BNL-22473
AMD 757
CONF-770521--1

THE DATA MODEL OF THE BNL ARCHIVE AND DISSEMINATION SYSTEM*

J. Heller
Department of Computer Science
SUNY at Stony Brook
Stony Brook, New York 11790

L. Osterer
Applied Mathematics Department
Brookhaven National Laboratory
Upton, New York 11973

February 1977

*Work performed under the auspices of the U.S. Energy Research and Development Administration.

DISCLAIMER

THE DATA MODEL OF THE BNL ARCHIVE AND DISSEMINATION SYSTEM

J. Heller

SUNY at Stony Brook

L. Osterer

Brookhaven National Laboratory

Abstract

The Data Model, i.e. the information content of the data base as it is viewed by the users, of the BNL Archive and Dissemination System is presented. The syntax of the data model is stated in BNF form and the semantic meaning is discussed. Examples of the use of the data model are given.

1. INTRODUCTION

The Brookhaven National Laboratory Archive and Dissemination (BNLAD) system[1] has been designed as a Data Base Management System[3] to deal with the logical accessing and dissemination of sequential files of machine readable data in a nonhomogeneous computer network. A key feature of the BNLAD system is a parser and compiler imbedded in all application programs[5] which can read a Data Model Description (DMD) and perform the desired accessing of files from logical information given by users at run time. In this paper we discuss the DMD of the BNLAD system called elsewhere[1,8] the Physical Logical Description (PLD). In section 2 we give the syntax of the DMD in BNF form, and discuss its semantical interpretation. In section 3 we give a set of examples taken from previously designed sequential files.

2. THE DATA MODEL DESCRIPTION

The Data Model, i.e. the conceptual view of data dealt with in the BNLAD system, is given initially by a

user who constructs an archived form of a sequential file[5] using as input the DMD and the data file. In this section we discuss the syntactical and grammatical properties of the DMD as treated by the BNLAD system.

The input DMD contains the following logical information

(i) the organization which created the file,
(ii) the title of the file,
(iii) the visual sticker label,
(iv) bibliographic reference and comments,
(v) block size
(vi) character type
and (vii) the data accessing description.

The input description is in free form character mode in logical blocks of eighty characters, where one or more blanks may separate words. There is a hierarchy represented by the DMD (figure 1) which has three logical subdivisions: (i) root information which is global to the data file being described, (ii) physical information describing the particular file, and (iii) the data accessing description for this file.

