DOCUMENT RESUME

ED 083 815 EM 011 536

AUTHOR	Bigelow, Richard H.
TITLE	REL - English Bulk Data Input.
INSTITUTION	California Inst. of Tech., Pasadena.
SPONS AGENCY	National Science Foundation, Washington, D.C.; Office of Naval Research, Washington, D.C.; Rome Air Development Center, Griffiss AFB, N.Y.
REPORT NO	CIT-REL-R-9
PUB DATE	Jul 73
NOTE	40p.; See also EM 011 530 through EM 011 535
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	Computers; Computer Science; Computer Storage Devices; Data; *Data Bases; Data Processing; *Electrcnic Data Processing; Information Processing; Information Storage; *Input Output; *Input Output Devices; Program Descriptions
IDENTIFIERS	Bulk Data Input Processor; Rapidly Extensible Language; REL; REL English Versions

ABSTRACT

A bulk data input processor which is available for the Rapidly Extensible Language (REL) English versions is described. In REL English versions, statements that declare names of data items and their interrelationships normally are lines from a terminal or cards in a batch input stream. These statements provide a convenient means of declaring some names and stating some facts, and they are especially useful in the interactive mode. However, these statement formats are not convenient for inputting large data bases which are usually available on tapes or cards and which are generally formatted in terms of fixed fields. The bulk data input processor reads such formatted files and puts data into the version. Topics discussed in this report include: the input file format, the logical record, the physical record, field, class, relation, table, order of cards, treatment of blanks, error handling and variable record processing. Two appendixes provide a summary of descriptor cards and a series of examples illustrating how the system operates. (Author/LB)

a project report on REL

Co-principal investigators:
Bozena Henisz Dostert
Frederick B. Thompson

California Institute of Technology
Pasadena, California, 91109

REL - English
Bulk Data Input

Richard H. Bigelow

REL Report No. 9
July 1973

California Institute of Technology
Pasadena, California

The research reported here was supported in part by the grants:

National Science Foundation grant #GH-31573
Rome Air Development Center contract #F30602-72-C-0249
Office of Naval Research contract #N00014-67-A-0094-0024

Table of Contents

Bulk Data Input

In REL English version, statements that declare names of data items and their interrelationships normally are lines from a terminal or cards in a batch input stream. These statements provide a convenient means of declaring some names and stating some facts, and they are especially useful in the interactive mode. However, these statement formats are not convenient for inputting a large data base. Such data bases usually are available on tapes or cards which are formatted in terms of fixed fields. A bulk data input processor is available in English versions that will read such formatted files and put the data into the version. The user must describe the format of the data records with descriptor cards, which precede the data.

Input file format

Both the descriptor cards and the data records must be placed in an alternate input file. In the interactive system this is done by reading the following cards:

```
$ALTINPUT   altname
      descriptor cards
      data cards

$ENDALT
```

altname is a 1-8 character name for the file. In the batch system the same cards may be included in the input stream. The descriptor and/or data cards may also be in a data set on disk (or tape) which is concatenated into the input stream. The data set's logical records must be less than or equal to 255 bytes long (i. e. the LRECL of the DCB must be ≤ 255).

In either system the alternate input file must have been defined before it is used, and it will be deleted at the end of the system's run. The file is used by the statement:

INPUT BULK DATA FROM altname

where altname is the name of the file.

The descriptor cards will be read. If no errors are detected on them, the data records will be read and the information will go into the data base. The descriptor cards will be printed, and all error messages and data records in error will be printed and also displayed at the terminal.

The descriptor cards define the format of the data records. They tell where the data fields are and how they are to be processed. Defaults can be specified for blank fields, coded values can be utilized, and numeric values can be scaled by a constant factor. Error messages can be issued if needed.

The data records are divided into logical records. Each logical record has the same structure, which is defined by the descriptors. The logical records occur sequentially. Each logical record is a sequence of physical records. A physical reccrd is a card or record in a data set. The descriptors give the order and formats of the physical records. The physical records are divided into fields which are defined by the user as certain columns on the card. These fields contain the items which go into the data base. (Note: An OS logical record in a data set on disk is a physical record as the term is used here.)

A physical record format can be repeated on several cards or skipped. Thus, not all logical records need have the same identical structure. But those physical records which do occur must be in the order specified by the descriptors.

The descriptor cards can be continued. This is done with a 0-2-8 punch. The 0-2-8 is deleted and the statement continues on the first column of the next card. The total length of the statement, including all characters on the last card, can not exceed 400.

The statements have the form:

descriptor code, such as LRD, beginning in column 1.

possibly one or two numbers to identify the physical record, field,
    or table defined by this descriptor. These numbers are
    separated by blanks.

a list of parameters separated by semi-colons. Each parameter is
    a keyword followed by an equals sign followed by a list of
    options separated by commas. No blanks can occur within a
    parameter list except within a literal. The order of the
    parameters is usually immaterial.

