NBSIR 81-2315

Draft Specification for A Structured Data Interchange Form

May 1981

Issued July 1981

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Center for Programming Science and Technology

NBSIR 81-2315

DRAFT SPECIFICATION FOR A STRUCTURED DATA INTERCAHNGE FORM

James P. Fry

Director
Database Systems Research Group

Wan-Ping Chiang
Donald A. DeSmith
David S. Leder
Duc Kim Nguyen
Robert W. Perreault

The Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48109

May 1981

Issued July 1981

Sponsored by
Center for Programming Science and Technology
Institute for Computer Sciences and Technology
U.S. Department of Commerce
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

PREFACE

Many computer applications of the Federal Government involve extensive data collections where the data has long-term importance or value and usually has been captured at considerable expense from Government operations or from the public. When the computer applications must be transferred from one computer system to another, transferring the data may be of equal or greater concern than transferring the associated computer programs. Federal Information Processing Standards for computer programming languages, accompanied by agency management emphasis on using high-level programming languages, provide an established and recognized approach for reducing the difficulty of program conversion. In the near future, the National Bureau of Standards also plans to develop Federal Information Processing Standards that particularly facilitate the transfer of data files between dissimilar computer systems. These standards will provide means that are independent of specific computer systems or software, for describing data items and structures, and for organizing data values on recording media for interchange. FIPS PUB 79, Magnetic Tape Labels and File Structures, issued in Fiscal Year 1980, represents the first of this group of standards. The following report addresses a follow-on objective, which is a standard means of transferring a database description and its data values on an interchange medium.

This report specifies a structured data interchange form for the transfer of databases between dissimilar computer systems and database management systems. The specification is written in the form of a candidate standard, so that interested parties can review the specific requirements, and not merely the philosophy of the approach. This report is published in order to obtain independent views on the technical merits, efficacy, and potential costs of this avenue for resolving database transfer problems.

At this time, this specification is not proposed as a Federal Information Processing Standard, and the National Bureau of Standards has not determined whether this approach is preferable among alternative candidates for a structured data interchange form. For example, consideration will be given also to a candidate based upon use of the American National Standards Institute BSR X3.87, Data Descriptive File, produced by the Technical Committee X3L5. Assessment of such other candidates, feasibility tests of database transfers using them, and reviews by Federal agencies and the professional community, are all components of the effort remaining before a FIPS is proposed.

This report was developed by the University of Michigan on contract to the National Bureau of Standards, with substantial technical direction provided by the staff of the Data Management and Programming Languages Division, NBS. The Michigan effort was directed by Mr. James Fry, who has previously directed database translation projects, including the development of a generalized database translation software system, and who also has participated for several years in professional studies of database conversion under the auspices of the Conference on Data Systems Languages (CODASYL). Thus, this candidate is based upon significant prior experience with this problem area, as well as substantial technical effort to formulate and refine this specification.

Comments on this report are welcome and are especially invited concerning the expected costs of implementing software to create and to process this structured data interchange form as well as the anticipated benefits which such a standard could bring to Federal agencies, relative to present methods of converting databases and interchanging complex data files. Comments should be directed to:

Dr. Dennis W. Fife, Chief
Data Management and Programming
Languages Division
National Bureau of Standards
Building 225/Room A255
Washington, DC 20234

TABLE OF CONTENTS

DRAFT SPECIFICATION FOR A STRUCTURED DATA INTERCHANGE FORM

James P. Fry
Wan-Ping Chiang
Donald A. DeSmith
David S. Leder
Duc Kim Nguyen
Robert W. Perreault

This report defines the format of a self-describing data interchange file that is media and machine independent, and that would facilitate the transfer of structured data between dissimilar computing systems and software. The specification is intended for interchanging data that has been structured under the discipline of a database management system, although it may also prove applicable to interchanging data that has been structured under other software disciplines, such as the COBOL programming language. The specification covers the three principal types of database management systems, namely the hierarchical, network, and relational types, but it is believed to be sufficiently general to accommodate data from other database management systems. The data interchange file consists not only of the data occurrences or values to be transferred, but also a description of their logical and physical characteristics and the relationships within the data. This minimizes or eliminates the need for additional, non-standardized information transfer in order to successfully perform a database transfer.

The report is written in the form of a candidate standard, so that interested parties can review specific requirements, rather than simply the philosophy or concepts of the approach. However, at this time, this specification is not proposed as a Federal Information Processing Standard. This report is published in order to obtain independent views on the technical merits, efficacy, and potential costs of this avenue for resolving database transfer problems.

Key words: Data description; database management; data conversion; data interchange form; data translation; portability; software conversion.

1. INTRODUCTION

This standard defines a structured data interchange form to facilitate the interchange of data between dissimilar database management systems operating in dissimilar computer hardware and software environments.

1.1 PURPOSE

The structured data interchange form accomplishes the following purposes:

1) Defines a self-describing data interchange form comprised of two sections: a description section and a data section.

2) Defines a character-oriented means of data interchange which is media independent.

3) Permits the interchange of structured data between computer systems utilizing dissimilar database management systems (DBMS).

4) Permits the interchange of structured data between dissimilar computer systems.

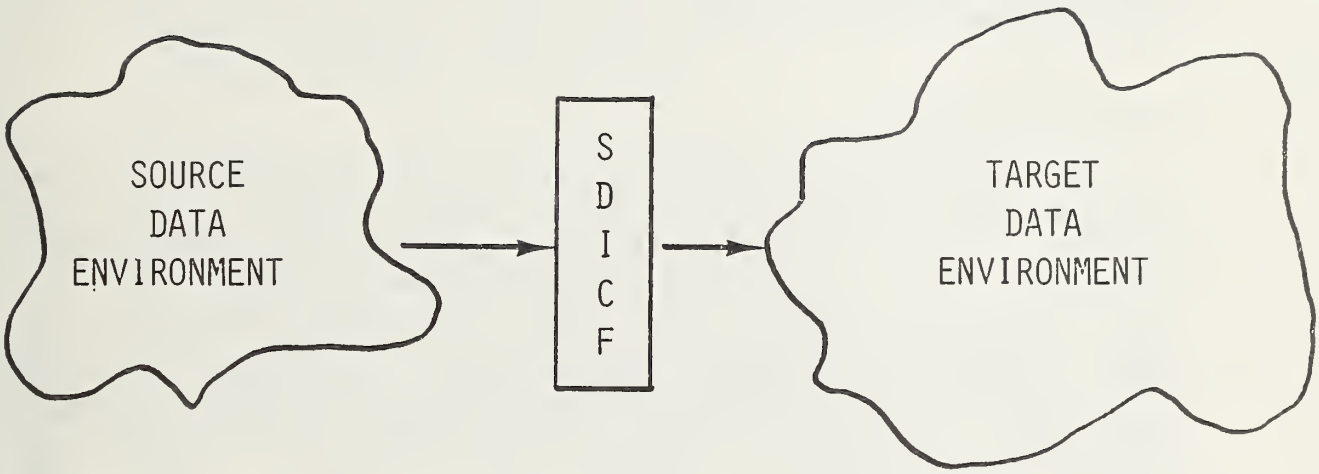
1.2 CONTEXT

The structured data interchange form (SDICF) provides a well-defined, standardized interface between dissimilar source and target environments. The SDICF is utilized within the context of structured data interchange as depicted in Figure 1-1. Structured data existing in the source environment is translated into the format defined by the SDICF. The data is then transferred to the target environment where it is translated into the format required by the target.

Schema information in the source DBMS environment may be lost during transformation to the SDICF due to the lack of an equivalent construct in the SDICF. If the user deems this lost information to be important to the target environment, an additional vehicle outside the SDICF must be utilized to transfer this information to the target environment.

1.3 SCOPE

This standard is intended for use in the interchange of structured data under the discipline of a DBMS although it



CONTEXT OF STRUCTURED DATA INTERCHANGE
FORM UTILIZATION

FIGURE 1-1

could also be utilized to exchange structured data under other disciplines such as the COBOL programming language. This standard is applicable to structured data where the relationships among the components of the data are explicitly described through a data definition facility. It is contended that the SDICF is general enough to accomodate data under the discipline of any type of DBMS (relational, network, or hierarchical). If the relationships among the data components are implicit or made apparent only at the computer instruction level, then the data is considered unstructured and data interchange is beyond the scope of this proposed standard.

1.4 APPLICATION

A SDICF file is defined to be the representation of a database which follows the SDICF specification given in Section 3 of this draft standard. A SDICF file is composed of two sections: a description section and a data section. The description section describes the logical structure of the database, selected storage structure information, and in addition, 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. Hence, a SDICF file is "self-describing"; the description section contains information describing the content of the data section.

A SDICF file is a sequential stream of ASCII characters. The SDICF is media-independent; the media to be used is not specified by this standard. A SDICF file consists of a data section and its associated description section. The simplest case is for a SDICF file to be interchanged as exactly one "file" as defined by the particular medium.

The implementation of this standard will require computer programs which create and process the SDICF file. It is envisioned that if a standard based upon this report is adopted in the Federal Government, producers of database management software or services would be required to provide the necessary software to create and utilize the SDICF. However, the specification of this software will not be covered in the standard and a variety of implementations may be feasible. Appendix A outlines potential approaches for clarification purposes only.

1.5 RELATED STANDARDS

Supporting this standard are the standards relevant to the media utilized. Standards relevant to the encoding of data are also utilized. The following standards are

particularly relevant.

1.5.1 Character Set and Representation (FIPS PUBS 1-1 and 7) (ANSI X3.4-1977)

The interchange data are recorded in their character format, i.e. the interchange form file can be viewed as a continuous stream of characters. The entire SDICF file is encoded in an 8-bit representation utilizing FIPS PUBS 1-1 and 7.

1.5.2 Representation of Numeric Values in Character Strings for Information Interchange (ANSI X3.42-1975)

Attributes with numeric values are represented by a character string as defined in this standard.

1.5.3 Magnetic Tape Labels and File Structure (FIPS PUB 79) (ANSI X3.27-1978)

FIPS PUB 79, Magnetic Tape Labels and File Structure for Information Interchange, details a standard tape labeling and, as a consequence, imposes a physical file structure on the stored data, i.e. how a logical record can be blocked on tape with necessary description to be deblocked.

1.5.4 Transmittal Form for Describing Magnetic Tape File Properties (FIPS PUB 53)

FIPS PUB 53 provides a standard magnetic tape transmittal form that specifies tape file properties, recording system characteristics, and file characteristics with additional space available for supplemental information.

1.6 REFERENCES AND RELATED DOCUMENTS

The following tutorials, guides, and specifications are useful background to the terminology and concepts used in this document.

1.6.1 Database Management Systems (ACM Computing Surveys, Vol. 8, No. 1, March 1976)

This special issue of Computing Surveys contains a series of tutorial articles on database management systems. Included are articles on CODASYL, hierarchical, and relational database systems as well as comparison and survey articles.

1.6.2 Guidelines for Planning and Management of Database Applications (FIPS PUB 77)

FIPS PUB 77 explains alternative software capabilities and recommends development practices for database applications with specific advice on applications planning and management, and on software selection.

1.6.3 CODASYL Database Management Systems (CODASYL Data Description Language Committee, Journal of Development, 1978)

This Journal of Development of the Data Description Language Committee of CODASYL (The Conference on Data Systems Languages) presents the technical specifications of the Schema Data Definition Language (DDL). This version of the specification clarifies and extends the capabilities presented in the 1973 report.

1.6.4 ANSI X3H2 Specifications for a Data Definition Language (ANSI X3H2-17-2 with editorial modifications as of March 2Ø, 1981. Draft Proposed American National Standard for a Data Definition Language for Network Structured Databases)

This draft proposal for an ANSI standard contains the specifications of a language for defining the logical structure of a network database, i.e., the Schema DDL.

1.6.5 Information Interchange Data Descriptive File (ANSI BSR X3.87)

This proposed standard accommodates data elements in a hierarchical interchange structure. It is intended to facilitate the interchange of data under the discipline of a programming language, not data under the discipline of a DBMS.

2. DEFINITIONS

This section defines terms which have a special meaning in the standard. Definition of terms in common use has been considered unnecessary.

aggregate. A named collection of data attributes within an entity.

aggregate unit. A description unit type that describes an aggregate.

area. A logical subdivision of the storage space of a database.

area unit. A description unit type that describes a logical area within the database.

association. A named relationship between two entities. Each association must have one entity declared as owner of the association and one or more entities declared as members of the association. Each occurrence of an association must contain one instance of its owner entity and may contain an arbitrary number of instances of each of its member entities.

association unit. A description unit type that describes an association.

attribute. The smallest unit of named data.

attribute unit. A description unit type that describes an attribute.

character. A letter, digit, or other symbol that is used as part of the organization, control, or representation of data.

CODASYL. Conference on Data Systems Languages. In this proposed standard it is used as a synonym for network type databases.

component. Term used in the structured data interchange form to refer to an attribute or an aggregate.

control records. A record appearing at the beginning of the data section and description section of a structured data interchange form file linking the correct description section with the appropriate data section.

data. Any representations such as characters to which

meaning is or might be assigned.

data environment. The various attributes of a given hardware/software configuration governing how data are stored, retrieved, and manipulated.

data interchange. The exchange of data between a sender computer system (the source) and a recipient computer system (the target).

data unit. The units of the structured data interchange form file containing the data values being exchanged.

data section. The section of the structured data interchange form file containing all the data units of a database being exchanged and a control record.

database administrator. The individual or group responsible for the design and use of a database.

DBA. Acronym for database administrator.

DBMS. Acronym for database management system.

description section. The section of the structured data interchange form file containing all the description units for the database(s) being exchanged and a control record.

description unit. A unit of the structured data interchange form describing the data to be exchanged, utilizing the specifications of the structured data interchange form.

delimiter. A single character that separates and organizes data.

domain. A pool of values from which attribute values are drawn.

domain unit. A description unit type describing a domain.

entity. A collection of attributes of interest used to model records or relations.

entity instance. Collection of data about a particular entity including attribute values.

entity unit. A description unit type describing an entity.

entity type. Same as entity.

environment. Same as data environment.

identifier. An unsigned integer with a maximum length of ten (10) characters that uniquely identifies an individual unit across all units of that particular unit type.

input transformation subsystem. The source data environment's software configuration responsible for constructing the structured data interchange form file from the source stored data.

integer. When appearing in a general format or rule the term "integer" means a decimal integer literal. An integer literal is a string of characters chosen from the digits "0" through "9", the plus sign ("+"), and the minus sign ("-"). A signed integer contains one sign character which must appear as the first character. The plus sign represents a positive value; the minus sign represents a negative value. An unsigned integer contains no sign characters and has a positive value. The decimal point is assumed to be immediately to the right of the rightmost digit.

inter-model. A data interchange where the source and target computer systems utilize different types of database management systems, e.g., network to relational, relational to hierarchical, etc.

intra-model. A data interchange where the source and target computer systems utilize the same type of database management system, e.g., relational to relational, network to network, etc.

ITS. Acronym for input transformation subsystem.

member. One or more of the entities in an association is declared as a member of that association. See association.

name. A sequence of not more than thirty (30) characters selected from the set of alpha-numeric characters and the character "-" that are utilized to identify an individual description unit.

OTS. Acronym for output transformation subsystem.

output transformation subsystem. The target data environment's software configuration that converts the interchange form file to corresponding data in the target data environment.

owner. One of the entities in an association is declared as the owner of that association. See

association.

schema. A conceptual map of the overall logical structure of the database.

SDICF. Acronym for structured data interchange form.

source. The computer system sending the data.

storage structure. The way data are physically stored including pointers, access methods, blocking, etc.

stored data. The file or database as it exists on a physical storage medium.

structured data. Data where the relationships between entities and attributes are explicit through use of a data definition mechanism. This mechanism generally takes the form of the data definition facility of a database management system.

structured data interchange form. A specified format for constructing a structured data interchange form file.

structured data interchange form file. A self-describing sequential character data stream composed of a description section and a data section.

structured data interchange system. The components of the data environment that affect the exchange of data.

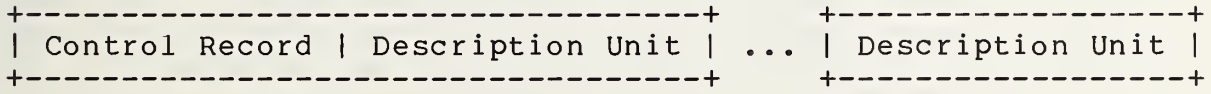
target. The computer system receiving the interchange data.

3. SPECIFICATION OF THE STRUCTURED DATA INTERCHANGE FORM

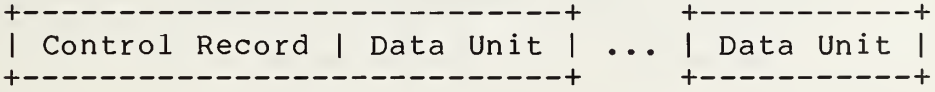
The specification of the structured data interchange form is given in this section. An overview of a SDICF file and the relationship of description sections and data sections is provided in Section 3.1. Section 3.2 defines the notation used throughout the rest of the section. The specifications of the description section and the data section of the SDICF are given in Sections 3.3 and 3.4, respectively.

3.1 GENERAL STRUCTURE

A SDICF file, as defined in Section 1.4, is composed of two sections: a description section and a data section. The description section, further detailed in Section 3.3, consists of one control record followed by several description units. In a similar manner the data section, further detailed in Section 3.4, consists of one control record followed by a number of data units. Figure 3-1 depicts the generalized schematic of a SDICF file.



(a) Description Section



(b) Data Section

Generalized Schematic of a SDICF File

FIGURE 3-1

While each section of a SDICF file is a contiguous set of units, the description section and data section need not be transmitted together. The control record of each section

is used to support separate transfer of a data section and its associated description section by means of an internal identifier linking the two sections. Control records also support the interchange of multiple instances of a source database conforming to a particular description section. For example, a company may send an entire SDICF file (both description and data sections) at one point in time and then periodically send an updated version of the database, i.e., a new data section which is still correctly described by the initial description section. In this case, one description section and many data sections form many SDICF files.

3.2 NOTATION AND SPECIAL CHARACTERS

The notation used in all formats and the rules which apply to all formats in the following specification are:

* The elements which make up a clause consist of upper case words, lower case words, special symbols and special delimiter characters.

* All underlined upper case words or letters are required when the clause is used.

* Upper case words or letters which are not underlined are included only as an aid to comprehension and should not be included in an instance of an SDICF file.

* Lower case words are generic terms which must be replaced by appropriate values.

    * The term 'name' appearing in a general format or rule means a sequence of not more than thirty (30) characters. Each character is selected from the set of alpha-numeric characters and the character '-'.

    * The term 'integer' appearing in a general format or rule means a decimal integer literal. An integer literal is a string of characters selected from the digits '0' through '9', the plus sign ('+'), and the minus sign ('-'). A signed integer contains one sign character which must appear as the first character. The plus sign represents a positive value and the minus sign represents a negative value. An unsigned integer contains no sign character and has a positive value. The decimal point is assumed to be immediately to the right of the rightmost digit.

    * The term 'identifier' appearing in a general

format or rule means an unsigned integer of not more than ten (10) characters.

* The meaning of enclosing a portion of a general format in special symbols *is as follows:*

  $[\ ]_m^n$ at least "m" occurrences, at most "n" occurrences (m>=Ø , n>Ø , n>=m , both integers). An asterisk ('*') as a superscript indicates a variable number of occurrences as the maximum.

  { } exactly one occurrence (Only one of the choices in the meta brackets is used. It is illegal to omit all of the options.)

  < > a comment describing the clause but not specifying the syntax.

Table 3-1 lists the special characters used and their meanings as defined in the SDICF. These special characters are required where they appear in the general format. Additionally, one or more consecutive spaces act as a separator. A space or spaces may optionally appear in addition to (i.e. preceding or following) any other terminator or separator except in the ATTRIBUTE VALUE clause of a data unit (Section 3.4.2) where spaces are considered as part of the attribute value.

;	=	field terminator
@	=	unit terminator
#	=	section terminator
?	=	literal next character
,	=	multiple-occurring identifier separator

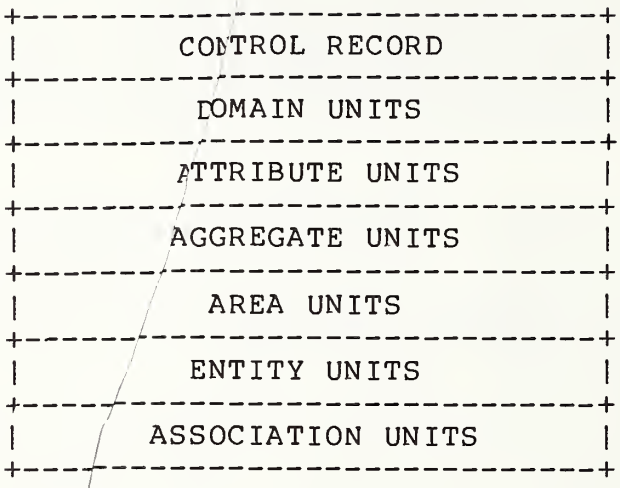
Delimiters and their Uses

TABLE 3-1

3.3 DESCRIPTION SECTION SPECIFICATION

The description section of a SDICF file contains one control record and up to six types of description units: (i) Domain units; (ii) Attribute units; (iii) Aggregate units; (iv) Area units; (v) Entity units; and (vi) Association units. Figure 3-2 provides an overview of a SDICF description section. These six types of description units are sufficient to represent the information content of the structured data in the SDICF. Only the description units

which are necessary to describe the data are included in a SDICF file.



Overview of Description Section

FIGURE 3-2

A more formal specification of the description section is given in Figure 3-3. The description units are grouped by type and are in the order indicated. A description section begins with exactly one control record. Description units for domains, aggregates, and areas are not always necessary. A SDICF file must contain description units for attributes, entities, and associations. Guidelines for the applicability of each type of description unit for different categories of data interchange are given in Section 4. Specifications for the control record and each of the six types of description units are given in Sections 3.3.1 through 3.3.7.

[<Control Record>]$_{1}^{1}$ [<Domain Unit>]$_{\emptyset}^{*}$

[<Attribute Unit>]$_{1}^{*}$ [<Aggregate Unit>]$_{\emptyset}^{*}$

[<Area Unit>]$_{\emptyset}^{*}$ [<Entity Unit>]$_{1}^{*}$ [<Association Unit>]$_{1}^{*}$ #

Description Section Specification

FIGURE 3-3

3.3.1 Control Record

The description section begins with a control record which contains the schema identifier, schema name, and the date created. The schema identifier is the mechanism to associate a data section with the correct description section.

General Format

DESCRIPTION ; SCHEMA IDENTIFIER IS schema-id# ;

NAME IS schema-name ; DATE IS date @

Where:

schema-id#	is the identifier of the schema.
schema-name	is the name of the schema.
date	is the date the description section is defined and is in the form YYMMDD.

Rules and Explanation

1. The DESCRIPTION clause identifies the control record and what follows as a description section.
2. The SCHEMA IDENTIFIER clause uniquely identifies a description section and is used to associate data section(s) with the description section.

3. The NAME clause assigns schema-name as the name of the description section. Names need not be unique whereas schema identifiers are.

3.3.2 Domain Units

A Domain defines the set of values from which the value of an attribute is drawn. Domain description units record the characteristics of name, identifier, and value types.

General Format

DOMAIN IDENTIFIER IS domain-id# ; NAME IS domain-name ;

$$\text{TYPE IS } \left\{ \begin{array}{ll} \left\{ \begin{array}{l} \text{CHARACTER} \\ \text{BIT} \end{array} \right\} & [\text{integer-1}]^{1}_{\emptyset} \\ \left\{ \begin{array}{l} \text{FIXED} \\ \text{FLOAT} \end{array} \right\} & \text{integer-2}[,\text{integer-3}]^{1}_{\emptyset} \end{array} \right\} @$$

Where:

domain-id#	is the identifier of the domain.
domain-name	is the name of the domain.
integer-1	is an unsigned integer specifying the length of a character or bit string. The value must be greater than zero.
integer-2	is an unsigned integer specifying the precision of a numeric attribute. The value must be greater than zero.
integer-3	is an integer specifying the scale factor of a numeric attribute.

Rules and Explanation

1. The DOMAIN IDENTIFIER clause uniquely identifies a domain. Each domain-id# in a description section must be unique.

2. The NAME clause assigns domain-name as the name of the domain.

3. The TYPE clause specifies the type of the attributes drawn from this domain. The representation of a numeric

attribute value in the SDICF is defined in Section 3.4.2, Data Units.

4. In the TYPE CHARACTER clause integer-1 specifies the maximum length of the character string which can be assigned to the attribute. If integer-1 is not specified, then integer-1 is assumed to be 1.

5. In the TYPE BIT clause integer-1 specifies the maximum length of the bit string which can be assigned to the attribute. If integer-1 is not specified, then integer-1 is assumed to be 1.

6. The TYPE FIXED clause specifies a fixed-point number where the precision is given by integer-2 and integer-3. Integer-2 specifies the total number of significant digits and integer-3 is the scale factor which specifies the decimal point as follows:

   - If integer-3 > Ø, the decimal point is assumed to be located integer-3 places to the left of the rightmost actual digit.

