An Experimental, Comprehensive Data Dictionary

Thomas R. Harbron
Professor of Computer Science
Anderson College
Anderson, Indiana

Christopher M. Funk
President, C. M. Funk & Co.
Lafayette, Indiana

ABSTRACT

This paper describes an experimental Comprehensive Data Dictionary (CDD). The purpose of the CDD is to describe all data objects precisely, from bits to databases, so that programs may manipulate these objects without continually redefining them.

The most complex part of the description concerns the ways in which data objects relate to each other. By precisely describing these relationships, the CDD allows relatively simple processors to perform the functions of database management systems (IMAGE), screen drivers (V/3000), report generators, query processors (QUERY) and other subsystems.

Application programs may be developed with relatively little effort since all descriptions, relationships, and conversions are described by the CDD and need not be included in the program.

The experimental CDD is described in detail and the experience of mapping applications into it is shared. Strengths and weaknesses are assessed and the direction of future developments indicated.

INTRODUCTION

Centrality of Data

A mature view of data processing is that programs are functions operating on data. This idea may be expressed in mathematical notation as:

$$Y := F(X)$$

where Y is the set of output data, X is the set of input data and F is the function of the program.

Very often the function is fairly simple and, when the program is examined, one finds that most of the program is concerned with describing either the data in sets X and Y, or elementary transformations between them. The actual, functional parts of the program constitute a relatively small portion of the total code. The problem is compounded by the need to repeat the data descriptions and elementary transformations in each and every program.

It is the purpose of a Comprehensive Data Dictionary to provide these descriptions in one central location. This has three immediate benefits for programs. First it eliminates the need to repeat the descriptions in each program, thereby considerably shortening the programs. Second, it provides a single, consistent description for all programs, thus eliminating conflicts. Third, it makes it possible to build general-purpose programs such as query processors, report generators, etc., thereby eliminating the need for most programming.

Traditional Weakness of Data Descriptions

The problem may not have begun with FORTRAN, but as the first popular, high-level language, FORTRAN did much to promote the idea that code was the main problem of data processing and data was only incidental to the code. Early FORTRAN compilers not only didn't require data declarations but, except for arrays, did not even permit them. Variables were "declared" simply by mentioning their names in the program. Data type was determined by the first letter of the variable name.

Later languages such as COBOL, and most recently PASCAL, have done much to restore data descriptions to their proper position where data within the program is concerned. Likewise systems developed in the last decade have included descriptions of data external to programs such as the schema of IMAGE and forms file of V/3000.

Each of these data descriptions, however, has only spanned a small and specific portion of the data used by an application. Not only does this result in a fragmented description, but numerous problems are created when the various descriptions do not totally coincide at the boundaries between them.

The Comprehensive Data Dictionary

The purpose of the Comprehensive Data Dictionary (CDD) is to provide a single source for descriptions of all data elements in an application. This includes simple data items, aggregations such as arrays, records, internal files including databases, and external files including reports and screens. Although not properly part of the data descriptions, it is easy to add access and security information to the CDD as well.

It is important to implement the CDD in such a way that it can be easily read by an automatic processor

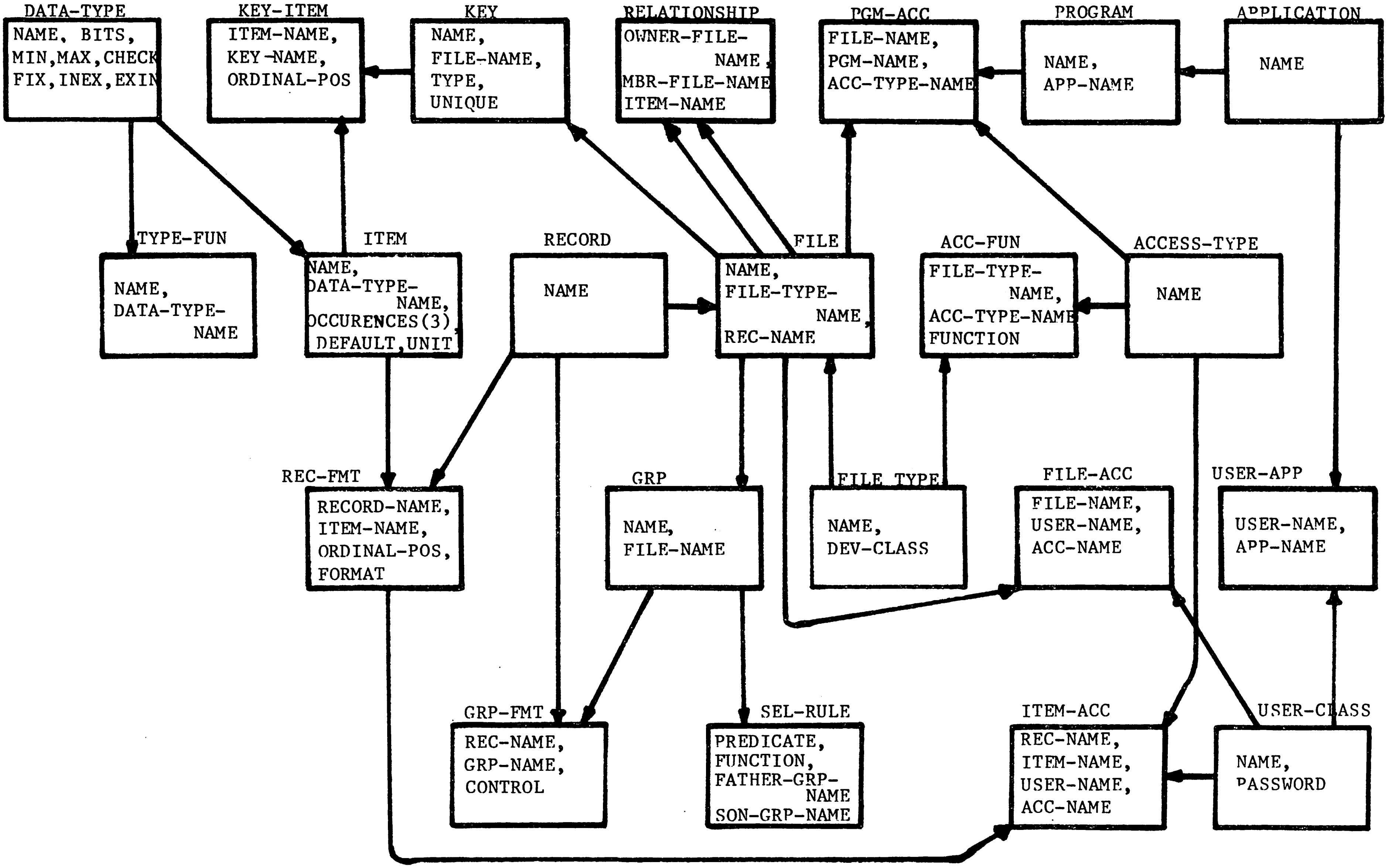

DATA-TYPE
NAME, BITS,
MIN,MAX,CHECK
FIX,INEX,EXIN
KEY-ITEM
ITEM-NAME,
KEY-NAME,
ORDINAL-POS
KEY
NAME,
FILE-NAME,
TYPE,
UNIQUE
RELATIONSHIP
OWNER-FILE-NAME,
MBR-FILE-NAME
ITEM-NAME
PGM-ACC
FILE-NAME,
PGM-NAME,
ACC-TYPE-NAME
PROGRAM
NAME,
APP-NAME
APPLICATION
NAME
TYPE-FUN
NAME,
DATA-TYPE-NAME
ITEM
NAME,
DATA-TYPE-NAME,
OCCURENCES(3),
DEFAULT,UNIT
RECORD
NAME
FILE
NAME,
FILE-TYPE-NAME,
REC-NAME
ACC-FUN
FILE-TYPE-NAME,
ACC-TYPE-NAME
FUNCTION
ACCESS-TYPE
NAME
REC-FMT
RECORD-NAME,
ITEM-NAME,
ORDINAL-POS,
FORMAT
GRP
NAME,
FILE-NAME
FILE TYPE
NAME,
DEV-CLASS
FILE-ACC
FILE-NAME,
USER-NAME,
ACC-NAME
USER-APP
USER-NAME,
APP-NAME
GRP-FMT
REC-NAME,
GRP-NAME,
CONTROL
SEL-RULE
PREDICATE,
FUNCTION,
FATHER-GRP-NAME
SON-GRP-NAME
ITEM-ACC
REC-NAME,
ITEM-NAME,
USER-NAME,
ACC-NAME
USER-CLASS
NAME,
PASSWORD

Figure 1

(report generator, query processor, program generator, etc.) as well as by people. Typically, the processor would read and store internally the descriptions relevant to the particular function being performed at the time.

