written, then transported, via magnetic tape, to the target DBMS's site. The files were read and transformed at the target site to construct a target database, and finally the resulting database was compared to the original database.

5.2 Details of Test Procedures

5.2.1 The Example Database.

Three DBMS's, representative of the principal data models (network, relational, and hierarchical), were used in the experiments. The experimental database was a subset of a "presidential" database, generated from ICST's source files. The schema diagrams for the three data models are given in Figures 5.1 through 5.3. In the relational schema diagram, key data items are underlined.



PARTY-HISTORY
party-name
year-started

LOSER-PARTY
loser-name
party-name

PRESIDENT
pres-name
birth-year
death-year
party-name
wife-name
state-name

ELECTION
year
pres-name
#-electoral-votes

CONGRESS
cong-#
senate-rep-%
senate-dem-%
house-rep-%
house-dem-%

ELECTION-LOSER
year
loser-name
#-electcral-votes

Figure 5.1 - Equivalent of Relational Schema ("keys" underlined)

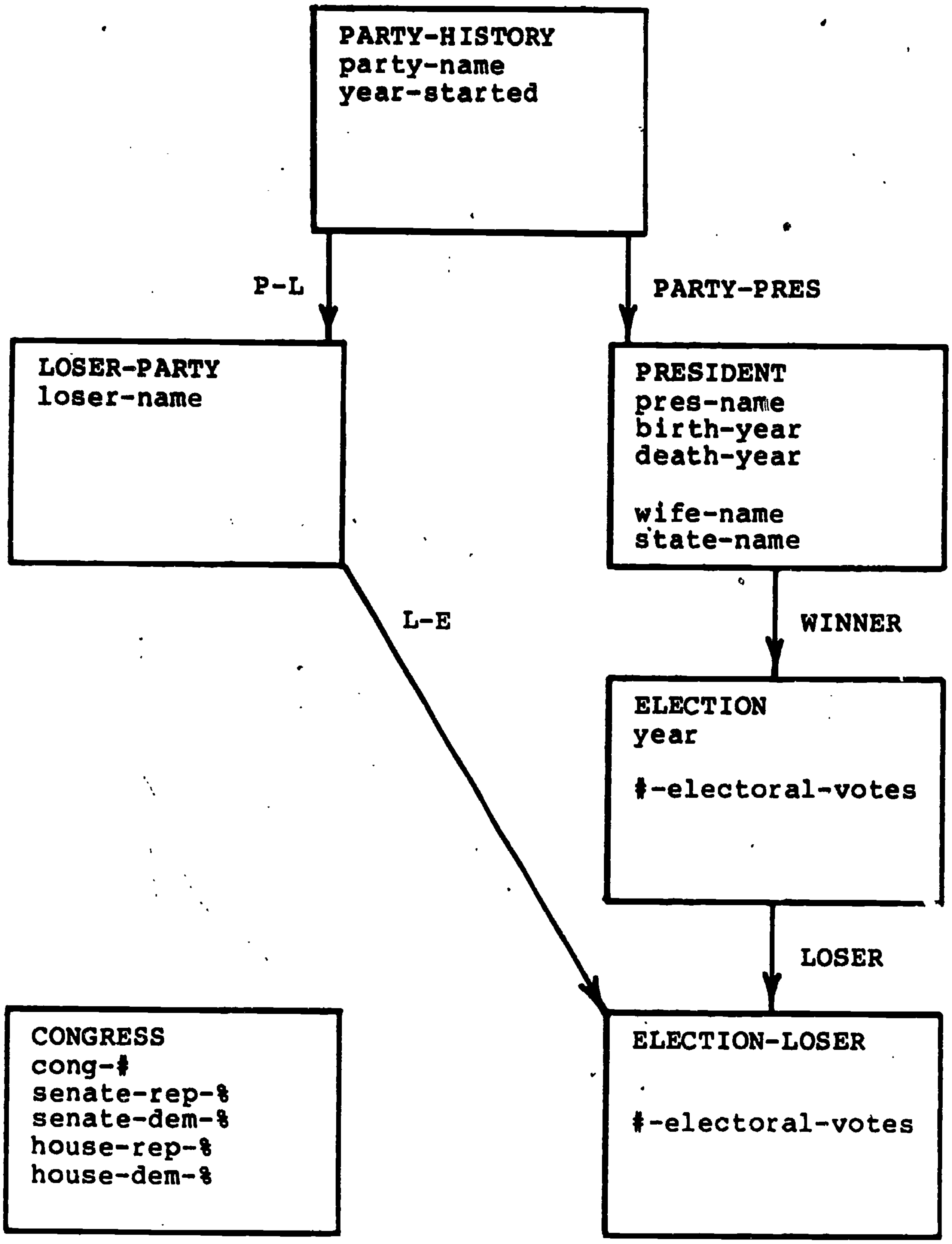


Figure 5.2 - Network Schema

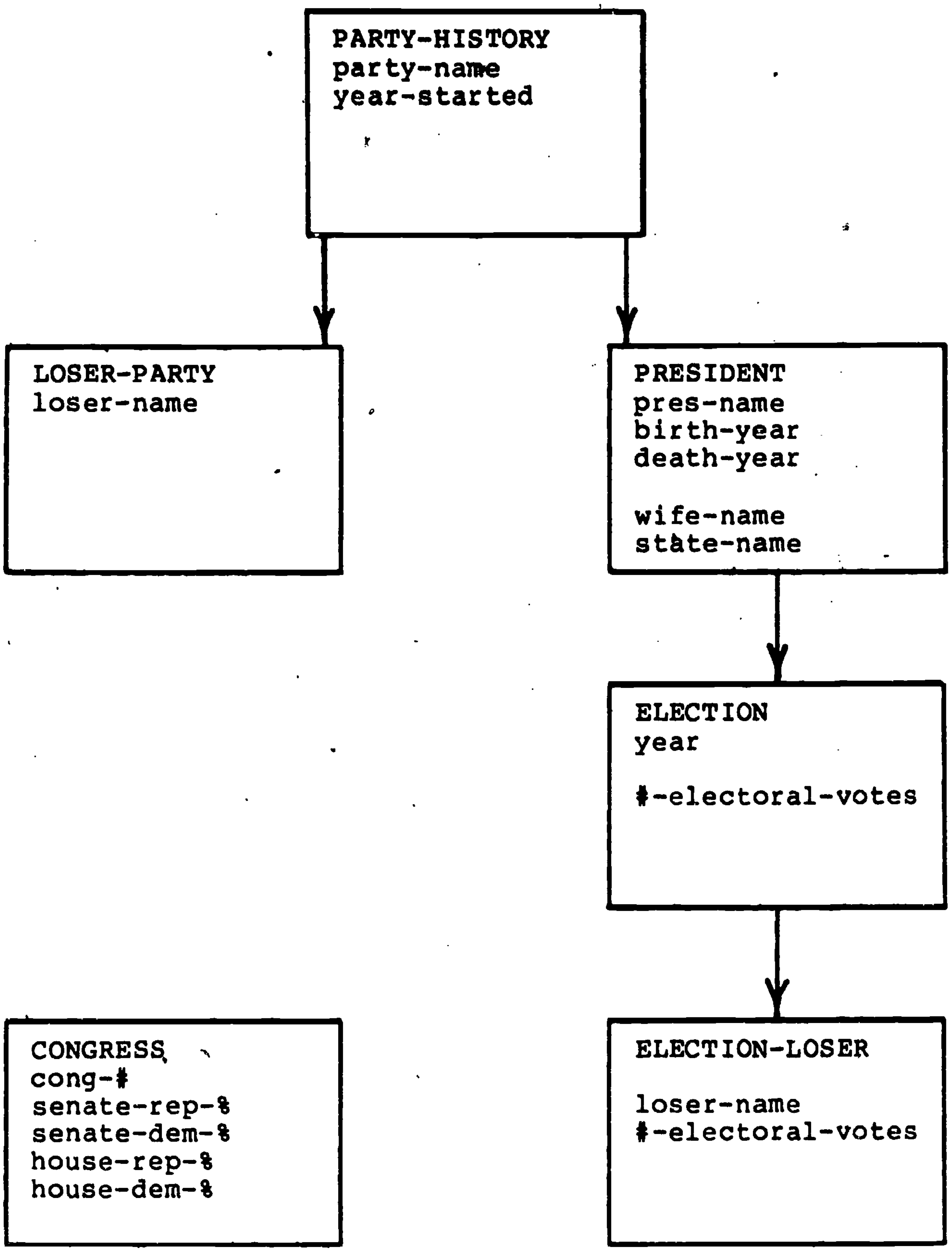


Figure 5.3 - Equivalent of Hierarchical Schema

5.2.2 Transformations for Interchange.

The transformations shown below were used for interchange between DBMS's. These transformations are specific to the presidential schemas. No attempt was made to solve the general problem of mapping between data models.

* Relational to hierarchical, or relational to network: When a record type contains the key of another record type, that attribute is removed from the first record type and a set (or hierarchy) with the first record type as member and the second record type as owner is created. For example, an ELECTION record occurrence loses its pres-name attribute and becomes a member of the WINNER set occurrence (or hierarchy occurrence) whose owner PRESIDENT record occurrence contains the appropriate value for pres-name.