   - If integer-3 < Ø, the decimal point is assumed to be located integer-3 places to the right of the rightmost actual digit.

   - If integer-3 = Ø, the decimal point is assumed to be located immediately to the right of the rightmost digit.

   If FIXED is specified but integer-3 is omitted the value of integer-3 is assumed to be zero.

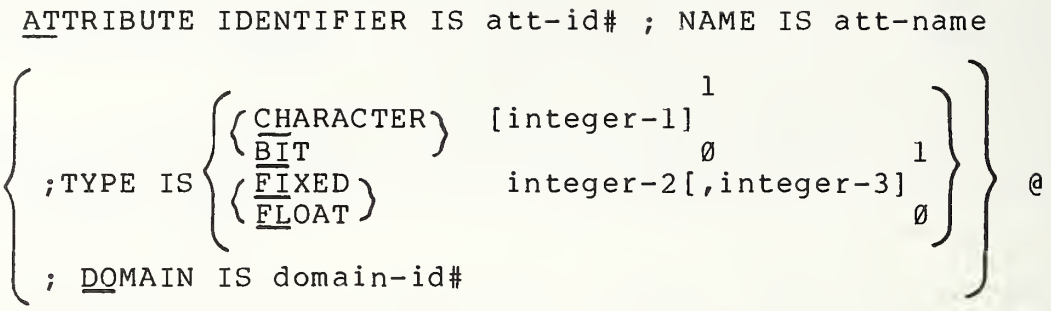
7. The TYPE FLOAT clause defines a real number represented as a floating point number, which consists of a significand and an exrad. The significand gives the significant digits of the number and the exrad specifies the magnitude of the significand as an integral power of ten. The precision is specified by integer-2 which is the minimum number of digits to be maintained for the significand. Integer-3 must not be specified.

8. Domains are definable only for relational databases.

3.3.3 Attribute Units

An Attribute is the smallest named piece of data which is accessable by the DBMS. Attribute description units record the characteristics of name, identifier, domain, and type.

General Format



Where:

att-id#	is the identifier of the attribute.
att-name	is the name of the attribute.
domain-id#	is the identifier of the domain over which the attribute is defined.
integer-1	is an unsigned integer specifying the length of a character or bit string.
integer-2	is an unsigned integer specifying the precision of a numeric attribute.
integer-3	is an integer specifying the scale factor of a numeric attribute.

Rules and Explanation

1. The ATTRIBUTE IDENTIFIER clause uniquely identifies an attribute. Each att-id# in a SDICF file must be unique.

2. The NAME clause assigns att-name as the name of the attribute.

3. The DOMAIN clause may be used only for relational databases and requires that the domain be previously defined by a Domain unit. Multiple attributes can be defined with the same domain.

4. The TYPE clause is used for all attributes not defined by a domain. The rules for its usage appear in Section 3.3.2, Domain Units.

5. Attribute units define what are termed fields, data

items, elements, etc. in DBMS's. An attribute corresponds to a CODASYL data item.

3.3.4 Aggregate Units

An Aggregate is a named collection of attribute values within an entity. There may be multiple sets of values for the aggregate. Aggregate description units record the characteristics of name, identifier, value set occurrences, and the component attributes or aggregates.

General Format

AGGREGATE IDENTIFIER IS agg-id#-1 ; NAME IS agg-name

; OCCURS { integer-1 / AT att-id#-1 } TIMES

; COMPONENTS ARE { AT att-id#-2 / AG agg-id#-2 } [[{ ,AT att-id#-3 / ,AG agg-id#-3 }]* ø @

Where:

agg-id#-1	is the identifier of the aggregate.
agg-name	is the name of the aggregate.
integer-1	is an integer specifying the number of times the aggregate repeats in the data. The value of integer-1 must be greater than zero.
att-id#-1	is the identifier of an attribute whose value specifies the number of times the aggregate repeats.
att-id#-2	is the identifier of a previously defined attribute which is one component of the aggregate.
agg-id#-2	is the identifier of a previously defined aggregate which is one component of the described aggregate.

att-id#-3	is the identifier of a previously defined attribute which is one component of the aggregate.
agg-id#-3	is the identifier of a previously defined aggregate which is one component of the described aggregate.

Rules and Explanation

1. The AGGREGATE IDENTIFIER clause uniquely identifies an aggregate. Each agg-id#-1 in a SDICF file must be unique.

2. The NAME clause assigns agg-name as the name of the aggregate.

3. The OCCURS clause specifies the number of times the aggregate repeats. The use of integer indicates that the number of occurrences is fixed for all entity instances. The use of att-id#-1 indicates a variable number of occurrences. For a given entity type, the number of occurrences of the aggregate is specified by the value of the attribute (defined by att-id#) for that entity instance. The value must be a positive integer or zero.

4. The attribute referred to by att-id#-1 must be defined as an integer and must be part of the same entity as the subject aggregate.

5. Att-id#-1 can be used only if the aggregate being defined is not the component of another aggregate.

6. Att-id#-1 cannot refer to an attribute which is part of an aggregate defined with an OCCURS att-id#-1 clause.

7. If an aggregate is composed of exactly one attribute and no other aggregates then a vector results; otherwise a repeating group results.

8. The components can be any combination of attributes and other aggregates. If one aggregate is defined as a component of another, the two must be distinct (i.e., non-recursive).

9. An aggregate corresponds to a CODASYL data aggregate.

3.3.5 Area Units

An Area is a named collection of entities which designates a logical storage area for occurrences of these entities. Area description units record the characteristics of name and identifier.

General Format

AREA IDENTIFIER IS area-id# ; NAME IS area-name @

Where:

area-id# is the identifier of the area.

area-name is the name of the area.

Rules and Explanation

1. The AREA IDENTIFIER clause uniquely identifies an area. Each area-id# in a SDICF file must be unique.

2. The NAME clause assigns area-name as the name of the area.

3. An Area unit specifies a logical storage area for record types for a CODASYL-like network database. Use of it for other purposes is not foreseen.

3.3.6 Entity Units

An Entity is a named collection of attributes and aggregates. Entity description units record the characteristics of name, identifier, area, access information, components, primary keys, indexes, and associations.

General Format

```
ENTITY IDENTIFIER IS entity-id#

    ;NAME IS entity-name

    [;AREA IS area-id# ]*Ø

    [;LOCATION MODE IS { CALC ON att-id#-1
                       { DIRECT ON att-id#-2
                       { VIA SET assoc-id#-1
                       { SYSTEM DEFAULT      ]1Ø

        [;COMPONENTS ARE { AT att-id#-3 }]*Ø
                         { AG agg-id#-1 }

    [;PRIMARY KEY IS att-id#-4 [,att-id#-5]*Ø ]1Ø

    [;INDEXED BY att-id#-6 [,att-id#-7]*Ø ]*Ø

;IS IN ASSOCIATION assoc-id#-2 [,assoc-id#-3]*Ø  @
```

Where:

entity-id#	is the identifier of the entity.
entity-name	is the name of the entity.
area-id#	is the identifier of a previously defined area.
att-id#-1	is the identifier of the attribute type whose attribute values will be utilized to calculate the storage location of

individual entity instances.

att-id#-2 — is the identifier of the attribute type from which the attribute values will be utilized directly by the DBMS to assign a database key.

assoc-id#-1 — is the identifier of the association in which the entity participates. For this entity type, the entity instances will be stored as closely as possible to the owner entity instances.

att-id#-3 — is the identifier of an attribute type that is part of the entity.

agg-id#-1 — is the identifier of an aggregate type that is part of the entity.

att-id#-4 — is the identifier of an attribute that partially or completely defines the primary key.

att-id#-5 — is the identifier of an attribute type that partially defines the primary key.

att-id#-6 — is the identifier of an attribute that defines a secondary index for the entity.

att-id#-7 — is the identifier of an attribute that defines a secondary index for the entity.

assoc-id#-2 — is the identifier of an association in which the entity participates either as an owner or member.

assoc-id#-3 — is the identifier of an association in which the entity participates either as an owner or member.

Rules and Explanation

1. The ENTITY IDENTIFIER clause uniquely identifies an entity. Each entity-id# in a SDICF file must be unique.

2. The NAME clause assigns the entity-name as the name of the entity.

3. The AREA clause, used only for CODASYL-like DBMS's, specifies the area(s) in which instances of this entity may be stored. If multiple area-id#'s are specified in

one entity unit each must be unique. The areas specified must be previously defined in an Area unit.

4. The LOCATION MODE clause is a storage structure concept generally relevant to only CODASYL-like DBMS's and has the meaning specified in CODASYL reports. The attributes specified by att-id#-1 and att-id#-2 must be attributes previously defined in an Attribute unit. The association specified by assoc-id#-1 must be defined in an Association unit which follows and the entity must be defined as either a member or an owner of that association.

5. The COMPONENTS clause defines the attributes and aggregates of which the entity is composed. Zero or more components may be specified. The attributes and aggregates specified by att-id#-3 and agg-id#-1, respectively, must be previously defined in Attribute and Aggregate units. The order in which the components are defined must be the same as the order in which their values are given in the Data units (see Section 3.4.2).

6. The PRIMARY KEY clause specifies the attribute(s) of which the primary key is composed. The order in which attribute identifiers are given defines the order in which they are used as the primary key. The attributes specified by att-id#-4 and att-id#-5 must be defined to be in this entity by the COMPONENTS clause.

7. The INDEXED clause specifies one or more attributes upon which a secondary index is based. The attributes specified by att-id#-6 and att-id#-7 must be previously defined to be in this entity by the COMPONENTS clause.

8. The ASSOCIATION clause specifies the association(s) in which the entity participates as either a member or an owner. The associations specified by assoc-id#-2 and assoc-id#-3 must be defined in Association units which follow .

9. Entity units generally define what are termed records, segments, data records, etc. An entity corresponds to a CODASYL record type.

3.3.7 Association Units

The Association is a named collection of entities in which one entity serves as the owner and one or more entities are members of the association.

General Format

ASSOCIATION IDENTIFIER IS assoc-id# ; NAME IS assoc-name

; OWNER IS { entity-id#-1 / SYSTEM } [; MEMBER IS entity-id#-2]*1

[; ORDER IS { ASCENDING / DESCENDING } att-id#]*Ø @

Where:

assoc-id#	is the identifier of the association.
assoc-name	is the name of the association.
entity-id#-1	is the identifier of the entity type which is the owner of the association.
entity-id#-2	is the identifier of the entity type which is the member of the association.
att-id#	is the identifier of the attribute by which the member entity instances of the association are ordered as specified.

Rules and Explanation

1. The ASSOCIATION IDENTIFIER clause uniquely identifies an association. Each assoc-id# in a SDICF file must be unique.

2. The NAME clause assigns assoc-name as the name of the association. The NAME clause is required so if a name is not explicitly used in the source database one must be generated for the SDICF file.

3. The OWNER clause specifies the owner entity of the association, if one exists, and specifies that an entity is system-owned otherwise. All entities of a relational database would be system owned. The root entity of a hierarchical database would also be system owned. The entity specified by entity-id#-1 must be previously defined in an Entity unit.

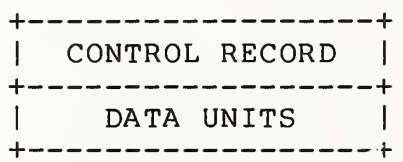
4. The MEMBER clause specifies the member entity(ies) of the association. All entities defined by Entity units must be defined as members of at least one association. If an entity is not explicitly related to another entity then it must be defined as the member of an association whose owner is SYSTEM.

5. The ORDER clause defines that the order of the entity specified by entity-id#-2 is based on the value of the attribute specified by att-id#, which must be a component of that entity. If the association is ordered on multiple attributes the order in which the attributes are given specifies the priority of the ordering (i.e. ordered by the first attribute, then by the second attribute within the first, etc.).

6. Associations are used to model CODASYL set-types, the parent-child relationship of hierarchical databases, and to provide a structuring mechanism where none explictly exists (e.g., in a relational database).

3.4 DATA SECTION SPECIFICATION

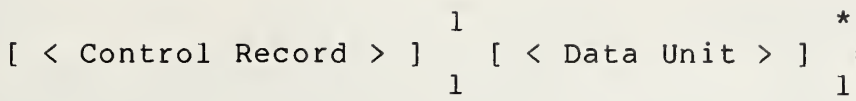
The data section of a SDICF file is composed of one control record and all the data units being interchanged. The control record indicates the description section with which the data section is associated (cf. Section 3.1). The data units contain all of the data (i.e., entity instances) being interchanged as well as the pointers necessary to reconstruct the database in the target environment. An overview of a data section is given in Figure 3-4.



Overview of Data Section

FIGURE 3-4

A more formal specification of the data section is given in Figure 3-5. A data section begins with exactly one control record. The control record is followed by any number of data units. Each entity instance in the source database is represented by one data unit. The specification of the control record is given in Section 3.4.1 while data units are specified in Section 3.4.2.



Data Section Specification

FIGURE 3-5

3.4.1 Control Record

The control record of a data section contains the information necessary to link a data section with its appropriate description section. It is almost identical to the control record of a description section (specified in Section 3.3.1).

General Format

DATA ; SCHEMA IDENTIFIER IS schema-id# ;

NAME IS schema-name ; DATE IS date @

Where:

schema-id#	is the identifier of the schema.
schema-name	is the name of the schema.
date	is the date the data section is created and is of the form YYMMDD.

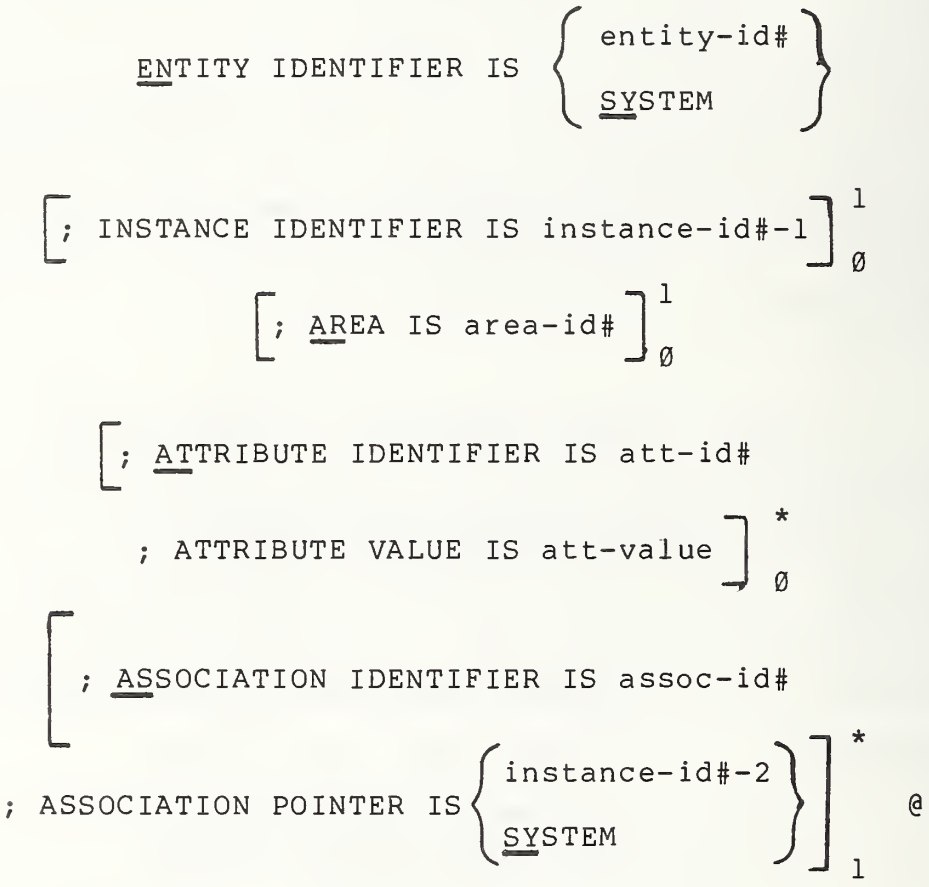
Rules and Explanation

1. The DATA clause identifies the control record and what follows as a data section.
2. The SCHEMA IDENTIFIER clause uniquely identifies the description section which describes this data section.
3. The NAME clause assigns schema-name as the name of the schema. This name should match that given in the associated description section.
4. The DATE clause is provided as a means of distinguishing sections associated with the same description section.

3.4.2 Data Units

The Data units contain the actual data being interchanged. Each Data unit contains the data of one entity instance plus other information needed to maintain database structure.

General Format



Where:

entity-id#	is the identifier of the entity to which the data unit corresponds.
instance-id#-1	is the entity instance identifier for the entity instance represented by this data unit.
area-id#	is the identifier of the area with which the entity instance represented by this data unit is associated.

att-id#	is the identifier of the attribute whose value is given by the following attribute value.
att-value	is the value of the attribute whose identifier is immediately preceding.
assoc-id#	is the identifier of the association whose association pointer is given by the following value.
instance-id#-2	is the identifier of an entity instance with which this entity instance participates in the association whose identifier is immediately preceding.

Rules and Explanation

1. The ENTITY IDENTIFIER clause defines the entity with which a Data unit is associated. The entity-id# must be defined in an Entity unit of the associated description section. Exactly one Data unit in a data section must have SYSTEM as its entity identifier. This Data unit represents the SYSTEM as the owner of associations. Only the ENTITY IDENTIFIER, ASSOCIATION IDENTIFIER, and ASSOCIATION POINTER clauses are permitted for this Data unit.

2. The INSTANCE IDENTIFIER clause uniquely identifies the entity instance represented by a Data unit. The instance-id#-1 must be unique across all Data units in a data section. This clause is required for all Data units except the one whose entity identifier is SYSTEM.

3. The AREA clause defines the area with which the entity instance represented by this data unit is associated. The area-id# must be defined in an Area unit of the associated description section. An area-id# must be specified if areas have been defined in the related description section.

4. The ATTRIBUTE IDENTIFIER and ATTRIBUTE VALUE clauses form an ordered pair which is used to specify the values of all attributes in the entity instance. All attributes defined for this entity must be included for integrity reasons. A null attribute value is indicated by a null string (i.e., two consecutive delimiters).

5. Att-value is a string of ASCII characters representing an attribute's value. Every attribute value has a type which is defined by the TYPE clause in the Attribute unit or Domain unit defining the attribute.

A. If the type is CHARACTER then any ASCII character sequence of length less than or equal to the maximum length is legal. If the character sequence is less than the maximum length, the character sequence is aligned at the left most point and spaces are assumed to fill the unspecified positions to the right of the sequence. Any spaces within the delimiters are characters rather than separators.

B. If the type is BIT then only the ASCII characters "Ø" and "1" are permitted. The value of a bit string att-val is the value formed by replacing each "Ø" character with the bit Ø and each "1" character by the bit 1. The length of a bit string att-value must equal the length defined for the attribute.

Note in Rules C, D, and E that the definition of forms NR1, NR2, and NR3 are specified in ANSI X3.42-1975 (American National Standard for the Representation of Numeric Values in Character Strings for Information Interchange). The meaning of the following characters surrounded by quotes is also specified in ANSI X3.42-1975 in the definition of forms NR1, NR2, and NR3: "b", "d", "s", and "w".

C. If the type is FIXED and integer-3 is equal to zero, then att-value is represented by a variable length numeric literal integer with an optional sign as defined in the representation of NR1. In addition, "b" (the number of digits before the implied radix point) must be less than or equal to integer-2 of the type clause. Also, "s" (the number of leading spaces) must be equal to zero and "b" must be greater than or equal to 1. The length "w" must be sufficient to represent all significant digits (all digits except leading zeroes).

An example follows of an attribute defined as FIXED with integer-2 equal to 6 (integer-3 equal to zero) with a negative value of 765. The following are acceptable representations:

-765
-Ø765
-ØØ765
-ØØØ765

D. If the type is FIXED and integer-3 is not equal to zero, then att-value is represented by a variable length fixed point numeric literal with an optional sign as defined in the representation of NR2. If integer-3 of the type clause is greater than zero, then "b" plus "d" is less than or equal to the larger of integer-2 or integer-3. If integer-3 of the type clause is less than zero, then "b" plus "d" is less than or equal to

integer-2 minus integer-3. In these two cases "s" is equal to zero and "b" is greater than or equal to 1. The length "w" must be sufficient to represent all significant digits (all digits except leading zeroes before the radix point and trailing zeroes after the radix point) as well as the radix point and sign.

An example follows of an attribute defined as FIXED with integer-2 equal to 5 and integer-3 equal to 2 with a positive value of 2.23. The following are acceptable represetations:

+2.23
 2.23
+Ø2.23Ø
+2.23ØØ

E. If the type is FLOAT, then att-value is represented by a sign (+, -, or space) followed by the significand (represented by a fixed point numeric literal) followed by the exrad (E, then plus or minus sign, then a three digit or less unsigned numeric literal) as defined in NR3. In the definition of NR3, "b" plus "d" must be less than or equal to integer-2, "s" must be equal to zero, and "f" (the number of digits in the exrad) must be less than or equal to 3. In addition, "b" must be greater than or equal to 1. The length "w" must be sufficient to represent all significant digits (all digits except leading zeroes before the radix point and trailing zeroes after the radix point) of the significand and all significant digits (no leading zeroes) of the radix plus other needed characters (signs, radix point, and E).

An example follows of an attribute defined as FLOAT with integer-2 equal to 5 with a positive value of 1ØØ67ØØ. The following are acceptable representations:

+1.ØØ67E+Ø6
 1.ØØ67E+ØØ6
+1.ØØ67E+6
 1Ø.Ø67E+Ø5
+1ØØ.67E+4

6. The ASSOCIATION IDENTIFIER and ASSOCIATION POINTER clauses form an ordered pair which is used to specify all association instances in which the entity is defined to participate as an owner or member. Assoc-id# must be defined in an Association unit of the related description section. Instance-id#-2 must be the identifier of another entity instance (i.e., Data unit) within the same data section.

7. Association instances are defined by the ASSOCIATION POINTER clause by means of "next pointers" (i.e., a

linked list). The owner instance of an association instance points to the first member instance. The member instance points to the next member instance unless it is the last member instance of the association in which case it points back to the owner instance. The SYSTEM is treated as the owner instance of all system-owned associations. The Data unit representing the SYSTEM points to the first member of all system-owned associations. The SYSTEM or an entity instance which does not participate in an association instance for which a definition exists contains a null ASSOCIATION POINTER clause (i.e., two consecutive delimiters).

8. Data units are used to represent CODASYL record occurrences, n-tuples of a relational database, and other groups of attributes which correspond to an entity.

9. The ordering of Data units in a data section is unspecified but may have an impact on performance.

4. CONSIDERATIONS FOR APPLICATION

The structured data interchange form defined in Section 3 is designed to be sufficiently general to describe structured data originating from dissimilar database management systems. Following the overview of Section 4.1 three examples are presented, each with a different type of DBMS being used to model the source database. By use of these examples, the functions of the various features in the SDICF are clarified.

4.1 OVERVIEW: REQUIRED VS. OPTIONAL FEATURES

Section 3.3 indicates that the SDICF description section contains both optional and required description units. For example, a syntactically valid description section must contain at least one entity unit but is not required to have any domain units. This section provides some guidelines for the inclusion of various types of description units in a description section. It also provides guidance for the inclusion of each of the clauses within the description units of a description section.

The lack of standard data models and the diversity of DBMS implementations requires great flexibility and richness in the SDICF constructs. As a result there exist many degrees of freedom in using these constructs to specify a data interchange and no two database administrators are likely to use exactly the same constructs. Therefore, this section does not define rules for using constructs, but rather it provides examples of "normal" utilization for three generic DBMS's -- those following the network, relational, and hierarchical data models. Since real DBMS's at most approximate these generic DBMS's and in many cases cannot be classified under this three-model scheme, the guidelines should be used for instructive purposes only.

Table 4-1 illustrates the features of the SDICF which are intended for use in accommodating each of the three major types of DBMS. Each cell of the table contains a symbol with one of the following meanings:

R - required feature
O - optional feature
E - feature is excluded from use

Table 4-1 is to be interpreted in the following manner. Each column is associated with a common type of DBMS. The symbol contained in a cell defined by a column and a description unit type (e.g., ATTRIBUTE UNIT) indicates whether the unit as a whole is required (R), optional (O), or excluded (E). Given that a description unit is included

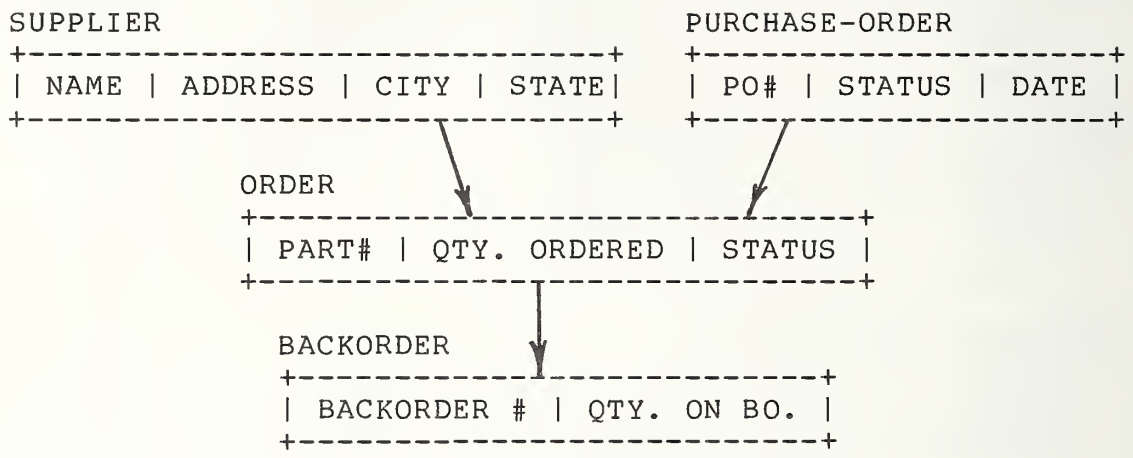
then the rows immediately following indicate the optionality of each individual clause. For example, a network DBMS requires an Attribute unit, but the domain component is excluded.