The Experiment

There comes a point where theoretical work must be reconciled with the "real world." That is the purpose of this experiment. The CDD model has been derived on a solid theoretical basis. The model conforms to that of a normalized network database. It has been implemented using a relation database system.

The CDD, thus implemented, has been used to, first, describe itself, a non-trivial exercise. Next a variety of applications, drawn from a production environment, have been described in the CDD. Some weaknesses have been uncovered by this process, as well as some things that work very well.

THE COMPREHENSIVE DATA DICTIONARY

A data structure diagram is used to describe the CDD as shown in Figure 1. This model, with its 22 entities, 29 relationships, and 37 attributes, is too detailed to describe as a whole. Instead, it will be described in six parts in the following sections. The reader may, however, wish to refer to Figure 1 from time-to-time to see how the various parts are related.

Data-Item Part

This part of the CDD describes data items, their aggregations, and their components. This part of the CDD is shown in Figure 2.

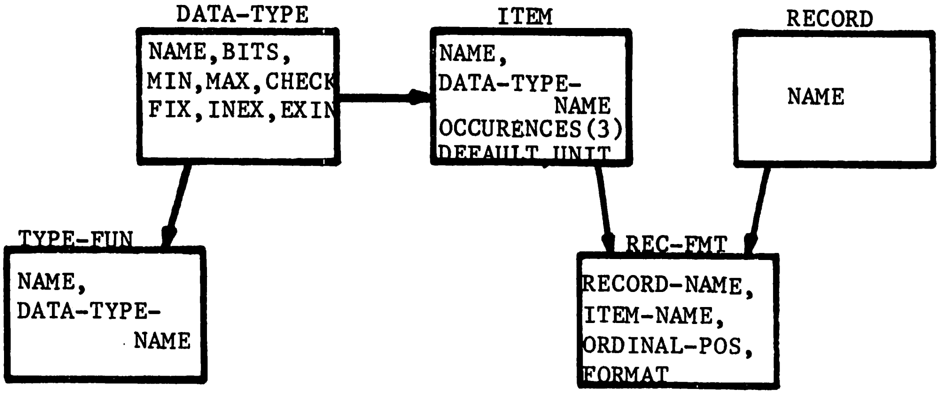


Figure 2

Before data items can be defined, it is necessary to define the basic data-types. Data-types may be defined in terms of their descriptions and the operations that may be performed on them. The descriptions and a basic set of functions are contained in the DATA-TYPE entity. Arithmetic, logical, and other functions are named, but not described in the TYPE-FUN entity.

An item may be a single occurrence of a date-type, or an array of up to three dimensions. A record is an aggregation of items and may be either an internal file, such as a disk file or database, or an external file such as a screen or report.

Record-format describes how items are related to records including position and format.

The following contains a description of each entity, its attributes, and relationships for this part.

ENTITY: DATA-TYPE

This entity describes a fundamental data-type such as byte, integer, real, etc. Only rarely should it be necessary to add a data-type once the basic set is in place. However, provision is made to describe new data-types in terms of their attributes. No semantic descriptions are provided.

Attribute: NAME

An ASCII character string of eight bytes containing the name of the data-type. This name will be referenced from other entities.

Attribute: BITS

The number of bits required by this data-type. Data-types will be assumed to start on word boundaries (high order end) except where assembled into arrays where they may be packed.

Attribute: MIN

This is the minimum value allowed for data of this type. Sixty-four bits are allowed for its representation. However, only the number of bits specified by the "BITS" attribute are used. If the numeric value of MIN cannot be represented in sixty-four bits or less, the value will be left justified and all truncated bits will be assumed to be zeroes.

Attribute: MAX

This is the maximum value allowed for data of this type. Storage is the same as for "MIN." If the numeric value of MAX cannot be represented in sixty-four bits or less the value will be left justified and all truncated bits will be assumed to be zeroes.

Attribute: CHECK

This is the name of a procedure which will check representations of this data-type to see if they contain legal values. It returns only a true/false indication.

Attribute: FIX

This is the name of a procedure which will check representations of this data-type to see if they contain legal values. In case of an illegal value, it will replace the illegal value with a default value appropriate to the illegal value. It may also return an indication of the error.

Attribute: INEX

This is the name of a procedure which will convert an internal representation of this data-type to an external (ASCII) form. In addition to the value of the data-item, it may also use a format description (see REC-FMT) to specify options in the conversion.

Attribute: EXIN

This is the name of a procedure which will convert an external representation of this data-type to an internal form. Again, a format description may be used to specify options in the conversion.

Relationship: TYPE-FUN

DATA-TYPE is related 1:N to TYPE-FUN. Each related TYPE-FUN is a legitimate function to use with this DATA-TYPE. The linking data-item is DATA-TYPE-NAME.

Relationship: ITEM

DATA-TYPE is related 1:N to ITEM. Each related ITEM is of this DATA-TYPE. The linking data-item is DATA-TYPE-NAME.

ENTITY: TYPE-FUN

This entity represents each function that is associated with a data-type.

Attribute: NAME

An ASCII character string of eight bytes that gives the name of the function.

Attribute: DATA-TYPE-NAME

The name of the data-type for which this is a function.

Relationship: DATA-TYPE

DATA-FUN is related N:1 to DATA-TYPE. The linking data-item is DATA-TYPE-NAME.

ENTITY: ITEM

This entity describes each unique data-item. The item may be a simple variable, or an array in 1, 2, or 3 dimensions.

Attribute: NAME

An ASCII character string of 12 bytes containing the name of the item.

Attribute: DATA-TYPE-NAME

The data-type of which this item is one occurrence.

Attribute: DEFAULT

A default value which is to be used for this item when no other value is available. Sixty-four bits are allowed for its representation, but only the bits required are used. In the case of array items, only the value for one element of the array is given.

Attribute: OCCURRENCES

This is a triple valued attribute which gives the three dimensions of the array if this item is an array. For a simple data-item, this attribute will have the value 1,1,1. For a one-dimensional array of order N, it will have the values N,1,1. For a two-dimensional array, values M,N,1; for three dimensions, values L,M,N.

Attribute: UNIT

This attribute is an ASCII string of eight characters used to indicate the unit of measurement, such as feet, yards, meters, etc. if no units of measurement are required, this field will be null.

Relationship: DATA-TYPE

ITEM is related N:1 to DATA-TYPE. Each item is of exactly one DATA-TYPE. DATA-TYPE-NAME is the linking data-item.

Relationship: REC-FMT

ITEM is related 1:N to REC-FMT. The linking data-item is ITEM-NAME.

Relationship: KEY-ITEM

ITEM is related 1:N to KEY-ITEM, with ITEM-NAME as the linking data-item. This relationship indicates which items are used as keys.

ENTITY: RECORD

This entity names a logical record which can be a part of one or more files. The record contains one or more data-items and may be of internal or external value.

Attribute: NAME

An ASCII character string sixteen bytes long containing the name of the record. This name will be referenced by other entities.

Relationship: REC-FMT

RECORD is related 1:N to REC-FMT, with RECORD-NAME as the linking data-item. This relationship defines the items contained in the record, their location, and their format.

Relationship: FILE

RECORD is related 1:N to FILE, and the linking data-item is RECORD-NAME. This relationship exists only for internal files and identifies the files in which each record occurs.

Relationship: GRP-FMT

RECORD is related 1:N to GRP-FMT, with the linking data-item being RECORD-NAME. This relationship exists only for external files and identifies the groups (and ultimately files) in which each record occurs.

ENTITY: REC-FMT

This entity (record format) represents the unique intersection of one item and one record. The entity contains information on how the item is related to the record.

Attribute: RECORD-NAME

The record name of which REC-FMT is a member.

Attribute: ITEM-NAME

The name of the item being described.

Attribute: ORDINAL-POS

An integer stating the ordinal position (1st, 2nd, 3rd, etc.) of the item in the record.

Attribute: FORMAT

This is a description of the format of the item for this particular record. This attribute will be used to determine dollar signs, commas, and other external features. The internal representation is indicated by a default format.

Relationship: ITEM-ACC

REC-FMT is related 1:N to ITEM-ACC, and the ITEM-NAME provides the link. This relationship exists as part of the security provisions and determines the access allowed each user-class to each item within each record.

Relationship: ITEM

REC-FMT is related N:1 to ITEM. The linking data-item is ITEM-NAME.

Relationship: RECORD

REC-FMT is related N:1 to RECORD, with RECORD-NAME providing the linkage.

☆ ☆ ☆

Internal File Part

This portion of the CDD describes internal files including disk files, databases, etc. This part of the CDD is shown in Figure 3.