A literal is any string of characters enclosed in single quotes. Literals are used to represent names, numbers, and times in the descriptors. The format of the literal must conform to that required by its usage. A name literal must start with a letter or digit. A number literal must be a numeric constant, and a time literal must be a constant date or the words NOW or TODAY. A quote is represented in a literal by two quotes. Literals are also used to represent logical record separators and logical and physical record identifiers. No restrictions exist on literals of these types; indeed, they may be all blank.

Name literals will be defined if they are not already defined. They will be given the type (e. g. NAME(ANIMATE)) of the field or table with which they are associated. The type can be specified for name literals in class and relation descriptors.

The descriptor statements follow. Required parameters are preceded by a *. Default options are underlined. Text enclosed in [ ] is optional and need not be given. Text enclosed in { } represents a choice; one of the enclosed forms must be given. Capital letters must appear as shown; small letter names identify types of text, such as numbers. Literals have been defined before; they are represented as 'literal' in this description. An integer is an integer constant; a number is any numeric constant.

Logical record

This card defines the boundaries of a logical record and the location of the physical record identifier.

code: LRD

parameters:

$$* \text{ SEP} = \begin{cases} \text{LRID , field} \\ \text{'literal' [, field]} \\ \text{NONE} \end{cases}$$

$$\text{PRIDFLD} = \left[\begin{array}{l} \text{field} \\ \text{NONE} \end{array}\right.$$

field is <integer, integer>

Field defines a field on a physical record. The first integer is the number of the first column of the field, and the second integer is the number of the last.

SEP describes the logical record separator, which defines the boundaries between logical records. There are three means of doing this. The first, specified by LRID, field, requires that each data card contain a field with a logical record id. The logical record consists of all cards with the same id in this field. These cards must be consecutive. The second format, 'literal' [, field], means that a card with the given literal in the field is a separator between logical records. The field begins in column one and is as long as the literal if no field is specified. If one is given, the literal is extended with blanks on the right if necessary. A separator card is otherwise ignored; it can have no data. The form SEP=NONE is discussed later.

Every data card may have a field with a literal that identifies the type of physical record that it is. PRIDFLD=field defines a default field to contain this id. It can be overridden by the physical record descriptors. If no PRIDFLD is given, then each physical record descriptor must define this field.

Physical record ids and logical record ids and separators are not necessary. They are desirable for error checking, and physical ids are required if a physical record may be omitted or repeated. If SEP=NONE is given, then there is no logical id or separator; if PRIDFLD=NONE is given, then there are no physical ids. If there are no physical ids, each physical record descriptor applies to exactly one physical record in the logical record. The logical record boundaries will be checked if SEP is not NONE. If physical ids are given but there is no logical id or separator, then each logical record is assumed to start with a physical record whose id matches that of the first physical record descriptor. Hence, the first physical record can not be omitted.

Only one LRD can be given, and it <u>must</u> be the first descriptor card.

Examples:

```
LRD    SEP=LRID, <73, 80>
LRD    PRIDFLD=<73, 80>;SEP=' ', <1, 80>
LRD    SEP=NONE;PRIDFLD=NONE
```

In the first example each data record will have a logical id in columns 73-80; those with the same id constitute a logical record. The physical record descriptors must give the locations of the physical ids. The second example defines an all-blank card as a logical separator; the physical ids are in columns 73-80 unless a PRD overrides this. Example three says that there are no logical or physical ids; each physical record type appears once in a logical record.

<u>Physical record</u>

This card defines the physical id and tells whether the record can be omitted or repeated.

code: PRD

record number: integer (must be less than 256)

parameters:

PRID = 'literal'[, field]

VAR = $\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\}$

MISS= $\left\{\begin{array}{l}\text{SKPPR[, ERROR]}\\ \underline{\text{SKPLR[, ERROR]}}\end{array}\right\}$

The record number must be unique within the logical record. It is used to identify this physical record in class or relation descriptors.

PRID gives the literal that will identify this type of physical record in the data. The field can be used to specify the position of the id; it overrides the position given by PRIDFLD in the LRD.

VAR specifies whether the physical record is variable. Normally it is not, so it can occur only once in the logical record in the data. However, if VAR=YES is given, many data cards can have the format of this physical record. They must all be consecutive.

MISS tells what to do if this physical record is missing in the data. SKPPR means that this physical record is skipped; SKPLR means that the logical record is skipped. ERROR indicates that the data card and an error message should be printed; otherwise, no indication of the action taken is given. These options have the same meaning in other descriptors.

If PRIDFLD =NONE was given on the LRD then no parameters can be given on the PRD except the physical record number. Otherwise, PRID is required.

Examples:

```
PRD   1    PRID='A';VAR=YES
PRD   21   MISS=SKPPR;PRID='R2', <1, 4>
PRD   3
```

Example 1 defines record number 1 to have id A; the field must be given in the LRD. These records can be repeated, but if none are given the logical record is skipped. Example 2 defines record number 21 with an id of R2 followed by two blanks in columns one through four. If this record is missing no error message is given and the next record is processed. Example 3 is valid only if PRIDFLD=NONE was specified; it is then required.