* Hierarchical to relational, or network to relational: A child record type inherits its parent's key. For example, an ELECTION record occurrence acquires, as the value of its pres-name attribute, the value in the pres-name attribute of the PRESIDENT record occurrence that owns the WINNER set occurrence (or hierarchy occurrence).

* Hierarchical to network: The loser-name attribute is removed from ELECTION-LOSER records, and an L-E set is created, as shown in the network schema diagram. Otherwise, the transformation is straightforward.

* Network to hierarchical: An ELECTION-LOSER record acquires, as the value of its loser-name attribute, the value of the loser-name attribute in the parent LOSER-PARTY record, and the L-E set is dropped. Otherwise, the transformation is straightforward.

5.3 Results of Tests

The most important result of the tests was that for each of the three DBMS's, both ICFs are adequate for representation and interchange of data occurrences. No major problems were found in the specifications.

With regard to space required for interchange files, the DDF files were approximately twice as large as the SDICF files for the sample databases. Computer time required was correspondingly somewhat greater for the DDF files. With regard to subjective issues, such as ease of use, time and effort required to develop dump and load programs, etc., both methods were rated acceptable, with no clear preference in forms.

There are still many aspects of the interchange forms that remain to be tested. These include:

* General software that can dump and read the ICF for any database (not just the presidential database).

* The interchange of schemas.

* The representation of arrays and floating point numbers.

* The use of multiple occurring data items and aggregates within a record.

5.4 Conclusions from Tests

The large file size and consequent added processing time of the DDF files seem to be negative considerations with regard to selecting a preferred ICF. However, the DDF specification has since been changed to eliminate the repetition of headers in fixed-length Data Records. These repeated headers account for most of the difference in file size and processing time.

The preliminary experiments have thus shown no clear technical advantage of one form over the other. Tests involving more complicated transformations and the features listed above are appropriate next steps. The DDF form has a clear practical advantage in that it is currently a draft international standard with a number of prototype implementations.

6. Cost/Benefit Study of Database Conversion

In order to determine the cost savings and other benefits that may be realized if standard database translation aids were available, ICST contracted a private firm to perform a preliminary cost/benefit study. The study makes a distinction between conversion, which includes both data and application programs, and database translation, which is only concerned with moving the data itself from one environment to another. The focus of the study is on database translation. The remainder of this chapter presents the major points of that study.

6.1 Reasons for Conversion

Conversion is precipitated by one or more situations occurring because of inherent characteristics of the information management industry. Specific reasons for conversion within private industry and Federal agencies include:

- acquisition of new hardware,
- saturation of existing database management capabilities,
- application migration between existing systems,
- introduction and implementation of new database management capabilities,
- standardization of database management system usage within an organization, and
- interchange of data between different installations.

The most frequent motivation for conversion is caused by an impending upgrade in hardware from one manufacturer's equipment to another's or between different hardware lines of the same vendor that are not software compatible. Because many of the commercially available database management software packages are offered by large mainframe vendors, a move from one hardware vendor to another frequently necessitates a change in database management software. It is also possible that change can occur by simply converting from a DBMS on the current equipment to the same DBMS on the target hardware.

6.2 Expenses of Dat. Translation

As noted above, database conversion most often occurs as part of a change in hardware or DBMS software. Frequently, desired cost/benefit parameters associated with the database translation portion are not segregated from the whole. When specific costs associated with the database translation portion of a DBMS/hardware conversion project are not detailed, valid estimates may be difficult to develop because there are no dependable percentage-of-the-whole rules that apply. It is also hard to delineate database translation costs in those DBMS conversion steps that overlap (i.e., testing and verification).

Case study data emphasizes the fact that the magnitude of database translation costs is directly affected by the existence or non-existence of off-the-shelf translation aids. This was dramatically illustrated by comparing two case studies, a conversion between two vendor-supplied DBMS's accomplished with the use of existing conversion aids, and an in-house DBMS to vendor-supplied DBMS conversion, which required the development of unique conversion aids. The cost ratio between the instance when conversion aids were available and the instance where they were not, was greater than 3 to 1.

6.3 Database Translation Within Federal Agencies

6.3.1 Case Study Results.

Twenty-nine Federal agencies supplied information for this study, through telephone conversations, on-site visits, and written communications. The frequency of database translation among this group can be summarized as follows:

* 20 agencies (69%) have performed at least one non-DBMS-to-DBMS conversion,

* 9 agencies (31%) have performed at least one DBMS-to-DBMS conversion,

* 1 agency has performed a DBMS-to-non-DBMS conversion,

* 10 agencies (34%) are planning a DBMS-to-DBMS conversion within the next 24 months,

* 2 agencies (7%) are planning non-DBMS-to-DBMS conversions within the next 24 months.

The magnitude of the database size in the case study translations ranged from 1,000 records (estimated at about 120,000 bytes) to 5 billion characters. Most translations were of databases in the range of 10 million to 120 million bytes.

The personnel effort involved in the case study translations was dependent on the availability of translation aids and the similarity of database structures in the source and target environments. In the few cases where an aid, usually a software-based "translation" package was available, managers noted that translation was "straight forward and simple". The magnitude of the personnel effort ranged as high as 12 person-years in those instances in which unique translation tools had to be developed and implemented. The lowest level of effort (1/2 person-year) was required in one instance in which a small database was translated between two DBMS's having very similar database structures.

Costs specific to database translation were, for the most part, not delineated from the total costs of the related database management system and/or hardware conversion effort. Expert sources have estimated that database translation costs run between 10 to 65 percent of the total costs of a DBMS conversion. Two separate private industry conversion sources estimated the average low-end cost of a "complete" (data and programs) DBMS conversion at $500,000. One private industry source related cost figures associated with three complete DBMS conversion projects of $2 million, $3 million, and $5 million dollars, respectively. As with the magnitude of personnel effort, costs are directly related to the existence of automated conversion aids that are specific to the source/target environments of the particular conversion. These aids were available for only two of the nine DBMS-to-DBMS conversions included in the case studies. Based on this information, it was estimated that database translation costs for the cases involved ranged from $25,000 to $350,000.

6.3.2 Federal DP Management Comments.

Interviews with Federal DP managers and information from the case studies indicate an expected increase in the frequency of DBMS-to-DBMS conversions. Only two of the contacted agencies were not currently using a DBMS package, and one of these was in a procurement process. All other agencies currently use at least one DBMS package (some installations have four or more different packages).

Most Federal contacts felt that database conversion is imminent for most DBMS user sites in light of the government's hardware replacement programs. This is exemplified by the U.S. Air Force's Phase IV project, recently awarded at $400 million and intended to replace existing hardware at several Air Force data processing centers.

Federal DP managers expressed a strong common concern about the rising frequency of database conversion in an environment where generalized translation aids or standard methodologies are scarce. Currently, as the common approach to database translation, Federal agencies develop customized translation aids for each transfer of data from one environment to another. This approach is inherently expensive: the aids are developed for use only one time and their development costs cannot be amortized. Additionally, reliability is at risk by the greater likelihood of creating errors as the database is passed from one translation aid to another. Tracing back through several passes to a source of error can be an unmanageable task within large translation efforts.

It has been demonstrated that significant gains in staff productivity are possible during database translation when automated translation aids are employed. Users have in several cases indicated that a factor of four productivity gain is possible. Unfortunately, currently available translation aids have limited scope and capabilities. DP managers also indicate that a standard DBMS would considerably ease future DBMS conversions, and that a standard data description (one that was model-independent) would facilitate all translations.

The lack of readily available technical expertise to support in-house translation efforts encourages Federal agencies to contract for DBMS conversion and database translation services. Conversion is now being contracted frequently enough that private industry is beginning to recognize an emerging market. The establishment of new companies and the organization of special divisions within existing companies to specifically address DBMS and database conversion are industry's responses to this growing need.

6.4 Private Industry Database Translation

6.4.1 Case Study Results.

The 17 contacts for the private industry case studies were divided into three classes:

- * Class 1 - DBMS and hardware vendors, many of which provide database conversion services.
- * Class 2 - Companies, other than vendors, that specialize in providing database translation services.
- * Class 3 - Companies that were end users of the products and services offered by Class 1 and Class 2 companies.

Class 1 and Class 2 companies exhibit a bias towards a finite number of source/target conversion environments despite a desire to market their products and services to a general audience. New translation aids seem to appear whenever a significant demand for the specific source/target combination is detected or whenever an end user is willing to pay for what may amount to a one-time solution.

All of the database conversion aids encountered during this study fell into one of two categories:

- * database "unload" translation aids that would input a DBMS-specific source database and output a linearized sequential file, and
- * database "load" translation aids that would input a linearized file and output a DBMS-specific target database.

There were also conversion aids that would allow application programs from the source environment to run in the target environment without modification.

The size of the databases involved in case study translations ranged from 1.2 million characters to 5 billion characters. One company reported that the process of loading a target database of this magnitude from a sequential interface file required 40 hours of CPU time.

The magnitude of the personnel efforts involved in database translations was affected by the availability of database translation aids. Case study examples describe instances where large databases were converted by efforts not exceeding a fraction (.1%) of a man-year when effective translation aids were involved. Instances where translation aids were developed in-line with the translation project required as much as 6 1/2 man-years of effort to translate the databases. Most of this effort was spent on the development, implementation, and testing of unique file translation programs designed to translate various parts of the source database to a sequential interface file or some other form acceptable to the "load" utility of a target DBMS (one instance involved 44 different source database files that had to be translated).