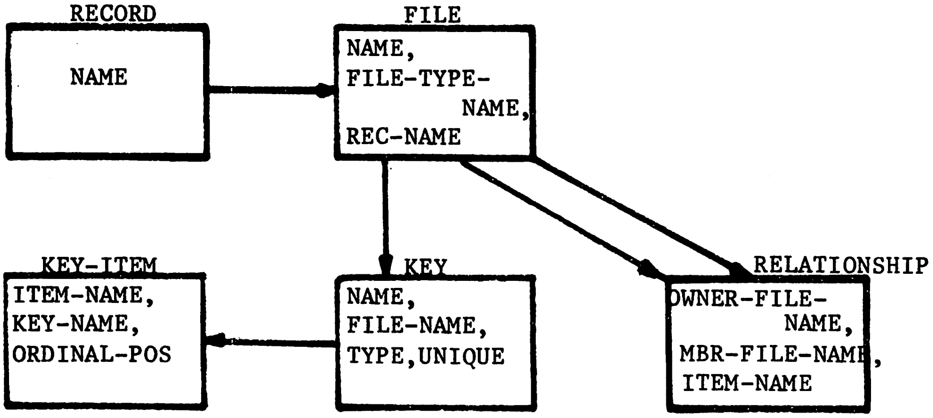


Figure 3

Each record type occurs in one or more files. Each file, usually has one or more keys by which records may be identified and retrieved. Each key, in turn, may consist of one or more data-items. The relationship between keys and data items is described by the entity KEY-ITEM.

The entity RELATIONSHIP is used to describe the relationship between records in one file and records in another file. For a given relationship, a file is either the owner or a member of the relationship. If a file is the owner of a relationship, the following conditions prevail:

1. Each owner record is related to zero or more records in the member file.

2. Each owner record shares with its member records a common value of the linking data-item.

3. An owner record may not be deleted if it is related to one or more member records.

The reader may recognize IMAGE "master" records as being owner types. In IMAGE the relationships are indicated by "chains" of pointers. Likewise, from the following constraints on member records, it may be seen that IMAGE "detail" records are member records.

1. Each member record is related to exactly one owner record in the relationship.

2. All member records share with their owner record a common value of the linking data-item.

3. A member record may not be added if no owner record exists with which it shares a common value of the linking data-item.

These rules not only define how a relationship is established between records in different files, but also prevent the infamous insertion and deletion anamolies from occurring in a normalized database. A file may simultaneously be a member of zero or more relationships and the owner of zero or more relationships. Note that in data structure diagrams, such as Figure 1, the arrow always points from the owner to the member in a relationship.

This description of internal files with keys and relationships, is equivalent to a database schema. Thus the

CDD subsumes the part of the database management system.

The entities not previously described are as follows:

ENTITY: FILE

The entity FILE describes a unique file of a given name. Files can be external in form, such as reports and screens, or internal in the form of disk and other storage medium files. External files may contain a variety of records and these records are collected into groups. The entities GRP and GRP-FMT are used to relate records to external files. Internal files normally contain one type of record. This relationship is shown by the 1:N relationship from record to file. An internal file may have one or more keys and relationships between internal files are given by the RELATIONSHIP entity.

Attribute: NAME
An ASCII character string with a maximum of twenty-six bytes containing the file name, group name, and account name necessary for accessing the file. This name will be referenced by other entities.

Attribute: FILE-TYPE-NAME
The name of the file-type to which a given file belongs.

Attribute: RECORD-NAME
The name of the record which occurs repeatedly to form the file. This attribute is valid only for internal files and will default when the file is of external form.

Relationship: FILE-TYPE
FILE is related N:1 to FILE-TYPE. The linking data-item is FILE-TYPE-NAME. This relationship indicates the file-type and, by implication, the functions for each file.

Relationship: FILE-ACC
This is a 1:N relationship between FILE and FILE-ACC with a linking data-item of FILE-NAME. The relationship indicates the access modes allowed to specific user-class for this file.

Relationship: PGM-ACC
FILE is related 1:N to PGM-ACC. The linking data-item is FILE-NAME. This relationship indicates the access mode used by a given program for each file.

Relationship: RECORD
FILE is related N:1 to RECORD with the linking data-item being RECORD-NAME. This relationship is valid only for files of an internal form and shows the normal pattern of one record type for an internal file.

Relationship: KEY
FILE is related 1:N to KEY and the linking data-item is FILE-NAME. Entity KEY and this relationship are valid only for internal files. Each key is a legitimate search item for the related file.

Relationship: GROUP
FILE is related 1:N to GROUP with the linking data-item being FILE-NAME. This relationship is valid only for external files and indicates the groups of records that are included in this file.

Relationship: OWNER-RELATIONSHIP
FILE is related to the entity RELATIONSHIP on the order of 1:N with OWNER-FILE-NAME being the linking data-item. This links each owner file to its corresponding relationships.

Relationship: MEMBER-RELATIONSHIP
FILE is related to the entity RELATIONSHIP on the order of 1:N with MEMBER-FILE-NAME being the linking data item. This links each member file to its corresponding relationship.

ENTITY: RELATIONSHIP

Attribute: OWNER-FILE-NAME
An ASCII string of 26 bytes that names the file which "owns" the relationship.

Attribute: MEMBER-FILE-NAME
An ASCII string of 12 bytes that names the file which is a "member" of the relationship.

Attribute: ITEM-NAME
An ASCII string of 12 bytes that names the data-item whose value is shared by the owner record and member records in this relationship.

Relationship: OWNER-FILE
RELATIONSHIP is related N:1 to FILE with OWNER-FILE-NAME being the linking data item. This links each member file to its corresponding relationships.

Relationship: MEMBER-FILE
RELATIONSHIP is related N:1 to FILE with MEMBER-FILE-NAME being the linking data-item. This links each member file to its corresponding relationships.

ENTITY: KEY

This entity identifies any and all keys for each internal file. The entity contains information on the name of the key, the file name to which it belongs, and the type of key.

Attribute: NAME
An ASCII string of 16 bytes containing the name of the key. This name will be referenced by KEY-ITEM.

Attribute: FILE-NAME
The name of the file to which a given key belongs.

Attribute: TYPE
This attribute is used to define the method of accessing a record by using the key. The type will differ according to whether the file is a sequential file, database file, etc.

Attribute: UNIQUE
This attribute has a value which is either true or false. If true, then each value of the key must be distinct from all other values of the key.

Relationship: KEY-ITEM

KEY is related 1:N to KEY-ITEM, with KEY-NAME providing the linkage. Any given key consists of one or more occurrences of KEY-ITEM. This allows a key to consist of composite data-items.

Relationship: FILE

KEY is related N:1 to FILE, with FILE-NAME providing the linkage.

ENTITY: KEY-ITEM

This entity represents the unique intersection of one key and one item. The entity contains information on how the item is related to the key.

Attribute: ITEM-NAME

The name of the item being described.

Attribute: KEY-NAME

The key name of which KEY-ITEM is a member.

Attribute: ORDINAL-POS

An integer stating the ordinal position (1st, 2nd, 3rd, etc.) of the item in the key.

Relationship: KEY

KEY-ITEM is related N:1 to KEY, with KEY-NAME being the linking data-item.

Relationship: ITEM

KEY-ITEM is related N:1 to ITEM, with ITEM-NAME being the linking data-item.

☆ ☆ ☆

External File Part

External files are those which are displayed externally from the computer system and generally are intended to be read and/or written by people as well as machines. Included in this category are formatted screens, reports, and graphical presentations.

Unlike internal files, which normally contain only one type of record, external files typically contain a variety of records. Organizing and sequencing this variety of records is the principal challenge in this part. The entities concerned in this organization are shown in Figure 4.

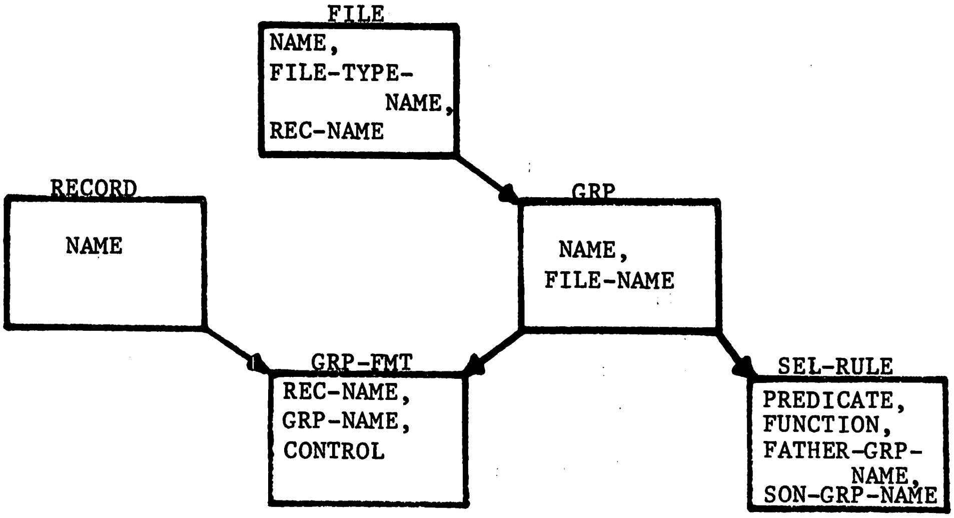


Figure 4

Each file consists of an aggregation of "groups" (GRP). A group is a group of records. The placement of each record within the group is controlled by the entity "group-format" (GRP-FMT). Since, typically, the rules for determining which group follows the previous one are data dependent, provision is made for a "selection rule" (SEL-RULE) to determine the sequence of groups within the file.