Field

This descriptor defines a field and tells how to obtain its value under various conditions, such as an all-blank field. Translation of coded fields and scaling by a numeric constant can be specified.

code: FD

record number: integer

field number: integer (must be less than 256)

parameters:

$$\text{*TYPE=}\left\{\begin{array}{l}\text{N}\\ \text{NA}\\ \text{C}\\ \text{CA}\\ \text{NUR}\\ \text{R}\\ \text{RA}\\ \text{NUM}\\ \text{T}\end{array}\right\}$$

*FLD=field

$$\text{BLANK=}\left\{\begin{array}{l}\text{skip-option}\\ \text{'literal'}\end{array}\right\}$$

$$\text{TRANS=}\quad\begin{array}{l}\underline{\text{SELF}}\\ \text{TABLE, integer}\end{array}\left[\;,\;\left\{\begin{array}{l}\text{SELF}\\ \text{skip-option}\\ \text{'literal'}\end{array}\right\}\right]$$

MULT= 'number'

DEF=defparm[, defparm]

$$\text{defparm is } \left\{\begin{matrix}\text{NO}\\ \text{YES}\end{matrix}\right\} \;,\; \left\{\begin{matrix}\text{ADD}\\ \text{skip-option}\end{matrix}\right\}$$

$$\text{skip-option is } \left\{\begin{matrix}\text{SKPFLD[, ERROR]}\\ \text{SKPPR[, ERROR]}\\ \text{SKPLR[, ERROR]}\end{matrix}\right\}$$

The field descriptors for a physical record follow the PRD. The physical record number must equal that of the last PRD. The field number must be unique in this physical record.

Each FD defines a field on the card for this physical record. FLD tells where the field is, and TYPE tells the kind of item in the field. The meanings of the codes are:

N	name	NUR	number relation
NA	name (animate)	R	relation
C	class	RA	relation (animate)
CA	class (animate)	NUM	number
T	time		

The skip-options are for errors. They indicate that this field, physical record, or logical record should be skipped. All class and relation descriptors that refer to a skipped field will not be processed. ERROR requests printing of the data card and a message.

BLANK tells what to do with a blank field. A skip-option can be given, or a default literal can be specified. The default for BLANK is SKPFLD, ERROR.

TRANS allows translation of a field by a table. The default of SELF means that no such translation is done. TABLE specified translation; the integer is the table number and must be less than 256.

The optional part of the parameter tells what to do if the field value is not found in the table. The value may be used as it is, a skip-option can be exercised, or a default literal value can be given. If nothing is given, SKPFLD, ERROR is assumed.

MULT is used to specify a scale factor for numeric fields. The value of the field is multiplied by the given number, which may be negative. This is done for values obtained from BLANK or TRANS processing as well.

The DEF parameter specifies what to do if a name is or isn't defined already. NO gives the action if the name is not defined and YES if it is. The action can be to add (define) the name or a skip-option. If the name already exists and is added, it will become ambiguous, and the new definition only will be used for the value of this field. In any other circumstance an ambiguous name is an error. A name is considered to exist <u>only</u> if it has the same type as for this field; names with different types are distinct. The default for DEF is NO, ADD, so undefined names are defined. Even if a YES defparm is given, NO, ADD remains in effect unless a NO defparm also appears. The DEF parameter does not apply to names obtained from translate tables or literals, but only to those on the data records.

TYPE must be given before a TRANS, BLANK, or MULT parameter.

Examples:

```
FD    1    5    FLD=<2, 6>;TYPE=N;TRANS=TABLE, 4, SELF;
                DEF=YES, SKPFLD, ERROR

FD    1    6    FLD=<21, 21>;TYPE=NUM;TRANS=TABLE, 1;
                MULT='10';BLANK='0'
```

Example 1 defines field 5 of physical record 1. It is a name in columns 2 through 6. The field is translated according to table 4; if the field value is not in the table it will be used as it is. In that case, it must not be defined yet. It will be defined. Example 2 defines a single-column numeric field in column 21. Its value is translated by table 1; the field is skipped if the value is not found. The output of the table is multiplied by 10. A blank field gives a value of 0.

Class

The next two descriptors tell how to process the data, i. e. what interrelationships exist between data items.

code: CD

parameters:

*MEMBER= $\begin{bmatrix} N \\ \overline{N}A \ , \end{bmatrix}$ value

*CLASS= $\begin{bmatrix} C \\ \overline{C}A \ , \end{bmatrix}$ value

TIME=time-option

Relation

code: RD

parameters:

$$* \text{ ARGUMENT} = \left[\begin{matrix} \underline{\text{N}} \\ \text{NA} \end{matrix} \quad , \right] \quad \text{value}$$

$$* \text{ VALUE} = \left[\begin{matrix} \underline{\text{N}} \\ \text{NA} \\ \text{NUM} \end{matrix} \quad , \right] \quad \text{value}$$

$$* \text{ RELATION} = \left[\begin{matrix} \underline{\text{R}} \\ \text{RA} \\ \text{NUR} \end{matrix} \quad , \right] \quad \text{value}$$