Case study conversion project costs ranged from $86,000 to $8 million, the average was $1.96 million. The studies indicate that database translation represents a low percentage of the total conversion costs when existing automated database conversion aids are available for the specific source/target combination involved. Database translation is a much higher percentage of the total costs when translation aids do not exist and have to be developed for a specific source/target combination. Automated translation aids did not exist for 4 (57%) of the 7 translations included in the case studies.

Database translation costs were conservatively estimated to range from $6,250 to $720,000 for the case studies. These figures do not reflect the costs of translation attempts that were aborted because effective translation aids did not exist or desired translation results were never attained; this event occurred in one of the case studies.

6.4.2 End User Comments.

The trend in private industry seems to favor contracting with vendors or conversion service firms. Of 7 translations reviewed in the case study, 6 were contracted to vendors or service firms having access to translation aids and expertise. All 6 appeared to be "successful" translations. The one attempted in-house translation effort was aborted and followed by a contracted effort.

A number of vendor and conversion consulting firm representatives expressed a strong acceptance and belief in the feasibility of a standard translation interface file format. Several private industry data processing managers noted that standard translation conventions, incorporated by the industry at large, would be adopted into their own information management policies in an attempt to reduce the costs and frustrations associated with DBMS conversion and database translation.

6.5 Evolving Solutions

There are currently two significantly different ways of addressing the challenges of database translation. One supports the development of automated translation tools that are unique to specific source and target environments. The other proposes standardized solutions intended to support all (or most) source and target environments.

6.5.1 Case-Specific Tools.

Vendors and conversion service companies have specific interests in the product(s) with which they are associated. These products may be unique DBMS packages or unique database translation tools. Vendors of DBMS packages offer limited-capability utilities that support translation from selected competitor environments or prior generation environments to the environment of the favored DBMS. Conversion service companies and consulting firms offer translation support for selected DBMS environments, concentrating on the development and use of translation aids specific to the environments in which expertise is maintained.

The set of "solutions" offered by these groups will require constant development and update as new database environments (new hardware and database management software) are introduced.

6.5.2 Standards.

The identification of acceptable standard methodologies, such as a family of standard DBMS's and a standard ICF form, offer a more effective solution to the problems of database translation than the currently used alternative involving the continuous development and/or enhancement of limited-capability translation products that are trying to keep up with new technology. Lower conversion costs insure that users are not "locked in" to one system, and thus have more flexibility as requirements change.

7. Conclusions

This report discusses various aspects of database translation, including: representation of database structures, analysis of potential costs, and feasibility experiments involving candidate interchange forms. We conclude that existence of standard data models and standard representations for database translation could produce substantial benefits to DBMS users in terms of cost savings and increased flexibility. With the proper tools, heretofore unthinkable database translations could be accomplished at minimum cost.

The proposed ANSI and ISO Data Descriptive File (DDF) appears to be the best available candidate for a general interchange form capable of representing nearly all commonly used database data structures. This report provides the additional specifications required to represent the structures of ANSI proposed network and relational data models in terms of the DDF. Acceptance of a general database interchange form will ensure more rapid development of additional conversion tools such as vendor-supplied functions for loading and unloading databases into standard formats and sophisticated model-to-model mapping capabilities. If users anticipate future conversions when selecting systems, then they can require vendors to provide data structures compatible with standard data models and automatic tools for convenient production and transmission of those structures via standard interchange forms.

8. Acknowledgements

We wish to acknowledge the contributions of Gary Sockut, for preliminary drafts of a Guide on Database Translation Approaches, Joseph Collica, Charles Sheppard, James Upperman, and Raymond Youstra, for their work on translation experiments, and Gene Dickamore, for the survey of database translation experiences in the Federal Government and private sector.

A. Appendix A - Mapping Data Structures to an ICF

A.1 Completing The Interchange Form

This section provides a specification for mapping data structures of the proposed American national standard network data model to DDF structures. In particular, specifications are given to produce DDF's for record type occurrences and set type occurrences. If the schema definition is transported separately as a character string conforming to the syntax of the NDL Schema Definition Language, then these two DDF's would be sufficient for transporting database occurrences. In those cases where the schema definition is not parsable at the target environment, we specify four optional DDF's to represent definitions for each of the major NDL structures (record types, components, set types, and members).

In the following subsections, underlined terms refer to definitions given in section 4.3, and terms delimited by double quotation marks refer to definitions given in the proposed DDF standard [ISO83a]. Items enclosed in angle brackets, <...>, refer to syntax specified by the NDL Schema Definition Language [X3H283a]. Specifications for the representation of numeric values as character strings are in American national standard X3.42 [ANSI75]. The DDF and X3.42 specifications, together with the rules and representations of this document, give a complete linear representation of most data structures commonly used in present day commercial database management systems.

A.2 The DDF For Record Type Occurrences

The population of each record type in the database is represented by a single DDF according to the following specifications:

1. The record name of the record type is the "file name" of the DDF and is the name associated with "tag 0...0" in the DDR. The "tag 0...0" does not occur in the DR's.

2. The name of the database in which the record type occurs is the name associated with "tag 0...2" in the DDR. The "tag 0...2" does not occur in the DR's.

3. The key word RECORD identifies the DDF as representing the occurrences of a record type and is the value associated with "tag 0...3" in the DDR. The "tag 0...3" does not occur in the DR's.

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

4. The DR's of a DDF representing a record type are in a one-to-one correspondence with the records of that record type.

5. Each record in the database is identified uniquely by an implementor defined character string value that represents the database key of the given record. This value becomes the "record identifier" for each DR and is associated with "tag 0...1" in the DR "data fields". The key word KEY is the value associated with "tag 0...1" in the DDR.

6. The component names associated with each record type are recorded as values of "tags" in the DDR. Tag numbers for data names must be greater than or equal to ten. The numerical ordering of data names implied by "tag numbers" has no semantic significance.

7. The value of a data item within a database record is recorded in a "data field" of the DR corresponding to that record. The "tag number" of the "data field" is identical to the "tag number" of the "data descriptive field" in the DDR that carries the component name of the given data item.

8. If a record type contains no arrays and if all values are character strings, then the DDF representing the record type population may be a "level 1" DDF. If a record type contains arrays or fixed or floating point numeric values, then the DDF representing that record type must be a "level 2" DDF.

9. A "level 1" DDF is identified as such by the character string "00" in relative positions 10-11 of the DDR leader. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

10. A "level 2" DDF is identified as such by the character string "06" in relative positions 10-11 of the DDR leader. This indicates that the first 6 characters of each "data descriptive field" in the DDR will consist of 2 "structure type code" characters, 2 additional "field control" characters, and 2 "user chosen graphics" characters. The 2 additional "field control" positions are reserved for future standardization and are always filled with zero digits. The "user chosen graphics" characters are used as allowed in the DDF for enhancing readability if Data Descriptive Files are printed out for human editing; these characters have no effect on the DDF itself, do not impact this specification, and could be filled with blank characters. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

11. In a "level 2" DDF representing a record type, the "structure type code" of all "tag numbers" less than ten is the character string "00". For "tag numbers" greater than or equal to ten the "structure type code" conveys the data type for each component name associated with a record type.

12. A component name that references character string values has "00" as its "structure type code" to indicate that these values are transported as character strings. All characters defined in ANS X3.4 are valid characters; additional characters from "extended character sets" can be represented as specified in the DDF. Integrity constraints such as the maximum length of character string values are not directly representable without extending the proposed structures of the DDF.

13. A component name that references fixed point values has either "01" or "02" as its "structure type code". If "01" is specified the values are transported as "implicit-point-representations" without an explicit radix point; if "02" is specified the values are transported as "explicit-point-unscaled representations" where the radix point is part of the data. In either case integrity constraints such as precision and scale factor are not directly representable without extending the structures of the DDF.

14. A component name that references floating point values has "03" as its "structure type code" to indicate that these values are transported as "explicit-point-scaled-representations" and must contain a radix point, an exponent character "E", and an "exrad" value. Integrity constraints such as precision are not directly representable without extending the structures of the DDF.

15. A component name that references an array of values has a "structure type code" to identify these values as elements of a "vector" or an "array". If the array is to be transported as a "vector" then its dimension must be equal to 1. In the "name/label/format" portion of the DDR "data descriptive field", "name" conveys the component name of the array, "label" conveys the dimension and extents of the array, and "format" conveys the "format controls" for the DR "data fields". The "label" is either a "vector label" or a "cartesian label" depending on the "structure type code" chosen. In the proposed DDF standard these labels convey "row and column headings" for appropriate cross sections of the array, from which dimension and extents can be inferred; in this specification for database translation the "vector" or "cartesian label" will consist only of "unsigned-implicit-point-representations" of positive integers to convey the extent in each direction of the array. A "vector label" consists of a single such integer representation while a "cartesian label" consists of a sequence of integer representations separated by the asterisk character "*". The length of the sequence determines the dimension of the array where the i-th integer in the sequence is the extent in the i-th direction. The "format" can be any valid "format control" specified in the DDF. This "format control" always describes a sequence of values, the sequence being the "row-major order" of elements of the "array".