Descriptions of the entities from this part are as follows:

ENTITY: GROUP

This entity exists for external files only and names each specific group of records which are part of a given file. An external file consists of one or more groups, each group containing one or more records.

Attribute: NAME

An ASCII character string of 16 bytes that names each group.

Attribute: FILE-NAME

The name of the file to which the group belongs.

Relationship: GRP-FMT

GROUP is related 1:N to GRP-FMT, with GROUP-NAME providing the link. Any given group consists of one or more occurrences of GRP-FMT. This relationship defines the records contained in the group.

Relationship: SEL-RULE

GROUP is related 1:N to SEL-RULE, with the linking data-item being GROUP-NAME. SEL-RULE (selection rule) determines if the current group will be repeated or a new group will be selected.

Relationship: FILE

GROUP is related N:1 to FILE, with the linking data-item being FILE-NAME.

ENTITY: GRP-FMT

This entity (group format) represents the unique intersection of one record and one group. It contains information on how the record is related to the group.

Attribute: RECORD-NAME

The name of the record being described.

Attribute: GROUP-NAME

The group name of which GRP-FMT is a member.

Attribute: CONTROL

An ASCII string of eight bytes used to indicate the placement of the record within the group.

Relationship: RECORD

GRP-FMT is related N:1 to RECORD, with RECORD-NAME being the linking data-item.

Relationship: GROUP

GRP-FMT is related N:1 to GROUP, with GROUP-NAME being the linking data-item.

ENTITY: SEL-RULE

This entity (selection rule) is used to determine if the current group will be repeated, a new group will be selected, or the file terminated. The entity contains information on which group is to be selected and which function to use (append, replace, add, etc.).

Attribute: PREDICATE

An ASCII string of 28 characters which is tested to determine which rule will be selected. The following conditions prevail:

1. Each predicate is a proposition which is either true or false when tested.

2. The predicates are tested in the order given, and the first predicate found true prevails. Subsequent predicates are not tested.

3. Each predicate consists of a data-item name, an operator, and either a constant or another data-item name.

4. Data-items must be described in the CCD. All constants and variables must be of the same data-type. Operators are >, =, <, >=, <=, <>.

Attribute: FUNCTION

An ASCII string of eight characters containing the function to be used. The following functions are available:

REPEATA — Repeat, appended; this option repeats the current group and appends it to the previous group.

REPEATO — Repeat, overlayed; this option repeats the current group and overlays the previous group (this option is designed for use with screens).

NEXTA — Next group, appended; this function obtains the next group and appends it to the previous group.

NEXTC — Next group, cleared; this function obtains the next group and will clear the screen (or go to the top of the next page) before displaying the group.

TERMINATE — End of file; no new groups are obtained.

Attribute: FATHER-GROUP-NAME

The GROUP-NAME of the father of the current group. This attribute is used when the rule references the previous group.

Attribute: SON-GROUP-NAME

The GROUP-NAME of the son of the current group. This attribute is used when the rule references the next group.

Relationship: GROUP

SEL-RULE is related N:1 to GROUP, with the GROUP-NAME providing the linkage. The GROUP-NAME can be either the father of the current group or the son of the current group.

* * *

Access Part

Like data-items, a complete description of files must include the functions that operate upon them. These are the access functions which this section is concerned with. The relevant entities are shown in Figure 5.

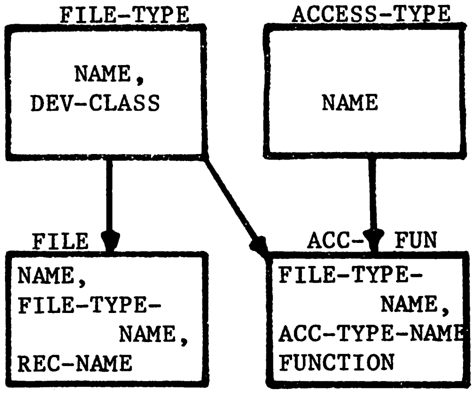


Figure 5

Each file must be of a type described by FILE-TYPE. These types may include sequential, direct access (hashing), indexed (KSAM, RELATE), IMAGE or other files. Each file contains an attribute which links it to a previously defined file type.

Likewise, there is a set of generic functions for files including read only, append only, update, read/write, etc. These are described in ACCESS-TYPE.

For each file-type and access-type, there is usually one function which provides that mode of access for that particular file type. Not all file-types support all modes of access.

The descriptions of these entities are as follows:

ENTITY: FILE-TYPE

This entity specifies the type of each file, and by relationship, the access function for each file type.

Attribute: NAME

An ASCII character string of eight bytes used to name the various file types.

Attribute: DEV-CLASS

An ASCII character string of eight bytes which contains the device class name on which the file type resides.

Relationship: FILE

FILE-TYPE is related 1:N to FILE, with FILE-TYPE-NAME being the linking data-item. This relationship links all files of a given type.

Relationship: ACC-FUN

FILE-TYPE is related 1:N to ACC-FUN, with FILE-TYPE-NAME being the linking data-item. This relationship indicates the functions for access of a given file-type.

ENTITY: ACCESS-TYPE

This entity represents the various access modes that are available for items, files, and programs. In the attribute ACCESS-TYPE, each bit of the integer represents an access function. If the bit corresponding to a given function is set to 1 then that function is allowed in the access type. An access type can consist of one or more functions. The functions — and their corresponding bit positions — available as part of the dictionary are:

Bit	Function	Explanation
7	Exclusive	Access to data is given to this user only
8	Read	User is allowed to read data
9	Append	User may append new data
10	Update	User may modify existing data
11	Delete	User may delete records
12	Create	User may create files
13	Purge	User may delete files
14	Execute	User is allowed to execute or stream files
15	Locking	Files or items may be locked to prevent concurrent access

Examples are shown below.

Access Type	Bit Pattern	Decimal Value
Read only shared access	0000000010000000	128
Read, update shared access with locking	0000000010100001	161
Read, append, update exclusive access	0000000111100000	480

Attribute: NAME

An integer containing the bit code representing the corresponding access type. NAME is referenced from other entities.

Relationship: ACC-FUN

This is a 1:N relationship between ACCESS-TYPE and ACC-FUN which indicates the functions which are used for data manipulation when a particular access mode is prevalent. The linking data-item for this relationship is ACCESS-TYPE-NAME.

Relationship: PGM-ACC

ACCESS-TYPE is related 1:N to PGM-ACC with the linking data-item being ACCESS-TYPE-NAME. This relationship indicates the mode of access used by a given program to a given file.

Relationship: FILE-ACC

The entity ACCESS-TYPE is related 1:N with FILE-ACC and has a linking data-item of ACCESS-TYPE-NAME. This relationship indicates the files which are accessible by a given user.

Relationship: ITEM-ACC

The entity ACCESS-TYPE has a 1:N relationship to ITEM-ACC which represents the items which are accessible by a particular user. The linking data-item is ACCESS-TYPE-NAME.

ENTITY: ACC-FUN

This entity represents the function that is used with a given access mode to reference a certain file type. Functions are external to the Data Dictionary and will be referenced when a file access is requested.

Attribute: FILE-TYPE-NAME

The name of a file type for which a function is used.

Attribute: ACC-TYPE-NAME

The name of an access type for which a function is used.

Attribute: FUNCTION

The ASCII character string of eight bytes which names the function.

Relationship: ACCESS-TYPE

ACC-FUN is related N:1 to ACCESS-TYPE with a linking data-item of ACCESS-TYPE-NAME. This relationship indicates the functions which are used by a given access type.

Relationship: FILE-TYPE

This is an N:1 relationship between ACC-FUN and FILE-TYPE which indicates the functions that are used by a given file type. The linking data-item is FILE-TYPE-NAME.

☆ ☆ ☆

Application Part

Although not properly a part of the data descriptions, it is helpful to have information on programs and applications in the CDD. Particularly useful is knowledge of the relationships between programs and files; which programs use which files and in which mode of access. This information is stored in the application part of the CDD as shown in Figure 6.