TIME=time-option

$$\text{value is} \quad \left\{ \begin{matrix} \text{<integer, integer>} \\ \text{'literal'} \end{matrix} \right\}$$

$$\text{time-option is} \quad \left\{ \begin{matrix} \underline{\text{ALLTIME}} \\ \left[\begin{matrix} \text{FROM} \\ \text{TO} \\ \text{BEFORE} \\ \text{AFTER} \\ \text{STARTING} \\ \text{STOPPING} \end{matrix} \right] \text{[,]value} \left[\text{[,]} \left\{ \begin{matrix} \text{FROM} \\ \text{TO} \\ \text{BEFORE} \\ \text{AFTER} \\ \text{STARTING} \\ \text{STOPPING} \end{matrix} \right\} \text{[,]value} \right] \\ \left\{ \begin{matrix} \text{IN} \\ \text{ON} \end{matrix} \right\} \text{[,] value} \end{matrix} \right\}$$

A value of the form <integer, integer> refers to a field. The first integer is the physical record number, and the second is the field number. A type given before a field reference will be checked against the field's type; they must be identical.

If a value is a literal, it will have the type specified if one is given . If not, it will have the default type. Note that for a VALUE parameter that is a numeric literal, the default type is still N (name) even though the relation is numeric.

ALLTIME is all of time. If only one of the prepositions FROM, TO, BEFORE, AFTER, STARTING, and STOPPING is used the other end of the time interval is all of the past or future; it is not today.

These descriptors tell what data is actually to be added. The class descriptor makes the MEMBER a member of the CLASS. A record descriptor makes VALUE a RELATION of ARGUMENT. The times may be applied to either.

Examples:

```
CD   MEMBER=<1, 5>;CLASS='SCHOOL'
CD   CLASS=CA, 'MALE';MEMBER=<2, 5>;TIME=BEFORE'NOW'
RD   ARGUMENT=NA, 'SUE';VALUE=NUM, <1, 6>;
       RELATION=NUR, 'AGE';TIME=IN, <1, 7>
RD   ARGUMENT=<1, 5>;VALUE='BILL';
       RELATION=<1, 8>;TIME=FROM<1, 9>TO<1, 10>
```

Example 1 makes the name in field 5 of record 1 a member of SCHOOL at all times. Example 2 makes field 5 of record 2 a member of the animate class MALE at all times before the date the input data is processed. In the third example, the animate name SUE and the number relation AGE are declared; SUE's AGE is the number in field 6 of record 1 at the time given in field 7 of record 1. Finally, BILL is the value of the relation in field 8 applied to the name in field 5 from the time given in field 9 through that in field 10.

Table

This descriptor defines a code and its value for a translation table.

code: TD

table number: integer (must be less than 256)

parameters:

* ARGUMENT='literal'

* IMAGE='literal'

This descriptor defines one entry in a TRANS table. The argument is any string of characters. The IMAGE literal will be given the type of the fields that refer to this table.

It will be more efficient if all descriptors for a table are consecutive, but this is not required.

Examples:

```
TD   4   ARGUMENT='AA';IMAGE='JOHN'
TD   6   ARGUMENT='+';IMAGE='1'
TD   6   ARGUMENT='-';IMAGE='-1'
```

In example 1, the argument AA will be translated to JOHN. In table 6, the field + will become 1, whereas - will be -1.

End

code: END

This indicates the end of the descriptors and the beginning of the data.

Order of cards

The LRD must be first. It is followed by the PRDs and FDs. The FDs for a given physical record must follow the PRD for that record. The FDs are optional; if omitted, the physical record is effectively skipped.

The CDs, RDs, and TDs must follow all the PRDs and FDs They can be in any order and are optional. TDs are needed only to define the tables referenced in the FDs. TDs for other tables are ignored. If there are no CDs or RDs the fields will be processed and names defined as required, but no data is inserted.

END is the last descriptor card. The data cards follow it.

Within a logical record the data cards must be in the same order as the PRDs.

Treatment of blanks

Within literals and fields on data cards, blanks are treated as they are in REL English. That is, leading and trailing blanks are deleted, and internal blank strings are compressed to single blanks. Thus, placement within a field is immaterial. However, blanks are not deleted from logical separators or record ids, and these must appear in the data exactly as they are in the descriptor.

Error handling

If any error is detected in the descriptor cards, the data cards are ignored. However, the names in literals in the descriptors have been defined if they were not already.

An ambiguous name is an error, unless this field makes the name ambiguous by DEF=YES, ADD. An error in a name, number, or time field causes the data field to be skipped.

If any field referred to in a CD or RD is skipped, the entire CD/RD is skipped for this physical record.

If a physical or logical record is skipped, names that have already been processed may be defined, depending on the DEF parameters. Hence, care should be taken to avoid defining these names again, and making them ambiguous. Names that occur later in the skipped record will not be defined. If a logical record is skipped, no CDs or RDs are processed for this logical record. If a physical record is skipped, all fields in this physical record will be skipped, and so CDs or RDs that refer to these fields will not be processed. This applies only to the data that appear on the card in error if a physical record is variable (VAR=YES). That is, data in the same fields on other cards of this physical record type will be processed.