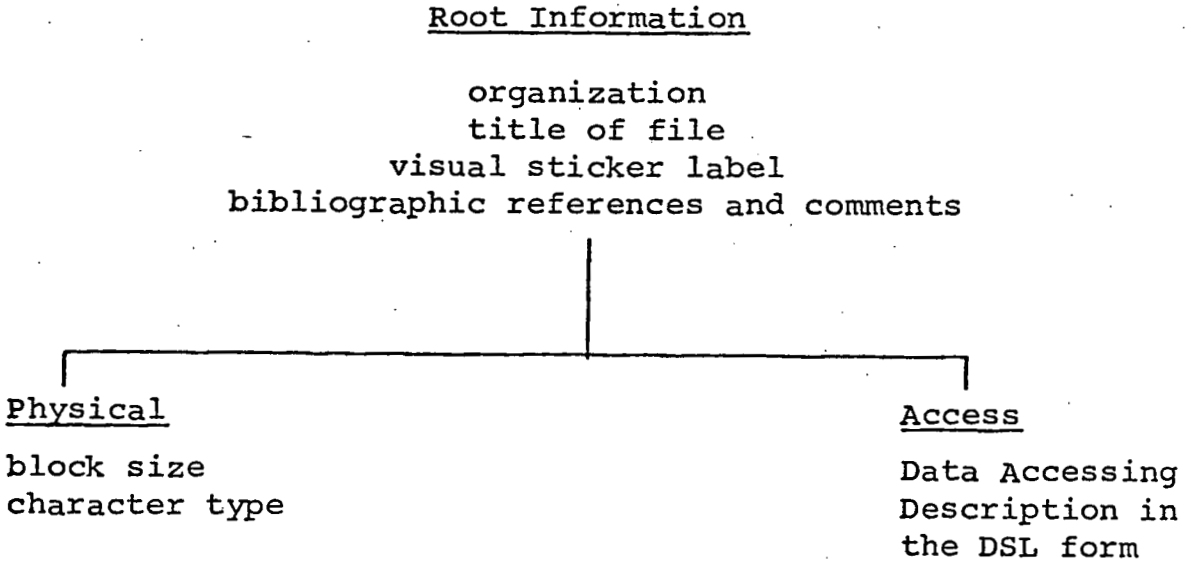

Figure 1

The Hierarchical Form of the
PLD used by the BNLADS

Since the input description is free form, it is necessary to delimit the various logical entities. There is a default set of delimiters which can be overridden if they appear in the input description. These overrides, if present, are listed before the root delimiters and indicate those delimiters that are changed.

The syntax of the DMD is

```
⟨DMD⟩::=[⟨override delimiters⟩]
    ++ROOT++ø++
    ORG=⟨organization⟩{⟨vdel⟩|ø}
    {PFN=|DD=|ASG=}⟨file title⟩{⟨vdel⟩|ø}
    VSN=⟨volume label⟩{⟨vdel⟩|ø}
    min[⟨tag⟩⟨tdel⟩⟨text⟩⟨vdel⟩]
     0

    ++PHYS++ROOT++
    BLKSIZE=⟨n⟩{⟨vdel⟩|ø}
    CHAR={A|B|E}{⟨vdel⟩|ø}

    ++BNLPLD++ROOT++⟨DSL⟩⟨rdel⟩
```

The order of the fields separated by {⟨vdel⟩|ø} can appear in any order within the node groups ++ROOT++ø++ and ++PHYS++ROOT++.

```
⟨override delimiters⟩::=[VDEL={*|⟨vdel⟩}]
                        [TDEL={ø|⟨tdel⟩}]
                        [LDEL={..|⟨ldel⟩}]
                        [FDEL={;|⟨fdel⟩}]
                        [SDEL={,|⟨sdel⟩}]
                        [RDEL={0*|⟨rdel⟩}].
```

(These can be thought of intuitively as value, tag, label, field, subfield and record delimiters respectively.) If any of the override delimiters are not present, the default values are taken and are indicated as the first character string in the choice bracket { }.

```
    ø::=any number of blanks
⟨vdel⟩::=⟨ch4⟩
⟨ldel⟩::=⟨ch4⟩
⟨fdel⟩::=⟨ch4⟩
⟨sdel⟩::=⟨ch4⟩
⟨rdel⟩::=⟨ch4⟩
 ⟨ch4⟩::=min max{⟨character⟩}
          1   4
```

```
⟨character⟩::=⟨cap⟩|⟨lower⟩|⟨digit⟩|⟨special⟩
      ⟨cap⟩::=A|B|...|Z
    ⟨lower⟩::=a|b|...|z
    ⟨digit⟩::=0|1|...|9
  ⟨special⟩::=+|-|/|*|.|,|;|:|'|"|
               !|#|$|%|&|(|)|=|_|
               @|||⟨|⟩|...
```

where the ellipses ... implies any other special character readable by the particular machine being used.

```
⟨organization⟩::=min{⟨character⟩}
                   1
 ⟨file title⟩::=⟨cap⟩min max‖⟨cap⟩|⟨digit⟩‖
                      0   7
⟨volume label⟩::=min max‖⟨cap⟩|⟨digit⟩‖
                   1   6
       ⟨tag⟩::=min{⟨character⟩}
                   1
      ⟨text⟩::=min{⟨character⟩}
                   1
        ⟨n⟩::=min max{⟨digit⟩}.
                   1   5
```

The character type of the file is given by the key word CHAR= where A implies ASCII, B implies BCD and E implies EBCDIC.