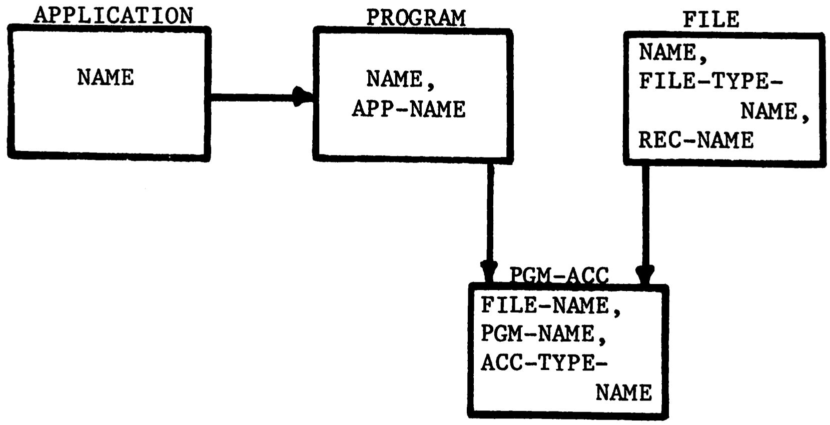


Figure 6

Each application area is given a name which is recorded in the entity APPLICATION. Each application owns a set of programs which are named in the PROGRAM entity. For each file accessed by each program, there is an entry in PGM-ACC which shows the mode of access for that particular program-file pair.

Since files commonly bridge application boundaries, there is no attempt to assign files to applications. The linkage exists implicitly through the programs.

The application part entities are described as follows:

ENTITY: APPLICATION

The entity APPLICATION represents the various applications whose data is described by the Data Dictionary. The users allowed to access an application are shown by the relationship to USER-APP.

Attribute: NAME

An ASCII character string of eight bytes containing the name of an application. This name will be referenced from other entities.

Relationship: PROGRAM

APPLICATION is related 1:N to PROGRAM with a linking data-item of APPLICATION-NAME. This relationship indicates the programs included in an application area.

Relationship: USER-APP

This is a 1:N relationship between APPLICATION and USER-APP which indicates the user's given access to an application. The linking data-item is APP-NAME.

ENTITY: PROGRAM

This entity gives the name of each program which is currently part of the Comprehensive Data Dictionary. The entity will also indicate the relationship any program has to an application area. The relationship between PROGRAM and PGM-ACC shows the access the program has to files.

Attribute: NAME

The ASCII character string of a maximum 26 bytes which contains the program name, group name, and account name necessary for accessing the file. This name will be referenced by other entities.

Attribute: APP-NAME

The name of the application to which this program belongs.

Relationship: APPLICATION

PROGRAM is related N:1 to APPLICATION. The linking data-item is APPLICATION-NAME. This relationship indicates the application area to which a program belongs.

Relationship: PGM-ACC

This is a 1:N relationship between PROGRAM and PGM-ACC which indicates the various access allowed between files and programs. The linking data-item is PROGRAM-NAME.

ENTITY: PGM-ACC

This entity is the unique intersection between ACCESS-TYPE, FILE, and PROGRAM. The entity represents the allowed file accesses for a given program. This entity is used to determine the mode of access allowed by each program to each file.

Attribute: FILE-NAME

The name of the file being accessed.

Attribute: PROGRAM-NAME

The program name of the program accessing the file.

Attribute: ACCESS-TYPE-NAME

The access type name which indicates the access mode for the access being defined.

Relationship: PROGRAM

PGM-ACC is related N:1 to PROGRAM with a linking data-item of PROGRAM-NAME. This relationship indicates which program is given access to the given file.

Relationship: FILE

This is a N:1 relationship between PGM-ACC and FILE which indicates the file which can be accessed by the program. The linking data-item is FILE-NAME.

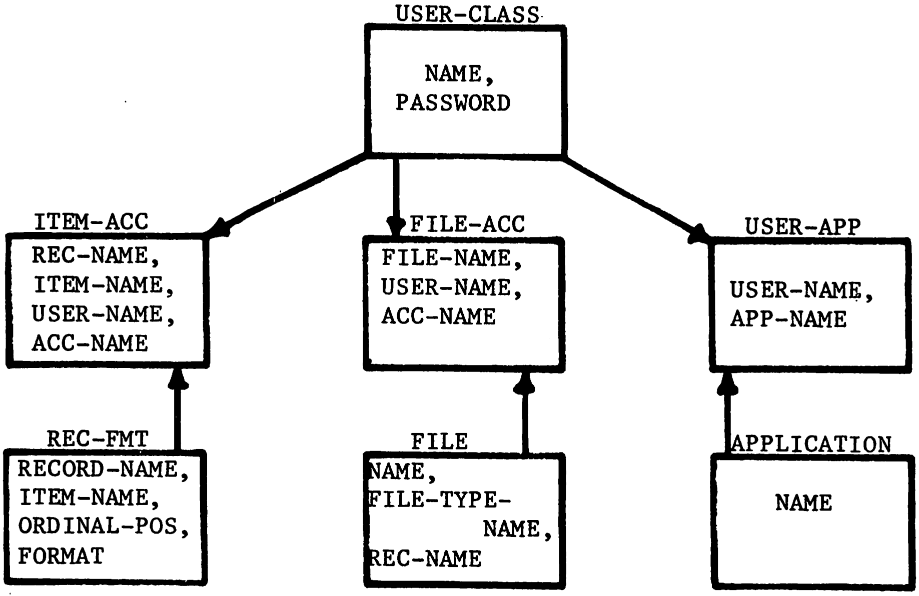
Relationship: ACCESS-TYPE

The entity PGM-ACC is related N:1 to ACCESS-TYPE with a linking data-item of ACCESS-TYPE-NAME. This relationship indicates the type of access the program may use when referencing the file for a given PGM-ACC.

⭑ ⭑ ⭑

Security Part

As with the application part, security is not properly a part of the data description. However, it is a necessary part of any application using the CDD and may conveniently be accommodated here. This information is contained in the part of the CDD shown in Figure 7.



USER-CLASS
NAME, PASSWORD
ITEM-ACC
REC-NAME, ITEM-NAME, USER-NAME, ACC-NAME
FILE-ACC
FILE-NAME, USER-NAME, ACC-NAME
USER-APP
USER-NAME, APP-NAME
REC-FMT
RECORD-NAME, ITEM-NAME, ORDINAL-POS, FORMAT
FILE
NAME, FILE-TYPE-NAME, REC-NAME
APPLICATION
NAME

Figure 7

Users of the CDD, or applications described therein, are identified by their USER-CLASS-NAME. Each such name has a password associated with it to verify authenticity. The name and password are recorded in the USER-CLASS entity.

The applications, and hence programs, to which a given user-class has access are determined by entries in the USER-APP entity. An entry must occur here for each user-class/application pair that is allowed.

Data access is controlled at two levels. A user must be allowed access at both levels to be successful. Where a conflict exists between the levels, the most restrictive case prevails.

Access to files is controlled by the FILE-ACC entity. For each allowed user-class/file pair, an entry names the user-class, file, and acces mode allowed.

Access to individual data-items is controlled within the context of records. For example, a certain user-class may be allowed to read the data-item EMPLOYEE-NAME within the context of a production record but not allowed to see the same item in the context of a payroll record.

Data-item access is controlled by the ITEM-ACC entity. The item, record, user, and access mode are identified to allow the user access to the specified item within the specified record.

The security part entities are as follows:

ENTITY: USER-CLASS

This entity represents the different classes of users that will be able to reference the data described by the Data Dictionary. Each user class is allowed access to a limited set of applications, files, and data-items. The type of access is controlled in each case.

Attribute: NAME

An ASCII string of eight bytes containing the name of a user classification. This name will be referenced from other entities.

Attribute: PASSWORD

An ASCII string of eight bytes containing the password which controls the availability of specific user classification accesses.

Relationship: USER-APP

The entity USER-CLASS is related 1:N to USER-APP with a linking data-item of USER-CLASS-NAME. This relationship indicates which applications are accessible by a user classification, and the modes of access allowed.

Relationship: FILE-ACC

USER-CLASS is related 1:N to FILE-ACC. The linking data-item is USER-CLASS-NAME. The files accessible by a user classification and the mode of access are indicated through this relationship.

Relationship: ITEM-ACC

USER-CLASS is related 1:N to ITEM-ACC and the linking data-item is USER-CLASS-NAME. The items accessible by a user classification and the mode of access are indicated through this relationship.

ENTITY: USER-APP

This entity represents the unique intersection between a user classification and an application. The intersection shows a user classification that is allowed access to a given application.

Attribute: USER-CLASS-NAME

The name of a user classification for which an application access is being defined.

Attribute: APP-NAME

The name of an application for which an access is being defined.

Relationship: USER-CLASS

This is a N:1 relationship between USER-APP and USER-CLASS. The linking data-item is USER-CLASS-NAME. The relationship indicates the user classification which is given access to an application.