[NOTE: The above rule is a mis-use of the DDF vector and cartesian labels -- Non database implementations of the proposed standard would read the integers as row names and the dimension and extent would be lost. It would be awkward to use the labels correctly to convey dimension and extents -- the DDF structures could be extended so that array dimension and extents can be directly conveyed.]

16. A component name that references an array of character string values has either "10" or "20" as its "structure type code". If "10" is specified the array dimension must be equal to 1. All characters defined in ANS X3.4 are valid characters except that any "delimiters" defined in the "format control", or implied by default, must not appear in the data; additional characters from "extended character sets" can be represented as specified by the DDF. Integrity constraints such as the maximum length of character string values are not directly representable without extending the structures of the DDF.

17. A component name that references an array of fixed point values has one of the character strings "11", "12", "21", or "22" as its "structure type code". If "11" or "12" is specified the array dimension must be equal to 1. If "11" or "21" is specified the values are transported as "implicit-point-representations" without an explicit radix point; if "12" or "22" is specified the values are transported as "explicit-point-unscaled-representations" where the radix point is part of the data. In either case, integrity constraints such as precision and scale factor are not directly representable without extending the structures of the DDF.

18. A component name that references an array of floating point values has either "13" or "23" as its "structure type code". If "13" is specified the array dimension must be equal to 1. The values are transported as "explicit-point-scaled-representations" and must contain a radix point, an exponent character "E", and an "exrad" value. Integrity constraints such as precision are not directly representable without extending the structures of the DDF.

19. If the system record is contained in the database, then the record type whose record name is SYSTEM must be transported as a separate DDF having only one DR with a single "record identifier" in the KEY "data field" and having no other "data fields". The "record identifier" chosen to represent this record will be used in all subsequent references to the system record.

A.3 The DDF For Set Type Occurrences

The population of each set type in the database is represented by a single DDF according to the following specifications:

1. The set name of the set type is the "file name" of the DDF and is the value associated with "tag 0...0" in the DDR. The "tag 0...0" does not occur in the DR´s.

2. The name of the database in which the record type occurs is the name associated with "tag 0...2" in the DDR. The "tag 0...2" does not occur in the DR´s.

3. The key word SET identifies the DDF as representing a set type and is the value of "tag 0...3" in the DDR. The "tag 0...3" does not occur in the DR´s.

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

4. The record name of the owner record type of the set type represented by the DDF is the value associated with "tag 0...4" in the DDR. The "tag 0...4" does not occur in the DR´s. If the set type is a singular set type then the owner record name is the key word SYSTEM.

   [NOTE: This tag may be omitted if the DDF for set definition is included in the database translation.]

   [NOTE: This tag is reserved for future stardardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

5. The record names of the member record types of the set type represented by the DDF are names associated with "tag 0...5" in the DDR. The record names are separated by one or more blank characters. The order in which the record names occur is not significant to this specification. The "tag 0...5" does not occur in the DR´s.

   [NOTE: This tag may be omitted if the DDF for set definition is included in the database translation.]

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

6. The DR´s of a DDF representing a set type are in a one-to-one correspondence with the sets of that set type.

7. The DDF representing a set type is a "level 2" DDF. A "level 2" DDF is identified as such by the character string "06" in relative positions 10-11 of the DDR leader. This indicates that the first 6 characters of each "data descriptive field" in the DDR will consist of 2 "structure type code" characters, 2 additional "field control" characters, and 2 "user chosen graphics" characters. The 2 additional "field control" positions are reserved for future standardization and are always filled with zero digits. The "user chosen graphics" characters are used as allowed in the DDF for enhancing readability if Data Descriptive Files are printed out for human editing; these characters have no effect on the DDF itself, do not impact this specification, and could be filled with blank characters. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

8. In a "level 2" DDF representing a set type, the "structure type code" of all "tag numbers" less than ten is the character string "00".

9. The key word OWNER is the value associated with "tag 0...1" in the DDR. The value of "tag 0..1" in each DR is the implementor defined character string value that serves as a representation for the database key of the owner record of the set represented by that DR.

10. The key word MEMBER is the value associated with "tag 0...10" in the DDR. The "structure type code" of this tag is "10" to indicate that the values conveyed by the DR´s are sequences of character strings. The "vector label" is not used and its contents are undefined. The "format" can be any valid "format control" specified in the DDF except that any "delimiter" specified by an "arbitrary non-numeric delimiter" in the "format control" must not appear in the data described by that "format control".

11. The value of "tag 0...10" in a DR is a sequence of identifiers referencing the database keys of the member records of the set represented by that DR. If the DR represents an empty set; that is, if the set has no member records, then "tag 0...10" does not occur in that DR. The order of the values in the sequence is significant in that it is the order of the member records in the database.

A.4 The Descriptive Files For Database Definition

A network database schema need not be part of the database translation. The necessary structure definitions are contained in the DDF's for record types and set types. However these record and set DDF's do not contain integrity constraints such as conditions for data items or insertion and retention criteria for sets. In addition, the structure definitions are scattered among the various files. DDF's for database definition are optional files that list all structures contained in the database and declare all the integrity constraints associated with each structure as specified by the proposed network model DDL. DDF's for database definition may be included in the database translation to assist in database reconstruction at the receiving end.

The database schema is represented by four Data Descriptive Files, one for each of the major entries in the proposed network model DDL. The four DDF's together contain information equivalent to that contained in a schema defined by the dpANS DDL.

A.5 The DDF For Record Definition

Structures and constraints defined by NDL Schema Definition Language <record type>'s are represented in a DDF according to the following specifications:

1. The key word RECORD is the "file name" of the DDF and is the value associated with "tag 0...0" in the DDR. The "tag 0...0" does not occur in the DR's.

2. The name of the database described by the schema is the name associated with "tag 0...2" in the DDR. The "tag 0...2" does not occur in the DR´s.

3. The key word SCHEMA identifies the DDF as representing a portion of a database schema and is the value associated with "tag 0...3" in the DDR. The "tag 0...3" does not occur in the DR´s.

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

4. The DR´s of a DDF for record definition are in a one-to-one correspondence with the <u>record types</u> of the database.

5. The DDF for record definition is a "level 1" DDF. A "level 1" DDF is identified as such by the character string "00" in relative positions 10-11 of the DDR leader. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

6. The key word NAME is the value associated with "tag 0...1" in the DDR. The value of "tag 0...1" in each DR is the <u>record name</u> of the <u>record type</u> associated with that DR.

7. The key word COMPONENT is the value associated with "tag 0...10" in the DDR. The value of "tag 0...10" in a DR is a list of <u>component names</u> occuring in the <u>record type</u>. The names are separated by one or more blank characters. The order in which the names appear is not significant. If a <u>record type</u> has no <u>data items</u> then "tag 0...10" does not occur in its corresponding DR.

8. The key word CHECK is the value associated with "tag 0...11" in the DDR. The value of "tag 0...11" in a DR is a character string representation of <condition>´s specified by CHECK clauses in the corresponding record type. Multiple <condition>´s are separated by the ampersand character (&). If no CHECK is specified for a given <u>record type</u> then "tag 0...11" does not occur in its corresponding DR.

9. The key word UNIQUE is the value associated with "tag 0...12" in the DDR. The value of "tag 0...12" in a DR is a character string representation of all UNIQUE clauses specified for the corresponding <u>record type</u>.

Multiple clauses are separated by the ampersand character (&), and the component identifiers that make up each clause are separated by one or more blank characters. If no UNIQUE constraints are specified for a given record type then "tag 0...12" does not occur in its corresponding DR.

A.6 The DDF For Data Component Definition

Structures and constraints defined by NDL Schema Definition Language <component type>'s for describing data items and arrays are represented in a DDF according to the following specifications:

1. The key word COMPONENT is the "file name" of the DDF and is the value associated with "tag 0...0" in the DDR. The "tag 0...0" does not occur in the DR's.

2. The name of the database described by the schema is the name associated with "tag 0...2" in the DDR. The "tag 0...2" does not occur in the DR's.

3. The key word SCHEMA identifies the DDF as representing a portion of a database schema and is the value associated with "tag 0...3" in the DDR. The "tag 0...3" does not occur in the DR's.

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

4. The DR's of a DDF for data component definition are in a one-to-one correspondence with all component names, qualified by their host record names, in the database.

5. The DDF for data component definition is a "level 1" DDF. A "level 1" DDF is identified as such by the character string "00" in relative positions 10-11 of the DDR leader. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

6. The key word NAME is the value associated with "tag 0...1" the the DDR. The value of "tag 0...1" in each DR is a character string consisting of a component name followed by a record name. The two names are

separated by one or more blank characters. The component name qualified by its record name identifies the data items described by the DR.

7. The key word TYPE is the value associated with "tag 0...10" in the DDR. The value of "tag 0...10" in each DR is a character string describing the data type of the component name represented by that DR. It consists of one of the key words CHARACTER, BIT, FIXED, or FLOAT, followed by two integers if FIXED is specified and followed by one integer otherwise. The integers represent the length of CHARACTER or BIT types, the precision of a FLOAT type, or the precision and scale factor of a FIXED type. The character string format for an integer is defined in the proposed network model DDL; the tokens are separated by one or more blank characters.