	RELATIONAL	NETWORK	HIERARCHICAL
DOMAIN UNIT	O	E	E
IDENTIFIER	R	-	-
NAME	R	-	-
TYPE	R	-	-
ATTRIBUTE UNIT	R	R	R
IDENTIFIER	R	R	R
NAME	R	R	R
TYPE	O	R	R
DOMAIN	O	E	E
AGGREGATE UNIT	E	O	O
IDENTIFIER	-	R	R
NAME	-	R	R
OCCURS	-	R	R
COMPONENTS	-	R	R
AREA UNIT	E	O	E
IDENTIFIER	-	R	-
NAME	-	R	-
ENTITY UNIT	R	R	R
IDENTIFIER	R	R	R
NAME	R	R	R
AREA	E	R	E
LOCATION MODE	E	R	E
COMPONENTS	R	R	R
PRIMARY KEY	R	O	O
INDEXED	O	O	O
ASSOCIATION	R	R	R
ASSOCIATION UNIT	R	R	R
IDENTIFIER	R	R	R
NAME	R	R	R
OWNER	R	R	R
MEMBER	R	R	R
ORDER	O	O	O
DATA UNIT	R	R	R
ENTITY ID	R	R	R
INSTANCE ID	O	O	O
AREA	E	O	E
ATTRIBUTE ID	R	O	R
ATTRIBUTE VALUE	R	O	R
ASSOCIATION ID	R	R	R
ASSOCIATION VALUE	R	R	R

SDICF Features and DBMS Types

TABLE 4-1

4.2 SDICF FILE EXAMPLES



Information Structure for
Supplier-Purchase Order Database

FIGURE 4-1

The information structure for a Supplier-Purchase Order database is illustrated in Figure 4-1. The next three subsections illustrate the modelling of the Supplier-Purchase Order database by each of three common DBMS types and a SDICF file resulting from each of the three source databases. Two points should be emphasized: (1) the SDICF specified in Section 3 is being utilized; and (2) the specific examples are not intended to depict optimal database structures but are a vehicle for discussing the application of the SDICF to different types of DBMS.

A simplifying assumption used for reducing the complexity of the examples is that the company has only one supplier of each part, but any single supplier may supply more than one part. The database used for the examples represents the situation where only one supplier exists. One purchase order has been issued for two different types of parts and one type of part has been backordered.

4.2.1 Network Database

The database described above can be represented as the network database whose schema appears in Figure 4-2. Data instances appear in Figure 4-3. The SDICF file representing this database is given in Figure 4-4.

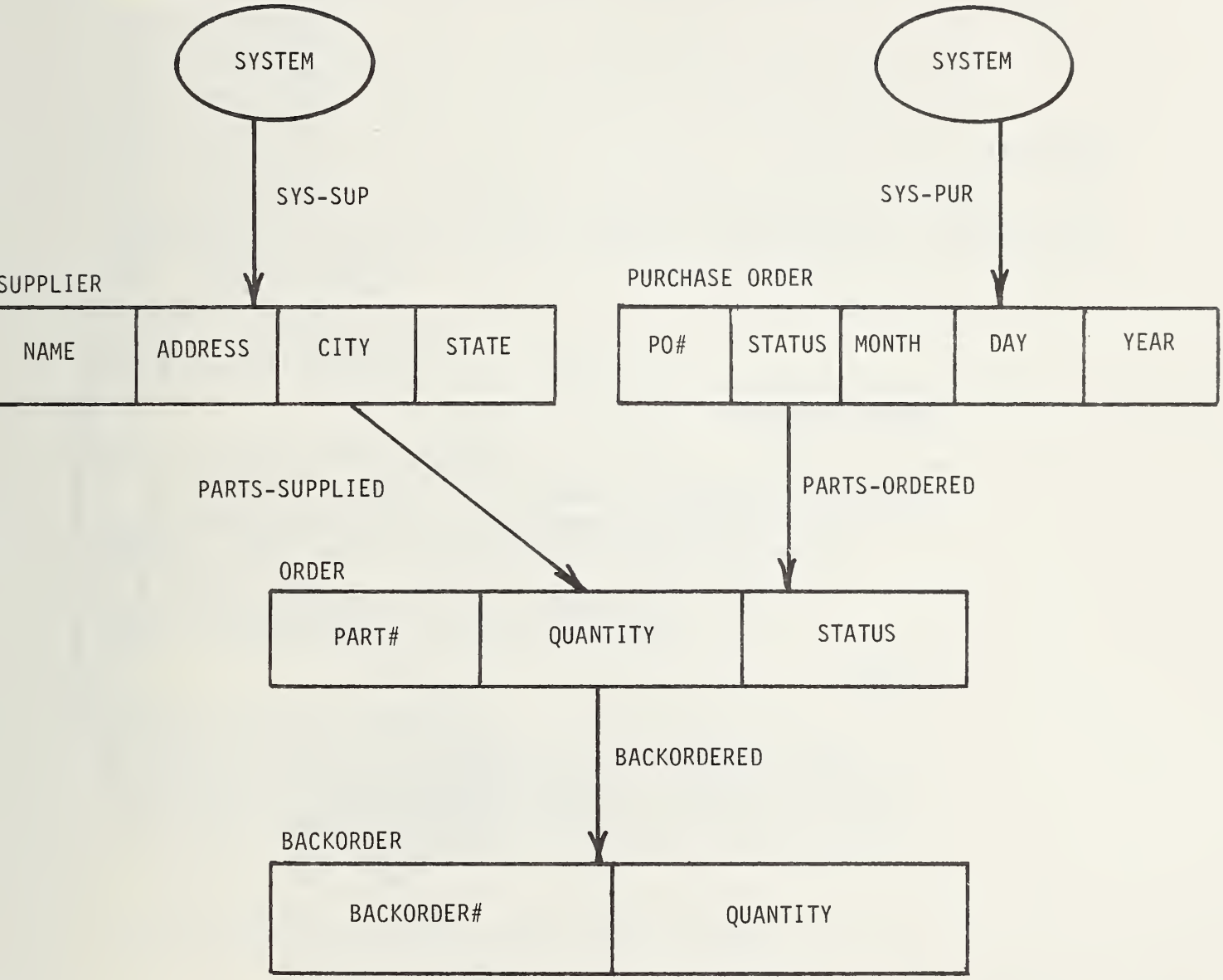
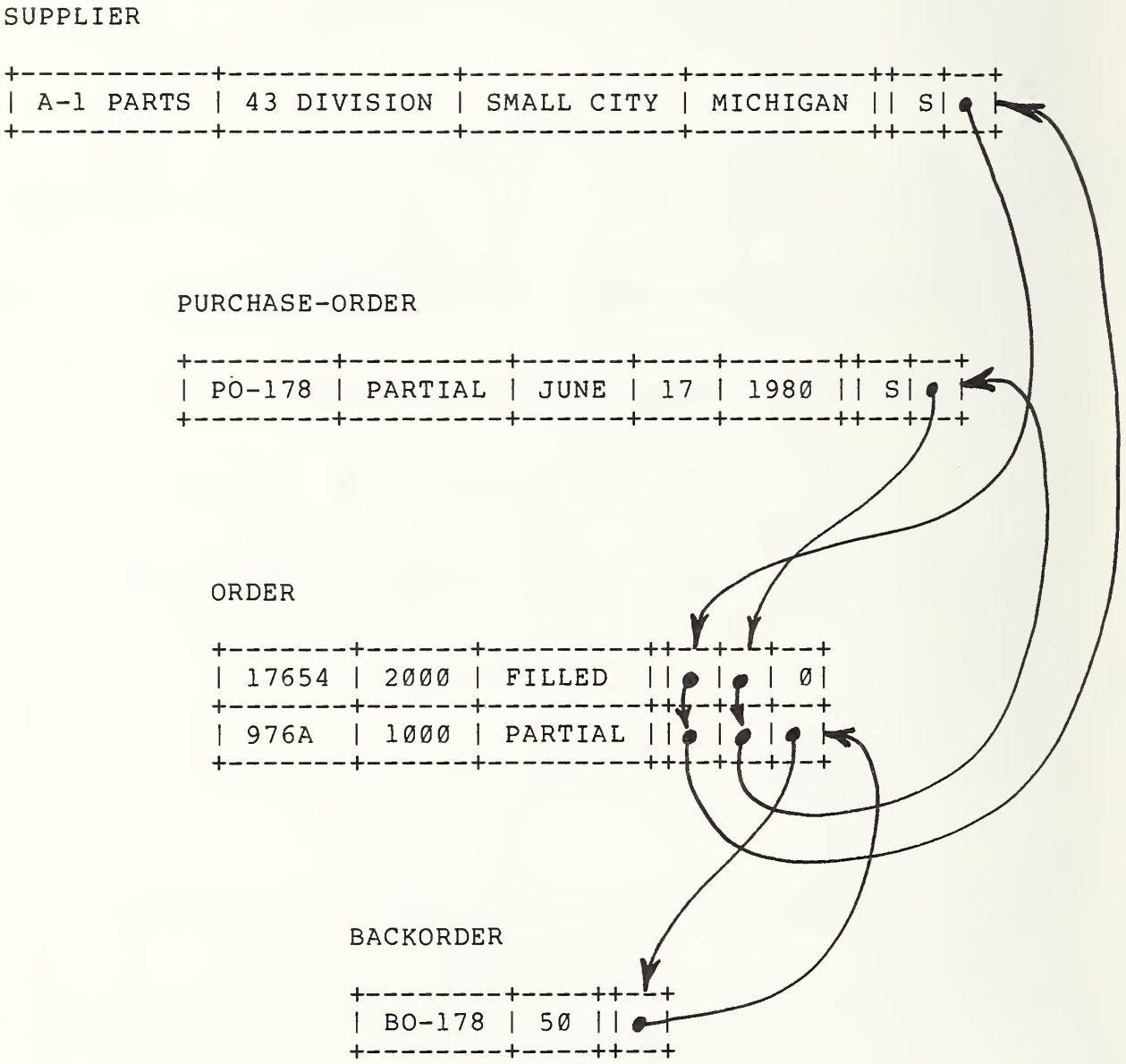


FIGURE 4-2
NETWORK DATABASE



Network Example Data Instances

FIGURE 4-3

```
DESCRIPTION;1;SUPPLIER-PURCHASE ORDER DB;801223@

AT1;NAME;CH30@
AT2;ADDRESS;CH30@
AT3;CITY;CH15@
AT4;STATE;CH15@
AT5;PO?#;CH7@
AT6;STATUS;CH7@
AT7;MONTH;CH9@
AT8;DAY;FI2@
AT9;YEAR;FI4@
AT10;PART?#;CH10@
AT11;QUANTITY;FI5@
AT12;STATUS;CH7@
AT13;BACKORDER?#;CH7@
AT14;QUANTITY;FI5@


AR1;SUPPLIER-ORDER-AREA@


EN1;SUPPLIER;AR1;SY;AT1;AT2;AT3;AT4;AS1,3@
EN2;PURCHASE-ORDER;AR1;CA5;AT5;AT6;AT7;AT8;AT9;
    AS2,4@
EN3;ORDER;AR1;VI4;AT10;AT11;AT12;AS3,4,5@
EN4;BACKORDER;AR1;VI5;AT13;AT14;AS5@


AS1;SYS-SUP;OWSY;ME1@
AS2;SYS-PUR;OWSY;ME2@
AS3;PARTS-SUPPLIED;OW1;ME3@
AS4;PARTS-ORDERED;OW2;ME3;AS5@
AS5;BACKORDERED;OW3;ME4@
#


DATA;1;SUPPLIER-PURCHASE ORDER DB;810103@


ENSY;AS1;1;AS2;2@
EN1;1;AR1;AT1;A-1 PARTS;AT2;43 DIVISION;AT3;SMALL CITY;
     AT4;MICHIGAN;AS1;SYSTEM;AS3;3@
EN2;2;AR1;AT5;PO-178;AT6;PARTIAL;AT7;JUNE;AT8;17;AT9;1980;
     AS2;SY;AS4;3@
EN3;3;AR1;AT10;17654;AT11;2000;AT12;FILLED;AS3;4;AS4;4;AS5;3@
EN3;4;AR1;AT10;976A;AT11;1000;AT12;PARTIAL;AS3;1;AS4;2;AS5;5@
EN4;5;AR1;AT13;BO-178;AT14;50;AS5;4@
#
```

SDICF File Representing Network Database

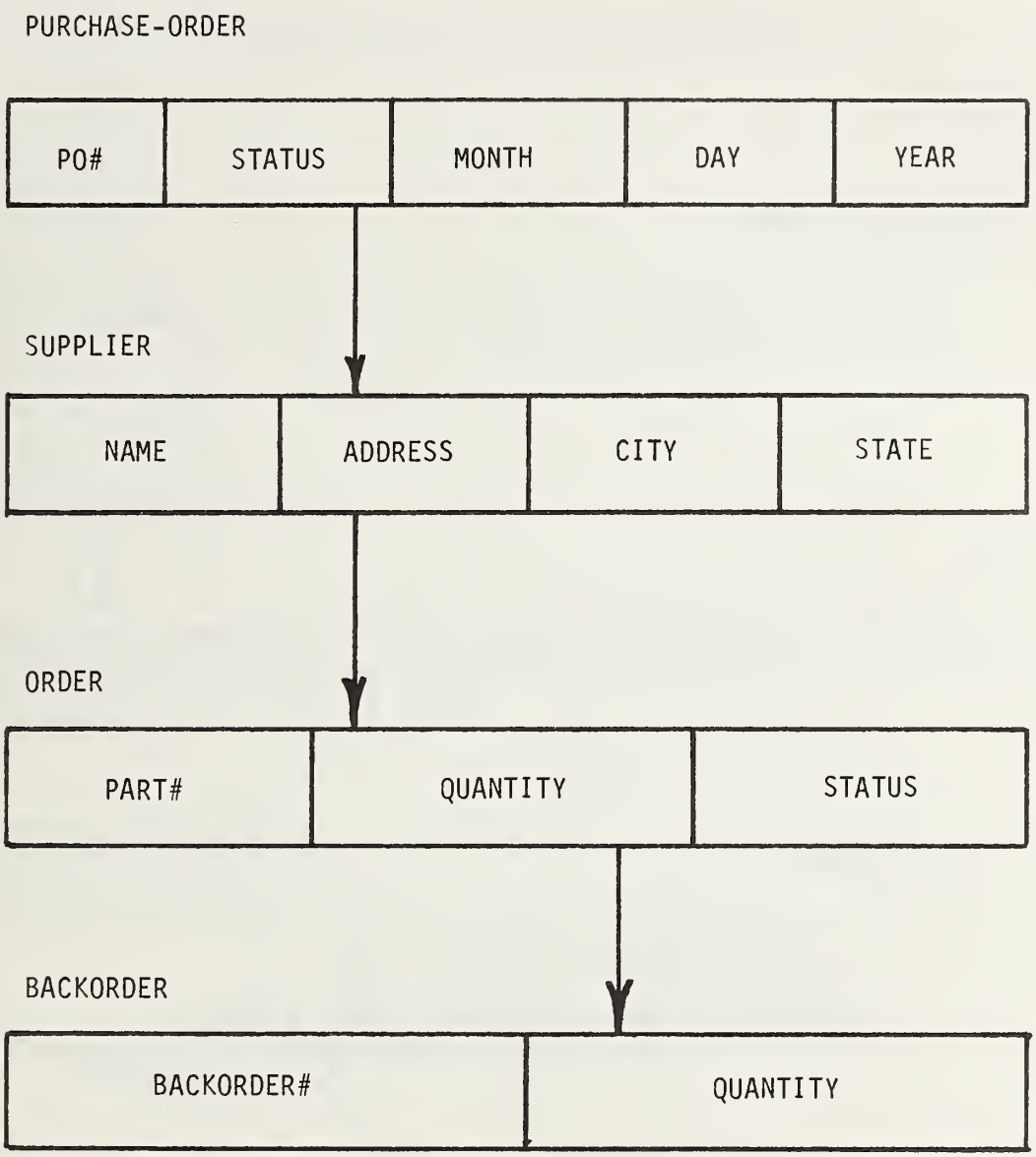
FIGURE 4-4

Comments/Considerations:

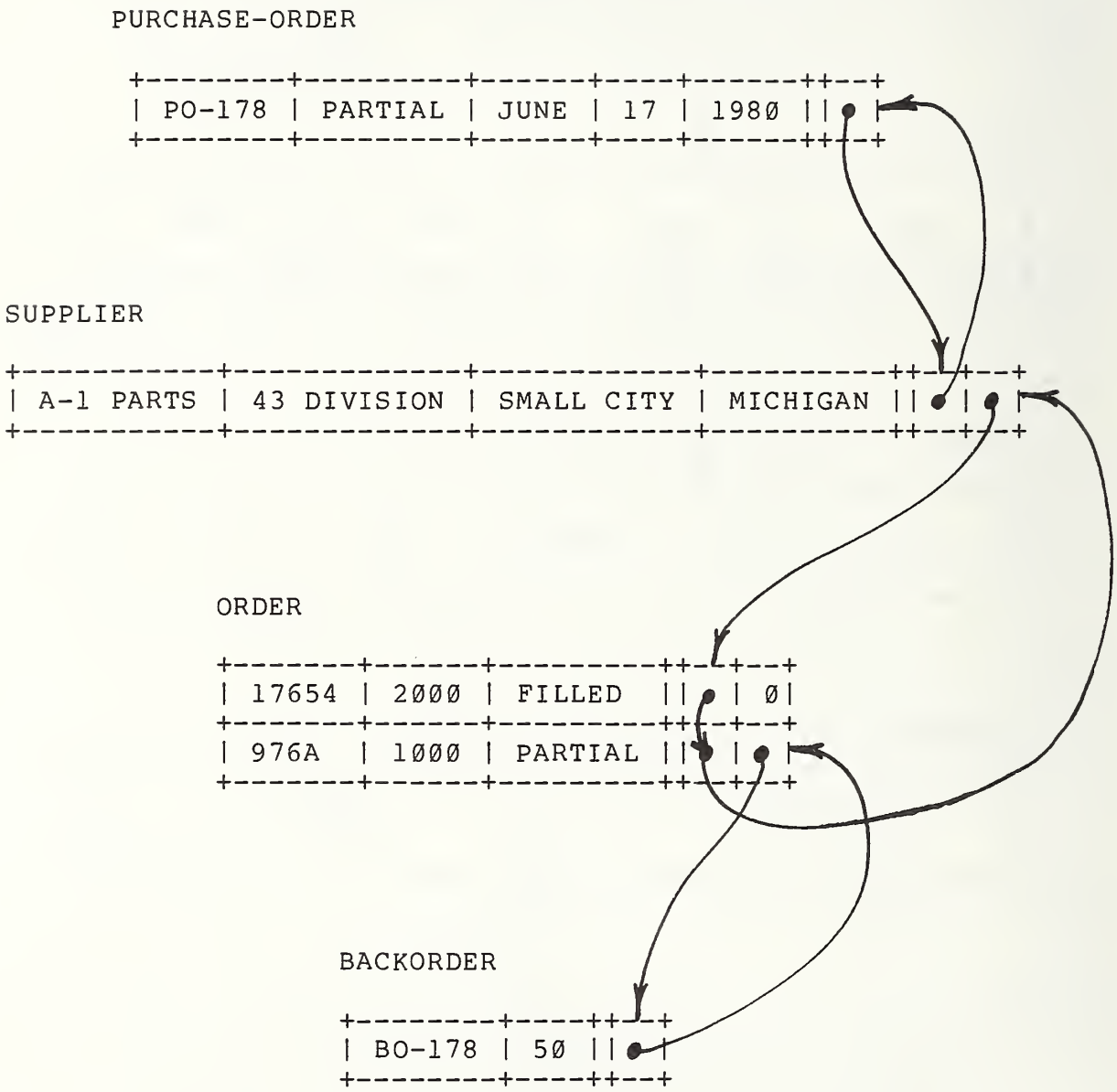
(1) The relative positioning of the record instances within a set is maintained by use of the Data units' ASSOCIATION POINTER CLAUSE, i.e., the owner points to the first member, the first member points to the second member, etc., and the last member points back to the owner. Each association pointer in the data unit is matched to the association listed in the ordered pair.

4.2.2 Hierarchical Database

Figure 4-5 depicts a hierarchical representation of the database described above. Data instances appear in Figure 4-6. The SDICF file representing this database is given in Figure 4-7.



HIERARCHICAL DATABASE
FIGURE 4-5



Hierarchical Example Data Instances

FIGURE 4-6

```
DESCRIPTION;2;SUPPLIER-PURCHASE ORDER DB;810101@


AT1;PO?#;CH7@
AT2;STATUS;CH7@
AT3;MONTH;CH9@
AT4;DAY;FI2@
AT5;YEAR;FI4@
AT6;NAME;CH30@
AT7;ADDRESS;CH30@
AT8;CITY;CH15@
AT9;STATE;CH15@
AT10;PART?#;CH10@
AT11;QUANTITY;FI5@
AT12;STATUS;CH7@
AT13;BACKORDER?#;CH10@
AT14;QUANTITY;FI5@


EN1;PURCHASE-ORDER;AT1;AT2;AT3;AT4;AT5;AS1,2@
EN2;SUPPLIER;AT6;AT7;AT8;AT9;AS2,3@
EN3;ORDER;AT10;AT11;AT12;AS3,4@
EN4;BACKORDER;AT13;AT14;AS4@


AS1;SYS-PUR;OWSY;ME1@
AS2;PUR-SUP;OW1;ME2@
AS3;SUP-ORD;OW2;ME3@
AS4;ORD-BAC;OW3;ME4@
#


DATA;2;SUPPLIER-PURCHASE ORDER DB;810101@


ENSY;AS1;1@
EN1;1;AT1;PO-178;AT2;PARTIAL;AT3;JUNE;AT4;17;AT5;1980;
    AS1;SY;AS2;2@
EN2;2;AT6;A-1 PARTS;AT7;43 DIVISION;AT8;SMALL CITY;AT9;MICHIGAN;
    AS2;1;AS3;3@
EN3;3;AT11;17654;AT12;2000;AT13;FILLED;AS3;4;AS4;3@
EN3;4;AT11;976A;AT12;1000;AT13;PARTIAL;AS3;2;AS4;5@
EN4;5;AT13;BO-178;AT14;50;AS4;4@
#
```

SDICF File Representing Hierarchical Database

FIGURE 4-7

Comments/Considerations:

(1) The relative positioning of record instances is maintained in the same manner as established for a network database as illustrated in Section 4.2.1.

(2) Naming conventions which are used in mapping the hierarchical DBMS schema into a SDICF file include:

    (a) "Parent" record types are equated to owner entity types in the associations

    (b) "Child" record types are equated to member entity types in the associations

    (c) Associations are named by concatenating the beginning portions of the names of both the owner entity type and the member entity type.

(3) The root record type is the member entity of an association owned by SYSTEM.

4.2.3 Relational Database

A relational equivalent of the information structure of the Supplier-Purchase Order database is provided in Figure 4-8. The relations for the data of the example are given in Figure 4-9. The SDICF file representing this database is depicted in Figure 4-10.