The data sub-language (DSL) is interpreted by a parser and compiler imbedded in the BNLAD system to perform the logical input and output in any application program. Its syntax is

```
⟨DSL⟩::=‖⟨do⟩|⟨get⟩|⟨labeled get⟩‖.
```

The ⟨do⟩ block's syntax is

```
⟨do⟩::=⟨label⟩⟨ldel⟩DOØ
       {⟨value⟩min[,⟨value⟩]|
                 0
       ⟨value⟩,⟨value⟩,...[⟨value⟩]|
       GETØ⟨field⟩⟨vdel⟩|
       UNTIL Ø⟨delimiter⟩|Ø}⟨vdel⟩
       ⟨DSL⟩
       END⟨label⟩⟨vdel⟩
```

where

⟨label⟩::=min$_{1}$⟨character⟩

⟨value⟩::=min$_{1}$⟨character⟩

⟨field⟩::=⟨field name⟩⟨sdel⟩
⟨format⟩
[⟨sdel⟩⟨text⟩]

⟨field name⟩::=min$_{1}$⟨character⟩}

⟨format⟩::=F⟨n⟩.⟨n⟩|F(⟨n⟩,⟨n⟩)|
I⟨n⟩|F(⟨n⟩)|
A⟨n⟩|A(⟨n⟩)|
A(⟨delimiter⟩)

⟨delimiter⟩::=min$_{1}$⟨character⟩

The ⟨value⟩, min$_{0}$⟨,⟨value⟩⟩ implies that the DO loop is executed successively for each ⟨value⟩.

The ⟨value⟩,⟨value⟩,...[⟨value⟩] form requires that the values be numeric. Then the DO loop is performed from the first ⟨value⟩ and each successive ⟨value⟩ determined by adding the increment equal to the difference of the first two values. The DO loop terminates when the ⟨value⟩ is larger than the last ⟨value⟩ if present, or when the end-file condition is raised.

The GET ⟨field⟩ form implies that the next field is numeric and specifies the number of times the DO loop is to be executed.

The UNTIL ⟨del⟩ form implies that the DO loop is to be executed until the delimiting characters of ⟨del⟩ are encountered in the input. This is an unusual feature of the DSL in that multiply imbedded do loops can be exited at any level.

The ⟨get⟩ syntax is

⟨get⟩::=GET⌀⟨field⟩
min$_{0}$⟨⟨sdel⟩⟨field⟩⟩⟨vdel⟩

All accessing implied by a GET statement accesses characters in STREAM mode.

The ⟨format⟩ has the interpretation given in PL/1, although we allow some extensions. The A(⟨delimiter⟩) format implies variable stream access until the characters of ⟨delimiter⟩ is encountered. If an application program reads a file the GET statement implies input; if an application program writes a file the GET statement implies output.

The ⟨labeled get⟩ block has the syntax

```
⟨labeled get⟩::=
   ⟨label⟩⟨ldel⟩GET⟨field⟩⟨vdel⟩
    min{⟨label⟩⟨ldel⟩IFb⟨value⟩⟨vdel⟩
     1
         ⟨DSL⟩
         ENDb⟨label⟩}
         ENDb⟨label⟩
```

A ⟨label⟩⟨ldel⟩ GET block triggers a CASE statement. The value of the field is found in one of the IF END blocks, triggers the execution of the ⟨DSL⟩ statement within that block. If no value matches an IF⟨value⟩, then the IF block with a null ⟨value⟩ is executed. If there is no null ⟨value⟩ and no match takes place, the error condition is raised and the program is terminated.

The present implementation allows any ⟨DMD⟩≤10000 characters in length. All delimiters can be surrounded by any number of blanks; all BNLAD system key words must be surrounded by at least one blank.