8. The key word CHECK is the value associated with "tag 0...11" in the DDR. The value of "tag 0...11" in a DR is a character string representation of <condition>'s specified by CHECK clauses in the corresponding component definition. Multiple <condition>'s are separated by the ampersand character (&). If no CHECK is specified for a given component then "tag 0...11" does not occur in its corresponding DR.

9. The key word DEFAULT is the value associated with "tag 0...12" in the DDR. The value of "tag 0...12" in a DR is a literal specifying the default value for the component name represented by that DR. The format of the <literal> is as specified in the proposed network model DDL. If a component name does not have an associated DEFAULT then "tag 0...12" does not occur in its corresponding DR.

10. The key word OCCURS is the value associated with "tag 0...13" in the DDR. The value of "tag 0...13" in a DR is a character string representing a sequence of integers that specify the extents in each direction of an array. The character string format for an <integer> is defined in the proposed network model DDL; the tokens are separated by one or more blank characters. If a component name does not reference an array then "tag 0...13" does not occur in its corresponding DR.

A.7 The DDF For Set Definition

Structures and constraints defined by NDL Schema Definition Language <set type>'s are represented in a DDF according to the following specifications:

1. The key word SET is the "file name" of the DDF and is the value associated with "tag 0...0" in the DDR. The "tag 0...0" does not occur in the DR's.

2. The name of the database described by the schema is the name associated with "tag 0...2" in the DDR. The "tag 0...2" does not occur in the DR's.

3. The key word SCHEMA identifies the DDF as representing a portion of a database schema and is the value associated with "tag 0...3" in the DDR. The "tag 0...3" does not occur in the DR's.

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

4. The DR's of a DDF for set definition are in a one-to-one correspondence with the set types of the database.

5. The DDF for set definition is a "level 1" DDF. A "level 1" DDF is identified as such by the character string "00" in relative positions 10-11 of the DDR leader. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

6. The key word NAME is the value associated with "tag 0...1" in the DDR. The value of "tag 0...1" in each DR is the set name of the set type associated with that DR.

7. The key word OWNER is the value associated with "tag 0...10" in the DDR. The value of "tag 0...10" in each DR is the record name of the owner record type of the set type represented by the DR.

8. The key word MEMBER is the value associated with "tag 0...11" in the DDR. The value of "tag 0...11" in each DR is a character string representing the record names of the member record types of the set type represented by that DR. The record names are separated by blank characters. The order in which

the names appear is not significant.

9. The key word ORDER is the value associated with "tag 0...12" in the DDR. The value of "tag 0...12" in each DR is a character string comprised of reserved words and is one of the following: FIRST, LAST, NEXT, PRIOR, DEFAULT, or SORTED.

10. The key word SORTED is the value associated with "tag 0...13" in the DDR. The value of "tag 0...13" in a DR is a character string comprised of either RECORD TYPE or DUPLICATES. The "tag 0...13" occurs in a DR only if the value of "tag 0...12" in that DR is the key word SORTED.

11. The key words RECORD TYPE comprise the value associated with "tag 0...14" in the DDR. The value of "tag 0...14" in a DR is a character string determined by a list of <u>record</u> <u>names</u> as specified by the RECORD TYPE sequence phrase in the proposed network model schema. The <u>record</u> <u>names</u> are separated by one or more blank characters. The "tag 0...14" occurs in a DR only if the value of "tag 0...13" in that DR is the character string RECORD TYPE.

12. The key word DUPLICATES is the value associated with "tag 0...15" in the DDR. The value of "tag 0...15" in a DR is a character string comprised of reserved words and is one of the following: PROHIBITED, FIRST, LAST, or DEFAULT. The "tag 0...15" does not occur in a DR unless the value of "tag 0...13" in that DR is the key word DUPLICATES.

A.8 The DDF For Member Record Definition

Structures and constraints defined by NDL Schema Definition Language <member clause>'s are represented by a DDF according to the following specifications:

1. The key word MEMBER is the "file name" of the DDF and is the value associated with "tag 0...0" in the DDR. The "tag 0...0" does not occur in the DR's.

2. The name of the database described by the schema is the name associated with "tag 0...2" in the DDR. The "tag 0...2" does not occur in the DR's.

3. The key word SCHEMA identifies the DDF as representing a portion of a database schema and is the value associated with "tag 0...3" in the DDR. The "tag 0...3" does not occur in the DR's.

   [NOTE: This tag is reserved for future standardization in the proposed DDF standard -- it could be used for this purpose in database interchange.]

4. The DR's of a DDF for member record definition are in a one-to-one correspondence with record types qualified by the set name of the set type in which they serve as member record types.

5. The DDF for member record definition is a "level 1" DDF. A "level 1" DDF is identified as such by the character string "00" in relative positions 10-11 of the DDR leader. Relative positions 10-11 in the DR leaders are not used to convey any database information; they are filled with the character string "00".

6. The key word NAME is the value associated with "tag 0...1" in the DDR. The value of "tag 0...1" in each DR is a character string consisting of a record name followed by a set name. The two names are separated by one or more blank characters. The record name qualified by its set name identifies the record type as a member of the specified set type.

7. The key word INSERTION is the value associated with "tag 0...10" in the DDR. The value of "tag 0...10" in each DR is one of the key words AUTOMATIC, STRUCTURAL, or MANUAL as specified in the INSERTION clause of the network model schema. If STRUCTURAL is specified then it is followed by the character string specified by the accompanying <structural specification>.

8. The key word RETENTION is the value associated with "tag 0...11" in the DDR. The value of "tag 0...11" in each DR is one of the key words FIXED, MANDATORY, or OPTIONAL as specified in the RETENTION clause of the proposed network model schema.

9. The key word UNIQUE is the value associated with "tag 0...12" in the DDR. The value of "tag 0...12" in a DR is a character string representation of all UNIQUE clauses specified for the corresponding member record type. The clauses are separated by the asterisk character (*) and the component-identifiers that make up each clause are separated by one or more blank

characters. If no UNIQUE clause is specified for a given member then "tag 0...12" does not occur in its corresponding DR.

10. The key word KEY is the value associated with "tag 0...13" in the DDR. The value of "tag 0...13" in a DR is a character string representing the ordering criteria as specified for member records of the member record type associated with the given DR. The format of the ordering criteria is that defined by the ASCENDING and DESCENDING phrases in the proposed network model DDL. If no KEY clause is specified for a given member then "tag 0...13" does not occur in its corresponding DR.

11. The key word DUPLICATES is the value associated with "tag 0...14" in the DDR. The value of "tag 0...14" in a DR is a character string of reserved words and is one of the following: PROHIBITED, FIRST, LAST, or DEFAULT. If no KEY clause is specified for a given member, or if the KEY clause does not contain a <key duplicates> phrase, then "tag 0...14" does not occur in its corresponding DR.

12. The key word CHECK is the value associated with "tag 0...15" in the DDR. The value of "tag 0...15" in a DR is a character string representation of <condition>'s specified by CHECK clauses in the corresponding member record type. Multiple <condition>'s are separated by the ampersand character (&). If no CHECK is specified for a given member then "tag 0...15" does not occur in its corresponding DR.

B. Appendix B - A Detailed Example of the X3L5 DDF

This section gives a detailed example of representing a network database as a character string for database translation, according to the proposed ANSI and ISO Data Descriptive File standard.

B.1 The Example Database

The database has the following complexities:

- Network Structure
- Many-to-many Relationships
- Information Bearing Sets
- NULL or DEFAULT values
- Variable Length Array
- Owner Records Without Members
- Member Records Without Owners
- Records With Identical Values
- Fixed Point Numeric Values

This example is easily expandable to include singular sets, record order keys, or nested repeating groups (e.g. on loser).

The schema of the example database "ELECTION" is depicted in a Bachman diagram in Figure B.1. The database consists of three record types: PRESIDENT, SENATE, and LIAISON, and two set types: SENATESTAFF and PRESAIDES.

According to the specifications contained in Appendix A, the network database is resolved into tabular counterparts and transmitted as a set of tables. Thus each record type and each set type forms a table to be described and transmitted as a data descriptive file. Figure B.2 shows the information contained in the five DDFs for this example. A logical record identifier, the field KEY, has been added to each record. The tags and associated values are displayed above each DDF representation.