SUPPLIER (SUPPLIER#,NAME,ADD,CITY,STATE)

SUPPLIES (SUPPLIER#,PART#)

PURCHASE-ORDER (PO#,STATUS,MONTH,DAY,YEAR)

ORDER (PO#,PART#,QTY_ORDERED,STATUS)

BACKORDER (PO#,PART#,QTY_ON_BACKORDER)

Definition of Relations for Example Database

FIGURE 4-8

SUPPLIER

1	A-1 PARTS	43 DIVISION	SMALL CITY	MICHIGAN

PURCHASE-ORDER

PO-178	PARTIAL	JUNE	17	1980

SUPPLIES

1	17654
1	976A

BACKORDER

BO-178	976A	50

ORDER-INFORMATION

PO-178	17654	2000	FILLED
PO-178	976A	1000	PARTIAL

Relational Example Data Instances

FIGURE 4-9

```
DESCRIPTION;3;Supplier-Purchase Order DB;810102@


DO1;SUP?#;CH7@
DO2;P?#;CH10@
DO3;PO?#;CH7@



AT1;SUPPLIER?#;DO1@
AT2;NAME;CH30@
AT3;ADD;CH30@
AT4;CITY;CH15@
AT5;STATE;CH15@
AT6;SUPPLIER?#;DO1@
AT7;PART?#;DO2@
AT8;PO?#;DO3@
AT9;STATUS;CH7@
AT10;MONTH;CH9@
AT11;DAY;FI2@
AT12;YEAR;FI4@
AT13;PO?#;DO3@
AT14;PART?#;DO2@
AT15;QTY-ORDERED;FI5@
AT16;STATUS;CH7@
AT17;PO?#;DO3@
AT18;PART?#;DO2@
AT19;QTY-ON-BACKORDER;FI5@


EN1;SUPPLIER;AT1;AT2;AT3;AT4;AT5;PR1;AS1@
EN2;PURCHASE-ORDER;AT8;AT9;AT10;AT11;AT12;PR8;AS2@
EN3;SUPPLIES;AT6;AT7;PR6,7;AS3@
EN4;ORDER;AT13;AT14;AT15;AT16;PR13,14;AS4@
EN5;BACKORDER;AT17;AT18;AT19;PR17,18;AS5@



AS1;SYS-SUP;OWSY;ME1;AS1@
AS2;SYS-PUR;OWSY;ME2@
AS3;SYS-SUP;OWSY;ME3;AS6@
AS4;SYS-ORD;OWSY;ME4;AS13@
AS5;SYS-BAC;OWSY;ME5@
#
```

Description Section of SDICF File Representing

Relational Database

FIGURE 4-10(a)

```
DATA;3; SUPPLIER-PURCHASE ORDER DB;810103@

ENSY;AS1;1;AS2;2;AS3;3;AS4;5;AS5;7@
EN1;1;AT1;A-1 PARTS;AT2;43 DIVISION;AT3;SMALL CITY;AT4;MICHIGAN;
    AS1;SY@
EN2;2;AT8;PO-178;AT9;PARTIAL;AT10;JUNE;AT11;17;AT12;1980;AS2;SY@
EN3;3;AT6;1;AT7;17654;AS3;4@
EN3;4;AT6;1;AT7;976A;AS3;SY@
EN4;5;AT13;PO-178;AT14;17654;AT15;2000;AT16;Filled;AS4;6@
EN4;6;AT13;PO-178;AT14;976A;AT15;1000;AT16;Partial;AS4;SY@
EN5;7;AT17;BO-178;AT18;976A;AT19;50;AS5;SY@
#
```

Data Section of SDICF File Representing
Relational Database

FIGURE 4-10(b)

Comments/Considerations:

(1) Each relation participates in exactly one association of which 'SYSTEM' is the owner entity type. Each relation, therefore, is a member entity type.

(2) The relative positioning of tuple instances within a single relation of the source database is maintained in a manner similar to the ordering of record instances as discussed in Section 4.2.1.

(3) Naming conventions used in the mapping of the relational database into the SDICF file include: Association types are named by concatenating the prefix 'SYS-' with the beginning of the relation name.

4.3 INTRA- AND INTER-DATA MODEL INTERCHANGE

This section presents the basic constructs of three general data models and discusses the interchange of databases between them. Without going into the details of different DBMS implementation using these data models, a set of basic constructs for each is provided in Section 4.3.1. These constructs are derived from the "schema definition languages" of the systems using these data models. Since the SDICF is an "integration" of information described by these schema definition languages, the constructs of the SDICF are a composite of the three data models. Section 4.3.2 discusses the different types of constructs that are produced in the cases of intra-data model and inter-data model data interchanges.

4.3.1 Basic Constructs of Data Models

The basic constructs of the three data models indicated in Figure 4-11 are as follows:

1. Relational data model: domains and relations which in turn consist of a key (primary key) and components.

2. Network data model: areas, named sets (with some having the system assigned as owner) and record types which in turn may consist of keys, groups, and fields. A group consists of fields.

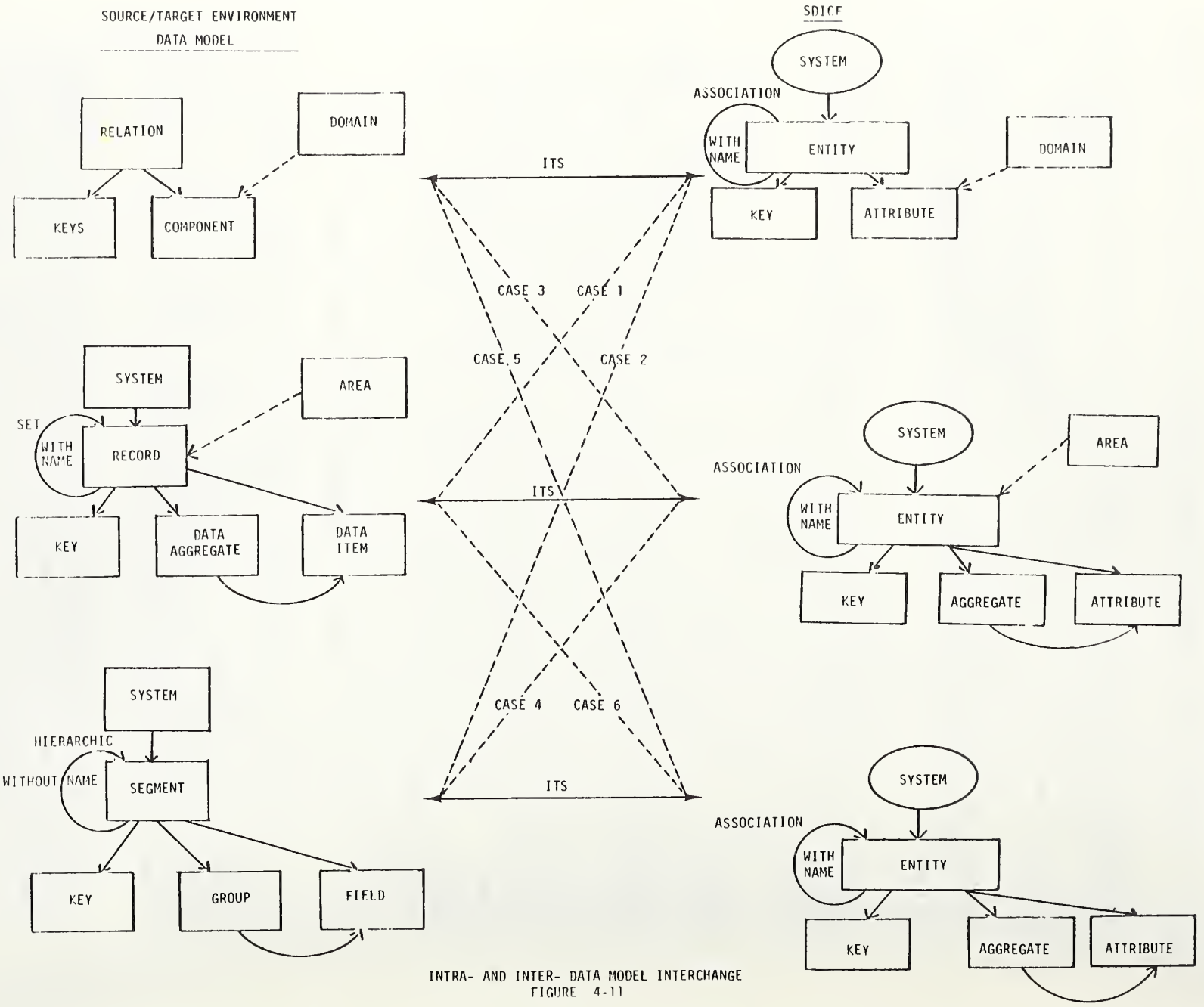
3. Hierarchical data model: unnamed hierarchical relationships (with some having the system assigned as parent) and segments which in turn may consist of keys, groups, and fields. A group consists of fields.

The SDICF is a "composite" data form of the three general data models. It includes such constructs as: entities, areas, domains, and named associations. Entities in turn may consist of keys (primary key), aggregates, and attributes. Aggregates consist of attributes. Comparing the constructs of the three data models with those of the SDICF, two types of constructs are observed. Those constructs that exist in all data models are called common constructs. Examples are entities and attributes. Those constructs which are specific to a particular data model are non-common constructs. Examples are "domains" of the relational data model, "areas", and "data aggregates" of the network data model, "group" of the hierarchical data model. Non-common constructs are particularly useful in intra-model data interchange.

4.3.2 Intra- and Inter-Data Model Data Interchange

There are three data models and two transformations to be considered for inter-data model interchange. Two data models are explicit -- the source and target models; one is implicit -- the model of the SDICF. The transformation from source to SDICF is performed by the input transformation system and the transformation from the SDICF to the target is performed by the output transformation system.

In a data interchange it is assumed that the source system has no "knowledge" of the target system's data model. Therefore the ITS which performs the mapping to the SDICF is constructed to perform a one-to-one mapping (see Figure 4-11, horizontal line). For example, in the case where the source data model is relational, the mapping to the SDICF is the following: a "relation" is mapped to an "entity", a "key" to a "key", a "domain" to a "domain", and a "component" to an "attribute".



INTRA- AND INTER- DATA MODEL INTERCHANGE
FIGURE 4-11

The primary input to the OTS is the SDICF file constructed from the source DBMS. It is used to construct common constructs of the target data model. For example, if the SDICF is constructed from a relational source data model and the target data model is also relational (intra-data model interchange) then the process of the OTS is relatively easy. It only has to map "entity" to "relation", "domain" to "domain", "key" to "key", and "attribute" to "component". If the target data model is not relational (inter-data model interchange) then the situation becomes more complicated for the OTS. The following is a discussion of the mappings required for the two cases of data interchange.

4.3.2.1 Intra-Data Model Data Interchange
(See Figure 4-11, solid horizontal lines.)

The mapping performed by the OTS for intra-data model data interchange is essentially one-to-one. The following table shows the correspondence:

RELATIONAL SOURCE DATA MODEL	--->	SDICF DATA MODEL	--->	RELATIONAL TARGET DATA MODEL
relation		entity		relation
domain		domain		domain
key		key		key
component		attribute		component

NETWORK SOURCE DATA MODEL	------>	SDICF DATA MODEL	------>	NETWORK TARGET DATA MODEL
record		entity		record
set with name		association w/ name		set with name
key		key		key
data aggregate		aggregate		data aggregate
data item		attribute		data item
area		area		area

HIERARCHICAL SOURCE DATA MODEL	--->	SDICF DATA MODEL	--->	HIERARCHICAL TARGET DATA MODEL
segment		entity		segment
hierarchy w/o name		association w/ name		hierarchy w/o name
key		key		key
group		aggregate		group
field		attribute		field

4.3.2.2 Inter-Data Model Data Interchange

As indicated by dashed non-horizontal lines in Figure 4-11, there are six cases of inter-data model data interchange:

Case 1.	Relational ---> SDICF --> Network	(RN)
Case 2.	Relational ---> SDICF --> Hierarchical	(RH)
Case 3.	Network ------> SDICF --> Relational	(NR)
Case 4.	Network ------> SDICF --> Hierarchical	(NH)
Case 5.	Hierarchical -> SDICF --> Relational	(HR)
Case 6.	Hierarchical -> SDICF --> Network	(HN)

In all of the above cases the "common" constructs can be constructed by the OTS using the corresponding constructs of the SDICF. The "non-common" constructs such as "domain" and "area", however, must be provided by the database administrator. These six cases are described in detail below. Because the specific transformations are DBMS dependent no detailed algorithms are specified.

CASE 1. Relational --> SDICF --> Network (RN)

Target Construct	Source Construct from SDICF
1) record	- entity
data item	- attribute of entity
data aggregate	- entity
key	- key of entity
2) named set type	- entity
3) area	- constructed by DBA

CASE 2. Relational --> SDICF --> Hierarchical (RH)

Target Construct	Source Construct from SDICF
1) segment	- entity
field	- attribute of entity
group	- entity
key	- key of entity
2) hierarchy without name	- entity

CASE 3: Network --> SDICF --> Relational (NR)

Target Construct	Source Construct from SDICF
1) relation	- entity - (repeating) aggregate - association with name
component	- attribute of the entity -(repeating) aggregate
key	- key of entity or (repeating) aggregate - key of owner entity
2) domain	- constructed by DBA

CASE 4: Network --> SDICF --> Hierarchical (NH)

Target Construct	Source Construct from SDICF
1) segment	-entity
field	-attribute
group	-aggregate
key	-key of entity
2) hierarchy without name	-association

CASE 5: Hierarchical --> SDICF --> Relational (HR)

Target Construct	Source Construct from SDICF
1) relation	-entity -(repeating)aggregate -association with name
component	-attribute of entity or (repeating) aggregate -key of owner
key	-key of entity or (repeating) aggregate -key of owner
2) domain	- constructed by DBA

CASE 6. Hierarchical --> SDICF --> Network (HN)

Target Construct	Source Construct from SDICF
1) record	- entity
data item	- attribute of entity
data aggregate	- aggregate
key	- key
2) named set type	- association
3) area	- constructed by DBA

APPENDIX A

GENERAL CHARACTERIZATION OF I/O SUBSYSTEM

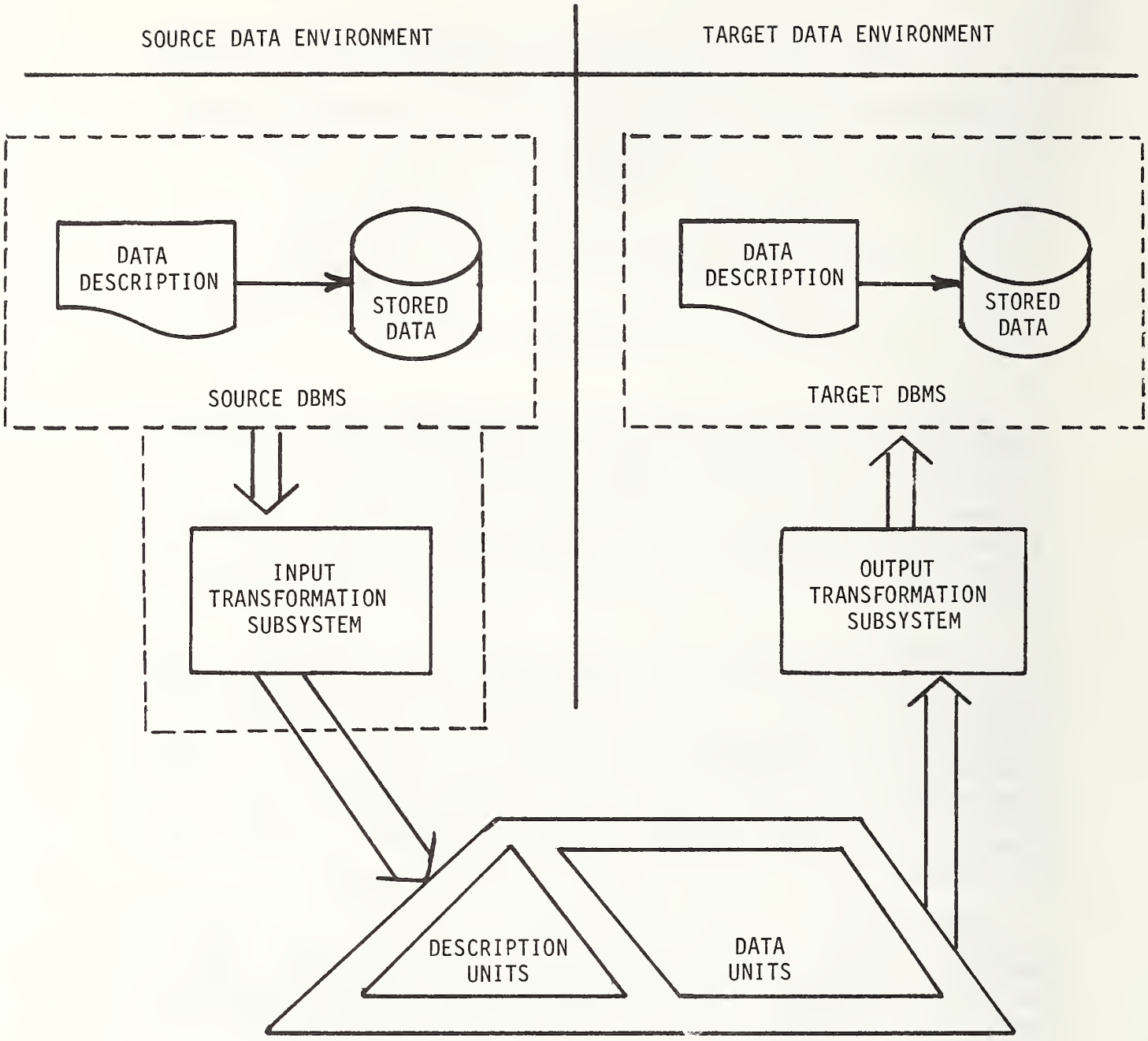
1. INTRODUCTION

The problem of structured data interchange is directly related to the complexity of and differences in the source and target stored-data. The purpose of the SDICF interface is to mitigate this problem by providing a common interchange form for the source and target systems to translate their stored data. To provide a context for this interchange a Structured Data Interchange System (SDIS) is introduced.

As shown in Figure A-1, the Structured Data Interchange System is composed of five components: (1) the source data environment which contains the data to be interchanged, an explicit description of that data, and a DBMS; (2) the input transformation subsystem (ITS) which converts the source database to a SDICF file; (3) the SDICF file which is the representation of the description and contents of the source data; (4) the output transformation subsystem (OTS) which converts the SDICF file to a database in the target environment; and (5) the target data environment.

The purpose of this appendix is to formulate a range of alternatives for implementing an ITS and an OTS, and to provide a general characterization of one example approach. The characterization concentrates on high level constructs and is intended as an aid in understanding the transformation subsystems.

In this appendix and the following appendices commercial database management systems or utilities are mentioned by trade names as necessary to illustrate the structured data interchange form and the input and output transformation subsystems. Inclusion of a system in this report in no case implies a recommendation or endorsement by the National Bureau of Standards.



STRUCTURED DATA INTERCHANGE SYSTEM
FIGURE A-1

2. APPROACH

The range of approaches to the design of a Stuctured Data Interchange System spans a spectrum from very specialized programs of relative simplicity to extremely general utilities of extreme complexity. The choice of a design approach influences many other important design issues. Three such alternatives are examined in the following paragraphs with special attention being paid to the design implications resulting from the alternative chosen.

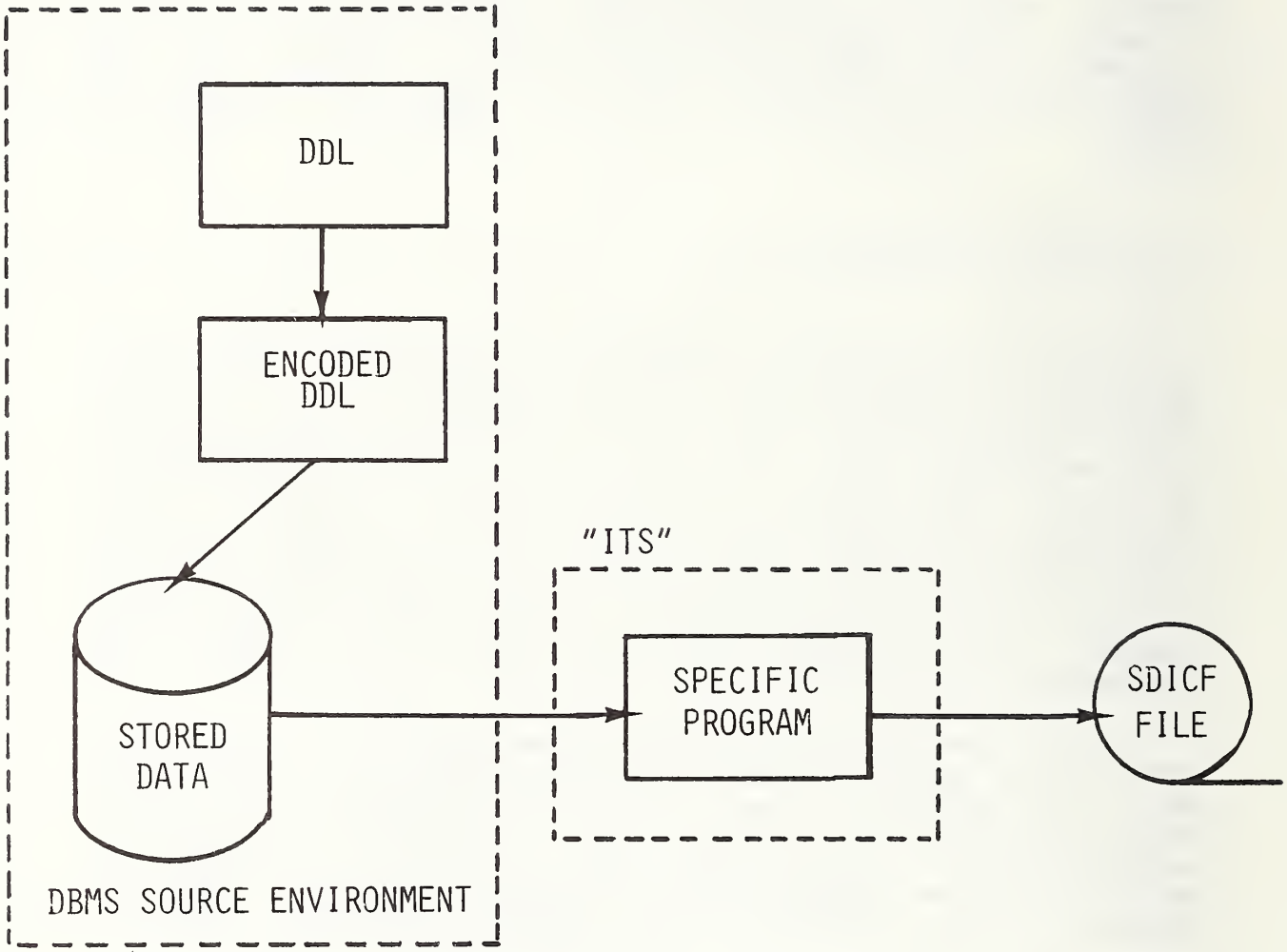
2.1 SPECIAL PROGRAMS

Figure A-2 illustrates the use of a specialized program which works by directly accessing the data in the source environment to produce data for the SDICF file. In this case the information about the structure and format of the stored data and the SDICF file is "built-in" to the program. Although a DBMS accessing mechanism might be used to read the data, it probably would not be used in order to attain high efficiency. This approach has the advantage of the fastest interchange process in terms of machine execution, but the inherent disadvantage of an inflexible design (any change in the format of the stored data results in re-programming and redesigning the ITS).

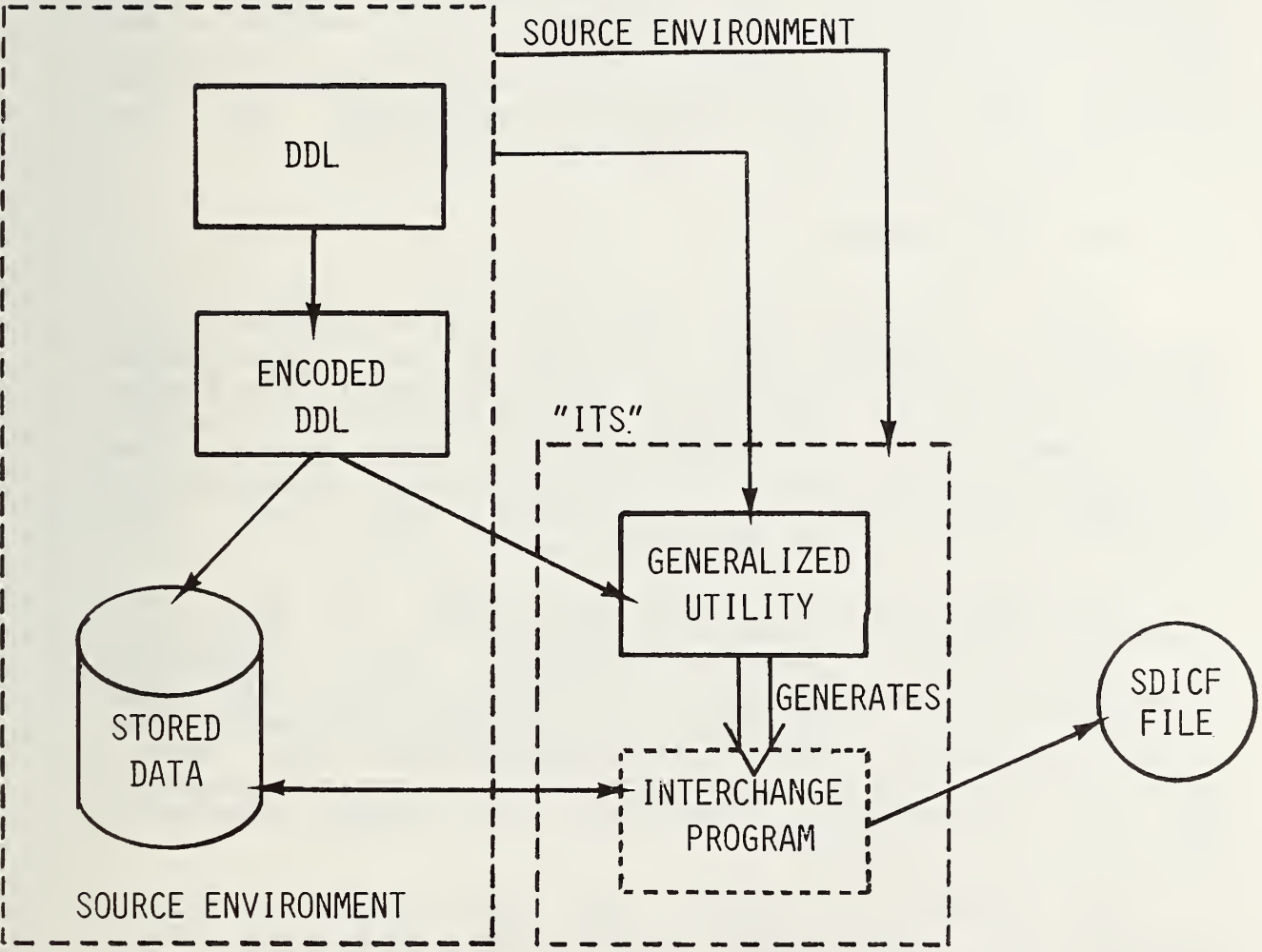
There are two design implications resulting from this alternative. Since the program is specialized, it will serve only one database. Thus, it could not be used on different databases even if they were on the same computer system. The second point to note is that a specialized access program is not likely to be transportable between dissimilar computer systems.