3. EXAMPLES OF THE DMD

In this section we give a set of examples which illustrate the syntax and semantics of the DMD of the BNLAD system to show how the system will be used to describe existing data.

Fixed Field Records

The first example we consider is typical of many sequential files which contain a single format of fixed

fields: some meteorology tapes[9] archived at BNL are described by the DMD

```
VDEL=== RDEL=0==
++ROOT** **   COMMENT CLIMATOLOGY TAPES== EDFILE DATACL==
    PFN=DATACL ==  ORG=BNL MET == VSN=N12345 ==
CONTACT TISCHLER, JOYCE BNL MET==
COMMENT APRIL 1967 - CURRENT==
REF SINGER, I. A. AND SMITH, M. E.,
 JN. METEOR. VOL. 10, NO. 2, APR.1953, P. 121-126.==
++PHYS++ROOT++  BLKSIZE= 83== CHAR= B==
++BNLPLD++ROOT++
 INPUT.. DO ==
GET  -60 IN TEMPERATURE DEG C,F4.1;
     -30 IN TEMPERATURE DEG C,F4.1;
     -3 IN TEMPERATURE DEG C,F4.1;
     SHELTER IN DEG C, F4.1;
     37' TEMPERATURE,F4.1;
     75' TEMPERATURE,F4.1;
     150' TEMPERATURE,F4.1;
     300' TEMPERATURE,F4.1;
     410' TEMPERATURE,F4.1;
     27' DIRECTION DEGREES,F3.0;
     37' SPEED,F3.1,METERS/SEC.;
     150' DIRECTION,F3.0,DEGREES;
     150' SPEED,F3.1,METERS/SEC.;
     355' DIRECTION,F3.0,DEGREES;
     355' SPEED,F3.1,METERS/SEC.;
     GUSTINESS,I2,1=A 2=B1 3=B2 4=C 5=D
     NET RADIOMETER LANGLEYS,F5.0;
     PYRHELIOMETER LANGLEYS,F6.0;
     PRECIPITATION,F4.0,INCHES;
     SKIP,A(1),BLANK NOT USED;
     TIME,I4,HOUR*10;
     YEAR-MONTH,A2,JAN=A FEB=B ... DEC=L -- 1960=A 1961=B ... ;
    REL HUMIDITY,F5.0==
    END INPUT==
     0==
..
```

The override delimiter == for ⟨vdel⟩ was necessary because the ⟨text⟩ subfield of the field TIME uses an *.

Fixed Field, Data Determined Format

There are many organizations of data where a specific field determines the format of a fixed set of fields after this field. For example, in many card format files the need to conform to eighty columns requires different formats in some portions of the card to allow for a large variety of fields. Consider a set of card formats in which the columns 1-19 are the same, column 20 determines the format of the remaining columns. The data file is made up of card images in any order. A typical DMD description is

```
LDEL=:Ø
++ROOT++Ø++ORG=NYS Historical Survey*
VSN=N31552*DATE 1972*
COMMENT Data collected on field trip
A/37-1 June, 1972*PFN=NYSHS6*
++PHYS++ROOT++CHAR=B*
BLKSIZE=80*
++BNLADS++ROOT++

        records:  DO*
                  GET recorder,A(19)*
                  cols Ø21-80:GET type,A(1)*
                  site:IFØ1*
                  GETØsite name,A(20);
                  location,A(20);
                  owner,A(10);
classifications:  DOØ1,10*
                  GETØkey,A(1);
                  END classifications*
                  END site*
      building:   IFØ2*
                  GETØname,A(15);
                      loc,A(10);
                      stories,I(2);
                      owner,A(21)*
classifications:  DOØ1,30*
                  GETØkey,A(1),
                  END classifications*
                  END building*
                other:  IF*
                  GET comment,A(60)*
                  END other*
                  ENDØcolsØ21-80*
                  END records*
```

In this example, if column 20 is neither 1 or 2, the remainder of the input card is read as A(60) and tagged comment.