Relationship: APPLICATION

USER-APP is related N:1 to APPLICATION with the linking data-item of APPLICATION-NAME. The relationship indicates the application to which a user classification is given access.

ENTITY: ITEM-ACC

This entity represents the unique intersection of three entities REC-FMT, USER-CLASS, and ACCESS-TYPE. The intersection defines an access by indicating which user classification can reference an item in a particular record and what mode of access is permitted.

Attribute: ITEM-NAME

The name of the item for which an access is being defined.

Attribute: RECORD-NAME

The name of the record for which an access is being defined.

Attribute: USER-NAME

The name of the user classification for the access being defined.

Attribute: ACCESS-TYPE-NAME

The name of the access type or mode for the item access being defined.

Relationship: USER-CLASS

ITEM-ACC is related N:1 to USER-CLASS with the linking data-item being USER-CLASS-NAME. This relationship indicates the user classification that is given access to an item.

Relationship: ACCESS-TYPE

This N:1 relationship between ITEM-ACCESS and ACCESS-TYPE indicates the access mode allowed in referencing the item. The linking data-item is ACCESS-TYPE-NAME.

Relationship: REC-FMT

ITEM-ACCESS is related N:1 to REC-FMT and the relationship indicates which item of a particular record will be referenced through the access defined. The linking data-items for this relationship are RECORD-NAME and ITEM-NAME.

ENTITY: FILE-ACC

This entity is the unique intersection of USER-CLASS, ACCESS-TYPE and FILE. This entity represents the modes of access allowed in referencing a given file by a user class. The access is defined by the different relationships that are present in this entity.

Attribute: FILE-NAME

The name of the file for which the access is being defined.

Attribute: USER-CLASS-NAME

The name of the user classification for which the access is being defined.

Attribute: ACCESS-TYPE-NAME

The access type name which defines the file access.

Relationship: FILE

This is a N:1 relationship between FILE-ACC and FILE. The linking data-item is FILE-NAME. This relationship indicates the file which will be referenced through the access defined.

Relationship: USER-CLASS

FILE-ACC is related N:1 to USER-CLASS and the relationship indicates the user classification that is given access to a file. The linking data-item is USER-CLASS-NAME.

Relationship: ACCESS-TYPE

FILE-ACCESS is related N:1 to ACCESS-TYPE with a linking data-item of ACCESS-TYPE. The access mode allowed by the defined access is indicated by this relationship.

☆ ☆ ☆

IMPLEMENTATION

Implementation of the CDD, done to date, is in three phases. First, a database is built to hold the data of the CDD. Second, the CDD is used to describe itself — a non-trivial exercise. Third, the CDD is used to describe some real-world applications.

The strengths and weaknesses of this CDD are assessed, based on the limited experience gained to date. Finally, future developments are briefly discussed.

Mapping the Model to a Database

The model of the CDD, described in the previous section, is in the form of a normalized, network, database. Thus, it is only natural to seek a database management system (DBMS) with which to implement it. Although IMAGE is based on the network model, it was rejected because of its rigidity and the limits of its two-level structure. Instead, RELATE/3000,* a relational DBMS was selected.

In mapping the model into RELATE, each entity becomes a Relation, or file. These files may each be indexed on any combination of keys. Attributes become data-items. Relationships cannot be explicitly shown in a relational DBMS, but are implicitly linked by shared data-item values.

Mapping the CDD into Itself

As a first exercise in mapping applications into the CDD, it was decided to map it into itself; i.e., use the CDD to describe itself. Since the CDD contains 22 entities, 37 data-items, and 29 relationships, the exercise is not trivial.

The initial mapping of the CDD into itself, using RELATE, is shown in Appendix A. Notice that some files are empty because the corresponding entities are not needed in this application. For example, at this time, there are no external files associated with the CDD, so the corresponding entities are empty.

Several small problems were encountered in this exercise. Several of the data-item and entity names had to be modified to conform to the naming conventions of RELATE. Since a full set of functions was not immediately defined for the standard data types, the TYPE-FUN entity was left empty. Likewise no programs were intially associated with the CDD.

Some problems of greater significance also appeared. One, that will doubtless reoccur in other applications, is that of composite data-items used as links in entity relationships. For example, both record-name and item-name are used as the linking item between REC-FMT and ITEM-ACC. Neither alone is sufficient. Yet provision is made for only one linking item in the RELATIONSHIP entity.

Another is the magnitude of records that can occur in some entities. For example, ITEM-ACC is limited only by the product of the number of records in ITEM, RECORD, ACCESS-TYPE, and USER-CLASS. At one time the number of records in these entities were 37, 22, 13, and 2 respectively giving a potential of 21,164 records in ITEM-ACC. While the actual number was only 284 it is still too large. Some kind of "wild-card" notation is being considered to reduce the number of records.

Another troublesome area is the representation of the values of MIN and MAX in DATA-TYPE, and DEFAULT in ITEM. The intention is that the binary or internal representations of these values be stored. However, this would require that these items be of different data-types in different records — a complexity beyond the ability of most DBMSs to handle. Two alternatives are apparent; either store them in external form, in which case all are stored as ASCII character strings; or declare them type long and left justify the actual value within the 64 bits.

Mapping Applications to the CDD

The press deadline for submission of this paper occurred too soon to allow much experimentation with real applications. The authors will be able to share these experiences when the paper is presented.

However, on the basis of early work done, some things have become obvious, and several changes or redefinitions are clearly indicated.

First, there is a substantial weakness in the area of composite data types. An additional entity needs to be created to link a composite data-type with its components. This will have several advantages over the present mechanism.

1. Arrays of any number of dimensions can be declared.
2. Composite types may have components of several different types.
3. Composite types may become components of more complex types.

Second, the whole access area is proving troublesome. Several issues need to be better defined including:

1. Better definitions of access modes and allowed combinations of modes.
2. A notation for item access that does not require a separate entry for each user-record-item intersection.

Third, some minor changes are needed in GROUP and GRP-FMT to accommodate the structure clash between external files and physical pages and screens. An attribute (LINE-NBR) can be added to GROUP to indicate the last line on which that group is allowed to begin. A current line number greater than this will trigger a new page.

Likewise a standard group must be added to each external file which will be inserted whenever a new page is triggered. This same group will also, automatically, begin each external file, thus eliminating initializing problems.

On the whole, real applications appear to be mapping in with very few other problems. In particular, the group structure for external file descriptions seems to work well. A final judgment must, however, await trials with "strange" external files as well as more standard ones.

*RELATE is a trademark of Computer Resources Incorporated, 2570 El Camino Real, Mountain View, CA 94040.

Future Developments

The next step is to complete the current development phase, i.e., testing the model against a variety of applications and refining it as indicated.

The next phase is to develop a "front-end" program to interface between the CDD and its manager. This program would perform the functions of adding, deleting, and modifying the contents of the CDD while checking for consistency. It would also provide formatted reports on the contents of the CDD.

To this point, the CDD will not have been used by processors to do production data processing. While it may prove very useful for documentation purposes, the principal value of the CDD is in its use in production. The development of the processors required to apply the CDD to production can proceed in three phases. While there is some overlap and interaction, they may proceed somewhat independently.

The first processor is a query/report/screen processor. It will move data between internal and external files. Thus, to produce a new report, it is only necessary to describe the report in the CDD. The processor can then produce the report from internal files. Likewise data could be transmitted between screens and internal files.

The second processor integrates the DBMS with the CDD. As mentioned earlier, presently available DBMSs each have a separate "schema" which describes only the data in the database. This processor combines the DBMS with the CDD so that internal files are described in only one place.

The third processor is a program generator which relies on the CDD for all data descriptions. This may either be a compiler or interpreter. In either case very high-level statements would allow most programs to be expressed in a fraction of the number of statements required by typical languages. By removing the data descriptions and conversions from the program, only the functional parts need be expressed.

CONCLUSION

The CDD has, initially, shown the capacity to contain the total data descriptions needed for applications. Thus, it is a suitable base on which to build sophisticated processors which will greatly reduce the need for applications' programming.

Research will continue in this direction. Meanwhile, it is hoped that others will benefit by this study and, in turn, contribute their experiences with data dictionaries to the common body of knowledge.

ACKNOWLEDGEMENT

The authors wish to thank Steve Beasley and Ken Knepp for the considerable time and effort they have contributed to this project. The typing was done by Maxine Loeber whose accuracy is greatly appreciated. Finally, support for this project by C. M. Funk & Co. and Anderson College is gratefully acknowledged.