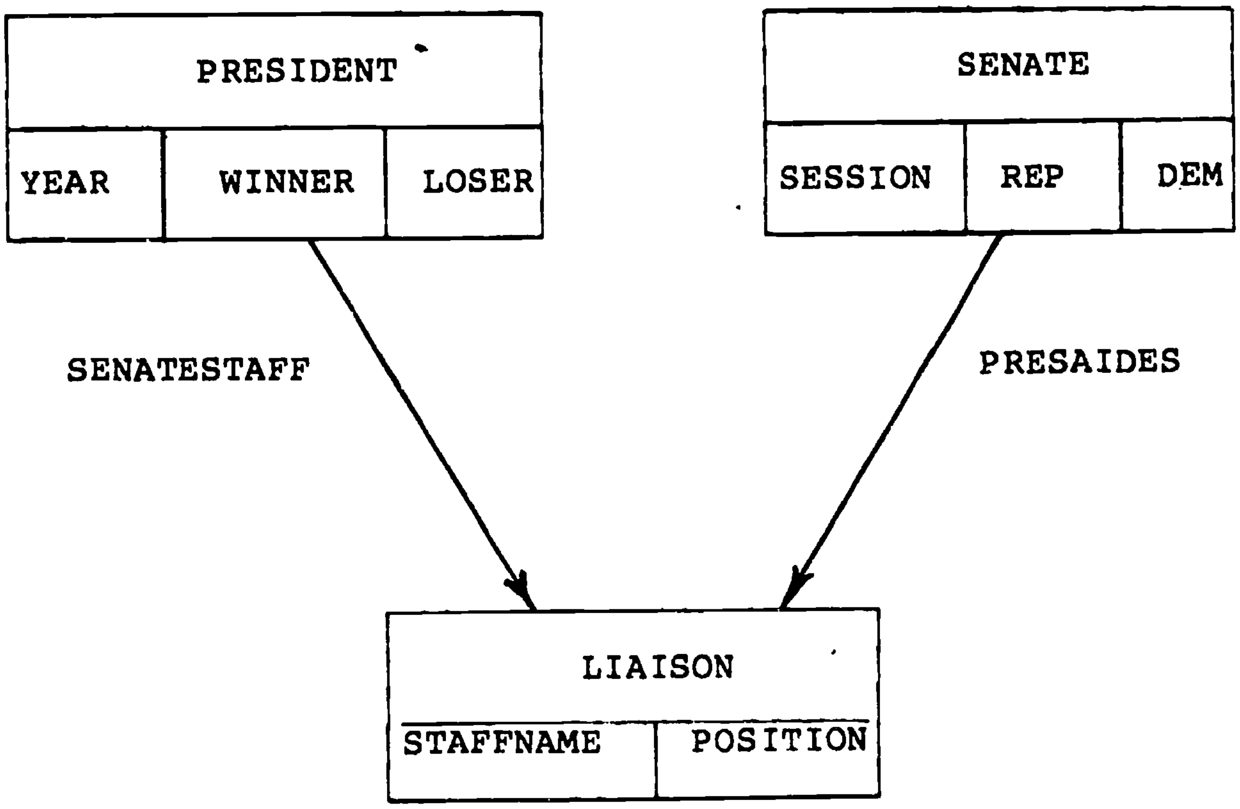


Figure B.1 - Bachman Diagram of ELECTION Database

0: PRESIDENT 2: ELECTION 3: RECORD

KEY	YEAR	WINNER	LOSERS
01	1960	Kennedy	Nixon
02	1963	Johnson	N/A
03	1964	Johnson	Goldwater
04	1968	Nixon	Humphrey Wallace
05	1972	Nixon	McGovern Wallace Paulsen
06	1974	Ford	N/A

0: SENATE 2: ELECTION 3: RECORD

KEY	SESSION	REP	DEM
07	89	32.2	66.8
08	90	36.3	62.6
09	91	43.8	49.4
10	92	45.1	53.2
11	93	43.0	55.8
12	94	-9.9	-9.9

0: LIAISON 2: ELECTION 3: RECORD

KEY	STAFFNAME	POSITION
13	Johnson	VP
14	Sorensen	Lobby
15	Humphrey	VP
16	Humphrey	VP
17	Rostow	Advisor
18	Baker	Advisor
19	Agnew	VP
20	Haig	Chief Staff
21	Agnew	VP
22	Agnew	VP
23	Ford	Appt VP
24	Haldeman	Chief Staff
25	Haig	Advisor
26	Haig	Chief Staff
27	Rockefeller	Appt VP

0:SENATESTAFF 2:ELECTION 3:SET
4: PRESIDENT 5: LIAISON

OWNER	MEMBER
01	13 14
02	
03	15 16 17 18
04	19
05	20 21 22 23 24
06	25 26 27

0:PRESAIDES 2:ELECTION 3:SET
4: SENATE 5: LIAISON

OWNER	MEMBER
07	15
08	16 17 18
09	19
10	20 21
11	22 23 24 25
12	26 27

Figure B.2 - X3L5 DDF Representation of ELECTION Database

B.2 Encoding the Example Database

The remainder of this section contains the encoding of the ELECTION database as a character string according to the specifications of Appendix A. A character-by-character description of the DDF is given below. Examples are from the DDF for the "PRESIDENT" record type (refer to Figure B.3), unless otherwise indicated.

B.2.1 Notation for Example.

1. In the following examples, these printable characters have been substituted for delimiters (the column/row references to ANSI X3.32 for the nonprintable characters are given in parentheses):

;	Field terminator (1/14)
&	Unit (subfield) terminator (1/15)

2. The "space" character is used in the character string representation to separate fields in the Leader, Directory, and Fields of the DDR and DR. This is for editorial purposes only and is not part of the recorded character string. Where the actual "space" character is required to be recorded, it will be represented by the "#" character.

3. CP is an abbreviation for character position.

B.2.2 The Data Descriptive Record.

The Data Descriptive Record (DDR) consists of three areas: Leader, Directory, and Data Descriptive Area.

Description of DDR Leader

The DDR Leader is a 24-character area.

CP 0-4 Record Length - Total length of the DDR in bytes. 0 signifies a length in excess of 99999.

Example: "00181" indicates a DDR length of 181 bytes.

CP 5 Implementation Level
Digit 1, 2, or 3, indicating that the file conforms to a level 1, 2, or 3 file.

Example: "2" indicates a level 2 DDF, meaning that compound data (eg. arrays), as well as elementary character data can be represented.

CP 6 Leader Identifier - Specifies that the record is the DDR and contains the character "L".

CP 7 Inline Code Extension Indicator - Indicates if inline escape sequences are used in data fields to designate extended coded character sets.
a. # means no extensions are used
b. E means that extensions are used

CP 8 Reserved for future standardization

CP 9 Application Indicator - Reserved for future standardization and contains #.

CP 10-11 Field Control Length - Number of bytes of the Field devoted to codes and delimiters.
a. 00 for elementary character data fields
b. 06 for compound data fields
c. 09 if extended code sets used in fields

Example: "06" indicates that compound data fields occur.

CP 12-16 Base Address of Data Descriptive Area - The position of the first data descriptive field. This is equivalent to the combined length in bytes of the leader and directory.

Example: "00081" indicates that the data descriptive area begins at relative position 81.

CP 17-19 Extended Character Set Indicator - Specifies the use of default coded character set extensions.

a) (2/0)(2/0)(2/0) - Only the ISO 646 character set has been designated as the default for the file.
b) (2/0)(2/1)(2/0) - Extended character sets have been designated as the default for one or more data fields.
c) Any other value - An extended character set has been designated as the default for the entire file.

Example: "###" indicates that the default extended character set is in use.

CP 20-23 Entry Map - Specifies the lengths of directory entry subfields. This field has the identical structure in both the DDR and the DR leaders.

CP 20 Size of Field Length - Specifies the size in bytes of the field length subfield of directory entries.

CP 21 Size of Field Position - Specifies the size in bytes of the field position subfield of directory entries.

CP 22 Reserved for future standardization - Reserved for an extended entry map and contains "0".

CP 23 Size of Field Tag - Specifies the size in bytes of the field tag subfield of the directory entries.

Example: "3302" indicates 3 bytes for each of the field length and field position subfields and 2 bytes for the tag subfield.

Description of DDR Directory

Directory entries are 3-tuples, consisting of Field Tag, Field Length, and Field Position.

Field Tag - Identifies a data descriptive field. The meanings of the various tags are given in the previous appendix. Tags 00 through 09 indicate special fields, while tags 10 and above refer to the actual data fields.

Field Length - Specifies the length in bytes of the field to which it corresponds.

Field Position - Specifies the relative position of the first byte in the field referenced by the entry.

Examples: "00 016 000" indicates the file control field (tag 00) which is 16 bytes in length. The associated field is "00 00 ;& PRESIDENT;".

"01 010 016" indicates the record identifier field (tag 01) which is 10 bytes in length and begins at relative position 16. The associated field is "00 00 ;& KEY;".

Note that the directory is terminated by the delimiter ";".

Description of DDR Data Descriptive Area

The data descriptive area of the DDR contains in its data fields information that defines and describes the corresponding data fields of the DR´s (those that have the same tag) and provides control parameters.

For compound data fields, each field is composed of several subfields:

CP 0-3	Field Controls
CP 0	Data Structure a) 0 - Elementary b) 1 - Vector c) 2 - Array
CP 1	Data Type a) 0 - Character b) 1 - Implicit point c) 2 - Explicit point d) 3 - Explicit point, scaled e) 4 - Character mode bit string f) 5 - Bit field g) 6 - Mixed types
CP 2-3	00
CP 4-5	Field and Unit Terminators

For elementary data fields, the above subfields are omitted. For example, the LIAISON Record Type (Figure B.5) consists only of elementary data. The DDF is Level 1, as indicated in the DDR Directory, and the DDR Fields do not contain codes and delimiters.

The Field Controls and Terminators subfields are followed by the Name, Label, and Format Controls subfields. These subfields are delimited by the unit terminator.

Examples: "00 00 ;& WINNER;" corresponds to tag 11 and indicates an elementary character string. WINNER is the data field name.