Variable Unordered Fields

Many data base management systems accept free format tagged delimited character strings as input to the data base.[7,10,11] A SYSTEM 2000[7] input file could be described by the BNLAD system's DMD as follows:

```
VDEL===ßRDEL=0==ß
ORG=BNL==CONTACT FUCHEL,K.==
VSN=N31256==DATE 1977==PFN=S2K14==
Ref SRI Systems Corporation, SYSTEM2000,
Reference Manual, (1974)==
Comment See fold out example in Ref SRI==
++PHYS++ROOT++
CHAR=B==BLKSIZE=80==
++BNLPLD++ROOT++
all ß records..DO==
record..DOßUNTILßEND*==
GETßtag or value,A(*)==
END record==
ENDßallßrecords==0==
```

Since the * is used as a delimiter in SYSTEM 2000's input, we override it with == for ⟨vdel⟩ and 0== for ⟨rdel⟩. (Note: The hierarchical structure defined by the S2K DESCRIBE module is not automatically associated with this data.)

Another example of unordered variable field data used as input to a DBMS is typical of the Museum Computer Network[11] and United Nations input to the GRIPHOS DBMS. The input data are sequences of attribute value pairs of character strings. Each record is terminated by 0*, value is terminated by * and each attribute is terminated by ß.

```
VDEL===ßRDEL=0==ß
++ROOT++ß++
Ref Vance, D. Manual for Museum Computer
Network GRIPHOS Application, CCAL
Publ. SUNY at Stony Brook, N.Y. (1976).==
```

```
Ref Heller, J.  GRIPHOS Data Base
Administrator's Guide, MCN Publ. (1974)==
++PHYS++ROOT++
CHAR=E==BLKSIZE=80==
++BNLPLD++ROOT++
all⌀records..DO==
record..DO UNTIL 0*==
GET tag,A(⌀);
    value,A(*)==
END⌀record==
END⌀all⌀records==0==
```

Fixed Field Variable Format

Many cases of fixed field formatted files, in which the formats cyclically change, exist in practice. Nardi and Heller[8] give examples which we can visualize as follows:

Year	City	Month	Region	Avg. Temp.	Avg. Pressure
1978	NY	Jan		7.1	14.37
	Denver	Jan		8.2	14.92
	Miami	Jan		15.3	14.86
	NY	Feb		11.6	14.39
	Denver	Feb		12.9	14.86
	Miami	Feb		16.4	14.62
	⋮	⋮		⋮	⋮
1978	NY	Dec			
1978	Denver	Dec			
1978	Miami	Dec			

		Avg. Temp.	Avg. Pressure	Avg. Rainfall
1978	NE	18.6	14.36	105.77
⋮	SE			
	NW			
1978	SW			
1979 NY	Jan	recorded in machine readable form		
⋮				

Figure 2

The DMD for this sequential file which stores the previously unrecorded description is:

```
++ROOT++Ø++ORG=BNL*VSN=N33241*
CONTACT Nardi, J.; BNL; Upton, N.Y.*
PFN=METEX3*
++PHYS++ROOT++CHAR=B*BLKSIZE=2000*
++BNLPLD++ROOT++

Year..DOØ1978,1979,...*
month..DOØJan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec*
city..DOØN.Y.,Denver,Miami*
     GET ave temp,F(10.1);centigrade;
         ave pressure,F(10.1),#/sq.in*
     ENDØcity*
     ENDØmonth*

region..DOØNE,SE,NW,SW*
     GETØave temp,F(10.2);
         ave pressure,F(5.2);
         ave rainfall,F(5.1),cms*
     ENDØregion*
     ENDØyear*
```