APPENDIX A

INITIAL MAPPING OF THE COMPREHENSIVE DATA DICTIONARY INTO ITSELF USING RELATE/3000

FILE: DATATYP

ITEMS:	DATATYP	BITS	MIN	MAX	CHECK	FIX	INEX	EXIN
	INTEGER	16	-32768	32767			ASCII	BINARY
	REAL	32	-1.15792*10^{76}	1.15792*10^{76}			'INEXT	'EXTIN
	LONG	64	-1.15792*10^{76}	1.15792*10^{76}			'INEXT	'EXTIN
	BYTE	8	0	255				
	LOGICAL	16	0	65535			ASCII	BINARY
	DOUBLE	32	-2147483648	2147483648			DASCII	DBINARY

FILE: ITEM

ITEMS:

ITEM	DATATYP	OCCUR	OCCUR	OCCUR	DEFAULT	UNIT
DATATYP	BYTE	8	1	1		
BITS	INTEGER	1	1	1		
MIN	LONG	1	1	1	0	
MAX	LONG	1	1	1	0	
CHECK	BYTE	8	1	1		
FIX	BYTE	8	1	1		
INEX	BYTE	8	1	1		
EXIN	BYTE	8	1	1		
ITEM	BYTE	12	1	1		
KEY	BYTE	16	1	1		
POSITION	INTEGER	1	1	1	1	
FILE	BYTE	26	1	1		
TYPE	BYTE	8	1	1		
UNIQUE	BYTE	1	1	1	T	
OWN_FILE	BYTE	26	1	1		
MBR_FILE	BYTE	26	1	1		
PROGRAM	BYTE	26	1	1		
ACCTYPE	INTEGER	1	1	1		
APPLICA	BYTE	8	1	1		
TYPEFUN	BYTE	8	1	1		
OCCUR1	INTEGER	1	1	1	1	
OCCUR2	INTEGER	1	1	1	1	
OCCUR3	INTEGER	1	1	1	1	
DEFAULT	BYTE	8	1	1		
UNIT	BYTE	8	1	1		
RECORD	BYTE	16	1	1		
FILETYP	BYTE	8	1	1		
FUNCTION	BYTE	8	1	1		
FORMAT	BYTE	20	1	1		
GROUP	BYTE	16	1	1		
DEV_CLASS	BYTE	8	1	1		
USER	BYTE	8	1	1		
CONTROL	BYTE	8	1	1		
PREDICATE	BYTE	28	1	1		
FTHRGRP	BYTE	16	1	1		
SONGRP	BYTE	16	1	1		
PASSWORD	BYTE	8	1	1		

FILE: RECORD

ITEMS: RECORD

RC-DATATYP
RC-TYPEFUN
RC-ITEM
RC-RECORD
RC-RECFMT
RC-FILE
RC-GROUP
RC-GRPFMT
RC-SELRULE
RC-RLTNSHP
RC-KEY
RC-KEYITEM
RC-PGMACC
RC-ACCFUN
RC-FILETYP
RC-ITEMACC
RC-PROGRAM
RC-APPLICA
RC-ACCTYPE
RC-FILLACC
RC-USERAPP
RC-USERCLS

FILE: RECFMT

ITEMS:

RECORD	ITEM	POS	FORMAT
RC-DATATYP	DATATYP	1	
RC-DATATYP	BITS	2	
RC-DATATYP	MIN	3	
RC-DATATYP	MAX	4	
RC-DATATYP	CHECK	5	
RC-DATATYP	FIX	6	
RC-DATATYP	INEX	7	
RC-DATATYP	EXIN	8	
RC-TYPEFUN	TYPEFUN	1	
RC-TYPEFUN	DATATYP	2	
RC-ITEM	ITEM	1	
RC-ITEM	DATATYP	2	
RC-ITEM	OCCUR1	3	
RC-ITEM	OCCUR2	4	
RC-ITEM	OCCUR3	5	
RC-ITEM	DEFAULT	6	
RC-ITEM	UNIT	7	
RC-RECORD	RECORD	1	
RC-RECFMT	RECORD	1	
RC-RECFMT	ITEM	2	
RC-RECFMT	POSITION	3	
RC-RECFMT	FORMAT	4	
RC-FILE	FILE	1	
RC-FILE	FILETYP	2	
RC-FILE	RECORD	3	
RC-GROUP	GROUP	1	
RC-GROUP	FILE	2	
RC-GRPFMT	RECORD	1	
RC-GRPFMT	GROUP	2	
RC-GRPFMT	CONTROL	3	
RC-GRPFMT	POSITION	4	
RC-SELRULE	PREDICATE	1	
RC-SELRULE	FUNCTION	2	
RC-SELRULE	FTHRGRP	3	
RC-SELRULE	SONGRP	4	
RC-RLTNSHP	OWN_FILE	1	
RC-RLTNSHP	MBR_FILE	2	
RC-RLTNSHP	ITEM	3	
RC-KEY	KEY	1	
RC-KEY	FILE	2	
RC-KEY	TYPE	3	
RC-KEY	UNIQUE	4	
RC-KEYITEM	ITEM	1	
RC-KEYITEM	KEY	2	
RC-KEYITEM	POSITION	3	
RC-PGMACC	FILE	1	
RC-PGMACC	PROGRAM	2	
RC-PGMACC	ACCTYPE	3	
RC-ACCFUN	FILETYP	1	
RC-ACCFUN	ACCTYPE	2	

```
FILE:  RECFMT

ITEMS:  RECORD              ITEM           POS FORMAT

        RC-ACCFUN           FUNCTION         3
        RC-FILETYP          FILETYP          1
        RC-FILETYP          DEV_CLASS        2
        RC-ITEMACC          RECORD           1
        RC-ITEMACC          ITEM             2
        RC-ITEMACC          USER             3
        RC-ITEMACC          ACCTYPE          4
        RC-PROGRAM          PROGRAM          1
        RC-PROGRAM          APPLICA          2
        RC-APPLICA          APPLICA          1
        RC-ACCTYPE          ACCTYPE          1
        RC-FILEACC          FILE             1
        RC-FILEACC          USER             2
        RC-FILEACC          ACCTYPE          3
        RC-USERAPP          USER             1
        RC-USERAPP          APPLICA          2
        RC-USERCLS          USER             1
        RC-USERLCS          PASSWORD         2
```

```
FILE:  FILE

ITEMS:  FILE                          FILETYP   RECORD

        DATATYP                       RELATE    RC-DATATYP
        TYPEFUN                       RELATE    RC-TYPEFUN
        ITEM                          RELATE    RC-ITEM
        RECFMT                        RELATE    RC-RECFMT
        RECORD                        RELATE    RC-RECORD
        FILE                          RELATE    RC-FILE
        GROUP                         RELATE    RC-GROUP
        GRPFMT                        RELATE    RC-GRPFMT
        SELRULE                       RELATE    RC-SELRULE
        RLTNSHP                       RELATE    RC-RLTNSHP
        KEY                           RELATE    RC-KEY
        KEYITEM                       RELATE    RC-KEYITEM
        PGMACC                        RELATE    RC-PGMACC
        ACCFUN                        RELATE    RC-ACCFUN
        FILETYP                       RELATE    RC-FILETYP
        ITEMACC                       RELATE    RC-ITEMACC
        PROGRAM                       RELATE    RC-PROGRAM
        APPLICA                       RELATE    RC-APPLICA
        ACCTYPE                       RELATE    RC-ACCTYPE
        FILEACC                       RELATE    RC-FILEACC
        USERAPP                       RELATE    RC-USERAPP
        USERCLS                       RELATE    RC-USERCLS
```

FILE: RLTNSHP

ITEMS: OWN_FILE	MBR_FILE	ITEM
DATATYP	TYPEFUN	DATATYP
DATATYP	ITEM	DATATYP
ITEM	RECFMT	ITEM
ITEM	KEYITEM	ITEM
RECFMT	ITEMACC	ITEM
RECORD	RECFMT	RECORD
RECORD	GRPFMT	RECORD
RECORD	FILE	RECORD
FILE	GROUP	FILE
FILE	PGMACC	FILE
FILE	RLTNSHP	OWN_FILE
FILE	RLTNSHP	MBR_FILE
FILE	KEY	FILE
KEY	KEYITEM	KEY
GROUP	GRPFMT	GROUP
GROUP	SELRULE	FTHRGRP
FILE	FILEACC	FILE
APPLICA	PROGRAM	APPLICA
PROGRAM	PGMACC	PROGRAM
APPLICA	USERAPP	APPLICA
USERCLS	USERAPP	USER
USERCLS	FILEACC	USER
USERCLS	ITEMACC	USER
ACCTYPE	FILEACC	ACCTYPE
ACCTYPE	ITEMACC	ACCTYPE
ACCTYPE	ACCFUN	ACCTYPE
ACCTYPE	PGMACC	ACCTYPE
FILETYP	ACCFUN	FILETYP
FILETYP	FILE	FILETYP