2.2 GENERALIZED UTILITY

At the other end of the spectrum is the development of a generalized utility to facilitate the interchange process. As illustrated in Figure A-3 the generalized utility utilizes both the source data description and expert knowledge of the DBMS environment to generate a program which accesses the stored-data in the source environment and produces the SDICF file. An example of a commercially available product to perform this is IBM's Data Extraction, Processing and Restructuring System (SH20-2178-0). The generalized utility approach has the advantage of flexibility and efficiency in a high volume and



SPECIALIZED ITS
FIGURE A-2



GENERALIZED ITS
FIGURE A-3

multi-database situation. The main drawback is the cost of developing the generalized software.

Several design implications result from this alternative. While it provides the flexibilty to be used on many different databases utilizing perhaps different DBMS's, it is generally not applicable to dissimilar operating systems or hardware environments. The design of the utility for this case would be extremely complex since it would need to address different interfaces among the DBMS, the operating system and the hardware. This combination of factors results in an extremely complex piece of software which is generally considered to be unattainable given the current state of the translation technology and the proliferation of hardware and operating systems.

2.3 THE MIDDLE GROUND

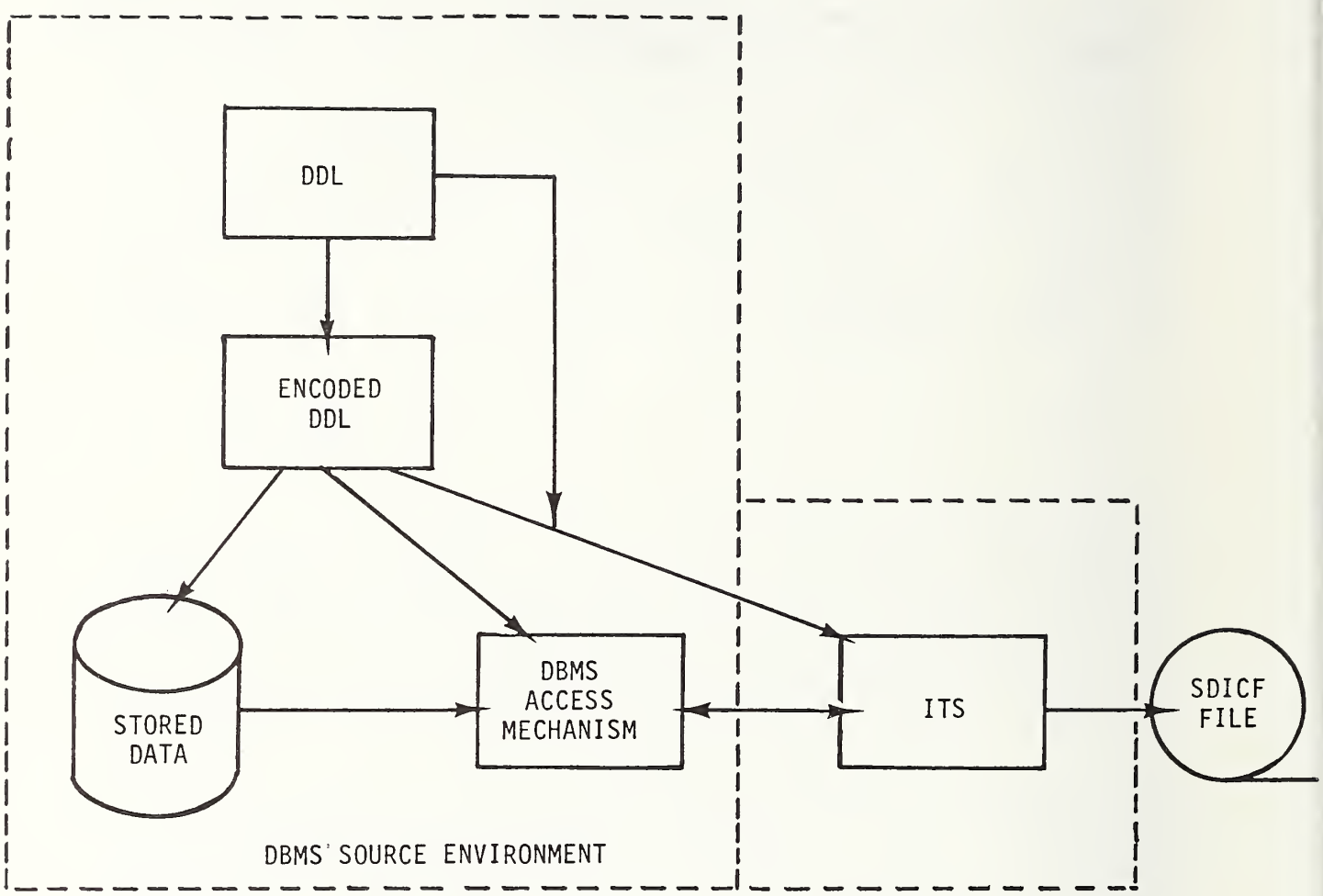
Three variations are provided for a middle ground approach to the design of a Structured Data Interchange System. The first variation, which is depicted in Figure A-4(a), involves the interactive access of the data from the source DBMS database. This approach allows the ITS to take advantage of the relatively efficient access mechanism of the DBMS but requires the ITS to generate the "calls" for data and also to handle the resulting data stream.

By requiring the source environment to place the stored-data in a linearized representation, the complexity of the ITS is greatly reduced. This linearized representation can take one of two forms: (1) a dump tape specific to the DBMS (e.g., an IMS dump tape would be a linearized representation of the database with the records placed in top-to-bottom, left branch first preorder format); and (2) a "standard" linearized data stream developed explicitly for the SDICF.

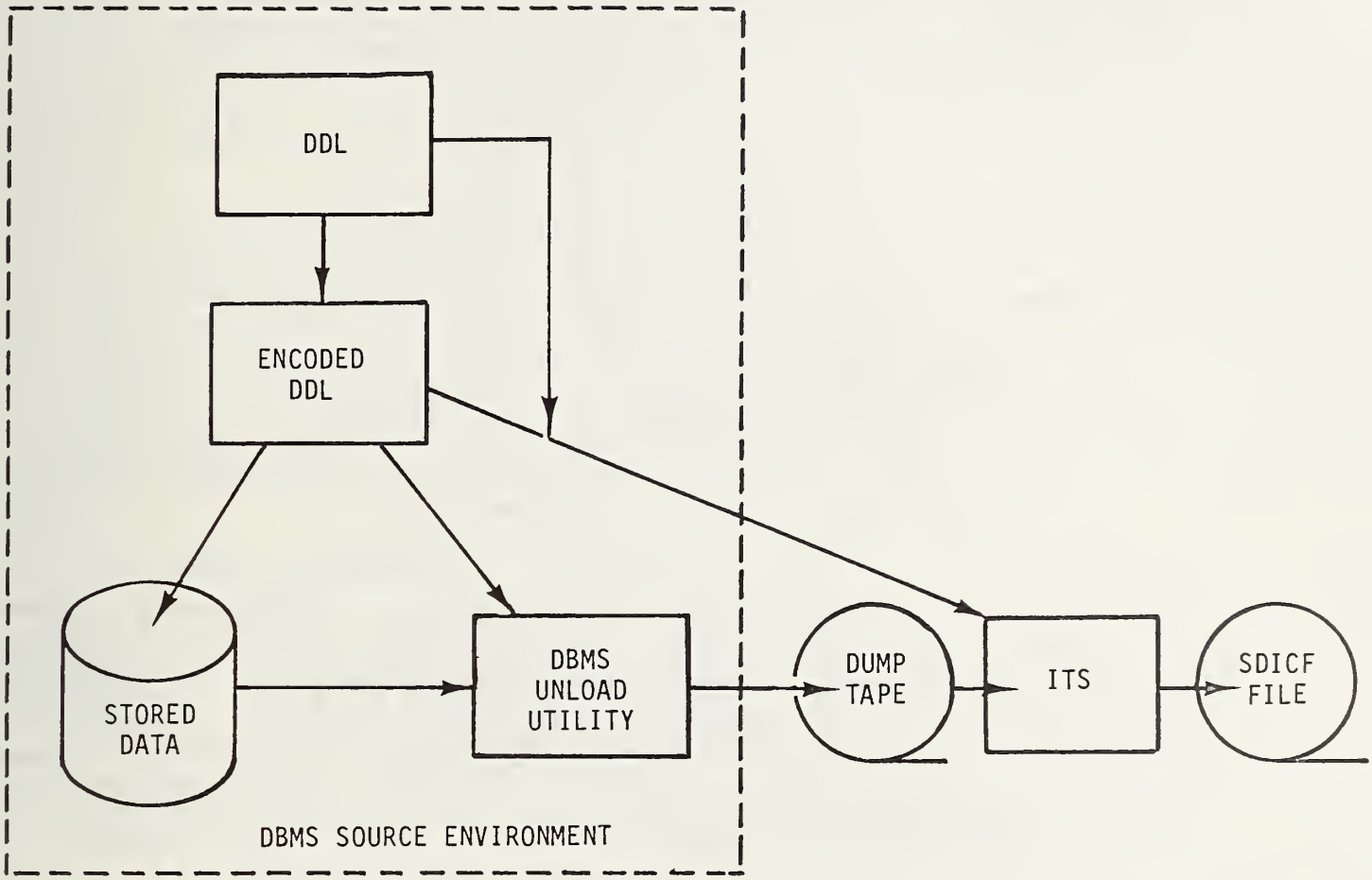
As depicted in Figure 4(b), the second variation utilizes the DBMS's unload utility and requires the ITS to interface with the resulting dump tape data stream. This greatly simplifies the stored-data complexity but requires a generalized tape read program. In the third variation, as indicated in Figure 4(c), a special source DBMS program is written to create a special SDICF input data stream. This would simplify the ITS accessing and make it possible to write a generalized ITS which could be adapted to any source environment. It does, however, require that a generalized program be written for each DBMS which is a duplication of the DBMS unload utility.

For the purpose of this discussion, the approach depicted in Figure A-4(b) has been adopted - the use of the

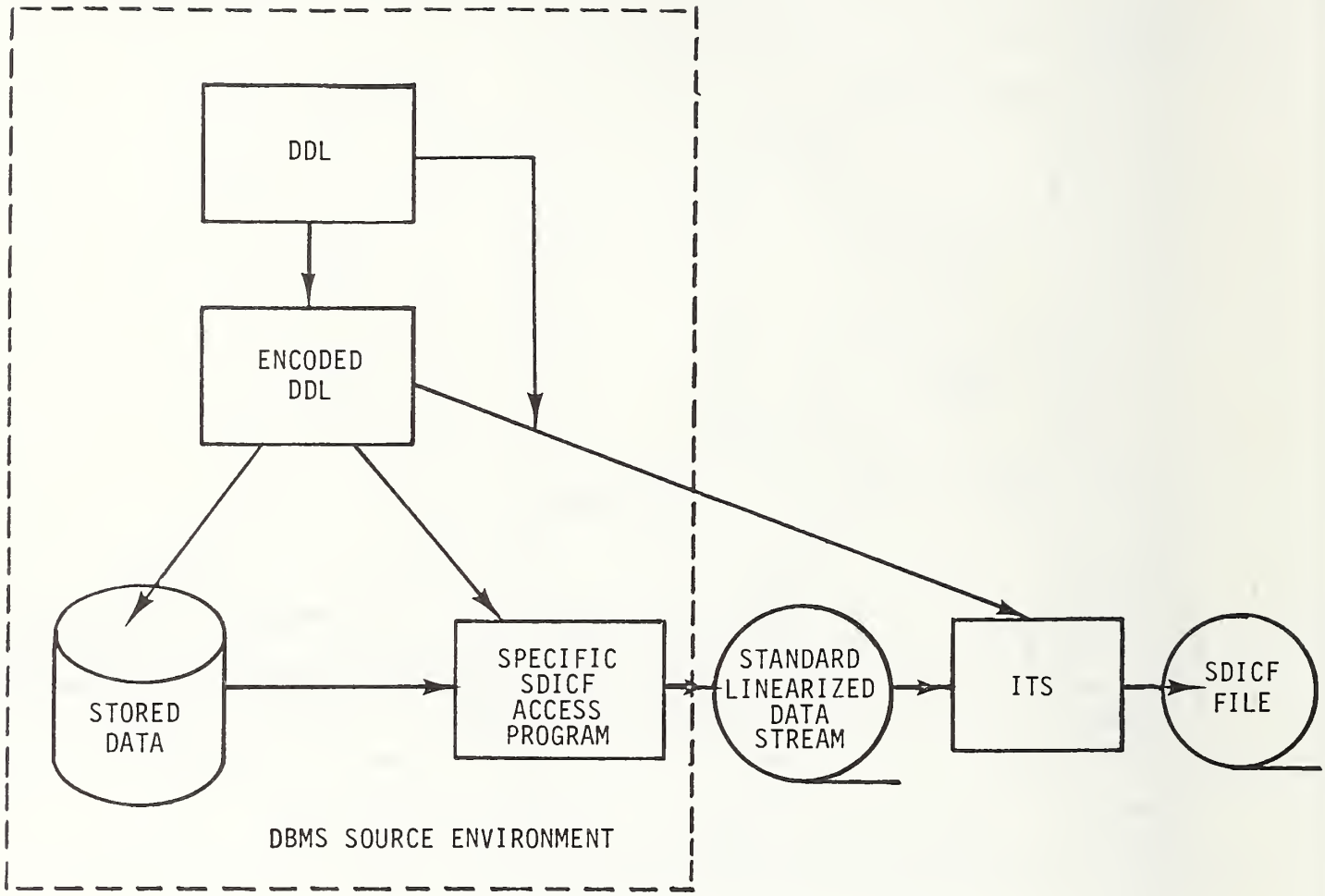
source DBMS dump tape. This approach reflects the overall philosophy and design of the SDICF by dividing the overall SDIS (see Figure A-1) design task into two parts: the design of the ITS in the source environment and the design of the OTS in the target environment.



ITS USING DBMS ACCESS MECHANISMS
FIGURE A-4(a)



ITS USING DBMS DUMP TAPE
FIGURE A-4(b)



ITS USING STANDARD LINEARIZED DATA STREAM
FIGURE A-4(c)

3. COMPONENTS OF INPUT AND OUTPUT TRANSFORMATION SUBSYTEMS

This section provides furthur details and refinements of the issues involved in the design of both the input and the output subsystems. It is based upon the alternative identified in Section 2.3 of this appendix.

3.1 OVERVIEW OF COMPONENTS

The ITS receives as input the description of the source database in unload utility format and produces the SDICF file. The OTS receives as input the SDICF file and creates a skeleton DBMS data description, and generates data for the target DBMS to load. After the database administrator completes the target data description the DBMS load utility is envoked to create the database. Three generic components are recognized to exist in both the ITS and the OTS subsystems (see Figure A-5).

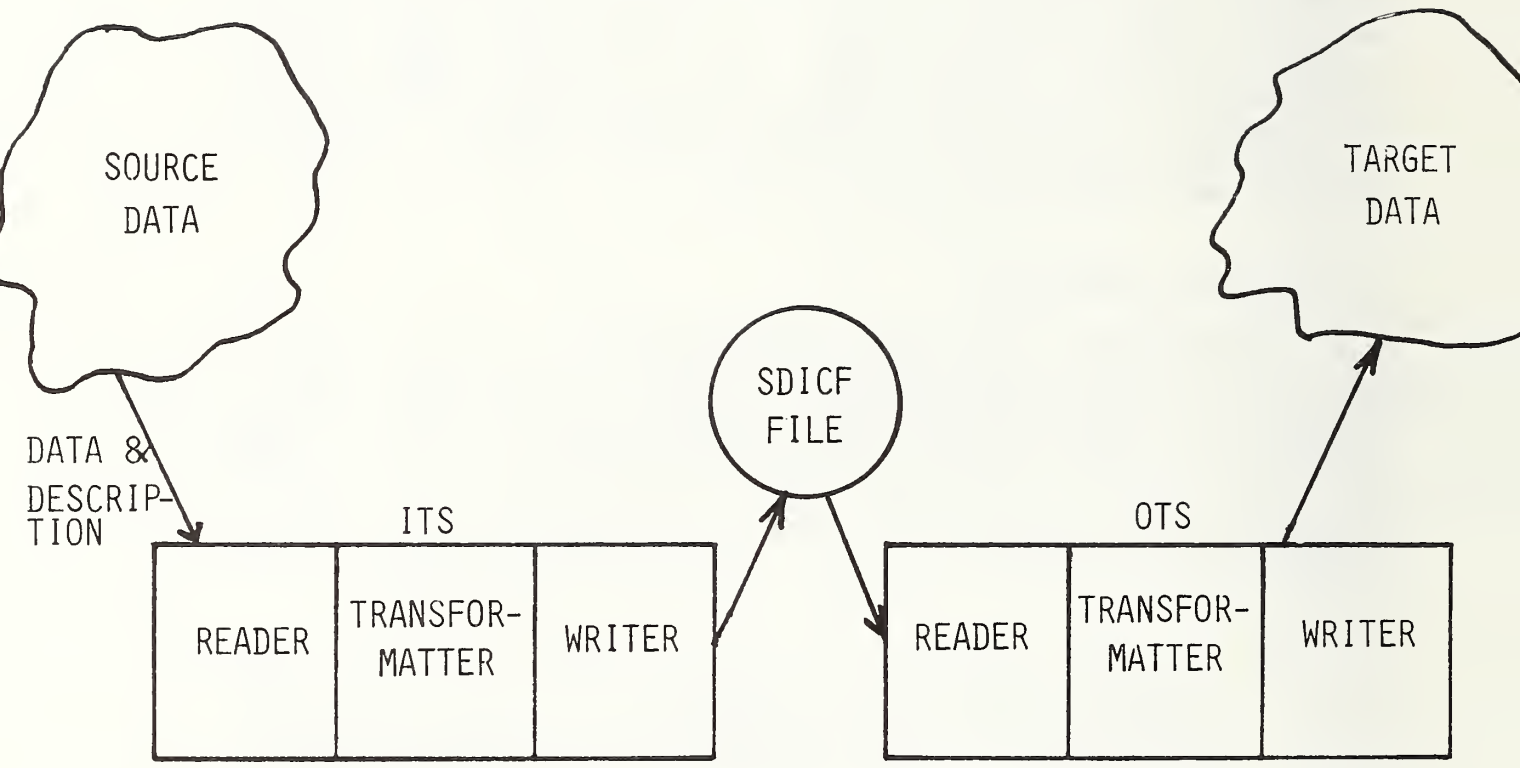
3.2 READER COMPONENT

In the ITS, the READER "processes" the data as defined by the source DBMS schema definition language and its mapping to the sequential unload format. This process mirrors the DBMS load program which may be modified to serve as the READER. As such, the ITS READER design is greatly dependent upon the source DBMS environment. The READER component is further divided into two sub-components (see Figure A-6):

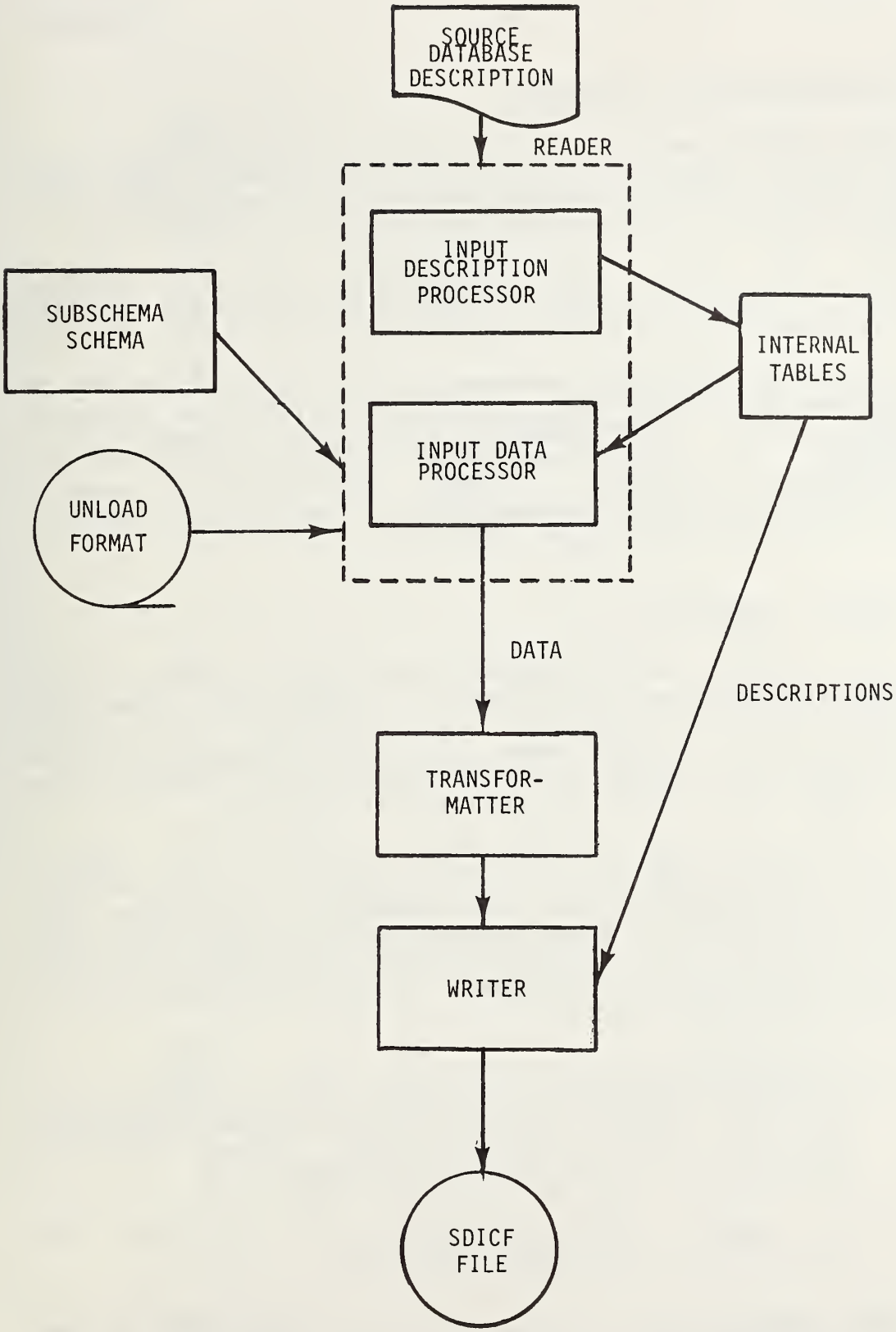
1) An Input Description Processor which processes the schema definition as defined in the source DBMS schema definition language and generates internal tables to be used in accessing and transforming the data; and

2) An Input Data Processor which reads the source data using the internal tables.

In the OTS, the READER "accesses" the data in the SDICF file. Rather than being DBMS dependent as the ITS READER is, the OTS READER is SDICF dependent and can be optimized accordingly. The description units and the linearized data units make the OTS READER component relatively easy to implement.



COMPONENTS OF ITS AND OTS
FIGURE A-5



EXPANDED ITS READER COMPONENT
FIGURE A-6

3.3 TRANSFORMATTER COMPONENT

The Transformatter performs the transformations on the data that are required to restructure the data for the SDICF file. Basically this component performs the inter-data model mappings required for the implicit SDICF data model.

In the OTS, this component performs similar transformations on the data in order to restructure the data for the data model of the target database management system.

The TRANSFORMATTER components of both subsystems may be modularized since they deal with common interfaces. However, both of them are "DBMS" dependent and their functionality changes in the source and target environment. In the ITS this component performs a simple restructuring function while in the OTS this component may be required to perform a more powerful restructuring function.