Combinations of Fixed Field and Variable Field Records

Very often, data records are composed of fixed field information followed by a variable number of variable fields of data. Data conforming to the ERDA-ANSI standard for data dissemination[6] is such an example. This standard format which develops a generalization of the ANSI standard for bibliographic interchange of data[2], contains a fixed leader, a variable directory of attributes (i.e. tags) and positions, and a variable number of fields of data. In order to accept and produce data in the interchange format so that the BNLAD system is compatible, a sequential file of data of this type could be described by the DMD:

```
++ROOT++Ø++
Ref ANSI Z39.2-1971*
Ref Merrill, D. and Austin D., ERDA Interlaboratory
Working Group for Data Exchange (IWGDE),
LBL-5329, (Sept. 1976)*
```

```
Ref Heller, J., The Architecture of the BNL
Archive and Dissemination System,
BNL-AMD 752 Report (Dec. 1976).
PFN=ERDA03*VSN=N32156*
ORG=BNL AMD*
CONTACT Heller, J.; SUNY; Dept. of Comp. Sci.,
Stony Brook, N.Y. 11974; (516) 246-7146*
CONTACT Osterer, L.; BNL, AMD, Upton, N.Y. 11973;
(516) 345-4156; FTS 8-664-4156*
++PHYS++ROOT++
CHAR=A*BLKSIZE=2048*
++BNLPLD++ROOT++
all disseminated data..DO*
GET  segment control word, A(5);
     subsystem control, A(1);
     character set control, A(1);
     reserved for future use, A(4);
     field control count, A(1);
     base address of data, A(5);
     reserved for future use, A(3);
     entry map, A(4), ANSI Z39.2-1971*
directory..DO UNTIL #*
     GET tag,A(3);
         length of field,A(4);
         starting character position,A(5)*
     END directory*
padding..DO UNTIL $*
     GET fill character,A(1)*
     END padding*
user's data..DO UNTIL #*
field..DO UNTIL #*
     GET one char,A(1), user's data considered to
         be in STREAM mode character type*
     END field*
     END user's data*
END all disseminated data* 0*
```

In the above example $ and # represent the ANSI characters 1/13 and 1/14 respectively.

Hierarchical Variable Fields

Many examples of data organization require a hierarchy because there is a necessity to indicate the logical connectivity of parts of a set of data representing a user defined logical record. For example, suppose we are collecting bibliographic data about articles in a journal. Each issue will be viewed as a record composed of a variable number of article subparts. The record is viewed in a hierarchical fashion (see figure 3).

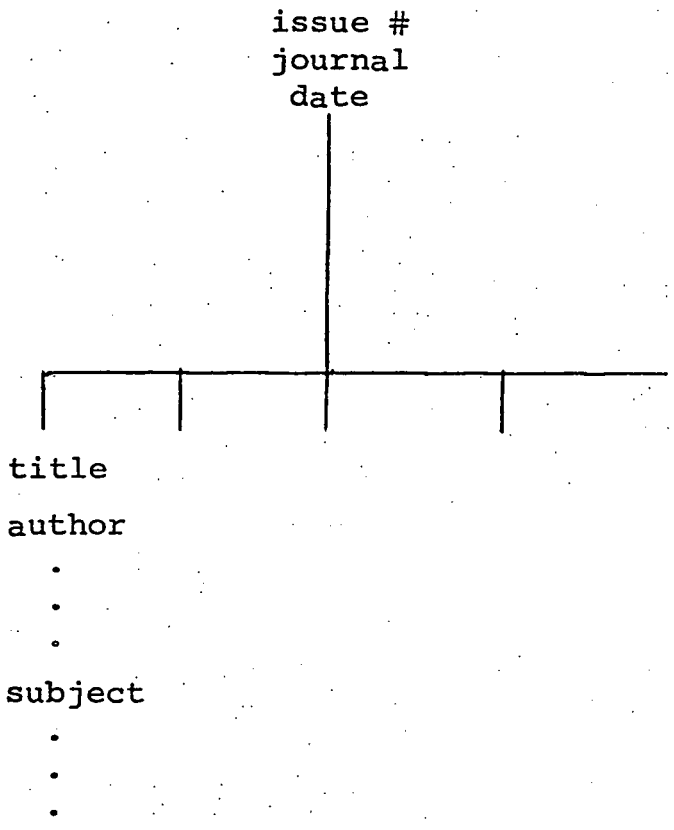

Figure 3

A hierarchical record in which each article contains one title, a variable number of authors and a variable number of subjects.