"01 00 ;& YEAR;" corresponds to tag 10 and indicates an implicit point numeric representation. YEAR is the data field name.

"10 00 ;& LOSERS&3&(A(,));" corresponds to tag 12 and indicates a character vector with the name LOSERS.

The label "3" denotes the number of elements of the vector. The format controls "(A(,))" signify character data, with elements delimited by commas.

The first two fields have special meanings:

"00 00 ;& PRESIDENT;" is the File Control Field and corresponds to tag 00. PRESIDENT is the file name of the DDF (and also the record name for this record type DDF).

"00 00 ;& KEY;" is the Record Identifier Field and corresponds to tag 01. This indicates that the first field in each DR is a record identifier (a database key, in this specification).

B.2.3 The Data Records.

Each Data Record (DR) consists, like the DDR, of three areas: Leader, Directory, and User Data Area. For a DDF that consists of only fixed-length data fields, the leaders and directories of the DR's will be identical. In this case, only the leader and directory of the first DR need be specified.

Many of the DR fields have identical specifications to the corresponding DDR fields, so the reader will be referred to the DDR discussion where applicable.

Description of DR Leader

CP 0-4 Record Length - Total length of the DR in bytes.

Example: "00079" indicates a DR length of 79 bytes.

CP 5 Reserved for future standardization.

CP 6 Leader Identifier - Specifies that the record is a DR.
- a) D means that a leader and directory appear in all subsequent DR's.
- b) R means that the leader and directory of the first DR apply to all subsequent DR's.

Examples: "D" indicates that a leader and directory appear in each DR.

In the DDF for the SENATE Record Type, "R" indicates that a leader and directory appear only in the first DR.

CP 7-11 Reserved for future standardization.

CP 12-16 Base Address of User Data Area - The position of the first user data field of a DR record.

Example: "00057" indicates that the user data begins at relative position 57.

CP 17-19 Reserved for future standardization.

CP 20-23 Entry Map - Specifies the lengths of directory entry subfields. Refer to the description for the DDR.

Description of DR Directory

Directory entries are 3-tuples, consisting of Field Tag, Field Length, and Field Position. The descriptions given for the DDR Directory apply here as well.

Examples: In the first DR of the PRESIDENT Record Type, "01 003 000" indicates the record identifier field (tag 01) which is 3 bytes in length. The associated field is "01;".

Also in this DR, "10 005 003" indicates the data corresponding to tag 10, namely "YEAR". The associated data field is "1960;".

Description of DR User Data Area

The User Data Fields of the DR contain the actual database. For a DDF containing only elementary character data fields, the data fields contain a single string of characters. For a DDF containing compound data fields, the data can be characters, delimiters, and bit strings which conform to the specifications given in the corresponding data descriptive field.

Example: "04;1968;Nixon;Humphrey,Wallace;" indicates a record identifier of 04, a "YEAR" field of 1968, a "WINNER" field of Nixon, and 2 elements in the vector "LOSERS", namely Humphrey and Wallace.

```
Informational                   Recorded on Media
-------------------------------------------------------------------

Data Descpt. Record      |
                         |
DDR Leader               |  00181 2 L # # # 06 00081 ### 3302
DDR Directory            |  00 016 000  01 010 016  02 015 026  03 013 041
                         |  10 011 054  11 013 065  12 022 078;
                         |
DDR Fields               |
Tag 00 (File Ctrl)       |  00 00 ;&  PRESIDENT;
Tag 01 (Record Id)       |  00 00 ;&  KEY;
Tag 02                   |  00 00 ;&  ELECTION;
Tag 03                   |  00 00 ;&  RECORD;
Tag 10                   |  01 00 ;&  YEAR;
Tag 11                   |  00 00 ;&  WINNER;
Tag 12                   |  10 00 ;&  LOSERS&3&(A(,));
                         |
                         |
Data Records             |
                         |
DR Leader                |  00079 # D ##### 00057 ### 3302
DR Directory             |  01 003 000  10 005 003  11 008 008  12 006 016;
DR Fields                |  01;1960;Kennedy;Nixon;
                         |
DR Leader                |  00077 # D ##### 00057 ### 3302
DR Directory             |  01 003 000  10 005 003  11 008 008  12 004 016;
DR Fields                |  02;1963;Johnson;N/A;
                         |
DR Leader                |  00083 # D ##### 00057 ### 3302
DR Directory             |  01 003 000  10 005 003  11 008 008  12 010 016;
DR Fields                |  03;1964;Johnson;Goldwater;
                         |
DR Leader                |  00088 # D ##### 00057 ### 3302
DR Directory             |  01 003 000  10 005 003  11 006 008  12 017 014;
DR Fields                |  04;1968;Nixon;Humphrey,Wallace;
                         |
DR Leader                |  00096 # D ##### 00057 ### 3302
DR Directory             |  01 003 000  10 005 003  11 006 008  12 025 014;
DR Fields                |  05;1972;Nixon;McGovern,Wallace,Paulsen;
                         |
DR Leader                |  00074 # D ##### 00057 ### 3302
DR Directory             |  01 003 000  10 005 003  11 005 008  12 004 013;
DR Fields                |  06;1974;Ford;N/A;
```

Figure B.3 - DDF for the PRESIDENT Record Type

```
Informational                  Recorded on Media
------------------------------------------------------------------
                          |
Data Descpt. Record       |
                          |
DDR Leader                |  00178 2 L # # # 06 00081 ### 3302
DDR Directory             |  00 013 000  01 010 013  02 015 023  03 013 038
                          |  10 014 051  11 017 065  12 015 082;
                          |
DDR Fields                |
Tag 00 (File Ctrl)        |  00 00 ;&  SENATE;
Tag 01 (Record Id)        |  00 00 ;&  KEY;
Tag 02                    |  00 00 ;&  ELECTION;
Tag 03                    |  00 00 ;&  RECORD;
Tag 10                    |  01 00 ;&  SESSION;
Tag 11                    |  02 00 ;&  REPUBLICAN;
Tag 12                    |  02 00 ;&  DEMOCRAT;
                          |
                          |
Data Records              |
                          |
RR Leader                 |  00073 # D ##### 00057 ### 3302
DR Directory              |  01 003 000  10 003 003  11 005 006  12 005 011;
DR Fields                 |  07;89;32.2;66.8;
                          |
DR Fields                 |  08;90;36.3;62.6;
                          |
DR Fields                 |  09;91;43.8;49.4;
                          |
DR Fields                 |  10;92;45.1;53.2;
                          |
DR Fields                 |  11;93;43.0;55.8;
                          |
DR Fields                 |  12;94;-9.9;-9.9;
```

Figure B.4 - DDF for the SENATE Record Type

Informational	Recorded on Media
Data Descpt. Record	
DDR Leader	00120 1 L # # # 00 00073 ### 3302
DDR Directory	00 008 000 01 004 008 02 009 012 03 007 021 10 010 028 11 009 038;
DDR Fields	
Tag 00 (File Ctrl)	LIAISON;
Tag 01 (Record Id)	KEY;
Tag 02	ELECTION;
Tag 03	RECORD;
Tag 10	STAFFNAME;
Tag 11	POSITION;
Data Records	
DR Leader	00063 # D ##### 00049 ### 3302
DR Directory	01 003 000 10 008 003 11 003 011;
DR Fields	13;Johnson;VP;
DR Leader	00067 # D ##### 00049 ### 3302
DR Directory	01 003 000 10 009 003 11 006 012;
DR Fields	14;Sorenson;Lobby;
DR Leader	00064 # D ##### 00049 ### 3302
DR Directory	01 003 000 10 009 003 11 003 012;
DR Fields	15;Humphrey;VP;
DR Leader	00064 # D ##### 00049 ### 3302
DR Directory	01 003 000 10 009 003 11 003 012:
DR Fields	16;Humphrey;VP;
DR Leader	00067 # D ##### 00049 ### 3302
DR Directory	01 003 000 10 007 003 11 008 010;
DR Fields	17;Rostow;Advisor;
DR Leader	00066 # D ##### 00049 ### 3302
DR Directory	01 003 000 10 006 003 11 008 009;
DR Fields	18;Baker;Advisor;

Figure B.5 - DDF for the LIAISON Record Type

```
DR Leader        | 00061 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 006 003  11 003 009;
DR Fields        | 19;Agnew;VP;
                 |
DR Leader        | 00069 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 005 003  11 012 008;
DR Fields        | 20;Haig;Chief Staff;
                 |
DR Leader        | 00061 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 006 003  11 003 009;
DR Fields        | 21;Agnew;VP;
                 |
DR Leader        | 00061 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 006 003  11 003 009;
DR Fields        | 22;Agnew;VP;
                 |
DR Leader        | 00065 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 005 003  11 008 008;
DR Fields        | 23;Ford;Appt VP;
                 |
DR Leader        | 00073 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 009 003  11 012 012;
DR Fields        | 24;Haldeman;Chief Staff;
                 |
DR Leader        | 00065 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 005 003  11 008 008;
DR Fields        | 25;Haig;Advisor;
                 |
DR Leader        | 00069 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 005 003  11 012 008;
DR Fields        | 26;Haig;Chief Staff;
                 |
DR Leader        | 00072 # D ##### 00049 ### 3302
DR Directory     | 01 003 000  10 012 003  11 008 015;
DR Fields        | 27;Rockefeller;Appt VP;
```