3.4 WRITER COMPONENT

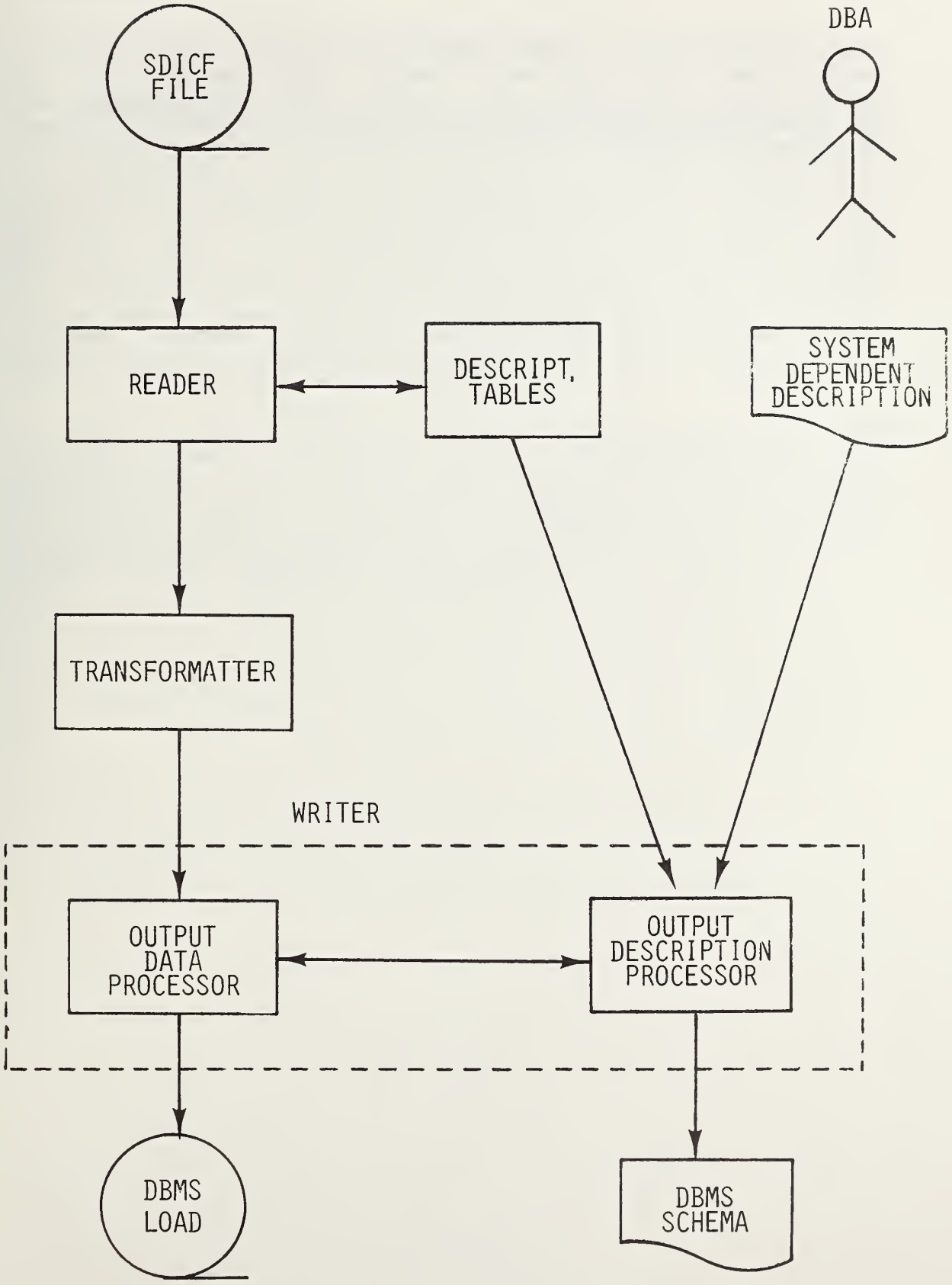
In the ITS, this component takes the data generated by the TRANSFORMATTER and creates the SDICF file. The ITS WRITER'S output is specific to the SDICF format and is independent of the source DBMS environment. As with the OTS READER, the SDICF characteristics simplify the design of the ITS WRITER making it relatively easy to implement.

In the OTS, however, the WRITER component must create a specific target DBMS load tape from the data transformed by the OTS TRANSFORMATTER using the particular DBMS schema. The diversity in DBMS load tape formats does not make this task easy or generalizable. In parallel with the READER component of the ITS, the WRITER component of the OTS consists of two sub-components (see Figure A-7):

1) An Output Description Processor which produces an initial target database schema from the SDICF description units. The additional system dependent descriptions are provided by the DBA; and

2) An Output Data Processor which performs the population of the DBMS load tape.

Two important factors contribute to the complexity of the Output Data Processor. The first factor is the use of the pointer output media mechanisms to implement relationships which have to be created from the SDICF's own pointer mechanism. The second factor is the issue of ordering of data instances for the target DBMS load program. In addition to the "load order" there may also be a sequence for the



EXPANDED OTS WRITER COMPONENT

FIGURE A-7

records specified in the target schema which would require a sort of the data instances. Overall the designer of the Output Data Processor must realize that this sub-component may require multiple passes of the data in order to create the DBMS load tape.

4. OBSERVATIONS

Several important observations can be made from the discussion of the input and output subsystems. First, given the alternative identified in Section 2.3, both the ITS and the OTS are DBMS and computer system dependent, but they are not database dependent. This suggests that only one ITS and one OTS need to be constructed for a single type of DBMS used on similar series of computer systems. Second, the OTS is primarily driven by the data description units contained in the SDICF and partially driven by the target DBMS. Finally, in both the source and the target environment, the DBA is needed to provide specifications and to control the process of data interchange.

Even within the design approach, discussed within this appendix, there are still many implementations. The implementation decision depends on the generality of the interchange system required. In Appendix B, an example processing sequence is shown which demonstrates a feasible, though minimal, design of an ITS and an OTS using the "middle ground" approach suggested here.

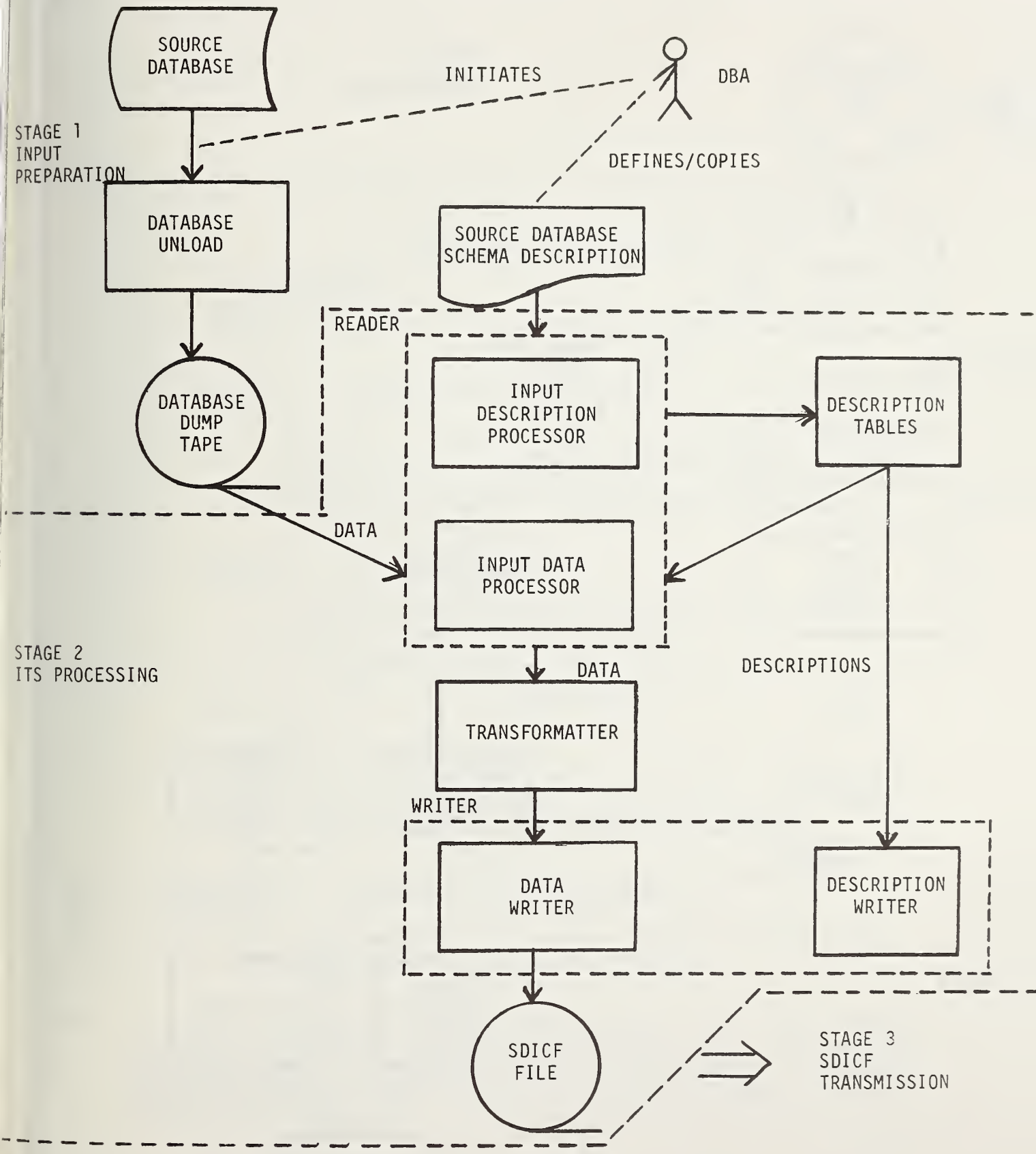
APPENDIX B

AN EXAMPLE

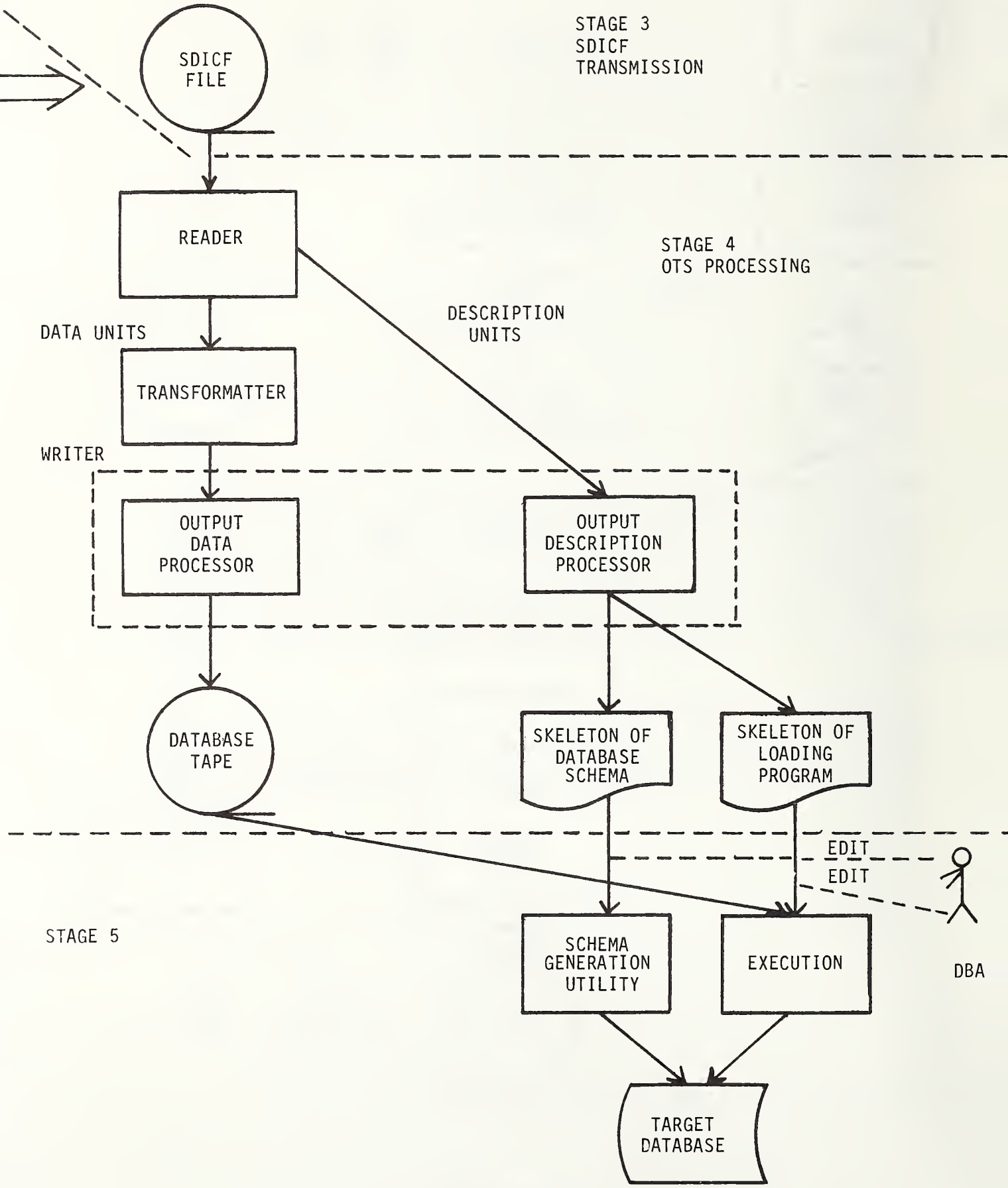
1. INTRODUCTION

A complete example is developed to demonstrate a possible processing sequence for a basic Input Transformation Subsystem (ITS) and Output Transformation Subsystem (OTS) as specified in Appendix A. The following example describes the interchange process between a hierarchical Database Management System (IMS) in the source environment and a network Database Management System (IDMS) in the target environment. The sample database is given in Figures 4-4 and 4-5.

The basic design assumes that the input to the ITS is the source database "unload format", and the output from the OTS is a sequential data stream that can be used by a DBMS utility to populate the target database. The processing flow of the ITS and the OTS is shown schematically in Figures B-1 and B-2.



ITS PROCESSING
FIGURE B-1



OTS PROCESSING
FIGURE B-2

2. DATA INTERCHANGE PROCESS

Generally, the data interchange process consists of five stages:

STAGE 1: SOURCE INPUT PREPARATION

1) Securing access to the source database description in the source DBMS schema definition language; and
2) Generation of the source database dump tape.

STAGE 2: ITS PROCESSING

1) Construction of SDICF description units, and
2) Generation of SDICF data units.

STAGE 3: SDICF FILE TRANSMISSION

STAGE 4: OTS PROCESSING

1) Input and processing of SDICF description units
2) Transformation of the SDICF data units.

STAGE 5: TARGET DATABASE CREATION

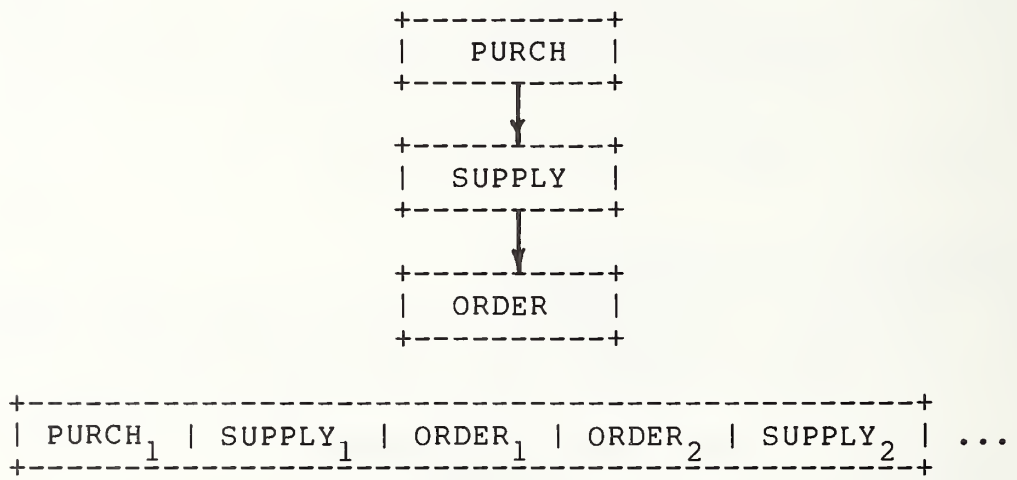
1) Preparation of the target database schema, and
2) Population of the target database

The following example demonstrates an interchange process involving a hierarchical database (IMS) and network database (IDMS). Each stage in the process will be discussed in further detail. The example used for discussion is shown in Section 4.2.2.

2.1 SOURCE INPUT PREPARATION

The preparation includes two distinct steps: securing access to the source database description in the existing source DBMS data description language and the creation of the source database dump tape using the DBMS unload utilities.

The dump format of an IMS database is partitioned into a segment-by-segment sequential organization. The hierarchical relationship among segments is preserved by their being ordered from top to bottom with respect to parent-child relationships and from left to right for the instances of the same segment in the hierarchy. [IBM, IMS/VS Utilities Reference Manual, GC33-0009.] A general view of the organization for this example can be found in Figure B-3.



Example of Preorder Dump Tape Sequence

FIGURE B-3

Each segment contains three (3) fields: an unload control field, an IMS-system-dependent field, and an actual segment data field.

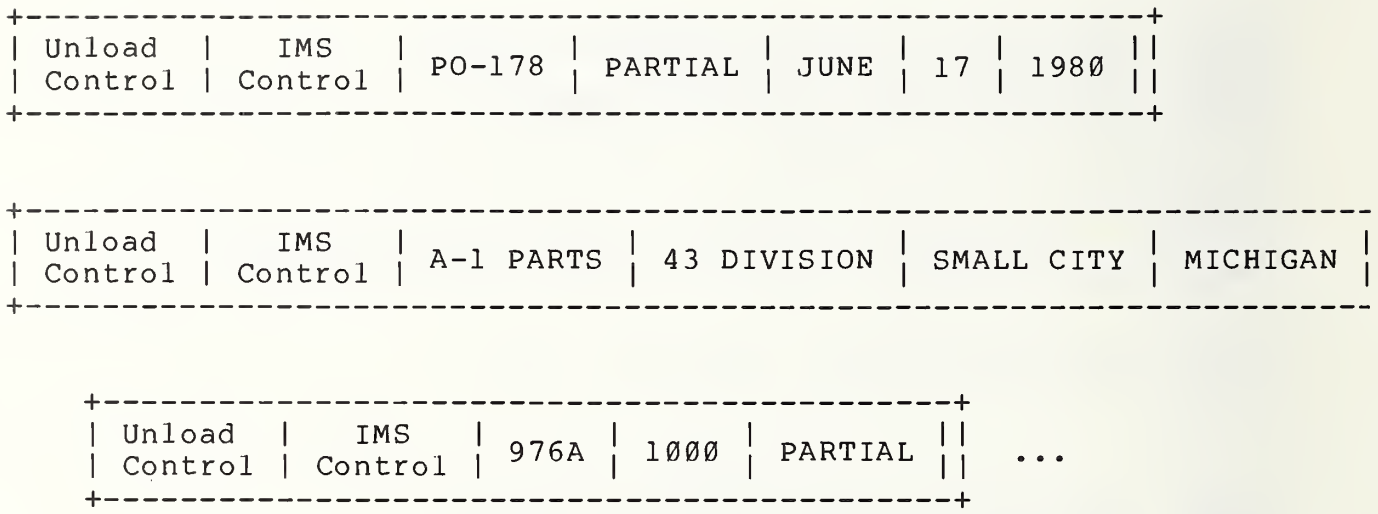
The source database description consists of a set of statements logically describing the database. These statements are written using the IMS hierarchical schema definition language. The source database description for this example is shown in Figure B-4. An example of the IMS dump tape is shown in Figure B-5.

```
DBD     NAME=SUPPLIER-PURCHASE-ORDER-DATABASE
SEGM    NAME=PURCHASE-ORDER, BYTES=4
FIELD   NAME=(PO#,SEQ), BYTES=17, START=1
FIELD   NAME=STATUS, BYTES=17, START=18
   .
   .
   .
SEGM    NAME=SUPPLIER, PARENT=PURCHASE-ORDER, BYTES=90


FIELD   NAME=(NAME,SEQ,M), BYTES=30, START=1
   .
   .
   .
SEGM    NAME=ORDER, PARENT=SUPPLIER, BYTES=32
FIELD   NAME=(PART#,SEQ), BYTES=10, START=1
   .
   .
   .
SEGM    NAME=BACKORDER, PARENT=ORDER, BYTES=15
FIELD   NAME=(BACKORDER#,SEQ), BYTES=10, START=1
   .
   .
   .
```

IMS Source Database Description

FIGURE B-4



Example IMS Dump Tape

FIGURE B-5

2.2 INPUT TRANSFORMATION SUBSYSTEM (ITS)

The ITS accepts two inputs: the database dump tape and the source database description, and converts them into an SDICF file. The basic components of the ITS include database description tables, a READER, a TRANSFORMATTER, and a WRITER.

A database description table is analogous to a symbol table which contains all the necessary attributes of the "names" defined in the source database description. The tables are generated by the Input Description Processor to provide the patterns for the Input Data Processor to match the record instances read from the unload tape and to provide sufficient information to construct the SDICF description units. Figure B-6(a and b) illustrates the database description tables for the sample IMS database.

The ITS processing consists of two steps: description unit construction and data unit construction.

2.2.1 Description Unit Construction

(1) SOURCE DESCRIPTION ANALYZER

This can be viewed as a simple parser analyzing the statements in the IMS source database description in order to produce simple parsing tables (description tables).

* input: source database description
* output: description tables

In particular, this input description analyzer for IMS schema can be designed as a 3-pass analyzer.

1st Pass: --Read each Segment description from the database description, and pick out all Fields

Construct the attribute description table by filling in field names, lengths, and types; and assign a unique identification number.

2nd Pass: --Read each segment description again.

--Pick out Segment Names

--Match all fields of a segment against the attribute description table to obtain identification numbers for each field

ATTRIBUTE NAME	TYPE	ID
PO#	CHARACTER 7	1
STATUS	CHARACTER 7	2
MONTH	CHARACTER 9	3
DAY	FIXED 2	4
YEAR	FIXED 4	5
NAME	CHARACTER 30	6
ADDRESS	CHARACTER 30	7
CITY	CHARACTER 15	8
STATE	CHARACTER 15	9
PART#	CHARACTER 10	10
QUANTITY	FIXED 5	11
STATUS	CHARACTER 7	12
BACKORDER#	CHARACTER 7	13
QUANTITY	FIXED 5	14

Attribute Description Table

ITS Description Tables

FIGURE B-6(a)

ENTITY NAME	ID	ATTRIBUTE LIST
PURCHASE-ORDER	1	AT1;AT2;AT3;AT4;AT5;
SUPPLIER	2	AT6;AT7;AT8;AT9;
ORDER	3	AT10;AT11;AT12;
BACKORDER	4	AT13;AT14;AT15;

Entity Description Table

ASSOCIATION NAME	ID	MEMBER	OWNER
SYS-PUR	1	1	SYSTEM
PUR-SUP	2	2	1
SUP-ORD	3	3	2
ORD-BAC	4	4	3

Association Description Table

ITS Description Tables

FIGURE B-6(b)

--Construct the entity description table by filling in segment name, the item list of field identification numbers and assigning a unique identification number to each segment.

3rd Pass: --Reread all segment descriptions and keep track of PARENT phrase.

--Construct the association description table by generating association name using the first 3 characters of a PARENT segment name concatenated with the first 3 characters of the child's segment name; and assigning unique identification number to each association name. OWNER and MEMBER of the association are assigned the identification number of the PARENT segment and the CHILD segment, respectively.

(2) DESCRIPTION UNIT WRITER

This writer uses the parsed elements in the description tables, formats them into SDICF description units, and writes them out to the SDICF file.

* input: description tables
* output: SDICF description units in SDICF file

2.2.2 Data Unit Construction

(1) SOURCE DATA PROCESSOR

This processor reads data records from the IMS dump tape, matches them using the description tables, and passes these parsed records to the Transformatter. In other words, it pre-processes the IMS database record instances into a form acceptable to the transformatter.

* input: IMS dump tape, description tables
* output: parsed database records

/* Procedure to read instance-by-instance the segments from IMS database dump tape. */

Read Control FIELD of an IMS segment instance

Retrieve its SEGMENT NAME name

and LENGTH OF SEGMENT ℓ in number of bytes

Read ℓ bytes beyond the Control FIELD as DATA FIELD

Search description tables for name.

Each field of the segment DATA FIELD is recognized using data type attribute information. The appropriate attribute information is obtained from the attribute description table by referencing the attributes in the entity description table for the segment.

(2) TRANSFORMATTER

This component reorganizes the IMS database record instances into SDICF data units which conform to the description units previously produced. It is a specialized program for IMS-SDICF conversion. Therefore, the SDICF format is hardcoded and needs no input for such format. It reads the IMS records in parsed form and transforms them into SDICF data units for the WRITER to output to tape (SDICF file).

* input: IMS records in parsed form
* output: SDICF data units

(3) WRITER

This is a specialized tape writer in which the data units from the Transformatter are organized into blocks

before actual writing with tape marks.