Variable record processing

A physical record type may occur a variable number of times in a logical record if VAR=YES is given. A CD/RD may refer to fields of such records, and the action of the CD/RD will be repeated for each data card of the record. If more than one field is in a variable record, then the variable fields are processed in parallel. That is, all of them are stepped for each repetition. These fields may be in different records. Those which are repeated (i. e. actually occur more than once) must all be repeated the same number of times.

Example

```
LRD   PRIDFLD=<1, 1>;SEP=NONE
PRD  2   PRID='T';VAR=YES
FD  2   1   FLD=<2, 5>;TYPE=T
PRD  1   PRID='M';VAR=YES
FD  1   2   FLD=<2, 10>;TYPE=NA
PRD  4   PRID='C';VAR=YES
FD  4   1   FLD=<2, 10>;TYPE=CA;TRANS=TABLE, 1, SELF
CD   CLASS=<4, 1>;MEMBER=<1, 2>;TIME=IN<2, 1>
TD  1   ARGUMENT='M';IMAGE='MALE'
TD  1   ARGUMENT='F';IMAGE='FEMALE'
END
T1960
M JOHN
M MARY
C M
C F
```

Since both the M and C cards are repeated, they must be repeated the same number of times. The effect is to make JOHN a MALE in 1960 and MARY a FEMALE in 1960. If

T1970

also appeared after the

T1960

then John would be a MALE in 1960 and MARY would be a FEMALE in 1970.

Appendix A

Summary of Descriptor Cards

Parameters preceded by * are required.

Logical Record Descriptor:

code: LRD

parameters:

$$\text{* SEP} = \left\{ \begin{array}{l} \text{LRID, field} \\ \text{'literal' [, field]} \\ \text{NONE} \end{array} \right.$$

$$\text{PRIDFLD} = \left\{ \begin{array}{l} \text{field} \\ \text{NONE} \end{array} \right.$$

field: <integer, integer>

Physical Record Descriptor:

code: PRD

record number: integer (<256)

parameters:

PRID = 'literal'[, field]

$$\text{VAR} = \left\{ \begin{array}{l} \text{YES} \\ \underline{\text{NO}} \end{array} \right.$$

$$\text{MISS} = \left\{ \begin{array}{l} \text{SKPPR [, ERROR]} \\ \underline{\text{SKPLR [, ERROR]}} \end{array} \right.$$

Field Descriptor:

code: FD

record number: integer

field number: integer (<256)

parameters:

```
    * TYPE  =  ⎧ N
               ⎪ NA
               ⎪ C
               ⎪ CA
               ⎨ NUR
               ⎪ R
               ⎪ RA
               ⎪ NUM
               ⎩ T

    * FLD =  field

    BLANK  =  ⎧ skip-option
              ⎩ 'literal'

    TRANS =   SELF                 ⎡     ⎧ SELF        ⎤
              TABLE, integer       ⎢  ,  ⎨ skip-option ⎥
                                   ⎣     ⎩ 'literal'   ⎦

    MULT =   'number'

    DEF =   defparm [ , defparm]
```

defparm:

$$\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\} \;,\; \left\{\begin{matrix}\underline{\text{ADD}}\\ \text{skip-option}\end{matrix}\right\}$$

skip-option:

$$\begin{matrix}\underline{\text{SKPFLD}}\ [\ ,\underline{\text{ERROR}}]\\ \text{SKPPR}\ [\ ,\text{ERROR}]\\ \text{SKPLR}\ [\ ,\text{ERROR}]\end{matrix}$$

<u>Class Descriptor:</u>

code: CD

parameters:

$$\text{* MEMBER} = \left[\begin{matrix}\underline{\text{N}}\\ \text{NA}\end{matrix}\ ,\right]\quad \text{value}$$

```
* CLASS = ┌ C    ┐       value
          │ C̄A  ,│
          └      ┘
TIME = time-option
```

<u>Relation Descriptor:</u>

```
code:      RD

parameters:

    * ARGUMENT  = ┌ N    ┐    value
                  │ N̄A  ,│
                  └      ┘
    * VALUE  = ┌ N        ,┐   value
               │ N̄A        │
               └ NUM       ┘

    * RELATION  =     R       ,       value
                     R̄A
                     NUR

    TIME  =  time-option

value:     ⎧<integer, integer>
           ⎩'literal'

time-option:

      ⎧ ALLTIME                                                                     ⎫
      ⎪                                                                             ⎪
      ⎪ ⎧FROM     ⎫     [ , ] value  ┌ [ , ]  ⎧FROM     ⎫ [ =]value ┐                ⎪
      ⎪ ⎪TO       ⎪                  │        ⎪TO       ⎪           │                ⎪
      ⎨ ⎨BEFORE   ⎬                  │        ⎨BEFORE   ⎬           │                ⎬
      ⎪ ⎪AFTER    ⎪                  │        ⎪AFTER    ⎪           │                ⎪
      ⎪ ⎪STARTING ⎪                  │        ⎪STARTING ⎪           │                ⎪
      ⎪ ⎩STOPPING ⎭                  └        ⎩STOPPING ⎭           ┘                ⎪
      ⎪                                                                             ⎪
      ⎩ ⎧IN⎫ [ , ] value                                                            ⎭
        ⎩ON⎭
```

<u>Table Descriptor</u>:

code: TD

table number: integer (<256)

parameters:

* ARGUMENT = 'literal'

* IMAGE = 'literal'

<u>End Description</u>

code: END

Appendix B

EXAMPLE #1

The data consists of a deck of several hundred cards, each with the following simple format:

Columns 1 through 15: a person's name

Columns 16 through 30: the name of the school they are attending.