Figure B.5 - DDF for the LIAISON Record Type (continued)

Informational	Recorded on Media
Data Descpt. Record	
DDR Leader	00188 2 L # # # 06 00081 ### 3302
DDR Directory	00 018 000 01 012 018 02 015 030 03 010 045
	04 016 055 05 014 071 10 022 085;
DDR Fields	
Tag 00 (File Ctrl)	00 00 ;& SENATESTAFF;
Tag 01 (Record Id)	00 00 ;& OWNER;
Tag 02	00 00 ;& ELECTION;
Tag 03	00 00 ;& SET;
Tag 04	00 00 ;& PRESIDENT;
Tag 05	00 00 ;& LIAISON;
Tag 10	10 00 ;& MEMBER&&(5A(,));
Data Records	
DR Leader	00050 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 006 003;
DP Fields	01;13,14;
DR Leader	00036 # D ##### 00033 ### 3302
DR Directory	01 003 000;
DR Fields	02;
DR Leader	00056 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 012 003;
DR Fields	03;15,16,17,18;
DR Leader	00047 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 003 003;
DR Fields	04;19;
DR Leader	00059 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 015 003;
DR Fields	05;20,21,22,23,24;
DR Leader	00053 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 009 003;
DR Fields	06;25,26,27;

Figure B.6 - DDF for the SENATESTAFF Set Type

Informational	Recorded on Media
Data Descpt. Record	
DDR Leader	00183 2 L # # # 06 00081 ### 3302
DDR Directory	00 018 000 01 012 018 02 015 030 03 010 045 04 013 055 05 014 068 10 022 082;
DDR Fields	
Tag 00 (File Ctrl)	00 00 ;& PRESAIDES;
Tag 01 (Record Id)	00 00 ;& OWNER;
Tag 02	00 00 ;& ELECTION;
Tag 03	00 00 ;& SET;
Tag 04	00 00 ;& SENATE;
Tag 05	00 00 ;& LIAISON;
Tag 10	10 00 ;& MEMBER&&(4A(,));
Data Records	
DR Leader	00047 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 003 003;
DR Fields	07;15;
DR Leader	00053 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 009 003;
DR Fields	08;16,17,18;
DR Leader	00047 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 003 003;
DR Fields	09;19;
DR Leader	00050 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 006 003;
DR Fields	10;20,21;
DR Leader	00056 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 012 003;
DR Fields	11;22,23,24,25;
DR Leader	00050 # D ##### 00041 ### 3302
DR Directory	01 003 000 10 006 003;
DR Fields	12;26,27;

Figure B.7 - DDF for the PRESAIDES Set Type

C. References

[ANSI75] American National Standards Institute, Inc., American National Standard for the Representation of Numeric Values in Character Strings for Information Interchange, ANSI X3.42-1975, August 4, 1975.

[ANSI77] American National Standards Institute, Inc., American National Standard Code for Information Interchange, ANSI X3.4-1977, June 9, 1977.

[ANSI81] American National Standards Institute, Inc. American National Standard for an Initial Graphics Exchange Specification, ANSI Y14.26M, September 1981, ASME, 345 E. 47th St, New York, NY 10017.

[ASTR75] Astrahan, M. M. and D. D. Chamberlin, "Implementation of a Structured English Query Language", in Communications of the ACM 18:10 (1975), pp. 580-587.

[BSI82] British Standards Institution, "The structure and representation of data for interchange at the application level (DIAL)", Draft for Development, Part 1 1981, Part 2 1982, BSI, 2 Park Street, London W1A 2BS, UK.

[CCITT83] International Telephone and Telegraph Consultative Committee, "Message Handling Systems: Presentation Transfer Syntax and Notation", Draft Recommendation X.409, 1983.

[CCSDS83] Consultative Committee for Space Data Systems, "Standard Format Data Units", Draft Report of Panel 2 on Standard Data Interchange Structures, June 1983.

[CHEN76] Chen P.P., "The Entity Relationship Model -- Toward a Unified View of Data", in ACM Transactions on Database Systems (March 1976), pp. 9-36.

[CODD70] Codd, E. F., "A Relational Model for Large Shared Data Banks", in Communications of the ACM 13:6 (1970), pp. 377-387.

[COLL80] Collica, Joseph, Mark Skall, and Gloria Bolotsky, "Conversion of Federal ADP Systems: A Tutorial", NBS SP 500-62, National Bureau of Standards, Washington, D.C., August, 1980.

[DBTG71] "April 1971 Report of the CODASYL Database Task Group (DBTG). Published by ACM, BCS, and IFIP.

[ER79] Proceedings of the International Conference on Entity-Relationship Approach to Systems Analysis and Design. Los Angeles, CA, December 10-12, 1979.

[ER81] Entity-Relationship Approach to Information Modeling and Analysis: Proceedings of the Second International Conference on Entity-Relationship Approach. Washington, DC, October 12-14, 1981.

[FRY81] Fry, J. P., et al., Draft Specification for a Structured Data Interchange Form, rep. NBSIR 81-2315, National Bureau of Standards, July 1981.

[ISO83a] International Organization for Standardization, "Information Processing - Specification for a Data Descriptive File for Information Interchange", ISO DIS 8211, May 1983.

[ISO83b] International Organization for Standardization, "Open Systems Interconnection - File Transfer Overview, Virtual Filestore, File Service Definition, File Protocol Specification", working draft documents ISO/TC97/SC16 N1669, N1670, N1671, N1672.

[JOHN82] Johnson, Rowland, "A Data Model for Integrating Statistical Interpretations", in Proceedings of the First LBL Workshop on Statistical Database Management (March 1982), pp. 176-189.

[KALI81] "DIF: A Format for Data Exchange between Application Programs", Byte, Volume 6, Number 11, November 1981.

[KELL81] Kelly, J.C., Wolf, Robert, Kennicott, Philip and Roger N. Nagel, "A Technical Briefing on the Initial Graphics Exchange Specification (IGES)", rep. NBSIR 81-2297, National Bureau of Standards, July 1981.

[NASA82] National Aeronautics and Space Administration, Procedings of the Workshop on Self-Describing Data Structures, Nick Roussopoulos, editor, University of Maryland, October 27-28, 1982.

[NBS83] National Bureau of Standards, "Message Format for Computer-Based Message Systems", FIPS PUB 98, March 1983.

[SHIP81] Shipman, David, "The Functional Data Model and the Datalanguage DAPLEX", in ACM Transactions on Database Management Systems (March 1981), pp. 140-173.

[SMIT83] Smith, Bradford M., "IGES: A Key to CAD/CAM Systems Integration", in IEEE Computer Graphics and Applications (November 1983), pp. 78-83.

[STON76] Stonebraker, M., E. Wong, P. Kreps, and G. Held, "The Design and Implementation of INGRES", in ACM Transactions on Database Systems 1:3 (1976), pp. 189-222.

[TAYL82] Taylor, Robert W., "Experiences Using Generalized Data Translation Techniques for Database Interchange", North-Holland Publishing Company, Computers & Standards 1 (1982), pp. 111-118.

[X3H283a] "Draft Proposed Network Database Language", ANSC X3H2, document X3H2-83-151, August 1983.

[X3H283b] "Draft Proposed Relational Database Language", ANSC X3H2, document X3H2-83-152, August 1983.

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM.
BIBLIOGRAPHIC DATA SHEET (See instructions)

1. PUBLICATION OR REPORT NO.
NBS SP 500-115

2. Performing Organ. Report No.

3. Publication Date
May 1984

4. TITLE AND SUBTITLE
Computer Science and Technology:
Report on Approaches to Database Translation

5. AUTHOR(S)
Leonard Gallagher and Sandra Salazar

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as item 6.

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 84-601055

□ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

Transporting a database from a source to a target environment has often been an expensive and complex project. In large part this is due to the lack of standards for data models and database interchange forms. This report describes approaches to database translation, discusses candidate interchange forms, and recommends a method for representing the data structures of newly proposed network and relational data models in a form suitable for database interchange. Methods for representing other commonly used database structures in terms of the proposed standard structures show that automated database translation is feasible for most currently installed data models.

A review of various candidate interchange forms shows that the proposed ANSI and ISO Data Descriptive File appears to be the best candidate for character representation of nearly all commonly used database data structures. The form also allows interchange of binary strings in the data fields. Acceptance of standard data models and general database interchange forms could produce substantial benefits to DBMS users in terms of cost savings and increased flexibility. Subsequent vendor supplied, automated tools for reading and writing database structures into standard forms for interchange would make data sharing between non-homogeneous installations a convenient and inexpensive operation.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

ANSI; conversion; data interchange; data models; database; DBMS; data descriptive file; ISO DDF; interchange forms; software standards; translation.

13. AVAILABILITY

XX Unlimited

For Official Distribution, Do Not Release to NTIS

XX Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
87

15. Price

BEST COPY AVAILABLE

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ____

Company ____

Address ____

City ____________________ State ____________ Zip Code __________

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $18; foreign $22.50. Single copy, $5.50 domestic; $6.90 foreign.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the above NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.

Order the following NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.