* Input: SDICF data units
* Output: SDICF file on tape

2.3 SDICF FILE TRANSMISSION

The resulting SDICF file is shown in Figure B-7. The transmission of the SDICF file can be done by on-line data transmission, off-line mailing, or some other method.

```
DESCRIPTION;2;Supplier-Purchase Order DB;810101@


AT1;PO?#;CH7@
AT2;STATUS;CH7@
AT3;MONTH;CH9@
AT4;DAY;FI2@
AT5;YEAR;FI4@
AT6;NAME;CH30@
AT7;ADDRESS;CH30@
AT8;CITY;CH15@
AT9;STATE;CH15@
AT10;PART?#;CH10@
AT11;QUANTITY;FI5@
AT12;STATUS;CH7@
AT13;BACKORDER?#;CH10@
AT14;QUANTITY;FI5@


EN1;PURCHASE-ORDER;AT1;AT2;AT3;AT4;AT5;AS1,2@
EN2;SUPPLIER;AT6;AT7;AT8;AT9;AS2,3@
EN3;ORDER;AT10;AT11;AT12;AS3,4@
EN4;BACKORDER;AT13;AT14;AS4@


AS1;SYS-PUR;OWSY;ME1@
AS2;PUR-SUP;OW1;ME2@
AS3;SUP-ORD;OW2;ME3@
AS4;ORD-BAC;OW3;ME4@
#


DATA;2;Supplier-Purchase Order DB;810101@


ENSY;AS1;1@
EN1;1;AT1;PO-178;AT2;PARTIAL;AT3;JUNE;AT4;17;AT5;1980;
    AS1;SY;AS2;2@
EN2;2;AT6;A-1 PARTS;AT7;43 DIVISION;AT8;SMALL CITY;AT9;MICHIGAN;
    AS2;1;AS3;3@
EN3;3;AT11;17654;AT12;2000;AT13;FILLED;AS3;4;AS4;3@
EN3;4;AT11;976A;AT12;1000;AT13;PARTIAL;AS3;2;AS4;5@
EN4;5;AT13;BO-178;AT14;50;AS4;4@
#
```

SDICF File Representing Hierarchical Database

FIGURE B-7

2.4 OTS PROCESSING

The OTS accepts the SDICF file as input and produces a database tape, a skeleton of the target database schema, and a skeleton of the target database loading program. The basic components of the OTS are the READER, the TRANSFORMATTER, and the WRITER.

The transformation from a SDICF into an IDMS loadable format consists of two stages: description processing and data transformation of the SDICF file. Recall that the description units sequentially precede the data units.

2.4.1 Description Processing

This step produces a skeleton IDMS schema, with the missing information to be provided by the target DBA.

(1) DESCRIPTION UNIT READER

This component reads the tape sequentially, deblocking the data into logical records (or units) which will be used for input into the output description processor.

* Input: SDICF tape
* Output: SDICF description units

(2) TARGET DESCRIPTION ANALYZER

This analyzer compiles the description units into a skeleton IDMS schema and creates an IDMS loading program.

* Input: SDICF description units
* Output: - IDMS schema skeleton
  - IDMS loading program skeleton

The compilation of the description units into an IDMS schema can be viewed as a 2-pass process. The first pass scans through all description units to determine the number of Record types from Entity units, and Set types from Association units. Then the second pass fills in the details of Record types and Set types obtainable from description units. For instance, the data type of each item in the record type can be found in attribute units. The empty schema of Figure B-8(a through c) is used to create the partially completed schema of Figure B-9(a through c) from the SDICF file as indicated by "[ ]". The DBA then edits this schema (indicated by "< >") to create the complete schema of Figure B-10(a through c).

However, the generation of the loading program skeleton is more complicated. As illustrated in the IDMS utility manual [Cullinane Corporation, IDMS Utilities, Revision 1, Release 5.5], one can use the description units to edit the given program by replacing with appropriate attribute names and associated type clauses. This process is more complex because of the unknown number of entity units and attribute units. Therefore the basic skeleton has to be flexible in order to allow extension.

2.4.2 Data Transformation

The main objective of this stage is to transform the SDICF data units into IDMS loadable form.

(1) DATA UNIT READER

Similar to the description unit reader, this reader will deblock and read data units record-by-record.

* Input: SDICF file
* Output: SDICF data units

(2) TRANSFORMATTER

The transformatter is a "hardcoded" program tailored to transform data units in the SDICF file into IDMS loadable format. It will reorganize items in the record instances and present them to the output data processor.

* Input: SDICF data units
* Output: IDMS loadable record instances

(3) TARGET DATA PROCESSOR

This processor handles the actual editing of the record instances before performing the tape output sequence such as tape labelling, blocking of logical records into physical ones, etc. The output tape is in loadable form.

* Input: IDMS loadable record instances
* Output: IDMS loadable database tape

```
*
* IDMS  SCHEMA DESCRIPTION
*
    SCHEMA DESCRIPTION
    SCHEMA NAME IS                  RANGE IS    THRU    .
    AUTHOR
    DATE
    INSTALLATION
    REMARKS
*
* AREA DESCRIPTION
*
    AREA DESCRIPTION.
    AREA NAME IS                    RANGE IS    THRU    .
    AREA NAME IS                    RANGE IS    THRU    .
    AREA NAME IS                    RANGE IS    THRU    .
```

Empty IDMS Schema

FIGURE B-8(a)

```
*
* RECORD DESCRIPTION
*
    RECORD DESCRIPTION
    RECORD NAME
    RECORD ID
    LOCATION MODE          USING

    WITHIN
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
*
    RECORD NAME
    RECORD ID
    LOCATION MODE IS
    WITHIN
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
*
    RECORD NAME
    RECORD ID
    LOCATION MODE          USING

    WITHIN
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
*
    RECORD NAME
    RECORD ID
    LOCATION MODE          USING

    WITHIN
         Ø5                                PIC
         Ø5                                PIC
         Ø5                                PIC
```

Empty IDMS Schema

FIGURE B-8(b)

```
*
* SET DESCRIPTION
*
    SET DESCRIPTION
*
    SET NAME IS
    ORDER IS
    MODE IS
    OWNER IS
    MEMBER IS
*
    SET NAME IS
    ORDER IS
    MODE IS
    OWNER IS
    MEMBER IS
*
    SET NAME IS
    ORDER IS
    MODE IS
    OWNER IS
    MEMBER IS
*
    SET NAME IS
    ORDER IS
    MODE IS
    OWNER IS
    MEMBER IS
```

Empty IDMS Schema

FIGURE B-8(c)

```
*
* IDMS  SCHEMA DESCRIPTION
*
    SCHEMA DESCRIPTION.
    SCHEMA NAME IS              RANGE IS    THRU    .
    AUTHOR.                  .
    DATE.
    INSTALLATION.
    REMARKS.
*
* AREA DESCRIPTION
*
    AREA DESCRIPTION.
    AREA NAME IS                RANGE IS    THRU    .
    AREA NAME IS                RANGE IS    THRU    .
    AREA NAME IS                RANGE IS    THRU    .
```

Partially Complete IDMS Schema

FIGURE B-9(a)

```
*
* RECORD DESCRIPTION
*
    RECORD DESCRIPTION.
    RECORD NAME [ PURCHASE-ORDER ] .
    RECORD ID [ ØØ1 ] .
    LOCATION MODE          USING

    WITHIN                      .
         Ø5  [ PO# ]                    PIC [ X(17) ] .
         Ø5  [ STATUS ]                 PIC [ X(17) ] .
         Ø5  [ MONTH ]                  PIC [ X(9) ] .
         Ø5  [ DAY ]                    PIC [ 99 ] .
         Ø5  [ YEAR ]                   PIC [ 9999 ] .
*
    RECORD NAME [ SUPPLIER ] .
    RECORD ID [ ØØ2 ] .
    LOCATION MODE IS                        .
    WITHIN                      .
         Ø5  [ NAME ]               PIC [ X(3Ø) ] .
         Ø5  [ ADDRESS ]            PIC [ X(3Ø) ] .
         Ø5  [ CITY ]               PIC [ X(15) ] .
         Ø5  [ STATE ]              PIC [ X(15) ] .
*
    RECORD NAME  [ ORDER ] .
    RECORD ID  [ ØØ3 ] .
    LOCATION MODE          USING

    WITHIN                      .
         Ø5  [ PART# ]                PIC [ X(1Ø) ] .
         Ø5  [ QUANTITY ]             PIC [ 9(5) ] .
         Ø5  [ STATUS ]               PIC [ X(17) ] .
*
    RECORD NAME  [ BACKORDER ] .
    RECORD ID  [ ØØ4 ] .
    LOCATION MODE          USING

    WITHIN                      .
         Ø5  [ BACKORDER# ]         PIC [ X(7) ] .
         Ø5  [ QUANTITY ]           PIC [ 9(5) ] .
```

Partially Complete IDMS Schema

FIGURE B-9(b)

```
*
* SET DESCRIPTION
*
    SET DESCRIPTION .
*
    SET NAME IS [ SYS-PUR ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS           .
    OWNER IS [ SYSTEM ] .
    MEMBER IS [ PURCHASE-ORDER ] .


*
    SET NAME IS [ PUR-SUP ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS           .
    OWNER IS [ PURCHASE-ORDER ] .
    MEMBER IS [ SUPPLIER ]
*
    SET NAME IS [ SUP-ORD ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS           .
    OWNER IS [ SUPPLIER ] .
    MEMBER IS [ ORDER ]
*
    SET NAME IS [ SUP-BAC ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS           .
    OWNER IS [ SUPPLIER ] .
    MEMBER IS [ BACKORDER ]
```

Partially Complete IDMS Schema

FIGURE B-9(c)

```
*
* IDMS  SCHEMA DESCRIPTION
*
    SCHEMA DESCRIPTION.
    SCHEMA NAME IS < IMSIDMS > RANGE IS    THRU    .
    AUTHOR.  < D.K. NGUYEN > .
    DATE.
    INSTALLATION.
    REMARKS.  < THIS IS SCHEMA FOR THE INTERCHANGED IMS
                DATABASE > .
*
* AREA DESCRIPTION
*
    AREA DESCRIPTION.
    AREA NAME IS < SUPPLIER-AREA > RANGE IS     THRU     .
    AREA NAME IS < ORDER-AREA > RANGE IS     THRU     .
    AREA NAME IS < PURCHASE-AREA > RANGE IS     THRU     .
```

Complete IDMS Schema

FIGURE B-1Ø(a)

```
*
* RECORD DESCRIPTION
*
    RECORD DESCRIPTION.
    RECORD NAME [ PURCHASE-ORDER ] .
    RECORD ID [ ØØ1 ] .
    LOCATION MODE < CALC > USING < PO#
                   DUPLICATES NOT ALLOWED > .
    WITHIN < SUPPLIER-AREA > .
         Ø5  [ PO# ]              PIC [ X(17) ] .
         Ø5  [ STATUS ]           PIC [ X(17) ] .
         Ø5  [ MONTH ]            PIC [ X(9) ] .
         Ø5  [ DAY ]              PIC [ 99 ] .
         Ø5  [ YEAR ]             PIC [ 9999 ] .
*
    RECORD NAME [ SUPPLIER ] .
    RECORD ID [ ØØ2 ] .
    LOCATION MODE IS < VIA PUR-SUP SET > .
    WITHIN < SUPPLIER-AREA > .
         Ø5  [ NAME ]           PIC [ X(3Ø) ] .
         Ø5  [ ADDRESS ]        PIC [ X(3Ø) ] .
         Ø5  [ CITY ]           PIC [ X(15) ] .
         Ø5  [ STATE ]          PIC [ X(15) ] .
*
    RECORD NAME  [ ORDER ] .
    RECORD ID  [ ØØ3 ] .
    LOCATION MODE < CALC > USING < PART# DUPLICATES
                   NOT ALLOWED > .
    WITHIN  < ORDER-AREA > .
         Ø5  [ PART# ]              PIC [ X(1Ø) ] .
         Ø5  [ QUANTITY ]           PIC [ 9(5) ] .
         Ø5  [ STATUS ]             PIC [ X(17) ] .
*
    RECORD NAME  [ BACKORDER ] .
    RECORD ID  [ ØØ4 ] .
    LOCATION MODE < CALC > USING < BACKORDER#
                   DUPLICATES NOT ALLOWED > .
    WITHIN  < ORDER-AREA > .
         Ø5  [ BACKORDER# ]        PIC [ X(7) ] .
         Ø5  [ QUANTITY ]          PIC [ 9(5) ] .
```

Complete IDMS Schema

FIGURE B-1Ø(b)

```
*
* SET DESCRIPTION
*
    SET DESCRIPTION .
*
    SET NAME IS [ SYS-PUR ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS < CHAIN > .
    OWNER IS [ SYSTEM ] .
    MEMBER IS [ PURCHASE-ORDER ] .
         < MANDATORY >
         < NEXT DBKEY POSITION IS 1 >
         < ASCENDING KEY IS PO#
              DUPLICATES NOT ALLOWED > .
*
    SET NAME IS [ PUR-SUP ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS < CHAIN > .
    OWNER IS [ PURCHASE-ORDER ] .
    MEMBER IS [ SUPPLIER ] < MANDATORY >
                < AUTOMATIC LINKED TO OWNER
                  NEXT DBKEY POSITION IS 1
                  OWNER DBKEY POSITION IS 2 > .
*
    SET NAME IS [ SUP-ORD ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS < CHAIN > .
    OWNER IS [ SUPPLIER ] .
    MEMBER IS [ ORDER ] < MANDATORY >
                < AUTOMATIC
                  NEXT DBKEY POSITON 1
                  ASCENDING PART# DUPLICATES
                    NOT ALLOWED > .
*
    SET NAME IS [ SUP-BAC ] .
    ORDER IS [ SYSTEM-DEFAULT ] .
    MODE IS < CHAIN > .
    OWNER IS [ SUPPLIER ] .
    MEMBER IS [ BACKORDER ] < MANDATORY >
                  < AUTOMATIC
                    NEXT DBKEY POSITION 1
                    ASCENDING KEY IS BACKORDER#
                    DUPLICATES NOT ALLOWED > .
```

Complete IDMS Schema

FIGURE B-10(c)

2.5 TARGET DATABASE CREATION

This final step uses the three outputs of the OTS: the skeleton of the IDMS schema, the skeleton of the IDMS loading program, and the IDMS loadable database tape to create a new database. It consists of two steps:

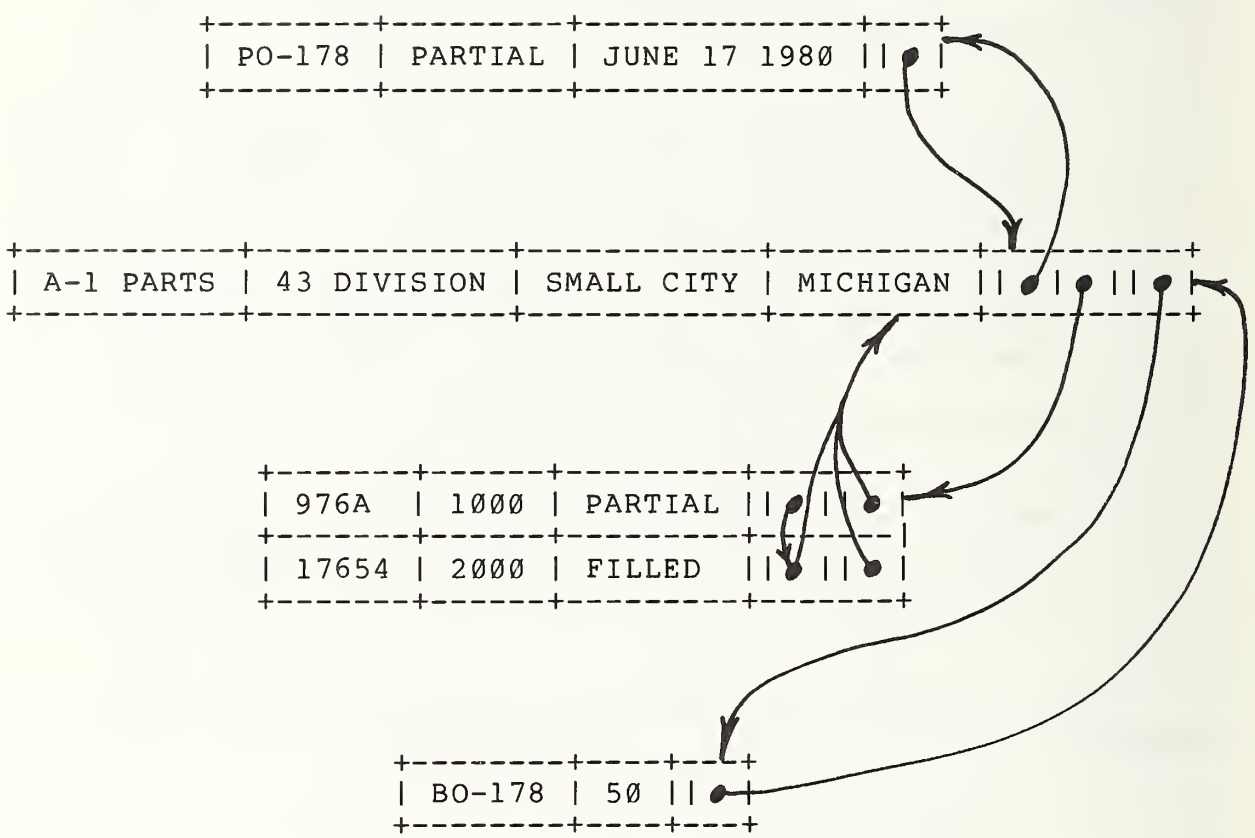
1) IDMS schema preparation and generation

   Due to the insufficient information provided by the IMS schema definition facility, the skeleton of IDMS schema will only contain partial descriptions of the target database. DBA intervention is required here to fill in the missing information (i.e., LOCATION MODE, etc.) and special restrictions (for security, integrity, etc.) in the schema. After this is done, the DBA can use the enhanced database schema to generate a target IDMS schema.

2) Population program preparation and execution.

   A similar type of modification or fill-in must be made to the skeleton of the loading program, and then, together with the loadable database, the DBA can create a new IDMS database in the target environment.

The created IDMS database is illustrated in Figure B-11.



Example of IDMS Database Instances

FIGURE B-11

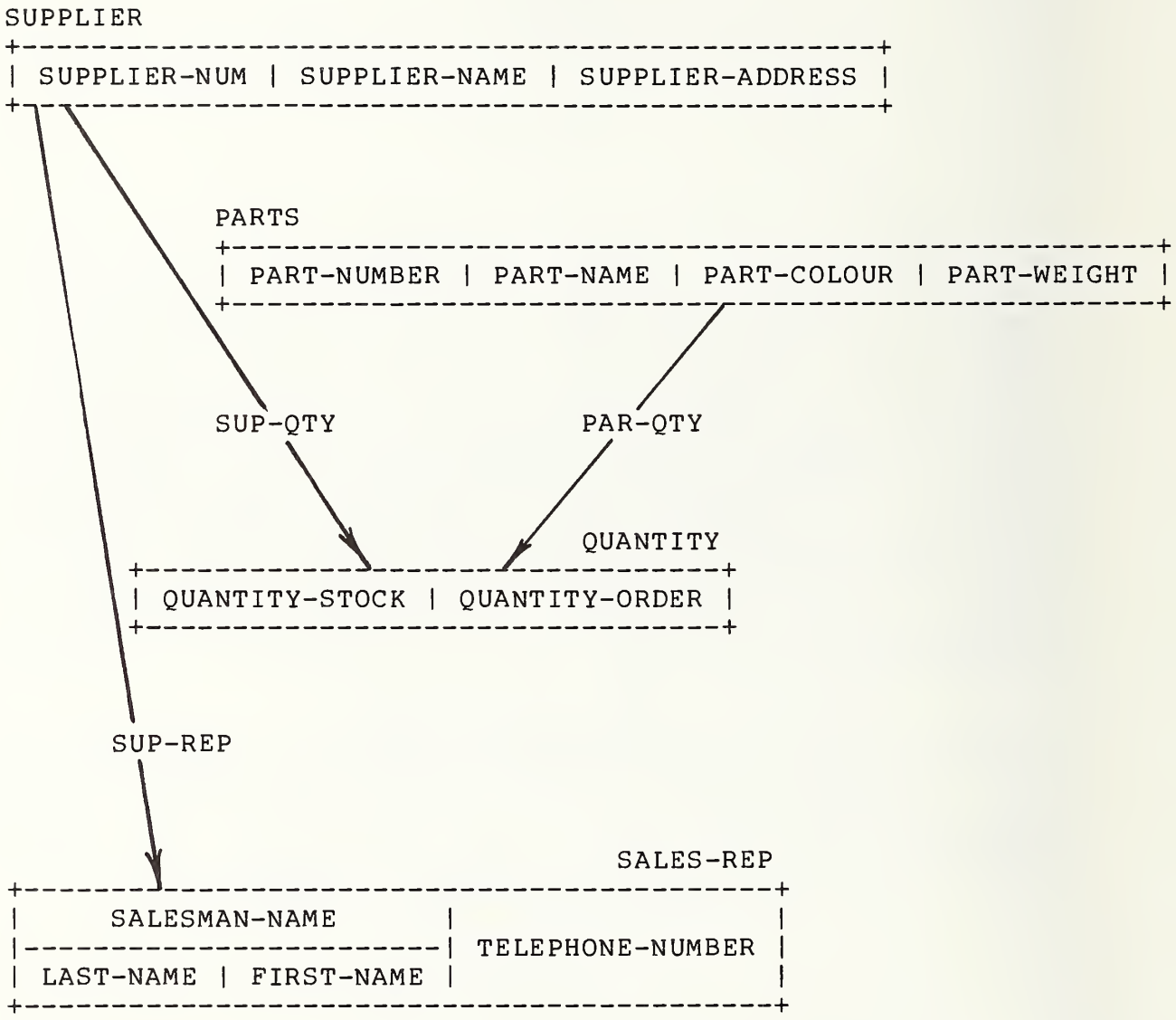
APPENDIX C

AN EXAMPLE

1. INTRODUCTION

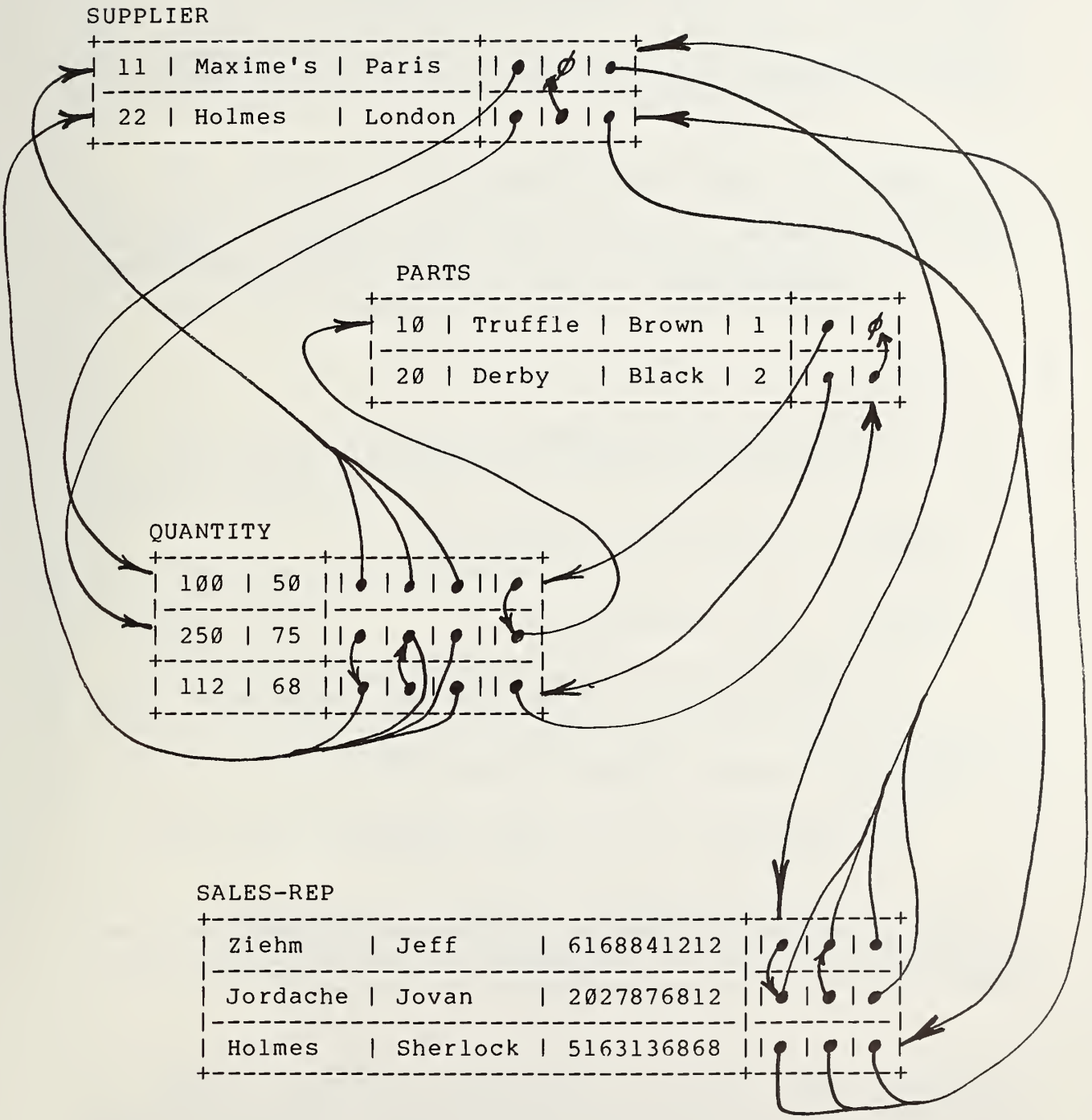
Like Appendix B, this Appendix develops a complete example to demonstrate a possible processing sequence for a basic Input Transformation Subsystem (ITS) and Output Transformation Subsystem (OTS) design as defined in Appendix A. The following example demonstrates an interchange process involving a network database management system (IDMS) in the source environment [IDMS Database Design and Definition Guide, Cullinane Corporation, Release 5.5], and a relational database management system (System R) in the target environment [System R: A Relational Approach to Database Management, ACM Transactions on Database Systems, Vol. 1, No. 2, June 1976]. Each stage in the process is discussed in further detail. The example used for discusssion is shown in Figures C-1 and C-2.