Columns 31 and 32: the person's age.

For example:

```
MARY A. JONES        YALE                        21
```

It is desired to enter into the data base the information corresponding to the following three sentences:

<name> is a student.

<name>'s school is <school>.

<name>'s age is <age>.

For the example shown, this would be:

Mary A. Jones is a student.

Mary A. Jones' school is Yale.

Mary A. Jones' age is 21.

If the person named or the school named are not already in the lexicon, they should now be automatically added as the data goes in.

This deck constitutes a single logical record. Each individual card is a separate physical record. Since this is the case, we don't need to worry about an identifying field, a d thus we do not have to specify any physical record id field. However, we must indicate this fact. We must, however, specify how logical records are to be distinguished. In this example none is needed, so we specify none. The Logical Record Discriptor Card is thus:

```
LRD    PRIDFLD=NONE;SEP=NONE
```

Now for the Physical Record Descriptor Card. Since there is only one card per physical record, we will give it the record number 1. There is no need in this simple case to designate any parameters. Thus the descriptor card should be

```
PRD    1
```

Each card has three fields which we must now describe. The first Field Descriptor Card will be:

```
FD    1    1    TYPE=NA;FLD=<1, 15>
```

This indicates that the field in columns 1 through 15 is to be treated as a name of an animate individual. Since no other parameters are included, the following actions will be taken. If this field is blank, this field and any data using this field will be skipped and an error message printed. If the name occuring in this field is not already in the lexicon, it will be added; this action is equivalent to a statement: <name>: =name animate.

The other two fields are similarly described by Field Description cards:

```
FD   1   2   TYPE=N;FLD=<16, 30>
FD   1   3   TYPE=NUM;FLD=<31, 32>
```

Now we must describe how to process the data using Class Descriptor Cards and Relation Descriptor Cards. The first item of data is that the person named in the first field is to be made a member of the class of students:

```
CD   MEMBER=<1, 1>;CLASS=CA, 'STUDENT'
```

The <1, 1> on this card indicates the member name is to be found on physical record 1, first field. N te that this physical record number and field number are on the first Field Descriptor Card.

To add the data:

<name>'s school is <school>.

use the Relation Descriptor Card

```
RD   ARGUMENT=<1, 1>;VALUE=<1, 2>;RELATION='SCHOOL'
```

To add the data:

<name>'s age is <age>.

```
RD   ARGUMENT=<1, 1>;VALUE=<1, 3>;RELATION=NUR, 'age'
```

On the first of these, we did not show the type of the literal "school," since the default type is relation. In the second, we needed to indicate that "age" is a number relation.

To indicate that all descriptor cards have now been included, an END card is used.

To recapitulate, the Descriptor Cards for controlling the adding of data from the given deck are:

```
LRD   PRIDFLD=NONE;SEP=NONE
PRD    1
FD    1   1    TYPE=NA;FLD=<1, 15>
FD    1   2    TYPE=N;FLD=<16, 30>
FD    1   3    TYPE=NU;FLD=<31, 32>
CD    MEMBER=<1, 1>;CLASS=CA, 'STUDENT'
RD    ARGUMENT=<1, 1>;VALUE=<1, 2>;RELATION='SCHOOL'
RD    ARGUMENT=<1, 1>;VALUE=<1, 3>RELATION=NUR, 'AGE'
END
```

EXAMPLE #2

The data consists of a deck of cards, each successive three cards of which refer to a different individual. Such a set of three cards therefore constitutes a logical record referring to this individual. Their formats are:

Card A:

Column 1 to 30	the individual's name
Column 31 to 50	date of birth
Column 52	sex, coded M for male, F for female
Column 54 to 72	occupation group, eg. "teacher"

Card B:

Column 1 to 30	the individual's name
Column 31 to 60	the name of the person's spouse
Column 62 to 65	the year of their marriage
Column 67 to 68	the number of their children

Card C:

Column 1 to 30	the individual's name
Column 31 to 50	the date on which the data on this card was taken
Column 52	attitudes on a five point
Column 54	scale on Viet Nam,
Column 56	Watergate and Dope.

Data corresponding to the information in the following sentences is to be entered.

<name> was a $\left\{\begin{matrix}\text{male}\\ \text{female}\end{matrix}\right\}$ starting <date>.

The occupation of <name> is <occupation>.

The spouse of <name> is <spouse> since <year>.

The number of children of <name> is <number>.

The Viet Nam attitude of <name>

was $\left\{\begin{array}{ll}\text{<1>} & \text{strongly opposed}\\ \text{<2>} & \text{opposed}\\ \text{<3>} & \text{no concern}\\ \text{<4>} & \text{favorable}\\ \text{<5>} & \text{strongly favorable}\end{array}\right\}$ on <date>.

The Watergate attitude...