FILE: FILETYP

ITEMS: FILETYP	DEV_CLAS
SEQUEN	DISC
DIR-ACC	DISC
KSAM	DISC
IMAGE-MA	DISC
IMAGE-DE	DISC
RELATE	DISC
PROGRAM	DISC
JCL	DISC

FILE: ACCTYPE

ITEMS: ACCTY
128
64
160
224
240
129
65
161
225
241
384
320
416
480
496
504
508
2
258

FILE: ACCFUN

ITEMS:	FILETYP	ACCTY FUNCTION
	SEQUEN	128
	SEQUEN	64
	SEQUEN	160
	SEQUEN	224
	SEQUEN	240
	SEQUEN	129
	SEQUEN	65
	SEQUEN	161
	SEQUEN	225
	SEQUEN	241
	SEQUEN	384
	SEQUEN	320
	SEQUEN	416
	SEQUEN	480
	SEQUEN	496
	SEQUEN	504
	SEQUEN	508
	DIR-ACC	128
	DIR-ACC	64
	DIR-ACC	160
	DIR-ACC	224
	DIR-ACC	240
	DIR-ACC	129
	DIR-ACC	65
	DIR-ACC	161
	DIR-ACC	225
	DIR-ACC	241
	DIR-ACC	384
	DIR-ACC	320
	DIR-ACC	416
	DIR-ACC	480
	DIR-ACC	496
	DIR-ACC	504
	DIR-ACC	508
	KSAM	128
	KSAM	64
	KSAM	160
	KSAM	224
	KSAM	240
	KSAM	129
	KSAM	65
	KSAM	161
	KSAM	225
	KSAM	241
	KSAM	384
	KSAM	320
	KSAM	416
	KSAM	480
	KSAM	496
	KSAM	504

FILE: ACCFUN

ITEMS:	FILETYP	ACCTY FUNCTION
	KSAM	508
	IMAGE-MA	128
	IMAGE-MA	64
	IMAGE-MA	160
	IMAGE-MA	224
	IMAGE-MA	240
	IMAGE-MA	129
	IMAGE-MA	65
	IMAGE-MA	161
	IMAGE-MA	225
	IMAGE-MA	241
	IMAGE-MA	384
	IMAGE-MA	320
	IMAGE-MA	416
	IMAGE-MA	480
	IMAGE-MA	496
	IMAGE-MA	504
	IMAGE-MA	508
	IMAGE-DE	128
	IMAGE-DE	64
	IMAGE-DE	160
	IMAGE-DE	224
	IMAGE-DE	240
	IMAGE-DE	129
	IMAGE-DE	65
	IMAGE-DE	161
	IMAGE-DE	225
	IMAGE-DE	241
	IMAGE-DE	384
	IMAGE-DE	320
	IMAGE-DE	416
	IMAGE-DE	480
	IMAGE-DE	496
	IMAGE-DE	504
		508
	RELATE	128
	RELATE	64
	RELATE	160
	RELATE	224
	RELATE	240
	RELATE	129
	RELATE	65
	RELATE	161
	RELATE	225
	RELATE	241
	RELATE	384
	RELATE	320
	RELATE	416
	RELATE	480
	RELATE	496

FILE: ACCFUN

ITEMS:

FILETYP	ACCTY FUNCTION
RELATE	504
RELATE	508
PROGRAM	128
PROGRAM	64
PROGRAM	160
PROGRAM	224
PROGRAM	240
PROGRAM	129
PROGRAM	65
PROGRAM	161
PROGRAM	225
PROGRAM	241
PROGRAM	384
PROGRAM	320
PROGRAM	416
PROGRAM	480
PROGRAM	496
PROGRAM	504
PROGRAM	508
JCL	128
JCL	64
JCL	160
JCL	224
JCL	240
JCL	129
JCL	65
JCL	161
JCL	225
JCL	241
JCL	384
JCL	320
JCL	416
JCL	480
JCL	496
JCL	504
JCL	508

FILE: ITEMACC

ITEMS:

ITEM	RECORD	ACCTY	USER
DATATYP	RC-DATATYP	508	MANAGER
BITS	RC-DATATYP	508	MANAGER
MIN	RC-DATATYP	508	MANAGER
MAX	RC-DATATYP	508	MANAGER
CHECK	RC-DATATYP	508	MANAGER
FIX	RC-DATATYP	508	MANAGER
INEX	RC-DATATYP	508	MANAGER
EXIN	RC-DATATYP	508	MANAGER
TYPEFUN	RC-TYPEFUN	508	MANAGER
DATATYP	RC-TYPEFUN	508	MANAGER
ITEM	RC-ITEM	508	MANAGER
DATATYP	RC-ITEM	508	MANAGER
OCCUR1	RC-ITEM	508	MANAGER
OCCUR2	RC-ITEM	508	MANAGER
OCCUR3	RC-ITEM	508	MANAGER
DEFAULT	RC-ITEM	508	MANAGER
UNIT	RC-ITEM	508	MANAGER
RECORD	RC-RECORD	508	MANAGER
RECORD	RC-RECFMT	508	MANAGER
ITEM	RC-RECFMT	508	MANAGER
POSITION	RC-RECFMT	508	MANAGER
FORMAT	RC-RECFMT	508	MANAGER
FILE	RC-FILE	508	MANAGER
FILETYP	RC-FILE	508	MANAGER
RECORD	RC-FILE	508	MANAGER
GROUP	RC-GROUP	508	MANAGER
FILE	RC-GROUP	508	MANAGER
RECORD	RC-GRPFMT	508	MANAGER
GROUP	RC-GRPFMT	508	MANAGER
CONTROL	RC-GRPFMT	508	MANAGER
POSITION	RC-GRPFMT	508	MANAGER
PREDICATE	RC-SELRULE	508	MANAGER
FUNCTION	RC-SELRULE	508	MANAGER
FTHRGRP	RC-SELRULE	508	MANAGER
SONGRP	RC-SELRULE	508	MANAGER
OWN_FILE	RLTNSHP	508	MANAGER
MBR_FILE	RLTNSHP	508	MANAGER
ITEM	RLTNSHP	508	MANAGER
KEY	RC-KEY	508	MANAGER
FILE	RC-KEY	508	MANAGER
TYPE	RC-KEY	508	MANAGER
UNIQUE	RC-KEY	508	MANAGER
ITEM	RC-KEYITEM	508	MANAGER
KEY	RC-KEYITEM	508	MANAGER
POSITION	RC-KEYITEM	508	MANAGER
FILE	RC-PGMACC	508	MANAGER
PROGRAM	RC-PGMACC	508	MANAGER
ACCTYPE	RC-PGMACC	508	MANAGER
FILETYP	RC-ACCFUN	508	MANAGER
ACCTYPE	RC-ACCFUN	508	MANAGER

FILE: ITEMACC

ITEMS:

ITEM	RECORD	ACCTY	USER
FUNCTION	RC-ACCFUN	508	MANAGER
FILETYP	RC-FILETYP	508	MANAGER
DEV_CLASS	RC-FILETYP	508	MANAGER
ITEM	RC-ITEMACC	508	MANAGER
RECORD	RC-ITEMACC	508	MANAGER
ACCTYPE	RC-ITEMACC	508	MANAGER
USER	RC-ITEMACC	508	MANAGER
PROGRAM	RC-PROGRAM	508	MANAGER
APPLICA	RC-PROGRAM	508	MANAGER
APPLICA	RC-APPLICA	508	MANAGER
ACCTYPE	RC-ACCTYPE	508	MANAGER
FILE	RC-FILEACC	508	MANAGER
USER	RC-FILEACC	508	MANAGER
ACCTYPE	RC-FILEACC	508	MANAGER
USER	RC-USERAPP	508	MANAGER
APPLICA	RC-USERAPP	508	MANAGER
USER	RC-USERCLS	508	MANAGER
PASSWORD	RC-USERCLS	508	MANAGER

FILE: FILEACC

ITEMS:

FILE	USER	ACCTY
DATATYP	MANAGER	508
TYPEFUN	MANAGER	508
ITEM	MANAGER	508
RECFMT	MANAGER	508
RECORD	MANAGER	508
FILE	MANAGER	508
GROUP	MANAGER	508
GRPFMT	MANAGER	508
SELRULE	MANAGER	508
RLTNSHP	MANAGER	508
KEY	MANAGER	508
KEYITEM	MANAGER	508
PGMACC	MANAGER	508
ACCFUN	MANAGER	508
FILETYP	MANAGER	508
ITEMACC	MANAGER	508
PROGRAM	MANAGER	508
APPLICA	MANAGER	508
ACCTYPE	MANAGER	508
FILEACC	MANAGER	508
USERAPP	MANAGER	508
USERCLS	MANAGER	508

The following files have trivial contents at this time, being either empty or having only one entry:

TYPEFUN
GROUP
GRPFMT
SELRULE
PROGRAM
APPLICA
USERCLS
KEY
KEYITEM
PGMACC
USERAPP