The basic design assumes that the input to the ITS is the source database "unload format", and the output from the OTS is a sequential data stream that can be populated into the target database using a target DBMS utility. The processing flow of the ITS and the OTS is shown diagramatically in Figures B-1 and B-2 in Appendix B.



Example Network Database Schema

FIGURE C-1



Example Database Source Instances

2. DATA INTERCHANGE PROCESS

To reiterate, the data interchange process consists in general of five stages:

STAGE 1: SOURCE INPUT PREPARATION

1) Securing access to the source database description in the source DBMS schema definition language; and
2) Generation of the source database unload tape.

STAGE 2: ITS PROCESSING

1) Construction of SDICF description units, and
2) Generation of SDICF data units.

STAGE 3: SDICF FILE TRANSMISSION

STAGE 4: OTS PROCESSING

1) Input and processing of SDICF description units
2) Transformation of the SDICF data units.

STAGE 5: TARGET DATABASE CREATION

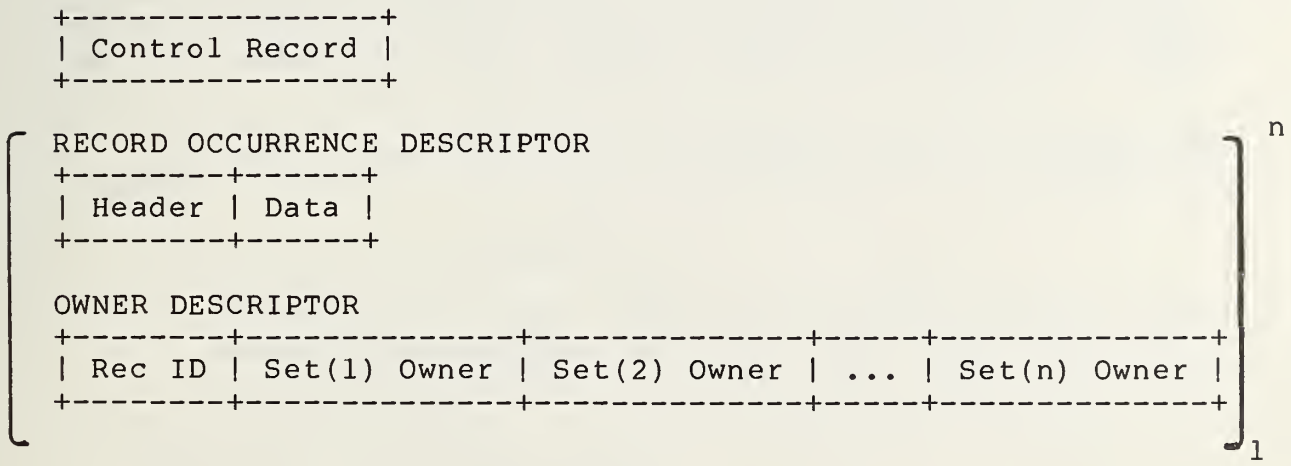
1) Preparation of the target database schema, and
2) Population of the target database

This example discusses an interchange process from an IDMS network database to a System R relational database. The sample database is given in Figures C-1 and C-2.

2.1 SOURCE INPUT PREPARATION

The preparation includes two distinct steps: securing access to the source database description in the existing source DBMS data definition language and the creation of the source database unload tape using the DBMS unload utilities. The source database description in the example utilizes the IDMS Schema DDL facility as shown in Figure C-4 (a through c). (All upper case words denote variable names, lower case text denotes the fixed syntax of IDMS DDL).

The unload format of an IDMS database, utilizing the utility IDMSUNLD, partitions the database into three types of records: a control record, record occurrence descriptor records, and owner descriptor records [IDMS Utilities, Revision 1, Release 5.5]. An overview of the organization of this unload tape is presented in Figure C-3.



Overview of IDMS Unload Tape Sequence

FIGURE C-3

This utility basically unloads the database sequentially on a record-by-record basis. It provides a control record for the entire unloaded file, and, for each record instance, a record occurrence descriptor and an owner descriptor. The control record contains such information as the length of the largest record occurrence, name of

subschema or schema, etc. The record occurrence descriptor consists of two parts: the fixed length header where a unique identification number, the record length and some IDMS physical storage details are stored ; and the variable length data field which has the data item values. The owner descriptor provides the identification of all set occurrences in which the record occurrence participates as a member. Each set occurrence is specified by the set name, the identification number of the owner record occurrence, and possibly the location mode of the owner record.

2.2 INPUT TRANSFORMATION SUBSYSTEM (ITS)

The ITS accepts the source database and produces the SDICF file. The inputs at this stage consist of the source database description, i.e. the IDMS Schema DDL, and the unloaded database on tape. The basic processing components are previously described in Appendix A. These components are used to construct description units and the data units for the SDICF. The following description is tailored for IDMS-SDICF transformations.

2.2.1 Description Unit Construction

This step involves the analysis of the source description and the compilation of the SDICF description units.

(1) SOURCE DESCRIPTION ANALYZER

This analyzer is a simple parser which reads the IDMS schema statements [Figure C-4] and constructs description tables (symbol tables) for the "names" declared in areas, record types, and set types. Representations of these description tables are given in Figure C-5(a and b).

A simple algorithm for this analyzer can be sketched as follows:

(a) Read Area Description

Construct an Area Description Table of areas with each entry composed of the area name, a unique identification number, and the page range associated with the area.

(b) Read Record Description

Construct an Attribute Description Table of all the items which have been declared with picture clauses. Each entry is composed of the item name, a uniquely assigned identification number, and the data type of the item.

Construct an Aggregate Description Table of all repeating groups with each entry composed of its name, a uniquely assigned id, and the components of the group. For nested groups the inner-most groups are entered before the outer groups. Using this approach, the component list will consist of either item identifiers or another group identifier.

(c) Reread Record Description

Construct a partial Entity Description Table with each entry composed of the record name, a uniquely assigned identification number, the area in which it is located, the location mode of either CALC or DIRECT, and all the components.

(d) Read Set Description

Construct an Association Description Table with each entry composed of the owner record identification number, member record identification number, ordering (and the associated key to be sorted on), and a uniquely assigned identification number. Edit the record table to fill in the location mode on all records of VIA-SET mode by inserting the unique set identification numbers.

(2) DESCRIPTION UNIT WRITER

This component constructs a control record for the SDICF file, formats each entry in the ITS Description Tables into SDICF format, and writes them. The description section of the example database is depicted at the top of Figure C-6.

2.2.2 Data Unit Construction

This step performs the following tasks:

- Reads and deblocks unload data on tape into logical records.

- Parses the data records into elements using the description tables.

- Formats these elements into SDICF data units.

(1) SOURCE DATA PROCESSOR

This component consists of a tape reader and a simple parser. The tape reader mainly reads the unload tape in units of physical blocks and deblocks them into logical records. These data records are then fed into the IDMS unload-format parser, which is a "hardcoded" program to analyze the IDMS unloaded record descriptors into their correspondent elements, i.e. tokens, using the description tables. The intermediate tokens are passed to the transformatter.

```
*
*  Schema description for a simple PART_SUPPLIER database,
*  devised by D.LEDER, modified and written by D.K.NGUYEN
*  using IDMS schema DDL.
*
schema description.
schema name is PARTSUPP version 1.
author. D-K-NGUYEN.
date. 03/08/81.
installation. DSRG.
remarks. SCHEMA DESCRIPTION FOR A SIMPLE IDMS NETWORK DATABASE.
*
*   File description logically situates the boundary location
*  where the areas can be placed in logical block number range.
*
file description.
*
file name ALL-NAMES assign to ALL-NAMES.
*
file name ALL-PARTS assign to ALL-PARTS.
*
*   Area description defines the logical memory page range
*  assigned to the range of block number range in the file(s)
*  where the record occurrences are residing.
*
area description.
*
area name is SUPPLIER-SALES
     range is 1001  through  1101
     within file ALL-NAMES from 1 through 101.
*
area name is PART-QUANT
     range is 2001 through 2201
     within file ALL-PARTS from 1 through 201.
```

IDMS Schema

FIGURE C-4(a)

```
*
*   Record description defines the record structure in terms of
*  of its components, basically a hierarchical structure.
*
record description.
*
record name is SUPPLIER.
record id is 101.
location mode is CALC using SUPPLIER-NUM.
within SUPPLIER-SALES area.
     05  SUPPLIER-NUM                  PIC  9(6) .
     05  SUPPLIER-NAME                 PIC  X(20).
     05  SUPPLIER-ADDRESS              PIC  X(40).
*
record name is SALES-REP.
record id  102.
location mode is via SUP-REP set.
within SUPPLIER-SALES area.
     05  SALESMAN-NAME
       10  LAST-NAME                   PIC  X(20).
       10  FIRST-NAME                  PIC  X(10).
     05  TELEPHONE-NUMBER              PIC  X(10).
*
record name is PARTS.
record id  200.
location mode is CALC using PART-NUMBER.
within PART-QUANT area.
     05  PART-NUMBER                 PIC  9(9) .
     05  PART-NAME                   PIC  X(20).
     05  PART-COLOUR                 PIC  X(10).
     05  PART-WEIGHT                 PIC  9999 .
*
record name is QUANTITY.
record id 205.
location mode is VIA PAR-QTY set.
within PART-QUANT area.
     05  QUANTITY-STOCK              PIC  9(10).
     05  QUANTITY-ORDER              PIC  9(10).
```

IDMS Schema

FIGURE C-4(b)

```
*
*   Set description defines all relationships among the record-types
*  defined above, and also specified how each set-type occurence
*  is linked (IDMS specialty).
*
set description.
*
set name is SUP-QTY.
order is LAST.
mode is CHAIN linked to PRIOR.
owner is SUPPLIER      NEXT dbkey position is 1
                       PRIOR dbkey position is 2.
member is QUANTITY     NEXT  dbkey position is 1
                       PRIOR dbkey position is 2
     linked to owner  OWNER dbkey position is 3
     MANDATORY automatic
     ascending key is QUANTITY-STOCK
     duplicates is FIRST.
*
set name is SUP-REP.
order is SORTED.
mode is CHAIN.
owner is SUPPLIER    NEXT  dbkey position is 3.
member is SALES-REP  NEXT  dbkey position is 1
                     PRIOR dbkey position is 2
    linked to owner  OWNER dbkey position is 3
    OPTIONAL automatic
    ascending key is LAST-NAME
    duplicates NOT allowed.
*
set name is PAR-QTY.
order is FIRST.
mode is CHAIN linked to PRIOR.
owner is PARTS   NEXT  dbkey position is 1
                 PRIOR dbkey position is 2.
member is QUANTITY
                 NEXT  dbkey position is 4
    MANDATORY automatic
    ascending key is QUANTITY-STOCK
    duplicates is LAST.
*
*  ----- SYSTEM-set types are not specified in IDMS database.
*
```

IDMS Schema

FIGURE C-4(c)

Area-name	Area-id#
SUPPLIER-SALES	1
PART-QUANT	2

Area Description Table

Attribute Name	Att-id#	Type
SUPPLIER-NUM	1	FIXED 6
SUPPLIER-NAME	2	CHARACTER 20
SUPPLIER-ADDRESS	3	CHARACTER 10
LAST-NAME	4	CHARACTER 20
FIRST-NAME	5	CHARACTER 10
TELEPHONE-NUMBER	6	CHARACTER 10
PART-NUMBER	7	FIXED 9
PART-NAME	8	CHARACTER 20
PART-COLOUR	9	CHARACTER 10
PART-WEIGHT	10	FIXED 4
QUANTITY-STOCK	11	FIXED 10
QUANTITY-ORDER	12	FIXED 10

Attribute Description Table

ITS Description Tables

FIGURE C-5(a)

Aggregate Name	ag-id#	att-list	OCCURS
SALESMAN-NAME	1	AT4; AT5	NIL

Aggregate Description Table

Entity Name	en-id#	area-id#	att-list	mode
SUPPLIER	1	1	AT1; AT2; AT3	CALC AT1
SALES-REP	2	1	AG1; AT6	VIA SET 2
PARTS	3	2	AT7;AT8;AT9;AT10	CALC AT7
QUANTITY	4	2	AT11; AT12	VIA SET 3

Entity Description Table

Association Name	as-id#	Owner-id#	Member-id#	Order
SUP-QTY	1	1	4	LAST
SUP-REP	2	1	2	ASCEND AT4
PAR-QTY	3	3	4	FIRST

Association Description Table

ITS Description Tables

FIGURE C-5(b)

* input: - description tables, and
  - IDMS unload tape.
* output: IDMS record occurrence elements.

In particular, the IDMS unload-format parser can be designed based on the following algorithm:

- Read the record-occurrence descriptor headers (These headers are all fixed size).
- Extract the length of the data field of the record occurrence descriptor.
- Read the data field.
- Extract record identification to pattern match with the description tables for their attributes.
- Read the owner descriptor header.
- Extract the length of the descriptor.
- Read all the set-types that the record participates in.
- Use the Serial Numbers as the unique identification number for matching the owner record-types.
- Pass these elements, i.e. attribute descriptions, and association descriptions, to the Transformatter.

(2) TRANSFORMATTER

The task of this module is to format the tokens passed by the SOURCE DATA PROCESSOR into SDICF data units; such as prefixing the identification number with appropriate keywords, i.e. AT, EN, AG, etc..., to conform to the description units constructed earlier.

* input : record-type elements
* output: SDICF data units.

The data section resulting from the example database is given at the bottom of Figure C-6.

(3) WRITER

This is a specialized tape writer which blocks the data formatted by the above Transformatter according to the tape standard before writing the tape.

* Input: SDICF data units,
* Output: SDICF file on tape.

2.3 SDICF FILE TRANSMISSION

The SDICF file to be transmitted is given in Figure C-6. The transmission of the SDICF file can be done by the mailing of magnetic tape or some other method.

```
DESCRIPTION;100;PARTSUPP;810313@

AT1;SUPPLIER-NUM;FI6@
AT2;SUPPLIER-NAME;CH20@
AT3;SUPPLIER-ADDRESS;CH40@
AT4;LAST-NAME;CH20@
AT5;FIRST-NAME;CH10@
AT6;TELEPHONE-NUMBER;CH10@
AT7;PART-NUMBER;FI9@
AT8;PART-NAME;CH20@
AT9;PART-COLOUR;CH10@
AT10;PART-WEIGHT;FI4@
AT11;QUANTITY-STOCK;FI10@
AT12;QUANTITY-ORDER;FI10@

AG1;SALESMAN-NAME;AT4,AT5@

AR1;SUPPLIER-SALES@
AR2;PART-QUANT@

EN1;SUPPLIER;AR1;CA1;AT1;AT2;AT3;AS1,2,4@
EN2;SALES-REP;AR1;VI2;AG1;AT6;AS2@
EN3;PARTS;AR2;CA7;AT7;AT8;AT9;AT10;AS3,5@
EN4;QUANTITY;AR2;VI3;AT11;AT12;AS1,3@

AS1;SUP-QTY;OW1;ME4@
AS2;SUP-REP;OW1;ME2;AS4@
AS3;PAR-QTY;OW3;ME4@
AS4;SYS-SUP;OWSY;ME1@
AS5;SYS-PAR;OWSY;ME3@
#


DATA;100;PARTSUPP;810313@

ENSY;AS4;1;AS5;9@
EN1;1;AR1;AT1;11;AT2;MAXIME'S;AT3;PARIS;AS1;6;AS2;3;AS4;2@
EN2;3;AR1;AT4;ZIEHM;AT5;JEFF;AT6;6168841212;AS2;4@
EN2;4;AR1;AT4;JORDACHE;AT5;JOVAN;AT6;2027876812;AS2;1@
EN1;2;AR1;AT1;22;AT2;HOLMES;AT3;LONDON;AS1;7;AS2;5;AS4;SY@
EN2;5;AR1;AT4;HOLMES;AT5;SHERLOCK;AT6;5163136868;AS2;2@
EN3;9;AR2;AT7;10;AT8;TRUFFLE;AT9;BROWN;AT10;1;AS3;6;AS5;10@
EN4;6;AR2;AT11;100;AT12;50;AS1;1;AS3;7@
EN4;7;AR2;AT11;250;AT12;75;AS1;8;AS3;9@
EN3;10;AR2;AT7;20;AT8;DERBY;AT9;BLACK;AT10;2;AS3;8;AS5;SY@
EN4;8;AR2;AT11;112;AT12;68;AS1;2;AS3;10@
#
```

SDICF File for Example Database

FIGURE C-6

2.4 OTS PROCESSING

The OTS accepts the SDICF file as its only input and produces a database tape, a database schema, and a set of control commands that call the database population utility for creating the new target database. Due to the fact that the details of the output formats and commands for System R are proprietary, only an outline of the database tape and the database schema are shown here. They may not accurately represent the final products.

The processing in the OTS consists of two stages: description processing and data transformation of the SDICF file. The results of description processing are a database schema, a set of database population commands, and a schema table that is used by the output data processor for tailoring the data to conform to the schema.

The following section outlines the general mapping steps for transforming the SDICF description units (in this case representing a network database in the source) into a relational schema table.

2.4.1 SDICF ---> Relational Mapping Algorithm

The general procedure appears below.

NR1: Convert repeating group type aggregates (if any exist) into separate entities and create an association.

NR2: Convert associations into attributes or part of the key in an entity.

NR3: Map entities into relations (deleting any area or location mode information.)

NR4: Map key to key.

NR5: Map attributes to components of the relations.

After applying these mapping steps to the SDICF description section, the following illustrates the transformations which will have occurred at each of these steps.

After NR1: No change is effected because no repeating-group-type aggregate was in the data. However, the non-repeating aggregates are replaced in their respective entities with their individual attributes.

After NR2: Two new attributes are produced:

```
AT13;SALES-REP-1;FIXED4@
AT14;REP-NAME-1;FIXED4@
```

Attributes AT1 and AT7 are added to entity EN4 as the primary key.

Attributes AT1 and AT13 are added to entity EN2 as the primary key.

After NR3, NR4, and NR5: An internal schema table is produced which contains the components, relations and keys. The components are:

```
AT1   SUPPLIER-NUM;      FIXED 4 @
AT2   SUPPLIER-NAME;     CHARACTER 20 @
AT3   SUPPLIER-ADDRESS;  CHARACTER 40 @
AT4   LAST-NAME;         CHARACTER 20 @
AT5   FIRST-NAME;        CHARACTER 10 @
AT6   TELEPHONE-NUMBER;  CHARACTER 10 @
AT7   PART-NUMBER;       FIXED 9 @
AT8   PART-NAME;         CHARACTER 20 @
AT9   PART-COLOUR;       CHARACTER 10 @
AT10  PART-WEIGHT;       FIXED 4 @
AT11  QUANTITY-STOCK;    FIXED 10 @
AT12  QUANTITY-ORDER;    FIXED 10 @
AT13  SALES-REP-1;       FIXED 4 @
AT14  REP-NAME-1;        FIXED 4 @
```

The relations and keys are:

```
SUPPLIER;  AT1;AT2;AT3;          KEY=AT1;
SALES-REP; AT5;AT6;              KEY=AT1;AT13;
PARTS;     AT7;AT8;AT9;AT10;     KEY=AT7;
QUANTITY;  AT11;AT12;            KEY=AT1;AT7;
```

The OTS then uses this schema table to produce the schema definition and commands for the System R DBMS as shown in Figure C-7. The Output Data Processor uses the schema table to produce the rest of the loadable tape which contains the data conforming to the description in the produced schema. In other words, the loadable tape of System R is similar to the SDICF file in the sense that it also consists of two parts. The first part contains information on the positions of each component (albeit much less sophisticated than the SDICF), and the second part contains the data to be loaded.

```
CREATE TABLE SUPPLIER
     (SUPPLIER-NUM (DECIMAL(4),NONULL)),
     (SUPPLIER-NAME (CHAR(2Ø))),
     (SUPPLIER-ADDRESS (CHAR(4Ø)))

CREATE TABLE PARTS
     (PART-NUMBER (DECIMAL(9),NONULL)),
     (PART-NAME (CHAR(2Ø))),
     (PART-COLOUR (CHAR(1Ø))),
     (PART-WEIGHT (DECIMAL(4)))

CREATE TABLE SALES-REP
     (SUPPLIER-NUM (DECIMAL(4),NONULL)),
     (SALES-REP-1 (INTEGER,NONULL)),
     (TELEPHONE-NUMBER (CHAR(1Ø)))

CREATE TABLE QUANTITY
     (SUPPLIER-NUM (DECIMAL(4),NONULL)),
     (PART-NUMBER (DECIMAL(9),NONULL)),
     (QUANTITY-STOCK (DECIMAL(1Ø))),
     (QUANTITY-ORDER (DECIMAL(1Ø)))

CREATE TABLE REP-NAME
     (SUPPLIER-NUM (DECIMAL(4),NONULL)),
     (SALES-REP-1 (INTEGER,NONULL)),
     (REP-NAME-1 (INTEGER,NONULL)),
     (FIRST-NAME (CHAR(1Ø))),
     (LAST-NAME (CHAR(2Ø)))
```

System R

Schema Definition with Embedded Commands

FIGURE C-7

2.5 TARGET DATABASE CREATION

This final step uses the two outputs of the OTS: the schema definition commands and the System R loadable tape to create a new database. It consists of two steps:

1) System R schema preparation and generation.

   The definition of relations is provided by the OTS as shown in Figure C-6 but the DBA may modify or supplement this definition. For example, System R supports access aids such as indices and links. Also, various views can be defined as well as integrity controls. Most of these definitions can be done at any time, even during database operation, and only a minimal amount of specification is required before the actual database population.

2) Database population.

   Database population is performed by the use of System R utility. The load tape produced by the OTS is used as input by the utility which populates the database as defined in the description part.

A sample of a possibly populated System R database is shown in Figure C-8.

SUPPLIER

11	Maxime's	Paris
22	Holmes	London

PARTS

10	Truffle	Brown	1
20	Derby	Black	2

QUANTITY

11	10	100	50
22	10	250	75
22	20	112	68

SALES-REP

11	1111	6168841212
11	2222	2027876812
22	3333	5163136868

REP-NAME

11	1111	1000	Ziehm	Jeff
11	2222	2000	Jordache	Jovan
22	3333	3000	Holmes	Sherlock

Example of Populated System R Database Relation Instances

Figure C-8

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 81-2315	2. Performing Organ. Report No.	3. Publication Date August 1981

4. TITLE AND SUBTITLE

Draft Specification for Structured Data Interchange Form

5. AUTHOR(S) *James P. Fry, Wan-Ping Chiang, Donald A. DeSmith, David S. Leder, Duc Kim Nguyen, Robert W. Perreault*

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

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

Database Systems Research Group
Graduate School of Business Admin
The University of Michigan
Ann Arbor, Michigan 48109

7. Contract/Grant No. *NB79SBCA0201*

8. Type of Report & Period Covered *Interim*

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

National Bureau of Standards
Department of Commerce
Washington, D. C. 20234

10. SUPPLEMENTARY NOTES

☐ 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)

This report defines the format of a self-describing data interchange file that is media and machine independent, and that would facilitate the transfer of structured data between dissimilar computing systems and software. The specification is intended for interchanging data that has been structured under the discipline of a database management system, although it may also prove applicable to interchanging data that has been structured under other software disciplines, such as the COBOL programming language. The specification covers the three principal types of database management systems, namely the hierarchical, network, and relational types, but it is believed to be sufficiently general to accommodate data from other database management systems. The data interchange file consists not only of the data occurrences or values to be transferred, but also a description of their logical and physical characteristics and the relationships within the data. This minimizes or eliminates the need for additional, non-standardized information transfer in order to successfully perform a database transfer.

The report is written in the form of a candidate standard, so that interested parties can review specific requirements, rather than simply the philosophy or concepts of the approach. However, at this time, this specification is not proposed as a Federal Information Processing Standard. This report is published in order to obtain independent views on the technical merits, efficacy, and potential costs of this avenue for resolving database transfer problems.

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

Software conversion; database management; data conversion; data interchange form; data translation; portability; data description.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ 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 125

15. Price $11.00