The Dope attitude...

For card A:

If date of birth or sex are missing, the entire logical record is to be skipped and an error message printed.

If the occupation is missing, the default occupation "unknown" is to be inserted.

For card B:

If the name of the spouse is missing, skip the card, but don't issue error message.

If the spouse is given, but date of marriage is not, skip card and issue error message.

If the number of children is missing, use the number 0.

For card C:

If the date is missing, use "July 17, 1972"

If any of the three responses are missing, use "no response."

<u>Descriptor Cards:</u>

```
LRD   SEP=LRID, <1, 30>;PRIDFLD=NONE

PRD   1

FD    1   1    TYPE=NA;FLD=<1, 30>

FD    1   2    TYPE=T;FLD=<31, 50>;BLANK=SKPLR, ERROR

FD    1   3    TYPE=C;FLD=<52, 52>;BLANK=SKPLR, ERROR;
               TRANS=TABLE, 1, SKPLR, ERROR

FD    1   4    TYPE=N;FLD=<54, 72>;BLANK='UNKNOWN'

PRD   2

FD    2   2    TYPE=NA;FLD=<31, 60>;BLANK=SKPPR

FD    2   3    TYPE=T;FLD=<62, 65>;BLANK=SKPPR, ERROR

FD    2   4    TYPE=NU;FLD=<67, 68>;BLANK='0'

PRD   3

FD    3   2    TYPE=T;FLD=<31, 50>;BLANK='JULY 17, 1972'

FD    3   3    TYPE=N;FLD=<52, 52>;BLANK='NO RESPONSE';
               TRANS=TABLE, 2

FD    3   4    TYPE=N;FLD=<54, 54>;BLANK='NO RESPONSE';
               TRANS=TABLE, 2

FD    3   5    TYPE=N;FLD=<56, 56>;BLANK='NO RESPONSE';
               TRANS=TABLE, 2

CD    MEMBER=<1, 1>;CLASS=<1, 3>;TIME=STARTING<1, 2>

RD   ARGUMENT=<1, 1>;VALUE=<1, 4>;RELATION='OCCUPATION'

RD   ARGUMENT=<1, 1>;VALUE=<2, 2>;RELATION='SPOUSE';
               TIME=STARTING<2, 3>
```

```
RD    ARGUMENT=<1, 1>;VALUE=<2, 4>;RELATION=NUR,
             'NUMBER OF CHILDREN'

RD    ARGUMENT=<1, 1>;VALUE=<3, 3>;RELATION='VIET NAM
             ATTITUDE';TIME=ON<3, 2>

RD    ARGUMENT=<1, 1>;VALUE=<3, 4>;RELATION='WATERGATE
             ATTITUDE';TIME=ON<3, 2>

RD    ARGUMENT=<1, 1>;VALUE=<3, 5>;RELATION='DOPE ATTITUDE';
             TIME=ON<3, 2>

TD    1    ARGUMENT='M';IMAGE='MALE'

TD    1    ARGUMENT='F';IMAGE='FEMALE'

TD    2    ARGUMENT='1';IMAGE='STRONGLY OPPOSED'

TD    2    ARGUMENT ='2';IMAGE='OPPOSED'

TD    2    ARGUMENT='3';IMAGE='NO CONCERN'

TD    2    ARGUMENT='4';IMAGE='FAVORABLE'

TD    2    ARGUMENT='5';IMAGE='STRONGLY FAVORABLE'

END
```

EXAMPLE #3

This example is of a much simplified bibliographic record file. It consists of a sequence of logical records, each consisting of 2 or more cards. The first card has the letter A in column 1, an identifying number in columns 2, 3, and 4, and the name of an author in columns 6 through 35. Subsequent cards have a T in column 1, the same identifying number in columns 2, 3 and 4, a date in columns 6 though 9, and a title in columns 11 through 80. There may be any number of such title cards so long as there is at least one. An example of such a logical record is the three card images:

```
A025   WALTER SCOTT
TO25   1894 IVENHOE
TO25   1906 WAVERLY
```

Descriptor cards:

```
LRD   SEP=LRID, <2, 4>;PRIDFLD=<1, 1>

PRD   1   PRID='A'

FD    1   1   TYPE=NA;FLD=<6, 35>;BLANK=SKPLR, ERROR

PRD   2   PRID='T';VAR=YES

FD    2   1   TYPE=T;FLD=<6, 9>

FD    2   2   TYPE=N;FLD=<11, 80>

RD    ARGUMENT=<2, 2>;VALUE=<1, 1>;RELATION='AUTHOR';
              TIME=IN<2, 1>

END
```

Example 4

The Fortune 500 Data File

The Fortune 500 Data File is a computer-readable version of the data published by Fortune magazine for the 500 corporations with the largest sales. The data base tape is copyrighted by Time-Life, Inc. We rented a copy of this tape and put some of the data into an REL version.