If the sequential file representing these variable strings were separated by an = and each variable set of authors and subjects were separated by 0= and each issue were separated by 00=, the data is formatted as follows:

```
18 12=CACM=December,1975=Programming
Languages, Natural Languages, and Mathematics=
Nour, Peter=0=Analogies related to social aspects=
language quality=language development=artificial
auxiliary languages=0=Exception Handling:
Issues and a Proposed Notation=Goodenough,J.B.=
0=mutilevel exit=GOTO statement=error conditions=
structured programming=0=The intrinsically
Exponential Complexity of the Circularity Problem
for Attribute Grammars=Jayayeri, M=
Ogden, W.F.=Rounds, W.C.=0=Attribute
grammars=circularity problem=context free
grammars=computational complexity=exponential
time=0=
                          .
                          .
                          .
Automatic Data Structure Choice in a Language
of Very High Level=Schwartz, J.T.=0=
program optimization=automatic programming=
high-level languages=0=00=18 10=
CACM=October, 1975=a Preliminary System
for the Design of DBTG Data Structures-
Gerritsen, R=0=network model of data bases=
Data Base Task Group=data base design=data
structure=0=
                          .
                          .
                          .
Horner's Rule for the Evaluation of a
General Closed Queueing Network=
Reiser, M=Kobayashi, H=0=Queueing
Networks=queueing theory=Horner's Rule=
service rate=0=00=
                          .
                          .
                          .
```

The DMD for a file of issues would be of the form:

```
++ROOT++Ø++
Ref CACM 1975*DD=CACM75*
ORG=SUNY Biblo. Proj.*VSN=ACM014*
CONTACT Finermann, A.; SUNY; Dept of Computer
Science; Stony Brook, N.Y. 11974*

++PHYS++ROOT++
BLKSIZE=32000*CHAR=E*

++BNLPLD++ROOT++

all issues..DO*
     GET issue #,A(=);
         journal,A(=);
         date,A(=)*
articles..DO UNTIL 0==*
     GET title,A(=)*
authors..DO UNTIL 0=*
     GET author,A(=)*
     END authors*
subjects..DO UNTIL 0=*
     GET subject,A(=)*
         END subjects*
     END articles*
     END all issues*
           0*
```

REFERENCES

1. S. Abbey, K. Fuchel, J. Heller, K. S. Lin, L. Osterer, The BNL Archive and Dissemination System, AMD 752R1 (1977).

2. ANSI Z39.2 - 1971 Standard for Bibliographic Interchange on Magnetic Tape.

3. C. J. Date, Introduction to Database Systems, Addison-Wesley Publishing Company (1975).

4. K. Fuchel, J. Heller and J. Tischler, System 2000 Tape Inventory and Description, memo (1975).

5. J. Heller, The Architecture of the BNL Archive and Dissemination System. BNL report AMD 752 (1976).

6. D. Merrill and D. Austin, ERDA Interlaboratory Working Group for Data Exchange (IWGDE), LBL-5329 (Sept. 1976).

7. MRI Systems Corporation, SYSTEM 2000, Reference Manual (1974).

8. J. Nardi, and J. Heller, A Machine Interpretable Design for Physical and Logical Description of Sequentially Archived Data, Procs. of the Berkeley Workshop on Distributed Data Management and Computer Networks, AMD 728 (May 1976).

9. J. Tischler, Note describing a set of Meteorology tapes at BNL (1976).

10. United Nations Headquarters Library Documentation Division, Computer Assisted Indexing Project, COMP/1 (May 1969).

11. D. Vance, Manual for Museum Computer Network GRIPHOS Application, CCAL Publ., SUNY at Stony Brook (1976).