We were told that the tape contained data for 1954-1971. It had 151-byte logical records in the following format:

Bytes	S	Field name
1-2		year
3-5		sales rank
6-40		company name
41-65		address
66-74		sales
75-83		assets
84-86		assets rank
87-95	*	income
96-98		income rank
99-107	*	invested capital
108-110		invested capital rank
111-119		number of employees
120-122		no. of employees rank
123-125		income/sales - % (decimal point between 124 and 125)
126-128		income/sales rank
129-131		income/capital - % (dec. pt. between 130 and 131)
132-134		income/capital rank
135-139	*	earnings/share: current (dec. pt. between 137 and 138)
140-144	*	earnings/share: ten years ago (dec. pt. between 142 and 143)

Bytes	S	Field name
145-148	*	earnings/share: growth rate - % (dec. pt. between 146 and 147)
149-151		earnings/share: growth rate rank

Fields with a * under S were signed by an 11 punch over the low order digit if negative.

We discovered that the tape was written at 800 bpi with a block size of 3775, giving 25 logical records per block. Partial clumps of the tape in EBCDIC and hexadecimal for the years 1954 and 1965 showed that the data set was incomplete and not fully described. The data was preceded by over a hundred blank logical records and then three records containing a copy right notice. The address fields were all blank. For 1954 the fields for invested capital, income/sales, income/capital, and earnings/share were all zeroes. For 1965 these fields were valid (the first year for which this was true) except that some of the growth rates and their ranks were 0. Sales, assets, and capital were given in thousands. The company names were given with non-EBCDIC codes for some special characters, including codes not valid on 026 keypunches. The signed fields contained a 12 punch over the low order position if positive.

A program was written to select some of the data fields for the years 1962 - 1971. The fields were translated to change the invalid codes found in the data and to convert the numeric fields to normal signed numbers with decimal points. This program produced a disk file with 80-byte logical records blocked to 800-byte blocks. Each original record produced two new records with these formats:

	Bytes	Field
Record 1		
	1	literal '1'
	2-5	year: 19yy
	6-8	sales rank
	9-43	company name
	44-52	sales in thousands
	53-61	assets in thousands
	62-64	assets rank
	65-74	income in thousands
	75-77	income rank
Record 2		
	1	literal '2'
	2-11	invested capital in thousands
	12-14	capital rank
	15-23	no. of employees
	24-26	no. of employees rank
	27-33	earnings/share - current
	34-39	earnings/share - growth rate - %
	40-42	earnings/share - growth rate rank

80-byte records were used since the alternate input reader requires them, although otherwise bulk data input can process longer records.

We then decided to put only the data for years 1967 - 1971 in the system, and only in smaller amounts. This was done to break the input job up into smaller units. A program was written and run to produce five data sets, each containing the records for only one of these years. We then ran several jobs, each of which put in only one year's data.

The first five jobs constructed two classes, COMPANY and FORTUNE 500 COMPANY. Each company was in the first class at all times, but in the second only for those years when it was in the Fortune 500 list. We then ran five jobs to construct the number relations - SALES, INCOME, and RANK BY SALES. Five more jobs were run to construct the GROWTH RATE relation. SALES and INCOME were multiplied by .001 to convert them to units of millions. The card input for the first GROWTH RATE job was:

```
ENTER FORTUNE500
GROWTH RATE: =NUMBER RELATION
$ALTINPUT DATA
LRD   SEP=NONE;PRIDFLD=<1, 1>
PRD   1   PRID='1'
FD    1   1    TYPE=T;FLD=<4, 5>;TRANS=TABLE, 1
FD    1   3    TYPE=N;FLD=<9, 43>
PRD   2   PRID='2'
FD    2   6    TYPE=NUM;FLD=<34, 39>
RD    ARGUMENT=<1, 3>;VALUE=<2, 6>;RELATION=NUR,
           'GROWTH RATE'
TD    1   ARGUMENT='67';IMAGE='1967'
TD    1   ARGUMENT='68';IMAGE='1968'
TD    1   ARGUMENT='69';IMAGE='1969'
TD    1   ARGUMENT='70';IMAGE='1970'
TD    1   ARGUMENT='71';IMAGE='1971'
END
//   DD   DSN=FORTUNE. YR1, DISP=SHR
//   DD   *
$ENDALT
INPUT BULK DATA FROM DATA
EXIT
```

The second card, which declares GROWTH RATE, is not strictly necessary, but is desirable. Under certain rare circumstances a class or relation name encountered by the bulk data processor may not be defined by the current programs. The time field need not be evaluated by a translate table, but this mechanism is faster than parsing when a field can have only a limited set of values. We have found that limiting the input to about 2000 cards is necessary to avoid a bug in the current system.

These jobs required the following elapsed times, measured from job start to job end by the times printed on the operator's console:

For 3 relations	For 1 relation
3: 35 = 215	2: 15 = 135
7: 54 = 474	2: 46 = 166
12: 14 = 734	6: 18 = 378
14: 17 = 857	7: 33 = 5453
16: 50 = 1010	9: 01 = 541
54: 50 = 3290	27: 53 = 1673

The times are in minutes: seconds and seconds for each of the five jobs and the total. Each job adds 500 items to one or three relations. The times are not linear with the total number of data items. Times for other jobs may be estimated from these. These jobs were run on an IBM 370/135 with 2314 disks in a 182K partition.